Assignment IV (MA226)

Name: Sourav Bikash

Roll No.:11012338

Submission Date: 01/02/2013 Time:23:59 hrs.

Aim of the Problem:

The problem involves general sampling methods. Earlier we have introduced random number generation. Hence forth we assume the availability of random numbers through linear congruence generators. We also assume the availability of a sequence of random numbers. A simulation algorithm transforms these samples of uniform distribution to samples of other distribution. We will generate two distributions:

i) Exponential Distribution
ii) Gamma Distribution

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.
Using the uniform sample we can generate samples of other distribution using various algorithms. In this assignment we make use of two most widely accepted general techniques:

i) The inverse-transform method.
ii) The acceptance-rejection method.

The inverse-transform method:

This sets $X=F^{-1}(U)$, $U\sim Unif[0,1]$
Where F^{-1} is the inverse of F and Unif[0,1] denotes the uniform distribution on [0,1].

The acceptance-rejection method:

The acceptance rejection method, introduction by Von Neumann, is among the most widely accepted method for generating random samples. Suppose we want to generate samples from a function f(x) defined on the set S. And g(x) is the function from which we know how to generate samples. For the density function

$$f(x) \leq cg(x) \text{ for all } x \in S$$

for some constant c. here we generate a sample X from g(x) and accept the sample with probability f(x)/cg(x).

Part I:

This question wants to simulate a sample of size 5000 of exponential type with mean=5. We use the following conversion:

$$z=-5*(\log(u))$$

A sequence of x_i was generated using the linear congruence generator. Using this a sequence of U_i was obtained and corresponding z was collected. The frequencies of the obtained U_i was plotted using a barplot.

C++ implementation:

```
#include<cstdio>
#include<iostream>
#include<cmath>
using namespace std;
int main()
{
	int xo=5;
	int initial=((1597*xo)+1)%244944;
	double u=(double)initial/244944;
	int x;long count=0;
	double z;
	double sum=0;double ave;
	double max=0,min;
	x=((1597*initial)+1)%244944;
	while(count!=5000)
	{
		u=(double)x/244944;
		x=((1597*x)+1)%244944;
		z=-5*(log(u));
		if(count==0)
			min=z;
		if(z<=min)
			min=z;
		if(z>=max)
			max=z;
		sum=sum+z;
		cout<<z<<"\n";
		count++;
	}
	ave=sum/5000;
	cout<<"maximum="<<max<<"\n";
	cout<<"minimum="<<min<<"\n";
	cout<<"average="<<ave<<"\n";
	return 0;
}
```

The program generated the output which can be view in “output1.txt” and “output12.txt” enclosed.

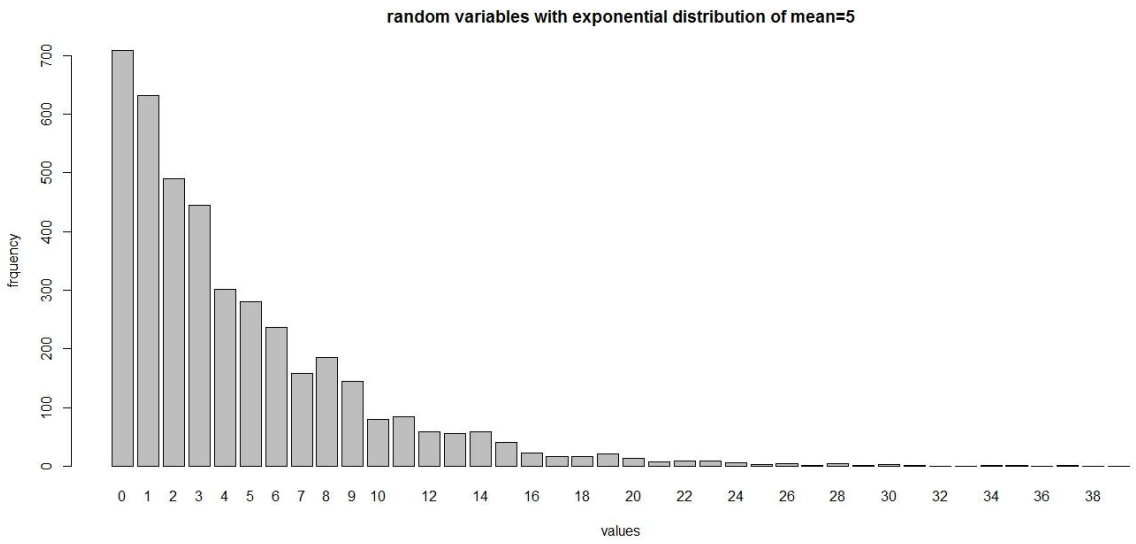
Maximum=38.4514
Minimum=0.000857412
Average=4.96654

Implementation using R:

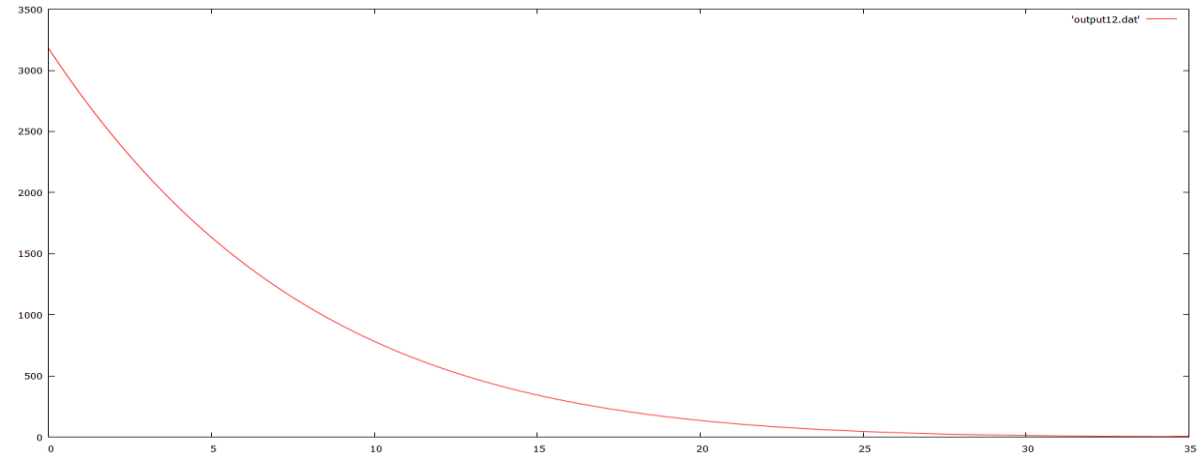
```
xo<-5
count<-1
f<-array(0,c(40))
```

```
initial<-((1597*xo+1))%% 244944
u<-initial/244944
z<-0
xaxis<-0:39
x<-initial
x<-((1597*x)+1)%% 244944
while (count<5000){
        u<-x/244944
        z<--5*log(u)
        f[floor(z)]<-f[floor(z)]+1
        print(paste(z))
        x <- ((1597*x)+1)%% 244944
        count<-count+1
}#using a while loop for the problem
print(f)
barplot(f,names.arg=xaxis,main="random variables with exponential distribution of mean=5",ylab="frquency",xlab="values")
```

Using the output bar plot were generated:



The curve was smoothen and the following was obtained:



Part II:

This part focuses on the use of a sequence of samples. Here also we generate 5000 gamma samples from a sequence of 5 exponential samples;the following conversion is used:

$$X = -\ln \prod_{i=1}^{\alpha} (U_i)$$

C++ implementation:

```
#include<iostream>
#include<cstdio>
#include<cmath>
using namespace std;
int main()
{
	int xo=5;
	int initial=((1597*xo)+1)%244944;
	int ini1=((1697*xo)+1)%244911;
	int ini2=((1797*xo)+1)%244923;
	int ini3=((1897*xo)+1)%244977;
	int ini4=((1997*xo)+1)%244887;
	double u1=(double)initial/244944;
	double u2=(double)ini1/244911;
	double u3=(double)ini2/244923;
	double u4=(double)ini3/244977;
	double u5=(double)ini4/244987;
	int x,x1,x2,x3,x4;long count=0;
	double z,z1,z2,z3,z4;
	double sum=0;double ave;
	double max=0,min;
	x=((1597*initial)+1)%244944;
	x1=((1697*ini1)+1)%244911;
	x2=((1797*ini2)+1)%244923;
	x3=((1897*ini3)+1)%244977;
	x4=((1997*ini4)+1)%244887;
	while(count!=5000)
	{
		u1=(double)x/244944;
		x=((1597*x)+1)%244944;
		u2=(double)x1/244911;
		x1=((1697*x1)+1)%244911;
		u3=(double)x2/244923;
		x2=((1797*x2)+1)%244923;
		u4=(double)x3/244977;
		x3=((1897*x3)+1)%244977;
		u5=(double)x4/244987;
		x4=((1997*x4)+1)%244887;
		z=-5*log(u5*u1*u2*u3*u4);
		if(count==0)
			min=z;
		if(z<=min)
			min=z;
		if(z>=max)
			max=z;
		sum=sum+z;
		cout<<z<<"\n";
		count++;
```

```
        }
        ave=sum/5000;
        cout<<"maximum="<<max<<"\n";
        cout<<"minimum="<<min<<"\n";
        cout<<"average="<<ave<<"\n";
        return 0;
}
```

The program generates:

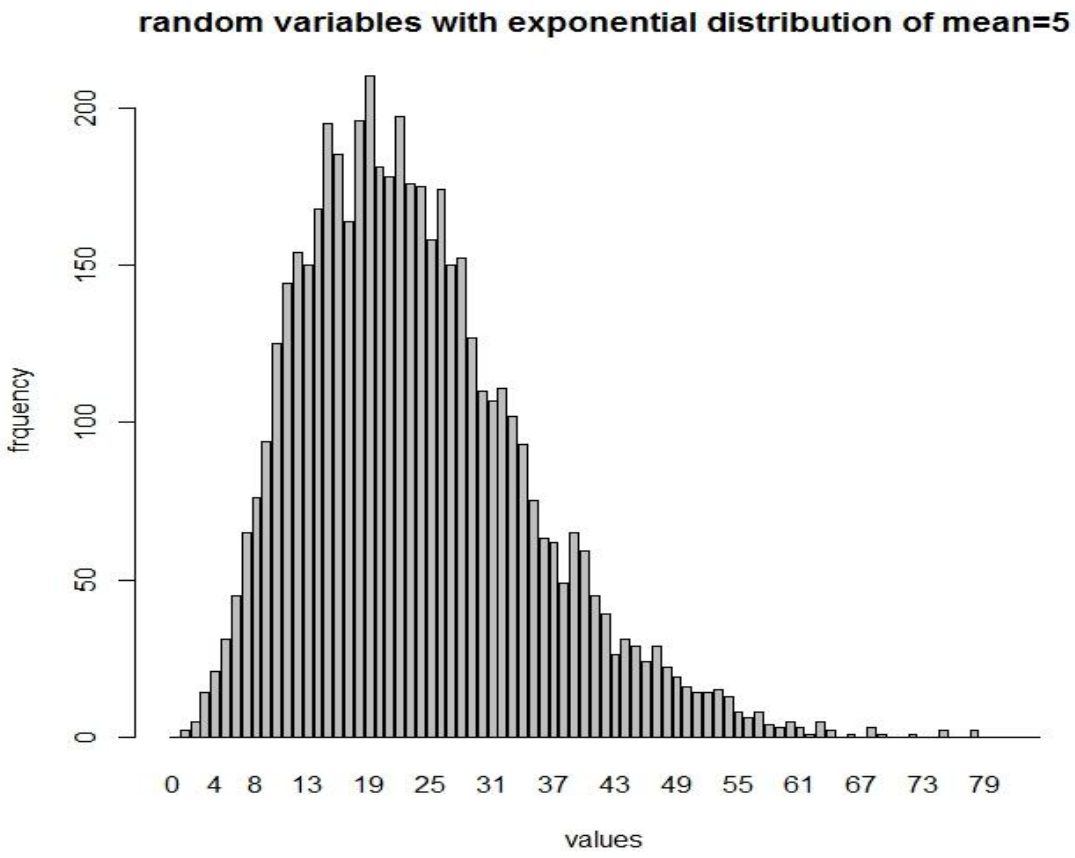
Maximum=84.1948
Minimum=1.90412
Average=25.0022

Implementation using R:

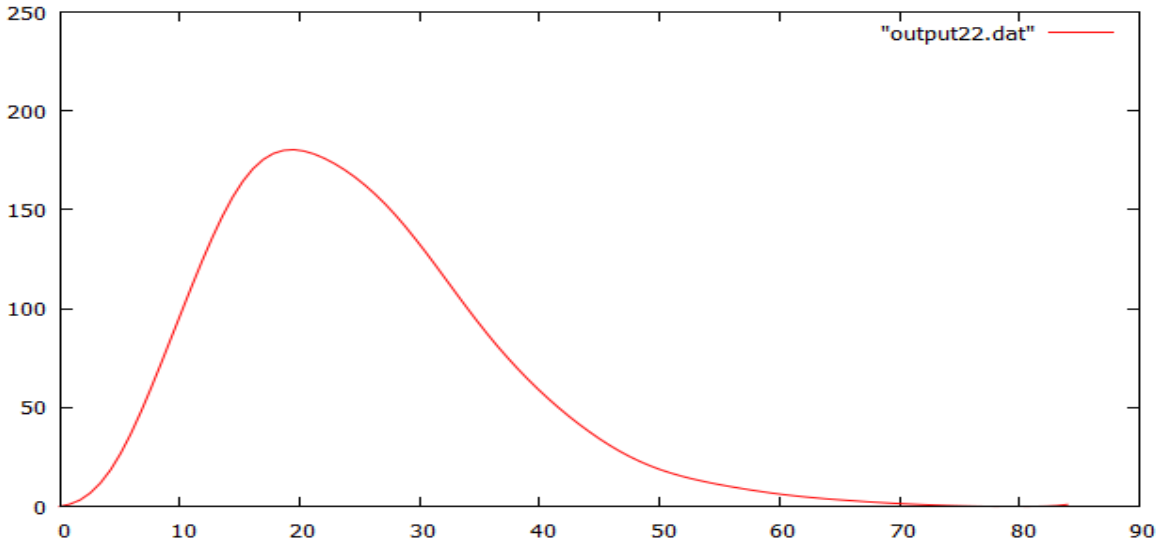
```
xo<-5
count<-1
f<-array(0,c(85))
initial<-((1597*xo+1))%% 244944
ini1<-((1697*xo)+1)%%244911
ini2<-((1797*xo)+1)%%244923
ini3<-((1897*xo)+1)%%244977
ini4<-((1997*xo)+1)%%244887
u<-initial/244944
u2=ini1/244911;
u3=ini2/244923;
u4=ini3/244977;
u5=ini4/244987;
z<-0
xaxis<-0:84
x<-initial
x1<-ini1
x2<-ini2
x3<-ini3
x4<-ini4
x<-((1597*x)+1)%% 244944
while (count<5000){
        u<-x/244944
        u2=x1/244911;
        u3=x2/244923;
        u4=x3/244977;
        u5=x4/244987;
        z<--5*log(u*u2*u3*u4*u5)
        f[floor(z)]<-f[floor(z)]+1
        print(paste(z))
        x <- ((1597*x)+1)%% 244944
        x1<-((1697*x1)+1)%%244911
        x2<-((1797*x2)+1)%%244923
        x3<-((1897*x3)+1)%%244977
        x4<-((1997*x4)+1)%%244887
        count<-count+1
```

}#using a while loop for the problem
print(f)
barplot(f,names.arg=xaxis,main="random variables with exponential distribution of mean=5",ylab="frquency",xlab="values")

The bar plots:



The smooth curve:



Part III:

This section uses the acceptance rejection method to generate from 20x(1-x)^3.

R implementation:

```
density <-function (n1) 20*n1*((1-n1)^3)
RejectionSampling <- function(n)
{
	RN <- NULL
	for(i in 1:n)
	{
		OK <- 0
		while(OK<1)
		{
			T <- runif(1,min = 0, max = 1)
			U <- runif(1,min = 0, max = 1)
			if(2.5*U <= density(T))
			{
				OK <- 1
				RN <- c(RN,T)
			}
		}
	}
	return(RN)
}
sample <-RejectionSampling(5000)
hist(sample, freq = TRUE, xlim = c(0,1), breaks = 100, main = "Rejection Sampling of
f(x)=20x(x-1)^3");
```

The Bar-plot obtained: