Assignment V (MA226)

Name: Sourav Bikash

Roll No.:11012338

Submission Date: 07/02/2013 Time:23:59 hrs.

Aim of the Problem:

The problem involves the use of given known distributions to generate other distribution of numbers.

Mathematical Analysis/Theory:

Here we make use of the acceptance-rejection method along with the inverse transform method. We will generate:

i) Normal variates using double exponential distribution.

ii) Half-standard normal distribution using exponential distribution with mean 1 by acceptance-rejection method.

The inverse-transform method:

This sets $X=F^{-1}(U)$, U~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

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

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 1000 of normal type.
We use the following conversion:

$$z=-1*(\log(u))$$

The algorithm is therefore:

1. Generate u_1, u_2 and u_3 from a $U(0; 1)$.
2. Set $x = -\log(u_1)$
3. If $u_2 > \exp(-(x - 1)^2/2)$ (*rejection*) then goto 1.
4. Else (*acceptance*) if $u_3 \cdot 0:5$ then set $x = ¡x$.
5. Return x.

Implementation using R:

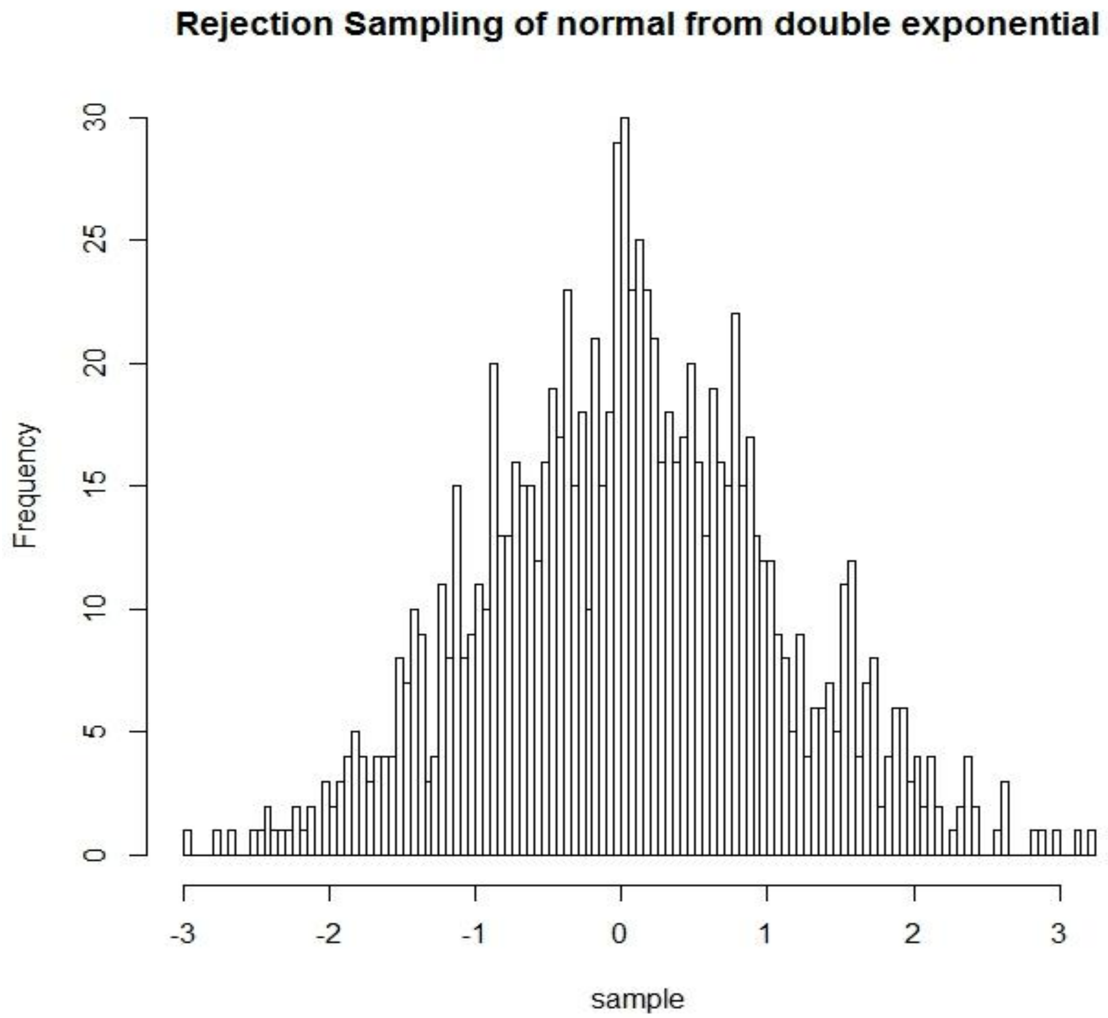
```
RejectionSampling <- function(n)
{
     RN <- NULL;
     p<- 0;
     q<- 0;
     for(i in 1:n)
     {


          ok<-0;
          while(ok<1)
          {
               U1 <- runif(1,min = 0, max = 1);
               U2 <- runif(1,min = 0, max = 1);
               U3 <- runif(1,min = 0, max = 1);
               x<- -1*log(U1);
               q<-q+1;
               if(U2< exp(-((x-1)^2)/2))
               {
                    if(U3<= 0.5)
                    {
                         x<- -1*x;
                    }
                    ok<- 1;
                    RN<- c(RN,x);
                    p<- p+1;

               }
          }
     }

     print(paste(p/q));
     return(RN);
}
sample<- RejectionSampling(100000);
hist(sample,freq = TRUE, breaks = 100, main = "Rejection
Sampling of normal from double exponential");
```

Output:-

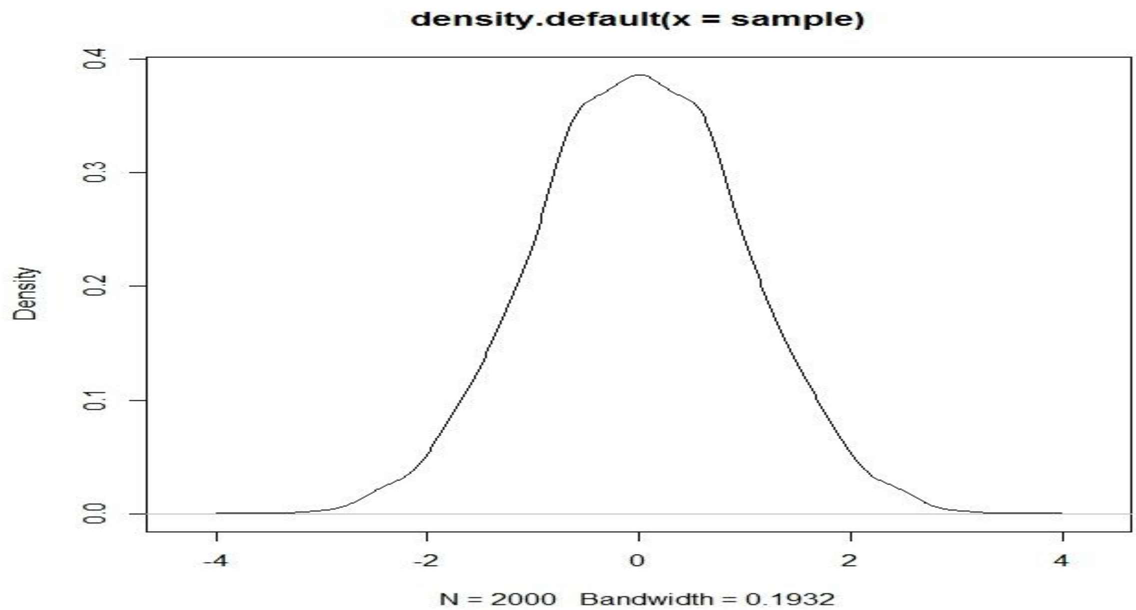
Acceptance probability=0.760456273764259

The c value:$\sqrt{(2e/\pi)}$

Using the output bar plot were generated:



The density plot of the so generated random numbers:



Conclusion:
By looking at the density plot of the random numbers so generated we can say this is the standard normal distribution curve. Thus our claim is true.

Part II:

This question wants to simulate a sample of size 1000 of standard half-normal type.
This is to be done from exponential distribution of mean one.
We use conversion factor same as before.

Therefore the algorithm is:

1. Generate $Y \sim Exp(1)$, $Y = -\log(U_1)$, $U_1 \sim U(0, 1)$.
2. Generate another U_2 _ $U(0, 1)$.
3. Test U2< exp(-((x-1)^2)/2)
   if true set X = Y.
4. Repeat if not.

Implementation using R:

```
RejectionSampling <- function(n)
{
	RN <- NULL;
	p<- 0;
	q<- 0;
	for(i in 1:n)
	{


		ok<-0;
		while(ok<1)
		{
			U1 <- runif(1,min = 0, max = 1);
			U2 <- runif(1,min = 0, max = 1);
			#U3 <- runif(1,min = 0, max = 1);
			x<- -1*log(U1);
			q<-q+1;
			if(U2< exp(-((x-1)^2)/2))
			{
				ok<- 1;
				RN<- c(RN,x);
				p<- p+1;
			}
		}
	}

	print(paste(p/q));
	return(RN);
```

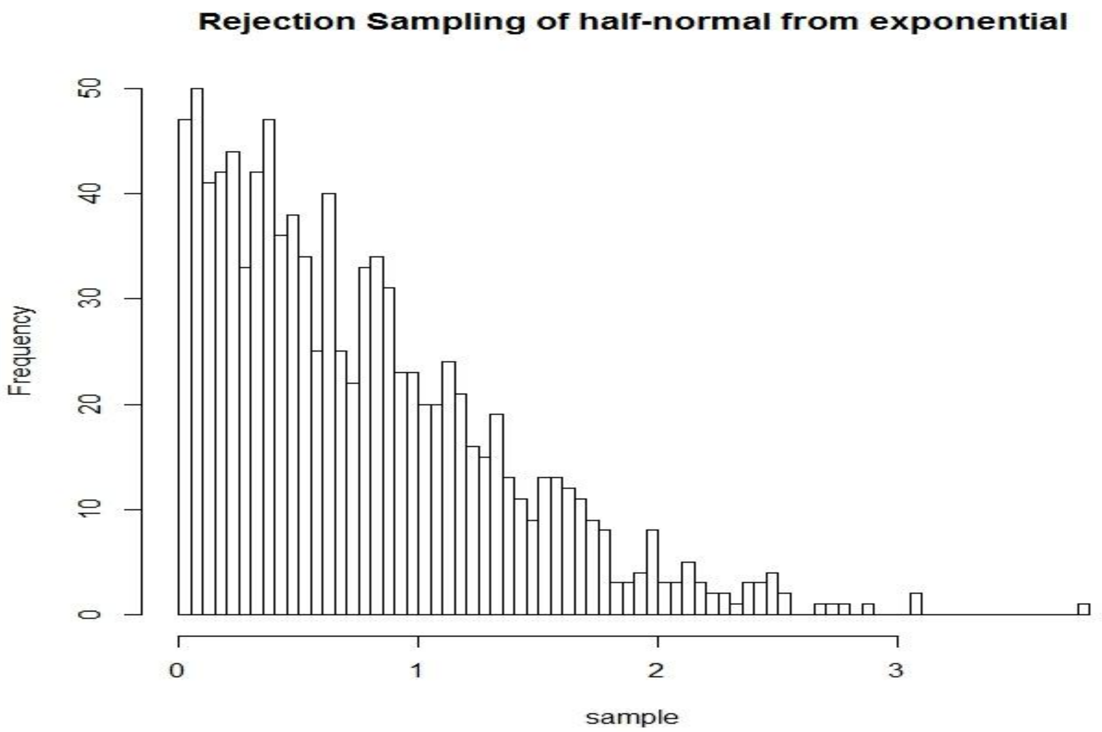
```
}
sample<- RejectionSampling(1000);
hist(sample,freq = TRUE, breaks = 100, main = "Rejection
Sampling of half-normal from exponential");
```

Output:

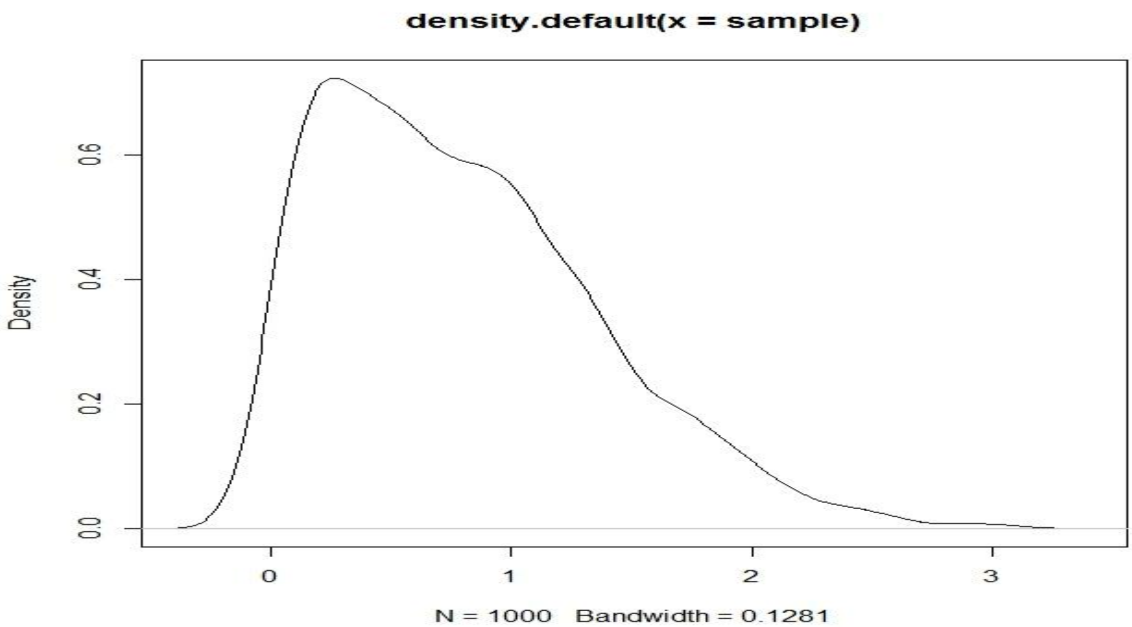
Acceptance probability=0.771604938271605

The c value:$\sqrt{(2e/\pi)}$

The bar plots:



The density plot curve:



Conclusion:
By looking at the density plot of the random numbers so generated we can say this is the standard normal distribution curve. Thus our claim is true.

Part III(a):

This problem generates 10 random numbers from the above probability mass function using usual procedure (inverse transform) of generating random number from discrete distribution defined on finite number of points.

R implementation:

```
generateRandom <- function(x)
     {
          if(x < 0.45)          {return(3);}
          else if(x < 0.70)     {return(2);}
          else if(x < 0.85)     {return(4);}
          else if(x < 0.95)     {return(5);}
          else                  {return(1);}
     }
u <- runif(10, min=0, max=1);#u contains 10 random numbers
r <- unlist(lapply(u, generateRandom));#r contains the
generated random variables X
print(r);#printing the array r
print(var(r));#calculate variance
print(mean(r));#calculate mean
```

Output obtained:
2 2 4 5 3 3 3 3 3 4
Variance=0.844444444
Mean=3.2

Part III(b):

This problem generates 10 random numbers from the above probability mass function using acceptance-rejection method of generating random number from discrete distribution defined on finite number of points.

R implementation:
```
generateRandom <- function(n)
     {
          RN <- NULL;
          f <- c(0,0,0,0,0);
          p <- c(0.05, 0.25, 0.45, 0.15, 0.10);
          for(i in 1:n)
          {
               ok <- 0;
               while(ok < 1)
               {
```

```
                    u <- runif(1, min=0, max=1);
                    y <- floor(u*5) + 1;
                    u0 <- runif(1, min=0, max=1);
                    if(u0 <= p[y]/(2.25*0.2))
                    {
                         RN <- c(RN, y);
                         ok <- 1;
                         f[y] <- f[y]+1;
                    }
               }
          }
          return(RN);
     }
r <- generateRandom(10);
print(r);
print(mean(r));
print(var(r));
plot(r, type="p");
```

Output generated:-
2 2 3 4 3 3 2 5 3 2
Mean: 2.9
Variance: 0.988889
We can see 3 has maximum probability(0.45) thus the mean is near 3.
Plot of the random numbers: