**Assignment VI (MA226)**

**Name: Sourav Bikash**

**Roll No.:11012338**

**Submission Date: 01/03/2013 Time:23:59 hrs.**

**Aim of the Problem:**

The problem involves the use of the Box-Muller Method to generate standard normal numbers. The second part involves the use of a modification of the Box-Muller Method done by Marsaglia-Bray.

**Mathematical Analysis/Theory:**

Firstly we make use of the transformation theorem:

**Theorem:**

Suppose *X* is a random variable in IR*n* with density *f*(*x*) *>* 0 on the support *S*. The transformation *h* : *S → B, S,B ⊆* IRnis assumed to be invertible and the inverse be continuously differentiable on *B*. *Y* := *h*(*X*) is the transformed random variable. Then *Y* has the density

***f*(*h−*1(*y*))=|*∂*(*x*1*, ..., xn*)/*∂*(*y*1*, ..., yn*)| *, y∈ B***

where *x* = *h−*1(*y*) and *∂*(*x*1*, ..., xn*)/*∂*(*y*1*, ..., yn*) is the determinant of the Jacobian matrix of all first-order derivatives of *h−*1(*y*).

In this assignment we make use of the transformation in **R**2 to generate normal variates.

**Method of Box-Muller:**

To apply the Theorem we start with the unit square S := [0, 1]2 and the density of the bivariate uniform distribution. The transformation is

**y1 = √(−2 log x1 )(cos 2πx2) =: h1(x1, x2)**

**y2 =√(−2 log x1 )(sin 2πx2 ) =: h2(x1, x2)**

The function h(x) is defined on [0, 1]2 with values in IR2. The inverse function

h−1 is given by

**x1 = exp{- (y21 + y22)/2}**

**x2 =(arctan( y2/y1))/2π**

**Therefore the Algorithm:**



**The Variant of Marsaglia:**

The variant of Marsaglia prepares the input in the above Algorithm such that trigonometric functions are avoided. For U ~ U[0, 1] we have V := 2U- 1 ~ U[-1, 1]. (Temporarily we misuse also the financial variable V for local

purposes.) Two values V1, V2 calculated in this way define a point in the

(V1, V2)-plane. Only points within the unit disk are accepted:

**D := {(V1, V2) : V 12 + V 22 < 1}; accept only (V1, V2) ∈ D .**

In case of rejectance both values V1, V2 must be rejected. As a result, the surviving (V1, V2) are uniformly distributed on D with density f(V1, V2) = 1/π for (V1, V2) ∈ D. A transformation from the disk D into the unit square S := [0, 1]2 is defined by

**x1=V12+V22**

**x2=(1/2π)(arg(V1,V2))**

That is the Cartesian Co-ordinates V1,V2 on D are mapped to the squared radius and the normalized angle. For illustration, see Figure. These “polar -coordinates” (*x*1*, x*2) are uniformly distributed on *S.*

With these variables the relations:

**Cos 2πx2=V1/√( V12+V22)**

**Sin 2πx2= V2/√( V12+V22)**

Hold, which means that it is no longer necessary to evaluate trigonometric relations in the Box-Muller Method.



**Figure describing the transformations of Box-Muller and Marsaglia Variant.**

**The Algorithm:**



**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 *·* 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);

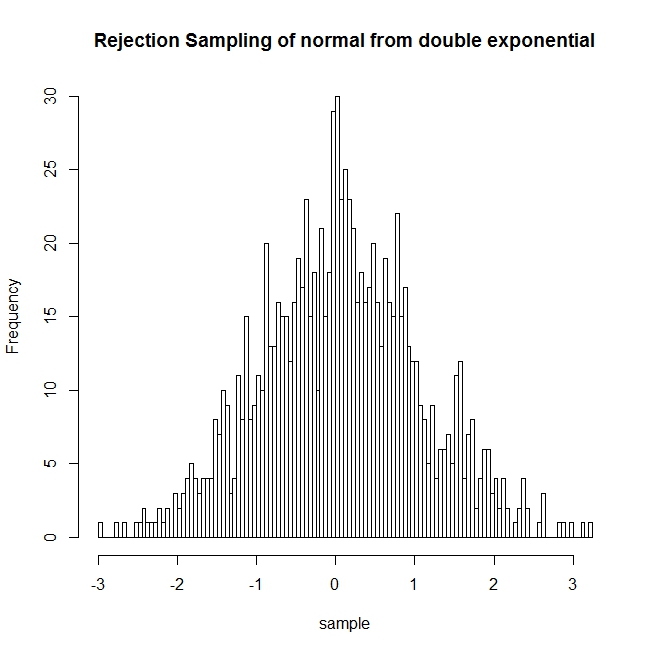
hist(sample,freq = TRUE, breaks = 100, main = "Rejection Sampling of normal from double exponential");

**Output:-**

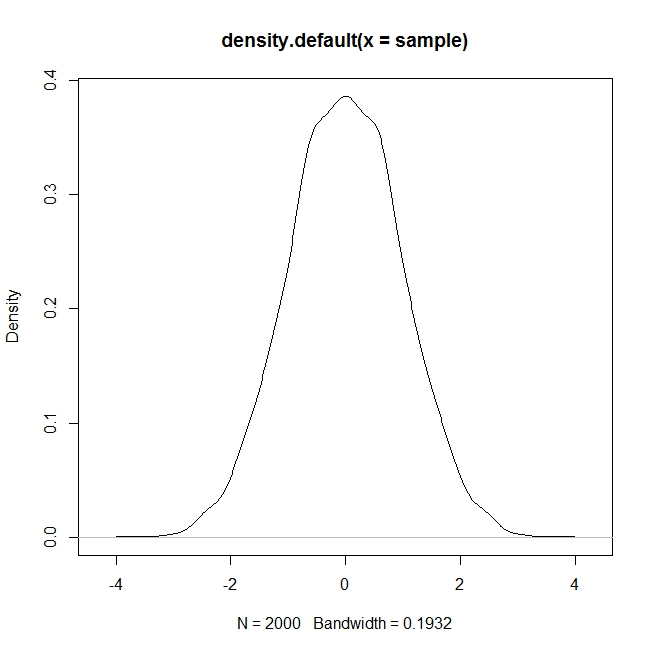
**Acceptance probability=0.760456273764259**

**The c value:√(2e/π)**

**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 ~Exp(1), Y = −log(U1), U1 ~ U(0, 1).

2. Generate another U2 \_ 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);

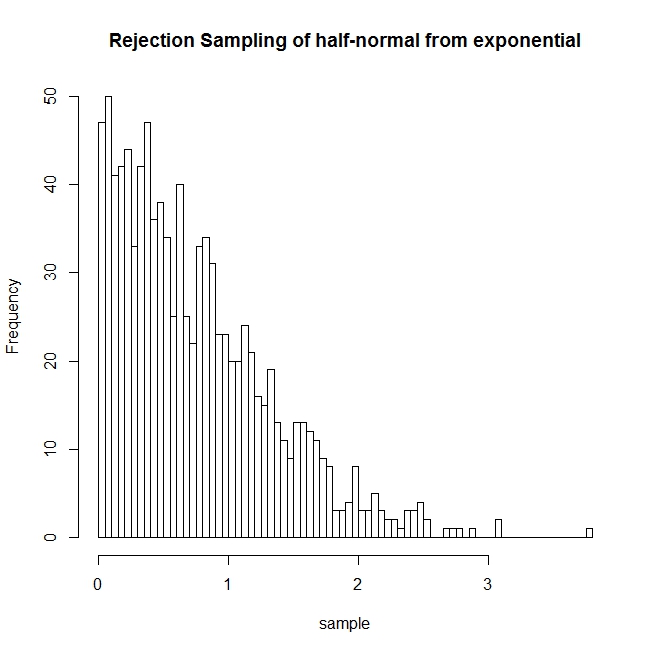
hist(sample,freq = TRUE, breaks = 100, main = "Rejection Sampling of half-normal from exponential");

**Output:**

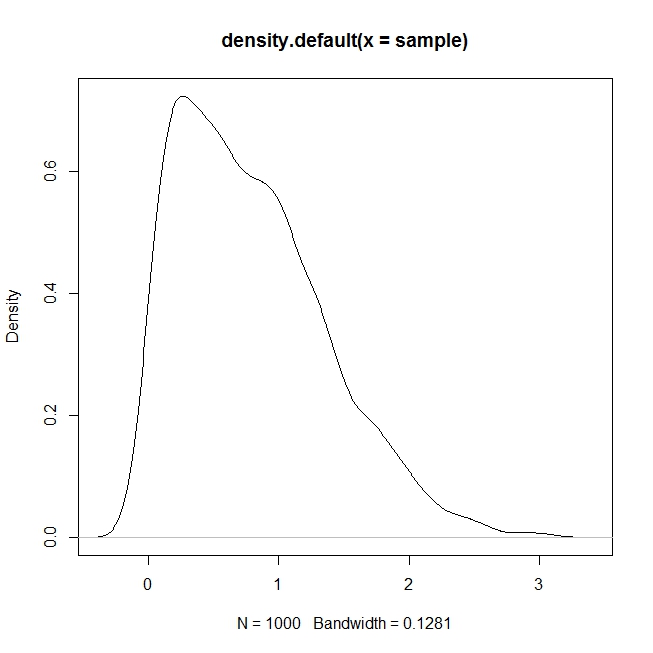
**Acceptance probability=0.771604938271605**

**The c value:√(2e/π)**

**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:-**

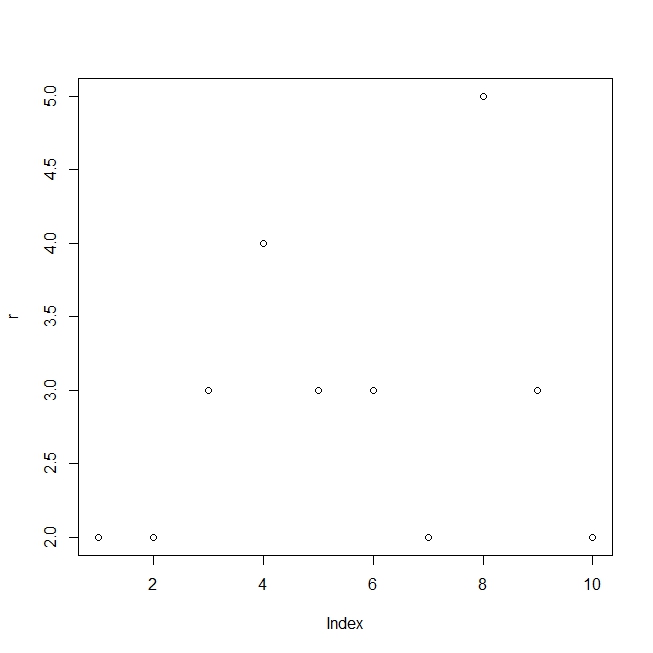
2 2 3 4 3 3 2 5 3 2

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:**

****