MONTE CARLO ENDSEM REPORT – Vaibhav Saxena (140123041)

Question 1

Algorithm for generating bivariate normal distribution (X,Y) using Cholesky's decomposition:

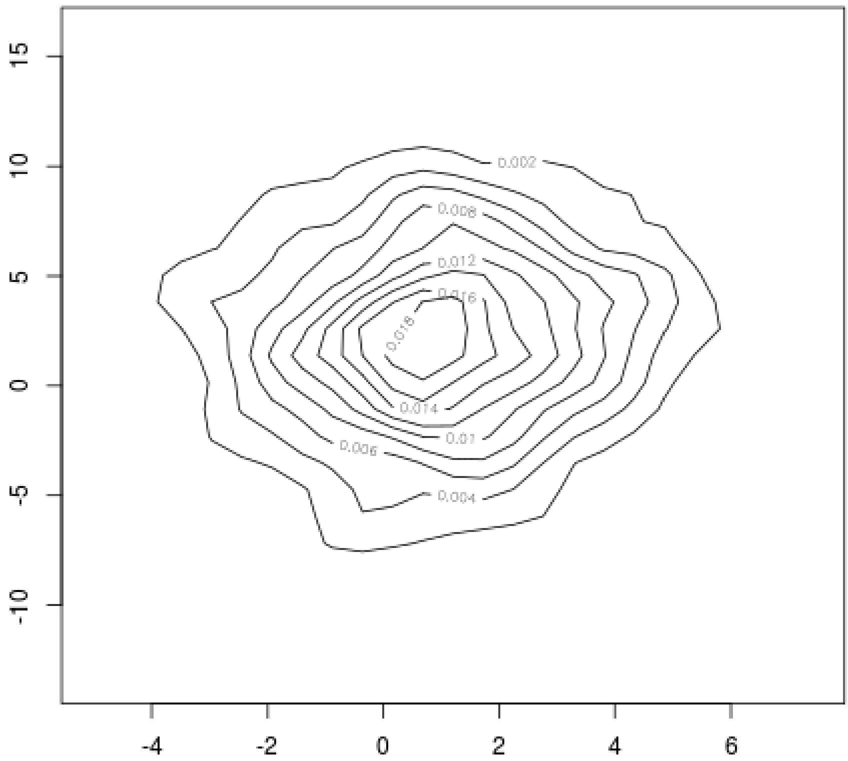
1. Generate Z1, Z2 ~ N(0,1)
2. Set X = μ1 + σ1*Z1
3. Set Y = μ2 + σ2*ρ*Z1 + σ2*√(1 - ρ^2)*Z2

Here μ1 = 1, σ1 = 2, μ2 = 2, σ2 = 4

Code for R:

```
require(MASS)

# For X
mu1 <- 1
sig1 <- 2

# For Y
mu2 <- 2
sig2 <- 4

rho <- 0.05

normvar <- c(0.9597, -1.3404, 1.2238, 0.2551)

# Using Cholesky's decomposition method

noofrand <- 1000
z1 <- rnorm(noofrand, mean=0, sd=1)
z2 <- rnorm(noofrand, mean=0, sd=1)
X <- mu1 + sig1*z1
Y <- mu2 + sig2*rho*z1 + sig2*sqrt(1 - rho^2)*z2

g <- kde2d(X,Y)

png(file="contour.png")
contour(g)
dev.off()
```

The contour graph for the bivariate distribution (X,Y) can be shown in the figure below:



<u>Question 2</u>

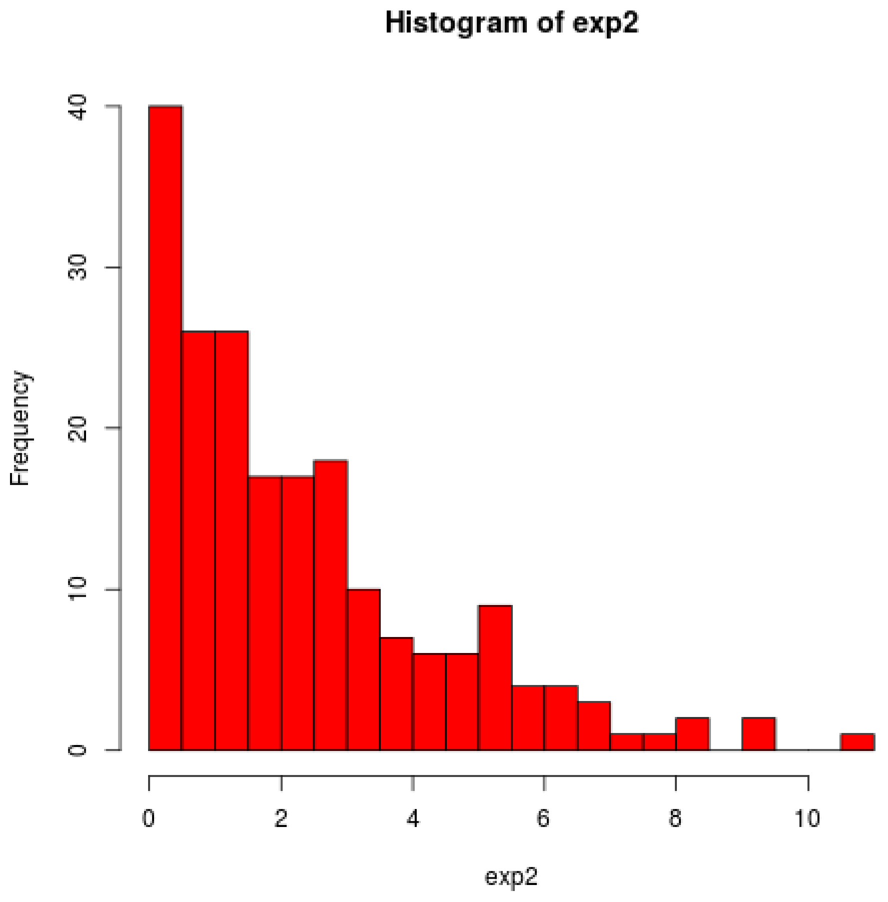
<u>Code for R:</u>

```
alpha <- 0.4
noofrand <- 200

# Exponential distribution with mean 2
exp2 <- -2*log(runif(noofrand, min=0, max=1))

# Exponential distribution with mean 3
exp3 <- -3*log(runif(noofrand, min=0, max=1))

mixed <- vector('numeric')

for(i in 1:noofrand) {
        if(runif(1,min=0,max=1) < alpha) {
                mixed <- c(mixed, exp2[i])
        }
        else {
                mixed <- c(mixed, exp3[i])
        }
}
```

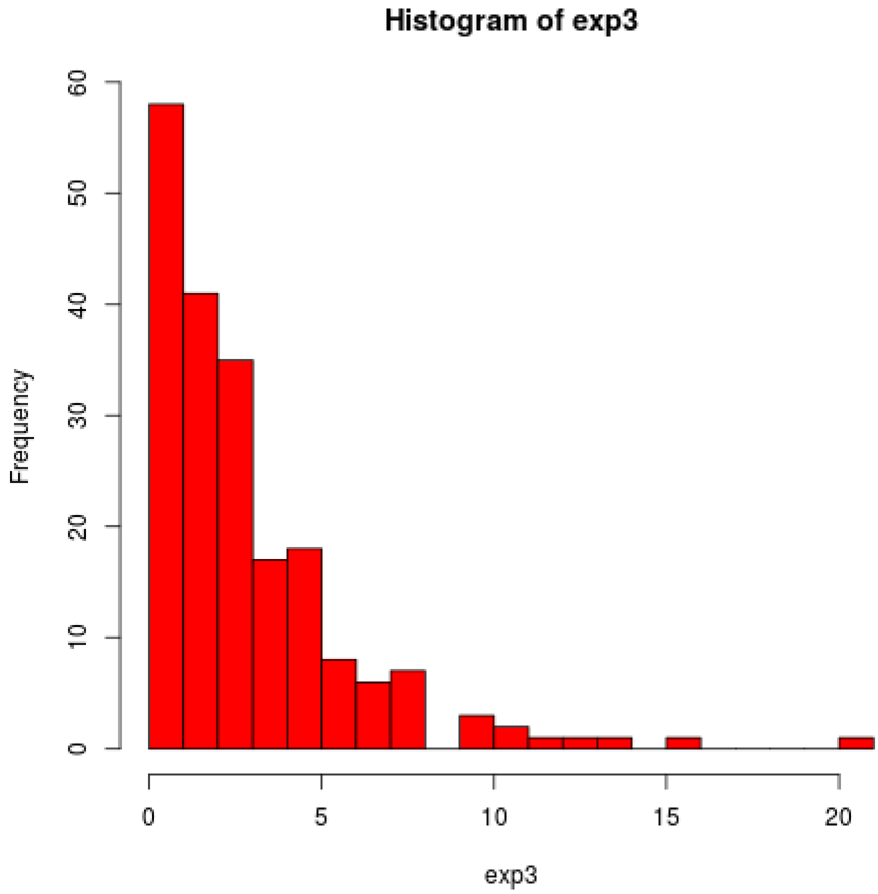
```
png(file="exponential_2")
hist(exp2, col="red", breaks=20)
dev.off()

png(file="exponential_3")
hist(exp3, col="red", breaks=20)
dev.off()

png(file="mixed_distribution")
hist(mixed, col="red", breaks=20)
dev.off()
```

The plotted histrograms are as follows:

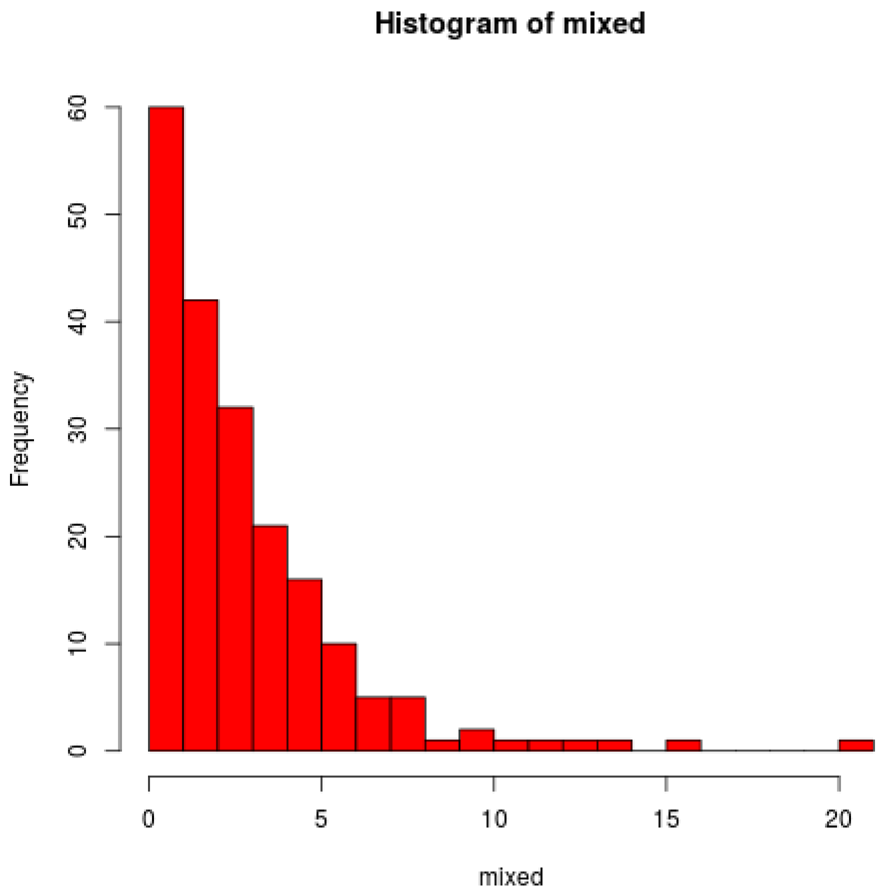
For exponential with mean = 2



For exponential with mean = 3



For mixed distribution:



<u>Question 3</u>

<u>Code for R:</u>

```
# Naive Monte Carlo method
montecarlo <- function(n, num_estimates) {
        estimates <- vector(length=num_estimates)
        for(i in 1:num_estimates) {
                u <- runif(n, min=0, max=1)
                y <- 2*exp(-4*u*u)
                estimates[i] <- mean(y)
        }
        return(c(mean(estimates), var(estimates)))
}

# Antithetic estimate method
antithetic <- function(n, num_estimates) {
        estimates <- vector(length=num_estimates)
        for(i in 1:num_estimates) {
                u <- runif(n, min=0, max=1)
                v <- 1-u
                y1 <- 2*exp(-4*u*u)
                y2 <- 2*exp(-4*v*v)
                y <- (y1+y2)/2
                estimates[i] <- mean(y)
        }
        return(c(mean(estimates), var(estimates)))
}

# Control variate method
controlvariate <- function(n, num_estimates) {
        estimates <- vector(length=num_estimates)
        for(i in 1:num_estimates) {
                u <- runif(n,min=0,max=1)
                x <- 2*exp(-4*u*u)
                y <- 2*exp(-2*runif(n,min=0,max=1))
                c <- -1*cov(x,y)/var(y)

                w <- x + c*(y - mean(y))
                estimates[i] <- mean(w)
        }
        return(c(mean(estimates), var(estimates)))
}

N <- c(100,1000,10000)
num_estimates <- 100

naive_mean <- vector(length=length(N))
naive_var <- vector(length=length(N))
anti_mean <- vector(length=length(N))
anti_var <- vector(length=length(N))
```

```
control_mean <- vector(length=length(N))
control_var <- vector(length=length(N))

for(i in 1:length(N)) {
	naive_mean[i] <- montecarlo(N[i], num_estimates)[1]
	naive_var[i] <- montecarlo(N[i], num_estimates)[2]
	anti_mean[i] <- antithetic(N[i], num_estimates)[1]
	anti_var[i] <- antithetic(N[i], num_estimates)[2]
	control_mean[i] <- controlvariate(N[i], num_estimates)[1]
	control_var[i] <- controlvariate(N[i], num_estimates)[2]
}

for(i in 1:length(N)) {
	cat("For N =", N[i],"\n")
	cat("Naive Monte Carlo: \n")
	cat("mean =",naive_mean[i],"\n")
	cat("variance =",naive_var[i],"\n")

	cat("Antithetic Estimate: \n")
	cat("mean =",anti_mean[i],"\n")
	cat("variance =",anti_var[i],"\n")
	red1 <- 100*(naive_var[i] - anti_var[i])/naive_var[i]
	cat("Variance reduction =",red1,"\n")

	cat("Control Variate: \n")
	cat("mean =",control_mean[i],"\n")
	cat("variance =",control_var[i],"\n")
	red2 <- 100*(naive_var[i] - control_var[i])/naive_var[i]
	cat("Variance reduction =",red2,"\n")
	cat("\n")
}
```

<u>Output:</u>

```
For N = 100
Naive Monte Carlo:
mean = 0.8839491
variance = 0.005563168
Antithetic Estimate:
mean = 0.8807665
variance = 8.24897e-05
Variance reduction = 98.51722
Control Variate:
mean = 0.8750775
variance = 0.004606603

For N = 1000
Naive Monte Carlo:
mean = 0.8893941
variance = 0.000378082
```

Antithetic Estimate:
mean = 0.8824278
variance = 8.552757e-06
Variance reduction = 97.73786
Control Variate:
mean = 0.8794938
variance = 0.0004289711

For N = 10000
Naive Monte Carlo:
mean = 0.8820661
variance = 4.520528e-05
Antithetic Estimate:
mean = 0.8820634
variance = 1.030306e-06
Variance reduction = 97.72083
Control Variate:
mean = 0.8820974
variance = 4.503716e-05

N = 100	Mean	Variance
Naive	0.8839491	0.005563168
Antithetic	0.8807665	8.24897e-05
Control	0.8750775	0.004606603

Variance reduction = 98.51722 %

N = 1000	Mean	Variance
Naive	0.8893941	0.000378082
Antithetic	0.8824278	8.552757e-06
Control	0.8794938	0.0004289711

Variance reduction = 97.73786 %

N = 10000	Mean	Variance
Naive	0.8820661	4.520528e-05
Antithetic	0.8820634	1.030306e-06
Control	0.8820974	4.503716e-05

Variance reduction = 97.72083 %

Question 4

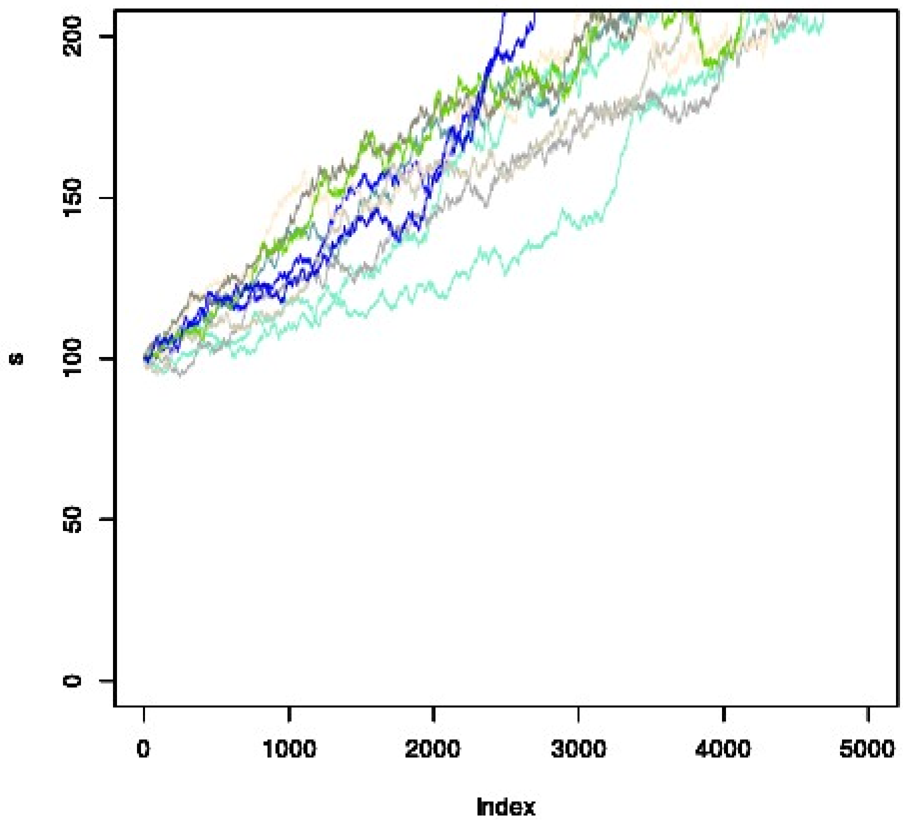
Code for R:

```
# Box-Muller to generate normal distribution
boxmuller <- function(n) {
	R <- -2*log(runif(n,min=0,max=1))
	u <- runif(n,min=0,max=1)
	u <- 2*pi*u
	x <- sqrt(R)*cos(u)
	return(x)
}

t <- seq(from=0,to=5,length.out=5000)

mu <- 0.2
sig <- 0.1

s5 <- vector(length=10)

#Generating Geometric Brownian Motion
png(file="gbm.png")
for(n in 1:10) {
	s <- vector(length=5000)
	s[1] <- 100
	for(i in 2:5000) {
		s[i] <- s[i-1]*exp((mu - 0.5*sig*sig)*(t[i]-t[i-1]) + sig*sqrt(t[i]-t[i-1])*boxmuller(1))
	}
	s5[n] <- s[5000]
	plot(s, type="l", col=colors()[as.integer(runif(1)*100)], ylim=c(100-100,100+100))
	par(new=TRUE)
}

s5meanth <- s[1]*exp(mu*5)
s5varth <- s[1]*s[1]*exp(mu*5)*exp(mu*5)*(exp(sig*sig*5)-1)

cat("Simulated E(S(5)) =", mean(s5), "\n")
cat("Simulated Var(S(5)) =", var(s5), "\n")
cat("Theoretical E(S(5)) =", s5meanth, "\n")
cat("Theoretical Var(S(5)) =", s5varth, "\n")
```

Output:

Simulated E(S(5)) = 275.367
Simulated Var(S(5)) = 2421.426
Theoretical E(S(5)) = 271.8282
Theoretical Var(S(5)) = 3788.45

The generated Geometric Brownian is plotted as below:



Observation:
The simulated and theoretical values of E(S(5)) and Var(S(5)) are in vicinity. Hence the simulated brownian motion is an acceptable approximation.