Homework 2

Sasikanth Nagalla

13 April 2016

Question 1: Bivariate Normal Random

```
lgm <- function(n,x0){
  a <- 7^5
  b <- 0
  m <- 2^31 - 1
  rand <- numeric(n)
  rand[1] <- x0
  for (i in 2:n) {
    rand[i] = (rand[i-1]*a)%%m
  }
  return (rand/m)
}

simnorm <- function(n,m){
unimatrix <- matrix(lgm(n,m),n/2,2)
Zone <- sqrt(-2*log(unimatrix[,1]))*cos(2*pi*unimatrix[,2])
Ztwo <- sqrt(-2*log(unimatrix[,1]))*sin(2*pi*unimatrix[,2])
Z <- c(Zone, Ztwo)
return (Z)
}

simmvn <- function(n,m,mu,Sigma){
rho <- Sigma[1,2]
Z <- matrix(simnorm(2*n,m),ncol=2)
X <- mu[1] + Sigma[1,1]*Z[,1]
Y <- mu[2] + rho*Sigma[2,2]*X + sqrt(1-rho^2)*Sigma[2,2]*Z[,2]
return (cbind(X,Y))
}

# Correlation
Sigma <- matrix(c(1,-0.7,-0.7,1),2,2)
mu <- c(0,0)
data <- matrix(simmvn(n = 10000, 0.001,mu, Sigma), ncol =2)
xbar <- data[,1] - mean(data[,1])
ybar <- data[,2] - mean(data[,2])
correlation <- sum(xbar*ybar)/sqrt(sum(xbar*xbar)*sum(ybar*ybar))
```

The value of ρ is: -0.7043615

Question 2: Monte Carlo Simulation

```
Sigma <- matrix(c(1,0.6,0.6,1),2,2)
mu <- c(0,0)
data<- matrix(simmvn(n = 10000,0.001, mu, Sigma),ncol=2)

func <-function(X,Y){
return (X^3 + sin(Y) + Y*X^2)
}

temp = 0
for (i in 1:nrow(data)){
temp = temp + max(0,func(data[i,1],data[i,2]))
}

E <- temp/i
```

The value of E is: 1.5439161

Question 3

3.a

```
n <- 100000
rand <- simnorm(n,2)
a1 <- 5*rand^2 + sin(rand*sqrt(5))
Ea1 <- sum(a1)/n
a2 <- exp(0.5/2) * cos(rand*sqrt(0.5))
Ea2 <- sum(a2)/n
a3 <- exp(3.2/2) * cos(rand*sqrt(3.2))
Ea3 <- sum(a3)/n
a4 <- exp(6.5/2) * cos(rand*sqrt(6.5))
Ea4 <- sum(a4)/n

var(a1)
```

```
## [1] 51.06435
```

```
var(a2)
```

```
## [1] 0.1283656
```

```
var(a3)
```

```
## [1] 11.34109
```

```
var(a4)
```

```
## [1] 332.5487
```

Ea1 : 5.0247868
Ea2 : 0.9988549
Ea3 : 0.9924087
Ea4 : 1.0283125

3.b

The values of the last three integrals are equal to one since there are independent of t and equal to 1.

3.c variance reduction technique

```
ran <- -rand
b1 <- (5*ran^2 + sin(ran*sqrt(5)) + a1)/2
Eb1 <- sum(b1)/n
b2 <- (exp(0.5/2) * cos(ran*sqrt(0.5)) + a2)/2
Eb2 <- sum(b2)/n
b3 <- (exp(3.2/2) * cos(ran*sqrt(3.2)) + a3)/2
Eb3 <- sum(b3)/n
b4 <- (exp(6.5/2) * cos(ran*sqrt(6.5)) + a4)/2
Eb4 <- sum(b4)/n

var(b1)
```

```
## [1] 50.57551
```

```
var(b2)
```

```
## [1] 0.1283656
```

```
var(b3)
```

```
## [1] 11.34109
```

```
var(b4)
```

```
## [1] 332.5487
```

Eb1 : 5.0230352
Eb2 : 0.9988549
Eb3 : 0.9924087
Eb4 : 1.0283125
I used the antithetic variates for variance reduction but the results have not improved and the variance have remained the same except in the case of Ea1. Variance reduction technique gave improved result Eb1.

Question 4

4.a

```
n <- 10000
r <- 0.04
sigma <- 0.2
K <- 100
s0 <- 88
T <- 5
rand <- simnorm(n,0.001)
price <- numeric()
for (i in 1:length(rand)){
price[i] <- exp(-r*T)*max(0,s0*(exp((r-0.5*sigma^2)*T + sigma*sqrt(T)*rand[i])) - K)
}
Ca1 <- mean(price)
var(price)
```

```
## [1] 1134.466
```

Ca1 : 18.817809
variance : 1134.4659406

4.b

```
library("OptionPricing")
```

```
## Warning: package 'OptionPricing' was built under R version 3.2.3
```

```
price2 <- numeric()
for (j in 1:length(rand)){
price2[j] <- 0.5*(price[j] + exp(-r*T)*max(0,s0*exp((r-0.5*sigma^2)*T + sigma*sqrt(T)*
(-rand[j])) - K))
}
Ca2 <- mean(price2)
var(price2)
```

```
## [1] 503.1248
```

```
blsprice <- BS_EC(5,100,0.04,0.2,88)[1]
```

The black scholes price is: 18.2837657
Ca2: 18.7450547
variance : 503.1248323
The variance reduction technique using antithetic variates has reduced the variance and improved the accuracy of the result.

Question 5

5.a

```
S0 <- 88
sigma <- 0.18
r <- 0.04
n <- 10
nsims <- 10000
S <- matrix(NA,nrow=nsims,ncol=10)
for (i in 1:n){
Z <- simnorm(nsims,0.01)
S[,i] <- S0 * exp(sigma*sqrt(i)*Z + (r-0.5*sigma^2)*i)
}
ES <- c(S0,colMeans(S))
plot(0:n,ES,ylim=c(0,300),main="Stock Price Path", xlab= "Time", ylab="Price", type
='o', col = "red", pch=10, lty = 1, lwd =2)
legend(8,300,c("E(S)"),col=("red"), lty=c(1), lwd=c(2.5))

## simulate stock paths
library(sde)
```

```
## Warning: package 'sde' was built under R version 3.2.4
```

```
## Loading required package: MASS
## Loading required package: stats4
## Loading required package: fda
```

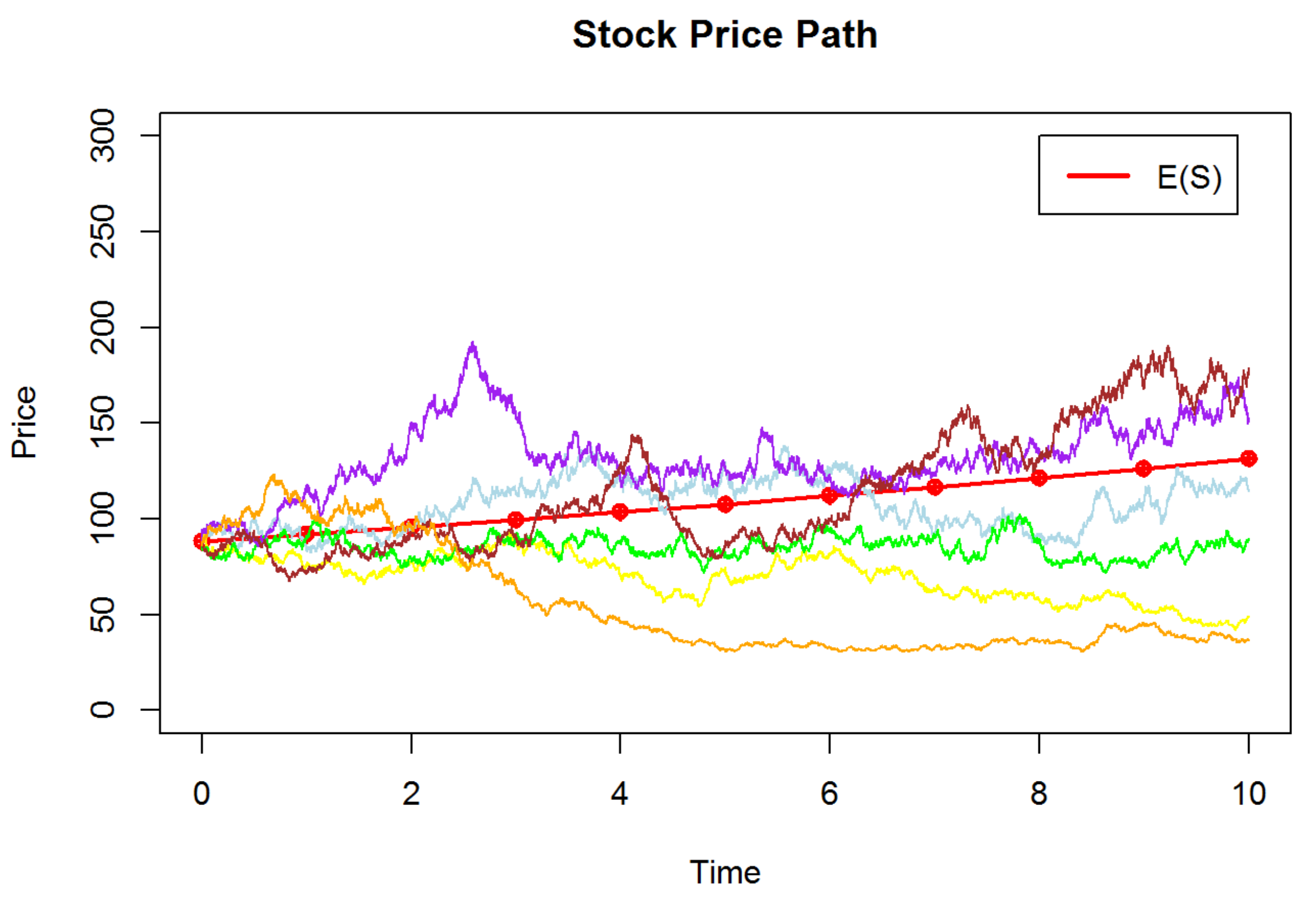
```
## Warning: package 'fda' was built under R version 3.2.4
```

```
## Loading required package: splines
## Loading required package: Matrix
##
## Attaching package: 'fda'
##
## The following object is masked from 'package:graphics':
##
##     matplot
##
## Loading required package: zoo
```

```
## Warning: package 'zoo' was built under R version 3.2.3
```

```
##
## Attaching package: 'zoo'
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
##
## sde 2.0.14
## Companion package to the book
## 'Simulation and Inference for Stochastic Differential Equations With R Examples'
## Iacus, Springer NY, (2008)
## To check the errata corrige of the book, type vignette("sde.errata")
```

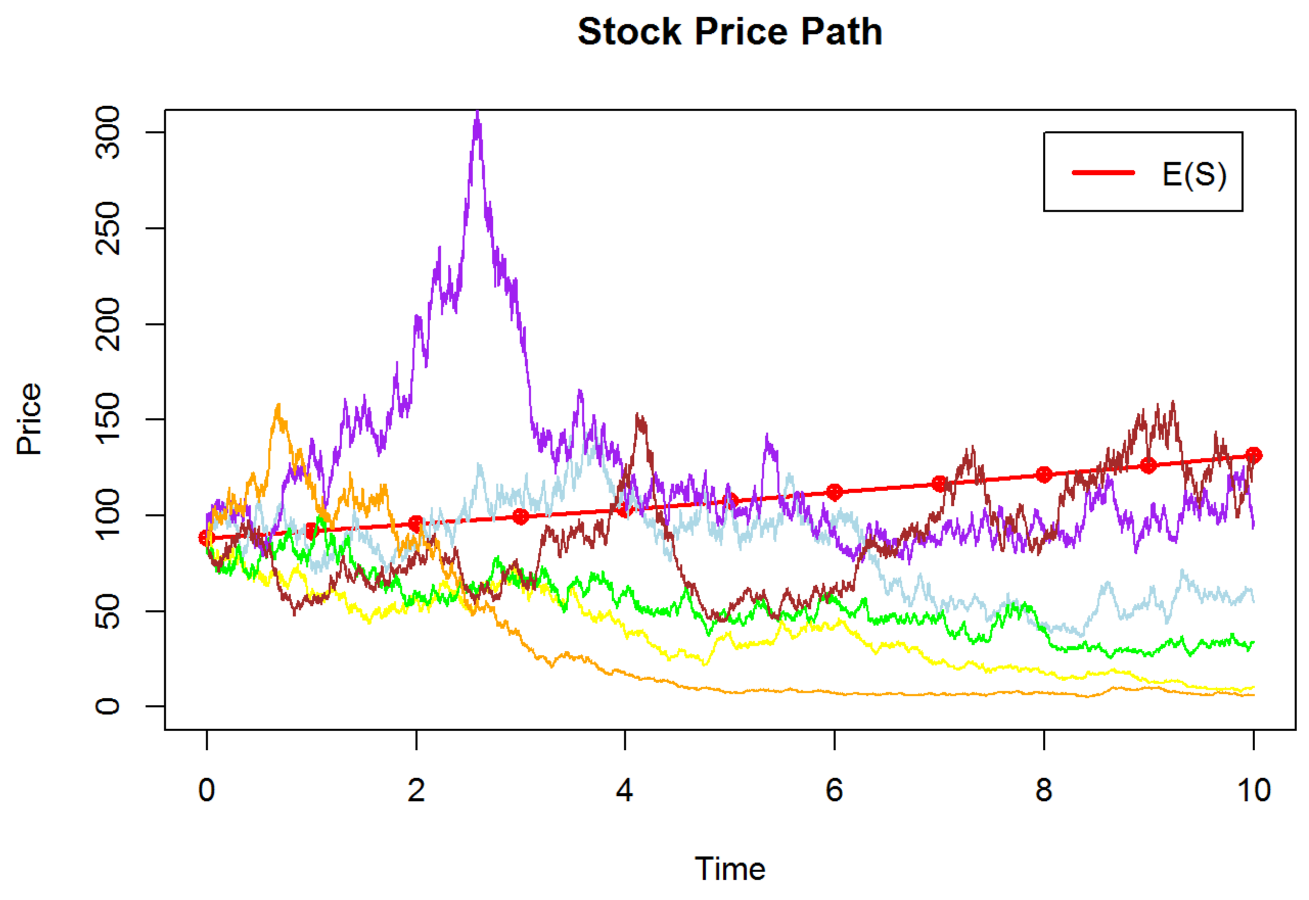
```
suppressWarnings(suppressMessages(library(sde)))
t<- n/nsims
simS <- matrix(NA,ncol=7, nrow = nsims+1)
simS[,7] <- seq(0,10,10/nsims)
col <- c("lightblue","yellow","green","purple","brown","orange")
for (j in 1:6) {
#simS[,j] <- GBM(S0,r,sigma,T=10,N=1000)
simS[,j] <- S0*cumprod(1+ c(0,(r-sigma*sigma*0.5)*t + sigma*sqrt(t)*simnorm(nsims,j/1
0)))
lines(simS[,7],simS[,j],type='l', col=col[j])
}
```



5.d

```
sigma <- 0.35
nsims <- 10000
S <- matrix(NA,nrow=nsims,ncol=10)
for (i in 1:n){
Z <- simnorm(nsims,0.01)
S[,i] <- S0 * exp(sigma*sqrt(i)*Z + (r-0.5*sigma^2)*i)
}
ES <- c(S0,colMeans(S))
plot(0:n,ES,ylim=c(0,300),main="Stock Price Path", xlab= "Time", ylab="Price", type
='o', col = "red", pch=10, lty = 1, lwd =2)
legend(8,300,c("E(S)"),col=("red"), lty=c(1), lwd=c(2.5))

## simulate stock paths
library(sde)
t<- n/nsims
simS <- matrix(NA,ncol=7, nrow = nsims+1)
simS[,7] <- seq(0,10,10/nsims)
col <- c("lightblue","yellow","green","purple","brown","orange")
for (j in 1:6) {
#simS[,j] <- GBM(S0,r,sigma,T=10,N=1000)
simS[,j] <- S0*cumprod(1+ c(0,(r-sigma*sigma*0.5)*t + sigma*sqrt(t)*simnorm(nsims,j/1
0)))
lines(simS[,7],simS[,j],type='l', col=col[j])
}
```



As σ increases the expected path E(S) diverges from the random simulated paths. Also, the spread between the stock paths also increases.

Question 6

6.a

```
# Euler Method
n <- 10000
tmax <- 1
h <- tmax/n
time <- seq(0,1,1/n)
F <- numeric()
F[1]<-0
for (j in 1:(n+1)){
F[j+1] = F[j] + 4*h*sqrt(1-time[j]^2)
}
Ia <- F[j]
```

Ia : 3.1417915

6.b

```
# Monte Carlo
uniform <- lgm(n,0.01)
M <- numeric()
for (i in 1:length(uniform)){
M[i] <- 4*sqrt(1-uniform[i]^2)
}
Ib <- mean(M)
```

Ib : 3.1184567
Variance : 0.8323677

6.c

```
#Importance Sampling
n <- 10000
x<- lgm(n,1)
a<- 0.74
# used for g(x) function
max <- 1/(1-a/3)
#since the range is (0,1) then h(x) is uniform
h <- lgm(n,2)
sample <- numeric()
for(i in 1:n){
if( x[i] <= (1-a*h[i]^2)){
sample <- c(sample,h[i])
}
}
# after the sample for t(x) is obtained, use them to calculate the expectation value
t <- numeric()
for (i in 1:length(sample)) {
t[i] <- 4*(1-a/3)*sqrt(1-sample[i]^2)/(1-a*sample[i]^2)
}
Ic <- mean(t)
```

Ic : 3.1491572
variance : 0.0176987

The use of importance sampling doesn’t improve the result than Euler method. But it has lesser variance compared to Euler’s method.