MGMTMFE 405 Project 2

Yanxiang Zhao

January 24, 2019

```
# ANSWERS
# set seed
seed <<- as.numeric(Sys.time())
```

Question 1

```
##### Question 1 #####
q1 <- function(seed){
  n <- 10000
  z1 <- rnorm_pmm(seed, n)
  z2 <- rnorm_pmm(seed*100, n)
  mu <- c(0,0)
  sigma <- matrix(c(1,-0.7, -0.7, 1), 2, 2)
  # generate x~N(0,1) and y~N(0,1) with Cov(x,y) = -0.7
  x <- mu[1] + sigma[1,1]*z1
  y <- mu[2] + sigma[1,2]/sigma[1,1]*z1 + sigma[2,2]*sqrt(1-(sigma[1,2]/(sigma[1,1]*sigma[2,2]))^2)*z2
  # compute correlation
  rho <- (1/(n-1)*sum((x-sum(x)/n)*(y-sum(y)/n)))/
         (sqrt(1/(n-1)*sum((x-mean(x))^2))*sqrt(1/(n-1)*sum((y-mean(y))^2)))
  return(rho)
}
# find the rho
rho <- q1(seed)
cat("rho = ", rho)
```

```
## rho =  -0.6982796
```

Question 2

```
##### Question 2 #####
q2 <- function(seed){
  n <- 10000
  z1 <- rnorm_pmm(seed, n)
  z2 <- rnorm_pmm(seed*100, n)
  rho <- 0.6
  # generate x~N(0,1) and y~N(0,1) with Cov(x,y) = -0.7
  x <- z1
  y <- rho*z1 + sqrt(1-rho^2)*z2
  # compute expected value
  output <- mean(apply(cbind(rep(0, n), x^3+sin(y)+x^2*y),1,max))
  return(output)
}
# find the expected value
```

```r
E <- q2(seed)
cat("The expected value = ", E)
```

```
## The expected value =  1.51894
```

Question 3

(a)

```r
##### Question 3 #####
# (a) Estimate expected values by simulation
q3a1 <- function(seed){
  w_5 <- sqrt(5)*rnorm_pmm(seed, 10000)
  output <- (w_5^2+sin(w_5))
  return(output)
}
q3a2 <- function(seed, t){
  z1 <- rnorm_pmm(seed, 10000)
  w <- sqrt(t)*z1
  output <- (exp(t/2)*cos(w))
  return(output)
}
# outputs
Ea1 <- q3a1(seed)
Ea2 <- q3a2(seed, 0.5)
Ea3 <- q3a2(seed, 3.2)
Ea4 <- q3a2(seed, 6.5)
rbind(mean = c(Ea1 = mean(Ea1), Ea2 = mean(Ea2), Ea3 = mean(Ea3), Ea4 = mean(Ea4)),sd = c(sd(Ea1), sd(E
```

```
##           Ea1       Ea2       Ea3        Ea4
## mean 4.982587 1.0011348 0.9896037  0.8569935
## sd   7.085861 0.3544853 3.3638103 18.1008166
```

(b)

The expected value of the last three integrals all equal to 1, but the longer integrals (t = 3.2 and 6.5) have higher variance; therefore, the simulated results can be more widely spread

(c)

```r
# (c) variance reduction
q3b1 <- function(seed){
  z1 <- rnorm_pmm(seed, 10000)
  w_5a <- sqrt(5)*z1
  w_5b <- sqrt(5)*(-1*z1)
  output <- apply(cbind(w_5a^2+sin(w_5a),w_5b^2+sin(w_5b)), 1, mean)
  return(output)
}
```

```r
q3b2 <- function(seed, t){
  z1 <- rnorm_pmm(seed, 10000)
  w_a <- sqrt(t)*z1
  w_b <- sqrt(t)*(-1*z1)
  output <- apply(cbind(exp(t/2)*cos(w_a),exp(t/2)*cos(w_b)),1,mean)
  return(output)
}
# outputs: variance reduction
Eb1 <- q3b1(seed)
Eb2 <- q3b2(seed, 0.5)
Eb3 <- q3b2(seed, 3.2)
Eb4 <- q3b2(seed, 6.5)
cat("variance reduction\n")
```

```
## variance reduction
```

```r
rbind(mean = c(Eb1 = mean(Eb1), Eb2 = mean(Eb2), Eb3 = mean(Eb3), Eb4 = mean(Eb4)),sd = c(sd(Eb1), sd(E
```

```
##           Eb1       Eb2       Eb3         Eb4
## mean 4.977131 1.0011348 0.9896037  0.8569935
## sd   7.047462 0.3544853 3.3638103 18.1008166
```

I used antithetic variates to reduce the variance. It has little effect on Eb1 to reduce the variance and it has no effect on Eb2, Eb3 and Eb4. The reason for this is because of the trigonometric functions in the simulation. They are asymmetric at $W_t = 0$, where it is the expected value of the Brownian motions. When I use the antithetic variates, both variates yield the same output at $N(Z_i)$ and $N(-Z_i)$.

Question 4

(a)

```r
##### Question 4 #####
# (a) Estimate the price of a European Call option with simulation
q4a <- function(seed){
  n <- 10000
  # define variables as given
  r <- 0.04
  sigma <- 0.2
  s_0 <- 88
  t <- 5
  x <- 100
  # generate random normal variates
  z1 <- rnorm_pmm(seed, n)
  # calculate the payoffs
  s_T <- s_0*exp((r-sigma^2/2)*t+sigma*sqrt(t)*z1)
  payoffs <- apply(cbind(rep(0,n), s_T-x), 1, max)
           # apply the max function each row to find payoffs >= 0
  # take PV of expected payoff
  c <- exp(-r*t)*(payoffs)
  return(c)
```

```r
}
# output
Ca1 <- q4a(seed)
cat("European call price by simulation: $", mean(Ca1), "with sd = $", sd(Ca1))
```

```
## European call price by simulation: $ 18.01764 with sd = $ 31.68527
```

(b)

```r
# (b) 1. Compute option price by Black-Scholes formula
# define variables as given
r <- 0.04
sigma <- 0.2
s_0 <- 88
t <- 5
x <- 100
# compute d1, d2
d1 <- (log(s_0/x)+(r+sigma^2/2)*t)/(sigma*sqrt(t))
d2 <- d1-sigma*sqrt(t)
# output: Black-Sholes
Cb1 <- s_0*pnorm(d1)-x*exp(-r*t)*pnorm(d2)
       #pnorm() is the built-in normal cumulative density function
cat("European call price by Black-Scholes model: $", Cb1)
```

```
## European call price by Black-Scholes model: $ 18.28377
```

```r
# (b) 2. use variance reduction
q4b <- function(seed){
  n <- 10000
  # define variables as given
  r <- 0.04
  sigma <- 0.2
  s_0 <- 88
  t <- 5
  x <- 100
  # generate random normal variates
  z1 <- rnorm_pmm(seed, n)
  # calculate the payoffs with antithetic variates
  # (+z1)
  s_T1 <- s_0*exp((r-sigma^2/2)*t+sigma*sqrt(t)*z1)
  payoffs1 <- apply(cbind(rep(0,n), s_T1-x), 1, max)
  c1 <- exp(-r*t)*(payoffs1)
  # (-z1)
  s_T2 <- s_0*exp((r-sigma^2/2)*t+sigma*sqrt(t)*(-z1))
  payoffs2 <- apply(cbind(rep(0,n), s_T2-x), 1, max)
  c2 <- exp(-r*t)*(payoffs2)
  # take the expected value of the mean
  c <- (apply(cbind(c1,c2),1,mean))
  return(c)
}
```

```r
# output
Cb2 <- q4b(seed)
cat("European call price by simulation with variance reduction: $", mean(Cb2), "with sd = $", sd(Cb2))
```

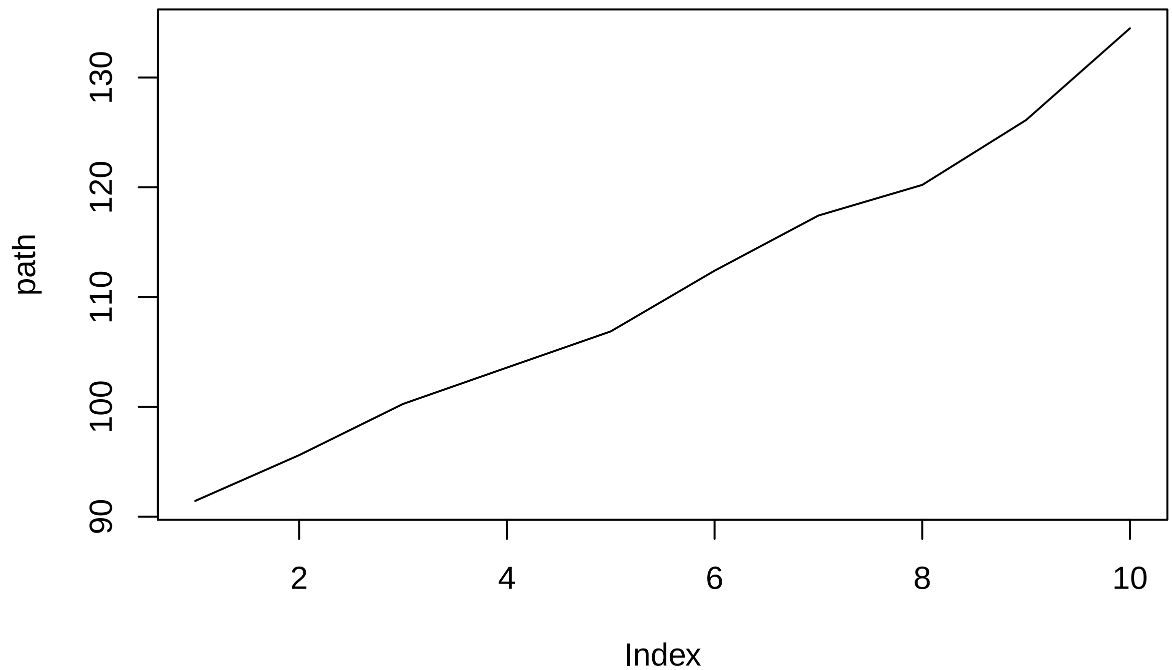
```
## European call price by simulation with variance reduction: $ 18.24949 with sd = $ 18.56688
```

There is a significant imporvement on accuricy with the variance decrease. The mean get closer to the Black-Scholes theoretical value, and the standard deviation is lower.

Question 5

(a)

```r
##### Question 5 #####
# (a) find E[Sn] and plot
q5a <- function(seed, t){
  n <- 1000
  # define variables as given
  r <- 0.04
  sigma <- 0.18
  s_0 <- 88
  # generate random normal variates
  z1 <- rnorm_pmm(seed, n)
  # calculate s_T with antithetic variates
  # (+z1)
  s_T1 <- s_0*exp((r-sigma^2/2)*t+sigma*sqrt(t)*z1)
  # (-z1)
  s_T2 <- s_0*exp((r-sigma^2/2)*t+sigma*sqrt(t)*(-z1))
  # take the expected value of the mean
  s_T <- mean(apply(cbind(s_T1,s_T1),1,mean))
  return(s_T)
}
# generate the path
path <- rep(0,10)
for(t in 1:10){
  path[t] <- q5a(seed*t, t)
}
plot(path, type = "l")
```



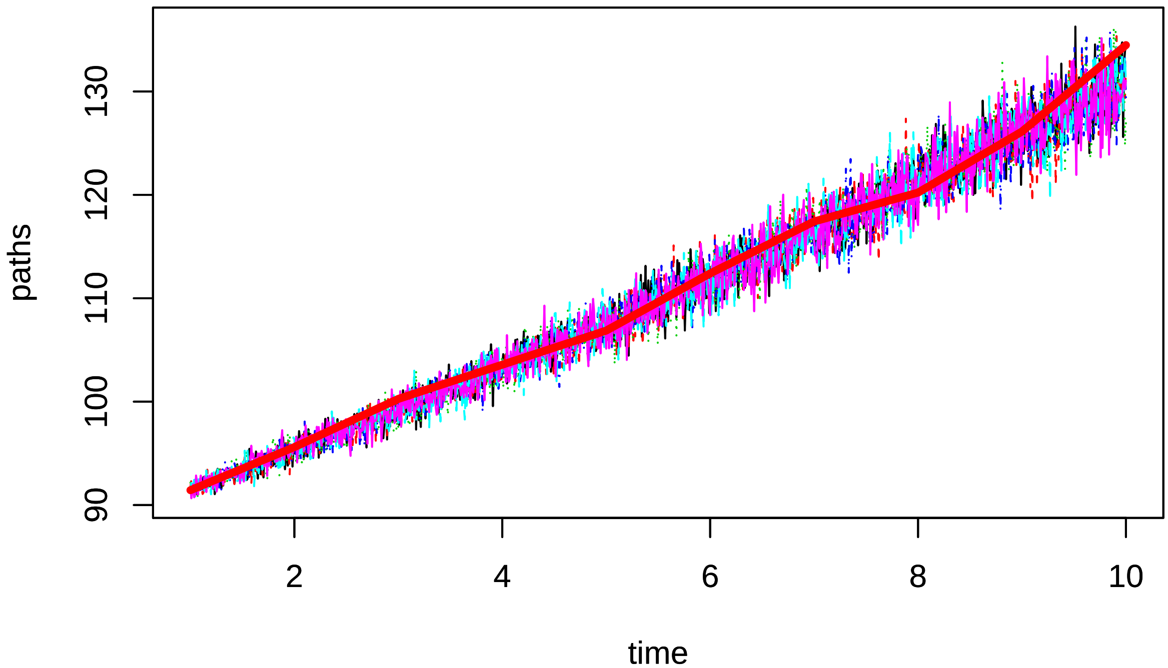
(b)

```
# (b) finer time intervals
N <- 1000
# generate 6 paths
paths <- matrix(rep(rep(0,N),6),N,6)
for (i in 1:6) {
  step <- 1
  # take finer time intervals
  for(t in seq(1, 10, (10-1)/(N-1))){
    paths[step,i] <- q5a(seed*t*i, t)
    step <- step + 1
  }
}
```

(c)

```
# (c) plot a and b in one plot
time <- matrix(rep(seq(1, 10, (10-1)/(N-1)),6),N,6)
matplot(time, paths, type = "l")
par(new=TRUE)
```

```r
plot(path, type = "l", lwd = 4, col = "red",
     ylim=c(min(paths), max(paths)), xlab = "time", ylab = "paths")
```



(d)

The ES_n graph will take a upward trend (move upward overtime) and it will be more volatile when the σ increases from 18% to 35%, because the increase in σ increase the drift by $\frac{\sigma_\Delta^2}{2}dt$, and the diffusion by $\sigma_\Delta dW_t$, where σ_Δ is the magnitude of change. For the same reason, the 6 plots of S_t will be tilted upward and be more volatile when the σ increases.

Question 6

(a)

```r
##### Problem 6 #####
# (a) Euler method
q6a <- function(x){
  pi <- 4*sqrt(1-x^2)
  return(pi)
}
# Riemann sum
rSum <- 0
```

```r
loops <- 1000000
for(i in seq(0,1,1/loops)){
  rSum <- rSum + q6a(i)*(1/loops)
}
# output
Ia <- rSum
cat("Euler method: pi = ", Ia)
```

```
## Euler method: pi =  3.141595
```

(b)

```r
# (b) simulation
q6b <- function(seed){
  pi <- (4*sqrt(1-runif(seed, 10000)^2))
  return(pi)
}
Ib <- q6b(seed)
cat("Monte Carlo simulation: pi = ", mean(Ib), ", sd = ", sd(Ib))
```

```
## Monte Carlo simulation: pi =  3.134569 , sd =  0.9043493
```

(c)

```r
# (c) importance sampling method
# model by Wei Cai, from the notes
x <- runif(seed, 10000)
a <- 0.74
g_x <- 4*sqrt(1-x^2)
t_x <- (1-0.74*x^2)/(1-a/3)
# output
Ic <- mean(g_x/t_x)
cat("Importance Sampling method: pi = ", Ic, ", sd = ", sd(g_x/t_x))
```

```
## Importance Sampling method: pi =  3.130337 , sd =  0.3249426
```

Importance Sampiling method helps the simulation to have a lower variance, but it compromised some of the accuracy. As we can see, the expected value from Importance Sampling method is biased downward.