# Assignment 9

YUYU ZHANHG

14/11/2020

## Exercise 7.5.1

a.Implement the important sampling method

```
f <- function(x) 1/(5*sqrt(2*pi))* x^2 * exp(- (x - 2)^2 / 2)
integrate(f, -Inf, Inf)
```

```
## 1 with absolute error < 1.1e-05
```

```
## Monte Carlo approximation
nrep1 <- 1000
nrep2 <- 10000
nrep3 <- 50000
n <- 10000

i.n <- replicate(nrep1, {x <- rnorm(n, -1, 2); mean(f(x) / dnorm(x, -1, 2))})
mean(i.n)
```

```
## [1] 0.9995351
```

```
sd(i.n)
```

```
## [1] 0.03401919
```

```
i.c <- replicate(nrep1, {x <- rcauchy(n, -1, 2); mean(f(x) / dcauchy(x, -1, 2))})



mean(i.c)
```

```
## [1] 1.000029
```

```
sd(i.c)
```

```
## [1] 0.02869831
```

As it shows the estimated standard variance is 0.029.

b. Design a better importance sampling method to estimate $E[X^2]$

I would like to use Gamma distribution as g(x) because we want to find out E($x^2$) and the estimator of $\mu = \frac{1}{n}\sum h(Z_i)$

c Implement your method and estimate E[$x^2$] using using 1000, 10000 and 50000 samples. Also estimate the variances of the importance sampling estimates.

```
isAppr <- function(n, h, df, dg, rg, ...) {
  x <- rg(n, ...)
```

```
  mean( h(x) * df(x) / dg(x, ...) )
}

alpha <- 2
h <- function(x) 1/(5*sqrt(2*pi))* x^2 * exp(- (x - 2)^2 / 2)
beta <- 2
df <- function(x) dexp(x, rate = 1 / beta)

mySummary <- function(nrep, n, h, df, dg, rg) {
##    browser()
    sim <- replicate(nrep, isAppr(n, h, df, dg, rg))
    c(mean = mean(sim), sd = sd(sim))
}

sapply(1:6,
       function(shp) {
           rg <- function(n) rgamma(n, shape = shp, scale = beta)
           dg <- function(x) dgamma(x, shape = shp, scale = beta)
           mySummary(1000, 1000, h, df, dg, rg)
       })
```

```
##             [,1]        [,2]        [,3]       [,4]       [,5]       [,6]
## mean 0.134515022 0.134593018 0.134447491 0.13463415 0.13302471 0.12980339
## sd   0.004811509 0.004102777 0.006378017 0.01316625 0.02784066 0.06866118
```

I just assume that g(x) is Gama distribution. and the result is more relaible than the standard normal distribution.

d. Compare the two results from the two methods and comment.

after changing the g(x) from standard normal density to gamma distribution the estimated error narrows.

### Exercise 7.5.2

1.Write down and implement an algorithm to sample the path of S(t)

$$\frac{dS(t)}{S(t)} = \mu dt + \sigma dB(t)$$

Notice that the left hand side of this equation looks similar to the derivative of logS(t) . Applying Ito's Lemma to logS(t) gives:

$$d(logS(t)) = (logS(t))'\mu S(t)dt + (logS(t))'\sigma S(t)dB(t) + \frac{1}{2}(logS(t))''\sigma^2 S(t)^2 dt$$

This becomes:

$$d(logS(t)) = \mu dt + \sigma dB(t) - \frac{1}{2}\sigma^2 dt = (\mu - \frac{1}{2}\sigma^2)dt + \sigma dB(t)$$

This is an Ito drift-diffusion process. It is a standard Brownian motion with a drift term. Since the above formula is simply shorthand for an integral formula, we can write this as:

$$log(S(t)) - log(S(0)) = (\mu - \frac{1}{2}\sigma^2)t + \sigma B(t)$$

Finally, taking the exponential of this equation gives:

$$S(t) = S(0)exp((\mu - \frac{1}{2}\sigma^2)t + \sigma B(t))$$

This is the solution the stochastic differential equation

2. Set $\sigma = 0.5$ and T=1

```
blackScholes <- function(strike, time, S0, rfrate, sigma) {
    crit <- (log(strike / S0) - (rfrate - sigma^2 / 2) * time)/ sigma / sqrt(time)
    S0 * pnorm(sigma * sqrt(time) - crit) -
      strike * exp(- rfrate * time) * pnorm(-crit)
}


Sa<- function(strike, time, S0, rfrate, sigma) {
    crit <- (log(strike / S0) - (rfrate - sigma^2 / 2) * time)/ sigma / sqrt(time)
    (1/12 )*S0*(13)/2-
      strike * exp(- rfrate * time) * pnorm(-crit)
}

Sg<- function(strike, time, S0, rfrate, sigma,n) {
   crit <- (log(strike / S0) - (rfrate - sigma^2 / 2) * time)/ sigma / sqrt(time)
  (sqrt(factorial(12)))  -
        strike * exp(- rfrate * time) * pnorm(-crit)
}
nsim<-5000
strike<-1.1
S0 <-1
time<-1
rfrate <- 0.05
sigma <- 0.5
## analytic solution
blackScholes(strike, time, S0, rfrate, sigma)
```

```
## [1] 0.1796232
```

```
Sa(strike, time, S0, rfrate, sigma)
```

```
## [1] 0.1579748
```

```
Sg(strike, time, S0, rfrate, sigma)
```

```
## [1] 21885.72
```

```
## Monte Carlo approximation
myApprox <- function(msim, strike, time, S0, rfrate, signma) {
    wt <- rnorm(nsim, sd = sqrt(time) * sigma)
    value <- mean(pmax(S0 * exp((rfrate - sigma^2 / 2) * time + wt) - strike, 0))
    exp(-rfrate * time) * value
}




mcAppr<-replicate(200, myApprox(nsim, strike, time, S0, rfrate, sigma))
```

```r
sd(mcAppr)
```

```
## [1] 0.005392516
```

```r
mean(mcAppr)
```

```
## [1] 0.1793667
```

By change the value of strike price , we see the larger K is , the smaller corrlation Pa and Pe with.

c. Set T=1 and K=1.5

```r
myApprox <- function(msim, strike, time, S0, rfrate, signma) {
    wt <- rnorm(nsim, sd = sqrt(time) * sigma)
    value <- mean(pmax(S0 * exp((rfrate - sigma^2 / 2) * time + wt) - strike, 0))
    exp(-rfrate * time) * value
}

S0 <-1
time<-1
rfrate <- 0.05
sigma <- 0.4

nsim<-5000
strike<-1.5
mcAppr<-replicate(200, myApprox(nsim, strike, time, S0, rfrate, sigma))
sd(mcAppr)
```

```
## [1] 0.002495185
```

```r
mean(mcAppr)
```

```
## [1] 0.04832051
```

well it is obvious that the greater $\sigma$ is , the correlation coefficients get greater.

d. Set $\sigma$=0.5 and K=1.5

```r
myApprox <- function(msim, strike, time, S0, rfrate, signma) {
    wt <- rnorm(nsim, sd = sqrt(time) * sigma)
    value <- mean(pmax(S0 * exp((rfrate - sigma^2 / 2) * time + wt) - strike, 0))
    exp(-rfrate * time) * value
}

S0 <-1
time<-0.8
rfrate <- 0.05
sigma <- 0.5

nsim<-5000
strike<-1.5
mcAppr<-replicate(200, myApprox(nsim, strike, time, S0, rfrate, sigma))
sd(mcAppr)
```

```
## [1] 0.003086645
```

```r
mean(mcAppr)
```

```
## [1] 0.0613267
```

It is shown that the greater T is , the correlation coefficients get greater.

e. Set $\sigma$=0.4 and K=1.5 T=1

```
rBM <- function(n, tgrid, sigma) {
    tt <- c(0, tgrid)
    dt <- diff(tt)
    nt <- length(tgrid)
    dw <- matrix(rnorm(n * nt, sd = sigma * sqrt(dt)), n, nt, byrow = TRUE)
    t(apply(dw, 1, cumsum))
}

callValLognorm <- function(S0, K, mu, sigma) {
    d <- (log(S0 / K) + mu + sigma^2) / sigma
    S0 * exp(mu + 0.5 * sigma^2) * pnorm(d) - K * pnorm(d - sigma)
}

optValueAppr <- function(n, r, sigma, S0, K, tgrid) {
    wt <- rBM(n, tgrid, sigma)
    ## payoff of call option on arithmetic average
    nt <- length(tgrid)
    TT <- tgrid[nt]
    St <- S0 * exp((r - sigma^2 / 2) * matrix(tgrid, n, nt, byrow = TRUE) + wt)
    pAri <- pmax(rowMeans(St) - K, 0)
    vAri <- mean(pAri)
    ## underlying asset price
    ST <- St[, nt]
    vAs <- vAri - cov(ST, pAri) / var(ST) * (mean(ST) - exp(r * TT) * S0)
    ## value of standard option
    pStd <- pmax(ST - K, 0)
    pStdTrue <- callValLognorm(S0, K, (r - 0.5 * sigma^2) * TT,
                               sigma * sqrt(TT))
    vStd <-  vAri - cov(pStd, pAri) / var(pStd) * (mean(pStd) - pStdTrue)
    ## payoff of call option on geometric average
    pGeo <- pmax(exp(rowMeans(log(St))) - K, 0)
    tbar <- mean(tgrid)
    sBar2 <- sigma^2 / nt^2 / tbar * sum( (2 * seq(nt) - 1) * rev(tgrid) )
    pGeoTrue <- callValLognorm(S0, K, (r - 0.5 * sigma^2) * tbar,
                               sqrt(sBar2 * tbar))
    vGeo <- vAri - cov(pGeo, pAri) / var(pGeo) * (mean(pGeo) - pGeoTrue)
    ## sim <- data.frame(pAri, ST, pStd, pGeo)
    ## result
    c(vAri, vAs, vStd, vGeo) * exp(-r * TT)
}

r <- 0.05; sigma <- 0.4; S0 <-  1; K <-  1.5
tgrid <-  seq(0, 1.2, length = 14)[-1]
sim <- replicate(200, optValueAppr(500, r, sigma, S0, K, tgrid))
apply(sim, 1, mean)
```

```
## [1] 0.01283299 0.01290992 0.01295427 0.01320438
```

```
apply(sim, 1, sd)
```

```
## [1] 0.003224560 0.002772687 0.002170337 0.000445242
```

based on the result above, we could see that the standard error are smaller then the SD of the MC estimator for $\mathrm{E}(P_A)$ that has no control variate.