Assignment 9

YUYU ZHANHG

14/11/2020

7.5.1

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

a.

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

```r
## 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] 1.0013
```

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

```
## [1] 0.03470389
```

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



mean(i.c)
```

```
## [1] 1.000617
```

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

```
## [1] 0.02968756
```

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

```r
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)
       })
```

c

```
##             [,1]        [,2]        [,3]       [,4]       [,5]       [,6]
## mean 0.134666983 0.134360700 0.134724782 0.13347857 0.13262571 0.13257157
## sd   0.005051541 0.004137639 0.006451849 0.01297198 0.02704814 0.06681102
```

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

d

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

```r
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)
```

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

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

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

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

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

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

```r
## 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))
sd(mcAppr)
```

```
## [1] 0.005769494
```

```r
mean(mcAppr)
```

```
## [1] 0.179939
```

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

```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)
```

c

```
## [1] 0.002584633
```

```r
mean(mcAppr)
```

```
## [1] 0.0486021
```

well it is obvious that the greater σ is , the correlation coefficients get greater.

####d

```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] 0.003205864
```

```r
mean(mcAppr)
```

```
## [1] 0.06205304
```

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