MGMTMFE 405 - Project 5

Yanxiang Zhao

February 22, 2019

Functions

Question 1 Functions

```
##### QUESTION 1 #####
# function to simulate stock paths
sPaths <- function(s0, r, sigma, t, nSim, n){
  dt <- t/n
  sim <- list()
  for (i in 1:(nSim/2)){
    dw <- sqrt(dt)*rnorm(n)
    s_sim1 <- rep(s0,n)
    s_sim2 <- rep(s0,n)
    for (s in 2:n){
      # create anthithetic paths
      s_sim1[s] <- s_sim1[s-1]*(1 + r*dt + sigma*dw[s])
      s_sim2[s] <- s_sim2[s-1]*(1 + r*dt - sigma*dw[s])
    }
    sim_i <- list(s_sim1, s_sim2)
    sim <- append(sim, sim_i)
  }
  out <- matrix(unlist(sim), nSim, n, byrow = T)
  return(out)
}
# least squared Monte Carlo simulation
lsmc <- function(paths, strike, r, t, nSim, n, func, k){
  dt <- t/n
  # calculate exercise value at each time
  ev <- ifelse(strike-paths>0, strike-paths,0)
  # construct index
  ind <- cbind(diag(0, nSim, n-1), ifelse(ev[,n]>0, 1,0))
  # for loop through time
  for (i in 1:(n-2)){
    # define x
    x <- paths[,(n-i)]
    # define discount factors
    dsc <- exp(-r*dt*matrix(rep(1:i,each = nSim), nSim, i))
    # define y
    y <- apply(matrix(ind[,(n-i+1):n]*ev[,(n-i+1):n]*dsc,
                      nSim, i),1,sum)
    # define basis functions
    if (func == "Laguerre"){
      L <- function(x, k){
        out <- switch(k,
                      exp(-x/2),
                      exp(-x/2)*(1-x),
```

```r
                     exp(-x/2)*(1-2*x+x^2/2),
                     exp(-x/2)*(1-3*x+(3*x^2)/2-x^3/6))
      return(out)
    }
  }
  else if (func == "Hermite"){
    L <- function(x, k){
      out <- switch(k,
                     x^0,
                     2*x,
                     4*x^2-2,
                     8*x^3-12*x)
      return(out)
    }
  }
  else if (func == "Monomials"){
    L <- function(x, k){
      return(x^(k-1))
    }
  }
  else {
    stop("Incorrect function type!")
  }
  # Find matrix A and vector b
  A <- diag(0, k, k)
  b <- vector()
  for (p in 1:k){
    for (q in 1:k){
      A[p,q] <- sum(L(x,p)*L(x,q))
    }
    b[p] <- sum(y*L(x,p))
  }
  # find a_i coefficients, using cholesky roots to inverse matrix
  a <- chol2inv(chol(A))%*%b

  # find expected continuation values
  est <- vector()
  for (p in 1:k){
    est <- cbind(est, L(x,p))
  }
  expCv <- est%*%a

  # rewirte index
  ind[,(n-i)] <- ifelse(ev[,(n-i)]>expCv, 1, 0)
  # write off 1s in previous
  ind <- t(apply(ind, 1, function(x) {
    if (sum(x)>1){
      replace(x, (which(x>0)[1]+1):n, 0)
    }
    else {x}
  }))
}
# create discount matrix
```

```
  dsct <- exp(-r*dt*matrix(rep(0:(n-1),each = nSim), nSim, n))

  # compute option value at time 0
  v0 <- mean(apply(ind*dsct*ev, 1, sum))
  return(v0)
}
```

Question 2 Functions

```
##### QUESTION 2 #####
# function to price a forward-start European put option
fwdPutEu <- function(s0, r, sigma, t_x, t_n, nSim, n){
  dt <- t_n/n
  # find # steps for t
  fwdStep <- floor(t_x/dt)
  paths <- sPaths(s0, r, sigma, t_n, nSim, n)
  put <- exp(-r*t_n)*mean(ifelse(paths[,fwdStep]>paths[,n], paths[,fwdStep]-paths[,n], 0))
  return(put)
}
# function to price a forward-start American put option
fwdPutAm <- function(s0, r, sigma, t_x, t_n, nSim, n, func, term){
  dt <- t_n/n
  fwdStep <- floor(t_x/dt)
  paths <- sPaths(s0, r, sigma, t_n, nSim, n)
  strike <- paths[,fwdStep]
  # use LSMC to find put price at t_x using stock price from t_x to t_n
  put_tx <- lsmc(paths[,fwdStep:n], strike, r, t_n-t_x, nSim, n-fwdStep+1, func, term)
  put <- exp(-r*t_x)*put_tx
  return(put)
}
```

Question 1

```
# set parameters
s0 <- c(36, 40, 44)
t <- c(0.5, 1, 2)
term <- 2:4
strike <- 40
r <- .06
sigma <- .2
nSim <- 100000
n <- 20
# (a) Laguerre polynomials
func <- "Laguerre"
put_1a <- rep(list(diag(0,3,3)),3)
for (i in 1:length(s0)){
  for (j in 1:length(t)){
    # simulate stock price paths
    paths <- sPaths(s0[i], r, sigma, t[j], nSim, n)
```

```
    for (k in 1:length(term)){
      cat("Laguerre - i=",i, "j=",j, "k=",k, "\n")
      put_1a[[i]][j,k] <- lsmc(paths, strike, r, t[j], nSim, n, func, term[k])
    }
  }
}
# (b) Hermite polynomials
func <- "Hermite"
put_1b <- rep(list(diag(0,3,3)),3)
for (i in 1:length(s0)){
  for (j in 1:length(t)){
    # simulate stock price paths
    paths <- sPaths(s0[i], r, sigma, t[j], nSim, n)
    for (k in 1:length(term)){
      cat("Hermite - i=",i, "j=",j, "k=",k, "\n")
      put_1b[[i]][j,k] <- lsmc(paths, strike, r, t[j], nSim, n, func, term[k])
    }
  }
}
# (c) Simple Monomials
func <- "Monomials"
put_1c <- rep(list(diag(0,3,3)),3)
for (i in 1:length(s0)){
  for (j in 1:length(t)){
    # simulate stock price paths
    paths <- sPaths(s0[i], r, sigma, t[j], nSim, n)
    for (k in 1:length(term)){
      cat("Monomials - i=",i, "j=",j, "k=",k, "\n")
      put_1c[[i]][j,k] <- lsmc(paths, strike, r, t[j], nSim, n, func, term[k])
    }
  }
}
```

S0	T	k = 2	k = 3	k = 4	Theoretical Value
36	0.5	3.932902	4.068248	4.159211	4.2059
	1	3.892404	4.082968	4.282081	4.4927
	2	3.879300	4.059111	4.327712	4.8369
40	0.5	1.455967	1.689181	1.750823	1.7886
	1	1.601209	1.885293	2.139126	2.3090
	2	1.784304	2.065041	2.393952	2.8756
44	0.5	0.511839	0.589174	0.591827	0.6342
	1	0.744885	0.940780	1.049343	1.1197
	2	0.988500	1.180811	1.410762	1.6889

Laguerre Polynomials

Figure 1: "Laguerre Polynomials"

S_0	T	k = 2	k = 3	k = 4	Theoretical Value
36	0.5	4.061969	4.153720	4.149758	4.2059
	1	4.276895	4.399100	4.400738	4.4927
	2	4.587585	4.668188	4.727417	4.8369
40	0.5	1.643428	1.694753	1.724972	1.7886
	1	2.125260	2.156258	2.230139	2.3090
	2	2.680486	2.676226	2.770106	2.8756
44	0.5	0.500927	0.535986	0.568324	0.6342
	1	0.946540	0.984304	1.034119	1.1197
	2	1.494189	1.513548	1.568952	1.6889

Hermite Polynomials

Figure 2: "Hermite Polynomials"

S_0	T	k			Theoretical Value
		2	3	4	
36	0.5	4.053792	4.153462	4.145801	4.2059
	1	4.282590	4.396137	4.398127	4.4927
	2	4.568978	4.655633	4.719089	4.8369
40	0.5	1.647869	1.701991	1.727499	1.7886
	1	2.142059	2.174031	2.233835	2.3090
	2	2.665665	2.681818	2.770036	2.8756
44	0.5	0.503964	0.541032	0.570485	0.6342
	1	0.948718	0.985482	1.026759	1.1197
	2	1.492478	1.522689	1.576860	1.6889

Figure 3: "Simple Monomials"

Comments on the results:

Comparing the option prices from the different polynomials, I found:

- The option prices from the Laguerre polynomial converge quickly, but with the lowest accuracy.
- The option prices from the Hermite polynomial converge relatively slowly, and with better accuracy than the ones from Laguerre polynomial.
- the option prices from the simple monomial are very similar to the ones from the Hermite polynomial. They have the similar order of convergency and the similar accuracy.

Question 2

```
# set parameters
s0 <- 65
r <- .06
sigma <- .2
t_x <- 0.2
t_n <- 1
nSim <- 10000
n <- 200
```

(a) forward-start European put

```
fwdPutEu(s0, r, sigma, t_x, t_n, nSim, n)
```

```
## [1] 3.178205
```

(b) forward-start American put

```
# use Hermite polynomial for better accuracy
func <- "Hermite"
term <- 4
fwdPutAm(s0, r, sigma, t_x, t_n, nSim, n, func, term)
```

```
## [1] 3.687197
```