Untitled

Shihao Zhang

2018-10-26

HW3 Problem 1 1.Implement the Explicit Finite Difference method

```
Explicit_finite <- function(isCall,K,Tm,S0,sigma,r,div,N,Nj){
  #precompute constants
  dt <- Tm/N
  dx <- sigma*sqrt(3*dt)
  nu <- r-div-0.5*sigma^2
  edx <- exp(dx)
  pu <- 0.5*dt*((sigma/dx)^2+nu/dx)
  pm <- 1-dt*(sigma/dx)^2-r*dt
  pd <- 0.5*dt*((sigma/dx)^2-nu/dx)
  #initialize asset prices at maturity
  V=S=matrix(0,nrow=2*Nj+1,ncol=N+1)
  cp <- ifelse(isCall,1,-1)
  S[2*Nj+1,N+1] <- S0*exp(-Nj*dx)
  for (i in (2*Nj):1) {
    S[i,N+1]=S[i+1,N+1]*exp(dx)
  }
  #initial option value at maturity
  for (i in (2*Nj+1):1) {
    V[i,N+1] <- max(0,cp*(S[i,N+1]-K))
  }
  #step back through lattice
  for (j in N:1) {
    for (i in (2*Nj):2) {
      V[i,j] <- pu*V[i-1,j+1]+pm*V[i,j+1]+pd*V[i+1,j+1]
    }
    #boundary conditions
    stockTerm <- ifelse(isCall,S[1,N+1]-S[2,N+1],
                        S[2*Nj,N+1]-S[2*Nj+1,N+1])
    V[2*Nj+1,j] <- V[2*Nj,j]+ifelse(isCall,0,stockTerm)
    V[1,j] <- V[2,j]+ifelse(isCall,stockTerm,0)
  }
  V[Nj+1,1]
}
```

2.Implement the Implicit Finite Difference method

```
Implicit_finite <- function(isCall,K,Tm,S0,sigma,r,div,N,Nj){
  #Implicit Finite Difference Method with i times,2*i+1 final nodes
  #precompute constants
  dt <- Tm/N
  dx <- sigma*sqrt(3*dt)
  nu <- r-div-0.5*sigma^2
  edx <- exp(dx)
  pu <- -0.5*dt*((sigma/dx)^2+nu/dx)
  pm <-  1.0+dt* (sigma/dx)^2+r*dt
  pd <- -0.5*dt*((sigma/dx)^2-nu/dx)
```

```r
  firstRow <- 1
  nRows = lastRow = 2*Nj+1
  firstCol <- 1
  middleRow <- Nj+1
  nCols = lastCol = N+1

  cp <- ifelse(isCall,1,-1)

  #initialize asset prices, derivative prices
  pmp=pp=V=S=matrix(0,nrow=nRows,ncol=nCols)
  S[lastRow,lastCol] <- S0*exp(-Nj*dx)
  for (j in (lastRow-1):firstRow) {
    S[j,lastCol]=S[j+1,lastCol]*exp(dx)
  }

  #initial option value at maturity
  for (j in (firstRow):lastRow) {
    V[j,lastCol] <- max(0,cp*(S[j,lastCol]-K))
  }

  #compute derivative boundary condition
  lambdaL <- ifelse(isCall,0,(S[lastRow,lastCol]-S[lastRow-1,lastCol]))
  lambdaU <- ifelse(isCall,(S[firstRow,lastCol]-S[firstRow+1,lastCol]),0)

  #step back through lattice
  for (i in (lastCol-1):firstCol) {
    #solve implicit tridiagonal system
    #substitute boundary condition at j=-Nj into j=-Nj+1
    pmp[(lastRow-1),i] <- pm+pd
    pp[(lastRow-1),i]  <- V[(lastRow-1),lastCol]+pd*lambdaL
    #eliminate upper diagonal
    for (j in (lastRow-2):(firstRow+1)) {
        pmp[j,i] <- pm-pu*pd/pmp[j+1,i]
        pp[j,i] <- V[j,i+1]-pp[j+1,i]*pd/pmp[j+1,i]
    }
    #use boundary condition at j=Nj and equation at j=Nj-1
    V[firstRow,i] <- (pp[(firstRow+1),i]+pmp[(firstRow+1),i]
                                 *lambdaU/(pu+pmp[(firstRow+1),i]))
    V[(firstRow-1),i] <- V[(firstRow),i]-lambdaU
    #back-substitution
    for (j in (firstRow+2):lastRow) {
        V[j,i] <- (pp[j,i]-pu*V[j-1,i])/pmp[j,i]
    }
    V[lastRow,i] <- V[(lastRow-1),i]-lambdaL
  }
  V[Nj+1,1]
}

Implicit_finite(T,100,1,100,0.2,0.06,0.03,3,3)
```

```
## [1] 3.843225
```

3.Implement the Crank-Nicolson Finite Difference method

```r
Crank_Nicolson_finite <- function(isCall,K,Tm,S0,sigma,r,div,N,Nj){
  #precompute constants
  dt <- Tm/N
  dx <- sigma*sqrt(3*dt)
  nu <- r-div-0.5*sigma^2
  edx <- exp(dx)
  pu <- -0.25    *dt*((sigma/dx)^2+nu/dx)
  pm <-  1.0 +0.5*dt* (sigma/dx)^2+ 0.5*r*dt
  pd <- -0.25    *dt*((sigma/dx)^2-nu/dx)

  firstRow <- 1
  nRows = lastRow = 2*Nj+1
  firstCol <- 1
  middleRow <- Nj+1
  nCols = lastCol = N+1

  cp <- ifelse(isCall,1,-1)

  #initialize asset prices, derivative prices
  pmp=pp=V=S=matrix(0,nrow=nRows,ncol=nCols)
  S[lastRow,lastCol] <- S0*exp(-Nj*dx)
  for (j in (lastRow-1):firstRow) {
    S[j,lastCol]=S[j+1,lastCol]*exp(dx)
  }

  #initial option value at maturity
  for (j in (firstRow):lastRow) {
    V[j,lastCol] <- max(0,cp*(S[j,lastCol]-K))
  }

  #compute derivative boundary condition
  lambdaL <- ifelse(isCall,0,(S[lastRow,lastCol]-S[lastRow-1,lastCol]))
  lambdaU <- ifelse(isCall,(S[firstRow,lastCol]-S[firstRow+1,lastCol]),0)

  #step back through lattice
  for (i in (lastCol-1):firstCol) {
    #solve system
    #substitute boundary condition at j=-Nj into j=-Nj+1
    pmp[(lastRow-1),i] <- pm+pd
    pp[(lastRow-1),i]  <- -pu*V[(lastRow-2),lastCol]-(pm-2)*V[(lastRow-1),lastCol]-pd*V[(lastRow),lastC
    #eliminate upper diagonal
    for (j in (lastRow-2):(firstRow+1)) {
        pmp[j,i] <- pm-pu*pd/pmp[j+1,i]
        pp[j,i] <- -pu*V[j-1,i+1]-(pm-2)*V[j,i+1]-pd*V[j+1,i+1]-pp[j+1,i]*pd/pmp[j+1,i]
    }
    #use boundary condition at j=Nj and equation at j=Nj-1
    V[firstRow,i] <- (pp[(firstRow+1),i]+pmp[(firstRow+1),i]
                              *lambdaU/(pu+pmp[(firstRow+1),i]))
    V[(firstRow-1),i] <- V[(firstRow),i]-lambdaU
    #back-substitution
    for (j in (firstRow+2):lastRow) {
        V[j,i] <- (pp[j,i]-pu*V[j-1,i])/pmp[j,i]
    }
```

```
    V[lastRow,i] <- V[(lastRow-1),i]-lambdaL
  }
  V[Nj+1,1]
}
N <- 1221
Nj <- 60
Crank_Nicolson_finite(T,100,1,100,0.25,0.06,0.03,N,Nj)
```

```
## [1] 10.6603
```

```
Crank_Nicolson_finite(F,100,1,100,0.25,0.06,0.03,N,Nj)
```

```
## [1] 8.144481
```

4.Calculate and report the price for European Call and Put using all the FD methods

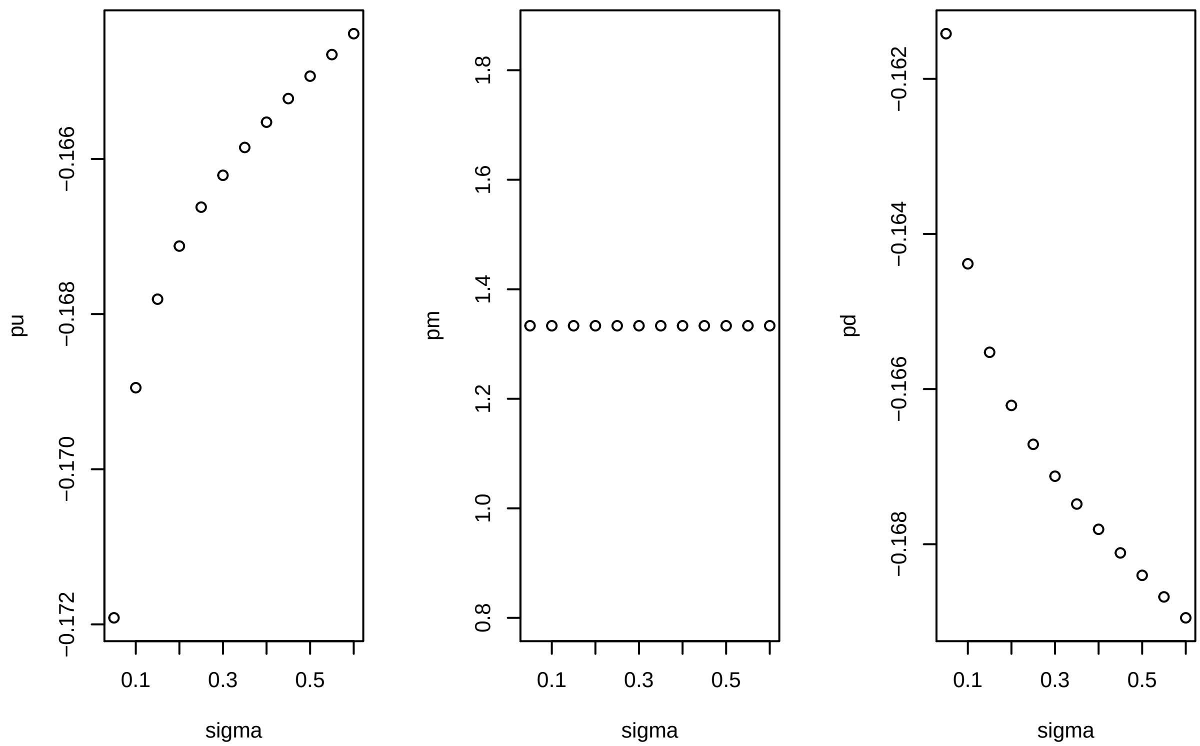
```
#Given the data
K <- 100
Tm <- 1
S0 <- 100
sigma <- 0.25
r <- 0.06
div <- 0.03
#Assuming N&Nj
N <- 1000
Nj <- 60

c1 <- Explicit_finite(T,K,Tm,S0,sigma,r,div,N,Nj)
p1 <- Explicit_finite(F,K,Tm,S0,sigma,r,div,N,Nj)
c2 <- Implicit_finite(T,K,Tm,S0,sigma,r,div,N,Nj)
p2 <- Implicit_finite(F,K,Tm,S0,sigma,r,div,N,Nj)
c3 <- Crank_Nicolson_finite(T,K,Tm,S0,sigma,r,div,N,Nj)
p3 <- Crank_Nicolson_finite(F,K,Tm,S0,sigma,r,div,N,Nj)
call <- c(c1,c2,c3)
put <- c(p1,p2,p3)
tablecp <- data.frame(call,put)
rownames(tablecp) <- c("Explicit_finite","Implicit_finite","Crank_Nicolson_finite")
tablecp
```

```
##                           call      put
## Explicit_finite       11.01115 8.142838
## Implicit_finite       10.86195 8.140935
## Crank_Nicolson_finite 10.86527 8.142252
```

5.plot on the same graph the implicit finite updating coefficients A;B;C

```
#Given the data
K <- 100
Tm <- 1
S0 <- 100
r <- 0.06
div <- 0.03
#Calculate
sigma <- seq(0.05,0.60,by=0.05)
dt <- Tm/N
dx <- sigma*sqrt(3*dt)
nu <- r-div-0.5*sigma^2
```

```r
pu <- -0.5*dt*((sigma/dx)^2+nu/dx)#A
pm <-  1.0+dt* (sigma/dx)^2+r*dt#B
pd <- -0.5*dt*((sigma/dx)^2-nu/dx)#C
par(mfrow=c(1,3))
plot(pu~sigma)
plot(pm~sigma)
plot(pd~sigma)
```




```r
#By observation, we can see as sigma goes up, pu goes up, pm remain constant
#for a certain level, pd goes down clearly. pu is concave and pd is convex.
#Also pu&pd is negative.
```

6.Estimate the number

```r
for (i in 1:10000) {
  sigma <- 0.25
  Tm <- 1
  Nj <- i
  #the stability&convergence condition
  N <- ceiling(3*((2*Nj+1)/6)^2)
  dt <- Tm/N
  dx <- sigma*sqrt(3*dt)
  error <- dx^2+Tm/N
  if(error < 0.001){
    break
  }
}
```

```
N
```

```
## [1] 1221
```

```
Nj
```

```
## [1] 60
```

7.We have to substrate result in EFD,IFD,CNFD method, and use black-scholes price to calculate the N and Nj.

```
#BSmodel
BSmodel <- function(s,t,t2,K,sigma,r,type){
d1 <- (log(s/K)+(r+0.5*sigma^2)*(t2-t))/sigma*sqrt(t2-t)
2
d2 <- d1-sigma*sqrt(t2-t)
if(type == "calls")
c <- s*pnorm(d1)-K*(exp(-r*(t2-t)))*pnorm(d2)
if(type == "puts")
c <- K*(exp(-r*(t2-t)))*pnorm(-d2)-s*pnorm(-d1)
return(c)
}
BSC <- BSmodel(100,0,1,100,0.25,0.03,"calls")
BSP <- BSmodel(100,0,1,100,0.25,0.03,"puts")
#Given the data
K <- 100
Tm <- 1
S0 <- 100
sigma <- 0.25
r <- 0.06
div <- 0.03
#call error for explicit finite
for (i in 1:100) {
  sigma <- 0.25
  Tm <- 1
  Nj <- i
  #the stability&convergence condition
  N <- ceiling(3*((2*Nj+1)/6)^2)
  dt <- Tm/N
  dx <- sigma*sqrt(3*dt)
  error <- abs(Explicit_finite(T,K,Tm,S0,sigma,r,div,N,Nj)-BSC)
  if(error < 0.001){
    break
  }
}
result1 <- paste("Explicit call:","N=",N,",Nj=",Nj,",dx=",dx,",dt=",dt)
#put error for explicit finite
for (i in 1:100) {
  sigma <- 0.25
  Tm <- 1
  Nj <- i
  #the stability&convergence condition
  N <- ceiling(3*((2*Nj+1)/6)^2)
  dt <- Tm/N
  dx <- sigma*sqrt(3*dt)
  error <- abs(Explicit_finite(F,K,Tm,S0,sigma,r,div,N,Nj)-BSP)
```

```
  if(error < 0.001){
    break
  }
}
result2 <- paste("Explicit put:","N=",N,",Nj=",Nj,",dx=",dx,",dt=",dt)
#call error for implicit finite
for (i in 1:100) {
  sigma <- 0.25
  Tm <- 1
  Nj <- i
  #the stability&convergence condition
  N <- ceiling(3*((2*Nj+1)/6)^2)
  dt <- Tm/N
  dx <- sigma*sqrt(3*dt)
  error <- abs(Implicit_finite(T,K,Tm,S0,sigma,r,div,N,Nj)-BSC)
  if(error < 0.001){
    break
  }
}
result3 <- paste("Implicit call:","N=",N,",Nj=",Nj,",dx=",dx,",dt=",dt)
#put error for implicit finite
for (i in 1:100) {
  sigma <- 0.25
  Tm <- 1
  Nj <- i
  #the stability&convergence condition
  N <- ceiling(3*((2*Nj+1)/6)^2)
  dt <- Tm/N
  dx <- sigma*sqrt(3*dt)
  error <- abs(Implicit_finite(F,K,Tm,S0,sigma,r,div,N,Nj)-BSP)
  if(error < 0.001){
    break
  }
}
result4 <- paste("Implicit put:","N=",N,",Nj=",Nj,",dx=",dx,",dt=",dt)

#call error for Crank Nicolson finite
for (i in 1:100) {
  sigma <- 0.25
  Tm <- 1
  Nj <- i
  #the stability&convergence condition
  N <- ceiling(3*((2*Nj+1)/6)^2)
  dt <- Tm/N
  dx <- sigma*sqrt(3*dt)
  error <- abs(Implicit_finite(T,K,Tm,S0,sigma,r,div,N,Nj)-BSC)
  if(error < 0.001){
    break
  }
}
result5 <- paste("Crank Nicolson call:","N=",N,",Nj=",Nj,",dx=",dx,",dt=",dt)
#put error for Crank Nicolson finite
for (i in 1:100) {
```

```
  sigma <- 0.25
  Tm <- 1
  Nj <- i
  #the stability&convergence condition
  N <- ceiling(3*((2*Nj+1)/6)^2)
  dt <- Tm/N
  dx <- sigma*sqrt(3*dt)
  error <- abs(Implicit_finite(F,K,Tm,S0,sigma,r,div,N,Nj)-BSP)
  if(error < 0.001){
    break
  }
}
result6 <- paste("Crank Nicolson put:","N=",N,",Nj=",Nj,",dx=",dx,",dt=",dt)
result1
```

```
## [1] "Explicit call: N= 3367 ,Nj= 100 ,dx= 0.00746240950950534 ,dt= 0.000297000297000297"
```

```
result2
```

```
## [1] "Explicit put: N= 3367 ,Nj= 100 ,dx= 0.00746240950950534 ,dt= 0.000297000297000297"
```

```
result3
```

```
## [1] "Implicit call: N= 3367 ,Nj= 100 ,dx= 0.00746240950950534 ,dt= 0.000297000297000297"
```

```
result4
```

```
## [1] "Implicit put: N= 3367 ,Nj= 100 ,dx= 0.00746240950950534 ,dt= 0.000297000297000297"
```

```
result5
```

```
## [1] "Crank Nicolson call: N= 3367 ,Nj= 100 ,dx= 0.00746240950950534 ,dt= 0.000297000297000297"
```

```
result6#Due to the environment of my computer, I can only iterate up to 100times
```

```
## [1] "Crank Nicolson put: N= 3367 ,Nj= 100 ,dx= 0.00746240950950534 ,dt= 0.000297000297000297"
```

8.Calculate the hedge sensitivities for the European call option

```
#Given the data
K <- 100
Tm <- 1
S0 <- 100
sigma <- 0.25
r <- 0.06
div <- 0.03
N <- 1221
Nj <- 60
isCall <- T
delta <- 0.01
#Calculate with sensitivity =0.01
Delta <- ((Explicit_finite(isCall,K,Tm,S0+delta,sigma,r,div,N,Nj))-
  (Explicit_finite(isCall,K,Tm,S0-delta,sigma,r,div,N,Nj)))/(2*delta)
Gamma <- ((Explicit_finite(isCall,K,Tm,S0+delta,sigma,r,div,N,Nj))
          -2*(Explicit_finite(isCall,K,Tm,S0,sigma,r,div,N,Nj))
          +(Explicit_finite(isCall,K,Tm,S0-delta,sigma,r,div,N,Nj)))/(delta*delta)
Theta <- ((Explicit_finite(isCall,K,Tm+delta,S0,sigma,r,div,N,Nj))-
  (Explicit_finite(isCall,K,Tm-delta,S0,sigma,r,div,N,Nj)))/(2*delta)
Vega <- ((Explicit_finite(isCall,K,Tm,S0,sigma+delta,r,div,N,Nj))-
  (Explicit_finite(isCall,K,Tm,S0,sigma-delta,r,div,N,Nj)))/(2*delta)
```

```r
#The greeks
thegreeks <- data.frame(Delta,Gamma,Theta,Vega)
thegreeks
```

```
##       Delta    Gamma    Theta     Vega
## 1 0.5791198 1.862252 5.772949 37.56374
```

Problem 2. Finite difference method applied to market data a.Download Option prices&calculate impilied volatility I choose the AAPL's options, and the expiry date is 2018-11-16,2018-12-21,2019-01-18, seperately

```r
#set the current short-term interest rate
r <- 0.022#federal rate at 2018-10-28
library(TTR)
library(quantmod)
```

```
## Loading required package: xts

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Version 0.4-0 included new data defaults. See ?getSymbols.
```

```r
Aequity=function(symbol){
A_equity <- getSymbols(symbol,src="yahoo",from="2018-10-26",to="2018-10-27",auto.assign = FALSE)
}
AAPL <- Aequity("AAPL")
```

```
## 'getSymbols' currently uses auto.assign=TRUE by default, but will
## use auto.assign=FALSE in 0.5-0. You will still be able to use
## 'loadSymbols' to automatically load data. getOption("getSymbols.env")
## and getOption("getSymbols.auto.assign") will still be checked for
## alternate defaults.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.

##
## WARNING: There have been significant changes to Yahoo Finance data.
## Please see the Warning section of '?getSymbols.yahoo' for details.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.yahoo.warning"=FALSE).
```

```r
AAPL <- AAPL$AAPL.Adjusted[1]#AAPL stock price
A_price <- AAPL
A1 <- getOptionChain("AAPL",Exp = '2018-11-16',src = "yahoo",auto.assign = FALSE)#Option1
A2 <- getOptionChain("AAPL",Exp = '2018-12-21',src = "yahoo",auto.assign = FALSE)#Option2
A3 <- getOptionChain("AAPL",Exp = '2019-01-18',src = "yahoo",auto.assign = FALSE)#Option3
TTM1 <- 19/365
TTM2 <- 54/365
TTM3 <- 82/365
# Black-Scholes Option Value
BSmodel <- function(S0, Sigma, t, K, r, optionType){
```

```r
  d1 = (log(S0/K) + (r + (Sigma^2)/2) * t)/(Sigma * sqrt(t))
  d2 = d1 - Sigma * sqrt(t)
  if(optionType == 'call'){
    call = S0 * pnorm(d1) - K * exp(-r * t) * pnorm(d2)
    return(call)
  }else if(optionType == "put"){
    put = K * exp(-r * t) * pnorm(-d2) - S0 * pnorm(-d1)
    return(put)
  }
}

#Implement the Bisection method
vrange=c(0,1)
t1=0
bisection=function(f, vrange, tol){
  while((vrange[2]-vrange[1]) >= tol){
    x=0.5*(vrange[1]+vrange[2])
if(f(x)*f(vrange[2])>0)
  {
  vrange[2]=x
}else if(f(x)*f(vrange[2])<0){
  vrange[1]=x
}
  t1=1+t1
}
return(x)
}
AM1 <- A1
AM2 <- A2
AM3 <- A3
# Using bisection and BS model
# Calculate the implied volatility for AM_option1
AMcall1=AM1$calls$Strike[1:10]
AMimpcall1=AMimpcall2=AMimpcall3=AMimput1=AMimput2=AMimput3=c()
# Call
for(i in 1:length(AMcall1)){
  AMvolc1 = function(sigma){
    BSmodel(A_price, sigma, TTM1,AMcall1[i], r, "call") - 0.5 * (AM1$calls$Bid[i] +AM1$calls$Ask[i])
  }
  AMimpcall1[i] = bisection(AMvolc1, vrange, 1e-2)
}
#Put
AMput1=AM1$puts$Strike[1:10]
for(i in 1:length(AMput1)){
  AMvolp1 = function(sigma){
    BSmodel(A_price, sigma, TTM1,AMput1[i], r, "put") - 0.5 * (AM1$puts$Bid[i] + AM1$puts$Ask[i])
  }
  AMimput1[i] = bisection(AMvolp1, vrange, 1e-2)
}

# Second Option
AMcall2=AM2$calls$Strike[1:10]
```

```r
# Call
for(i in 1:length(AMcall2)){
  AMvolc2 = function(sigma){
    BSmodel(A_price, sigma, TTM2,AMcall2[i], r, "call") - 0.5 * (AM2$calls$Bid[i] + AM2$calls$Ask[i])
  }
  AMimpcall2[i] = bisection(AMvolc2, vrange, 1e-6)
}

#Put
AMput2=AM2$puts$Strike[1:10]
for(i in 1:length(AMput2)){
  AMvolp2 = function(sigma){
    BSmodel(A_price, sigma, TTM2,AMput2[i], r, "put") - 0.5 * (AM2$puts$Bid[i] + AM2$puts$Ask[i])
  }
  AMimput2[i] = bisection(AMvolp2, vrange, 1e-6)
}

# Third Option

AMcall3=AM3$calls$Strike[1:10]

# Call
for(i in 1:length(AMcall3)){
  AMvolc3 = function(sigma){
    BSmodel(A_price, sigma, TTM3,AMcall3[i], r, "call") - 0.5 * (AM3$calls$Bid[i] + AM3$calls$Ask[i])
  }
  AMimpcall3[i] = bisection(AMvolc3, vrange, 1e-6)
}

#Put
AMput3=AM3$puts$Strike[1:10]
for(i in 1:length(AMput3)){
  AMvolp3 = function(sigma){
    BSmodel(A_price, sigma, TTM3,AMput3[i], r, "put") - 0.5 * (AM3$puts$Bid[i] + AM3$puts$Ask[i])
  }
  AMimput3[i] = bisection(AMvolp3, vrange, 1e-6)
}
result <- data.frame(AMimpcall1,AMimput1,AMimpcall2,AMimput2,AMimpcall3,AMimput3)
result
```

```
##    AMimpcall1  AMimput1   AMimpcall2  AMimput2   AMimpcall3     AMimput3
## 1   0.0078125 0.5078125 9.536743e-07 0.6738443 9.536743e-07 9.536743e-07
## 2   0.0078125 0.4921875 9.536743e-07 0.6107340 9.536743e-07 9.536743e-07
## 3   0.0078125 0.4765625 9.536743e-07 0.5955954 9.536743e-07 9.536743e-07
## 4   0.0078125 0.4609375 6.000757e-01 0.6197882 9.536743e-07 9.536743e-07
## 5   0.0078125 0.4453125 9.536743e-07 0.6157713 9.536743e-07 8.227053e-01
## 6   0.0078125 0.4296875 9.575281e-01 0.5700560 9.536743e-07 7.744722e-01
## 7   0.0078125 0.4296875 9.536743e-07 0.5591574 9.536743e-07 7.418604e-01
## 8   0.0078125 0.4296875 9.536743e-07 0.5431070 9.536743e-07 6.871042e-01
## 9   0.0078125 0.4296875 9.852800e-01 0.5209475 9.536743e-07 6.770277e-01
## 10  0.0078125 0.6640625 9.536743e-07 0.5030870 9.536743e-07 6.523314e-01
```

b.Use the Explicit, Implicit, and Crank-Nicolson Finite Difference schemes

```r
#Use the calculated implied volatility to obtain a space parameter deltax
#Call 1
sigmax <- AMimpcall1
space_parameter <- function(sigmax){
  for (i in 1:10000) {
  sigma <- sigmax
  Tm <- 1
  Nj <- i
  #the stability&convergence condition
  N <- ceiling(3*((2*Nj+1)/6)^2)
  dt <- Tm/N
  dx <- sigma*sqrt(3*dt)
  error <- dx^2+Tm/N
  if(error < 0.001){
    break
  }
  }
  return(c(N,Nj))
}
parameter <- sapply(sigmax, space_parameter)
N_Call1 <- parameter[1,]
Nj_Call1 <- parameter[2,]

#Similiarly do for put1,call2,put2,call3,put3
#Put1
sigmax <- AMimput1
parameter <- sapply(sigmax, space_parameter)
N_Put1 <- parameter[1,]
Nj_Put1 <- parameter[2,]
#call2
sigmax <- AMimpcall2
parameter <- sapply(sigmax, space_parameter)
N_Call2 <- parameter[1,]
Nj_Call2 <- parameter[2,]
#Put2
sigmax <- AMimput2
parameter <- sapply(sigmax, space_parameter)
N_Put2 <- parameter[1,]
Nj_Put2 <- parameter[2,]
#Call3
sigmax <- AMimpcall3
parameter <- sapply(sigmax, space_parameter)
N_Call3 <- parameter[1,]
Nj_Call3 <- parameter[2,]
#Put3
sigmax <- AMimput3
parameter <- sapply(sigmax, space_parameter)
N_Put3 <- parameter[1,]
Nj_Put3 <- parameter[2,]
```

```r
#Use explicit finite difference schemes
#Call1
list_Call <- cbind(isCall,K,Tm,S0,sigma,r,div,N,Nj)
EFD_Call1 <- c()
for (i in 1:10) {
isCall[i] <- T
K[i] <- AM1$calls$Strike[i]
Tm[i] <- TTM1
S0[i] <- as.numeric(A_price)
sigma[i] <- AMimpcall1[i]
r[i] <- 0.022
div[i] <- 0
N[i] <- N_Call1[i]
Nj[i] <- Nj_Call1[i]
EFD_Call1[i] <- Explicit_finite(isCall[i],K[i],Tm[i],S0[i],sigma[i],r[i],div[i],N[i],Nj[i])
}

#Put1
EFD_Put1 <- c()
for (i in 1:10) {
isCall[i] <- F
K[i] <- AM1$puts$Strike[i]
Tm[i] <- TTM1
S0[i] <- as.numeric(A_price)
sigma[i] <- AMimput1[i]
r[i] <- 0.022
div[i] <- 0
N[i] <- N_Put1[i]
Nj[i] <- Nj_Put1[i]
EFD_Put1[i] <- Explicit_finite(isCall[i],K[i],Tm[i],S0[i],sigma[i],r[i],div[i],N[i],Nj[i])
}

#Use implicit finite difference schemes
#Call1
list_Call <- cbind(isCall,K,Tm,S0,sigma,r,div,N,Nj)
IFD_Call1 <- c()
for (i in 1:10) {
isCall[i] <- T
K[i] <- AM1$calls$Strike[i]
Tm[i] <- TTM1
S0[i] <- as.numeric(A_price)
sigma[i] <- AMimpcall1[i]
r[i] <- 0.022
div[i] <- 0
N[i] <- N_Call1[i]
Nj[i] <- Nj_Call1[i]
IFD_Call1[i] <- Implicit_finite(isCall[i],K[i],Tm[i],S0[i],sigma[i],r[i],div[i],N[i],Nj[i])
}

#Put1
IFD_Put1 <- c()
for (i in 1:10) {
isCall[i] <- F
```

```r
K[i] <- AM1$puts$Strike[i]
Tm[i] <- TTM1
S0[i] <- as.numeric(A_price)
sigma[i] <- AMimput1[i]
r[i] <- 0.022
div[i] <- 0
N[i] <- N_Put1[i]
Nj[i] <- Nj_Put1[i]
IFD_Put1[i] <- Implicit_finite(isCall[i],K[i],Tm[i],S0[i],sigma[i],r[i],div[i],N[i],Nj[i])
}


#Use Crank-Nicolson finite difference schemes
#Call1
list_Call <- cbind(isCall,K,Tm,S0,sigma,r,div,N,Nj)
CNFD_Call1 <- c()
for (i in 1:10) {
isCall[i] <- T
K[i] <- AM1$calls$Strike[i]
Tm[i] <- TTM1
S0[i] <- as.numeric(A_price)
sigma[i] <- AMimpcall1[i]
r[i] <- 0.022
div[i] <- 0
N[i] <- N_Call1[i]
Nj[i] <- Nj_Call1[i]
CNFD_Call1[i] <- Crank_Nicolson_finite(isCall[i],K[i],Tm[i],S0[i],sigma[i],r[i],div[i],N[i],Nj[i])
}

#Put1
CNFD_Put1 <- c()
for (i in 1:10) {
isCall[i] <- F
K[i] <- AM1$puts$Strike[i]
Tm[i] <- TTM1
S0[i] <- as.numeric(A_price)
sigma[i] <- AMimput1[i]
r[i] <- 0.022
div[i] <- 0
N[i] <- N_Put1[i]
Nj[i] <- Nj_Put1[i]
CNFD_Put1[i] <- Crank_Nicolson_finite(isCall[i],K[i],Tm[i],S0[i],sigma[i],r[i],div[i],N[i],Nj[i])
}

table_result <- cbind(EFD_Call1,EFD_Put1,IFD_Call1,IFD_Put1,CNFD_Call1,CNFD_Put1)
table_result#This is result I use EFD,IFD,CNFD calculating option price for 1 month maturity for AAPL
```

```
##       EFD_Call1  EFD_Put1 IFD_Call1  IFD_Put1 CNFD_Call1 CNFD_Put1
##  [1,]  213.8029  2.901117  209.6222  2.900857   209.6544  2.901052
##  [2,]  211.3058  3.975941  207.1737  3.975324   207.2055  3.975700
##  [3,]  208.8086  5.360265  204.7251  5.359323   204.7566  5.359863
##  [4,]  206.3115  7.107362  202.2766  7.106195   202.3077  7.106849
##  [5,]  201.3172  9.266641  197.3796  9.265423   197.4100  9.266104
##  [6,]  193.8258 11.871239  190.0340 11.870182   190.0633 11.870784
```

```
## [7,] 191.3287 15.214233 187.5855 15.213497 187.6144 15.213940
## [8,] 166.3573 18.940123 163.1003 18.939797 163.1254 18.940036
## [9,] 146.3802 22.990156 143.5122 22.990244 143.5343 22.990277
## [10,] 141.3859 68.892393 138.6151 68.892947 138.6365 68.892589
```

c.Calculating corresponding Greeks using EFD

```
#For Call
#Calculate with sensitivity =0.01
delta1 <- matrix(rep(0.01,10),nrow=10,ncol=1)
TTM1_10 <- matrix(rep(TTM1,10),nrow=10,ncol=1)
S0_10 <- matrix(rep(as.numeric(A_price),10),nrow=10,ncol=1)
r_10 <- matrix(rep(0.022,10),nrow=10,ncol=1)
div_10 <- matrix(rep(0,10),nrow=10,ncol=1)
isCall_10 <- matrix(rep(T,10),nrow=10,ncol=1)
#Delta
delta <- matrix(nrow=10,ncol=1)
isCall <- matrix(nrow=10,ncol=1)
K <- matrix(nrow=10,ncol=1)
Tm <- matrix(nrow=10,ncol=1)
S0 <- matrix(nrow=10,ncol=1)
sigma <- matrix(nrow=10,ncol=1)
r <- matrix(nrow=10,ncol=1)
div <- matrix(nrow=10,ncol=1)
N <- matrix(nrow=10,ncol=1)
Nj <- matrix(nrow=10,ncol=1)
EFD_Call1_Dealta <- matrix(nrow=10,ncol=1)
for (i in 1:10) {
delta[i] <- delta1[i]
isCall[i] <- isCall_10[i]
K[i] <- AM1$calls$Strike[i]
Tm[i] <- TTM1_10[i]
S0[i] <- S0_10[i]
sigma[i] <- AMimpcall1[i]
r[i] <- r_10[i]
div[i] <- div_10[i]
N[i] <- N_Call1[i]
Nj[i] <- Nj_Call1[i]
}
for (i in 1:10) {
  EFD_Call1_Dealta[i] <- ((Explicit_finite(isCall[i],K[i],Tm[i],S0[i]+delta[i],sigma[i],r[i],div[i],N[i]
  (Explicit_finite(isCall[i],K[i],Tm[i],S0[i]-delta[i],sigma[i],r[i],div[i],N[i],Nj[i])))/(2*delta[i])
}
#Gamma
EFD_Call1_Gamma <- matrix(nrow=10,ncol=1)
for (i in 1:10) {
  EFD_Call1_Gamma[i] <- ((Explicit_finite(isCall[i],K[i],Tm[i],S0[i]+delta[i],sigma[i],r[i],div[i],N[i]
          -2*(Explicit_finite(isCall[i],K[i],Tm[i],S0[i],sigma[i],r[i],div[i],N[i],Nj[i]))
          +(Explicit_finite(isCall[i],K[i],Tm[i],S0[i]-delta[i],sigma[i],r[i],div[i],N[i],Nj[i])))/(del
}
#Theta
EFD_Call1_Theta <- matrix(nrow=10,ncol=1)
for (i in 1:10) {
  EFD_Call1_Theta[i] <- ((Explicit_finite(isCall[i],K[i],Tm[i]+delta[i],S0[i],sigma[i],r[i],div[i],N[i]
  (Explicit_finite(isCall[i],K[i],Tm[i]-delta[i],S0[i],sigma[i],r[i],div[i],N[i],Nj[i])))/(2*delta[i])
```

```
}
#Vega
EFD_Call1_Vega <- matrix(nrow=10,ncol=1)
for (i in 1:10) {
  EFD_Call1_Vega[i] <- ((Explicit_finite(isCall[i],K[i],Tm[i],S0[i],sigma[i]+delta[i],r[i],div[i],N[i],N
  (Explicit_finite(isCall[i],K[i],Tm[i],S0[i],sigma[i]-delta[i],r[i],div[i],N[i],Nj[i])))/(2*delta[i])
}

##----------
#For put
#Calculate with sensitivity =0.01
delta1 <- matrix(rep(0.01,10),nrow=10,ncol=1)
TTM1_10 <- matrix(rep(TTM1,10),nrow=10,ncol=1)
S0_10 <- matrix(rep(as.numeric(A_price),10),nrow=10,ncol=1)
r_10 <- matrix(rep(0.022,10),nrow=10,ncol=1)
div_10 <- matrix(rep(0,10),nrow=10,ncol=1)
isCall_10 <- matrix(rep(F,10),nrow=10,ncol=1)
#Delta
delta <- matrix(nrow=10,ncol=1)
isCall <- matrix(nrow=10,ncol=1)
K <- matrix(nrow=10,ncol=1)
Tm <- matrix(nrow=10,ncol=1)
S0 <- matrix(nrow=10,ncol=1)
sigma <- matrix(nrow=10,ncol=1)
r <- matrix(nrow=10,ncol=1)
div <- matrix(nrow=10,ncol=1)
N <- matrix(nrow=10,ncol=1)
Nj <- matrix(nrow=10,ncol=1)

for (i in 1:10) {
delta[i] <- delta1[i]
isCall[i] <- isCall_10[i]
K[i] <- AM1$puts$Strike[i]
Tm[i] <- TTM1_10[i]
S0[i] <- S0_10[i]
sigma[i] <- AMimput1[i]
r[i] <- r_10[i]
div[i] <- div_10[i]
N[i] <- N_Put1[i]
Nj[i] <- Nj_Put1[i]
}
#Delta
EFD_Put1_Dealta <- matrix(nrow=10,ncol=1)
for (i in 1:10) {
  EFD_Put1_Dealta[i] <- ((Explicit_finite(isCall[i],K[i],Tm[i],S0[i]+delta[i],sigma[i],r[i],div[i],N[i]
  (Explicit_finite(isCall[i],K[i],Tm[i],S0[i]-delta[i],sigma[i],r[i],div[i],N[i],Nj[i])))/(2*delta[i])
}
#Gamma
EFD_Put1_Gamma <- matrix(nrow=10,ncol=1)
for (i in 1:10) {
  EFD_Put1_Gamma[i] <- ((Explicit_finite(isCall[i],K[i],Tm[i],S0[i]+delta[i],sigma[i],r[i],div[i],N[i],N
          -2*(Explicit_finite(isCall[i],K[i],Tm[i],S0[i],sigma[i],r[i],div[i],N[i],Nj[i]))
          +(Explicit_finite(isCall[i],K[i],Tm[i],S0[i]-delta[i],sigma[i],r[i],div[i],N[i],Nj[i])))/(del
```

```r
}
#Theta
EFD_Put1_Theta <- matrix(nrow=10,ncol=1)
for (i in 1:10) {
  EFD_Put1_Theta[i] <- ((Explicit_finite(isCall[i],K[i],Tm[i]+delta[i],S0[i],sigma[i],r[i],div[i],N[i],N
  (Explicit_finite(isCall[i],K[i],Tm[i]-delta[i],S0[i],sigma[i],r[i],div[i],N[i],Nj[i])))/(2*delta[i])
}
#Vega
EFD_Put1_Vega <- matrix(nrow=10,ncol=1)
for (i in 1:10) {
  EFD_Put1_Vega[i] <- ((Explicit_finite(isCall[i],K[i],Tm[i],S0[i],sigma[i]+delta[i],r[i],div[i],N[i],Nj
  (Explicit_finite(isCall[i],K[i],Tm[i],S0[i],sigma[i]-delta[i],r[i],div[i],N[i],Nj[i])))/(2*delta[i])
}


#The greeks
EFDthegreeks <- data.frame(EFD_Call1_Dealta,EFD_Put1_Dealta,EFD_Call1_Gamma,EFD_Put1_Gamma,
                           EFD_Call1_Theta,EFD_Put1_Theta,EFD_Call1_Vega,EFD_Put1_Vega)
EFDthegreeks
```

```
##    EFD_Call1_Dealta EFD_Put1_Dealta EFD_Call1_Gamma EFD_Put1_Gamma
## 1                 1      -0.1938659   -3.126388e-09  -1.554312e-10
## 2                 1      -0.2526299    1.989520e-09  -1.199041e-10
## 3                 1      -0.3215295    1.136868e-09  -6.306067e-10
## 4                 1      -0.4158924   -3.126388e-09  -3.907985e-10
## 5                 1      -0.5010605    1.136868e-09  -1.012523e-09
## 6                 1      -0.5885851    3.979039e-09   4.618528e-10
## 7                 1      -0.6710587    1.136868e-09   5.506706e-10
## 8                 1      -0.7459407    2.557954e-09   6.039613e-10
## 9                 1      -0.8107606    2.273737e-09   6.394885e-10
## 10                1      -0.9582773    2.842171e-09  -9.947598e-10
##    EFD_Call1_Theta EFD_Put1_Theta EFD_Call1_Vega EFD_Put1_Vega
## 1       0.05442406       65.73867       1.077301     13.653755
## 2       0.10936117       73.93713       1.077301     15.846306
## 3       0.16429828       80.19709       1.077301     17.764884
## 4       0.21923539       83.13260       1.077301     19.187951
## 5       0.32910961       81.99054       1.077301     19.676660
## 6       0.49392095       76.34069       1.077301     19.206191
## 7       0.54885806       69.93980       1.077301     17.833024
## 8       1.09822917       60.68500       1.077301     15.688473
## 9       1.53772606       49.96883       1.077301     13.210448
## 10      1.64760029       21.43524       1.077301      4.335384
```

d.Create a table

```r
#For call
#Time to maturity T
TTM1_10 <- matrix(rep(TTM1,10),nrow=10,ncol=1)
Tm_Call1 <- matrix(nrow=10,ncol=1)
for (i in 1:10) {
  Tm_Call1[i] <- TTM1_10[i]
}
#Strike price
K_Call1 <- matrix(nrow=10,ncol=1)
for (i in 1:10) {
```

```
  K_Call1[i] <- AM1$calls$Strike[i]
}
#Type of the option
Type_Call1 <- matrix("Call",nrow=10,ncol=1)
#Ask price A
A_Call1 <- matrix(nrow=10,ncol=1)
for (i in 1:10) {
  A_Call1[i] <- AM1$calls$Ask[i]
}
#Bid price B
B_Call1 <- matrix(nrow=10,ncol=1)
for (i in 1:10) {
  B_Call1[i] <- AM1$calls$Bid[i]
}
#Market price Cm
Cm_Call1 <- matrix(nrow=10,ncol=1)
for (i in 1:10) {
  Cm_Call1[i] <- (A_Call1[i]+B_Call1[i])/2
}
#Implied volatility
sigma_BSMimp_Call1 <- matrix(nrow=10,ncol=1)
for (i in 1:10) {
  sigma_BSMimp_Call1[i] <- AMimpcall1[i]
}
#option price calculated with EFD, IFD, and CNFD
#EFD_Call1,EFD_Put1,IFD_Call1,IFD_Put1,CNFD_Call1,CNFD_Put1

#For put
#Time to maturity T
TTM1_10 <- matrix(rep(TTM1,10),nrow=10,ncol=1)
Tm_Put1 <- matrix(nrow=10,ncol=1)
for (i in 1:10) {
  Tm_Put1[i] <- TTM1_10[i]
}
#Strike price
K_Put1 <- matrix(nrow=10,ncol=1)
for (i in 1:10) {
  K_Put1[i] <- AM1$puts$Strike[i]
}
#Type of the option
Type_Put1 <- matrix("Put",nrow=10,ncol=1)
#Ask price A
A_Put1 <- matrix(nrow=10,ncol=1)
for (i in 1:10) {
  A_Put1[i] <- AM1$puts$Ask[i]
}
#Bid price B
B_Put1 <- matrix(nrow=10,ncol=1)
for (i in 1:10) {
  B_Put1[i] <- AM1$puts$Bid[i]
}
#Market price Cm
Cm_Put1 <- matrix(nrow=10,ncol=1)
```

```r
for (i in 1:10) {
  Cm_Put1[i] <- (A_Put1[i]+B_Put1[i])/2
}
#Implied volatility
sigma_BSMimp_Put1 <- matrix(nrow=10,ncol=1)
for (i in 1:10) {
  sigma_BSMimp_Put1[i] <- AMimput1[i]
}

Result_table_Call1 <- cbind(Tm_Call1,K_Call1,Type_Call1,A_Call1,B_Call1,Cm_Call1,sigma_BSMimp_Call1,EFD_
colnames(Result_table_Call1) <- c("Time to maturity T","Strike price","Type of the option","Ask price A
                                  "Implied volatility ","EFD_Call1","IFD_Call1","CNFD_Call1 ")

Result_table_Put1 <- cbind(Tm_Put1,K_Put1,Type_Put1,A_Put1,B_Put1,Cm_Put1,sigma_BSMimp_Put1,EFD_Put1,IF
colnames(Result_table_Put1) <- c("Time to maturity T","Strike price","Type of the option","Ask price A"
                                  "Implied volatility ","EFD_Put1","IFD_Put1","CNFD_Put1 ")
Result_table_Call1
```

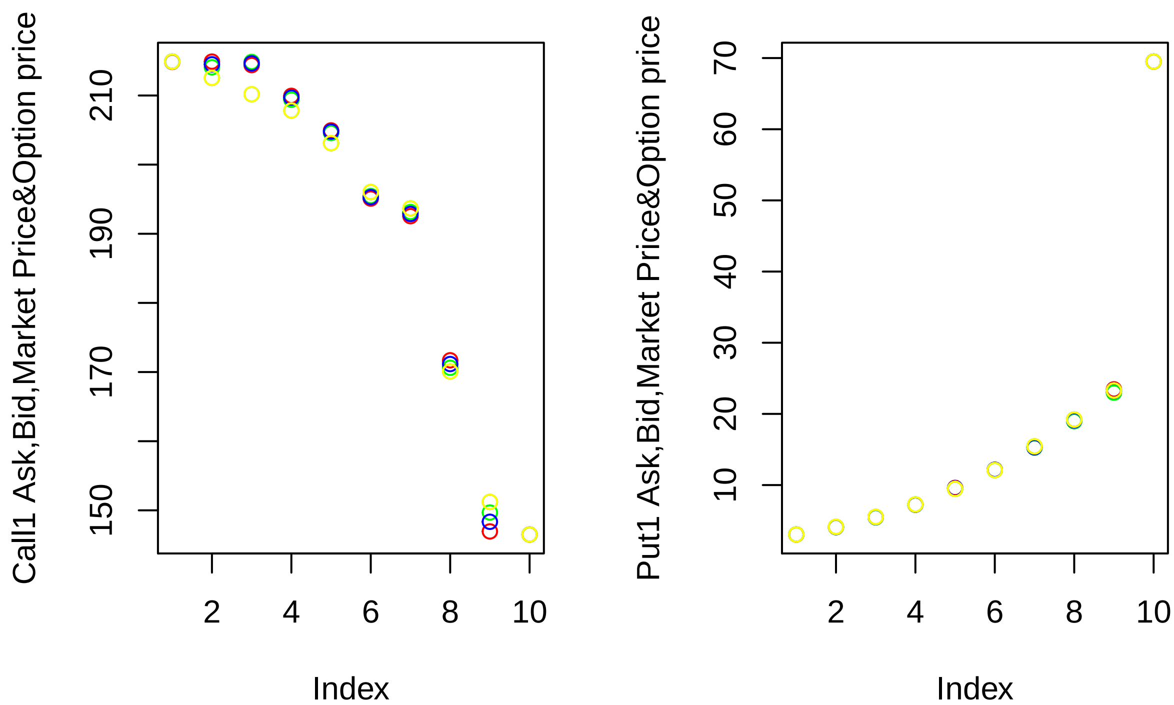
```
##       Time to maturity T   Strike price Type of the option Ask price A
##  [1,] "0.0520547945205479" "2.5"        "Call"             "214.85"
##  [2,] "0.0520547945205479" "5"          "Call"             "214.9"
##  [3,] "0.0520547945205479" "7.5"        "Call"             "214.4"
##  [4,] "0.0520547945205479" "10"         "Call"             "209.95"
##  [5,] "0.0520547945205479" "15"         "Call"             "204.95"
##  [6,] "0.0520547945205479" "22.5"       "Call"             "195.1"
##  [7,] "0.0520547945205479" "25"         "Call"             "192.55"
##  [8,] "0.0520547945205479" "50"         "Call"             "171.7"
##  [9,] "0.0520547945205479" "70"         "Call"             "146.95"
## [10,] "0.0520547945205479" "75"         "Call"             "146.5"
##       Bid price B Market price Cm Implied volatility  EFD_Call1
##  [1,] "213.1"     "213.975"       "0.0078125"         "213.802902299362"
##  [2,] "212.25"    "213.575"       "0.0078125"         "211.305763675911"
##  [3,] "213.05"    "213.725"       "0.0078125"         "208.80862505246"
##  [4,] "207.4"     "208.675"       "0.0078125"         "206.311486429009"
##  [5,] "202.4"     "203.675"       "0.0078125"         "201.317209182107"
##  [6,] "192.95"    "194.025"       "0.0078125"         "193.825793311754"
##  [7,] "190.55"    "191.55"        "0.0078125"         "191.328654688303"
##  [8,] "167.2"     "169.45"        "0.0078125"         "166.357268453793"
##  [9,] "145.5"     "146.225"       "0.0078125"         "146.380159466186"
## [10,] "142.2"     "144.35"        "0.0078125"         "141.385882219284"
##       IFD_Call1          CNFD_Call1
##  [1,] "209.622177481867" "209.654422964734"
##  [2,] "207.173658425238" "207.205528950603"
##  [3,] "204.725139368609" "204.756634936472"
##  [4,] "202.27662031198"  "202.307740922342"
##  [5,] "197.379582198722" "197.40995289408"
##  [6,] "190.034025028835" "190.063270851688"
##  [7,] "187.585505972207" "187.614376837557"
##  [8,] "163.100315405917" "163.12543669625"
##  [9,] "143.512162952886" "143.534284583204"
## [10,] "138.615124839628" "138.636496554942"
```

```r
Result_table_Put1
```

```
##       Time to maturity T   Strike price Type of the option Ask price A
##   [1,] "0.0520547945205479" "197.5"      "Put"              "3.05"
##   [2,] "0.0520547945205479" "202.5"      "Put"              "4.1"
##   [3,] "0.0520547945205479" "207.5"      "Put"              "5.5"
##   [4,] "0.0520547945205479" "212.5"      "Put"              "7.25"
##   [5,] "0.0520547945205479" "217.5"      "Put"              "9.6"
##   [6,] "0.0520547945205479" "222.5"      "Put"              "12.15"
##   [7,] "0.0520547945205479" "227.5"      "Put"              "15.35"
##   [8,] "0.0520547945205479" "232.5"      "Put"              "19.2"
##   [9,] "0.0520547945205479" "237.5"      "Put"              "23.45"
## [10,] "0.0520547945205479" "285"        "Put"              "69.5"
##       Bid price B Market price Cm Implied volatility  EFD_Put1
##   [1,] "2.87"      "2.96"          "0.5078125"         "2.90111687290975"
##   [2,] "3.9"       "4"             "0.4921875"         "3.97594054830765"
##   [3,] "5.25"      "5.375"         "0.4765625"         "5.36026462664208"
##   [4,] "7"         "7.125"         "0.4609375"         "7.10736181339746"
##   [5,] "9.2"       "9.4"           "0.4453125"         "9.26664052771537"
##   [6,] "11.8"      "11.975"        "0.4296875"         "11.8712394274354"
##   [7,] "14.9"      "15.125"        "0.4296875"         "15.2142332830367"
##   [8,] "18.55"     "18.875"        "0.4296875"         "18.9401225650712"
##   [9,] "22.5"      "22.975"        "0.4296875"         "22.9901563827111"
## [10,] "68.3"      "68.9"          "0.6640625"         "68.8923931117238"
##       IFD_Put1           CNFD_Put1
##   [1,] "2.90085673613569" "2.90105227115204"
##   [2,] "3.97532445558426" "3.97570008610227"
##   [3,] "5.35932254765075" "5.35986286416852"
##   [4,] "7.10619528506156" "7.10684924003307"
##   [5,] "9.26542291422515" "9.26610404120178"
##   [6,] "11.8701817039201" "11.8707842619188"
##   [7,] "15.2134967227365" "15.213940168827"
##   [8,] "18.9397972561638" "18.9400362945573"
##   [9,] "22.9902442257736" "22.9902771469607"
## [10,] "68.8929474786085" "68.8925887856023"
```

```r
#Figure for Call
par(mfrow=c(1,2))
plot(A_Call1,col="red",ylab = "Call1 Ask,Bid,Market Price&Option price")
par(new = TRUE)
plot(B_Call1,col="green",axes = FALSE,xlab = "", ylab = "")
par(new = TRUE)
plot(Cm_Call1,col="blue",axes = FALSE,xlab = "", ylab = "")
par(new = TRUE)
plot(EFD_Call1,col="cyan",axes = FALSE,xlab = "", ylab = "")
par(new = TRUE)
plot(IFD_Call1,col="gray",axes = FALSE,xlab = "", ylab = "")
par(new = TRUE)
plot(CNFD_Call1,col="yellow",axes = FALSE,xlab = "", ylab = "")
#----
#Figure for put
plot(A_Put1,col="red",ylab = "Put1 Ask,Bid,Market Price&Option price")
par(new = TRUE)
plot(B_Put1,col="green",axes = FALSE,xlab = "", ylab = "")
```

```r
par(new = TRUE)
plot(Cm_Put1,col="blue",axes = FALSE,xlab = "", ylab = "")
par(new = TRUE)
plot(EFD_Put1,col="cyan",axes = FALSE,xlab = "", ylab = "")
par(new = TRUE)
plot(IFD_Put1,col="gray",axes = FALSE,xlab = "", ylab = "")
par(new = TRUE)
plot(CNFD_Put1,col="yellow",axes = FALSE,xlab = "", ylab = "")
```




```r
#We can observe that the stock price is actually fallen between bid&ask spread
#And for AAPL stock, call option is close to stock price and put option is fall below stock price
```

Problem 3. 1.Discretize the derivatives Please see attached files with hand-written equation

2.What do you observe about the process of updating coefficients? For the updating coefficients, We can observe that the value of S cannot be too large. Otherwise pm could be negative,large number. Consider the three equation pu,pm&pd, we notice that it is similiar to the tirnomial tree model with Black-scholes equation. Also we can approximate sum(p)=1-r*delat t as exp(-r*delat t), with discount factor.

3.Implement the scheme

```r
#Parameter
isCall <- F #Put option
S0 <- 100
K <- 100
Tm <- 1
sigma <- sqrt(0.4*S0^1.5)
N <- 2000
```

```r
Nj <- 80

EFD_scheme <- function(isCall,K,Tm,S0,sigma,N,Nj){
  #precompute constants
  dt <- Tm/N
  dx <- sigma*sqrt(3*dt)
  edx <- exp(dx)
  pu <- dt*(0.2*S0^1.5/dx^2+sin(S0)/dx)
  pm <- 1-dt*(0.4*S0^1.5/dx^2)-r*dt
  pd <- dt*(0.2*S0^1.5/dx^2-sin(S0)/dx)
  #initialize asset prices at maturity
  V=S=matrix(0,nrow=2*Nj+1,ncol=N+1)
  cp <- ifelse(isCall,1,-1)
  S[2*Nj+1,N+1] <- S0*exp(-Nj*dx)
  for (i in (2*Nj):1) {
    S[i,N+1]=S[i+1,N+1]*exp(dx)
  }
  #initial option value at maturity
  for (i in (2*Nj+1):1) {
    V[i,N+1] <- max(0,cp*(S[i,N+1]-K))
  }
  #step back through lattice
  for (j in N:1) {
    for (i in (2*Nj):2) {
      V[i,j] <- pu*V[i-1,j+1]+pm*V[i,j+1]+pd*V[i+1,j+1]
    }
    #boundary conditions
    stockTerm <- ifelse(isCall,S[1,N+1]-S[2,N+1],
                        S[2*Nj,N+1]-S[2*Nj+1,N+1])
    V[2*Nj+1,j] <- V[2*Nj,j]+ifelse(isCall,0,stockTerm)
    V[1,j] <- V[2,j]+ifelse(isCall,stockTerm,0)
  }
  V[Nj+1,1]
}
```