H3

Homework 3

Question(a)

```
EFD.Eu <- function(type,K,T,S,sig,r,div,N,Nj,dx){
 #precompute constants
 dt = T/N
 nu = r - div - 0.5 * sig^2
 edx = exp(-dx)
 pu = 0.5*dt* ((sig/dx)^2 + nu/dx)
 pm = 1.0 - dt*(sig/dx)^2 - r*dt
 pd = 0.5*dt* ((sig/dx)^2 - nu/dx)

 #initialise asset prices at marturity
 St = array(0,dim=c(2*Nj+1))
 St[1] = S * exp(Nj*dx)
 for(j in 2:(2*Nj+1)){
   St[j] = St[j-1]*edx
 }

 #initialise option values to marturity
 C = array(0,dim = c(2,2*Nj+1))
 for(j in 1:(2*Nj+1)){
   if(type == "C")
    C[1,j] = max(0, St[j] - K)
   if(type == "P")
    C[1,j] = max(0, K - St[j])
 }

 #step back at though latice
 for(i in N:1){
   for(j in 2:(2*Nj)){
     C[2,j] = pu*C[1,j-1]+pm*C[1,j]+pd*C[1,j+1]
   }

   #boundary conditions
   if(type == "C"){
     C[2,1] = C[2,2] + (St[1]-St[2])
     C[2,2*Nj+1] = C[2,2*Nj]
   }
   if(type == "P"){
     C[2,1] = C[2,2]
     C[2,2*Nj+1] = C[2,2*Nj] + (St[2*Nj+1]-St[2*Nj])
   }
   for(j in 1:(2*Nj+1)){
    C[1,j] <- C[2,j]
   }
 }
```

```
 C[1,Nj+1]
}
```

Question(b)

```
IFD.Eu <- function(type,K,T,S,sig,r,div,N,Nj,dx){
 #precompute constants
 dt = T/N
 nu = r - div - 0.5 * sig^2
 edx=exp(-dx)
 pu = -0.5*dt* ((sig/dx)^2 + nu/dx)
 pm = 1.0 + dt*(sig/dx)^2 + r*dt
 pd = -0.5*dt* ((sig/dx)^2 - nu/dx)

 #initialise asset prices at marturity
 St = array(0,dim=c(2*Nj+1))
 St[1] = S * exp(Nj*dx)
 for(j in 2:(2*Nj+1)){
   St[j] = St[j-1]*edx
 }

 #initialise option values at marturity
 C = array(0,dim=c(2,2*Nj+1))
 for(j in 1:(2*Nj+1)){
   if(type == "C")
    C[1,j] = max(0, St[j] -K)
   if(type == "P")
    C[1,j] = max(0, K - St[j])
 }

 #compute derivative boundary condition
 if(type == "C"){
   lambda.U = St[1]-St[2]
   lambda.L = 0.0
 }
 if(type == "P"){
   lambda.L = -1*(St[2*Nj]-St[2*Nj+1])
   lambda.U = 0.0
 }

 #step back at though latice
 for(i in N:1){
  #substitute boundary condition at j=-Nj into j=-Nj+1
  pmp=pp=NULL
  pmp[2*Nj] = pm +pd
  pp[2*Nj] = C[1,2*Nj] + pd*lambda.L

  #eliminate upper diagonal
  for(j in (2*Nj-1):2){
    pmp[j] = pm-pu*pd/pmp[j+1]
    pp[j] = C[1,j]-pp[j+1]*pd/pmp[j+1]
  }
```

```
  #use boundary condition at j=Nj and equation at j=Nj-1
  C[2,1] = (pp[2]+pmp[2]*lambda.U)/(pu+pmp[2])
  C[2,2] = C[2,1]-lambda.U

  #back sutiatution
  for(j in 3:(2*Nj+1)){
    C[2,j] = (pp[j]-pu*C[2,j-1])/pmp[j]
  }
  for(j in 1:(2*Nj+1)){
    C[1,j] <- C[2,j]

    if(is.na(C[2,j])){
      C[2,j] <- C[1,j]
    }
  }
 }
 C[1,Nj+1]
}
```

Question(c)

```
N.compute1 <- function(T,sig){
  N = 500
  Nj = floor((6*sqrt(N/3)-1)/2)
  dt = T/N
  dx = 6 * sig * sqrt(T)/(2*Nj+1)
  ebs <- dx^2 + dt
  while(abs(ebs)>0.001){
    N <- N+1
    Nj <- floor((6*sqrt(N/3)-1)/2)
    dt = T/N
    dx = 6 * sig * sqrt(T)/(2*Nj+1)
    ebs <- dx^2 + dt
  }
  cat("N=",N,"Nj=",Nj,"dx=",dx)
}
N.compute1(T=1,sig=0.25)
```

```
## N= 1189 Nj= 59 dx= 0.01260504
```

Question(d)

```
efd.c <- EFD.Eu(type="C",K=100,T=1,S=100,sig=0.25,r=0.06,div=0.03,
                N=1189,Nj=59,dx=0.01260504)
efd.p <- EFD.Eu(type="P",K=100,T=1,S=100,sig=0.25,r=0.06,div=0.03,
                N=1189,Nj=59,dx=0.01260504)
ifd.c <- IFD.Eu(type="C",K=100,T=1,S=100,sig=0.25,r=0.06,div=0.03,
                N=1189,Nj=59,dx=0.01260504)
ifd.p <- IFD.Eu(type="P",K=100,T=1,S=100,sig=0.25,r=0.06,div=0.03,
```

```
                N=1189,Nj=59,dx=0.01260504)
cat("EFD call option price = ",efd.c, "IFD call option price = ",ifd.c)
```

```
## EFD call option price =  11.01157 IFD call option price =  11.00915
```

```
cat("EFD put option price = ",efd.p, "IFD put option price = ",ifd.p)
```

```
## EFD put option price =  8.122201 IFD put option price =  8.14097
```

Question(e)

```
# Function to calculate opntion price by BS formular
BS <- function(type, S0, K, tau, r, sigma,div){

  d1 <- (log(S0/K)+(r-div+sigma^2/2)*tau) / (sigma*sqrt(tau))
  d2 <- d1 - sigma*sqrt(tau)
  if(type=="C"){
    Price <- S0*exp(-div*tau)*pnorm(d1) - K*exp(-r*tau)*pnorm(d2)
  }
  if(type=="P"){
    Price <- K*exp(-r*tau)*pnorm(-d2) - S0*exp(-div*tau)*pnorm(-d1)
  }
  return(Price)
}

N.compute2EFD <- function(type,K,T,S,sig,r,div){

  theoretical.value <- BS(type, S, K, T, r, sig,div)
  N=1700
  Nj <- floor((6*sqrt(N/3)-1)/2)
  dx = 6 * sig * sqrt(T)/(2*Nj+1)
  ebs <- theoretical.value - EFD.Eu(type,K,T,S,sig,r,div,N,Nj,dx)
  while(abs(ebs)>0.001){
    N <- N+1
    Nj <- floor((6*sqrt(N/3)-1)/2)
    dx = 6 * sig * sqrt(T)/(2*Nj+1)
    ebs <- theoretical.value - EFD.Eu(type,K,T,S,sig,r,div,N,Nj,dx)
  }
  dt = T/N
  cat("EFD method: N=",N,"Nj=",Nj,"dx=",dx,"dt=",dt)
}

N.compute2IFD <- function(type,K,T,S,sig,r,div){
  theoretical.value <- BS(type, S, K, T, r, sig,div)
  N=4213
  Nj <- floor((6*sqrt(N/3)-1)/2)
  dx = 6 * sig * sqrt(T)/(2*Nj+1)
  ebs <- theoretical.value - IFD.Eu(type,K,T,S,sig,r,div,N,Nj,dx)
  while(abs(ebs)>0.001){
    N <- N+1
    Nj <- floor((6*sqrt(N/3)-1)/2)
    dx = 6 * sig * sqrt(T)/(2*Nj+1)
    ebs <- theoretical.value - IFD.Eu(type,K,T,S,sig,r,div,N,Nj,dx)
```

```
  }
  dt = T/N
  cat("IFD method: N=",N,"Nj=",Nj,"dx=",dx,"dt=",dt)
}
cat("Call ",N.compute2EFD(type="C",K=100,T=1,S=100,sig=0.25,r=0.06,div=0.03))
```

```
## EFD method: N= 1705 Nj= 71 dx= 0.01048951 dt= 0.0005865103Call
```

```
#cat("Put ",N.compute2EFD(type="P",K=100,T=1,S=100,sig=0.25,r=0.06,div=0.03))
cat("Call ",N.compute2IFD(type="C",K=100,T=1,S=100,sig=0.25,r=0.06,div=0.03))
```

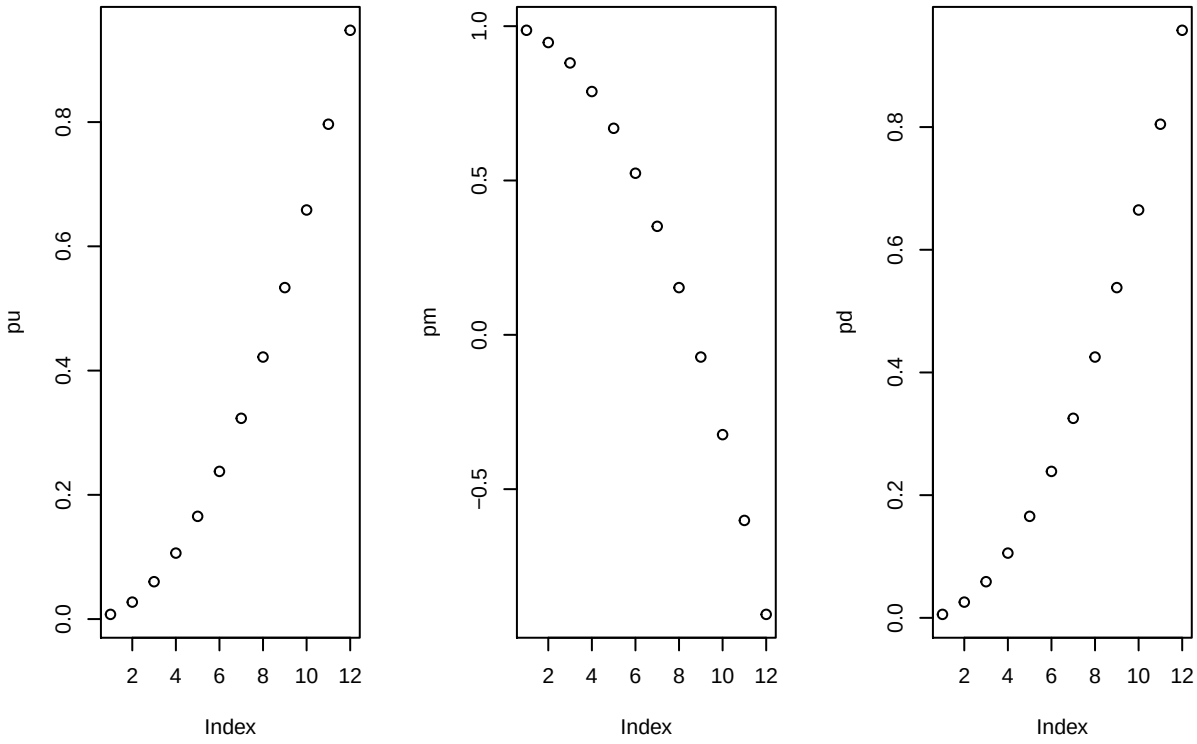
```
## IFD method: N= 4214 Nj= 111 dx= 0.006726457 dt= 0.0002373042Call
```

```
#cat("Put ",N.compute2IFD(type="P",K=100,T=1,S=100,sig=0.25,r=0.06,div=0.03))
```

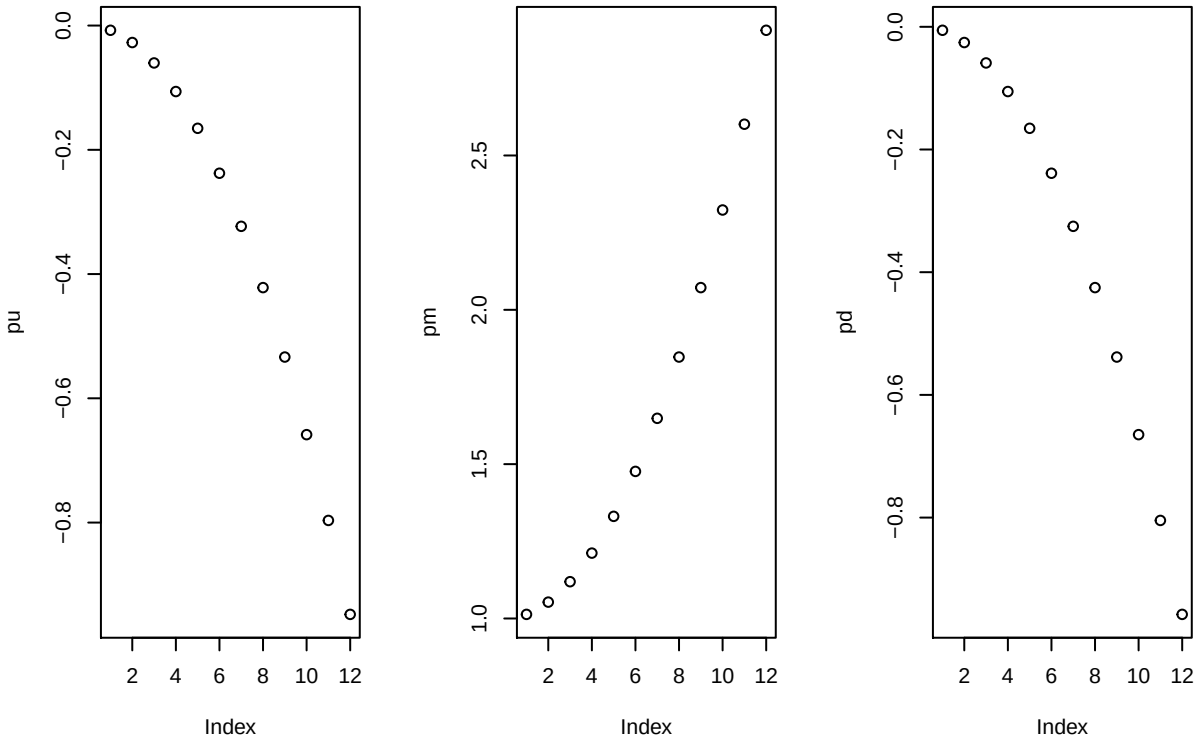
comments: from the result, we can see that EFD method converges more faster than IFD method. I make put as a comment, because it runs too slow.

Question(f)

```
plot.prob <- function(method,T,r,div,N,dx){
 dt = T/N
 pu=pm=pd=c()
 if(method == "EFD"){
   for(sig in seq(0.05,0.6,0.05)){
     nu = r - div - 0.5 * sig^2
     pu = c(pu,0.5*dt* ((sig/dx)^2 + nu/dx))
     pm = c(pm,(1.0 - dt*(sig/dx)^2 - r*dt))
     pd = c(pd,0.5*dt* ((sig/dx)^2 - nu/dx))
   }
 }
if(method == "IFD"){
   for(sig in seq(0.05,0.6,0.05)){
     nu = r - div - 0.5 * sig^2
     pu = c(pu,-0.5*dt* ((sig/dx)^2 + nu/dx))
     pm = c(pm,(1.0 + dt*(sig/dx)^2 + r*dt))
     pd = c(pd,-0.5*dt* ((sig/dx)^2 - nu/dx))
   }
 }
 par(mfrow=c(1,3))
 plot(pu)
 plot(pm)
 plot(pd)
}
plot.prob(method="EFD",T=1,r=0.06,div=0.03,N=1189,dx=0.01260504)
```




```
plot.prob(method="IFD",T=1,r=0.06,div=0.03,N=1189,dx=0.01260504)
```



comment: In EFD method, with the bigger sigma, pu and pd are getting bigger and bigger, but pm getting more and more small. In IFD method, with the bigger sigma, we have the opposite situation. pu and pd are getting more and more small, but pm are getting bigger.

Question(g)

```
CNFD.Eu <- function(type,K,T,S,sig,r,div,N,Nj,dx){
 #precompute constants
 dt = T/N
 nu = r - div - 0.5 * sig^2
 edx=exp(-dx)
 pu = -0.25*dt* ((sig/dx)^2 + nu/dx)
 pm = 1.0 + 0.5*dt*(sig/dx)^2 + 0.5*r*dt
 pd = -0.25*dt* ((sig/dx)^2 - nu/dx)

 #initialise asset prices at marturity
 St = array(0,dim=c(2*Nj+1))
 St[1] = S * exp(Nj*dx)
 for(j in 2:(2*Nj+1)){
   St[j] = St[j-1]*edx
 }

 #initialise option values at marturity
 C = array(0,dim = c(2,2*Nj+1))
 for(j in 1:(2*Nj+1)){
```

```
   if(type == "C")
    C[1,j] = max(0, St[j] -K)
   if(type == "P")
    C[1,j] = max(0, K - St[j])
 }

 #compute derivative boundary condition
 if(type == "C"){
   lambda.U = St[1]-St[2]
   lambda.L = 0.0
 }
 if(type == "P"){
   lambda.L = -1*(St[2*Nj]-St[2*Nj+1])
   lambda.U = 0.0
 }

 #step back at though latice
 for(i in N:1){
  #substitute boundary condition at j=-Nj into j=-Nj+1
  pmp=pp=NULL
  pmp[2*Nj] = pm +pd
  pp[2*Nj] = -pu*C[1,2*Nj-1] -(pm-2)*C[1,2*Nj] - pd*C[1,2*Nj+1] + pd*lambda.L

  #eliminate upper diagonal
  for(j in (2*Nj-1):2){
    pmp[j] = pm-pu*pd/pmp[j+1]
    pp[j] = -pu*C[1,j-1]-(pm-2)*C[1,j]-pd*C[1,j+1]-pp[j+1]*pd/pmp[j+1]
  }
  #print(pp)
  #use boundary condition at j=Nj and equation at j=Nj-1
  C[2,1] = (pp[2]+pmp[2]*lambda.U)/(pu+pmp[2])
  C[2,2] = C[2,1]-lambda.U

  #back substiatution
  for(j in 3:(2*Nj+1)){
    C[2,j] = (pp[j]-pu*C[2,j-1])/pmp[j]

    if(is.na(C[2,j])){
      C[2,j] <- C[1,j]
    }
  }
  for(j in 1:(2*Nj+1)){
    C[1,j] <- C[2,j]
  }
 }

 C[1,Nj+1]
}
efdc <- EFD.Eu(type="C",K=100,T=1,S=100,sig=0.25,r=0.06,div=0.03,
               N=1189,Nj=59,dx=0.01260504)
ifdc <- IFD.Eu(type="C",K=100,T=1,S=100,sig=0.25,r=0.06,div=0.03,
               N=1189,Nj=59,dx=0.01260504)
cnfdc <- CNFD.Eu(type="C",K=100,T=1,S=100,sig=0.25,r=0.06,div=0.03,
```

```
                N=1189,Nj=59,dx=0.01260504)
efdp <- EFD.Eu(type="P",K=100,T=1,S=100,sig=0.25,r=0.06,div=0.03,
               N=1189,Nj=59,dx=0.01260504)
ifdp <- IFD.Eu(type="P",K=100,T=1,S=100,sig=0.25,r=0.06,div=0.03,
               N=1189,Nj=59,dx=0.01260504)
cnfdp <- CNFD.Eu(type="P",K=100,T=1,S=100,sig=0.25,r=0.06,div=0.03,
                 N=1189,Nj=59,dx=0.01260504)

callprice <- c(efdc,ifdc,cnfdc)
putprice <- c(efdp,ifdp,cnfdp)

callprice <- matrix(callprice,ncol=1)
colnames(callprice) <- c("call option value")
rownames(callprice) <- c("EFD call","IFD call","CNFD call")
callprice
```

```
##           call option value
## EFD call           11.01157
## IFD call           11.00915
## CNFD call          11.01036
```

```
putprice <- matrix(putprice,ncol=1)
colnames(putprice) <- c("put option value")
rownames(putprice) <- c("EFD put","IFD put","CNFD put")
putprice
```

```
##          put option value
## EFD put          8.122201
## IFD put          8.140970
## CNFD put         8.143741
```

comment: from the table above, we can see that the price calculate by these three method are similar to each other.

Question(h)

```
newEFD <- function(type,K,T,S,sig,r,div,N,Nj,dx){
 #precompute constants
 dt = T/N
 nu = r - div - 0.5 * sig^2
 edx = exp(-dx)
 pu = 0.5*dt* ((sig/dx)^2 + nu/dx)
 pm = 1.0 - dt*(sig/dx)^2 - r*dt
 pd = 0.5*dt* ((sig/dx)^2 - nu/dx)

 #initialise asset prices at marturity
 St = array(0,dim=c(2*Nj+1))
 St[1] = S * exp(Nj*dx)
 for(j in 2:(2*Nj+1)){
   St[j] = St[j-1]*edx
 }

 #initialise option values to marturity
```

```
 C = array(0,dim = c(2,2*Nj+1))
 for(j in 1:(2*Nj+1)){
   if(type == "C")
    C[1,j] = max(0, St[j] - K)
   if(type == "P")
    C[1,j] = max(0, K - St[j])
 }

 #step back at though latice
 for(i in N:1){
   for(j in 2:(2*Nj)){
     C[2,j] = pu*C[1,j-1]+pm*C[1,j]+pd*C[1,j+1]
   }

   #boundary conditions
   if(type == "C"){
     C[2,1] = C[2,2] + (St[1]-St[2])
     C[2,2*Nj+1] = C[2,2*Nj]
   }
   if(type == "P"){
     C[2,1] = C[2,2]
     C[2,2*Nj+1] = C[2,2*Nj] + (St[2*Nj+1]-St[2*Nj])
   }
   for(j in 1:(2*Nj+1)){
    C[1,j] <- C[2,j]
   }
 }
 list(S=round(St,4),V=round(C,4))
}

S <- newEFD(type="C",K=100,T=1,S=100,sig=0.25,r=0.06,div=0.03,
            N=1189,Nj=59,dx=0.01260504)$S
V <- newEFD(type="C",K=100,T=1,S=100,sig=0.25,r=0.06,div=0.03,
            N=1189,Nj=59,dx=0.01260504)$V

# Delta
delta <- function(Nj,V,S){
  return((V[1,Nj+1]-V[1,Nj-1])/(S[Nj+1]-S[Nj-1]))
}
delta <- delta(59,V,S)

# Gamma
gamma <- function(Nj,V,S){
  return(2*((V[1,Nj+1]-V[1,Nj])-(V[1,Nj]-V[1,Nj-1]))/(S[Nj+1]-S[Nj-1]))
}
gamma <- gamma(59,V,S)

# Theta
dt <- 1/1189
C1 <- EFD.Eu(type="C",100,1+dt,100,0.25,0.06,0.03,1189,59,dx=0.01260504)
C2 <- EFD.Eu(type="C",100,1,100,0.25,0.06,0.03,1189,59,dx=0.01260504)
theta <- (C1-C2)/dt
```

```r
# Vega
sigma <- 0.25
delta.sig <- 0.001*sigma
dx1 = 6 * (sigma+delta.sig) * sqrt(1)/(2*59+1)
dx2 = 6 * (sigma-delta.sig) * sqrt(1)/(2*59+1)
C1 <- EFD.Eu(type="C",100,1,100,sigma+delta.sig,0.06,0.03,1189,59,dx1)
C2 <- EFD.Eu(type="C",100,1,100,sigma-delta.sig,0.06,0.03,1189,59,dx2)
vega <- (C1-C2)/(2*delta.sig)

DF <- data.frame(delta,gamma,theta,vega)
DF
```

```
##       delta       gamma    theta     vega
## 1 0.5980181 -0.02632094 5.775552 37.56406
```

Problem 2

Question(a)

```r
getwd() # get currency work directory
```

```
## [1] "E:/RFiles/New folder"
```

```r
#read data that download from Yahoo finance
JPM1c <- read.csv("JPMCalls 170421.csv")
JPM2c <- read.csv("JPMCalls 170519.csv")
JPM3c <- read.csv("JPMCalls 170616.csv")
JPM1p <- read.csv("JPMPuts 170421.csv")
JPM2p <- read.csv("JPMPuts 170519.csv")
JPM3p <- read.csv("JPMPuts 170616.csv")
#calculate market price
MPrice1c <- matrix((JPM1c$Bid + JPM1c$Ask)/2) #one month
MPrice2c <- matrix((JPM2c$Bid + JPM2c$Ask)/2) #two month
MPrice3c <- matrix((JPM3c$Bid + JPM3c$Ask)/2) #three month
MPrice1p <- matrix((JPM1p$Bid + JPM1p$Ask)/2) #one month
MPrice2p <- matrix((JPM2p$Bid + JPM2p$Ask)/2) #two month
MPrice3p <- matrix((JPM3p$Bid + JPM3p$Ask)/2) #three month

S0 <- 91.41 # Close price of JPM at 2017-03-07
#strike price of JPM option
K1c <- matrix(JPM1c$Strike) #one month
K2c <- matrix(JPM2c$Strike) #two month
K3c <- matrix(JPM3c$Strike) #three month
K1p <- matrix(JPM1p$Strike) #one month
K2p <- matrix(JPM2p$Strike) #two month
K3p <- matrix(JPM3p$Strike) #three month
#calculate time t of each option
tau1 <- as.numeric(difftime("2017-04-21","2017-03-07",units = "days"))/365
tau2 <- as.numeric(difftime("2017-05-19","2017-03-07",units = "days"))/365
tau3 <- as.numeric(difftime("2017-06-16","2017-03-07",units = "days"))/365

# Function to calculate opntion price by BS formular
```

```
BS <- function(type, S0, K, tau, r, sigma,div){

  d1 <- (log(S0/K)+(r-div+sigma^2/2)*tau) / (sigma*sqrt(tau))
  d2 <- d1 - sigma*sqrt(tau)
  if(type=="C"){
    Price <- S0*exp(-div*tau)*pnorm(d1) - K*exp(-r*tau)*pnorm(d2)
  }
  if(type=="P"){
    Price <- K*exp(-r*tau)*pnorm(-d2) - S0*exp(-div*tau)*pnorm(-d1)
  }
  return(Price)
}
#Function to calculate error between BS and Market price
err <- function(type,S0,K,r,tau,sig,div,MPrice){
    BS(type,S0,K,tau,r,sig,div) - MPrice
}

#Function to find BS Implied Vol using Bisection Method
Ivol.BS <- function(type,S0, K, r,tau,div, MPrice){
    sig <- c()
#loop for every strike price and market price
  for(i in 1:10){
    a <- -0.001
    b <- 2
    c <- (a+b)/2
#Loop until that the value of function to sigma is less than tolerance level 1e-4
    while(abs(b-a) > 1e-4){
      fa <- err(type,S0,K[i],r,tau,a,div,MPrice[i])
      fc <- err(type,S0,K[i],r,tau,c,div,MPrice[i])
      if( fa * fc < 0)
          b <- c
      else
          a <- c
      c <- (a+b)/2
    }
   sig <- c(sig,c)
  }
  return(sig)
}

Imv1c <- Ivol.BS("C",S0,K1c,0.0075,tau1,0,MPrice1c) #one month
Imv2c <- Ivol.BS("C",S0,K2c,0.0075,tau2,0,MPrice2c) #two month
Imv3c <- Ivol.BS("C",S0,K3c,0.0075,tau3,0,MPrice3c) #three month
Imv1p <- Ivol.BS("P",S0,K1p,0.0075,tau1,0,MPrice1p) #one month
Imv2p <- Ivol.BS("P",S0,K2p,0.0075,tau2,0,MPrice2p) #two month
Imv3p <- Ivol.BS("P",S0,K3p,0.0075,tau3,0,MPrice3p) #three month
```

Question(b)

```
dx.compute <- function(T,sig){
  N = 2
  Nj <- floor((6*sqrt(N/3)-1)/2)
```

```
  dt = T/N
  delta.x <- c()
  for(i in 1:10){
    dx = 6 * sig[i] * sqrt(T)/(2*Nj+1)
    ebs <- dx^2 + dt
    while(abs(ebs)>0.001){
      N <- N+1
      Nj <- floor((6*sqrt(N/3)-1)/2)
      dt = T/N
      dx = 6 * sig[i] * sqrt(T)/(2*Nj+1)
      ebs <- dx^2 + dt
    }
    delta.x <- c(delta.x,dx)
  }
  cat("N=",N,"Nj=",Nj,"dx=",delta.x,"\n")
  return(delta.x)
}
c1 <- dx.compute(T=tau1,sig=Imv1c)
```

```
## N= 184 Nj= 22 dx= 0.0181344 0.01624841 0.01360232 0.01182492 0.01075905 0.01051044 0.009473149 0.009
```

```
c2 <- dx.compute(T=tau2,sig=Imv2c)
```

```
## N= 252 Nj= 26 dx= 0.01404165 0.01431982 0.0120203 0.01178231 0.01095708 0.01048729 0.01011331 0.00988
```

```
c3 <- dx.compute(T=tau3,sig=Imv3c)
```

```
## N= 387 Nj= 33 dx= 0.01669015 0.01688859 0.01442381 0.01468265 0.01361276 0.01173757 0.01105306 0.010
```

```
c1
```

```
##  [1] 0.018134397 0.016248411 0.013602315 0.011824916 0.010759048
##  [6] 0.010510441 0.009473149 0.009181678 0.009338844 0.009796053
```

```
p1 <- dx.compute(T=tau1,sig=Imv1p)
```

```
## N= 170 Nj= 22 dx= 0.0165656 0.01393379 0.01293365 0.01182206 0.01088192 0.009970363 0.009438858 0.008
```

```
p2 <- dx.compute(T=tau2,sig=Imv2p)
```

```
## N= 264 Nj= 27 dx= 0.01552059 0.01389143 0.01309621 0.01236651 0.01152363 0.01105007 0.01047525 0.010
```

```
p3 <- dx.compute(T=tau3,sig=Imv3p)
```

```
## N= 343 Nj= 31 dx= 0.01385003 0.01311289 0.0124614 0.01184966 0.01143063 0.01088924 0.0104702 0.01023
```

```
EFD.C1=EFD.P1=EFD.C2=EFD.P2=EFD.C3=EFD.P3=c()
IFD.C1=IFD.P1=IFD.C2=IFD.P2=IFD.C3=IFD.P3=c()
CNFD.C1=CNFD.P1=CNFD.C2=CNFD.P2=CNFD.C3=CNFD.P3=c()

for(i in 1:10){
  #EFD
  EFD.C1 <- c(EFD.C1,EFD.Eu(type="C",K=K1c[i],T=tau1,S=S0,sig=Imv1c[i],
                            r=0.0075,div=0,N=184,Nj=22,dx=c1[i]))
  EFD.P1 <- c(EFD.P1,EFD.Eu(type="P",K=K1p[i],T=tau1,S=S0,sig=Imv1p[i],
                            r=0.0075,div=0,N=170,Nj=22,dx=p1[i]))
  EFD.C2 <- c(EFD.C2,EFD.Eu(type="C",K=K2c[i],T=tau2,S=S0,sig=Imv2c[i],
                            r=0.0075,div=0,N=252,Nj=26,dx=c2[i]))
  EFD.P2 <- c(EFD.P2,EFD.Eu(type="P",K=K2p[i],T=tau2,S=S0,sig=Imv2p[i],
```

```
                              r=0.0075,div=0,N=264,Nj=27,dx=p2[i]))
  EFD.C3 <- c(EFD.C3,EFD.Eu(type="C",K=K3c[i],T=tau3,S=S0,sig=Imv3c[i],
                              r=0.0075,div=0,N=387,Nj=33,dx=c3[i]))
  EFD.P3 <- c(EFD.P3,EFD.Eu(type="P",K=K3p[i],T=tau3,S=S0,sig=Imv3p[i],
                              r=0.0075,div=0,N=343,Nj=31,dx=p3[i]))

  #IFD
  IFD.C1 <- c(IFD.C1,IFD.Eu(type="C",K=K1c[i],T=tau1,S=S0,sig=Imv1c[i],
                              r=0.0075,div=0,N=184,Nj=22,dx=c1[i]))
  IFD.P1 <- c(IFD.P1,IFD.Eu(type="P",K=K1p[i],T=tau1,S=S0,sig=Imv1p[i],
                              r=0.0075,div=0,N=170,Nj=22,dx=p1[i]))
  IFD.C2 <- c(IFD.C2,IFD.Eu(type="C",K=K2c[i],T=tau2,S=S0,sig=Imv2c[i],
                              r=0.0075,div=0,N=252,Nj=26,dx=c2[i]))
  IFD.P2 <- c(IFD.P2,IFD.Eu(type="P",K=K2p[i],T=tau2,S=S0,sig=Imv2p[i],
                              r=0.0075,div=0,N=264,Nj=27,dx=p2[i]))
  IFD.C3 <- c(IFD.C3,IFD.Eu(type="C",K=K3c[i],T=tau3,S=S0,sig=Imv3c[i],
                              r=0.0075,div=0,N=387,Nj=33,dx=c3[i]))
  IFD.P3 <- c(IFD.P3,IFD.Eu(type="P",K=K3p[i],T=tau3,S=S0,sig=Imv3p[i],
                              r=0.0075,div=0,N=343,Nj=31,dx=p3[i]))

  #CNFD
  CNFD.C1 <- c(CNFD.C1,CNFD.Eu(type="C",K=K1c[i],T=tau1,S=S0,sig=Imv1c[i],
                                 r=0.0075,div=0,N=184,Nj=22,dx=c1[i]))
  CNFD.P1 <- c(CNFD.P1,CNFD.Eu(type="P",K=K1p[i],T=tau1,S=S0,sig=Imv1p[i],
                                 r=0.0075,div=0,N=170,Nj=22,dx=p1[i]))
  CNFD.C2 <- c(CNFD.C2,CNFD.Eu(type="C",K=K2c[i],T=tau2,S=S0,sig=Imv2c[i],
                                 r=0.0075,div=0,N=252,Nj=26,dx=c2[i]))
  CNFD.P2 <- c(CNFD.P2,CNFD.Eu(type="P",K=K2p[i],T=tau2,S=S0,sig=Imv2p[i],
                                 r=0.0075,div=0,N=264,Nj=27,dx=p2[i]))
  CNFD.C3 <- c(CNFD.C3,CNFD.Eu(type="C",K=K3c[i],T=tau3,S=S0,sig=Imv3c[i],
                                 r=0.0075,div=0,N=387,Nj=33,dx=c3[i]))
  CNFD.P3 <- c(CNFD.P3,CNFD.Eu(type="P",K=K3p[i],T=tau3,S=S0,sig=Imv3p[i],
                                 r=0.0075,div=0,N=343,Nj=31,dx=p3[i]))
}

EFD.C <- c(EFD.C1,EFD.C2,EFD.C3)
EFD.P <- c(EFD.P1,EFD.P2,EFD.P3)
IFD.C <- c(IFD.C1,IFD.C2,IFD.C3)
IFD.P <- c(IFD.P1,IFD.P2,IFD.P3)
CNFD.C <- c(CNFD.C1,CNFD.C2,CNFD.C3)
CNFD.P <- c(CNFD.P1,CNFD.P2,CNFD.P3)

DF <- data.frame(EFD.C,EFD.P,IFD.C,IFD.P,CNFD.C,CNFD.P)
DF
```

```
##          EFD.C      EFD.P       IFD.C      IFD.P      CNFD.C     CNFD.P
## 1  12.47647242 0.03278332 12.47626567 0.05728999 12.47636034 0.05573603
## 2  10.15100110 0.07578808 10.14978267 0.09715942 10.15038568 0.09596133
## 3   7.72655378 0.12726833  7.72465740 0.14732101  7.72560167 0.14625417
## 4   5.55098056 0.19858642  5.54807176 0.21689699  5.54952498 0.21603454
## 5   3.72676802 0.33047342  3.72300241 0.34685011  3.72488665 0.34636104
## 6   2.41353670 0.55131166  2.40972560 0.56527606  2.41163283 0.56539564
## 7   1.24769843 1.00276063  1.24553405 1.01426371  1.24661498 1.01526098
## 8   0.62292795 1.72256677  0.62271531 1.73195813  0.62281857 1.73363521
```

```
## 9   0.32598738 2.81426717  0.32708861 2.82298914  0.32653626 2.82461722
## 10  0.08500176 4.31213481  0.08677958 4.32177498  0.08589286 4.32247621
## 11 12.27631217 0.10714336 12.27660490 0.13192187 12.27645222 0.13090426
## 12 10.29897797 0.24427919 10.29784875 0.26677363 10.29840982 0.26598604
## 13  7.87280205 0.36087724  7.87119764 0.38195742  7.87199771 0.38137376
## 14  6.07545433 0.53604987  6.07264284 0.55553985  6.07404831 0.55527007
## 15  4.32642543 0.76629045  4.32306444 0.78380817  4.32474596 0.78394963
## 16  2.93878302 1.18924052  2.93548240 1.20496847  2.93713382 1.20567680
## 17  1.87468248 1.75797733  1.87225550 1.77184660  1.87346876 1.77307957
## 18  1.13606352 2.63581046  1.13498103 2.64850360  1.13552058 2.65010088
## 19  0.65998798 3.66060200  0.66021172 3.67241922  0.66009788 3.67394469
## 20  0.23010253 5.20896558  0.23165841 5.22148531  0.23088075 5.22264244
## 21 22.04864260 0.44344881 22.05000919 0.46659505 22.04930111 0.46610464
## 22 19.85049320 0.60977024 19.85135646 0.63144909 19.85091413 0.63119698
## 23 17.25047665 0.85033166 17.25122065 0.87046523 17.25083942 0.87053507
## 24 15.17495837 1.18661655 15.17491649 1.20509829 15.17493320 1.20556074
## 25 12.89981505 1.69739927 12.89922714 1.71438322 12.89951842 1.71529363
## 26 10.45053101 2.32987614 10.44964043 2.34533241 10.45008395 2.34661077
## 27  8.40117946 3.20889961  8.39959866 3.22329625  8.40038824 3.22479384
## 28  6.62318939 4.38303616  6.62091355 4.39707568  6.62205149 4.39857382
## 29  4.97593350 5.73165938  4.97331848 5.74563074  4.97462653 5.74682584
## 30  3.62493111 7.40968706  3.62234140 7.42424094  3.62363683 7.42496442
```

Question(c)

```
S1c=V1c=S2c=V2c=S3c=V3c=c()
S1p=V1p=S2p=V2p=S3p=V3p=c()
v1.c1=v1.c2=v1.c3=v1.p1=v1.p2=v1.p3=c()
v2.c1=v2.c2=v2.c3=v2.p1=v2.p2=v2.p3=c()
for(i in 1:10){
  S1c <- c(S1c,newEFD(type="C",K=K1c[i],T=tau1,S=S0,sig=Imv1c[i],r=0.0075,div=0.0,
              N=184,Nj=22,dx=c1[i])$S[23]-newEFD(type="C",K=K1c[i],T=tau1,S=S0,
sig=Imv1c[i],r=0.0075,div=0.0,N=184,Nj=22,dx=c1[i])$S[21])
  V1c <- c(V1c,newEFD(type="C",K=K1c[i],T=tau1,S=S0,sig=Imv1c[i],r=0.0075,div=0.0,
              N=184,Nj=22,dx=c1[i])$V[1,23]-newEFD(type="C",K=K1c[i],T=tau1,S=S0,
sig=Imv1c[i],r=0.0075,div=0.0,N=184,Nj=22,dx=c1[i])$V[1,21])
  v1.c1 <- c(v1.c1,newEFD(type="C",K=K1c[i],T=tau1,S=S0,sig=Imv1c[i],r=0.0075,div=0.0,
              N=184,Nj=22,dx=c1[i])$V[1,23]-newEFD(type="C",K=K1c[i],T=tau1,S=S0,
sig=Imv1c[i],r=0.0075,div=0.0,N=184,Nj=22,dx=c1[i])$V[1,22])
  v2.c1 <- c(v2.c1,newEFD(type="C",K=K1c[i],T=tau1,S=S0,sig=Imv1c[i],r=0.0075,div=0.0,
              N=184,Nj=22,dx=c1[i])$V[1,22]-newEFD(type="C",K=K1c[i],T=tau1,S=S0,
sig=Imv1c[i],r=0.0075,div=0.0,N=184,Nj=22,dx=c1[i])$V[1,21])

  S2c <- c(S2c,newEFD(type="C",K=K2c[i],T=tau2,S=S0,sig=Imv2c[i],r=0.0075,div=0.0,
              N=252,Nj=26,dx=c2[i])$S[27]-newEFD(type="C",K=K2c[i],T=tau2,S=S0,
sig=Imv2c[i],r=0.0075,div=0.0,N=252,Nj=26,dx=c2[i])$S[25])
  V2c <- c(V2c,newEFD(type="C",K=K2c[i],T=tau2,S=S0,sig=Imv2c[i],r=0.0075,div=0.0,
              N=252,Nj=26,dx=c2[i])$V[1,27]-newEFD(type="C",K=K2c[i],T=tau2,S=S0,
sig=Imv2c[i],r=0.0075,div=0.0,N=252,Nj=26,dx=c2[i])$V[1,25])
  v1.c2 <- c(v1.c2,newEFD(type="C",K=K2c[i],T=tau2,S=S0,sig=Imv2c[i],r=0.0075,div=0.0,
              N=252,Nj=26,dx=c2[i])$V[1,27]-newEFD(type="C",K=K2c[i],T=tau2,S=S0,
sig=Imv2c[i],r=0.0075,div=0.0,N=252,Nj=26,dx=c2[i])$V[1,26])
```

```
  v2.c2 <- c(v2.c2,newEFD(type="C",K=K2c[i],T=tau2,S=S0,sig=Imv2c[i],r=0.0075,div=0.0,
              N=252,Nj=26,dx=c2[i])$V[1,26]-newEFD(type="C",K=K2c[i],T=tau2,S=S0,
sig=Imv2c[i],r=0.0075,div=0.0,N=252,Nj=26,dx=c2[i])$V[1,25])

  S3c <- c(S3c,newEFD(type="C",K=K3c[i],T=tau3,S=S0,sig=Imv3c[i],r=0.0075,div=0.0,
              N=387,Nj=33,dx=c3[i])$S[34]-newEFD(type="C",K=K3c[i],T=tau3,S=S0,
sig=Imv3c[i],r=0.0075,div=0.0,N=387,Nj=33,dx=c3[i])$S[32])
  V3c <- c(V3c,newEFD(type="C",K=K3c[i],T=tau3,S=S0,sig=Imv3c[i],r=0.0075,div=0.0,
              N=387,Nj=33,dx=c3[i])$V[1,34]-newEFD(type="C",K=K3c[i],T=tau3,S=S0,
sig=Imv3c[i],r=0.0075,div=0.0,N=387,Nj=33,dx=c3[i])$V[1,32])
  v1.c3 <- c(v1.c3,newEFD(type="C",K=K3c[i],T=tau3,S=S0,sig=Imv3c[i],r=0.0075,div=0.0,
              N=387,Nj=33,dx=c3[i])$V[1,34]-newEFD(type="C",K=K3c[i],T=tau3,S=S0,
sig=Imv3c[i],r=0.0075,div=0.0,N=387,Nj=33,dx=c3[i])$V[1,33])
  v2.c3 <- c(v2.c3,newEFD(type="C",K=K3c[i],T=tau3,S=S0,sig=Imv3c[i],r=0.0075,div=0.0,
              N=387,Nj=33,dx=c3[i])$V[1,33]-newEFD(type="C",K=K3c[i],T=tau3,S=S0,
sig=Imv3c[i],r=0.0075,div=0.0,N=387,Nj=33,dx=c3[i])$V[1,32])
}

for(i in 1:10){
  S1p <- c(S1p,newEFD(type="P",K=K1p[i],T=tau1,S=S0,sig=Imv1p[i],r=0.0075,div=0.0,
              N=170,Nj=22,dx=p1[i])$S[23]-newEFD(type="P",K=K1p[i],T=tau1,S=S0,
sig=Imv1p[i],r=0.0075,div=0.0,N=170,Nj=22,dx=p1[i])$S[21])
  V1p <- c(V1p,newEFD(type="P",K=K1p[i],T=tau1,S=S0,sig=Imv1p[i],r=0.0075,div=0.0,
              N=170,Nj=22,dx=p1[i])$V[1,23]-newEFD(type="P",K=K1p[i],T=tau1,S=S0,
sig=Imv1p[i],r=0.0075,div=0.0,N=170,Nj=22,dx=p1[i])$V[1,21])
  v1.p1 <- c(v1.p1,newEFD(type="P",K=K1p[i],T=tau1,S=S0,sig=Imv1p[i],r=0.0075,div=0.0,
              N=170,Nj=22,dx=p1[i])$V[1,23]-newEFD(type="P",K=K1p[i],T=tau1,S=S0,
sig=Imv1p[i],r=0.0075,div=0.0,N=170,Nj=22,dx=p1[i])$V[1,22])
  v2.p1 <- c(v2.p2,newEFD(type="P",K=K1p[i],T=tau1,S=S0,sig=Imv1p[i],r=0.0075,div=0.0,
              N=170,Nj=22,dx=p1[i])$V[1,22]-newEFD(type="P",K=K1p[i],T=tau1,S=S0,
sig=Imv1p[i],r=0.0075,div=0.0,N=170,Nj=22,dx=p1[i])$V[1,21])

  S2p <- c(S2p,newEFD(type="P",K=K2p[i],T=tau2,S=S0,sig=Imv2p[i],r=0.0075,div=0.0,
              N=264,Nj=27,dx=p2[i])$S[28]-newEFD(type="P",K=K2p[i],T=tau2,S=S0,
sig=Imv2p[i],r=0.0075,div=0.0,N=264,Nj=27,dx=p2[i])$S[26])
  V2p <- c(V2p,newEFD(type="P",K=K2p[i],T=tau2,S=S0,sig=Imv2p[i],r=0.0075,div=0.0,
              N=264,Nj=27,dx=p2[i])$V[1,28]-newEFD(type="P",K=K2p[i],T=tau2,S=S0,
sig=Imv2p[i],r=0.0075,div=0.0,N=264,Nj=27,dx=p2[i])$V[1,26])
  v1.p2 <- c(v1.p2,newEFD(type="P",K=K2p[i],T=tau2,S=S0,sig=Imv2p[i],r=0.0075,div=0.0,
              N=264,Nj=27,dx=p2[i])$V[1,28]-newEFD(type="P",K=K2p[i],T=tau2,S=S0,
sig=Imv2p[i],r=0.0075,div=0.0,N=264,Nj=27,dx=p2[i])$V[1,27])
  v2.p2 <- c(v2.p2,newEFD(type="P",K=K2p[i],T=tau2,S=S0,sig=Imv2p[i],r=0.0075,div=0.0,
              N=264,Nj=27,dx=p2[i])$V[1,27]-newEFD(type="P",K=K2p[i],T=tau2,S=S0,
sig=Imv2p[i],r=0.0075,div=0.0,N=264,Nj=27,dx=p2[i])$V[1,26])

  S3p <- c(S3p,newEFD(type="P",K=K3p[i],T=tau3,S=S0,sig=Imv3p[i],r=0.0075,div=0.0,
              N=343,Nj=31,dx=p3[i])$S[32]-newEFD(type="P",K=K3p[i],T=tau3,S=S0,
sig=Imv3p[i],r=0.0075,div=0.0,N=343,Nj=31,dx=p3[i])$S[30])
  V3p <- c(V3p,newEFD(type="P",K=K3p[i],T=tau3,S=S0,sig=Imv3p[i],r=0.0075,div=0.0,
              N=343,Nj=31,dx=p3[i])$V[1,32]-newEFD(type="P",K=K3p[i],T=tau3,S=S0,
sig=Imv3p[i],r=0.0075,div=0.0,N=343,Nj=31,dx=p3[i])$V[1,30])
  v1.p3 <- c(v1.p3,newEFD(type="P",K=K3p[i],T=tau3,S=S0,sig=Imv3p[i],r=0.0075,div=0.0,
              N=343,Nj=31,dx=p3[i])$V[1,32]-newEFD(type="P",K=K3p[i],T=tau3,S=S0,
```

```
sig=Imv3p[i],r=0.0075,div=0.0,N=343,Nj=31,dx=p3[i])$V[1,31])
  v2.p3 <- c(v2.p3,newEFD(type="P",K=K3p[i],T=tau3,S=S0,sig=Imv3p[i],r=0.0075,div=0.0,
              N=343,Nj=31,dx=p3[i])$V[1,31]-newEFD(type="P",K=K3p[i],T=tau3,S=S0,
sig=Imv3p[i],r=0.0075,div=0.0,N=343,Nj=31,dx=p3[i])$V[1,30])
}

# Delta
delta <- function(V,S){
  return(V/S)
}
delta1c <- delta(V1c,S1c)
delta2c <- delta(V2c,S2c)
delta3c <- delta(V3c,S3c)
delta1p <- delta(V1p,S1p)
delta2p <- delta(V2p,S2p)
delta3p <- delta(V3p,S3p)

# Gamma
gamma <- function(V1,V2,S){
  return(2*(V1-V2)/S)
}
gamma1c <- gamma(v1.c1,v2.c1,S1c)
gamma2c <- gamma(v1.c2,v2.c2,S2c)
gamma3c <- gamma(v1.c3,v2.c3,S3c)
gamma1p <- gamma(v1.p1,v2.p1,S1p)
gamma2p <- gamma(v1.p2,v2.p2,S2p)
gamma3p <- gamma(v1.p3,v2.p3,S3p)

# Theta
theta <- function(type,K,T,S,sig,r,div,N,Nj,dx){
  dt=T/N
  x <- c()
  for(i in 1:10){
  a <- (EFD.Eu(type,K[i],T+dt,S,sig[i],r,div,N,Nj,dx[i])-
           EFD.Eu(type,K[i],T,S,sig[i],r,div,N,Nj,dx[i]))/dt
  x <- c(x,a)
  }
  return(x)
}
theta1c <- theta(type="C",K=K1c,T=tau1,S=S0,sig=Imv1c,r=0.0075,div=0.0,
                 N=184,Nj=22,dx=c1)
theta2c <- theta(type="C",K=K2c,T=tau2,S=S0,sig=Imv2c,r=0.0075,div=0.0,
                 N=252,Nj=26,dx=c2)
theta3c <- theta(type="C",K=K3c,T=tau3,S=S0,sig=Imv3c,r=0.0075,div=0.0,
                 N=387,Nj=33,dx=c3)
theta1p <- theta(type="P",K=K1c,T=tau1,S=S0,sig=Imv1p,r=0.0075,div=0.0,
                 N=170,Nj=22,dx=p1)
theta2p <- theta(type="P",K=K1c,T=tau2,S=S0,sig=Imv2p,r=0.0075,div=0.0,
                 N=264,Nj=27,dx=p2)
theta3p <- theta(type="P",K=K1c,T=tau3,S=S0,sig=Imv3p,r=0.0075,div=0.0,
                 N=343,Nj=31,dx=p3)

# Vega
```

```
vega <- function(type,K,T,S,sig,r,div,N,Nj){
  delta.sig=dx1=dx2=C1=C2=x=c()
  for(i in 1:10){
    delta.sig <- c(delta.sig,0.001*sig[i])
    dx1 = c(dx1,6 * (sig[i]+delta.sig[i]) * sqrt(T)/(2*Nj+1))
    dx2 = c(dx2,6 * (sig[i]-delta.sig[i]) * sqrt(T)/(2*Nj+1))
    C1 <- c(C1,EFD.Eu(type,K[i],T,S,sig[i]+delta.sig[i],r,div,N,Nj,dx1[i]))
    C2 <- c(C2,EFD.Eu(type,K[i],T,S,sig[i]-delta.sig[i],r,div,N,Nj,dx2[i]))
    x <- c(x,(C1[i]-C2[i])/(2*delta.sig[i]))
  }
  return(x)
}
vega1c <- vega(type="C",K=K1c,T=tau1,S=S0,sig=Imv1c,r=0.0075,div=0.0,
                N=184,Nj=22)
vega2c <- vega(type="C",K=K2c,T=tau2,S=S0,sig=Imv2c,r=0.0075,div=0.0,
                N=252,Nj=26)
vega3c <- vega(type="C",K=K3c,T=tau3,S=S0,sig=Imv3c,r=0.0075,div=0.0,
                N=387,Nj=33)
vega1p <- vega(type="P",K=K1c,T=tau1,S=S0,sig=Imv1p,r=0.0075,div=0.0,
                N=170,Nj=22)
vega2p <- vega(type="P",K=K1c,T=tau2,S=S0,sig=Imv2p,r=0.0075,div=0.0,
                N=264,Nj=27)
vega3p <- vega(type="P",K=K1c,T=tau3,S=S0,sig=Imv3p,r=0.0075,div=0.0,
                N=343,Nj=31)

delta.c <- c(delta1c,delta2c,delta3c)
delta.p <- c(delta1p,delta2p,delta3p)
gamma.c <- c(gamma1c,gamma2c,gamma3c)
gamma.p <- c(gamma1p,gamma2p,gamma3p)
theta.c <- c(theta1c,theta2c,theta3c)
theta.p <- c(theta1p,theta2p,theta3p)
vega.c <- c(vega1c,vega2c,vega3c)
vega.p <- c(vega1p,vega2p,vega3p)

#Greeks for call option
DF1 <- data.frame(delta.c,gamma.c,theta.c,vega.c)
DF1
```

```
##       delta.c     gamma.c   theta.c    vega.c
## 1  0.88202713 -0.04223683 12.030567  7.192324
## 2  0.85102507 -0.04471235 12.406024  8.279545
## 3  0.81689079 -0.04641200 11.677982  9.356210
## 4  0.75155421 -0.05110624 11.740029 11.161071
## 5  0.64688427 -0.05653070 11.967333 12.411820
## 6  0.51377517 -0.05860240 11.915896 12.633518
## 7  0.35970030 -0.05319149  9.521035 10.966429
## 8  0.22548846 -0.04202822  6.991880  8.108173
## 9  0.13641639 -0.03028896  4.915073  6.093330
## 10 0.04478602 -0.01304877  2.038692  2.156738
## 11 0.89606299 -0.03295564  6.354821  8.418979
## 12 0.84080127 -0.03946080  8.273732 10.668531
## 13 0.80631238 -0.04082367  7.775088 11.852722
## 14 0.72531657 -0.04633878  8.857826 14.631616
## 15 0.63205451 -0.04947415  8.976751 15.881583
```

```
## 16 0.51917419 -0.05047742  8.738468 16.175681
## 17 0.39910050 -0.04775928  7.840867 15.454771
## 18 0.28684888 -0.04129923  6.493380 12.986592
## 19 0.19411244 -0.03297919  4.995985  9.718247
## 20 0.08328844 -0.01810150  2.675225  5.565037
## 21 0.94791801 -0.02526750  4.251014  6.147161
## 22 0.92252333 -0.02853231  5.579171  8.067325
## 23 0.92068476 -0.02646333  4.924471  8.196285
## 24 0.88293381 -0.03054220  6.371540 10.213175
## 25 0.85437393 -0.03210591  6.753985 12.296598
## 26 0.82958732 -0.03242447  6.423546 13.426654
## 27 0.77525450 -0.03543461  6.959462 15.413765
## 28 0.70238395 -0.03870773  7.581116 17.145353
## 29 0.61822608 -0.04043184  7.725054 18.765665
## 30 0.52270494 -0.04071608  7.607423 19.111512
```

```
#Greeks for put option
DF2 <- data.frame(delta.p,gamma.p,theta.p,vega.p)
DF2
```

```
##         delta.p      gamma.p    theta.p     vega.p
## 1  -0.005163511  0.006949631  8.4969032  6.1782750
## 2  -0.014168473  0.017110561  8.0325897  7.2396316
## 3  -0.025006262  0.022793688  9.4188242  9.2932661
## 4  -0.041196104  0.033468977 10.3529845 11.1018613
## 5  -0.070654336  0.039777247 10.7551699 12.3517003
## 6  -0.119982619  0.055836185  9.9875775 12.5752134
## 7  -0.208036739  0.046957520  8.1751197 10.9048765
## 8  -0.335104109  0.038590706  5.2794253  8.0525450
## 9  -0.499380502  0.004825562  2.2807136  5.0583232
## 10 -0.683956574 -0.039594125 -0.7152622  0.5698885
## 11 -0.015336572 -0.003331020  6.9790462  9.7316391
## 12 -0.037045666 -0.007455732  7.1678511 10.8848104
## 13 -0.055814337 -0.010717672  7.9714057 12.9374009
## 14 -0.083617300 -0.014678899  8.5611276 14.6580402
## 15 -0.121527778 -0.019519520  8.6216453 15.8329872
## 16 -0.182062956 -0.025554413  8.3782083 16.1149105
## 17 -0.260786441 -0.031312975  7.3526594 15.4546763
## 18 -0.363088894 -0.035706672  5.9748189 13.1486067
## 19 -0.482209362 -0.037128287  3.8690717 10.0563215
## 20 -0.603883049 -0.034579815  1.2544391  5.0563993
## 21 -0.057526777 -0.009425511  5.2348810 11.6489972
## 22 -0.080197612 -0.012350762  5.8671028 13.8439364
## 23 -0.112016646 -0.015779435  6.4457433 15.8625218
## 24 -0.155095338 -0.019888696  6.8424870 17.5251911
## 25 -0.214211373 -0.024221781  7.0568779 18.6726441
## 26 -0.285898837 -0.028222200  6.7956769 19.0114079
## 27 -0.373830326 -0.030915577  6.1677073 18.4980700
## 28 -0.472430945 -0.031749391  5.2876595 16.4280340
## 29 -0.577887223 -0.030214155  3.9761144 13.4483013
## 30 -0.678539020 -0.026088929  1.7614507  8.0504223
```

Question(d)

```
MPrice.C <- c(MPrice1c,MPrice2c,MPrice3c)
MPrice.P <- c(MPrice1p,MPrice2p,MPrice3p)

strikec <- c(K1c,K2c,K3c)
strikep <- c(K1p,K2p,K3p)

par(mfrow=c(1,2))
plot(strikec,MPrice.C,typ="b", col="blue", main=c("Option Pricing"),
     xlab="Strike Price", ylab="Option Price")
lines(strikec,EFD.C,col = "red")
lines(strikec,IFD.C.col = "green")
```

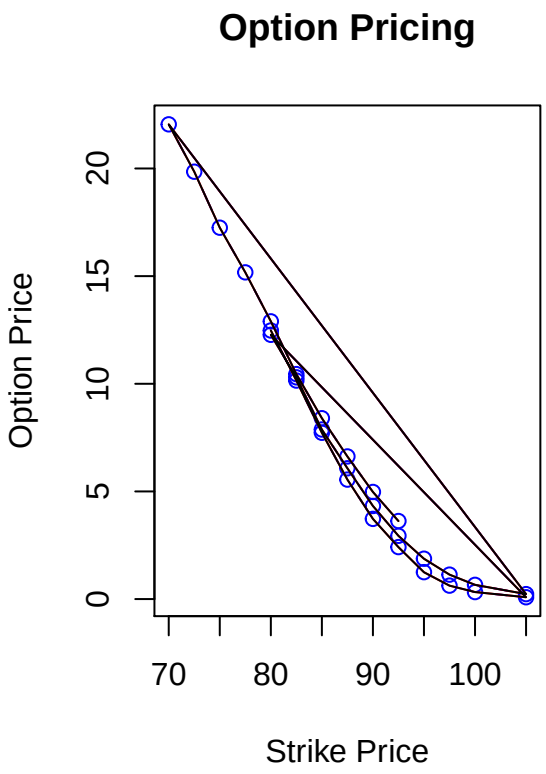
```
## Warning in plot.xy(xy.coords(x, y), type = type, ...): "IFD.C.col" is not a
## graphical parameter
```

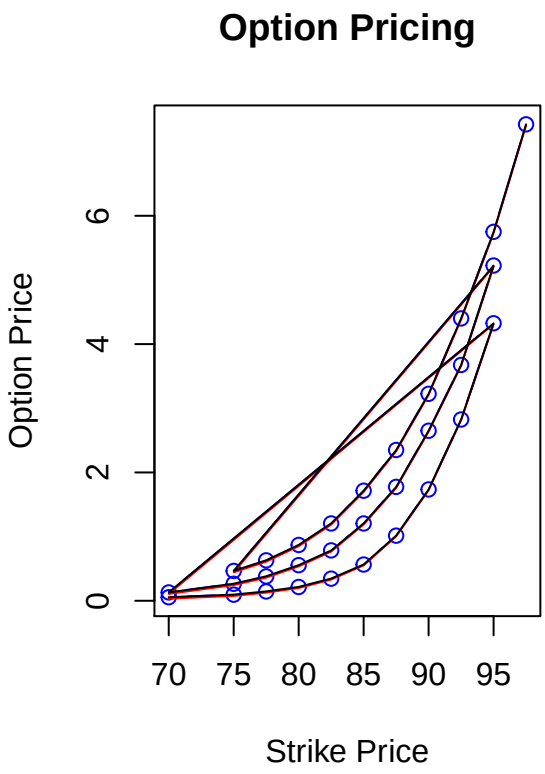
```
lines(strikec,CNFD.C)

plot(strikep,MPrice.P,typ="b", col="blue", main=c("Option Pricing"),
     xlab="Strike Price", ylab="Option Price")
lines(strikep,EFD.P,col = "red")
lines(strikep,IFD.P.col = "green")
```

```
## Warning in plot.xy(xy.coords(x, y), type = type, ...): "IFD.P.col" is not a
## graphical parameter
```

```
lines(strikep,CNFD.P)
```





problem 3

```
library(pracma)
HestonExplicitPDE <- function(params,K,r,div,S,V,Tm) {

    # Finite differences for the Heston PDE for a European Call
    #  Uses even grid sizes
    #  In 'T Hout and Foulon "ADI Finite Difference Schemes for Option Pricing
    #  in the Heston Modelo with Correlation" Int J of Num Analysis and Modeling, 2010.
    #  Thesis by Sensi Li and paper by Vassilis Galiotos
    #  INPUTS
    #     params = 6x1 vector of Heston parameters
    #     K = Strike price
    #     r = risk free rate
    #     div = Dividend yield
    #     S = vector for stock price grid
    #     V = vector for volatility grid
    #     Tm = vector for maturity grid
    # OUTPUT
    #     2-D array of size (nS+1)x(nV+1) for the call price

    # Heston parameters
    kappa = params[1];
```

```
  theta = params[2];
  sigma = params[3];
  v0    = params[4];
  rho   = params[5];
  lambda = params[6];

  # Length of stock, volatility, and maturity
  NS = length(S)
  NV = length(V)
  NT = length(Tm)
  Smin = S[1];  Smax = S[NS];
  Vmin = V[1];  Vmax = V[NV];
  Tmin = Tm[1];  Tmax = Tm[NT];

  # Increment for Stock Price, Volatility, and Maturity
  ds = (Smax-Smin)/(NS-1);
  dv = (Vmax-Vmin)/(NV-1);
  dt = (Tmax-Tmin)/(NT-1);

  # Initialize the 2-D grid with zeros
  U = matrix(0, nrow=NS, ncol=NV);

  # Temporary grid for previous time steps
  u = matrix(0, nrow=NS, ncol=NV);

  # Solve the PDE
  # Round each value of U(S,v,Tm) at each step
  # Boundary condition for tm = Maturity
  for (s in 1:NS) {
      for (v in 1:NV) {
          U[s,v] = max(S[s] - K, 0);
      }
  }
  # Boundary Conditions for everything else
  for (tm in 1:(NT-1))  {
      # Boundary condition for Smin and Smax
      for (v in 1:(NV-1)) {
          U[1,v] = 0
          U[NS,v] = max(0, Smax - K); # Galiotos uses U(NS-1,v) + ds;
      }
      # Boundary condition for Vmax
      for (s in 1:NS) {
         U[s,NV] = max(0, S[s] - K); # Galiotos uses U(s,NV-1);
}
# Update temp grid u[s,tm] with the boundary conditions
u = U
# Boundary condition for Vmin
# Previous time step values are in the temporary grid u(s,tm)
for (s in 2:(NS-1)) {
  DerV = (u[s,2] - u[s,1]) / dv;
  # PDE Points on the middle of the grid (non boundary)
  DerS = (u[s+1,1] - u[s-1,1])/2/ds;
  # Central difference for dU/dS
```

```
    U[s,1] = u[s,1]*(1 - r*dt - kappa*theta*dt/dv) +
    dt*0.5*(r-div)*(s-1)*( u[s+1,1] - u[s-1,1] ) +
    kappa*theta*dt/dv*u[s,2]
  }
  # Update the temporary grid u(s,tm) with the boundary conditions
  u = U
  # Interior points of the grid (non boundary)
  # Previous time step values are in the temporary grid u(s,tm)
  for (s in 2:(NS-1)) {
     for (v in 2:(NV-1)) {
     A = (1 - dt*(s-1)^2*(v-1)*dv - sigma^2*(v-1)*dt/dv - r*dt);
     B = (1/2*dt*(s-1)^2*(v-1)*dv - 1/2*dt*(r-div)*(s-1));
     CC = (1/2*dt*(s-1)^2*(v-1)*dv + 1/2*dt*(r-div)*(s-1));
     D = (1/2*dt*sigma^2*(v-1)/dv - 1/2*dt*kappa*(theta-(v-1)*dv)/dv);
     E = (1/2*dt*sigma^2*(v-1)/dv + 1/2*dt*kappa*(theta-(v-1)*dv)/dv);
     F = 1/4*dt*sigma*(s-1)*(v-1)*rho;
     U[s,v] = A*u[s,v] + B*u[s-1,v] + CC*u[s+1,v] +
     D*u[s,v-1] + E*u[s,v+1] + F*(u[s+1,v+1]+u[s-1,v-1]-u[s-1,v+1]-u[s+1,v-1])
     }
  }
 }
 # Return 2-D array of size (nS+1)x(nV+1)
   U
}
#function return price
runHeston <- function(S0 = 1, K = 0.5, Mat = 5, r = 0, div = 0, v0 = 0.1,
         kappa = 2, theta = 0.1, sigma = 0.2, rho = -0.3, lambda = 0)
{
   #parameter vector
   params = c(kappa, theta, sigma, v0, rho, lambda)
    #set min and max of asset price, volatility and maturity
   Smin = 0; Smax = 2*K;
   Vmin = 0; Vmax = 2*sigma;
   Tmin = 0; Tmax = Mat;
   #set number of grids
   NS = 40 # grid number of price
   NV = 40 # grid number of volatility
   NT = 1000 # grid number of maturity

   # The maturity time increment and grid
   dt = (Tmax-Tmin)/(NT-1);
   Tm = (0:(NT-1))*dt;

   # The asset price increment and grid
   ds = (Smax-Smin)/(NS-1);
   S = (0:(NS-1)) * ds;
   #The volatility increment and grid
   dv = (Vmax-Vmin)/(NV-1);
   V = (0:(NV-1)) * dv;
   # Solve the PDE
   U=HestonExplicitPDE(params,K,r,div,S,V,Tm);
   # return Call option price using interpolation method
   Call = interp2(V, S, U, v0,S0);
```

```
   #return Put option using call-put parity
   Put = Call - S0*exp(-div*Mat) + K*exp(-r*Mat);
   #Return both call and put
   result = c(Call, Put);
   return(result)
}
price1 <- runHeston(K=0.5)
price2 <- runHeston(K=0.75)
price3 <- runHeston(K=1)
price4 <- runHeston(K=1.25)
price5 <- runHeston(K=1.5)
#cat("K=0.5",price1,"\n","K=0.75",price2,"\n","K=1",price3,"\n",
#    "K=1.25",price4,"\n","K=1.5",price5)
```

comment: because the code knit too slow, I list the result with kappa = 2 as follow: call option: k=0.5, price1=0.5; k=0.75, price2=0.3699448; k=1,price3=0.2674649;k=1.25,price4=0.1931855; k=1.5,price5=0.1401597; put option: k=0.5, price1=0.5; k=0.75, price2=0.1199450; k=1,price3=0.2674650;k=1.25,price4=0.443 k=1.5,price5=0.6401597;