621_Homework 3

Liting Hu

Contents

Question 1 1
a) 1
b) 2
c) 3
d) 4
e) 5
f) 7
g) 8
h) 10
Question 2 11
a) 11
b) 14
c) 15
d) 17
Question 3 22

Question 1

a)

Implement the Explicit Finite Difference method to price both European Call and Put options:

```
# European - Explicit Finite Difference method
Option_Ex <- function(isCall=T, S0=100, K=100,
                      Tm=1, sigma=0.25, r=0.06, div=0.03, N=3, Nj=3) {
    # precompute constants
    dt <- Tm/N
    nu <- r - div - 0.5*sigma^2
    #dx <- 0.2
    dx <- sigma*sqrt(3*dt)
    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)

    # initialise asset prices at maturity
    St <- seq(1,2*Nj+1)
    St <- S0*edx^(St-1-Nj)

    # initialise option values at maturity
    temp <- matrix(0, ncol = (N + 1), nrow = (2*Nj + 1))
    if (isCall) {
        C <- temp
        C[, N+1] <- pmax(C[, N+1], St - K)
```

```r
        # step back
        for (i in N:1) {
            for(j in 2:(2*Nj)) {
                C[j, i] <- pu*C[j+1, i+1] + pm*C[j, i+1] + pd*C[j-1, i+1]
            }
            C[1, i] <- C[2, i]
            C[2*Nj+1, i] <- C[2*Nj, i] + (St[2*Nj+1] - St[2*Nj])
        }
        ans <- C[Nj+1, 1]
    }
    else {
        P <- temp
        P[, N+1] <- pmax(P[, N+1], K - St)

        # step back
        for (i in N:1) {
            for(j in 2:(2*Nj)) {
                P[j, i] <- pu*P[j+1, i+1] + pm*P[j, i+1] + pd*P[j-1, i+1]
            }
            P[2*Nj+1, i] <- P[2*Nj, i]
            P[1, i] <- P[2, i] + (St[2] - St[1])
        }
        ans <- P[Nj+1, 1]
    }
    return(ans)
}
```

b)

Implement the Implicit Finite Difference method to price European Call and Put options:

```r
# European - Implicit Finite Difference method
Option_Im <- function(isCall=T, S0=100, K=100,
                      Tm=1, sigma=0.25, r=0.06, div=0.03, N=3, Nj=3) {
    # precompute constants
    dt <- Tm/N
    nu <- r - div - 0.5*sigma^2
    #dx <- 0.2
    dx <- sigma*sqrt(3*dt)
    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)

    # initialise asset prices at maturity
    St <- seq(1, 2*Nj+1)
    St <- S0*edx^(St - 1 - Nj)

    # initialise option values at maturity
    C <- matrix(0, ncol = 2, nrow = (2*Nj + 1))
    if (isCall) {
        C[, 1] <- pmax(C[, 1], St - K)
```

```r
    }
    else {
        C[, 1] <- pmax(C[, 1], K - St)
    }

    #computer derivative boundary condition
    lambda_L <- C[2, 1] - C[1, 1]
    lambda_U <- C[2*Nj+1, 1] - C[2*Nj, 1]

    #step back through lattice
    for(i in (N-1) : 0){
        C <- solve_imp_tridiagonal(C, pu, pm, pd, lambda_L, lambda_U, Nj)
        C[, 1] <- C[, 2]
    }

    ans <- C[Nj+1, 1]
    return(ans)
}

# Solve implicit tridiagonal system
solve_imp_tridiagonal <- function(C,pu,pm,pd,lambda_L,lambda_U,Nj) {
    #substitute boundary condition at j=-Nj into j = -Nj+1
    vj <- Nj + 1
    pmp <- c()
    pp <- c()
    pmp[-Nj+vj+1] <- pm + pd
    pp[-Nj+vj+1] <- C[-Nj+1+vj, 1] + pd*lambda_L

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

    #use bounary condition at j = Nj and equation at j = Nj-1
    C[Nj+vj, 2] <- (pp[Nj-1+vj] + pmp[Nj-1+vj]*lambda_U)/(pu + pmp[Nj-1+vj])
    C[Nj-1+vj, 2] <- C[Nj+vj, 2] - lambda_U

    #back substitution
    for( j in (Nj-2):-Nj){
        C[j+vj, 2] = (pp[j+vj] - pu*C[j+1+vj, 2])/pmp[j+vj]
    }
    return(C)
}
```

c)

Assume

$$\epsilon \approx \Delta x^2 + \Delta t$$

Since the convergence condition is

$$\Delta x \geq \sigma\sqrt{3\Delta t}$$

so we have

$$\Delta t \leq \frac{\Delta x^2}{3\sigma^2}$$

$$\varepsilon \leq \left(1 + \frac{1}{3\sigma^2}\right) \Delta x^2 < 0.001$$

$$\Delta x < \frac{1}{10}\sqrt{\frac{3\sigma^2}{(3\sigma^2+1)10}}$$

From Les Clewlow and Chris Strickland, a reasonable range of asset price values at the maturity date of the option is three standard deviations either side of the mean. (

$$n_{SD} = 3 \times 2 = 6$$

)

$$\Delta x = \frac{n_{SD}\sigma\sqrt{T}}{2N_j + 1} < \frac{1}{10}\sqrt{\frac{3\sigma^2}{10(3\sigma^2+1)}}$$

$$N_j > 5 n_{SD}\sigma\sqrt{\frac{10(3\sigma^2+1)T}{3\sigma^2}} - \frac{1}{2}$$

Under these conditon,

$$\Delta t \leq \frac{\Delta x^2}{3\sigma^2} = \frac{0.001}{(3\sigma^2+1)}$$

$$N = 1/\Delta t \geq 1000\left(3\sigma^2 + 1\right)$$

With parameters given by (d):

$$\begin{aligned} &\Delta x = 0.01256562 \\ &N_j = 60 \\ &\Delta t = 0.0008421053 \\ &N = 1221 \end{aligned}$$

d)

```
epsilon <- 0.001
n <- 6
sigma <- 0.25
Nj <- ceiling(((n*sigma)/sqrt(0.001/(1+1/3/sigma^2))-1)/2)
Nj
```

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

```
N <- ceiling(3*((2*Nj+1)/n)^2)
N
```

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

```
Ex.C <- Option_Ex(N = N, Nj = Nj)
Ex.P <- Option_Ex(isCall = F, N = N, Nj = Nj)
Im.C <- Option_Im(N = N, Nj = Nj)
Im.P <- Option_Im(isCall = F, N = N, Nj = Nj)

re <- data.frame(c(Ex.C, Im.C), c(Ex.P, Im.P))
colnames(re) <- c("Call", "Put")
rownames(re) <- c("Explicit", "Implicit")
```

Prices under these four schemes are showed below. N is 60 and N_j is 1221, as calculated in part c.

```r
knitr::kable(re, caption = "Finite Difference methods")
```

Table 1: Finite Difference methods

	Call	Put
Explicit	11.01164	8.143260
Implicit	11.00928	8.141098

e)

```r
# BSM model
Option_BSM <- function(isCall = T, S0=100, K=100, Tm=1, sigma=.25, r=0.06, div=0.03) {
    d1 <- (log(S0/K) + (r - div + sigma^2/2)*Tm)/sigma/sqrt(Tm)
    d2 <- d1 - sigma*sqrt(Tm)
    if (isCall) {p <- S0*exp(-div*Tm)*pnorm(d1) - K*exp(-r*Tm)*pnorm(d2)}
    else {p <- K*exp(-r*Tm)*pnorm(-d2) - S0*exp(-div*Tm)*pnorm(-d1)}
    return(p)
}
```

Iterative procedure:

```r
# Explicit-Call ----
Nj.count <- 10
dif <- 1
bsp <- Option_BSM()
while (dif > epsilon) {
    Nj.count <- Nj.count + 1
    N <- ceiling(3*((2*Nj.count+1)/n)^2)
    op <- Option_Ex(N = N, Nj = Nj.count)
    dif <-  abs(op - bsp)
}
Ex.C <- op
Ex.C.Nj <- Nj.count
Ex.C.N <- N
```

```r
# Implicit-Call ----
Nj.count <- 10
dif <- 1
while (dif > epsilon) {
    Nj.count <- Nj.count + 10
    # When Nj is too large (150+), the implicit fucntion runs very slow.
    # So at first let the step length be 10
    N <- ceiling(3*((2*Nj.count+1)/n)^2)
    op <- Option_Im(N = N, Nj = Nj.count)
    dif <-  abs(op - bsp)
}
dif <- 1
Nj.count <- Nj.count - 10
while (dif > epsilon) {
    Nj.count <- Nj.count + 1
    N <- ceiling(3*((2*Nj.count+1)/n)^2)
    op <- Option_Im(N = N, Nj = Nj.count)
```

```r
    dif <-  abs(op - bsp)
}
Im.C <- op
Im.C.Nj <- Nj.count
Im.C.N <- N
```

```r
# Explicit-Put ----
Nj.count <- 10
dif <- 1
bsp <- Option_BSM(isCall = F)
while (dif > epsilon) {
    Nj.count <- Nj.count + 1
    N <- ceiling(3*((2*Nj.count+1)/n)^2)
    op <- Option_Ex(isCall = F, N = N, Nj = Nj.count)
    dif <-  abs(op - bsp)
}
Ex.P <- op
Ex.P.Nj <- Nj.count
Ex.P.N <- N
```

```r
# Implicit-Put ----
Nj.count <- 10
dif <- 1
while (dif > epsilon) {
    Nj.count <- Nj.count + 10
    # the step length is 10 as the same reason above
    N <- ceiling(3*((2*Nj.count+1)/n)^2)
    op <- Option_Im(isCall = F, N = N, Nj = Nj.count)
    dif <- abs(op - bsp)
}
dif <- 1
Nj.count <- Nj.count - 10
while (dif > epsilon) {
    Nj.count <- Nj.count + 1
    N <- ceiling(3*((2*Nj.count+1)/n)^2)
    op <- Option_Im(isCall = F, N = N, Nj = Nj.count)
    dif <-  abs(op - bsp)
}
Im.P <- op
Im.P.Nj <- Nj.count
Im.P.N <- N
```

```r
result2 <- data.frame(c(Ex.C, Im.C), c(Ex.P, Im.P))
colnames(result2) <- c("Call", "Put")
rownames(result2) <- c("Explicit", "Implicit")

Nj <- c(Ex.C.Nj, Im.C.Nj, Ex.P.Nj, Im.P.Nj)
N <- c(Ex.C.N, Im.C.N, Ex.P.N, Im.P.N)
dt <- 1/Nj
dx <- 0.25*sqrt(3*dt)
steps <- data.frame(Nj, N, dt, dx)
rownames(steps) <- c("Explicit.Call", "Implicit.Call", "Explicit.Put", "Implicit.Put")
```

Results are showed below, all of them are close to BSM model.

```r
knitr::kable(result2, caption = "Finite Difference methods")
```

Table 2: Finite Difference methods

	Call	Put
Explicit	11.01209	8.143999
Implicit	11.01209	8.143995
N, Nj, dt,	dx are silg	htly different from theoretical answers.

```r
knitr::kable(steps, caption = "Finite Difference methods")
```

Table 3: Finite Difference methods

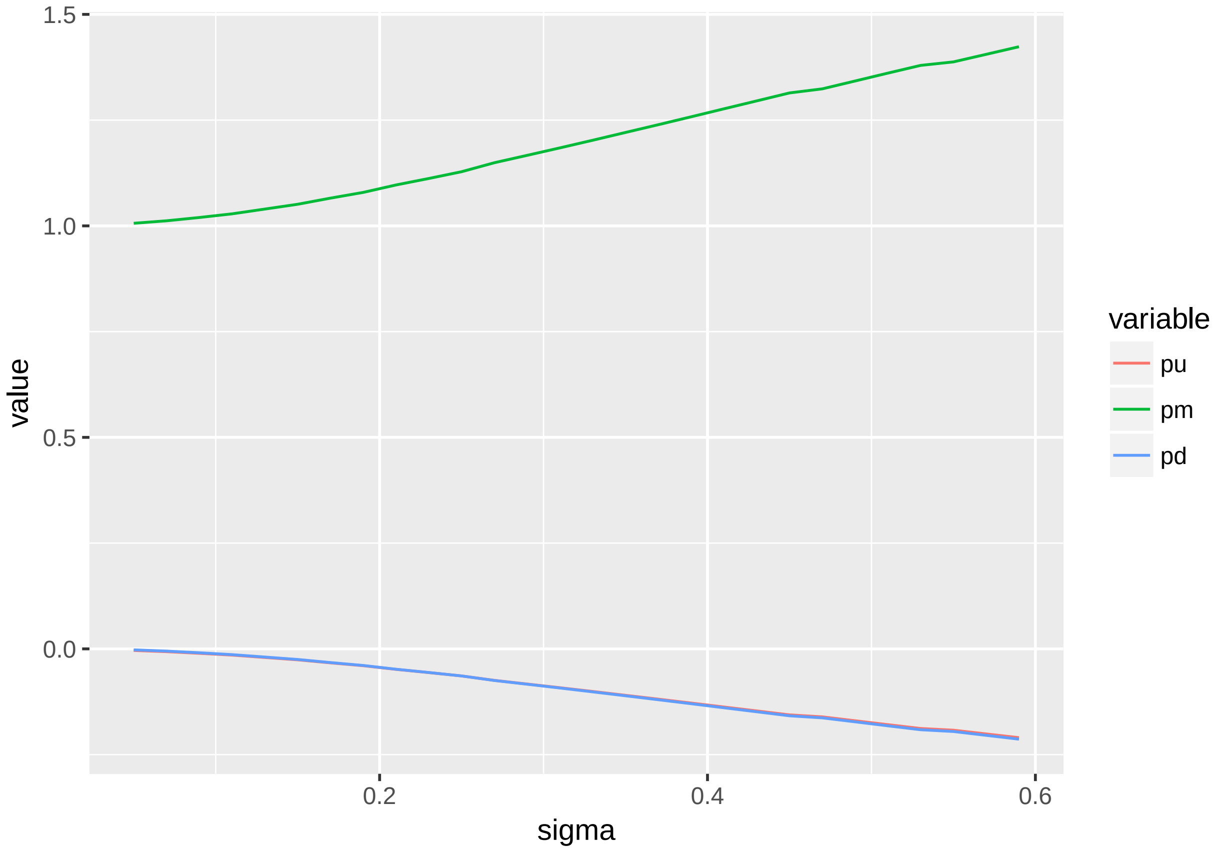
	Nj	N	dt	dx
Explicit.Call	71	1705	0.0140845	0.0513892
Implicit.Call	112	4219	0.0089286	0.0409159
Explicit.Put	79	2107	0.0126582	0.0487177
Implicit.Put	118	4681	0.0084746	0.0398621

f)

```r
sigmas <- seq(0.05, 0.6, by = 0.02)
Tm <- 1
div <- 0.03
r <- 0.06
pu <- c()
pm <- c()
pd <- c()

for (s in sigmas) {
    Nj <- ceiling(((n*s)/sqrt(0.001/(1+1/3/s^2))-1)/2)
    N = ceiling(3*((2*Nj+1)/n)^2)
    dt <- Tm/N
    #dx <- s*sqrt(3*dt)
    dx <- 0.02
    nu <- r - div - 0.5*s^2
    pu = c(pu, -0.5*dt*((s/dx)^2 + nu/dx))
    pm = c(pm, 1.0 + dt*(s/dx)^2 + r*dt)
    pd = c(pd, -0.5*dt*((s/dx)^2 - nu/dx))
}

ps <- data.frame("sigma" = sigmas, pu, pm, pd)
ps.long <- melt(ps, id="sigma")
ggplot(data = ps.long, aes(x = sigma, y=value, colour=variable)) + geom_line()
```



pu and pd have a similar path which has a tendency to decrease from zero. While pm is increasing as sigma grows larger.

g)

Implement the Crank-Nicolson Finite Difference method and price both European Call and Put options:

```
# European - The Crank-Nicolson Finite Difference method
Option_CN <- function(isCall=T, S0=100, K=100,
                      Tm=1, sigma=0.25, r=0.06, div=0.03, N=3, Nj=3) {
    # precompute constants
    dt <- Tm/N
    nu <- r - div - 0.5*sigma^2
    #dx <- 0.2
    dx <- sigma*sqrt(3*dt)
    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)

    # initialise asset prices at maturity
    St <- seq(1, 2*Nj+1)
    St <- S0*edx^(St - 1 - Nj)

    # initialise option values at maturity
    C <- matrix(0, ncol = 2, nrow = (2*Nj + 1))
```

```
    if (isCall) {
        C[, 1] <- pmax(C[, 1], St - K)
    }
    else {
        C[, 1] <- pmax(C[, 1], K - St)
    }

    # computer derivative boundary condition
    lambda_L <- C[2, 1] - C[1, 1]
    lambda_U <- C[2*Nj+1, 1] - C[2*Nj, 1]

    # step back through lattice
    for(i in (N-1) : 0){
        C <- solve_CK_tridiagonal(C, pu, pm, pd, lambda_L, lambda_U, Nj)
        C[, 1] <- C[, 2]
    }
    return(C[Nj+1, 1])
}

# Solve CK tridiagonal system
solve_CK_tridiagonal <- function(C, pu, pm, pd, lambda_L, lambda_U, Nj){

    #substitute boundary condition at j=-Nj into j = -Nj+1
    vj <- Nj + 1
    pmp <- c(); pp <- c()
    pmp[-Nj+vj+1] <- pm + pd
    pp[-Nj+vj+1] <- (-pu*C[-Nj+2+vj, 1])-(pm-2)*C[-Nj+1+vj, 1] - pd*C[-Nj+vj, 1] + pd*lambda_L

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

    #use bounary condition at j = Nj and equation at j = Nj-1
    C[Nj+vj, 2] <- (pp[Nj-1+vj] + pmp[Nj-1+vj]*lambda_U)/(pu + pmp[Nj-1+vj])
    C[Nj-1+vj, 2] <- C[Nj+vj, 2] - lambda_U

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

sigma <- 0.25
Nj <- ceiling(((n*sigma)/sqrt(0.001/(1+1/3/sigma^2))-1)/2)
N <- ceiling(3*((2*Nj+1)/n)^2)

CN.C <- Option_CN(N = N, Nj = Nj)
CN.P <- Option_CN(isCall = F, N = N, Nj = Nj)
result3 <- rbind(re, c(CN.C, CN.P))
```

```r
rownames(result3) <- c("EFD", "IFD", "CNFD")
```

```r
knitr::kable(result3, caption = "Finite Difference methods")
```

Table 4: Finite Difference methods

	Call	Put
EFD	11.01164	8.143260
IFD	11.00928	8.141098
CNFD	11.01046	8.142179

Prices by Crank-Nicolson are between what we calculate by explicit and implicit methods. All these three method have similar results.

h)

```r
# calculate hedge sensitivities
Greeks <- function(isCall=T, S0=100, K=100,
                   Tm=1, sigma=0.25, r=0.06, div=0.03, N=3, Nj=3) {
    # Output: Delta, Gamma, Theta, and Vega

    # precompute constants
    dt <- Tm/N
    nu <- r - div - 0.5*sigma^2
    #dx <- 0.2
    dx <- sigma*sqrt(3*dt)
    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)

    dsigma <- 0.0001*sigma # to compute vega, define delta sigma as a small fraction

    # initialise asset prices at maturity
    St <- seq(1, 2*Nj+1)
    St <- S0*edx^(St-1-Nj)

    # initialise option values at maturity
    temp <- matrix(0, ncol = (N + 1), nrow = (2*Nj + 1))
    if (isCall) {
        C <- temp
        C[, N+1] <- pmax(C[, N+1], St - K)

        # step back
        for (i in N:1) {
            for(j in 2:(2*Nj)) {
                C[j, i] <- pu*C[j+1, i+1] + pm*C[j, i+1] + pd*C[j-1, i+1]
            }
            C[1, i] <- C[2, i]
            C[2*Nj+1, i] <- C[2*Nj, i] + (St[2*Nj+1] - St[2*Nj])
```

```
        }
    }
    else {
        P <- temp
        P[, N+1] <- pmax(P[, N+1], K - St)

        # step back
        for (i in N:1) {
            for(j in 2:(2*Nj)) {
                P[j, i] <- pu*P[j+1, i+1] + pm*P[j, i+1] + pd*P[j-1, i+1]
            }
            P[2*Nj+1, i] <- P[2*Nj, i]
            P[1, i] <- P[2, i] + (St[2] - St[1])
        }
        C <- P
    }

    delta <- (C[Nj+2, 1] - C[Nj, 1])/(St[Nj+2] - St[Nj])
    gamma <- 2*((C[Nj+2, 1] - C[Nj+1, 1])/(St[Nj+2] - St[Nj+1]) -
                    (C[Nj+1, 1] - C[Nj, 1])/(St[Nj+1] - St[Nj]))/(St[Nj+2] - St[Nj])
    theta <- (C[Nj+1, 2] - C[Nj+1, 1])/dt
    vega <- (Option_Ex(isCall, S0, K, Tm, sigma + dsigma, r, div, N, Nj) -
        Option_Ex(isCall, S0, K, Tm, sigma - dsigma, r, div, N, Nj))/2/dsigma
    result <- c(delta, gamma, theta, vega)
    return(result)
}
```

Using parameters in (d), for a European option: Delta, Gamma, Theta, and Vega are

```
Greeks(N=N, Nj=Nj)
```

```
## [1]  0.57920467  0.01503145 -5.77668807 37.56433159
```

Question 2

a)

Doing same thing as in HW1:

```
# Download option prices
setwd("/Users/apple/Desktop/621/HW3")
readfile <- try(read.csv("Calls.csv"), TRUE)
if (inherits(readfile, "try-error")) {
    # Get option chains from Yahoo finance
    AAPL.OPTS <- getOptionChain("AAPL", NULL)
    C1 <- AAPL.OPTS$Apr.21.2017$calls
    P1 <- AAPL.OPTS$Apr.21.2017$puts
    C1$Ave.Price <- (C1$Bid + C1$Ask)/2
    P1$Ave.Price <- (P1$Bid + P1$Ask)/2
    C1 <- C1[, c(1, 4, 5, 8)]
    P1 <- P1[, c(1, 4, 5, 8)]
    C2 <- AAPL.OPTS$May.19.2017$calls
    P2 <- AAPL.OPTS$May.19.2017$puts
    C2$Ave.Price <- (C2$Bid + C2$Ask)/2
```

```
    P2$Ave.Price <- (P2$Bid + P2$Ask)/2
    C2 <- C2[, c(1, 4, 5, 8)]
    P2 <- P2[, c(1, 4, 5, 8)]
    C3 <- AAPL.OPTS$Jun.16.2017$calls
    P3 <- AAPL.OPTS$Jun.16.2017$puts
    C3$Ave.Price <- (C3$Bid + C3$Ask)/2
    P3$Ave.Price <- (P3$Bid + P3$Ask)/2
    C3 <- C3[, c(1, 4, 5, 8)]
    P3 <- P3[, c(1, 4, 5, 8)]

    temp <- merge(C1, C2, by = "Strike")
    calls <- merge(temp, C3, by = "Strike")
    temp <- merge(P1, P2, by = "Strike")
    puts <- merge(temp, P3, by = "Strike")
    coln <- c("Strike", "Bid1", "Ask1", "Apr.21.2017", "Bid2", "Ask2",
              "May.19.2017", "Bid3", "Ask3", "Jun.16.2017")
    colnames(calls) <- coln
    colnames(puts) <- coln
    calls <- calls[9:18, ]
    puts <- puts[10:19, ]
    print(calls)
    print(puts)
    write.csv(calls, file = "Calls.csv", row.names = F)
    write.csv(puts, file = "Puts.csv", row.names = F)
} else {
    calls <- read.csv("Calls.csv")
    puts <- read.csv("Puts.csv")
}

# todaystock <- getQuote("AAPL")
# S_0 <- todaystock[, 2] # The value of underlying
# When the option data are downloaded, the stock price is 140.26
S_0 <- 140.26

tau <- c(24/252, 44/252, 62/252) # time to maturity
r <- 0.0075 # the current short-term interest rate
fsigma <- function(isCall = T, sigma, K_i, maturity_i) {
    # Epsilon. To calculate implied vol
    S0 <- S_0
    if (isCall) {prices <- calls}
    else {prices <- puts}
    p <- prices[K_i, 3*maturity_i + 1]
    K <- calls[K_i, 1]
    price.by.bs <- Option_BSM(isCall, S0, K, tau[maturity_i], sigma, r)
    ans <- price.by.bs - p
    return(ans)
}

IV.Calls <- calls[, c(1, 4, 7, 10)]
IV.PUTS <- puts[, c(1, 4, 7, 10)]
for(i in 1:10) {
    for(j in 1:3) {
        # use bisection method to calculate implied vols
```

```
        a <- 0.001
        b <- 100
        epsilon <- abs(a - b)
        while(epsilon > 1e-4) {
            mid <- (a + b)/2
            if (fsigma(T, a, i, j)*fsigma(T, mid, i, j) < 0 ) b <- mid
            else a <- mid
            epsilon <- abs(a - b)
        }
        IV.Calls[i, j+1] <- a
        a <- 0.001
        b <- 100
        epsilon <- abs(a - b)
        while(epsilon > 1e-4) {
            mid <- (a + b)/2
            if (fsigma(F, a, i, j)*fsigma(F, mid, i, j) < 0 ) b <- mid
            else a <- mid
            epsilon <- abs(a - b)
        }
        IV.PUTS[i, j+1] <- a
    }
}
IV.Calls[,2:4][IV.Calls[,2:4]>99]=NaN
IV.PUTS[,2:4][IV.PUTS[,2:4]>99]=NaN # Implied vols calculated as in HW1
```

Implied vols are showed below:

```
knitr::kable(IV.Calls, caption = "Implied vols for Call Option")
```

Table 5: Implied vols for Call Option

Strike	Apr.21.2017	May.19.2017	Jun.16.2017
115	0.4026836	0.3594826	0.3218128
120	0.3400278	0.3123716	0.2821404
125	0.2631624	0.2652605	0.2515277
130	0.2175773	0.2302610	0.2251112
135	0.1694172	0.2104248	0.2096619
140	0.1503439	0.1962152	0.1975503
145	0.1427146	0.1881090	0.1917330
150	0.1512022	0.1863924	0.1887766
155	0.1697033	0.1898256	0.1897303
160	0.1907793	0.2000298	0.1938310

```
knitr::kable(IV.PUTS, caption = "Implied vols for Put Option")
```

Table 6: Implied vols for Put Option

Strike	Apr.21.2017	May.19.2017	Jun.16.2017
110	0.2813774	0.2470455	0.2293073
115	0.2446614	0.2194846	0.2047028
120	0.2007928	0.1931635	0.1821963
125	0.1556844	0.1642674	0.1593084
130	0.1058078	0.1352760	0.1358482

Strike	Apr.21.2017	May.19.2017	Jun.16.2017
135	0.0431520	0.0986553	0.1063800
140	NaN	NaN	NaN
150	NaN	0.1795260	0.1697033
155	NaN	0.1858202	0.1633137
160	NaN	0.1853434	0.1698940

b)

```
FD.calls <- calls
FD.puts <- puts
for (i in 1:3) {
    Strike.C <- IV.Calls[, 1]
    Strike.P <- IV.PUTS[, 1]
    for (j in 1:10) {
        sigma.C <- IV.Calls[j, i+1]
        sigma.P <- IV.PUTS[j, i+1]

        # Theoretical N and Nj to make error less than 0.001
        Nj.C <- ceiling(((n*sigma.C)/sqrt(0.001/(1+1/3/sigma.C^2))-1)/2)
        N.C <- ceiling(3*((2*Nj.C+1)/n)^2)
        Nj.P <- ceiling(((n*sigma.P)/sqrt(0.001/(1+1/3/sigma.P^2))-1)/2)
        N.P <- ceiling(3*((2*Nj.P+1)/n)^2)

        # compute all things we need
        if (is.na(sigma.C)) {
            FD.calls[j, (3*i-1):(3*i+1)] <- c(NaN, NaN, NaN)
        } else {
            FD.calls[j, 3*i-1] <- Option_Ex(S0 = S_0, K = Strike.C[j], Tm = tau[i],
                                            sigma = sigma.C, div = 0, N = N.C, Nj = Nj.C)
            FD.calls[j, 3*i] <- Option_Im(S0 = S_0, K = Strike.C[j], Tm = tau[i],
                                          sigma = sigma.C, div = 0, N = N.C, Nj = Nj.C)
            FD.calls[j, 3*i+1] <- Option_CN(S0 = S_0, K = Strike.C[j], Tm = tau[i],
                                            sigma = sigma.C, div = 0, N = N.C, Nj = Nj.C)
        }

        if (is.na(sigma.P)) {
            FD.puts[j, (3*i-1):(3*i+1)] <- c(NaN, NaN, NaN)
        } else {
            FD.puts[j, 3*i-1] <- Option_Ex(isCall = F, S0 = S_0, K = Strike.P[j], Tm = tau[i],
                                           sigma = sigma.P, div = 0, N = N.P, Nj = Nj.P)
            FD.puts[j, 3*i] <- Option_Im(isCall = F, S0 = S_0, K = Strike.P[j], Tm = tau[i],
                                         sigma = sigma.P, div = 0, N = N.P, Nj = Nj.P)
            FD.puts[j, 3*i+1] <- Option_CN(isCall = F, S0 = S_0, K = Strike.P[j], Tm = tau[i],
                                           sigma = sigma.P, div = 0, N = N.P, Nj = Nj.P)
        }
    }
}

coln11 <- c("Strikes", "EFD1", "IFD1", "CNFD1",
            "EFD2", "IFD2", "CNFD2",
            "EFD3", "IFD3", "CNFD3")
```

```r
# "1": maturity at Apr.21.2017; "2": maturity at May.19.2017; "3": maturity at Jun.16.2017.
colnames(FD.calls) <- coln11
colnames(FD.puts) <- coln11
```

All prices are showed below. For call and put options with same parameters, EFD < CNFD < IFD.

```r
knitr::kable(FD.calls, caption = "Prices of Call Option under Finite Difference methods")
```

Table 7: Prices of Call Option under Finite Difference methods

Strikes	EFD1	IFD1	CNFD1	EFD2	IFD2	CNFD2	EFD3	IFD3	C
115	26.2442479	26.2446193	26.2444336	27.1657112	27.1659754	27.1658432	27.7909872	27.7911483	27.79
120	21.3051521	21.3054591	21.3053056	22.3169532	22.3170696	22.3170113	22.9865207	22.9864932	22.98
125	16.2946716	16.2949015	16.2947865	17.4981196	17.4980286	17.4980740	18.3519631	18.3516458	18.35
130	11.4911052	11.4911323	11.4911187	12.9159820	12.9155651	12.9157734	13.9235491	13.9228671	13.92
135	6.8313319	6.8310441	6.8311879	8.8530152	8.8521741	8.8525946	10.0044844	10.0033983	10.00
140	3.1493963	3.1488126	3.1491045	5.4698114	5.4688057	5.4693086	6.6697733	6.6685687	6.66
145	1.0141693	1.0140576	1.0141133	3.0184531	3.0177819	3.0181175	4.1623013	4.1613939	4.16
150	0.2968091	0.2971749	0.2969920	1.5315199	1.5314952	1.5315074	2.4257675	2.4254919	2.42
155	0.1103227	0.1107484	0.1105357	0.7554071	0.7558981	0.7556525	1.3700761	1.3704341	1.37
160	0.0513264	0.0516857	0.0515061	0.4085293	0.4092408	0.4088851	0.7756126	0.7763782	0.77

```r
knitr::kable(FD.puts, caption = "Prices of Put Option under Finite Difference methods")
```

Table 8: Prices of Put Option under Finite Difference methods

Strikes	EFD1	IFD1	CNFD1	EFD2	IFD2	CNFD2	EFD3	IFD3	CNF
110	0.0089948	0.0091364	0.0090656	0.0298012	0.0300260	0.0299136	0.0568181	0.0570997	0.05695
115	0.0110931	0.0112389	0.0111660	0.0451849	0.0454259	0.0453055	0.0825374	0.0828188	0.08267
120	0.0119600	0.0120925	0.0120263	0.0747181	0.0749638	0.0748410	0.1320101	0.1322619	0.13213
125	0.0123456	0.0124574	0.0124015	0.1193623	0.1195638	0.1194631	0.2168942	0.2170384	0.21696
130	0.0089006	0.0089712	0.0089359	0.2081243	0.2081964	0.2081603	0.3721276	0.3720604	0.37209
135	0.0004073	0.0004151	0.0004112	0.3289484	0.3288009	0.3288745	0.5957460	0.5954114	0.59557
140	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
150	NaN	NaN	NaN	9.5827990	9.5828467	9.5828227	9.5126319	9.5125780	9.51260
155	NaN	NaN	NaN	13.8284533	13.8289794	13.8287163	13.3633848	13.3640115	13.36369
160	NaN	NaN	NaN	18.3635828	18.3642952	18.3639391	17.8672439	17.8681349	17.86768

c)

```r
G.calls <- c()
G.puts <- c()
for (i in 1:10) {
    Strike.C <- IV.Calls[i, 1]
    Strike.P <- IV.PUTS[i, 1]
    G.c <- c()
    G.p <- c()
    for(j in 1:3) {
        sigma.C <- IV.Calls[i, j+1]
        sigma.P <- IV.PUTS[i, j+1]
```

```r
        # Theoretical N and Nj to make error less than 0.001
        Nj.C <- ceiling(((n*sigma.C)/sqrt(0.001/(1+1/3/sigma.C^2))-1)/2)
        N.C <- ceiling(3*((2*Nj.C+1)/n)^2)
        Nj.P <- ceiling(((n*sigma.P)/sqrt(0.001/(1+1/3/sigma.P^2))-1)/2)
        N.P <- ceiling(3*((2*Nj.P+1)/n)^2)

        if (is.na(sigma.C)) {
            G.c <- c(G.c, NaN, NaN, NaN, NaN)
        } else {
            G.c <- c(G.c, Greeks(T, S_0, Strike.C, tau[j], sigma.C, r, 0, N.C, Nj.C))
        }

        if (is.na(sigma.P)) {
            G.p <- c(G.p, NaN, NaN, NaN, NaN)
        } else {
            G.p <- c(G.p, Greeks(F, S_0, Strike.P, tau[j], sigma.P, r, 0, N.P, Nj.P))
        }
    }
    G.calls <- rbind(G.calls, G.c)
    G.puts <- rbind(G.puts, G.p)
}
coln22 <- c("Delta1", "Gamma1", "Theta1", "Vega1",
            "Delta2", "Gamma2", "Theta2", "Vega2",
            "Delta3", "Gamma3", "Theta3", "Vega3")
# "1": maturity at Apr.21.2017; "2": maturity at May.19.2017; "3": maturity at Jun.16.2017.

colnames(G.calls) <- coln22
colnames(G.puts) <- coln22

G.calls <- cbind(calls[1], G.calls)
G.puts <- cbind(puts[1], G.puts)
```

Delta, Gamma, Theta, and Vega are showed below. P.S. “1”: maturity at Apr.21.2017; “2”: maturity at May.19.2017; “3”: maturity at Jun.16.2017.

```r
knitr::kable(G.calls, caption = "Call option greeks by Explicit Finite Difference methods")
```

Table 9: Call option greeks by Explicit Finite Difference methods

Strike	Delta1	Gamma1	Theta1	Vega1	Delta2	Gamma2	Theta2	Vega2	Delta3	
115	0.9520812	0.0057184	-9.925030	4.452174	0.9200489	0.0070508	-9.731476	8.900580	0.9090403	0
120	0.9389065	0.0082066	-10.161255	5.396507	0.8979971	0.0097228	-10.115031	10.676649	0.8844641	0
125	0.9288126	0.0119336	-8.985269	5.927487	0.8656843	0.0139142	-10.416809	12.981470	0.8413502	0
130	0.8799971	0.0212294	-10.728324	8.626587	0.8025864	0.0205751	-11.486767	16.382312	0.7740888	0
135	0.7795643	0.0404174	-12.186215	12.592527	0.6891301	0.0286311	-13.139425	20.691564	0.6690153	0
140	0.5313213	0.0610996	-14.128892	17.224915	0.5317355	0.0345700	-13.621001	23.308250	0.5345879	0
145	0.2369937	0.0499591	-10.253880	13.209261	0.3569468	0.0338240	-12.133360	21.739732	0.3888480	0
150	0.0808603	0.0228828	-5.226620	6.521595	0.2101664	0.0263840	-9.229074	17.125484	0.2577628	0
155	0.0309785	0.0095125	-2.723334	3.037876	0.1145084	0.0173820	-6.275235	11.380940	0.1599701	0
160	0.0142659	0.0044134	-1.591875	1.561383	0.0646279	0.0107602	-4.297694	7.520446	0.0965320	0

```r
knitr::kable(G.puts, caption = "Put option greeks by Explicit Finite Difference methods")
```

Table 10: Put option greeks by Explicit Finite Difference methods

Strike	Delta1	Gamma1	Theta1	Vega1	Delta2	Gamma2	Theta2	Vega2	Delta3
110	-0.0030363	0.0008022	-0.6196206	0.1982936	-0.0079905	0.0015274	-0.9065045	1.232705	-0.0136730
115	-0.0042122	0.0012123	-0.7073320	0.4065414	-0.0131359	0.0026351	-1.2324337	2.057030	-0.0215269
120	-0.0055753	0.0018732	-0.7350763	0.5833626	-0.0234132	0.0048874	-1.7662675	3.290418	-0.0366355
125	-0.0076499	0.0031668	-0.7451336	0.9227589	-0.0417539	0.0092639	-2.4117349	5.073662	-0.0642418
130	-0.0092185	0.0054666	-0.5909194	1.0843617	-0.0815703	0.0190163	-3.3337285	8.719314	-0.1174381
135	-0.0027782	0.0048605	-0.0857772	0.0974245	-0.1637755	0.0427018	-3.9124072	14.449805	-0.2161222
140	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
150	NaN	NaN	NaN	NaN	-0.7994337	0.0266431	-7.5260933	16.170059	-0.7682586
155	NaN	NaN	NaN	NaN	-0.8907988	0.0171676	-4.7788410	10.666668	-0.8789146
160	NaN	NaN	NaN	NaN	-0.9498839	0.0095077	-2.0625628	5.866024	-0.9329298

d)

```r
a.big.df <- as.data.frame(matrix(rep(0, 10), nrow = 1))
coln33 <- c("T", "Strike", "Type", "Ask", "Bid",
            "Market", "implied vol", "EFD", "IFD", "CNFD")
colnames(a.big.df) <- coln33
for (i in 1:3) {
    df1 <- cbind(tau[i], calls[, 1], "Call", calls[, (3*i-1):(3*i+1)],
                 IV.Calls[, i+1], FD.calls[, (3*i-1):(3*i+1)])
    colnames(df1) <- coln33
    df2 <- cbind(tau[i], puts[, 1], "Put", puts[, (3*i-1):(3*i+1)],
                 IV.PUTS[, i+1], FD.puts[, (3*i-1):(3*i+1)])
    colnames(df2) <- coln33
    a.big.df <- rbind(a.big.df, df1, df2)
}
a.big.df <- a.big.df[-1, ]
```

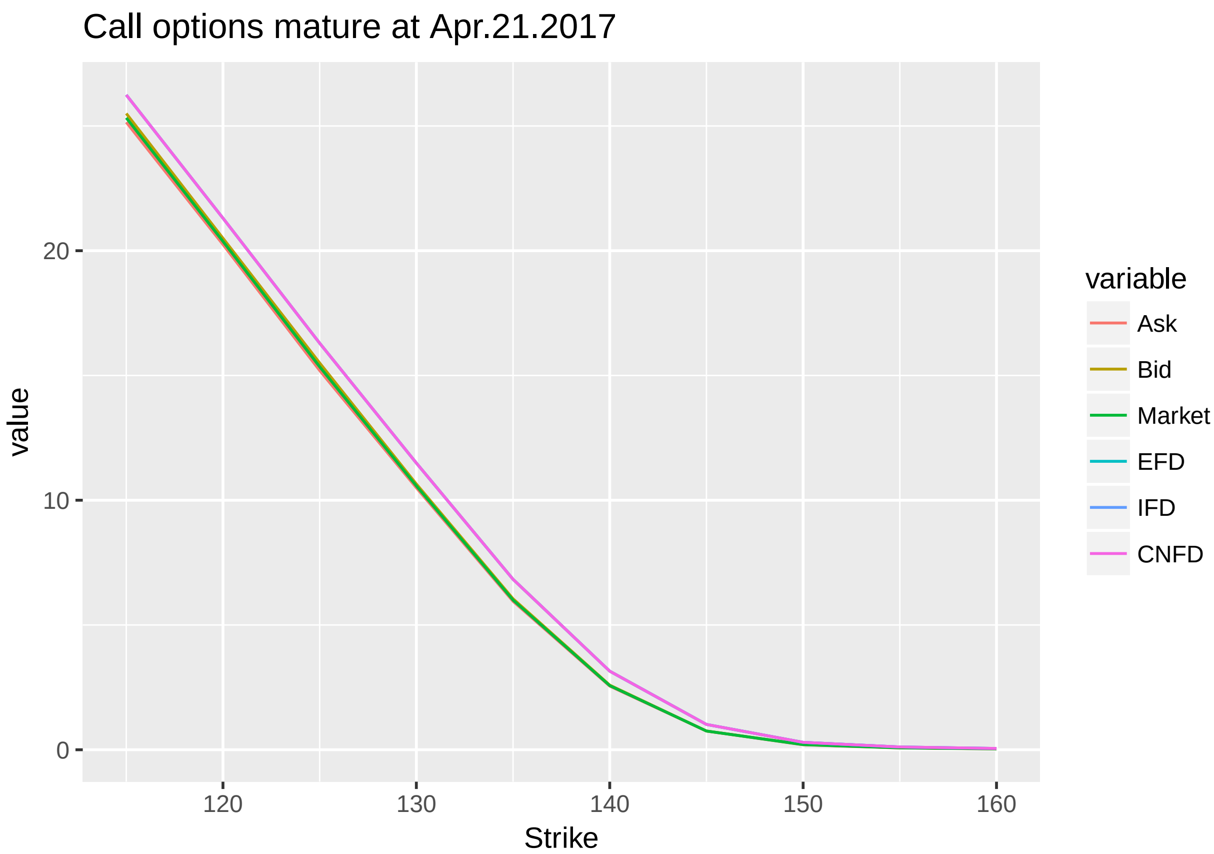
```r
knitr::kable(a.big.df)
```

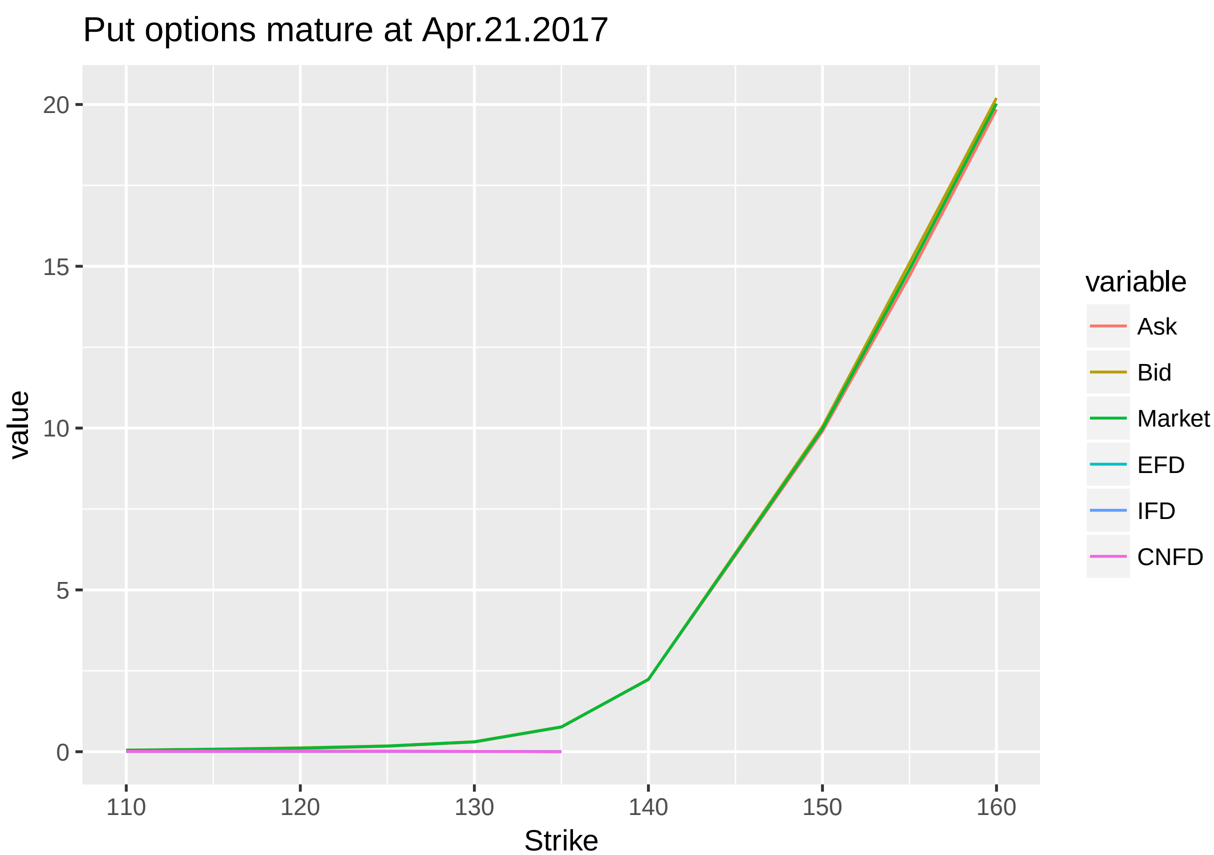
	T	Strike	Type	Ask	Bid	Market	implied vol	EFD	IFD	CNFD
2	0.0952381	115	Call	25.15	25.50	25.325	0.4026836	26.2442479	26.2446193	26.2444336
3	0.0952381	120	Call	20.25	20.50	20.375	0.3400278	21.3051521	21.3054591	21.3053056
4	0.0952381	125	Call	15.20	15.50	15.350	0.2631624	16.2946716	16.2949015	16.2947865
5	0.0952381	130	Call	10.50	10.65	10.575	0.2175773	11.4911052	11.4911323	11.4911187
6	0.0952381	135	Call	5.95	6.05	6.000	0.1694172	6.8313319	6.8310441	6.8311879
7	0.0952381	140	Call	2.55	2.59	2.570	0.1503439	3.1493963	3.1488126	3.1491045
8	0.0952381	145	Call	0.75	0.75	0.750	0.1427146	1.0141693	1.0140576	1.0141133
9	0.0952381	150	Call	0.20	0.21	0.205	0.1512022	0.2968091	0.2971749	0.2969920
10	0.0952381	155	Call	0.07	0.08	0.075	0.1697033	0.1103227	0.1107484	0.1105357
11	0.0952381	160	Call	0.03	0.04	0.035	0.1907793	0.0513264	0.0516857	0.0515061
12	0.0952381	110	Put	0.04	0.05	0.045	0.2813774	0.0089948	0.0091364	0.0090656
13	0.0952381	115	Put	0.07	0.08	0.075	0.2446614	0.0110931	0.0112389	0.0111660
14	0.0952381	120	Put	0.10	0.12	0.110	0.2007928	0.0119600	0.0120925	0.0120263
15	0.0952381	125	Put	0.17	0.18	0.175	0.1556844	0.0123456	0.0124574	0.0124015
16	0.0952381	130	Put	0.30	0.31	0.305	0.1058078	0.0089006	0.0089712	0.0089359

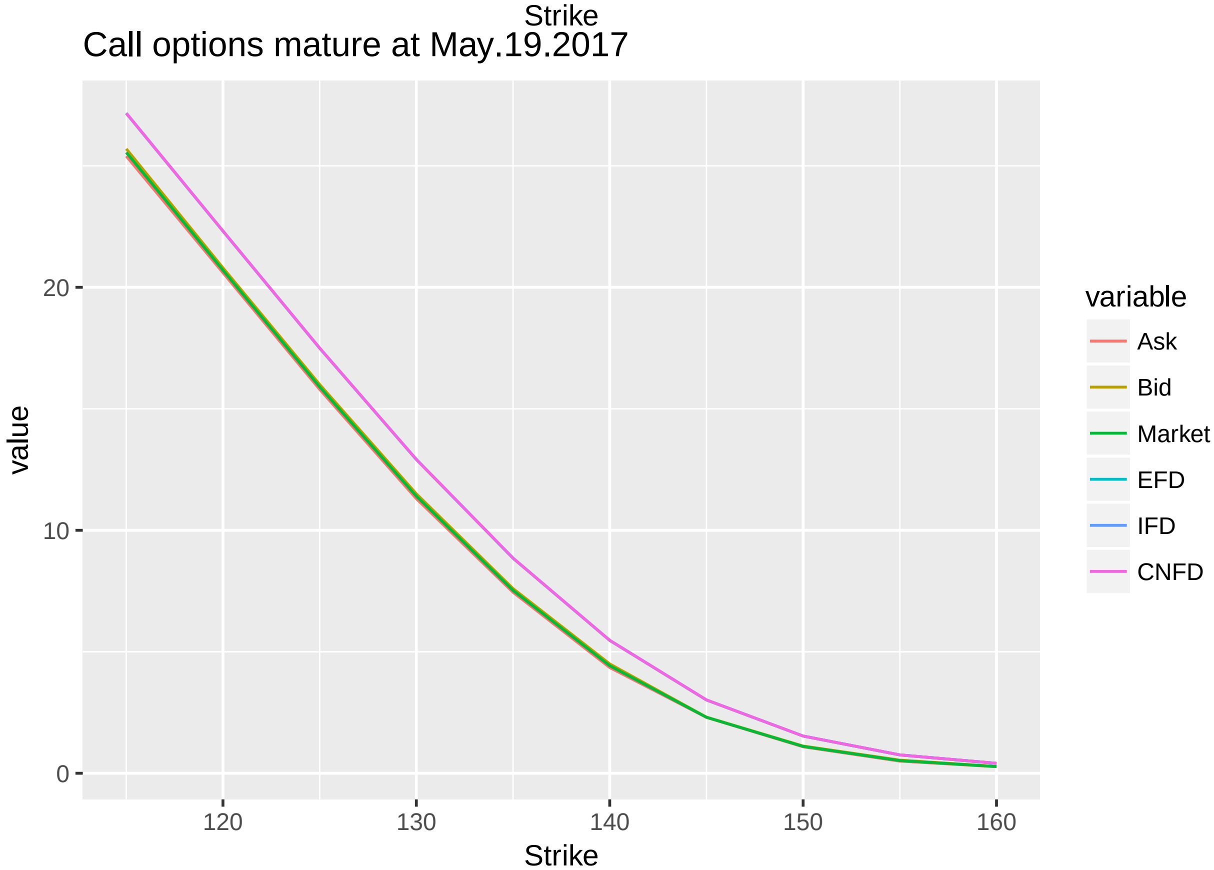
	T	Strike	Type	Ask	Bid	Market	implied vol	EFD	IFD	CNFD
17	0.0952381	135	Put	0.76	0.77	0.765	0.0431520	0.0004073	0.0004151	0.0004112
18	0.0952381	140	Put	2.24	2.23	2.235	NaN	NaN	NaN	NaN
19	0.0952381	150	Put	9.90	10.05	9.975	NaN	NaN	NaN	NaN
20	0.0952381	155	Put	14.70	15.10	14.900	NaN	NaN	NaN	NaN
21	0.0952381	160	Put	19.85	20.20	20.025	NaN	NaN	NaN	NaN
22	0.1746032	115	Call	25.40	25.70	25.550	0.3594826	27.1657112	27.1659754	27.1658432
23	0.1746032	120	Call	20.60	20.80	20.700	0.3123716	22.3169532	22.3170696	22.3170113
24	0.1746032	125	Call	15.80	16.00	15.900	0.2652605	17.4981196	17.4980286	17.4980740
25	0.1746032	130	Call	11.30	11.50	11.400	0.2302610	12.9159820	12.9155651	12.9157734
26	0.1746032	135	Call	7.45	7.60	7.525	0.2104248	8.8530152	8.8521741	8.8525946
27	0.1746032	140	Call	4.35	4.50	4.425	0.1962152	5.4698114	5.4688057	5.4693086
28	0.1746032	145	Call	2.30	2.31	2.305	0.1881090	3.0184531	3.0177819	3.0181175
29	0.1746032	150	Call	1.09	1.12	1.105	0.1863924	1.5315199	1.5314952	1.5315074
30	0.1746032	155	Call	0.50	0.54	0.520	0.1898256	0.7554071	0.7558981	0.7556525
31	0.1746032	160	Call	0.27	0.28	0.275	0.2000298	0.4085293	0.4092408	0.4088851
32	0.1746032	110	Put	0.13	0.17	0.150	0.2470455	0.0298012	0.0300260	0.0299136
33	0.1746032	115	Put	0.22	0.26	0.240	0.2194846	0.0451849	0.0454259	0.0453055
34	0.1746032	120	Put	0.40	0.42	0.410	0.1931635	0.0747181	0.0749638	0.0748410
35	0.1746032	125	Put	0.68	0.72	0.700	0.1642674	0.1193623	0.1195638	0.1194631
36	0.1746032	130	Put	1.27	1.32	1.295	0.1352760	0.2081243	0.2081964	0.2081603
37	0.1746032	135	Put	2.42	2.47	2.445	0.0986553	0.3289484	0.3288009	0.3288745
38	0.1746032	140	Put	4.40	4.55	4.475	NaN	NaN	NaN	NaN
39	0.1746032	150	Put	11.10	11.45	11.275	0.1795260	9.5827990	9.5828467	9.5828227
40	0.1746032	155	Put	15.55	15.95	15.750	0.1858202	13.8284533	13.8289794	13.8287163
41	0.1746032	160	Put	20.20	20.70	20.450	0.1853434	18.3635828	18.3642952	18.3639391
42	0.2460317	115	Call	25.35	25.75	25.550	0.3218128	27.7909872	27.7911483	27.7910676
43	0.2460317	120	Call	20.55	20.95	20.750	0.2821404	22.9865207	22.9864932	22.9865068
44	0.2460317	125	Call	16.00	16.35	16.175	0.2515277	18.3519631	18.3516458	18.3518043
45	0.2460317	130	Call	11.75	12.00	11.875	0.2251112	13.9235491	13.9228671	13.9232080
46	0.2460317	135	Call	8.10	8.30	8.200	0.2096619	10.0044844	10.0033983	10.0039414
47	0.2460317	140	Call	5.15	5.25	5.200	0.1975503	6.6697733	6.6685687	6.6691711
48	0.2460317	145	Call	3.05	3.10	3.075	0.1917330	4.1623013	4.1613939	4.1618476
49	0.2460317	150	Call	1.69	1.70	1.695	0.1887766	2.4257675	2.4254919	2.4256296
50	0.2460317	155	Call	0.89	0.93	0.910	0.1897303	1.3700761	1.3704341	1.3702549
51	0.2460317	160	Call	0.48	0.51	0.495	0.1938310	0.7756126	0.7763782	0.7759953
52	0.2460317	110	Put	0.25	0.28	0.265	0.2293073	0.0568181	0.0570997	0.0569590
53	0.2460317	115	Put	0.38	0.42	0.400	0.2047028	0.0825374	0.0828188	0.0826782
54	0.2460317	120	Put	0.63	0.67	0.650	0.1821963	0.1320101	0.1322619	0.1321360
55	0.2460317	125	Put	1.06	1.11	1.085	0.1593084	0.2168942	0.2170384	0.2169662
56	0.2460317	130	Put	1.84	1.90	1.870	0.1358482	0.3721276	0.3720604	0.3720939
57	0.2460317	135	Put	3.15	3.25	3.200	0.1063800	0.5957460	0.5954114	0.5955786
58	0.2460317	140	Put	5.15	5.30	5.225	NaN	NaN	NaN	NaN
59	0.2460317	150	Put	11.70	11.90	11.800	0.1697033	9.5126319	9.5125780	9.5126048
60	0.2460317	155	Put	15.80	16.25	16.025	0.1633137	13.3633848	13.3640115	13.3636980
61	0.2460317	160	Put	20.50	21.00	20.750	0.1698940	17.8672439	17.8681349	17.8676894

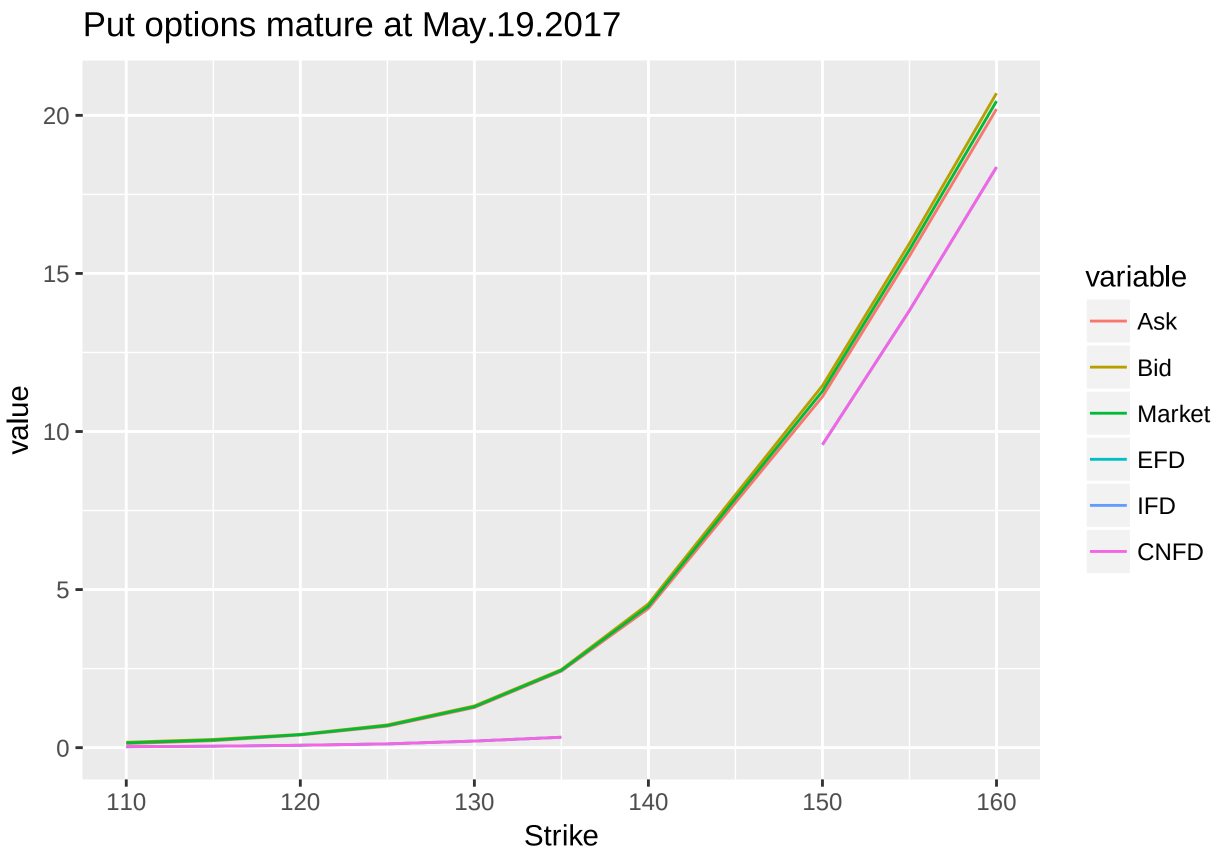
```
Captions <- c("Call options mature at Apr.21.2017", "Put options mature at Apr.21.2017",
              "Call options mature at May.19.2017", "Put options mature at May.19.2017",
              "Call options mature at Jun.16.2017", "Put options mature at Jun.16.2017")
for (i in 1:6) {
    df <- a.big.df[(10*i-9):(10*i), c(2, 4:6, 8:10)]
```

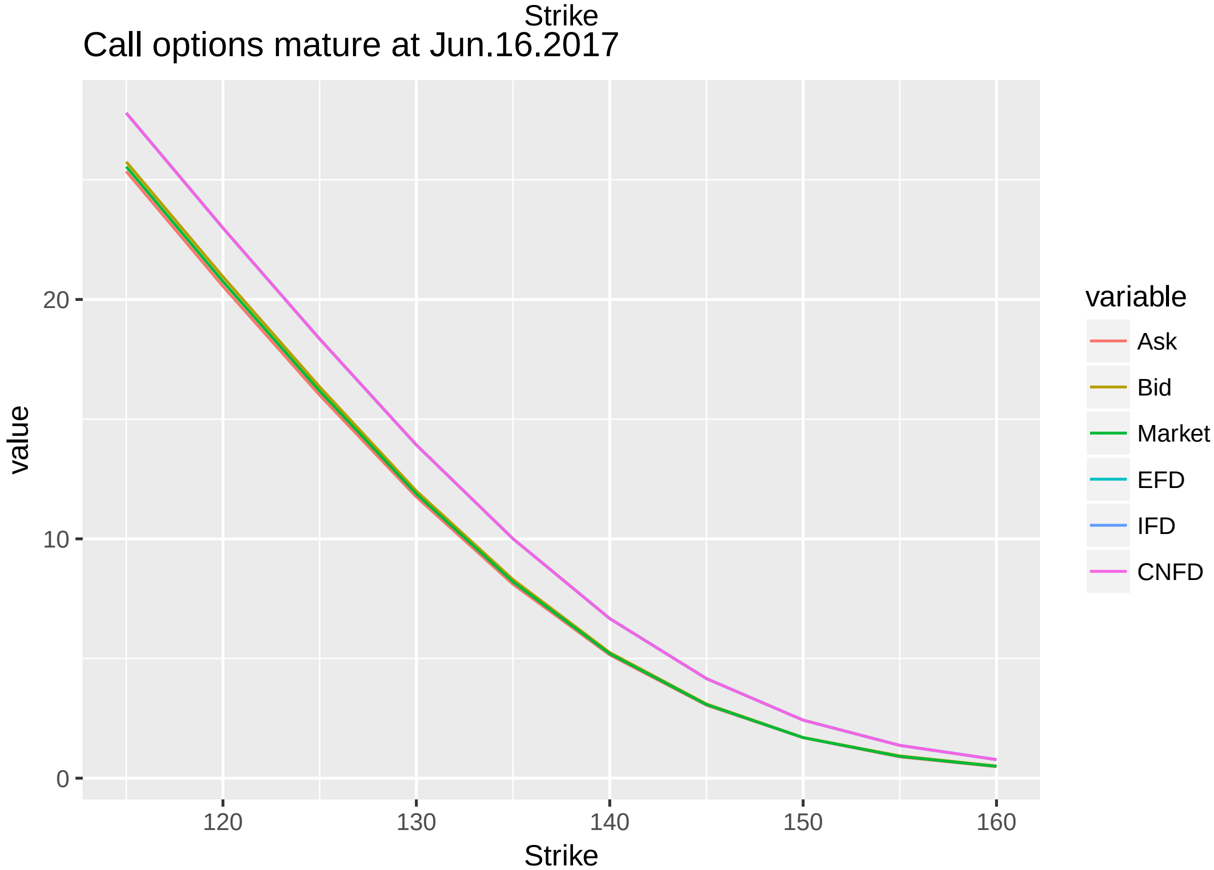
```
    df.long <- melt(df, id="Strike")
    print(ggplot(data=df.long, aes(x = Strike, y=value, colour=variable)) +
            geom_line() + ggtitle(Captions[i]))
    Sys.sleep(1)
}
```

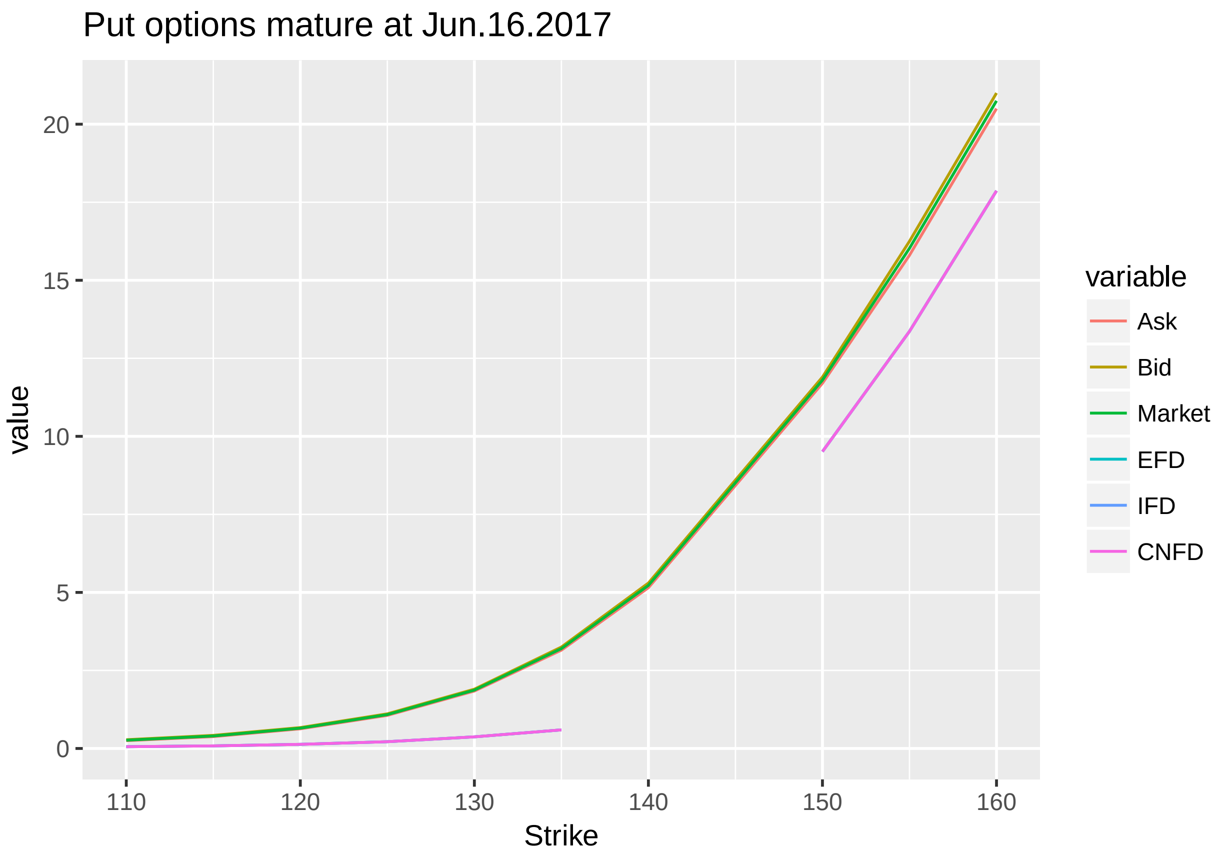



Put options mature at Apr.21.2017
value
0
5
10
15
20
110
120
130
140
150
160
Strike
variable
Ask
Bid
Market
EFD
IFD
CNFD


Call options mature at May.19.2017
value
0
10
20
120
130
140
150
160
Strike
variable
Ask
Bid
Market
EFD
IFD
CNFD


Put options mature at May.19.2017
value
Strike
variable
Ask
Bid
Market
EFD
IFD
CNFD


Call options mature at Jun.16.2017
value
Strike
variable
Ask
Bid
Market
EFD
IFD
CNFD



All six kinds of prices coincide excepe CNFD. The reason might be that it needs more steps to convergence than explicit and implicit methods.

Question 3

Define a function to calculate option price by heston model:

```
Option_Heston <- function(S0=1, K=1, V0=0.1, theta=0.1, sigma=0.1, rou=-0.3, lambda=0,
                          kappa=2, Tm=5, r=0, Nt=10000, Ns=40, Nj=40) {
    # Only for pricing European Put option in Heston model

    # Define maximum and minimun values for S, Nu, t
    Smin <- 0
    Smax <- 2*S0
    NUmin <- 0
    NUmax <- 0.5
    Tmin <- 0
    Tmax <- Tm

    # Building a uniform grid
    dS <- (Smax - Smin)/Ns
    St <- seq(Smin, Smax, by = dS)
    dnu <- (NUmax - NUmin)/Nj
    nu <- seq(NUmin, NUmax, by = dnu)
    dt <- (Tmax - Tmin)/Nt

    # initialise asset prices at maturity
```

```r
ST <- matrix(ncol = Nj + 1, nrow = Ns + 1)
for (i in 1:(Ns+1)) {ST[i, ] <- St[i]}

# initialise option values at maturity
UT <- ST
for (i in 1:(Ns+1)) {UT[i, ] <- pmax(K - UT[i, ], 0)}

# At time t-dt
UT_1 <- UT
# step back
for (k in 1:Nt) {
    # by time
    for (i in 2:Ns) {
        # by S
        for (j in 2:Nj) {
            # by nu

            # Based on the partial differential equation, calculate dU/dt
            # Then by dU/dt = ((U(t) - U(t-dt)))/dt, calcuate U(t-dt)
            dU.dS <- (UT[i+1, j] - UT[i-1, j])/2/dS
            dU.dnu <- (UT[i, j+1] - UT[i, j-1])/2/dnu
            dU.2dS <- (UT[i+1, j] - 2*UT[i, j] + UT[i-1, j])/dS^2
            dU.2dnu <- (UT[i, j+1] - 2*UT[i, j] + UT[i, j-1])/dnu^2
            dU.dS.dnu <- (UT[i+1, j+1] + UT[i-1, j-1] - UT[i+1, j-1] - UT[i-1, j+1])/4/dS/dnu

            dU.dt <- -(1/2*nu[j]*St[i]^2*dU.2dS + 1/2*sigma^2*nu[j]*dU.2dnu +
                           rou*sigma*nu[j]*St[i]*dU.dS.dnu + r*St[i]*dU.dS +
                           kappa*(theta - nu[j])*dU.dnu - r*UT[i, j])
            UT_1[i, j] <- UT[i, j] - dt*dU.dt
        }
    }

    # boundary conditon
    UT_1[1, ] <- UT_1[2, ] + dS # S=Smin
    UT_1[Ns+1, ] <- 0 # S=Smax

    # When nu = NUmax,  U(Si, NUmax, tn) = Si.
    # This column is identical in every step

    # When nu = NUmin
    for (l in 2:Ns) {
        dU.dS <- (UT[l+1, 1] - UT[l-1, 1])/2/dS
        dU.dnu <- (UT[l, 2] - UT[l, 1])/dnu
        dU.dt <- r*UT[l, 1] - r*St[l]*dU.dS - kappa*theta*dU.dnu
        UT_1[l, 1] <- UT[l, 1] - dt*dU.dt
    }

    UT <- UT_1
}

# Because the intervals in grid are small enough.
# we simply use the mean of surrounding grid points as our option price
ii <- S0/dS + 1
```

```r
    ii1 <- floor(ii)
    ii2 <- ii + 1
    jj <- V0/dnu + 1
    jj1 <- floor(jj)
    jj2 <- jj + 1
    ans <- (UT[ii1, jj1] + UT[ii1, jj2] + UT[ii2, jj1] + UT[ii2, jj2])/4
    return(ans)
}
```

```r
K <- c(0.5, 0.75, 1, 1.25, 1.5)
kappa <- c(1, 2, 4)

temp <- as.data.frame(matrix(ncol = 1, nrow = 5))
comparetable <- data.frame(temp)
rownames(comparetable) <- as.character(K)

for (i in 1:5) {
    kap = 2
    k <- K[i]
    PP <- Option_Heston(K = k, kappa = kap)
    comparetable[i, 1] <- PP
}
Real_value <- c(0.543017, 0.385109, 0.273303, 0.195434, 0.14121)
comparetable <- data.frame(comparetable, Real_value)
colnames(comparetable)[1] <- c("kappa=2")
```

The result is showed below:

```r
knitr::kable(comparetable, caption = "Comparison of solutions")
```

Table 12: Comparison of solutions

	kappa=2	Real_value
0.5	0.0400946	0.543017
0.75	0.1287683	0.385109
1	0.2602811	0.273303
1.25	0.4205618	0.195434
1.5	0.5980841	0.141210