Homework 4

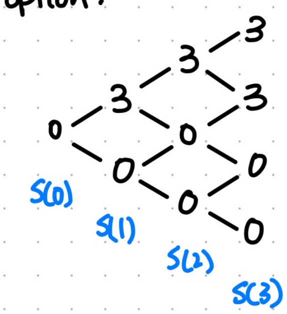
Zejie(Sandy) Gao

2024-02-24

Problem 1

Problem 1

Payoff of option:



$u = \frac{63.6}{60} = 1.06$ $\quad h = \frac{1}{12}$ year

$d = \frac{57.6}{60} = 0.96$ $\quad n = 3$

$q = \frac{e^{(r-\delta)h} - d}{u - d}$, $h = 1$, $\delta = 0$ $\quad T = \frac{3}{12}$ year

$= \frac{e^{0.04 \cdot \frac{1}{12}} - 0.96}{1.06 - 0.96} = 0.433389$

① At $t = 0$

$S(0) = 60$
$r = 4\%$
$C_u = 3$
$C_d = 0$

$C_0 = e^{-rnh} E^Q[\text{Payoff}]$

$= e^{-0.04 \times \frac{1}{4}} \times [q^3 \times 3 + 3 \times q^2 \times (1-q) \times 3 + 0 + 0]$

$= 1.19006328 \approx 1.19$

$C(1, 63.6) = e^{-0.04 \times \frac{1}{6}} \times [q^2 \times 3 + 2 \times 3 \times q(1-q) + 0]$

$= 2.02332188$

$C(1, 57.6) = e^{-0.04 \times \frac{1}{6}} \times [q^2 \times 3 + 0 + 0]$

$= 0.55973392$

$\Delta_0 = \frac{C_u - C_d}{uS_0 - dS_0} = \frac{2.02 - 0.56}{63.6 - 57.6} \approx 0.2439313$

$B_0 = e^{-rh}\left(\frac{uC_d - dC_u}{u - d}\right) = e^{-0.04 \times \frac{1}{12}} \left(\frac{1.06 \cdot 0.56 - 0.96 \cdot 2.02}{1.06 - 0.96}\right) \approx -13.44582$

→ Transactions:

	$\boxed{\text{Value}}$
· Sell Call for \$1.19	+1.19
· Buy 0.2439313 share @ \$60.	−14.64
· Take a loan of \$13.45.	+13.45
Cash Flow:	0

② At $t = h$

$S(1) = \$57.6$

$\Delta(1, 57.6) = 0.2249723$

$B(1, 57.6) \approx -12.40$

Transaction:	Value
• Buy 0.2249723 share @ 57.6 / share	-12.96
• Take loan of \$12.4.	$+12.4$
• 0.2439313 share are worth @ 57.6/share	$+14.05$
• Owe $13.45 \times e^{0.04/12}$	-13.49
Net Flow :	0

③ At $t = 2h$

$S(2) = \$61.06$

$\Delta(2, 61.06) = 0.49132$

$B(2, 61.06) = -28.70$

Transaction:	Value
• Buy 0.49132 share @ 61.06 share	-30.00
• Take loan of \$28.70	$+28.70$
• 0.2249723 share are worth @61.06/share	$+13.74$
• Owe $12.40 \times e^{0.04/12}$	-12.44
Net Flow :	0

③ At $t = 3h$

$S(3) = \$64.72$

Transaction: | | $\boxed{\text{Value}}$
--- | ---
• 0.49132 share are worth @ 64.72/share | +31.80
• Owe $28.70 \times e^{0.04/12}$ | −28.80
• Payoff of Call option at $S(3) = 64.72$ | −3
Net Flow : | 0

```r
binTree <- function(S=60, K=63, N=3, r=0.04, delta=0, u=1.06, d=0.96, h=1/12) {
    disc <- exp(-r * h)
    q <- (exp((r - delta) * h) - d) / (u - d)

    # Initialize arrays for option premia, delta, and B
    V <- array(0, dim=c(N+1, N+1))
    Del <- B <- array(0, dim=c(N+1, N+1))

    # Calculate the terminal payoffs for the binary option
    for (i in 0:N) {
        finalS <- S * u^i * d^(N-i)
        V[N+1, i+1] <- ifelse(finalS - K > 0, 3, 0) # Binary Call payoff
    }

    # Backward induction to calculate delta and B for the replicating portfolio
    for (j in N:1) {
        for (i in 0:(j-1)) {
            curS <- S * u^i * d^(j-1-i)
            # Calculate Delta and B at each node
            Del[j, i+1] <- (V[j+1, i+2] - V[j+1, i+1]) / (curS * (u - d))
            B[j, i+1] <- disc * (u * V[j+1, i+1] - d * V[j+1, i+2]) / (u - d)
            # Calculate the option value at each node
            V[j, i+1] <- Del[j, i+1] * curS + B[j, i+1]
        }
    }

    # Calculate the initial Delta and B for the replicating portfolio
    Delta0 <- (V[2, 2] - V[2, 1]) / (S * (u - d))
    B0 <- disc * (u * V[2, 1] - d * V[2, 2]) / (u - d)
    # The price of the option is the initial value of the replicating portfolio
    price <- Delta0 * S + B0

    return(list(price = price, premia = V, Delta = Del, Bank = B))
}
```

```r
# Calculate the price of the replicating portfolio
price_s0 <- binTree(S=60, K=63, r=0.04, u=1.06, d=0.96, N=3, h=1/12)
price_s0
```

```
## $price
## [1] 1.190063
##
## $premia
##           [,1]     [,2]     [,3] [,4]
## [1,] 1.1900633 0.000000 0.000000    0
## [2,] 0.5597339 2.023322 0.000000    0
## [3,] 0.0000000 1.295840 2.990017    0
## [4,] 0.0000000 0.000000 3.000000    3
##
## $Delta
##           [,1]      [,2] [,3] [,4]
## [1,] 0.2439313 0.0000000    0    0
## [2,] 0.2249723 0.2663799    0    0
```

```
## [3,] 0.0000000 0.4913522    0    0
## [4,] 0.0000000 0.0000000    0    0
##
## $Bank
##           [,1]      [,2]     [,3] [,4]
## [1,] -13.44582   0.00000 0.000000    0
## [2,] -12.39867 -14.91844 0.000000    0
## [3,]   0.00000 -28.70416 2.990017    0
## [4,]   0.00000   0.00000 0.000000    0
```

```r
cash_flow_t0 <- price_s0$premia[1,1]-price_s0$Delta[1,1]*60-price_s0$Bank[1,1]
cash_flow_t0
```

```
## [1] 0
```

```r
# Calculate the price of the replicating portfolio
price_s1 <- binTree(S=57.6, K=63, r=0.04, u=1.06, d=0.96, N=2, h=1/12)
price_s1
```

```
## $price
## [1] 0.5597339
##
## $premia
##           [,1]    [,2] [,3]
## [1,] 0.5597339 0.00000    0
## [2,] 0.0000000 1.29584    0
## [3,] 0.0000000 0.00000    3
##
## $Delta
##           [,1]      [,2] [,3]
## [1,] 0.2249723 0.0000000    0
## [2,] 0.0000000 0.4913522    0
## [3,] 0.0000000 0.0000000    0
##
## $Bank
##           [,1]      [,2] [,3]
## [1,] -12.39867   0.00000    0
## [2,]   0.00000 -28.70416    0
## [3,]   0.00000   0.00000    0
```

```r
cash_flow_t1 <- price_s0$Delta[1,1]*57.6+price_s0$Bank[1,1]*exp(0.04/12) - price_s1$Delta[1,1]*57.6 - pr
cash_flow_t1
```

```
## [1] 1.776357e-15
```

```r
# Calculate the price of the replicating portfolio
price_s2 <- binTree(S=61.06, K=63, r=0.04, u=1.06, d=0.96, N=1, h=1/12)
price_s2
```

```
## $price
## [1] 1.29584
```

```
##
## $premia
##         [,1] [,2]
## [1,] 1.29584    0
## [2,] 0.00000    3
##
## $Delta
##         [,1] [,2]
## [1,] 0.49132    0
## [2,] 0.00000    0
##
## $Bank
##           [,1] [,2]
## [1,] -28.70416    0
## [2,]   0.00000    0
```

```
cash_flow_t2 <- price_s1$Delta[1,1]*61.06+price_s1$Bank[1,1]*exp(0.04/12) - price_s2$Delta[1,1]*61.06 -
cash_flow_t2
```

```
## [1] 0.000899889
```

```
cash_flow_t3 <- price_s2$Delta[1,1]*64.72+price_s2$Bank[1,1]*exp(0.04/12) - 3
cash_flow_t3
```

```
## [1] -0.001768752
```

Problem 2

Consider a binomial model with $u = 1.04, d = 100/104, \delta = 0$ and interest rate r =3% a year, compounded continuously. Using T = 1 maturity of one year, initial stock price S(0) = 40 and N = 4 periods:

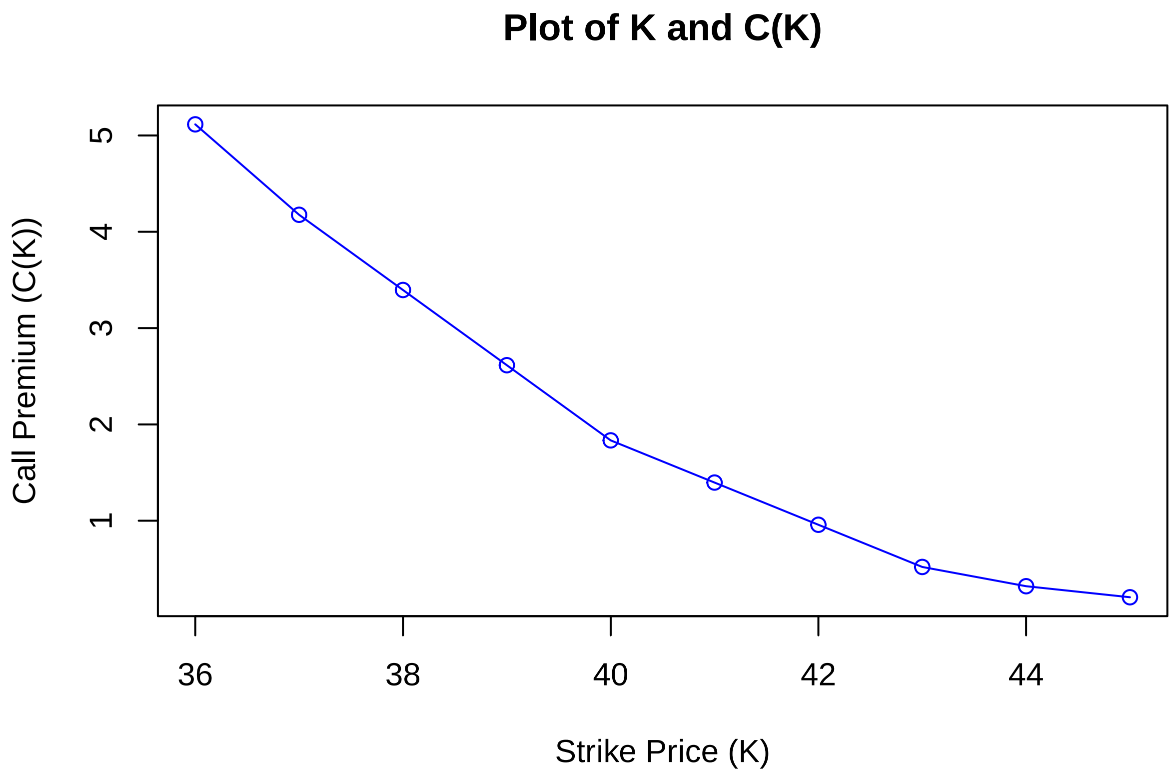
```
binTree_call <- function(S=49, K=50,N=4,r=0.03,delta=0,u=1.04, d=0.100/104, h=1) {

disc <- exp(-r*h)
q <- (exp( (r-delta)*h)-d)/(u-d)

V <- array(0, dim=c(N+1,N+1)) # matrix for storing all the option premia at each node
Del <- B <- array(0, dim=c(N+1,N+1))

for (i in 0:(N)) {  # terminal payoff
  finalS <- S*(u)^i*(d)^(N-i)
  V[N+1,i+1] <- max( finalS-K, 0)     # Call payoff
}
V_rn=V  # the direct risk-neutral computation -- gives same answers as V

for (j in (N):1) {  # column in the tree
    for (i in 0:(j-1)) {  # i counts number of up-moves in the j-th column
        curS = S * u^(i) * d^(j-1-i)
        B[j,i+1] = disc*( u*V[j+1,i+1] - d*V[j+1,i+2] )/(u - d)
        Del[j,i+1] = exp(-delta*h)*( V[j+1,i+2] - V[j+1,i+1] )/(curS * u - curS * d)
        V[j,i+1] = Del[j,i+1]*curS + B[j,i+1]
```

```
        V_rn[j,i+1] <- disc*( q*V_rn[j+1,i+2] + (1-q)*V_rn[j+1,i+1] )
    }
}

Delta0 <- exp(-delta*h)*(V[2,2]-V[2,1])/(S*(u-d))  # V[2,2] = C_u, V[2,1]=C_d
return (list(premia = V,Delta=Del,Bank=B) )
}
```

a) Find the premium of the European Call C(K) for K = 36, 37, 38, 39, 40, 41, 42, 43, 44, 45. You're encouraged to use a computer to do this faster. Create a plot of K$\rightarrow$ C(K)

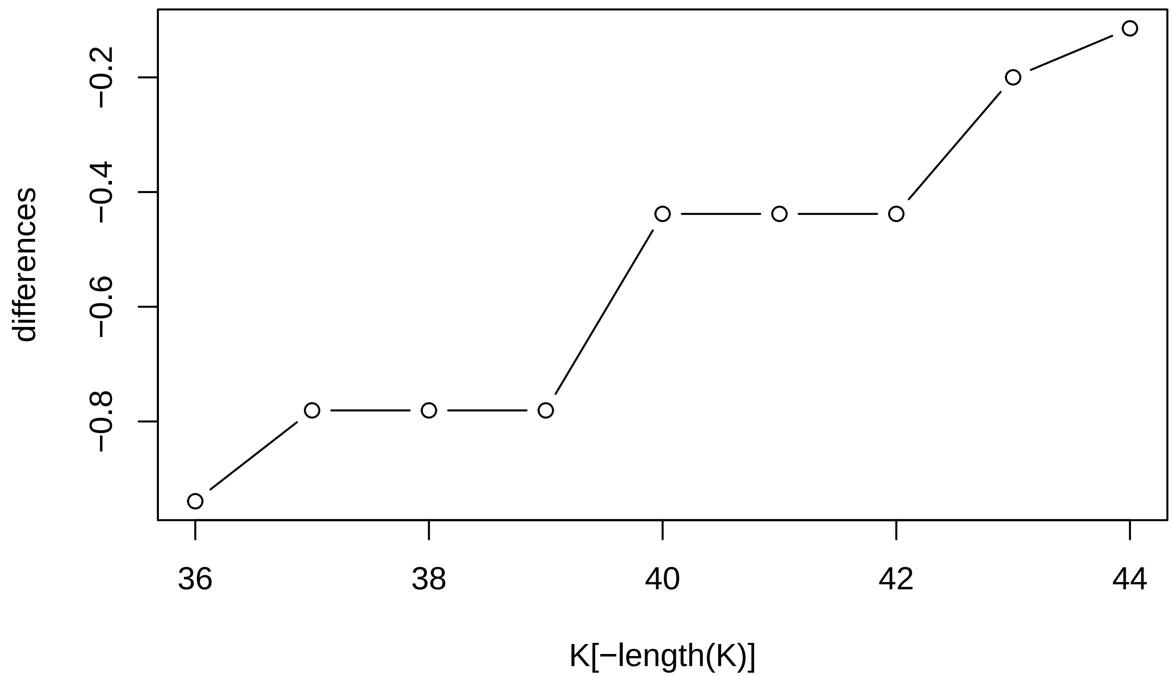
```
# Determing the cal premium under each K value
price_c <- function(a){
  price_set <- numeric(length(a))
  for (i in seq_along(a)){
     price_k <- binTree_call(S=40, K=a[i], r=0.03, u=1.04, d=100/104, N=4, h=1/4)
     price_set[i] <- price_k$premia[1,1]
  }
  names(price_set) <- paste("C", a, sep="")
  return(price_set)
}

K <- c(36, 37, 38, 39, 40, 41, 42, 43, 44, 45)

Call_Premium <- price_c(a = K)
Call_Premium
```

```
##       C36       C37       C38       C39       C40       C41       C42       C43
## 5.1154277 4.1763139 3.3956185 2.6149231 1.8342278 1.3961658 0.9581039 0.5200420
##       C44       C45
## 0.3200863 0.2055383
```



b) Write down a mathematical formula for the function K $\rightarrow$ C(K). Hint: this is a piecewise function. Try computing e.g. the Call premium when K $= 40 + \varepsilon$ when $\varepsilon = 0.1, 0.01, \ldots$ to see the pattern.

```r
# Calculate the differences between consecutive call premiums
differences <- diff(Call_Premium) / diff(K)

# Then plot the differences to visually inspect where the changes occur
plot(K[-length(K)], differences, type='b')
```




```
# based on the graph, the slope of C(K) changes at price 37, 39, 40, 42, 43
K <- c(36, 36.5)
y1 <- function(x){
  price <- 5.115428- ((x-36)*0.941978)
  }

Call_Premium <- price_c(a = K)
Call_Premium
```

```
##      C36    C36.5
## 5.115428 4.644439
```

```
K <- c(37, 37.5, 39)
y2 <- function(x){
  price <- 4.176314 - ((x-37)*0.780696)
  }

Call_Premium <- price_c(a = K)
Call_Premium
```

```
##      C37    C37.5      C39
## 4.176314 3.785966 2.614923
```

```r
K <- c(39, 39.5, 39.6)
y3 <- function(x){
  price <- 2.614923 - ((x-39)*0.780696)
  }

Call_Premium <- price_c(a = K)
Call_Premium
```

```
##      C39    C39.5    C39.6
## 2.614923 2.224575 2.146506
```

```r
K <- c(40, 40.1, 40.01)
y4 <- function(x){
  price <- 1.8342278 - ((x-40)*0.43806)
  }

Call_Premium <- price_c(a = K)
Call_Premium
```

```
##      C40    C40.1   C40.01
## 1.834228 1.790422 1.829847
```

```r
K <- c(42, 42.1, 42.01)
y5 <- function(x){
  price <- 0.9581039 - ((x-42)*0.438062)
  }

Call_Premium <- price_c(a = K)
Call_Premium
```

```
##       C42     C42.1    C42.01
## 0.9581039 0.9142977 0.9537233
```

```r
K <- c(43, 43.5, 43.02,43.9)
y6 <- function(x){
  price <- 0.5200420 - ((x-43)*0.2094454)
  }

Call_Premium <- price_c(a = K)
Call_Premium
```

```
##       C43     C43.5    C43.02     C43.9
## 0.5200420 0.3773603 0.5112807 0.3315411
```

For $36 \leq K < 37$: C(K)=5.115428−(K−36)×0.941978

For $37 \leq K < 39 : C(K) = 4.176314 - (K - 37) \times 0.780696$

For $39 \leq K < 40 : C(K) = 2.614923 - (K - 39) \times 0.780696$

For $40 \leq K < 42 : C(K) = 1.8342278 - (K - 40) \times 0.43806$

For $42 \leq K < 43 : C(K) = 0.9581039 - (K - 42) \times 0.438062$

For $43 \leq K$:

$C(K) = 0.5200420 - (K - 43) \times 0.2094454$

c) Compute the prices of the European Put with K = 38, 40, 42 and verify that Put-Call parity holds.

```
binTree_put <- function(S=49, K=50,N=10,r=0.05,delta=0,u=1.1, d=0.85, h=1) {

disc <- exp(-r*h)
q <- (exp( (r-delta)*h)-d)/(u-d)

V <- array(0, dim=c(N+1,N+1)) # matrix for storing all the option premia at each node
Del <- B <- array(0, dim=c(N+1,N+1))

for (i in 0:(N)) {  # terminal payoff
  finalS <- S*(u)^i*(d)^(N-i)
  V[N+1,i+1] <- max(K - finalS, 0)     # Put payoff
}
V_rn=V  # the direct risk-neutral computation -- gives same answers as V

for (j in (N):1) {  # column in the tree
    for (i in 0:(j-1)) {  # i counts number of up-moves in the j-th column
        curS = S * u^(i) * d^(j-1-i)
        B[j,i+1] = disc*( u*V[j+1,i+1] - d*V[j+1,i+2] )/(u - d)
        Del[j,i+1] = exp(-delta*h)*( V[j+1,i+2] - V[j+1,i+1] )/(curS * u - curS * d)
        V[j,i+1] = Del[j,i+1]*curS + B[j,i+1]
        V_rn[j,i+1] <- disc*( q*V_rn[j+1,i+2] + (1-q)*V_rn[j+1,i+1] )
    }
}

Delta0 <- exp(-delta*h)*(V[2,2]-V[2,1])/(S*(u-d))  # V[2,2] = C_u, V[2,1]=C_d
return (list(premia = V,Delta=Del,Bank=B) )
}
```

```
# Determing the put premium under each K value
price_p <- function(a){
  price_set <- numeric(length(a))
  for (i in seq_along(a)){
     price_k <- binTree_put(S=40, K=a[i], r=0.03, u=1.04, d=100/104, N=4, h=1/4)
     price_set[i] <- price_k$premia[1,1]
  }
  names(price_set) <- paste("P", a, sep="")
  return(price_set)
}

K <- c(38, 40, 42)

Put_Premium <- price_p(a = K)
Put_Premium
```

```
##       P38       P40       P42
## 0.2725488 0.6520491 1.7168163
```

- To verifying Call-put parity, take K = 38 as an example.

- Recall the formula of Call-Put Parity:

$$C - P = S_0 - \mathrm{PV}(K)$$

  , where dividend rate = 0.

```
# Given parameters
S0 <- 40  # Current price of the underlying asset
r <- 0.03  # Risk-free interest rate
T <- 1  # Time to expiration in years

# Calculate the present value of the strike prices
PV_K <- function(K) {
  K * exp(-r * T)
}

# Compute the call and put prices for each K (assuming price_c and price_p are defined)
K_values <- c(38, 40, 42)
call_prices <- price_c(K_values)  # Ensure this function is defined and returns call prices
put_prices <- price_p(K_values)   # Ensure this function is defined and returns put prices

# Calculate LHS and RHS of Put-Call Parity for each K
LHS <- call_prices - put_prices
RHS <- S0 - sapply(K_values, PV_K)

# Check if the Put-Call Parity holds for each K
parity_checks <- abs(LHS - RHS) < 1e-4  # Using a small tolerance for numerical comparisons

# Combine the results into a data frame for a clear presentation
results <- data.frame(Strike = K_values, Call = call_prices, Put = put_prices, LHS = LHS, RHS = RHS, Pa

# Print the results
print(results)
```

```
##     Strike      Call       Put        LHS        RHS Parity_Check
## C38     38 3.3956185 0.2725488  3.1230697  3.1230697         TRUE
## C40     40 1.8342278 0.6520491  1.1821787  1.1821787         TRUE
## C42     42 0.9581039 1.7168163 -0.7587124 -0.7587124         TRUE
```

Problem 3

Consider a binomial model with $\sigma = 0.24, \delta = 0.06$ and interest rate r=5% annual, both compounded continuously. Using T = 1 maturity of one year, initial stock price $S_0 = 100$ and N = 4 periods, consider the American Call C^{Am} with strike K = 95.

1. In which scenarios is early exercise rational?

Case 1: UP/ UP/ Ex

Case 2: Up/UP/UP/ Ex

Case 3: Up/UP/Down/ Ex

Case 3: Up/Down/ Up/ Ex

2. Find the premium of this Call today C_0^{Am}.

```
binTreeAmerican <- function(S=100, K=95, N=4, r=0.05, delta=0.06, h=1/4, optionType="call") {
    sigma <- 0.24

    u <- exp((r-delta)*h + sigma * sqrt(h))
    d <- exp((r-delta)*h - sigma * sqrt(h))
    disc <- exp(-r*h)
    q <- (exp((r-delta)*h)-d)/(u-d)

    # Initialize arrays for option values, delta, and B
    V <- array(0, dim=c(N+1, N+1))
    Del <- array(0, dim=c(N+1, N+1))
    B <- array(0, dim=c(N+1, N+1))

    # Terminal payoffs
    for (i in 0:N) {
        finalS <- S * u^i * d^(N-i)
        if (optionType == "call") {
            V[N+1, i+1] <- max(finalS - K, 0)
        } else { # optionType == "put"
            V[N+1, i+1] <- max(K - finalS, 0)
        }
    }

    # Backward induction for American option
    for (j in N:1) {
        for (i in 0:(j-1)) {
            curS <- S * u^i * d^(j-1-i)
            holdValue <- (q * V[j+1, i+2] + (1 - q) * V[j+1, i+1]) * disc
            if (optionType == "call") {
                exerciseValue <- max(curS - K, 0)
            } else { # optionType == "put"
                exerciseValue <- max(K - curS, 0)
            }
            V[j, i+1] <- max(holdValue, exerciseValue)
            Del[j, i+1] <- (V[j+1, i+2] - V[j+1, i+1]) / (curS * (u - d))
            B[j, i+1] <- (holdValue - Del[j, i+1] * curS) / disc
        }
    }

    # Calculate the initial Delta and B for the replicating portfolio
    Delta0 <- (V[2, 2] - V[2, 1]) / (S * (u - d))
    B0 <- disc * (u * V[2, 1] - d * V[2, 2]) / (u - d)
    # The price of the option is the initial value of the replicating portfolio
    price <- Delta0 * S + B0

    return(list(price = price, Delta = Del, B = B, premia = V))
}

# Example usage for an American call option
result <- binTreeAmerican(S=100, K=95, N=4, r=0.05, delta=0.06, h=1/4, optionType="call")
result
```

```
## $price
## [1] 12.21958
##
## $Delta
##             [,1]      [,2] [,3] [,4] [,5]
## [1,] 0.61207764 0.0000000    0    0    0
## [2,] 0.37484764 0.8399365    0    0    0
## [3,] 0.09897783 0.6302118    1    0    0
## [4,] 0.00000000 0.1895859    1    1    0
## [5,] 0.00000000 0.0000000    0    0    0
##
## $B
##             [,1]      [,2]      [,3]      [,4] [,5]
## [1,] -50.527103   0.00000   0.00000   0.00000    0
## [2,] -28.976448 -76.17753   0.00000   0.00000    0
## [3,]  -6.970641 -54.56335 -96.90689   0.00000    0
## [4,]   0.000000 -15.01656 -96.68704 -97.14465    0
## [5,]   0.000000   0.00000   0.00000   0.00000    0
##
## $premia
##             [,1]      [,2]      [,3]     [,4]     [,5]
## [1,] 11.3083192  0.000000  0.000000  0.00000  0.00000
## [2,]  4.5464948 19.234872  0.000000  0.00000  0.00000
## [3,]  0.8629895  8.821303 31.490877  0.00000  0.00000
## [4,]  0.0000000  1.859102 16.907226 47.26197  0.00000
## [5,]  0.0000000  0.000000  4.004983 30.86000 64.99942
```

3. Suppose the stock moves are Up/Up/Down/Down. Compute the replicating portfolio and the exercise strategy along that scenario.

At time 0:

```r
time0 <- binTreeAmerican(S=100, K=95, N=4, r=0.05, delta=0.06, h=1/4, optionType="call")
time0$premia[1,1] # premium at time 0
```

```
## [1] 11.30832
```

```r
time0$Delta[1,1] # percentage of share
```

```
## [1] 0.6120776
```

```r
time0$B[1,1] # investment or borrowing from bank
```

```
## [1] -50.5271
```

- Short call at $11.31
- Buying 0.61207764 100-valued shares in that market at $61.21
- Borrowing $50.53 in the bank

```r
net_cash_flow <- 11.31-61.21+50.53
net_cash_flow
```

```
## [1] 0.63
```

At time h up:

```r
time0 <- binTreeAmerican(S=100, K=95, N=4, r=0.05, delta=0.06, h=1/4, optionType="call")
time0$Delta[2,2] # percentage of share
```

```
## [1] 0.8399365
```

```r
time0$B[2,2] # investment or borrowing from bank
```

```
## [1] -76.17753
```

- 0.61207764 112.47-valued shares in that market are worth $70.72
- Owe $50.53 * e^(rh) in the bank, $51.17
- Buying 0.839937 112.47-valued shares in that market at $94.47
- Borrowing $76.18 in the bank

  ```r
  net_cash_flow <- 70.72-51.17-94.47 +76.18
  net_cash_flow
  ```

  ```
  ## [1] 1.26
  ```

At time 2h up:

```r
time0 <- binTreeAmerican(S=100, K=95, N=4, r=0.05, delta=0.06, h=1/4, optionType="call")
time0$Delta[3,3] # percentage of share
```

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

```r
time0$B[3,3] # investment or borrowing from bank
```

```
## [1] -96.90689
```

- 0.839937 126.49-valued shares in that market are worth $106.24
- Owe $76.18* e^(rh) in the bank, $72.08
- Could do the early excercise here, payoff for short call would be -(126.49-95)

  ```r
  net_cash_flow <- 106.24-72.08 -(126.49-95)
  net_cash_flow
  ```

  ```
  ## [1] 2.67
  ```

UP/UP/Down/Down is one of the scenario that can have early excercise of our call option.

Problem 4

Using the posted R script as a starting point, implement the binomial tree option pricing algorithm for European options.

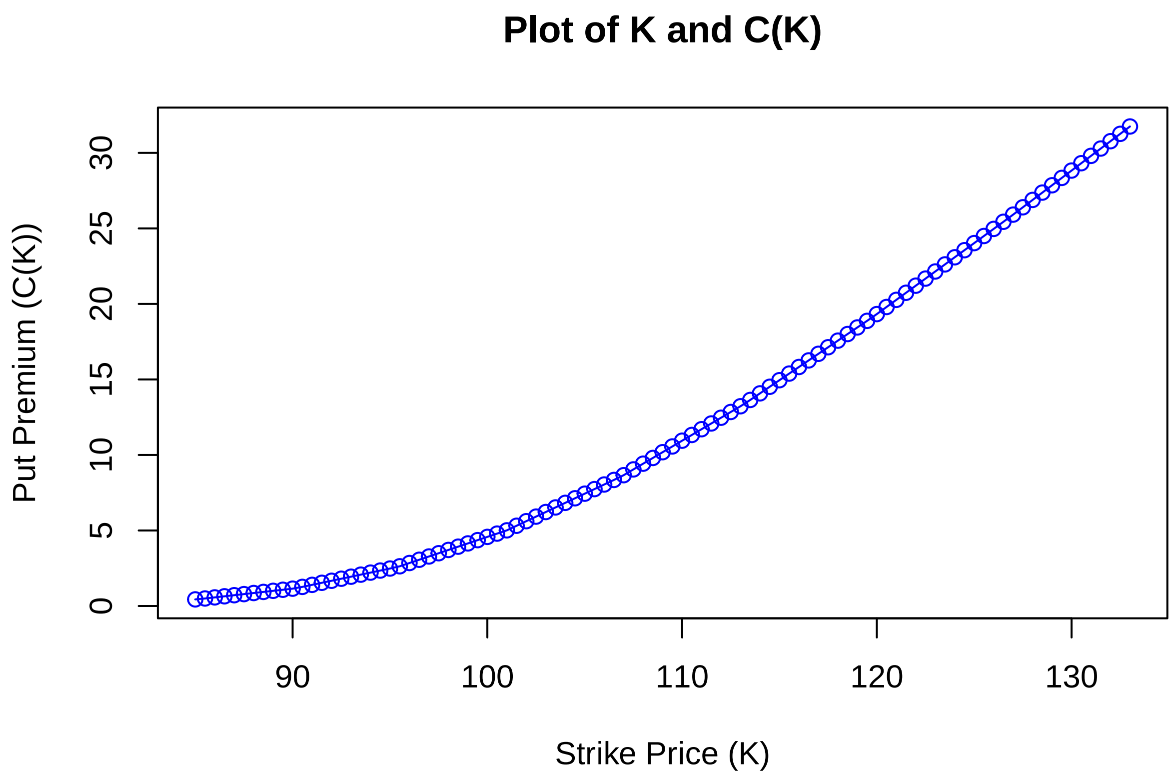
1. Consider a binomial model with = 0.18, and interest rate r of 2% a year, compounded continuously. Using T = 1/2 maturity of half a year, initial stock price S(0) = 100 and N = 20 periods, plot the premium of the European Put $P^E(K)$ as a function of strike K, with K = 85, 85.5, 88, . . . , 133

```r
binTree_put <- function(S=100, K=95,N=20,r=0.02,delta=0,h=1/40, sigma = 0.18) {

u <- exp((r-delta)*h + sigma * sqrt(h))
d <- exp((r-delta)*h - sigma * sqrt(h))
disc <- exp(-r*h)
q <- (exp( (r-delta)*h)-d)/(u-d)

V <- array(0, dim=c(N+1,N+1)) # matrix for storing all the option premia at each node
Del <- B <- array(0, dim=c(N+1,N+1))

for (i in 0:(N)) {  # terminal payoff
  finalS <- S*(u)^i*(d)^(N-i)
  V[N+1,i+1] <- max( K - finalS, 0)     # Put payoff
}
V_rn=V  # the direct risk-neutral computation -- gives same answers as V

for (j in (N):1) {  # column in the tree
    for (i in 0:(j-1)) {  # i counts number of up-moves in the j-th column
        curS = S * u^(i) * d^(j-1-i)
        B[j,i+1] = disc*( u*V[j+1,i+1] - d*V[j+1,i+2] )/(u - d)
        Del[j,i+1] = exp(-delta*h)*( V[j+1,i+2] - V[j+1,i+1] )/(curS * u - curS * d)
        V[j,i+1] = Del[j,i+1]*curS + B[j,i+1]
        V_rn[j,i+1] <- disc*( q*V_rn[j+1,i+2] + (1-q)*V_rn[j+1,i+1] )
    }
}

Delta0 <- exp(-delta*h)*(V[2,2]-V[2,1])/(S*(u-d))  # V[2,2] = C_u, V[2,1]=C_d
return (list(premia = V,Delta=Del,Bank=B) )
}
```

```r
# Determing the put premium under each K value
price_p <- function(a){
  price_set <- numeric(length(a))
  for (i in seq_along(a)){
     price_k <- binTree_put(K=a[i])
     price_set[i] <- price_k$premia[1,1]
  }
  names(price_set) <- paste("P", a, sep="")
  return(price_set)
}
```

```r
K <- seq(85, 133, by=0.5)

Put_Premium <- price_p(a = K)
```

```r
# plot
plot(K,Put_Premium,
     type = "o",
     col = "blue",
     xlab = "Strike Price (K)",
     ylab = "Put Premium (C(K))",
     main = "Plot of K and C(K)")
```



2. What is the smallest slope of K 7 $\to P^E(K)$? What is the largest slope of K 7 $\to P^E(K)$?

```r
# Calculate the differences in the Put_Premium
put_premium_differences <- diff(Put_Premium)

# Calculate the differences in K
k_differences <- diff(K)

# Calculate the slopes
slopes <- put_premium_differences / k_differences
```

```r
# Find the smallest slope
smallest_slope <- min(slopes)

# Find the largest slope
largest_slope <- max(slopes)

# Print the results
smallest_slope
```

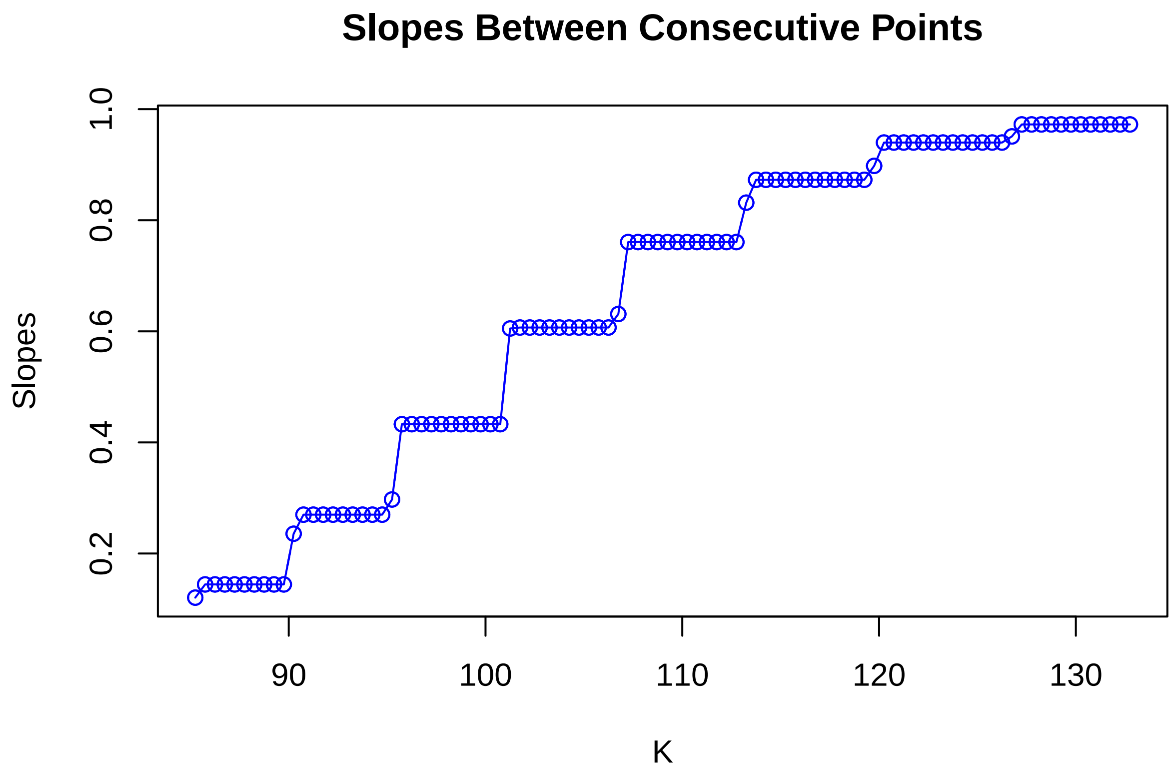
```
## [1] 0.1205331
```

```r
largest_slope
```

```
## [1] 0.9724919
```

3. Is the function K $\rightarrow P^E(K)$ concave or convex? How do the above questions relate to the material in Chapter 9?

```r
# Create a vector of midpoints for K to plot the slope against
K_midpoints <- K[-length(K)] + k_differences / 2

# Plot the slopes
plot(K_midpoints, slopes, type = 'b', col = 'blue', xlab = 'K', ylab = 'Slopes', main = 'Slopes Between

# Add a reference line at slope = 0 for clarity
abline(h = 0, col = 'red', lty = 2)

# Additional lines to connect the points can be added for clarity
lines(K_midpoints, slopes, col = 'blue', type = 'o')
```



Based on the graph above, the slope is generally increasing with the K value and it is positive. Therefore, the function K-> $P^E(K)$ is convex.

Problem 5

Use the same setting as Problem 4. Modify the binomial tree pricing algorithm to compute prices of an American Put $P^A(K)$ with maturity T and strike K. In chapter 9 , we have been told a principle that

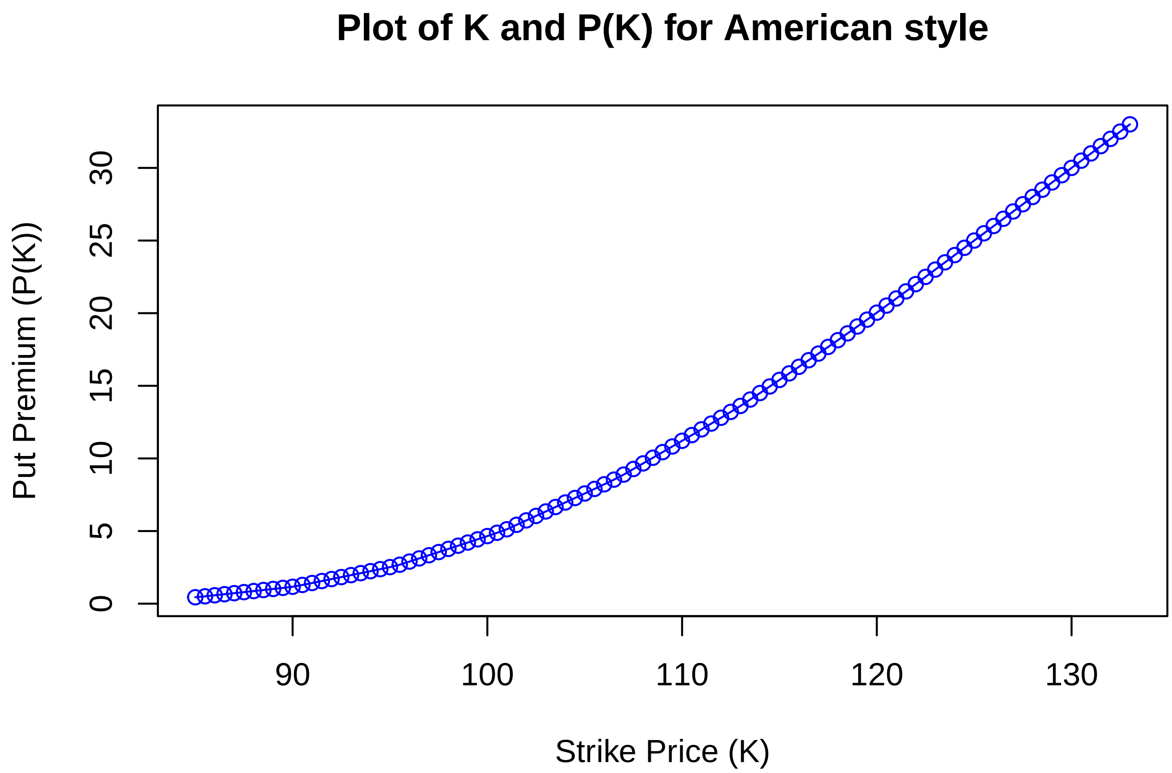
1. Compute $P^A(K)$ as a function of strike K, with K = 85, 85.5, 86, . . . , 133. Hand-in the plot of K $\rightarrow P^A(K)$

```
binTreeAmerican <- function(S=100, K=95,N=20,r=0.02,delta=0,h=1/40, sigma = 0.18, optionType="put") {

u <- exp((r-delta)*h + sigma * sqrt(h))
d <- exp((r-delta)*h - sigma * sqrt(h))
disc <- exp(-r*h)
q <- (exp( (r-delta)*h)-d)/(u-d)

    # Initialize arrays for option values
    V <- array(0, dim=c(N+1, N+1))

    # Calculate the terminal payoffs
    for (i in 0:N) {
```

```
        finalS <- S * u^i * d^(N-i)
        if (optionType == "call") {
            V[N+1, i+1] <- max(finalS - K, 0) # Terminal payoff for a call
        } else { optionType == "put"
            V[N+1, i+1] <- max(K - finalS, 0) # Terminal payoff for a put
        }
    }

    # Backward induction for American option
    for (j in N:1) {
        for (i in 0:(j-1)) {
            curS <- S * u^i * d^(j-1-i)
            holdValue <- (q * V[j+1, i+2] + (1 - q) * V[j+1, i+1]) * disc
            if (optionType == "call") {
                exerciseValue <- max(curS - K, 0)
            } else { # optionType == "put"
                exerciseValue <- max(K - curS, 0)
            }
            V[j, i+1] <- max(holdValue, exerciseValue) # Maximum of holding vs. exercising
        }
    }

    return(list(price = V[1,1], premia = V))
}
```

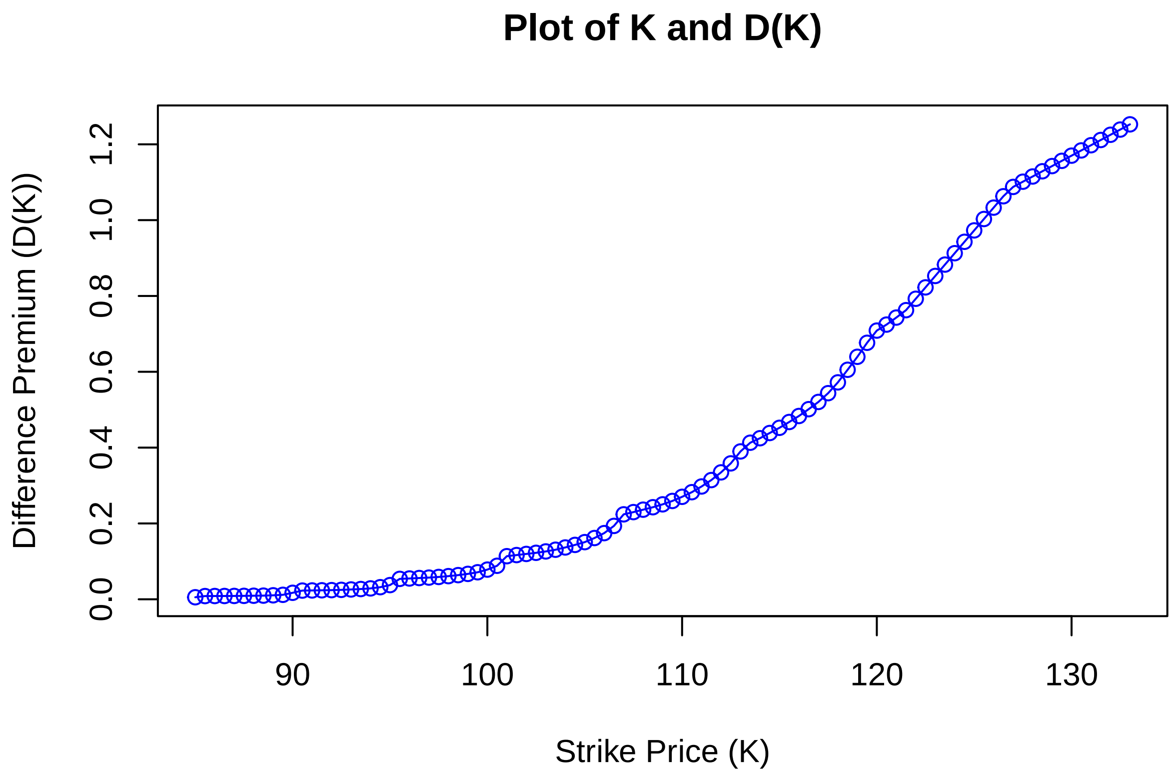
```
# Determing the put premium under each K value
price_p <- function(a){
  price_set <- numeric(length(a))
  for (i in seq_along(a)){
     price_k <- binTreeAmerican(K=a[i])
     price_set[i] <- price_k$premia[1,1]
  }
  names(price_set) <- paste("P", a, sep="")
  return(price_set)
}

K <- seq(85, 133, by=0.5)

Put_Premium_AM <- price_p(a = K)
```

```
# plot
plot(K,Put_Premium_AM,
     type = "o",
     col = "blue",
     xlab = "Strike Price (K)",
     ylab = "Put Premium (P(K))",
     main = "Plot of K and P(K) for American style")
```



2. Compute and plot the difference between the American and European Put premia $P^A(K) - P^E(K)$. For what strike is this difference the largest? When is it the smallest?

```
# genrate the difference of premia between american style and european style
Difference_AE <- Put_Premium_AM - Put_Premium

# ploting the difference with respect with strike K
plot(K,Difference_AE,
     type = "o",
     col = "blue",
     xlab = "Strike Price (K)",
     ylab = "Difference Premium (D(K))",
     main = "Plot of K and D(K)")
```




```r
# Identifying extremes
max_diff_k <- K[which.max(Difference_AE)]
min_diff_k <- K[which.min(Difference_AE)]

max_diff_k  # Strike price with the largest difference
```

```
## [1] 133
```

```r
min_diff_k  # Strike price with the smallest difference
```

```
## [1] 85
```

From the graph, it is clearly demonstrated that there is an increasing trend in the difference as the strike price rises. Therefore, the put premia may have the largest difference at the highest number of strike K and the smallest difference at the lowest number of strike K. After checking the maximum and minimum in R, we indeed have Strike price 133 with the largest difference, and Strike price 85 with the smallest difference.