Intro to Financial Engineering IEOR W4700

Homework 6

Linan Qiu
lq2137

October 25, 2015

Problem 1.

Problem 1(a).

To find q,

```
q_prob = function(r, delta_t, sigma) {
  u = exp(sigma*sqrt(delta_t))
  d = exp(-sigma*sqrt(delta_t))

  return((exp(r*delta_t) - d)/(u-d))
}
```

To build the stock tree,

```
build_stock_tree = function(S, sigma, delta_t, N) {
  tree = matrix(0, nrow=N+1, ncol=N+1)

  u = exp(sigma*sqrt(delta_t))
  d = exp(-sigma*sqrt(delta_t))

  for (i in 1:(N+1)) {
    for (j in 1:i) {
      tree[i,j] = S * u^(j-1) * d^((i-1)-(j-1))
    }
  }
  return(tree)
}
```

To value the binomial option using the stock tree generated,

```
value_binomial_option = function(tree, sigma, delta_t, r, X,
    type) {
  q = q_prob(r, delta_t, sigma)

  option_tree = matrix(0, nrow=nrow(tree), ncol=ncol(tree))
  if(type == 'put') {
    option_tree[nrow(option_tree),] = pmax(X - tree[nrow(
       tree),], 0)
  } else {
    option_tree[nrow(option_tree),] = pmax(tree[nrow(tree),]
        - X, 0)
  }

  for (i in (nrow(tree)-1):1) {
    for(j in 1:i) {
      option_tree[i, j] = ((1-q)*option_tree[i+1,j] + q*
         option_tree[i+1,j+1])/exp(r*delta_t)
    }
  }
  return(option_tree)
}
```

Putting this all together,

```
binomial_option = function(type, sigma, T, r, X, S, N) {
  q = q_prob(r=r, delta_t=T/N, sigma=sigma)
  tree = build_stock_tree(S=S, sigma=sigma, delta_t=T/N, N=N
    )
  option = value_binomial_option(tree, sigma=sigma, delta_t=
     T/N, r=r, X=X, type=type)
  delta = (option[2,2]-option[2,1])/(tree[2,2]-tree[2,1])
  return(list(q=q, stock=tree, option=option, price=option
     [1,1], delta=delta))
}
```

I coded this manually because none of the `R` packages (`fOption`, `m4fe`) seem to work (and be able to replicate the numbers on the slides). They also don't show the entire tree. So I coded my own. This code for Binomial European Option Pricing is (I made it open source) at https://github.com/linanqiu/binomial-european-option-r.

Using the variables in the question,

```
> print(binomial_option(type='put', sigma=0.33, T=1/4, r
   =0.05, X=48, S=50, N=3), 3)
```

```
$q
[1] 0.498

$stock
     [,1] [,2] [,3] [,4]
[1,] 50.0  0.0  0.0  0.0
[2,] 45.5 55.0  0.0  0.0
[3,] 41.3 50.0 60.5  0.0
[4,] 37.6 45.5 55.0 66.5

$option
      [,1]  [,2] [,3] [,4]
[1,]  2.25 0.000    0    0
[2,]  3.87 0.635    0    0
[3,]  6.47 1.271    0    0
[4,] 10.43 2.543    0    0

$price
[1] 2.25

$delta
[1] -0.339
```

The option price is 2.247762

Problem 1(b).

Shortcut function to calculate Δ from the tree produced by `binomial_option`:

```
delta = function(binomial_option, row, col) {
  stock_tree = binomial_option$stock
  option_tree = binomial_option$option
  return((option_tree[row+1, col+1] - option_tree[row+1, col
     ])/(stock_tree[row+1, col+1] - stock_tree[row+1, col]))
}
```

- At start, $S = 50$, so $\Delta = \dfrac{C_U - C_D}{S_U - S_D} = \dfrac{0.6353901 - 3.866524}{54.99739 - 45.45670} = -0.3386687$. However, this Δ is for buying a put. If we are selling (writing) a put, we use $-\Delta$ stocks, hence 0.3386687 stocks.

- If stock went up, we are at $S = 54.99739$ and $C = 0.6353901$. Then, $\Delta = \dfrac{0 - 1.2712151}{60.49427 - 50.00000} = -0.1211343$. Again, We use $-\Delta$ stocks, hence need 0.1211343 stocks.

- Now that stock went down, we are at $S = 50$, $C = 1.2712151$. Then $\Delta = \dfrac{0 - 2.5433006}{54.99739 - 45.45670} = -0.266574$. We use $-\Delta$ stocks, hence need 0.266574 stocks.

Problem 2.

Problem 2(a).

```
> print(binomial_option(type='call', sigma=0.33, T=1, r=0.1,
    X=100, S=100, N=6), 3)
$q
[1] 0.529

$stock
      [,1]  [,2]  [,3] [,4] [,5] [,6] [,7]
[1,] 100.0   0.0   0.0    0    0    0    0
[2,]  87.4 114.4   0.0    0    0    0    0
[3,]  76.4 100.0 130.9    0    0    0    0
[4,]  66.8  87.4 114.4  150    0    0    0
[5,]  58.3  76.4 100.0  131  171    0    0
[6,]  51.0  66.8  87.4  114  150  196    0
[7,]  44.6  58.3  76.4  100  131  171  224

$option
      [,1]  [,2]  [,3] [,4] [,5] [,6] [,7]
[1,] 17.28  0.00  0.00  0.0  0.0  0.0    0
[2,]  7.94 26.15  0.00  0.0  0.0  0.0    0
[3,]  2.26 13.27 38.46  0.0  0.0  0.0    0
[4,]  0.00  4.34 21.65 54.7  0.0  0.0    0
[5,]  0.00  0.00  8.36 34.2 74.7  0.0    0
[6,]  0.00  0.00  0.00 16.1 51.5 97.8    0
[7,]  0.00  0.00  0.00  0.0 30.9 71.4  124

$price
[1] 17.3

$delta
[1] 0.674
```

Problem 2(b).

```
> print(binomial_option(type='put', sigma=0.33, T=1, r=0.1,
   X=100, S=100, N=6), digits=3)
$q
[1] 0.529

$stock
      [,1]  [,2]  [,3] [,4] [,5] [,6] [,7]
[1,] 100.0   0.0   0.0    0    0    0    0
[2,]  87.4 114.4   0.0    0    0    0    0
[3,]  76.4 100.0 130.9    0    0    0    0
[4,]  66.8  87.4 114.4  150    0    0    0
[5,]  58.3  76.4 100.0  131  171    0    0
[6,]  51.0  66.8  87.4  114  150  196    0
[7,]  44.6  58.3  76.4  100  131  171  224

$option
      [,1]  [,2]  [,3]     [,4] [,5] [,6] [,7]
[1,]  7.76  0.00  0.00 0.00e+00    0    0    0
[2,] 12.55  3.73  0.00 0.00e+00    0    0    0
[3,] 19.43  6.82  1.09 0.00e+00    0    0    0
[4,] 28.37 12.07  2.35 1.42e-15    0    0    0
[5,] 38.38 20.34  5.08 3.05e-15    0    0    0
[6,] 47.36 31.59 10.95 6.59e-15    0    0    0
[7,] 55.44 41.66 23.62 1.42e-14    0    0    0

$price
[1] 7.76

$delta
[1] -0.326
```

Problem 2(c).

To satisfy put call parity, let C_C be price of call option and C_P be price of put option. Then, buying a call and selling a put should give us the same cash flow as a forward on the underlier.

$$
\begin{aligned}
C_C - C_P &= S - \frac{X}{e^r} \\
17.27535 - 7.759088 &= 100 - \frac{100}{e^{0.1}} \\
9.516258 &= 9.516258
\end{aligned}
$$

Problem 3.

Program written in above sections. Code is available at https://github.com/linanqiu/binomial-european-option-r.

```
periods = seq(100, 120)
option_price_vary_period = function(period) {
  print(period)
  option = binomial_option(type='call', sigma=0.2, T=1, r=0, X=100, S=100, N=period)
  return(option$price)
}
values = sapply(periods, option_price_vary_period)
library(ggplot2)
data = as.data.frame(list(periods=periods, values=values))
plot = ggplot(data=data) + geom_line(aes(x=periods, y=values)) + labs(title="Call Value", x="Periods", y="Value")
plot
```

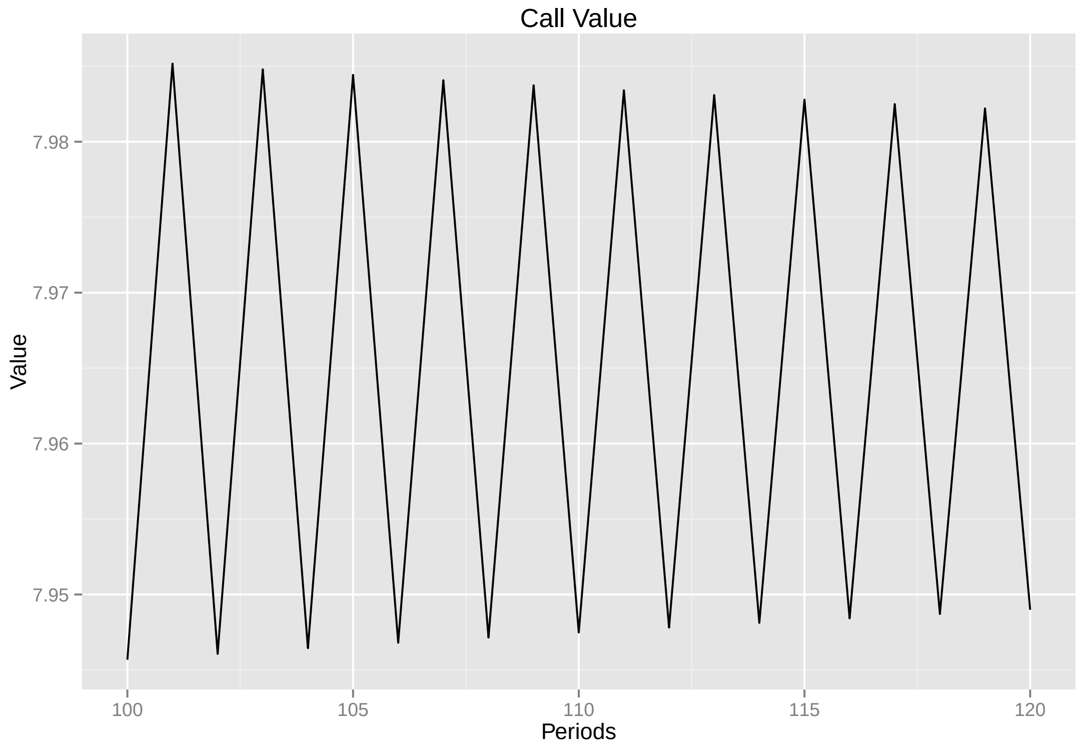

Figure 1: Plot of call option value calculated using the function above against periods (from 100 to 120)

Why do 100 to 120 when one can do more!

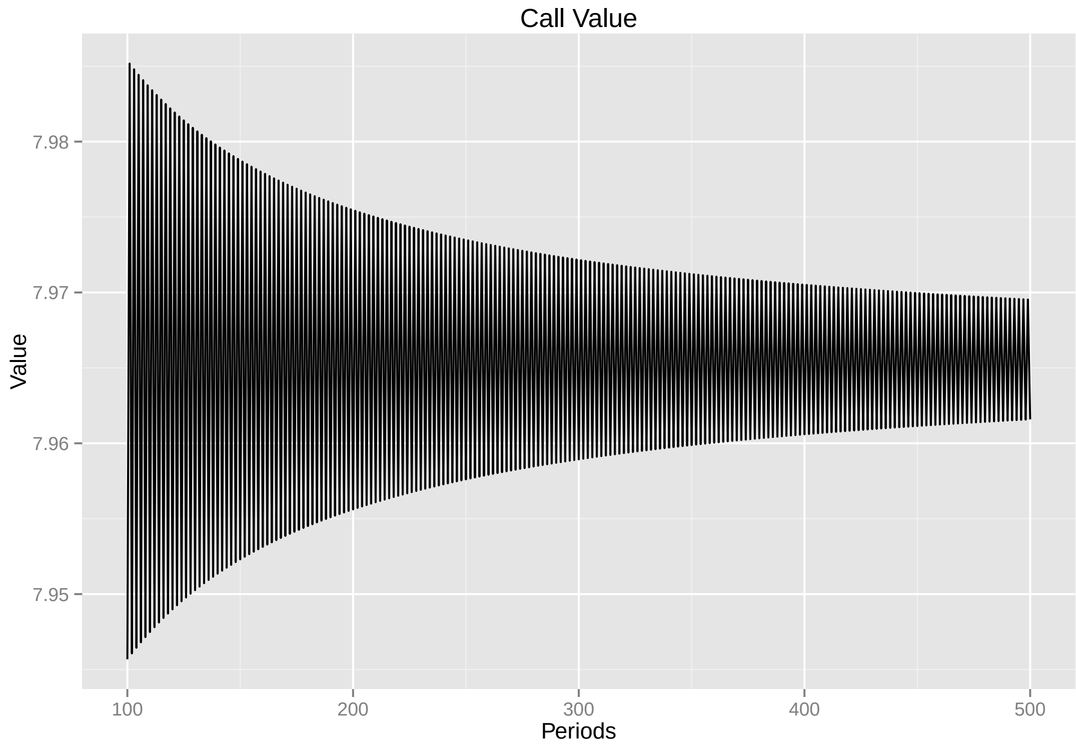

Figure 2: Plot of call option value calculated using the function above against periods (from 100 to 500). Also this is pretty pretty. After 500 the code becomes a little slow since I'm generating N^2 matrices.

Okay I managed to speed it up by parallelizing it. Let's try a 1000 periods now (this took a minute on 8 CPU cores). The code is

```
library(parallel)
cl = makeCluster(8)
clusterEvalQ(cl, source('binomial.R'))
periods = seq(100, 1000)
periods = sample(periods)
valuesPar = parSapply(cl=cl, periods, option_price_vary_
    period)
data = as.data.frame(list(periods=periods, values=valuesPar)
    )
plot = ggplot(data=data) + geom_line(aes(x=periods, y=values
    )) + labs(title="Call Value", x="Periods", y="Value")
plot
stopCluster(cl)
```

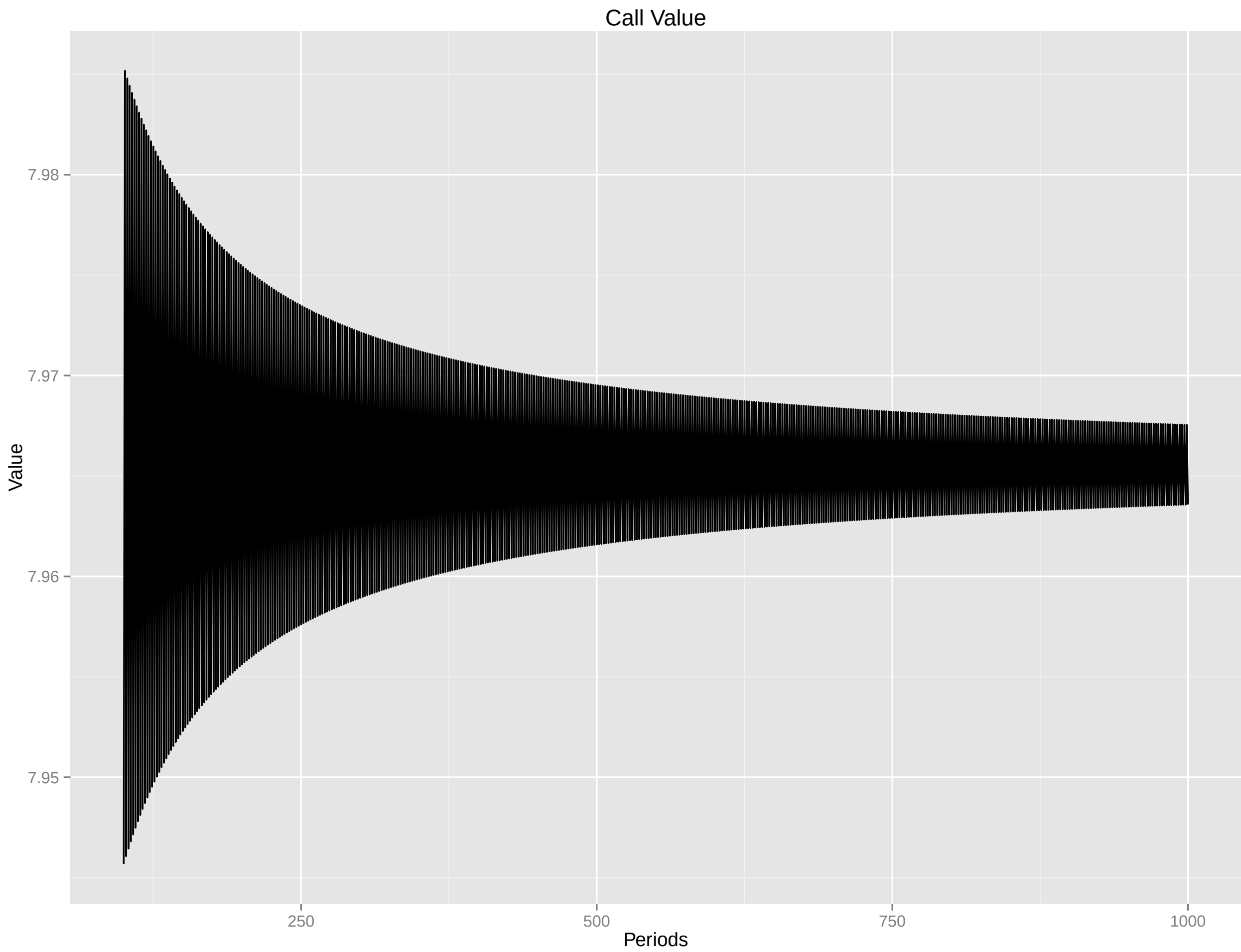

Figure 3: Plot of call option value calculated using the function above against periods (from 100 to 1000). This is almost beautiful.

Problem 4.

Build the tree manually. Let's find q the risk neutral "probability" first.

$$S = 50 = \frac{qS_U + (1-q)S_D}{R} = \frac{q55 + (1-q)45}{e^{0.1*0.5}}$$

Solving for q, $q = 0.7563555$

```
> uniroot(function(q) {(q*55 + (1-q)*45)/exp(0.1*0.5) - 50},
    interval=c(-1, 1))
$root
[1] 0.7563555

$f.root
[1] 0

$iter
[1] 1

$init.it
[1] NA

$estim.prec
[1] 1.756355
```

Then,

$$P = \frac{qP_U + (1-q)P_D}{R} = \frac{0 + (1-0.7563555)(50-45)}{e^{0.1*0.5}} = 1.158809$$

To verify this (since I'm revising for the midterm anyway), let's replicate the riskless bond. Consider a portfolio with Δ stocks and 1 put.

- When $S_U = 55$, $P_U = 0$. Portfolio is worth $\Delta 55$.
- When $S_D = 45$, $P_D = 5$. Portfolio is worth $\Delta 45 + 5$.

$$\Delta 55 = \Delta 45 + 5$$

Then, $\Delta = 0.5$, ie. we must long 0.5 stocks. The value of both portfolios are

- When $S_U = 55$, $P_U = 0$. Portfolio is worth $\Delta 55 = 27.5$.
- When $S_D = 45$, $P_D = 5$. Portfolio is worth $\Delta 45 + 5 = 27.5$.

That means the portfolio must be worth $\frac{27.5}{e^{0.1*0.5}} = 26.15881$ presently. That means

$$P + \Delta S = 26.15881 = P + 0.5(50)$$
$$P = 1.158809$$

The value of the put option is 1.158809 which verifies the answer from using binomial trees.

Problem 5.

Let D be the value of the derivative.

$$S = 50 = \frac{qS_U + (1-q)S_D}{R} = \frac{q27 + (1-q)23}{e^{0.1/6}}$$

Solving for q, $q = 0.6050396$

```
> uniroot(function(q) {(q*27 + (1-q)*23)/exp(0.1/6) - 25},
  interval=c(-1, 1))
$root
[1] 0.6050396

$f.root
[1] 0

$iter
[1] 1

$init.it
[1] NA

$estim.prec
[1] 1.60504
```

Then,

$$D = \frac{qD_U + (1-q)D_D}{R} = \frac{(0.6050396)27^2 + (1-0.6050396)23^2}{e^{0.1/6}} = 639.2642$$

Can be verified via portfolio replication method.

Suppose portfolio comprises Δ stocks and 1 D.

- When $S_U = 27$, $D_U = 27^2$. Portfolio is worth $\Delta 27 + D_U$
- When $S_D = 23$, $D_D = 23^2$. Portfolio is worth $\Delta 23 + D_D$

$$\Delta 27 + D_U = \Delta 23 + D_D$$
$$\Delta 27 + 27^2 = \Delta 23 + 23^2$$
$$\Delta = -50$$

We short 50 stocks. Then in both states, portfolio is worth $\Delta 27 + 27^2 = -621 = \Delta 23 + 23^2$
Then the value of both portfolios before two months is $\dfrac{\Delta 27 + 27^2}{e^{0.1/6}} = -610.7358$.

$$D + \Delta S = D - 50(25) = -610.7358$$
$$D = 639.2642$$

Verifies the answer above.