Intro to Financial Engineering IEOR W4700

Homework 8

Linan Qiu
lq2137

November 26, 2015

Problem 1.

By put call parity,

$$C - P = S_0 e^{(-qT)} - K e^{(-rT)}$$

Then, given the price of the call, S_t and K and r and T,

$$P = C - S_0 e^{(-qT)} + K e^{(-rT)} = 10 - 250 e^{\left(-0.04 * \frac{1}{r}\right)} + 245 e^{\left(-0.06 * \frac{1}{4}\right)} = 3.839967$$

Problem 2.

The adjusted stock price accounting for the dividend is

$$100 - 2 e^{\left(-0.06 * \frac{1}{6}\right)} = 98.0199$$

Evolving the stock price using a binomial tree as follows:

```
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)
}
```

Running the following code:

```
> stock_tree = build_stock_tree(S=100 - 2*exp(-0.06/6), 0.3, 1/12, 4)
> stock_tree
         [,1]      [,2]     [,3]     [,4]     [,5]
[1,] 98.01990   0.00000   0.0000   0.0000   0.0000
[2,] 89.88832 106.88709   0.0000   0.0000   0.0000
[3,] 82.43132  98.01990 116.5564   0.0000   0.0000
[4,] 75.59294  89.88832 106.8871 127.1005   0.0000
[5,] 69.32186  82.43132  98.0199 116.5564 138.5984
```

This is the stock tree after adjusting for dividends.

Now we have to add the discounted dividend back to the tree. The first row gets $2e^{\left(-0.06*\frac{2}{12}\right)}$ added, and the second row gets $2e^{\left(-0.06*\frac{1}{12}\right)}$ back.

```
> stock_tree[1,1] = stock_tree[1,1] + 2*exp(-0.06*2/12)
> stock_tree
          [,1]      [,2]     [,3]     [,4]     [,5]
[1,] 100.00000   0.00000   0.0000   0.0000   0.0000
[2,]  89.88832 106.88709   0.0000   0.0000   0.0000
[3,]  82.43132  98.01990 116.5564   0.0000   0.0000
[4,]  75.59294  89.88832 106.8871 127.1005   0.0000
[5,]  69.32186  82.43132  98.0199 116.5564 138.5984
> stock_tree[2,1] = stock_tree[2,1] + 2*exp(-0.06*1/12)
> stock_tree[2,2] = stock_tree[2,2] + 2*exp(-0.06*1/12)
> stock_tree
          [,1]      [,2]     [,3]     [,4]     [,5]
[1,] 100.00000   0.00000   0.0000   0.0000   0.0000
[2,]  91.87834 108.87712   0.0000   0.0000   0.0000
[3,]  82.43132  98.01990 116.5564   0.0000   0.0000
[4,]  75.59294  89.88832 106.8871 127.1005   0.0000
[5,]  69.32186  82.43132  98.0199 116.5564 138.5984
```

The resulting binary tree is shown directly above.

Problem 3.

Let the value of the binary option at time 0 be B_0.

$$B_0 = e^{(-rT)}\mathrm{E}(\theta(S_T - K)) = e^{(-rT)}P(S_T \geq K)$$

Now the probability that $S_T \geq K$ can be found by looking at the distribution of $\log S$, since $P(S_T \geq K) = P(\log S_T \geq \log K)$.

We know that S_T is log-normally distributed, hence $\log S_T$ is normally distributed with the following statistics:

$$\mathrm{E}(\log S_T) = \log S_0 + \left(\mu - \frac{1}{2}\sigma^2\right) T$$

$$\mathrm{Var}\left(\log S_T\right) = \sigma^2 T$$

Then,

$$Q(S_T) = \frac{\log S_T - \log S_0 - \left(\mu - \frac{1}{2}\sigma^2\right) T}{\sigma\sqrt{T}} \sim \mathrm{Norm}(0,1)$$

Then,

$$\begin{aligned}
P(S_T \geq K) &= P(\log S_T \geq \log K) \\
&= P\left(\frac{\log S_T - \log S_0 - \left(\mu - \frac{1}{2}\sigma^2\right) T}{\sigma\sqrt{T}} \geq \frac{\log K - \log S_0 - \left(\mu - \frac{1}{2}\sigma^2\right) T}{\sigma\sqrt{T}}\right) \\
&= 1 - P\left(\frac{\log S_T - \log S_0 - \left(\mu - \frac{1}{2}\sigma^2\right) T}{\sigma\sqrt{T}} < \frac{\log K - \log S_0 - \left(\mu - \frac{1}{2}\sigma^2\right) T}{\sigma\sqrt{T}}\right) \\
&= 1 - \mathrm{N}(Q(K)) \\
&= \mathrm{N}(-Q(K))
\end{aligned}$$

where N is the normal distribution function. Then the value B_0 is

$$B_0 = e^{(-rT)}\mathrm{N}\left(\frac{\log S_0 - \log K + \left(r - \frac{1}{2}\sigma^2\right) T}{\sigma\sqrt{T}}\right) = e^{(-rT)}\mathrm{N}(d_2)$$

where d_2 is as defined in Hull / slides. We know from risk neutral pricing that r is the risk free rate.

Problem 4.

The payoffs from this option is perfectly replicated by a portfolio of:

- long 1 vanilla call option with strike K
- long K digital call option with strike K

This is shown by the payoffs: let A be the binary asset-or-nothing call.

$$\begin{aligned}
A &= e^{(-rT)}\mathrm{E}(S\theta(S-K)) \\
&= e^{(-rT)}\mathrm{E}((S-K)\theta(S-K) + K\theta(S-K)) \\
&= e^{(-rT)}\mathrm{E}(\max(S-K,0)) + Ke^{(-rT)}\mathrm{E}(\theta(S-K)) \\
&= C + KB \\
&= S_0\mathrm{N}(d_1) - Ke^{(-rT)}\mathrm{N}(d_2) + Ke^{(-rT)}\mathrm{N}(d_2) \\
&= S_0\mathrm{N}(d_1)
\end{aligned}$$

where d_1 is as defined in Hull / slides.

Problem 5.

The European Call value using BSM is calculated as:

```
d1 = function(S, K, r, sigma, T) {
  return((log(S/K) + (r+(sigma^2)/2)*T)/(sigma*sqrt(T)))
}

d2 = function(S, K, r, sigma, T) {
  return(d1(S, K, r, sigma, T) - sigma*sqrt(T))
}

bsm_call = function(sigma, K, S, r, T) {
  d1_val = d1(S, K, r, sigma, T)
  d2_val = d2(S, K, r, sigma, T)
  return(list(price=S*pnorm(d1_val) - K*exp(-r*T)*pnorm(d2_val), delta=
      pnorm(d1_val)))
}
```

```
> bsm_call(sigma=0.1, T=1, r=0.08, K=51, S=50*exp(-0.1))
$price
[1] 1.066008

$delta
[1] 0.3639102
```

For an American option, let the dividend yield be d and the risk free probability be q.

$$qS_0u + (1-q)S_0d = S_0e^{((r-d)\Delta t)}$$

Setting q to be the subject,

$$q = \frac{e^{((r-d)\Delta t)} - d}{u-d}$$

In other words, all we have to do to account for the continuous dividend yield is to set q to the new quantity.

Then, we modify the binomial program we used in the previous homework. The stock tree building procedure remains the same:

```
build_stock_tree = function(S, sigma, T, N) {
  delta_t = 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)
}
```

Particularly, q calculation now accounts for dividends.

```
q_prob = function(r, delta_t, sigma, div=0) {
  u = exp(sigma*sqrt(delta_t))
  d = exp(-sigma*sqrt(delta_t))

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

As for building the option tree, I chose to use 4 trees (matrices) instead of a proper data structures (which I'd have used in any other language with proper data structures and object orientedness like `Python` or `Java` or even `JavaScript`). Unfortunately, I am using `R` and am too lazy to switch languages.

```
  delta_t = T / N
  q = q_prob(r, delta_t, sigma, div=div)

  underlying_tree = tree
  value_ne_tree = matrix(0, nrow=nrow(tree), ncol=ncol(tree))
  value_e_tree = matrix(0, nrow=nrow(tree), ncol=ncol(tree))
  option_value_tree = matrix(0, nrow=nrow(tree), ncol=ncol(tree))

  ## ONLY CALL OPTIONS NOW

  # set not exercise value to 0
  value_ne_tree[nrow(tree),] = 0
  # set exercise value to S - X
  value_e_tree[nrow(tree),] = tree[nrow(tree), ] - X
  # set option value to whichever is higher
  option_value_tree[nrow(tree),] = pmax(value_ne_tree[nrow(tree),], value_e_tree[nrow(tree),])


  for (i in (nrow(tree)-1):1) {
    for(j in 1:i) {
      # not exercise value is probability weighted discounted
      value_ne_tree[i, j] = ((1-q)*option_value_tree[i+1,j] + q*option_value_tree[i+1,j+1])/exp(r*delta_t)
      # exercise value = S - X
      value_e_tree[i, j] = tree[i, j] - X
      # set option value to whichever is higher
      option_value_tree[i, j] = max(value_ne_tree[i, j], value_e_tree[i, j])
    }
  }

  delta = (option_value_tree[2, 2] - option_value_tree[2, 1])/(tree[2, 2] - tree[2, 1])

  return(list(tree=tree, value_ne_tree=value_ne_tree, value_e_tree=value_e_tree, option_value_tree=option_value_tree, q=q, delta=delta))
}
```

Running the code using the following:

```
sigma=0.1
S=50
T=1
N=100
X=51
r=0.08
div=0.1

stock_tree = build_stock_tree(S=S, sigma=sigma, T=T, N=N)
```

```
option_tree = value_binomial_option(stock_tree, sigma=sigma, N=N, T=T, r=
    r, X=X, type="call", div=div)
```

This gave the following results (compared with European BSM pricing):

- European Call using BSM: $P = 1.066008$, $\Delta = 0.3639102$
- American Call using Binomial Tree: $P = 1.157862$, $\Delta = 0.3698668$

The results are in the following pages:

Problem 5(a). Stock Tree

```
> round(option_tree$tree[1:11, 1:11], 2)
       [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]  [,9] [,10] [,11]
 [1,] 50.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
 [2,] 49.50 50.50  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
 [3,] 49.01 50.00 51.01  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
 [4,] 48.52 49.50 50.50 51.52  0.00  0.00  0.00  0.00  0.00  0.00  0.00
 [5,] 48.04 49.01 50.00 51.01 52.04  0.00  0.00  0.00  0.00  0.00  0.00
 [6,] 47.56 48.52 49.50 50.50 51.52 52.56  0.00  0.00  0.00  0.00  0.00
 [7,] 47.09 48.04 49.01 50.00 51.01 52.04 53.09  0.00  0.00  0.00  0.00
 [8,] 46.62 47.56 48.52 49.50 50.50 51.52 52.56 53.63  0.00  0.00  0.00
 [9,] 46.16 47.09 48.04 49.01 50.00 51.01 52.04 53.09 54.16  0.00  0.00
[10,] 45.70 46.62 47.56 48.52 49.50 50.50 51.52 52.56 53.63 54.71  0.00
[11,] 45.24 46.16 47.09 48.04 49.01 50.00 51.01 52.04 53.09 54.16 55.26
```

∞

Problem 5(b). Not Exercise Value

```
> round(option_tree$value_ne_tree[1:11, 1:11], 2)
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11]
 [1,] 1.16 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00  0.00  0.00
 [2,] 0.98 1.35 0.00 0.00 0.00 0.00 0.00 0.00 0.00  0.00  0.00
 [3,] 0.82 1.15 1.56 0.00 0.00 0.00 0.00 0.00 0.00  0.00  0.00
 [4,] 0.68 0.97 1.34 1.80 0.00 0.00 0.00 0.00 0.00  0.00  0.00
 [5,] 0.56 0.81 1.13 1.55 2.07 0.00 0.00 0.00 0.00  0.00  0.00
 [6,] 0.46 0.67 0.96 1.32 1.79 2.37 0.00 0.00 0.00  0.00  0.00
 [7,] 0.37 0.55 0.80 1.12 1.54 2.06 2.70 0.00 0.00  0.00  0.00
 [8,] 0.30 0.45 0.66 0.94 1.31 1.78 2.36 3.05 0.00  0.00  0.00
 [9,] 0.23 0.36 0.54 0.79 1.11 1.53 2.05 2.69 3.45  0.00  0.00
[10,] 0.18 0.29 0.44 0.65 0.93 1.30 1.77 2.35 3.05  3.87  0.00
[11,] 0.14 0.23 0.35 0.53 0.78 1.10 1.52 2.04 2.68  3.44  4.34
```

Problem 5(c). Exercise Values

```
> round(option_tree$value_e_tree[1:11, 1:11], 2)
       [,1]  [,2]  [,3]  [,4]  [,5]  [,6] [,7] [,8] [,9] [,10] [,11]
 [1,] -1.00  0.00  0.00  0.00  0.00  0.00 0.00 0.00 0.00  0.00  0.00
 [2,] -1.50 -0.50  0.00  0.00  0.00  0.00 0.00 0.00 0.00  0.00  0.00
 [3,] -1.99 -1.00  0.01  0.00  0.00  0.00 0.00 0.00 0.00  0.00  0.00
 [4,] -2.48 -1.50 -0.50  0.52  0.00  0.00 0.00 0.00 0.00  0.00  0.00
 [5,] -2.96 -1.99 -1.00  0.01  1.04  0.00 0.00 0.00 0.00  0.00  0.00
 [6,] -3.44 -2.48 -1.50 -0.50  0.52  1.56 0.00 0.00 0.00  0.00  0.00
 [7,] -3.91 -2.96 -1.99 -1.00  0.01  1.04 2.09 0.00 0.00  0.00  0.00
 [8,] -4.38 -3.44 -2.48 -1.50 -0.50  0.52 1.56 2.63 0.00  0.00  0.00
 [9,] -4.84 -3.91 -2.96 -1.99 -1.00  0.01 1.04 2.09 3.16  0.00  0.00
[10,] -5.30 -4.38 -3.44 -2.48 -1.50 -0.50 0.52 1.56 2.63  3.71  0.00
[11,] -5.76 -4.84 -3.91 -2.96 -1.99 -1.00 0.01 1.04 2.09  3.16  4.26
```

Problem 5(d). Option Value

Option value is the max of exercise value or not exercise value at each stage.

```
> round(option_tree$option_value[1:11, 1:11], 2)
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11]
 [1,] 1.16 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00  0.00  0.00
 [2,] 0.98 1.35 0.00 0.00 0.00 0.00 0.00 0.00 0.00  0.00  0.00
 [3,] 0.82 1.15 1.56 0.00 0.00 0.00 0.00 0.00 0.00  0.00  0.00
 [4,] 0.68 0.97 1.34 1.80 0.00 0.00 0.00 0.00 0.00  0.00  0.00
 [5,] 0.56 0.81 1.13 1.55 2.07 0.00 0.00 0.00 0.00  0.00  0.00
 [6,] 0.46 0.67 0.96 1.32 1.79 2.37 0.00 0.00 0.00  0.00  0.00
 [7,] 0.37 0.55 0.80 1.12 1.54 2.06 2.70 0.00 0.00  0.00  0.00
 [8,] 0.30 0.45 0.66 0.94 1.31 1.78 2.36 3.05 0.00  0.00  0.00
 [9,] 0.23 0.36 0.54 0.79 1.11 1.53 2.05 2.69 3.45  0.00  0.00
[10,] 0.18 0.29 0.44 0.65 0.93 1.30 1.77 2.35 3.05  3.87  0.00
[11,] 0.14 0.23 0.35 0.53 0.78 1.10 1.52 2.04 2.68  3.44  4.34
```

Problem 6.

Problem 6(a).

The only part of the program we need to modify is:

```
value_binomial_option = function(tree, sigma, N, T, r, alpha=3, div=0) {
  delta_t = T / N
  q = q_prob(r, delta_t, sigma, div=div)

  option_tree = matrix(0, nrow=nrow(tree), ncol=ncol(tree))

  # option pays S^alpha at expiration
  option_tree[nrow(tree),] = tree[nrow(tree), ]^alpha

  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)
    }
  }

  delta = (option_tree[2, 2] - option_tree[2, 1])/(tree[2, 2] - tree[2,
     1])

  return(list(tree=tree, option_tree=option_tree, price=option_tree[1,1],
      q=q, delta=delta))
}
```

to account for the introduction of α and the different expiration price of power options.

Running this:

```
sigma=0.3
S=100
T=1/4
N=100
r=0.1
stock_tree = build_stock_tree(S=S, sigma=sigma, T=T, N=N)
option_tree = value_binomial_option(stock_tree, sigma=sigma, N=N, T=T, r=
    r, alpha=3)
```

We find that the price of the option is $P = 1124610.5$. We also obtain the following trees:

(i). Stock Tree

```
> round(option_tree$tree[1:11, 1:11], 2)
        [,1]   [,2]   [,3]   [,4]   [,5]   [,6]   [,7]   [,8]   [,9]  [,10]  [,11]
 [1,] 100.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00
 [2,]  98.51 101.51   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00
 [3,]  97.04 100.00 103.05   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00
 [4,]  95.60  98.51 101.51 104.60   0.00   0.00   0.00   0.00   0.00   0.00   0.00
 [5,]  94.18  97.04 100.00 103.05 106.18   0.00   0.00   0.00   0.00   0.00   0.00
 [6,]  92.77  95.60  98.51 101.51 104.60 107.79   0.00   0.00   0.00   0.00   0.00
 [7,]  91.39  94.18  97.04 100.00 103.05 106.18 109.42   0.00   0.00   0.00   0.00
 [8,]  90.03  92.77  95.60  98.51 101.51 104.60 107.79 111.07   0.00   0.00   0.00
 [9,]  88.69  91.39  94.18  97.04 100.00 103.05 106.18 109.42 112.75   0.00   0.00
[10,]  87.37  90.03  92.77  95.60  98.51 101.51 104.60 107.79 111.07 114.45   0.00
[11,]  86.07  88.69  91.39  94.18  97.04 100.00 103.05 106.18 109.42 112.75 116.18
```

(ii). Option Tree

```
> round(option_tree$option_tree[1:11, 1:11], 2)
           [,1]      [,2]      [,3]      [,4]    [,5]    [,6]    [,7]    [,8]    [,9]   [,10]   [,11]
 [1,] 1124610.5       0.0       0.0       0.0       0       0       0       0       0       0       0
 [2,] 1073862.9 1174993.2       0.0       0.0       0       0       0       0       0       0       0
 [3,] 1025405.3 1121972.1 1227633.1       0.0       0       0       0       0       0       0       0
 [4,]  979134.4 1071343.7 1172236.7 1282631.2       0       0       0       0       0       0       0
 [5,]  934951.4 1022999.8 1119340.0 1224753.1 1340093       0       0       0       0       0       0
 [6,]  892762.1  976837.3 1068830.3 1169486.6 1279622 1400130       0       0       0       0       0
 [7,]  852476.6  932758.0 1020599.8 1116714.1 1221880 1336949 1462856       0       0       0       0
 [8,]  814009.0  890667.7  974545.7 1066322.9 1166743 1276620 1396845 1528392       0       0       0
 [9,]  777277.2  850476.7  930569.8 1018205.5 1114094 1219013 1333813 1459424 1596864       0       0
[10,]  742202.9  812099.4  888578.2  972259.4 1063821 1164006 1273625 1393568 1524806 1668404       0
[11,]  708711.3  775453.7  848481.5  928386.7 1015817 1111481 1216154 1330684 1456000 1593118 1743149
```

Problem 6(b).

Let P be the power option.

$$P = S^{\alpha}$$

$dS = \mu S dt + \sigma S dZ$. Recall Ito's lemma which states that

$$dP = \left(\frac{\delta P}{\delta S} \mu S + \frac{\delta P}{\delta t} + \frac{1}{2} \frac{\delta^2 P}{\delta S^2} \sigma^2 S^2 \right) dt + \frac{\delta P}{\delta S} \sigma S dZ$$

Now,

$$\begin{aligned} \frac{\delta P}{\delta S} &= \alpha S^{\alpha - 1} \\ \frac{\delta^2 P}{\delta S^2} &= \alpha(\alpha - 1) S^{\alpha - 2} \\ \frac{\delta P}{\delta t} &= 0 \end{aligned}$$

Then,

$$\begin{aligned} dP &= \left(\mu \alpha S^{\alpha} + \frac{\alpha(\alpha - 1)}{2} \sigma^2 S^{\alpha} \right) dt + \alpha \sigma S^{\alpha} dZ \\ &= \left(\mu \alpha P + \frac{\alpha(\alpha - 1)}{2} \sigma^2 P \right) dt + \alpha \sigma P dZ \end{aligned}$$

$$\frac{dP}{P} = \left(\mu \alpha + \frac{\alpha(\alpha - 1)}{2} \sigma^2 \right) dt + \alpha \sigma dZ$$

Then we know that $\frac{dP}{P}$ is a Wiener process with drift constant $\left(\mu \alpha + \frac{\alpha(\alpha - 1)}{2} \sigma^2 \right)$ and variance $(\alpha \sigma)^2$.

Then, we know that the expected change between time 0 and time T is

$$\mathrm{E}(P_T) = P_0 \exp \left[\left(\mu \alpha + \frac{\alpha(\alpha - 1)}{2} \sigma^2 \right) T \right]$$

Let the value of the option be V.

$$V = e^{(-rT)}\mathrm{E}(P_T) = e^{(-rT)}P_0 \exp\left[\left(\mu\alpha + \frac{\alpha(\alpha-1)}{2}\sigma^2\right)T\right]$$

```
black_scholes_power = function(alpha, r, T, sigma, S) {
  discount = exp(-r*T)
  p0 = S^alpha
  expectation = exp((r*alpha + (alpha * (alpha - 1)) / 2 * sigma^2)*T)
  return(discount*p0*expectation)
}

black_scholes_power(alpha=3, r=0.1, T=0.25, sigma=0.3, S=100)
```

Now $P_0 = 100^3 = 1000000$, $r = 0.1$, $\mu = r = 0.1$ by risk neutral pricing, $T = 0.25$, $\alpha = 3$, $\sigma = 0.3$.

Then, $V = 1124682$