FE – 621 Computational Method in Finance (Assignment 2)

Student: Riley Heiman

Instructor: Sveinn Ólafsson

TA: Zhiyang Deng

1. Introduction

This paper outlines three pricing application of binomial tree. The first is European and American call & put options. The second is the barrier options such as Knock-Out. The final application is installment options. This paper exams these three methods.

2. Binomial Tree

The additive binomial tree can be used to price American & European call and put options. Appendix one outlines the mathematics of this model. Additionally, code is provided to deploy this model, along with a function to plot the binomial tree.

Figure 1 shows the option price of the binomial tree with respect to the size of the tree. The red line represents the closed-form Black-Scholes European call price. The plot shows the binomial tree method approaches the Black-Scholes price as the tree increases. The final price converges to $8.56

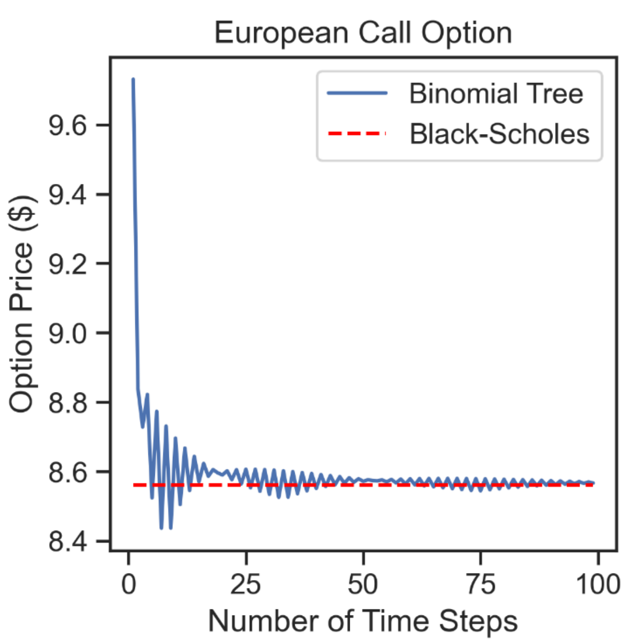


Figure 1. *T = .75, S = 50, K = $45, sigma = 35%, risk-free rate = 0%, N = 100*

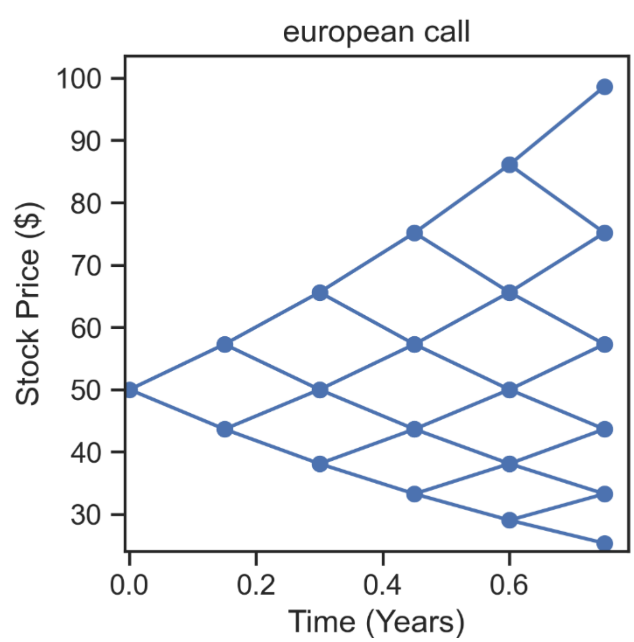


Figure 2. Same inputs as Figure 1, expect N = 5

Figure 2 depicts the binomial tree and future stock prices over time.

3. European Up-and-Out (Knock-Out)

The code for the vanilla European and American options can be altered to calculate the Knockout barrier options. Appendix 2 outlines more specifics on the mathematics. Figure 3 shows the binomial tree price converging to a closed-form solution. Figure 4 shows a zoomed in version of the same plot. There is an interesting zig-zag trend. The option price slowly approaches the closed form solution, and then jumps up. This process is slowly diminished as the tree size increases.

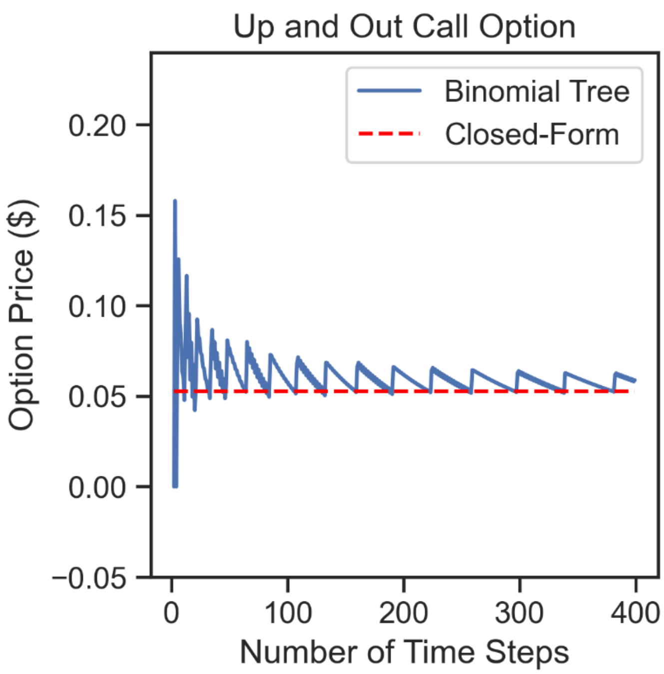


Figure 3. *S = $10, K = $10, T = .3,* σ = 20% , *risk-free rate = 1%, H = $11*

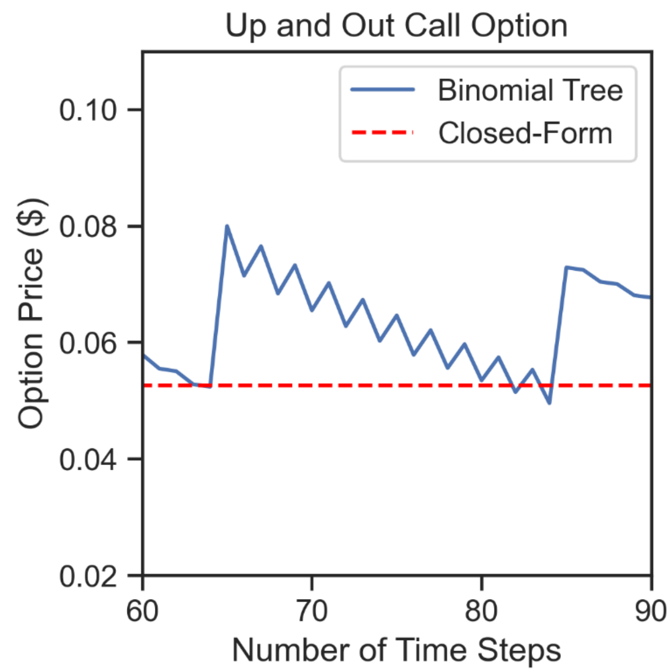


Figure 4. *Zoomed in version of figure 3.*

There are many different type of barrier options. For example, an Up-and-In call option is similar in many ways, expect the option cannot be exercised until the barrier is hit. $(S_T > H)$

The next page outlines a numerical recipe to value a Up-and-In call and put option.

3 c) Up and In Barrier Option (Numerical Recipe)

Inputs: $S_0,\ K,\ r,\ \sigma,\ T,\ N,\ H$

Step 1: Solve for the variables below:

$$u = e^{\sigma\sqrt{\Delta t}} \quad , \qquad d = \frac{1}{u}$$

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

Step 2: Build the tree, by calculating future stock prices

$$S_{i,j} = S_0 * u^i * d^j$$

i = # of up moves

j = # of down moves

Step 3: Calculate the option value at terminal notes

If $S_{n,k} > H$ then

$$V_{N,K}^A = \left(K - S_{N,K}\right)^+ \ \text{(Put)}$$

$$V_{N,K}^A = \left(S_{N,K} - K\right)^+ \ \text{(Call)}$$

Else: $V_{N,K}^A = \$0$

Step 4: Work backwards in the tree. Beginning at one step behind the terminal nodes:

If $S_{n,k} > H$ then,

$$V_{n,k}^A = max\left\{\left(K - S_{N,K}\right)^+ \ , \ e^{-r\Delta t}\left[q\, V_{n+1,k+1}^A + (1-q)V_{n+1,k}^A\right]\right\} \ \text{(American Put)}$$

Else: $V_{N,K}^A = \$0$

4. Installment Options

The python code for the vanilla European & American options can be altered slightly to calculate the price of the installment option. Figure 5 shows the arbitrage free price when $p = V_0(p)$. For the inputs provided, the graph shows the intersection, which values the installments at $\approx \$\mathbf{3.96}$

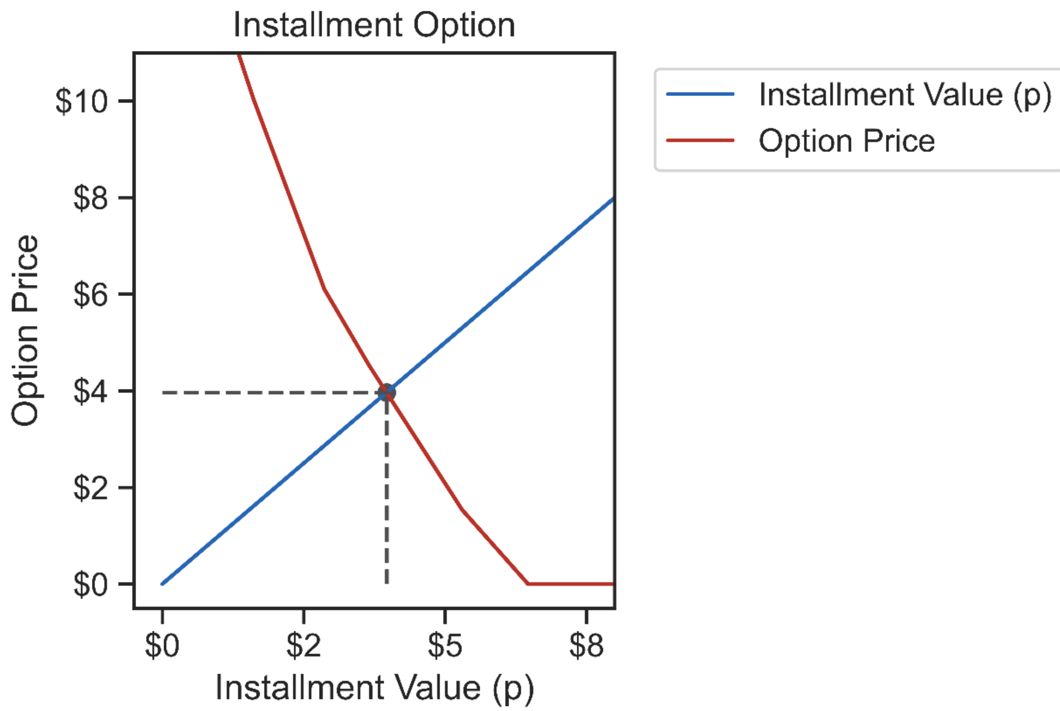


Figure 5.

S = $100, σ = 0.20, r = 0.04, K = $90, T = 1, and N = 5

Using the same inputs defined in Figure 5, the European call price = $ 15.9 using Binomial Tree.

Additionally, we can compare this price to a DCF method defined below. In conclusion, the DCF method is higher than the binomial tree price!

$$DCF = p\sum_{i=0}^{n-1} e^{-ri\frac{T}{n}}$$

$$\text{Discounted Cash Flow} = \$3.96 \; + \; \$3.96\, e^{-.04\frac{1}{5}} + \; \$3.96 \; * \; e^{-.04*2*\frac{1}{5}} + \$3.96 * \; e^{-.04*3*\frac{1}{5}} + \$3.96 \; * \; e^{-.04*4*\frac{1}{5}}$$

Discounted Cash Flow = $19.49

Appendix 1 – Binomial Tree Algorithm Outline (Additive Tree)

Inputs: $S_0,\ K,\ r,\ \sigma,\ T,\ N$

Step 1: Solve for the variables below:

$$u = e^{\sigma\sqrt{\Delta t}} \quad , \qquad d = \frac{1}{u}$$

$$p = \frac{1}{2} + \frac{r - \frac{1}{2}\sigma^2}{2\sigma}\sqrt{\Delta t}$$

$$\nu = r - \frac{1}{2}\sigma^2$$

$$\Delta x_u = \sqrt{\sigma^2 \Delta t + \nu^2 \Delta t^2} \quad , \qquad \Delta x_d = -\Delta x_u$$

Step 2: Build the tree, by calculating future stock prices

$$S_{i,j} = S_0 * e^{(j*\Delta x_u + (i-j)*\Delta x_d)}$$

i = # of periods in the future

j = # of up moves

Step 3: Calculate the option value at terminal notes:

$$V_{K,N}^{A} = \left(K - S_{i,j}\right)^{+} \text{ (Put)}$$

$$V_{K,N}^{A} = \left(S_{i,j} - K\right)^{+} \text{ (Call)}$$

Step 4: Work backwards in the tree. Beginning at one step behind the terminal nodes:

$$V_{K,N}^{K} = max\left\{\left(K - S_{i,j}\right)^{+},\ e^{-r\Delta t}\left[p\, V_{K+1,N+1}^{A} + (1-p)V_{K,N+1}^{A}\right]\right\} \text{ (American Put)}$$

European Option formula:

$$V_{K,N}^{K} = \left\{e^{-r\Delta t}\left[p\, V_{K+1,N+1}^{A} + (1-p)V_{K,N+1}^{A}\right]\right\}$$

Appendix 1 – Binomial Tree Python Code part 1(Additive Tree)

```
def BIN_TREE(S, K , T , N , r , sigma , c_or_p = 'CALL', a_or_e = 'AMERICAN'):
    delta_t = T / N

    time = []
    for i in range(0,N+1):
        time.append(delta_t*i)

    u = math.exp(sigma * np.sqrt(delta_t))
    d = 1/u

    p = .5 + ((r-.5*sigma*sigma)/(2*sigma))*math.sqrt(delta_t)
    v = r - (.5 *(sigma*sigma))

    delta_xu = math.sqrt(sigma*sigma*delta_t + ((v*v) * (delta_t*delta_t)))
    delta_xd = -1 * delta_xu

    # Start with empty data frame
    df = pd.DataFrame(np.zeros([N + 1, N + 2])*np.nan)
    df.iloc[:,0] = time

    for i in range(1,(N + 2)):
        for j in range(0,(N-i+2)): # THIS IS KEY !!!
            if j==0 and i==1:
                df.iloc[j,i] = S
            else:
                df.iloc[j,i] = S * math.exp((i-1)*delta_xu + j*delta_xd)
    n_row, n_col = df.shape
    v_df = df.copy()
         if c_or_p == 'CALL':
             for col in range(1, (n_col)):
                 row = N - col + 1
                 S_i = v_df[col].iloc[row]
                 v_df[col].iloc[row] = max([(S_i - K),0])
         elif c_or_p == 'PUT':
             for col in range(1, (n_col)):
                 row = N - col + 1
                 S_i = v_df[col].iloc[row]
                 v_df[col].iloc[row] = max([(K - S_i),0])
```

Appendix 1 – Binomial Tree Python Code part 2 (Additive Tree)

```
for t in reversed(range(1, n_col - 1)):
    for col in range(1, t + 1):
        row = t - col
        S_i = v_df[col].iloc[row]

        v_UP_col = col + 1
        v_UP_row = row
        up_value = v_df[v_UP_col].iloc[v_UP_row]

        v_DOWN_col = col
        v_DOWN_row = row + 1
        down_value = v_df[v_DOWN_col].iloc[v_DOWN_row]

        if c_or_p == 'CALL':
            if a_or_e == 'AMERICAN':
                Intrinsic_value = S_i - K
                Intrinsic_value = max([Intrinsic_value,0])

                est_value = math.exp(-r * delta_t)*(p * up_value + (1-p) * down_value)
                V = max([Intrinsic_value, est_value])

            elif a_or_e == 'EUROPEAN':
                est_value = math.exp(-r * delta_t)*(p * up_value + (1-p) * down_value)
                V = est_value

        elif c_or_p == 'PUT':
            if a_or_e == 'AMERICAN':
                Intrinsic_value = K - S_i
                Intrinsic_value = max([Intrinsic_value,0])

                est_value = math.exp(-r * delta_t)*(p * up_value + (1-p) * down_value)
                V = max([Intrinsic_value, est_value])
            elif a_or_e == 'EUROPEAN':
                est_value = math.exp(-r * delta_t)*(p * up_value + (1-p) * down_value)
                V = est_value

        v_df[col].iloc[row] = V

return(df, v_df)
```

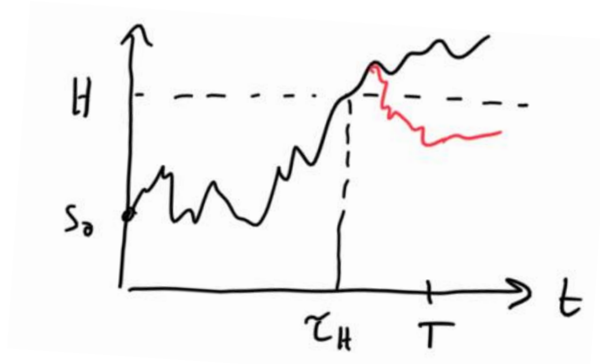
Appendix 1 - Binomial Tree Plot

```
def bin_plot(df, v_df, a_or_e, c_or_p, path = 'plots/name.png'):
    final_option_price = round(v_df[1].iloc[0], 2)
    n_row, n_col = df.shape

    min_time = df[0].iloc[0]
    max_time = df[0].iloc[(n_row-1)]

    max_price = max(df[(n_col-1)])
    min_price = min(df[1])

    sns.set(font_scale = 1.1)
    sns.set_style("ticks")
    fig, ax = plt.subplots(figsize=(4,4))
    plt.xlim((min_time-(.01*max_time)), max_time*1.05)
    plt.ylim(min_price*.95 , max_price*1.05)

    for t in range(1, n_col-1):
        for col in range(1, t + 1):
            time_i = df[0].iloc[(t-1)]
            row = t - col

            PN_Y = df[col].iloc[row]
            PN_X = time_i

            UN_Y = df[(col+1)].iloc[row]
            UN_X = df[0].iloc[t]

            DN_Y = df[col].iloc[(row+1)]
            DN_X = UN_X

            x = [UN_X, PN_X, DN_X]
            y = [UN_Y, PN_Y, DN_Y]
            plt.plot(x, y, 'bo-')

    plt.xlabel("Time (Years)")
    plt.ylabel("Stock Price ($)")
    main_title = str(a_or_e.lower()) + " " + str(c_or_p.lower())
    plt.title(main_title)
    plt.tight_layout()
    plt.savefig(path, dpi = 300)
    plt.show()
```

Appendix 2 – Up-and-Out Binomial Tree Outline (Multiplicative Tree)

"Up and Out": If the stock price at time t moves above H, then the option price at t becomes $0.

Inputs: $S_0,\ K,\ r,\ \sigma,\ T,\ N,\ H$



The graph above is from lecture notes

Step 1: Solve for the variables below:

$$u = e^{\sigma\sqrt{\Delta t}} \quad , \qquad d = \frac{1}{u}$$

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

Step 2: Build the tree, by calculating future stock prices

$$S_{i,j} = S_0 * u^i * d^j$$

i = # of up moves

j = # of down moves

Step 3: Calculate the option value at terminal notes

If $S_{n,k} < H$ then

$$V_{N,K}^{A} = (K - S_{N,K})^+ \ \text{(Put)}$$

$$V_{N,K}^{A} = (S_{N,K} - K)^+ \ \text{(Call)}$$

Else: $V_{N,K}^{A} = \$0$

Step 4: Work backwards in the tree. Beginning at one step behind the terminal nodes:

If $S_{n,k} < H$ then,

$$V_{n,k}^{A} = max\left\{(K - S_{N,K})^+ \ , \ e^{-r\Delta t}\left[q\, V_{n+1,k+1}^{A} + (1-q)V_{n+1,k}^{A}\right]\right\} \ \text{(American Put)}$$

Else: $V_{N,K}^{A} = \$0$

$$V_{n,k}^{K} = \left\{e^{-r\Delta t}\left[q\, V_{n+1,k+1}^{A} + (1-q)V_{n+1,k}^{A}\right]\right\} \ \text{(European Option formula)}$$

Appendix 2 – Up-and-Out Binomial Tree Outline (Multiplicative Tree)

Source: (https:// www.math.kth.se/matstat/seminarier/reports/K-exjobb09/090601a.pdf)

Closed-Form Solution

$$UO_{BS} = C_{BS}(S,K) - C_{BS}(S,H) - (H-K)e^{-rT}\Phi\big(d_{BS}(\mathrm{S},\mathrm{H})\big)$$

$$- \left(\frac{H}{S}\right)^{\frac{2\nu}{\sigma^2}} \left\{ C_{BS}\left(\frac{H^2}{S}, K\right) - C_{BS}\left(\frac{H^2}{S}, H\right) - (H-K)e^{-rT}\Phi\big(d_{BS}(\mathrm{H},\mathrm{S})\big) \right\}$$

Where:

$$\nu = r - \delta - \frac{\sigma^2}{2}$$

$$d_{BS}(S,K) = \frac{\ln\left(\frac{S}{K}\right) + \nu T}{\sigma\sqrt{T}}$$

$$\Phi(\mathrm{x}) \sim Normal\ CDF(0,1)$$

Appendix 3 – Installment Option Outline (Multiplicative)

Inputs: $S_0, \ K, \ r, \ \sigma, \ T, \ N, \ p$

p = 'installment value'

Example: T = [0, . 25, .5, .75, 1]

Step 1: Solve for the variables below:

$$u = e^{\sigma\sqrt{\Delta t}} \quad , \qquad d = \frac{1}{u}$$

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

Step 2: Build the tree, by calculating future stock prices

$$S_{i,j} = S_0 * u^i * d^j$$

i = # of up moves

j = # of down moves

Step 3: Calculate the option value at terminal notes

T = [0, . 25, .5, .75, 1]

$$V_{N,K}^{A} = \left(K - S_{N,K}\right)^{+} \text{ (Put)}$$

$$V_{N,K}^{A} = \left(S_{N,K} - K\right)^{+} \text{ (Call)}$$

Step 4: Work backwards in the tree. Beginning at one step behind the terminal nodes:

T = [0, . 25, .5, .75, 1]

$$V_{n,k}^{K} = \left\{e^{-r\Delta t}\left[q\, V_{n+1,k+1}^{A} + (1-q)V_{n+1,k}^{A}\right]\right\}$$

Step 5: Work backwards in the tree. Beginning at two steps behind the terminal nodes.

T = [0, . 25, .5, .75, 1]

$$V_{n,k}^{K} = \left\{e^{-r\Delta t}\left[q\left(V_{n+1,k+1}^{A} - p\right)^{+} + (1-q)\left(V_{n+1,k+1}^{A} - p\right)^{+}\right]\right\}$$