Group Work Project - M5

June 15, 2021

- **Course**: MScFE 620 Discrete-time Stochastic Processes
- **Professors**:
  - Sergio Garcia
  - Ivan Blanko
- **Class**: C21-S1
- Group 2, with members:

  - Juan Antonio Vargas Lopez; juanvarl@hotmail.com
  - Kanittha Setthapitayakul; kanittha.se@hotmail.com
  - Loc Nguyen; vinhloc30796

In [1]:
```
import numpy as np
np.set_printoptions(precision=4, suppress=True) # prettify numpy
import pandas as pd
pd.set_option('precision', 4) # prettify pandas
from binomial import binomial_tree, binomial_freq, terminal_coef
```

1. Write code to price a European Call option:

a. The underlying stock that is currently trading at US95. The option has a strike price of US105 and 1 year maturity. Use the Binomial model with the parameters r=0 and 3 steps in the pricing process.

Additionally, set the upper movement of the price (u) to an expression that will depend on the number on your group.

$$u = (1.10 + \frac{GroupNumber}{100}),$$

Group 1, u = 1.11. Group 2 = 1.12. Group 3 = 1.13, ..., Group 50 = 1.60

You can define $d = \frac{1}{u}$.

In [2]:
```
# Set starting price at $95
S0 = 95
# Set strike price at $105
strike_price = 105
# Calculate u, d, P_star
group_number = 2
u = 1.10 + group_number/100; d = 1/u
p = (1 - d) / (u - d)
p
```

Out[2]: `0.4716981132075471`

In [3]:
```
stock_prices_3steps = binomial_tree(
    N=3, # 3 steps
    S0=S0, # underlying stock starts at $95,
    u=u,
)['stock_prices']
df_stock_prices_3steps = pd.DataFrame(
    stock_prices_3steps
)
df_stock_prices_3steps.index.name = "Time Step"
df_stock_prices_3steps
```

Out[3]:

Time Step	0	1	2	3
0	95.0000	0.0000	0.000	0.0000
1	84.8214	106.4000	0.000	0.0000
2	75.7334	95.0000	119.168	0.0000
3	67.6191	84.8214	106.400	133.4682

In [4]:
```
terminal_coef_3steps = terminal_coef(
    N=3,
    u=u
)
terminal_coef_3steps
```

Out[4]: `array([0.1475, 0.395 , 0.3526, 0.105 ])`

In [5]:
```
option_payoff_3steps = np.where(
    stock_prices_3steps > strike_price, # if underlying stock is priced higher than s
    stock_prices_3steps - strike_price, # then value = difference (buy low)
    0 # else 0 (leaving option unexercised)
)
df_option_payoff_3steps = pd.DataFrame(
    option_payoff_3steps
)
df_option_payoff_3steps.index.name = "Time Step"
df_option_payoff_3steps
```

Out[5]:

Time Step	0	1	2	3
0	0.0	0.0	0.000	0.0000
1	0.0	1.4	0.000	0.0000
2	0.0	0.0	14.168	0.0000
3	0.0	0.0	1.400	28.4682

In [6]:
```
# Value of the option at T0 = Sigma(probability for each path * value for each path)
sum(
    terminal_coef_3steps \
    * option_payoff_3steps[-1,:]
)
```

Out[6]: `3.4814981494791057`

ANSWER:

The value of the European call option, as described by the prompt, is therefore, ≈ 3.4815.

b. Using the information from (a), show the value of the derivative, $H(\omega)$, for each of the paths.

In [7]:
```
df_option_payoff_3steps.iloc[3,:]
```

Out[7]:
```
0     0.0000
1     0.0000
2     1.4000
3    28.4682
Name: 3, dtype: float64
```

ANSWER:

The ending payoff $H(\omega)$ of the call option for each of the path is therefore:

- 0.00 with 0 ups
- 0.00 with 1 ups
- 1.40 with 2 ups
- ≈ 28.47 with 3 ups

2. Write code to price a European Put Option:

a. Consider the same parameters as in section (a) in part (1) above but now with N=2. What is the price of the option?

In [8]:
```
stock_prices_2steps = binomial_tree(
    N=2, # 3 steps
    S0=S0, # underlying stock starts at $95,
    u=u,
)['stock_prices']
df_stock_prices_2steps = pd.DataFrame(
    stock_prices_2steps
)
df_stock_prices_2steps.index.name = "Time Step"
df_stock_prices_2steps
```

Out[8]:

Time Step	0	1	2
0	95.0000	0.0	0.000
1	84.8214	106.4	0.000
2	75.7334	95.0	119.168

In [9]:
```
terminal_coef_2steps = terminal_coef(
    N=2,
    u=u,
)
terminal_coef_2steps
```

Out[9]: `array([0.2791, 0.4984, 0.2225])`

In [10]:
```
option_payoff_2steps = np.where(
    np.logical_and(
        stock_prices_2steps < strike_price, # if underlying stock is priced lower thai
        stock_prices_2steps != 0 # and underlying stock is not 0.00 (technical limita
    ),
    strike_price - stock_prices_2steps, # then value = difference (sell high)
    0 # else 0 (leaving option unexercised)
)
df_option_payoff_2steps = pd.DataFrame(
    option_payoff_2steps
)
df_option_payoff_2steps.index.name = "Time Step"
df_option_payoff_2steps
```

Out[10]:

Time Step	0	1	2
0	10.0000	0.0	0.0
1	20.1786	0.0	0.0
2	29.2666	10.0	0.0

In [11]:
```
# Value of the option at T0 = Sigma(probability for each path * value for each path)
sum(
    terminal_coef_2steps \
    * option_payoff_2steps[-1,:]
)
```

Out[11]: `13.152367390530443`

In [12]:
```
option_payoff_2steps[2,:]
```

Out[12]: `array([29.2666, 10.    ,  0.    ])`

ANSWER:

The value of the European call option, as described by the prompt, is therefore, ≈ 12.3166.

The ending payoff $H(\omega)$ for each path is:

- ≈ 29.27 with 0 ups (i.e. 2 downs)
- 10.00 with 1 up (i.e. 1 downs)
- 0.00 with 2 ups (i.e. 0 downs)

b. Construct a Table (alike the ones you have in the notes) that includes, for each price path and each t when it corresponds:

- the information on stock price evolution, $X_t(\omega)$,
- the value of the option, $V_t^H(\omega)$,
- the payoff of the option, $H(\omega)$,
- and the hedging strategy, φ_t^H.

ANSWER

The stock price evolution and corresponding option payoff can be seen below:

ω	$X_0(\omega)$	$X_1(\omega)$	$X_2(\omega)$	$H(\omega)$
(u,u)	95.0	106.4	119.168	0
(u,d)	95.0	106.4	95.0	10.0
(d,u)	95.0	84.8214	95.0	10.0
(d,d)	95.0	84.8214	75.7334	29.2667

We also have $P^* = \frac{1-d}{u-d} = \frac{1-\frac{1}{1.12}}{1.12-\frac{1}{1.12}} \approx 0.4717$ and $1 - P^* \approx 0.5283$

Using that, we can calculate:

- $V_1^H(\{u,.\}) = V_1^H(\{u,u\}) = V_1^H(\{u,d\}) \approx 0.4717 \cdot 0 + 0.5283 \cdot 10 \approx 5.283$
- $V_1^H(\{d,.\}) = V_1^H(\{d,u\}) = V_1^H(\{d,d\}) \approx 0.4717 \cdot 10 + 0.5283 \cdot 29.2666 \approx 20.1794$
- $V_0^H \approx 0.4717^2 \cdot 0 + 2 \cdot 0.4717 \cdot 0.5283 \cdot 10 + 0.5283^2 \cdot 29.2666 \approx 13.1523$

which is then tabulated as follows:

ω	V_0^H	V_1^H	V_2^H
$(u;u)$	13.1523	5.283	0
$(u;d)$	13.1523	5.283	10
$(d;u)$	13.1523	20.1794	10
$(d;d)$	13.1523	20.1794	29.2667

The hedging strategy is calculated by:

$$\varphi_t^H = \frac{(V_t^H - V_{t-1}^H)}{(X_t^H - X_{t-1}^H)}$$

We have the hedging strategy tabulated as follows:

ω	φ_1^H	φ_2^H
$(u;u)$	-0.6903	-0.4138
$(u;d)$	-0.6903	-0.4138
$(d;u)$	-0.6903	-1.0000
$(d;d)$	-0.6903	-1.0000

3. Market Completeness Revisited

a. Form a matrix with 2 rows and 2 columns. Each row contains a state of the world (up and down). Each column contains a security (stock and bond).

ANSWER:

Matrix A

State of the world	Stock	Bond
up	S_u	B_u
down	S_d	B_d

b. Pick a specific node. Write down the values for the A matrix

ANSWER:

In [13]:
```
# Create base variables
S_u = S0*u
S_d = S0*d
X_su = S_u-S0
X_sd = S_d-S0
B = 100 # Bonds

# Create np arrays
s_arr = np.array([S_u,S_d])
b_arr = B*np.ones(2)
A = np.concatenate((s_arr.reshape(2,1),b_arr.reshape(2,1)),axis=1)

# Show
A
```

Out[13]:
```
array([[106.4   , 100.    ],
       [ 84.8214, 100.    ]])
```

c. From that node, the stock price may go up or down. Write down the b matrix (which is a column matrix). The first value in b contains the option value if the stock price went up. The second value in b contains the option value if the stock price went down.

ANSWER:

Matrix B

	Option Value
up	V_u
down	V_d

In [14]:
```
option_type = 'Call'
if option_type =='Call':
    V_u = max(0, S_u - strike_price) # Long Call
    V_d = max(0, S_d - strike_price) # Long Call
elif option_type =='Put':
    V_u = max(0, strike_price - S_u) # Long Put
    V_d = max(0, strike_price - S_d) # Long Put
B = np.array([V_u,V_d])
```

d. The no-arbitrage equation can be written as Ax=b. How do you solve this equation for x?

ANSWER:

Assuming that A is invertible:

$$Ax = B \Rightarrow x = A^{-1}B$$

e. Using matrix algebra, solve for x

ANSWER:

In [15]:
```
try:
    A_inv = np.linalg.inv(A)
    x_arr = np.dot(A_inv,B)
except:
    x_arr = np.linalg.lstsq(A,B)

# Show
x_arr
```

Out[15]: `array([ 0.0649, -0.055 ])`

In [16]:
```
m_phi = int(x_arr[0]*1000000)/1000000
print(f"Hedging portfolio of focus stock equal to {m_phi}")
```

```
Hedging portfolio of focus stock equal to 0.064879
```

f. Show that your solution matches that from the binomial tree

ANSWER:

Consider probability p where the asset price goes up

$$p = \frac{1-d}{u-d}$$

In [17]:
```
p = (1-d)/(u-d)
print(p)
```

```
0.4716981132075471
```

Assume market is complete and the unique no-arbitrage price of H, we obtains the option value at $t = 0$ by:

$$\pi(H) = \sum_{\omega \in \Omega} H(\omega) \prod_{t=1}^{T} p^{\omega_t}(1-p)^{1-\omega_t}$$

In [18]:
```
V_0 = p*V_u+(1-p)*V_d
print(V_0)
```

```
0.6603773584905687
```

In [20]:
```
phi_u = (V_u-V_0)/(S_u-S0)
phi_d = (V_d-V_0)/(S_d-S0)
process_phi = [phi_u,phi_d]
print(process_phi)
```

```
[0.06487917907977514, 0.06487917907977515]
```

In [21]:
```
if int(process_phi[0]*1000000)/1000000==int(process_phi[1]*1000000)/1000000:
    bi_phi = int(process_phi[0]*1000000)/1000000
    print("Hedging portfolio of focus stock from the binomial tree equal to "+ str(int
else: print("Check pricing method")
```

```
Hedging portfolio of focus stock from the binomial tree equal to 0.064879
```

In [22]:
```
if m_phi == bi_phi:
    print("x from matrix algebra matches to hedging porfolio from binomial tree")
else:
    print("Check pricing method")
```

```
x from matrix algebra matches to hedging porfolio from binomial tree
```

4. Put Call Parity

a. When a stock option expires...

...there are 3 states of the world. 1) Final stock price > Strike 2) Final Stock price = Strike 3) Final Stock price < Strike. Consider a call and put with the same underlying, strike, European exercise style, and expiration. Assume a constant risk-free rate. For each state of the world, show the value of the following portfolios:

- i) Long 1 call and short 1 put
- ii) Long 1 stock and borrow K (strike) dollars at the risk free rate for T (option maturity) years

ANSWER:

The assumptions for this question are the following:

$S_0 = 100$, $u = 1.2$, $d = \frac{1}{u}$, $p = \frac{1-d}{u-d}$, $r = 2\%$, and $T = 5$

To compute the Put-Call parity, the following equation must hold:

$$C + PV(K) = P + S \tag{1}$$

On the left hand of the equation, C is the price of a Call option, $PV(K)$ is the present value of K the borrowed money at the risk free rate r for time T. On the right hand side of the equation, P is the price of Put option, and S the market value of the underlying asset at time T. We can rewrite the equation so that,

$$C - P = S - PV(K) \tag{2}$$

1) Final stock price > Strike: $S_T = 160$, $K = 110$

- i) The prices of the long call (C^+) and short put (P^-):

$$C^+ = 11.5928$$
$$P^- = 20.6501$$

  The portfolio value P_1 is:

$$P_1 = max(S_T - K, 0) - max(K - S_T, 0) = 50 - 0 = 50$$

- ii) The present value of the borrowed money and the price of the stock looks as follows:

$$PV(K) = \frac{K}{(1+r)^T} = \frac{110}{(1+2\%)^5} = 99.6304$$
$$S_T = S_5 = 160$$

  The portfolio value P_2 is:

$$P_2 = S_T - PV(K) = 160 - 99.6304 = 60.3696$$

2) Final Stock price = Strike: $S_T = 110$, $K = 110$

- i) The prices of the long call (C^+) and short put (P^-):

$$C^+ = 11.5928$$
$$P^- = 20.6501$$

  The portfolio value P_1 is:

$$P_1 = max(S_T - K, 0) - max(K - S_T, 0) = 0 - 0 = 0$$

- ii) The present value of the borrowed money and the price of the stock looks as follows:

$$PV(K) = \frac{K}{(1+r)^T} = \frac{110}{(1+2\%)^5} = 99.6304$$
$$S_T = S_5 = 110$$

  The portfolio value P_2 is:

$$P_2 = S_T - PV(K) = 110 - 99.6304 = 10.3696$$

3) Final Stock price < Strike: $S_T = 110$, $K = 160$

- i) The prices of the long call (C^+) and short put (P^-):

$$C^+ = 2.9110$$
$$P^- = 57.2548$$

  The portfolio value P_1 is:

$$P_1 = max(S_T - K, 0) - max(K - S_T, 0) = 0 - 50 = -50$$

- ii) The present value of the borrowed money and the price of the stock looks as follows:

$$PV(K) = \frac{K}{(1+r)^T} = \frac{160}{(1+2\%)^5} = 144.9169$$
$$S_T = S_5 = 110$$

  The portfolio value P_2 is:

$$P_2 = S_T - PV(K) = 110 - 144.9169 = -34.9169$$

b. Using your answers from the previous parts...

...check that the call price, the put price, the stock price, and the borrowed funds satisfy put-call parity. If not, explain why it may not match more precisely. (Hint: it may not match even if you did everything correctly!).

ANSWER:

From the three different scenarios, we saw that the Put-Call parity did not hold:

- 1) Final stock price > Strike: $S_T = 160$, $K = 110$

$$C - P = S - PV(K)$$
$$50 \neq 60.3696$$

- 2) Final Stock price = Strike: $S_T = 110$, $K = 110$

$$C - P = S - PV(K)$$
$$0 \neq 10.3696$$

- 3) Final Stock price < Strike: $S_T = 110$, $K = 160$

$$C - P = S - PV(K)$$
$$-50 \neq -34.9169$$

An arbitrator might purchase and trade a more costly portfolio for the less expensive. Since the portfolios are guaranteed to be mutually cancelled at time T, this technique would lock an arbitration profit that is equal to the difference in the two portfolios' values (Hull, 2018). In other words, if there exists an arbitrage opportunity, we should always sell the more expensive portfolio and buy the cheaper portfolio. Moreover, many investors who engage in the same trades notice arbitration chances like this. Prices will adjust to restore parity.

The Put-Call parity will only hold if and only if the options are European style, both call and put options must have the same strike price, the stock does not pay dividends, interest rates must remain unchanged until the expiration date, and there are no exchange or brokerage fees. For the previous examples, we are forcing S_T to have an especific value. Moreover, we are also assuming that the intial price $S_0 = 100$, which also affects the valuation of the both call and put options. As an example, if we consider the following scenario we found that by solving for $S_T = 90.5731$ with $K = 110$, the Put-Call parity holds. We can do the same exercise with the rest of the variables to find the implicit value that will make the parity to hold.

References

Hull, John C. (2018).*Options, Futures, and Other Derivatives, Global Edition*. Pearson.