Stochastic Processes for Finance: Assignment 1

Group 06: Alexander Robert Algmin Venegas (2871245) and Andreu Boix Torres (2868333)

Exercise 1

Consider the N-period binomial model with $r = log(2/3)$, $S_0 = 6$, $u = 5/6$, $d = 1/6$, and a forward whose pay-off is $C(S_0, \ldots, S_N) = S_N - 2$. Say, $p = 0.4$. Using the tools from Chapter 2 of the lecture notes, do the following:

- Check if the model is arbitrage free;
- For maturity $N = 2$, compute the corresponding replicating portfolio for the contract described above;
- Determine the contract price V_0, and also V_1;
- The replicating portfolio has a very distinctive characteristic; what is it? make a connection to how we priced a forward in the first chapter;
- Suppose someone is willing to buy the contract with maturity 2 for a price $P > V_0$. Construct an explicit arbitrage strategy.

Note: justify your answers by showing all intermediate computations in your solutions, otherwise you won't get any points for this exercise.

- Check if the model is arbitrage free;

Let us first suppose that the model is free of arbitrage. Then, we could construct an arbitrage strategy.

At first, we start with no money: $0 = V_0 = \phi_1 S_0 + \psi_1$. Then, $\phi_1 S_0 = -\psi_1$.

As the portfolio in the N-period binomial model is self-financing, we have that

$V_0 = \phi_1 S_0 + \psi_1 = \phi_2 S_0 + \psi_2$.

For that reason, it is also true that $\phi_2 S_0 = -\psi_2$.

Hence,

$V_1 = \phi_2 S_1 + \psi_2 e^r$ and, substituting for $\phi_2 S_0 = -\psi_2$ we get
$V_1 = \begin{cases} \psi_2(e^r - u) & wp \quad p \\ \psi_2(e^r - d) & wp \quad 1-p \end{cases}$

There is always a non-negative probability of earning money without risk if $e^r \leq d$ or $e^r \geq u$. Let us observe this further:

If $e^r > u$ then:
$V_1 = \begin{cases} \psi_2(e^r - u) > 0 & wp \quad p \\ \psi_2(e^r - d) > 0 & wp \quad 1-p \end{cases}$
So we have that $P(V_1 \geq 0) = 1, P(V_0 = 0)$ and $P(V_1 > 0) > 0$. There is arbitrage.

If $e^r < d >$ then:
$V_1 = \begin{cases} \psi_2(e^r - u) < 0 & wp \quad p \\ \psi_2(e^r - d) < 0 & wp \quad 1-p \end{cases}$
So we have that $P(V_1 \leq 0) = 1, P(V_0 = 0)$ and $P(V_1 < 0) > 0$. There is arbitrage (for the seller).

Arbitrage strategy:

Let's see the arbitrage strategies if $e^r \geq u$ (Case 1) or $e^r \leq d$ (Case 2)

Case 1 ($e^r \geq u (> d)$)

1. We have $V_0 = 0$
2. At $t = 0$ we borrow an asset and sell it for S_0.
3. Invest S_0 at rate $r = log(2/3)$
4. At $t = 1$ we pay back the asset. We get $V_1 = S_0 e^r - S_0 u \geq 0$ if the asset goes up, else $V_1 = S_0 e^r - S_0 d \geq 0$

Case 2 ($e^r \leq d (< u)$)

1. We have $V_0 = 0$
2. At $t = 0$ we borrow S_0 and buy an asset.
3. At $t = 1$ sell the asset and pay back the debt. We get $V_1 = S_0 d - S_0 e^r \geq 0$ if the asset goes down, else $V_1 = S_0 u - S_0 e^r \geq 0$

Note that if $e^r \geq u (< d)$ then (without any money) we can borrow an asset and sell it for S_0, then invest the money. At $t = 1$ we have to pay back the asset, so at least we would get $S_0 e^r - S_0 u \geq 0$.

If $e^r \leq d (< u)$ then (without any money) we can borrow S_0 and buy an asset. At $t = 1$ we have to pay back the money, so at least we would get $S_0 d - S_0 e^r \geq 0$.

As we have just seen, there is arbitrage if and only if $e^r \geq u$ or $e^r \leq d$.

In this case, we have that $e^r = \frac{2}{3} \in (d, u) = (\frac{1}{6}, \frac{5}{6})$, and hence, there is no arbitrage.

- For maturity $N = 2$, compute the corresponding replicating portfolio for the contract described above;

Let us recall the binomial model and our contract described. We are given a binomial model with:

- $r = \log(2/3)$

- $s_0 = 6$
- $u = 5/6$
- $d = 1/6$
- Probability $p = 0.4$
- Payoff of a forward contract $C(s_0, \ldots, S_N) = C(S_N) = S_N - 2$

Notice that it is not an option since we do not have the right of exercising or not the contract (there is not a maximum nor a minimum). So, if we buy the contract, the payoff is always $S_N - 2$. Here, 2 can be seen as the agreed price of the asset of the forward, the strike price K.

Given a portfolio (ϕ_n, ψ_n), where ϕ_n represents the number of assets at time n and ψ_n represents the amount of money invested in the risk-free asset at time n, we say that the portfolio is replicating for a contract if it is constructed so that its value matches the payoff of the contract at all times.

Given a portfolio, (ϕ_n, ψ_n), we can calculate the portfolio value at time n as: $V_n = \phi_n S_n + \psi_n e^{nr}$, with $n \geq 0$ and S_n the value of the asset at time n.

Let us calculate the portfolio. At $t = 1$, the portfolio value is denoted as V_1. The asset can move up to $S_1 = s_0 u$ or down to $S_1 = S_0 d$. Therefore, we have the following system of equations for the portfolio value. Notice that here ($t = n = 1$) we are calculating $(\phi_n, \psi_n) = (\phi_1, \psi_1)$.

$$\begin{cases} \phi_1 u s_0 + \psi_1 e^r &= V_1(s_0, s_0 u) \\ \phi_1 d s_0 + \psi_1 e^r &= V_1(s_0, s_0 d) \end{cases}$$

By solving the system (subtracting the first one and the second one for ϕ_1 and multiplying by d the first equation and by u the second one and subtracting the first one and the second one for ψ_1), we get the formulas for ϕ_1 and ψ_1:

$$\phi_1 = \frac{V_1(s_0,s_0u) - V_1(s_0,s_0d)}{s_0(u-d)}$$

$$\psi_1 = \frac{uV_1(s_0d) - dV_1(s_0,s_0u)}{e^r(u-d)}$$

The initial value V_0 of the portfolio at $t = 0$ is given by the risk-neutral pricing formula:

$$V_0 = e^{-r}(qV_1(s_0, s_0u) + (1-q)V_1(s_0, s_0d))$$

where q is the risk-neutral probability, calculated as:

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

At time $t = 2$, the asset price S_2 can take different values, and we need to calculate the payoffs based on these potential outcomes. We are essentially back to a one-period binomial model for each case.

At time $t = 2$, S_2 can be either $s_1 u$ or $s_1 d$. This leads to two possible payoffs: $C_2(s_1 u)$ or $C(s_1 d)$. We are back at the one-period binomial model, then, using the risk-neutral

pricing formula:

$$V_1 = V_1(S_1) = \begin{cases} e^{-r}\left(qC(s_0u^2) - (1-q)C(s_0du)\right), & \text{if} \quad S_1 = s_0u, \\ e^{-r}\left(qC(s_0ud) + (1-q)C(s_0d^2)\right), & \text{if} \quad S_1 = s_0d \end{cases}$$

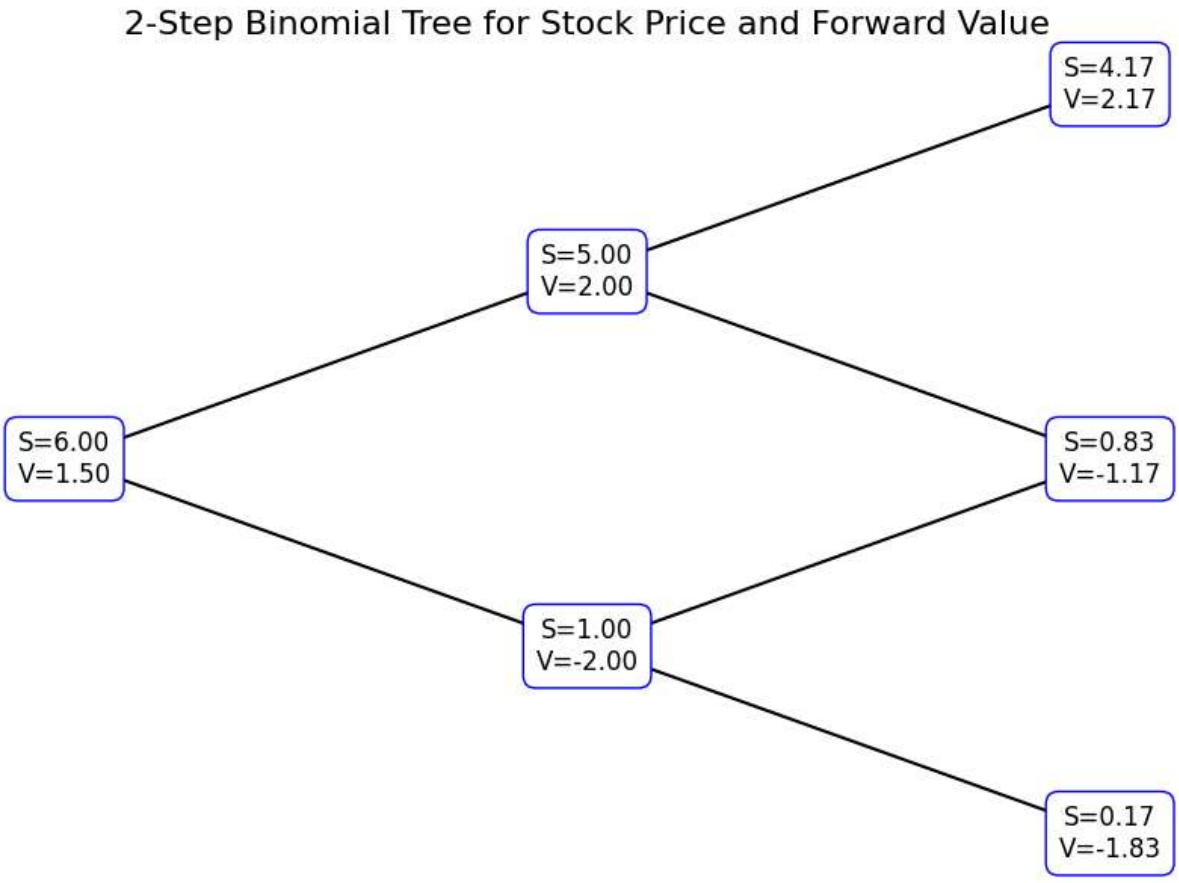
Where we have to notice that $V_2(s_0, S_1, S_2)$ is already at time $T = 2$, which is the maturity for the contract, and hence the value of the portfolio is the payoff $V_2(s_0, S_1, S_2) = C(S_2) = S_2 - 2$.

Let us first paint the tree in Python, in order to have a mental map of the prices for the stocks and of the values of the portfolio.

In [2]:

```
import numpy as np
import matplotlib.pyplot as plt

# Parameters for the binomial model
s0 = 6    # Initial stock price
K = 2     # Strike price
u = 5/6     # Up factor
d = 1/6     # Down factor
r = np.log(2/3)    # Risk-free rate
T = 1       # Time to maturity
n_steps = 2 # Number of steps

# Risk-neutral probability
q= (np.exp(r) - d) / (u - d)

# Compute stock prices at each node
S = np.zeros((n_steps + 1, n_steps + 1))  # Stock price tree
S[0, 0] = s0
for i in range(1, n_steps + 1):
    S[i, 0] = S[i - 1, 0] * u  # Up move
    for j in range(1, i + 1):
        S[i, j] = S[i - 1, j - 1] * d  # Down move

V = np.zeros((n_steps + 1, n_steps + 1))  # Contract value tree

# Calculate payoff at final nodes (step n_steps)
for j in range(n_steps + 1):
    V[n_steps, j] = S[n_steps, j] - K  # Payoff for a forward contract

# Backtrack to get the contract values at earlier nodes
for i in range(n_steps - 1, -1, -1):
    for j in range(i + 1):
        V[i, j] = np.exp(-r) * (q * V[i + 1, j] + (1 - q) * V[i + 1, j + 1])

# Plot the binomial tree
fig, ax = plt.subplots(figsize=(8, 6))
ax.set_title('2-Step Binomial Tree for Stock Price and Forward Value', fontsize=

# Plot the nodes and annotate with stock prices and contract values
for i in range(n_steps + 1):
    for j in range(i + 1):
        x = i  # horizontal position (step number)
        y = i - 2 * j  # vertical position (arbitrarily chosen for visualization
        ax.plot(x, y, 'bo', markersize=10)  # Plot node
        # Annotate with S_n (stock price) and V_n (contract value)
        ax.text(x, y, f"S={S[i, j]:.2f}\nV={V[i, j]:.2f}", ha='center', va='cent
```

```
                fontsize=12, bbox=dict(facecolor='white', edgecolor='blue', boxs

# Draw lines between the nodes to represent possible transitions
for i in range(n_steps):
    for j in range(i + 1):
        x = i
        y = i - 2 * j
        # Line to up move
        ax.plot([x, x + 1], [y, y - 1], 'k-', lw=1.5)
        # Line to down move
        ax.plot([x, x + 1], [y, y + 1], 'k-', lw=1.5)

# Hide axes for better visualization
ax.axis('off')
plt.tight_layout()
plt.show()
```



Although in the code above we already calculated the values, let us show it via calculations by hand.

We first calculate the stock prices in a forward recursive manner. Then, we will calculate the contract values in a recursively backwards manner.

Calculation of S_0, S_1, S_2 in a forward recursive manner:

$S_0 = s_0 = 6.$

Then, the possible values for S_1 are s_0u and s_0d.

$$S_1 = \begin{cases} s_0 \cdot u = 6 \cdot \frac{5}{6} = 5 \\ s_0 \cdot d = 6 \cdot \frac{1}{6} = 1 \end{cases}$$

On the other hand, the possible values for S_2 are $S_1 u$ and $S_1 d$. Let us calculate it:

$$S_2 = \begin{cases} S_1 \cdot u = \begin{cases} s_0 \cdot u \cdot u = 6 \cdot \frac{5}{6} \cdot \frac{5}{6} = \frac{25}{6} \\ s_0 \cdot u \cdot d = 6 \cdot \frac{5}{6} \cdot \frac{1}{6} = \frac{5}{6} \end{cases} \\ S_1 \cdot d = \begin{cases} s_0 \cdot d \cdot u = 6 \cdot \frac{1}{6} \cdot \frac{5}{6} = \frac{5}{6} \\ s_0 \cdot d \cdot d = 6 \cdot \frac{1}{6} \cdot \frac{1}{6} = \frac{1}{6} \end{cases} \end{cases}$$

Hence, there are three possible values for S_2: $s_0 u^2$, $s_0 ud$ and $s_0 d^2$.

Calculation of V_0, V_1, V_2 in a backwards recursive manner:

We will calculate the contract values in a recursively backwards manner. Let us start by calculating the possible values of $V_2(s_0, S_1, S_2) = C(S_2)$:

$$V_2(s_0, S_1, S_2) = C(s_0, S_1, S_2) = C(S_2) = S_2 - 2 = \begin{cases} C(s_0 u^2) = s_0 u^2 - 2 = \frac{25}{6} - 2 = \\ C(s_0 ud) = s_0 ud - 2 = \frac{5}{6} - 2 = \\ C(s_0 d^2) = s_0 d^2 - 2 = \frac{1}{6} - 2 = \end{cases}$$

We know:

$$V_1 = V_1(S_1) = \begin{cases} e^{-r} \left(qC(s_0 u^2) + (1-q) C(s_0 du) \right), & \text{if} \quad S_1 = s_0 u, \\ e^{-r} \left(qC(s_0 ud) + (1-q) C(s_0 d^2) \right), & \text{if} \quad S_1 = s_0 d \end{cases}$$

Let us first calculate q.

$$q = \frac{e^r - d}{u - d} = \frac{e^{log(2/3) - \frac{1}{6}}}{\frac{5}{6} - \frac{1}{6}} = \frac{\frac{4-1}{6}}{\frac{5-1}{6}} = \frac{3}{4}.$$

Hence, substituting for e^{-r}, q and $V_2(s_0, S_1, S_2) = C_2(S_2)$:

$$V_1 = V_1(S_1) = \begin{cases} e^{-r} \left(qC(s_0 u^2) + (1-q) C(s_0 du) \right) = e^{-r} \left(\frac{3}{4} \frac{13}{6} + \frac{1}{4} (-\frac{7}{6}) \right) = \frac{3}{2} \left(\frac{13}{8} - \right. \\ e^{-r} \left(qC(s_0 ud) + (1-q) C(s_0 d^2) \right) = e^{-r} \left(\frac{3}{4} (-\frac{7}{6}) + \frac{1}{4} (-\frac{1}{6}) \right) = \frac{3}{2} \left(- \right. \end{cases}$$

Hence, we get that

$$V_1(S_1) = \begin{cases} 2 & if \quad S_1 = s_0 u, \\ -2 & if \quad S_1 = s_0 d \end{cases}$$

Now, let us calculate V_0, to get the replicating portfolio at time $t = 0$:

$$V_0 = V_0(S_0) = e^{-r} \left(qV_1(s_0 u) + (1-q) V_1(s_0 d) \right) = e^{-r} \left(\frac{3}{4} \cdot 2 + \frac{1}{4} \cdot -2 \right) = \frac{3}{2} \left(\frac{3}{2} - \frac{1}{2} \right) =$$

.

Calculation of $\phi_0, \psi_0, \phi_1, \psi_1, \phi_2^u, \phi_2^d, \psi_2^u, \psi_2^d$ in a forward recursive manner:

From $V_0 = \phi_0 S_0 + \psi_0$, if we try and compute ϕ_0 and ψ_0:

$$\frac{3}{2} = 6\phi_0 + \psi_0$$

Observe that we have one degre of freedom. However, we obtain the restriction that the initial portfolio at time $t = 0$ must be of the form $\psi_0 = \frac{3}{2} - 6\phi_0$.

Let us now calculate the following portfolios.

From $V_1 = \phi_1 S_1 + \psi_1 e^r$, let us compute ϕ_1 and ψ_1:

$$\phi_1 = \frac{V_1(s_0,s_0u) - V_1(s_0,s_0d)}{s_0(u-d)} = \frac{2-(-2)}{6(\frac{2}{3})} = 1$$

and

$$\psi_1 = \frac{uV_1(s_0,s_0d) - dV_1(s_0,s_0u)}{e^r(u-d)} = \frac{-2\frac{5}{6} - 2\frac{1}{6}}{e^{log(2/3)}\frac{2}{3}} = \frac{-9}{2}$$

For (ϕ_2, ψ_2) we proceed with (almost) the same calculations but in 2 different cases:
Case 1 ($S_2 = uS_1$):
Let us then call ϕ_2 in this case ϕ_2^u and analogously ψ_2^u:

$$\begin{cases} \phi_2^u u^2 s_0 + \psi_2^u e^{2r} = & C(s_0, s_0u, s_0u^2) = & u^2 s_0 - 2, \\ \phi_2^u uds_0 + \psi_2^u e^{2r} = & C(s_0, s_0u, s_0ud) = & uds_0 - 2 \end{cases}$$

We can now subtract the two equations to get an equation in terms of only ϕ_2^u, and:

$$\phi_2^u \left(u^2 s_0 - uds_0\right) = u^2 s_0 - 2 - uds_0 + 2 = \left(u^2 s_0 - uds_0\right)$$

And hence,

$$\phi_2^u = \frac{(u^2 s_0 - uds_0)}{(u^2 s_0 - uds_0)} = 1.$$

Substituting now ϕ_2^u in the first equation, for instance, we can compute ψ_2^u, and:

$$\psi_2^u e^{2r} = u^2 s_0 - 2 - \phi_2^u u^2 s_0 = u^2 s_0 - 2 - u^2 s_0 = -2$$

And hence, $\psi_2^u = -\frac{2}{e^{2r}}$. But $e^{2r} = \frac{4}{9}$, and we can compute:

$$\psi_2^u = -\frac{9}{2}.$$

Hence,

$$\phi_2^u = 1$$

$$\psi_2^u = -\frac{9}{2}$$

Case 2 ($S_2 = dS_1$):
Let us then call ϕ_2 in this case ϕ_2^d and analogously ψ_2^d:

$$\begin{cases} \phi_2^d uds_0 + \psi_2^d e^{2r} = & C(s_0, s_0u, s_0ud) = & uds_0 - 2 \\ \phi_2^d d^2 s_0 + \psi_2^d e^{2r} = & C(s_0, s_0u, s_0u^2) = & d^2 s_0 - 2 \end{cases}$$

Hence,

$$\phi_2^d = 1$$

$$\psi_2^d = -\frac{9}{2}$$

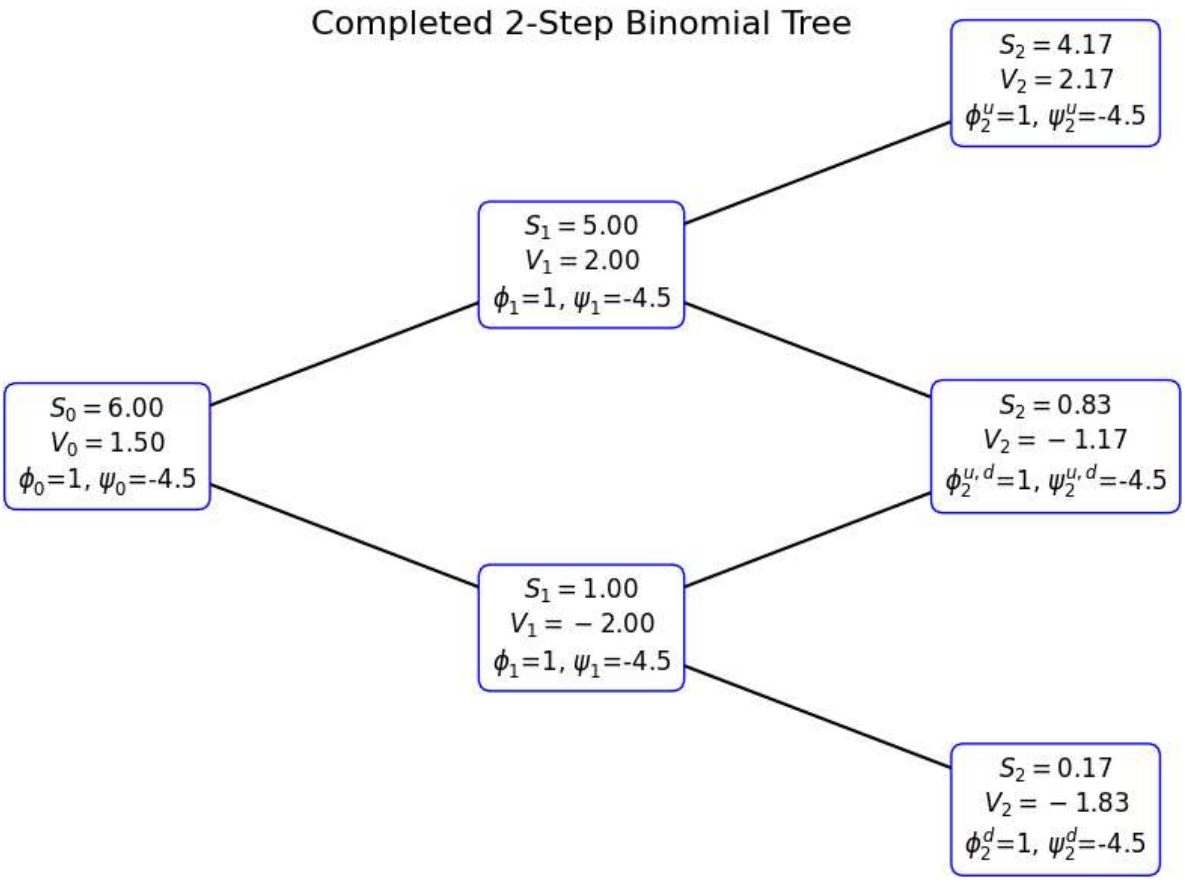
Adding the portfolio computations:

Let us now show the following Python code now adding the values of the replicating portfolio.

In [3]:

```
import numpy as np
import matplotlib.pyplot as plt

# Parameters for the binomial model
s0 = 6    # Initial stock price
K = 2     # Strike price
u = 5/6     # Up factor
d = 1/6     # Down factor
r = np.log(2/3)    # Risk-free rate
T = 1       # Time to maturity
n_steps = 2 # Number of steps

phi = 1 # Notice that the replicating portfolio is always constant
psi = -9/2
# Risk-neutral probability
q= (np.exp(r) - d) / (u - d)

# Compute stock prices at each node
S = np.zeros((n_steps + 1, n_steps + 1))  # Stock price tree
S[0, 0] = s0
for i in range(1, n_steps + 1):
    S[i, 0] = S[i - 1, 0] * u  # Up move
    for j in range(1, i + 1):
        S[i, j] = S[i - 1, j - 1] * d  # Down move

V = np.zeros((n_steps + 1, n_steps + 1))  # Contract value tree

# Calculate payoff at final nodes (step n_steps)
for j in range(n_steps + 1):
    V[n_steps, j] = S[n_steps, j] - K  # Payoff for a forward contract

# Backtrack to get the contract values at earlier nodes
for i in range(n_steps - 1, -1, -1):
    for j in range(i + 1):
        V[i, j] = np.exp(-r) * (q * V[i + 1, j] + (1 - q) * V[i + 1, j + 1])

# Plot the binomial tree
fig, ax = plt.subplots(figsize=(8, 6))
ax.set_title('Completed 2-Step Binomial Tree', fontsize=16)

# Plot the nodes and annotate with stock prices and contract values
for i in range(n_steps + 1):
    for j in range(i + 1):
        x = i  # horizontal position (step number)
        y = i - 2 * j  # vertical position (arbitrarily chosen for visualization
        ax.plot(x, y, 'bo', markersize=10)  # Plot node
        # Annotate with S_n (stock price) and V_n (contract value)
        superscript = ""

        if i == 2 and j == 0:
            superscript = "u"
        if i == 2 and j == 1:
            superscript = "u, d"
```

```
        elif i==2 and j == 2:
            superscript = "d"

        ax.text(x, y, f"$S_{{{i}}}={S[i, j]:.2f}$\n$V_{{{i}}}={V[i, j]:.2f}$\n$\
        ha='center', va='center', fontsize=12,
        bbox=dict(facecolor='white', edgecolor='blue', boxstyle='round,pad=0.5')

# Draw lines between the nodes to represent possible transitions
for i in range(n_steps):
    for j in range(i + 1):
        x = i
        y = i - 2 * j
        # Line to up move
        ax.plot([x, x + 1], [y, y - 1], 'k-', lw=1.5)
        # Line to down move
        ax.plot([x, x + 1], [y, y + 1], 'k-', lw=1.5)

# Hide axes for better visualization
ax.axis('off')
plt.tight_layout()
plt.show()
```



In the figure, $\phi_2^{u,d}$ means that either we can refer to ϕ_2^u or ϕ_2^d, and analogous for $\psi_2^{u,d}$.

Notice that we have selected, without loss of generality, that $\phi_0 = 1$, since we had one degree of freedom. Hence, ψ_0 gets completely determined by:

$$V_0 = \phi_0 S_0 + \psi_0,$$

$$\frac{3}{2} = 6 + \psi_0 \implies \psi_0 = \frac{3}{2} - 6 = -\frac{9}{2}.$$

Hence, the replicating portfolio is, in fact, constant along $t = 0, 1, 2$ and for the values of S_0, S_1, S_2.

- Determine the contract price V_0, and also V_1;

We can refer to the calculations performed above.

$$q = \frac{e^r - d}{u - d} = \frac{e^{log(2/3) - \frac{1}{6}}}{\frac{5}{6} - \frac{1}{6}} = \frac{\frac{4-1}{6}}{\frac{5-1}{6}} = \frac{3}{4}.$$

Once calculated q and $V_2(s_0, S_1, S_2) = C(S_2) = S_2 - 2$ for all the possible values of S_1, S_2, we can calculate V_1, and via the pricing formula using the risk-neutral probability, we computed: $V_1 = V_1(S_1) = \begin{cases} e^{-r}\left(qC(s_0u^2) + (1-q)C(s_0du)\right), & \text{if} \quad S_1 = s_0u, \\ e^{-r}\left(qC(s_0ud) + (1-q)C(s_0d^2)\right), & \text{if} \quad S_1 = s_0d \end{cases}$

Hence, substituting for e^{-r}, q and $V_2(s_0, S_1, S_2) = C_2(S_2)$:

$$V_1 = V_1(S_1) = \begin{cases} e^{-r}\left(qC(s_0u^2) + (1-q)C(s_0du)\right) = e^{-r}\left(\frac{3}{4}\frac{13}{6} + \frac{1}{4}(-\frac{7}{6})\right) = \frac{3}{2}\left(\frac{13}{8} - \\ e^{-r}\left(qC(s_0ud) + (1-q)C(s_0d^2)\right) = e^{-r}\left(\frac{3}{4}(-\frac{7}{6}) + \frac{1}{4}(-\frac{1}{6})\right) = \frac{3}{2}\left(- \end{cases}$$

Hence, we get that

$$V_1(S_1) = \begin{cases} 2 & if \quad S_1 = s_0u, \\ -2 & if \quad S_1 = s_0d \end{cases}$$

And, for V_0, we computed it using the values computed when calculating V_1, and:

$$V_0 = V_0(S_0) = e^{-r}\left(qV_1(s_0u) + (1-q)V_1(s_0d)\right) = e^{-r}\left(\frac{3}{4}\cdot 2 + \frac{1}{4}\cdot -2\right) = \frac{3}{2}\left(\frac{3}{2} - \frac{1}{2}\right) =$$

.

And, more precisely, using recursiveness:

$$V_0(s_0) = e^{-r}(qV_1(s_0, us_0) + (1-q)V_1(s_0, ds_0)) =$$
$$e^{-2r}(q(qC_2(s_0, us_0, u^2s_0) + (1-q)C_2(s_0, us_0, uds_0)) + (1-q)(qC_2(s_0, ds_0, dus_0) + (1$$
$$e^{-2r}(q(q(u^2s_0 - 2) + (1-q)(uds_0 - 2)) + (1-q)(q(dus_0 - 2) + (1-q)(d^2s_0 - 2))) =$$

To sum up, the contract prices at time $t = 0, 1, 2$ are the ones given by the computations for V_0, V_1 and V_2, that yield:

$$V_0 = \frac{3}{2}$$
$$V_1 = \begin{cases} 2 & if \quad S_1 = s_0u, \\ -2 & if \quad S_1 = s_0d \end{cases}$$
$$V_2 = \begin{cases} \frac{13}{6} & if \quad S_2 = s_0u^2 \\ -\frac{7}{6} & if \quad S_2 = s_0ud \\ -\frac{11}{6} & if \quad S_2 = s_0d^2 \end{cases}$$

- The replicating portfolio has a very distinctive characteristic; what is it? make a connection to how we priced a forward in the first chapter;

First of all, in the context of a replicating portfolio, the variables ϕ and ψ we saw that represented the following:

- ϕ the number of units of the underlying asset (a stock or commodity) held in the portfolio
- ψ the amount of money invested in the risk-free asset (a bond or just cash)

However, we have not given a clear interpretation for negative values of ϕ and/or ψ. When either of them are negative, it implies a particular type of financial position or strategy. Let's examine the meaning of negative ϕ and negative ψ separately and their implications in a portfolio.

Implications of negative ϕ: When ϕ is negative, we say that the investor is shorting the underlying asset. Shorting involves borrowing the asset, selling it immediately and aiming to repurchase it later at a lower price to return it to the lender and close the position. Therefore, the investor profits if the asset's price decreases.

For instance, $\phi = -1$ would mean to sell 1 unit of the asset, expecting to repurchase it later at a lower price. If the underlying asset would rise in value, the portfolio would lose value from the short position.

Implications of negative ψ: When ψ is negative, we say that the investor is borrowing money instead of holding an investment. Therefore, a negative ψ implies a position where the investor has taken out a loan to finance additional investment.

For instance, $\psi = -1$ would mean borrowing 1 unit of currency to invest in the underlying asset. If the return on the investments made with the borrowed money is larger than the cost of borrowing (note that it depends on the risk-free rate r), the investor profits. Else, when paying the interest of the loan, the investor would incurr in a loss.

Back to our case:

In our case, notice that the distinctive characteristic is that, for $n = 1, 2$, it happens that ψ is always $-\frac{9}{2}$ and $\phi = 1$. That means, that we mantain the same position at all times until maturity ($N = 2$). The position is specifically to borrow $\psi = \frac{9}{2}$ units of currency and buy one $\phi = 1$ unit of stock.

Let us now connect it to how we priced a forward in the first chapter. Recall that the payoff of the forward contract was $C(S_2) = S_2 - 2$. That means that the holder of the forward contract is going to receive $S_2 - 2$ at maturity, depending on S_2 the price of the underlying asset at time $t = 2$. We discussed how that the replicating portfolio was constant with $\phi = 1$ and $\psi = -\frac{9}{2}$ meant that to replicate the forward contract's payoff $S_2 - 2$, one must hold 1 unit of the underlying asset and borrow $\frac{9}{2}$ at the risk-free rate.

Recall how we priced the forward contract in a no-arbitrage environment. We claimed that the price of the forward contract with strike K was $f_0 = S_0 - Ke^{-rT}$. Then, we proved that it was indeed the fair price by showing that if the contract was priced as $F > f_0$ or $F < f_0$, we could construct an arbitrage strategy.

In particular, if $F > f_0$:

At time 0:

- We borrow S_0 at the risk-free rate
- We buy the underlying asset at the current price S_0
- We can write a forward contract and we sell it for F.
- We invest F at the risk-free interest rate.

At maturity (in this case at time $N = 2$):

- We deliver the asset to fulfill the forward contract, and we receive $K = 2$ for the asset
- We use the earnings ($Fe^{rN} + 2$) to repay the borrowed amount S_0e^{rN}

Hence, the profit is $Fe^{rN} + 2 - S_0e^{rN} = F - f_0 > 0$ which is positive by assumption.

And, otherwise, if $F < f_0$:

At time 0:

- We borrow the underlying asset and sell it immediately for S_0
- We borrow $F - S_0$ at the risk-free rate.
- We buy the forward contract for F, we can because now we have $F = F - S_0 + S_0$

At maturity (in this case at time $N = 2$):

- We buy back the asset at the forward's agreed strike price $K = 2$
- We repay with the asset to the lender of the underlying asset and we pay what we owe $(F - S_0)e^{rN}$.

Therefore, the profit is
$-K - (F - S_0)e^{rN} = S_0e^{rN} - K - Fe^{rN} = f_0e^{rN} - Fe^{rN} > 0$, which is positive by assumption.

In this way, we proved that effectively, the price of a forward contract is
$f_0 = S_0 - Ke^{-rT}$. In this case, $f_0 = S_0 - Ke^{-2r}$.

Coming back to our replicating portfolio, we showed that the portfolio replicated the payoff of the $S_2 - 2$ at maturity $N = 2$. But the value of the replicating portfolio at time $t = 0$ must equal the theoretical value of the forward contract, to ensure that no arbitrage opportunities exist. Hence $f_0 = V_0$, the value of the replicating portfolio at time $t = 0$.

Therefore:

$$f_0 = S_0 - 2e{-}rN = 6 - 2 \cdot \frac{9}{4} = \frac{3}{2}, \quad V_0 = \frac{3}{2}$$

And hence, effectively, $f_0 = V_0$, and we notice that there is an intimitate relationship between the replicating portfolio and the pricing of a forward contract. In particular, we notice that the price of the forward is the cost of the replicating portfolio.

- Suppose someone is willing to buy the contract with maturity 2 for a price $P > V_0$. Construct an explicit arbitrage strategy.

Notice that, since $f_0 = V_0$, we can refer back to the construction of the arbitrage strategy for a forward, and, referring to the strategy above, but now calling the overpriced forward contract price P instead of F:

In particular, supposing that someone is willing to buy the contract for $P > V_0 = f_0$.

At time 0:

- We borrow S_0 at the risk-free rate
- We buy the underlying asset at the current price S_0
- We can write a forward contract and we sell it for P.
- We invest P at the risk-free interest rate.

At maturity (in this case at time $N = 2$):

- We deliver the asset to fulfill the forward contract, and we receive $K = 2$ for the asset
- We use the earnings ($Pe^{rN} + 2$) to repay the borrowed amount $S_0 e^{rN}$

Hence, the profit is $Pe^{rN} + 2 - S_0 e^{rN} = P - f_0 = P - V_0 > 0$ which is positive by assumption.

Exercise 2

Let us prove Proposition 4.4. In the binomial model $d < e^r < u$, let (ϕ_n, ψ_n) be a self-financing portfolio with the associated value process V_n, $n \geq 0$. Then, the discounted value process $(\tilde{V}_n)_n = (e^{-nr} V_n)_n$ is a martingale under Q.

By construction, the value process associated to the binomial model is:

$$V_n = \phi_{n+1} S_n + \psi_{n+1} e^{nr} = \phi_n S_n + \psi_n e^{nr}$$

And the second equality is true because the portfolio $(\phi_n, \psi_n)_n$ is self-financing. Let us now show that $(\tilde{V}_n)_n$ is a martingale under Q.

Let us show that $(\tilde{V}_n)_n$ is adapted:

Let $(\mathcal{F}_n)$ be the natural filtration with respect to the prices $(\mathcal{F}_n)_n = \sigma(S_0, \ldots, S_N)$ where N is the maturity time.

Let $\tilde{V}_n = e^{-nr}V_n = e^{-nr}\left(\phi_{n+1}S_n + \psi_{n+1}e^{nr}\right) = \phi_{n+1}e^{-nr}S_n + \psi_{n+1}e^{nr-nr}$.

But this last term can be simplified, using that $\tilde{S}_n = e^{-nr}S_n$:

$\tilde{V}_n = \phi_{n+1}\tilde{S}_n + \psi_{n+1}$.

But notice that:

- The process of discounted prices $\tilde{S}_n$ is adapted to $\mathcal{F}_n$ because, fixed n, it is a constant multiplied by S_n, which we know because of the natural filtration at $\mathcal{F}_n = \sigma(S_0, \ldots, S_n)$. Hence, $\tilde{S}_n = e^{-rn}S_n$ is adapted to $\mathcal{F}_n$.
- By construction of the replicating portfolio of the binomial model, both ϕ_n and ψ_n are predictable. That means that $\mathbb{E}[\phi_{n+1}|\mathcal{F}_n] = \phi_n$ and $\mathbb{E}[\psi_{n+1}|\mathcal{F}_n] = \psi_n$.

Therefore, $(\tilde{V}_n)_n$ is a product of predictable and adapted summed to another predictable process. Thus, it is adapted.

Let us now show that $\mathbb{E}|\tilde{V}_n| < \infty$:

We will use the Martingale Representation theorem. Let $\phi_n = \frac{\Delta M_n}{\Delta \tilde{S}_n}$, for M_n a martingale process with respect to the natural filtration and $\tilde{S}_n$ the discounted stock price process. Analogously, let $\psi_n = \frac{\Delta M'_n}{\Delta \tilde{S}_n}$, for M'_n another martingale process with respect to the natural filtration and $\tilde{S}_n$ the discounted stock price process. Then, assuming that $|S_0| < \infty$:

$\mathbb{E}|\tilde{V}_n| = \mathbb{E}|\phi_{n+1}\tilde{S}_n + \psi_{n+1} \leq \mathbb{E}|\phi_{n+1}e^{-nr}u^n|S_0| + \psi_{n+1}| = |S_0|u^n e^{-nr}\mathbb{E}|\phi_{n+1}| + \mathbb{E}|\psi_{n+}$
.

Now, applying the Martingale Representation theorem for ϕ_{n+1} and ψ_{n+1}:
$\mathbb{E}|\tilde{V}_n| = |S_0|u^n e^{-nr}\mathbb{E}|\frac{\Delta M_{n+1}}{\Delta \tilde{S}_{n+1}}| + \mathbb{E}|\frac{\Delta M'_{n+1}}{\Delta \tilde{S}_{n+1}}|$

In the binomial model, the stock price process S_n (and hence $\tilde{S}_n = e^{-rn}S_n$) stays within a bounded range since the up multiplier is $u > 1$ and the down multiplier is $d > 0$. Specifically:

$S_n \geq S_0 d^n \quad S_n \leq S_0 u^n$.

Then, the increments are also bounded, and $\Delta\tilde{S}_{n+1}$, the increment of the discounted stock price process, is not too large or small, in the sense that $\mathbb{E}|\Delta\tilde{S}_{n+1}| < \infty$.

On the other hand, we will now use that the infimum of $|\Delta\tilde{S}_{n+1}|$ is strictly greater than 0 (specifically, that is bounded away from zero). We also assume that the up multiplier is $u < 1$ (so that the price goes up) and the down multiplier is $d < 1$ (so that the price goes down when multiplied).

$\Delta\tilde{S}_{n+1} = \tilde{S}_{n+1} - \tilde{S}_n$ and

$$\Delta\tilde{S}_{n+1} = \begin{cases} e^{-r}\tilde{S}_n u - e^{-r}\tilde{S}_n = e^{-r}\tilde{S}_n(u-1), & \text{in an up move} \\ e^{-r}\tilde{S}_n(d-1), & \text{in a down move} \end{cases}$$

Now using our assumptions on u and d, the absolute value of $\Delta\tilde{S}_{n+1}$ is given by:

$$|\Delta\tilde{S}_{n+1}| = e^{-r}|\tilde{S}_n| \max(|u-1|, |1-d|)$$

Hence, the infimum of $|\Delta\tilde{S}_{n+1}|$ is bounded by:

$$\inf|\Delta\tilde{S}_{n+1}| = e^{-r} \inf|\tilde{S}_n| \min(|u-1|, |1-d|)$$

Now, using that $\inf|\tilde{S}_n| = e^{-rn}d^n|S_0| > 0$ for every n, since e^{-rn} is positive and $0 < d < 1$.

Since M_n and M'_n are martingales, by definition they have finite absolute value expectations. We will now show that the increments of the martingales also have finite absolute value expectations.

$\mathbb{E}|M_n| < \infty$:

$$\mathbb{E}|\Delta M_{n+1}| = \mathbb{E}|M_{n+1} - M_n|$$

$$|M_{n+1} - M_n| \leq |M_{n+1}| + |M_n|$$

Taking expectations:

$$\mathbb{E}|M_{n+1} - M_n| \leq \mathbb{E}|M_{n+1}| + \mathbb{E}|M_n|$$

Where the inequality comes from the use of the triangle inequality. Notice that this argument is also valid for M'_n.

Let us now control the absolute value expectations of ϕ_{n+1} and ψ_{n+1}:

$$\mathbb{E}|\phi_{n+1}| = \mathbb{E}\left|\frac{\Delta M_{n+1}}{\Delta\tilde{S}_{n+1}}\right| \leq \frac{\mathbb{E}|\Delta M_{n+1}|}{\inf|\Delta\tilde{S}_{n+1}|}$$

But we have shown that $\mathbb{E}|\Delta M_{n+1}| < \infty$, $|\tilde{S}_{n+1}|$ is bounded above and below and $\inf|\tilde{S}_{n+1}| > 0$ for every n fixed. Hence,

$$\mathbb{E}|\phi_{n+1}| < \infty.$$

A similar argument is also valid for ψ_{n+1}:

$$\mathbb{E}|\psi_{n+1}| = \mathbb{E}\left|\frac{\Delta M'_{n+1}}{\Delta\tilde{S}_{n+1}}\right| \leq \frac{\mathbb{E}|\Delta M'_{n+1}|}{\inf|\Delta\tilde{S}_{n+1}|}$$

But we have shown that $\mathbb{E}|\Delta M'_{n+1}| < \infty$, $|\tilde{S}_{n+1}|$ is bounded above and below and $\inf|\tilde{S}_{n+1}| > 0$ for every n fixed. And:

$$\mathbb{E}|\psi_{n+1}| < \infty.$$

Coming back to $\mathbb{E}|\tilde{V}_n|$:

$$\mathbb{E}|\tilde{V}_n| = |S_0|u^n e^{-nr}\mathbb{E}|\frac{\Delta M_{n+1}}{\Delta\tilde{S}_{n+1}}| + \mathbb{E}|\frac{\Delta M'_{n+1}}{\Delta\tilde{S}_{n+1}}| < \infty$$

Let us show that $(\tilde{V}_n)_n$ complies with the martingale property:

Let us observe and prove (although we have seen that in class) that $\tilde{S}_n$ is a martingale (given $d < e^r < u$) with respect to the natural filtration of the stock prices $\mathcal{F}_n = \sigma(S_1, \dots, S_n)$.

- $\tilde{S}_n$ is adapted because $\tilde{S}_n = e^{-rn}S_n$ and we know the value of S_n at exactly $\mathcal{F}_n = \sigma(S_0, \dots, S_n)$.
- It is integrable because $\mathbb{E}|\tilde{S}_n| = e^{-rn}\mathbb{E}|S_n| \leq e^{-nr}u^n|S_0| < \infty$, assuming that $|S_0| < \infty$.
- It complies with the martingale property since (recall that $uq + d(1-q) = e^r$ because $q = \frac{e^r - d}{u-d}$):

  $\mathbb{E}_q[\tilde{S}_{n+1}|\mathcal{F}_n] = \mathbb{E}[\tilde{S}_{n+1}|S_0, \dots, S_n] = e^{-r(n+1)}S_n\left(uq + d(1-q)\right) = e^{-r(n+1)}S_n e^r =$

  .

Thus $\tilde{S}_n$ is a martingale with respect to the natural filtration of the stock prices $\mathcal{F}_n$. Let us now prove that $\tilde{V}_n$ is also a martingale with respect to the same filtration. We have already proven that it is adapted and measurable. Now:

$\mathbb{E}_q[\tilde{V}_{n+1}|\mathcal{F}_n] = \mathbb{E}_q[e^{-r(n+1)}V_{n+1}|\mathcal{F}_n] = \mathbb{E}_q[e^{-r(n+1)}(\phi_{n+2}S_{n+1} + \psi_{n+2}e^{(n+1)r})|\mathcal{F}_n] = \mathbb{E}_q[$

.

Where the last equality comes from the fact that $(\phi_n, \psi_n)_n$ is a self-financing portfolio, and hence $V_n = \phi_{n+1}S_n + \psi_{n+1}e^{rn} = \phi_n S_n + \psi_n e^{rn}$ for every n (and thus also for $n+1$).

Recall how above we used the Martingale Representation theorem for both ϕ_{n+1} and ψ_{n+1}. Here, we will only apply it to ϕ_{n+1}. Also, notice that $\tilde{S}_{n+1} = \tilde{S}_n + \Delta\tilde{S}_{n+1}$. Let us decompose it this way in the above formula in the following manner:

$\mathbb{E}_q[\tilde{V}_{n+1}|\mathcal{F}_n] = \mathbb{E}_q[\phi_{n+1}(\tilde{S}_n + \Delta\tilde{S}_{n+1}) + \psi_{n+1}] = \mathbb{E}_q[\phi_{n+1}\tilde{S}_n + \psi_{n+1} + \phi_{n+1}\Delta\tilde{S}_{n+1}] =$

Let us go term by term:

- First term. Notice that it is exactly $\tilde{V}_n$: $\mathbb{E}_q[\phi_{n+1}\tilde{S}_n + \psi_{n+1}|\mathcal{F}_n] = \mathbb{E}_q[\tilde{V}_n|\mathcal{F}_n] = \tilde{V}_n$ because we have proven that $\tilde{V}_n$ is adapted to $\mathcal{F}_n$.
- Second term: $\mathbb{E}_q[\phi_{n+1}\Delta\tilde{S}_{n+1}|\mathcal{F}_n] = \mathbb{E}_q[\Delta M_{n+1}|\mathcal{F}_n]$ because of the Martingale Representation theorem applied to ϕ_n. But the increment of the martingale can be decomposed as:

  $\mathbb{E}_q[\Delta M_{n+1}|\mathcal{F}_n] = \mathbb{E}_q[M_{n+1} - M_n|\mathcal{F}_n] = \mathbb{E}_q[M_{n+1}|\mathcal{F}_n] - \mathbb{E}_q[M_n|\mathcal{F}_n] = M_n - M_n =$

  .

  That is because M_n complies with the martingale property and because M_n is adapted. Hence, the second term is just 0.

Therefore,

$\mathbb{E}_q[\tilde{V}_{n+1}|\mathcal{F}_n] = \tilde{V}_n + 0 = \tilde{V}_n$

And we have proven that $\tilde{V}_n$ is a martingale with respect to the natural filtration of the stock prices $\mathcal{F}_n = \sigma(S_0, \ldots, S_n)$.

Exercise 3

Let $C = C(S_0, \ldots, S_N)$ be the pay-off of the European Call Option contract with strike K and maturity N and let $P = P(S_0, \ldots, S_N)$ be the pay-off of the European Put Option contract with same strike K and same maturity N. We are going to work on the natural filtration with respect of the prices $\mathcal{F}_n = \sigma(S_0, \ldots, S_n)$ for all n from 0 to N.

The European Call Option has pay-off:

$C_N = (S_N - K)^+ = \max(S_N - K, 0)$

On the other hand, the European Put Option has pay-off:

$P_N = (K - S_N)^+ = \max(K - S_N, 0)$

First, let us compute the values of C_n and P_n. For that, we have to get the fair price of the contracts with pay-off C at time n and P at time n, this is:

$C_n = e^{-r(N-n)}E_q[C(S_0, \ldots, S_N)|\mathcal{F}_n] = e^{-r(N-n)}E_q[C(S_N)|\mathcal{F}_n],$

$P_n = e^{-r(N-n)}E_q[P(S_0, \ldots, S_N)|\mathcal{F}_n] = e^{-r(N-n)}E_q[P(S_N)|\mathcal{F}_n],$

Therefore, we have that:

$C_n = e^{-(N-n)}E_q[(S_N - K)^+|\mathcal{F}_n]$

$P_n = e^{-r(N-n)}E_q[(K - S_N)^+|\mathcal{F}_n]$

Let us now compute the subtraction $C_n - P_n$:

$C_n - P_n = e^{-r(N-n)}\left(\mathbb{E}_q[(S_N - K)^+|\mathcal{F}_n] - (\mathbb{E}_q[(K - S_N)^+|\mathcal{F}_n])\right).$

By linearity of the expectation $\mathbb{E}_q$:

$C_n - P_n = e^{-r(N-n)}(\mathbb{E}_q[(S_N - K)^+ - (K - S_N)^+|\mathcal{F}_n]).$

Observe the following remark. What is the value of $(S_N - K)^+ - (K - S_N)^+$?

- If $S_N \geq K$, then $(K - S_N)^+ = 0$ and $(S_N - K)^+ - (K - S_N)^+ = S_N - K$.
- On the other hand, if $S_N < K$, then $(S_N - K)^+ = 0$.

Hence, observe that the value is $(S_N - K)^+ - (K - S_N)^+ = S_N - K$.

Let us plug this result in the equation that we had for the subtraction $C_n - P_n$:

$C_n - P_n = e^{-r(N-n)}\mathbb{E}_q[S_N - K|\mathcal{F}_n]$.

Because K is constant;

$e^{-r(N-n)}\mathbb{E}_q[S_N - K|\mathcal{F}_n] = e^{-r(N-n)}\mathbb{E}_q[S_N|\mathcal{F}_n] - Ke^{-r(N-n)}$

Observe that, taking into account that $(\tilde{S}_n)$ is a martingale and that $e^{rN}(\tilde{S}_N) = S_N$, we can substitute S_N in terms of $\tilde{S}_N$ by multiplicating by e^{rN} and e^{-rN} (notice that $e^{rN} \cdot e^{-rN} = 1$) and we get that:

$C_n - P_n = e^{-r(N-n)}\mathbb{E}_q[e^{rN}S_N e^{-rN}|\mathcal{F}_n] - Ke^{-r(N-n)}$

Since e^{rN} is deterministic, we can put it out from the expectation, and we get that:

$C_n - P_n = e^{-r(N-n)}e^{rN}\mathbb{E}_q[S_N e^{-rN}|\mathcal{F}_n] - Ke^{-r(N-n)} = e^{-r(N-n)}e^{rN}\mathbb{E}_q[\tilde{S}_N|\mathcal{F}_n] - Ke^{-r}$

Also, since $(\tilde{S}_n)$ is a martingale with respect to the natural filtration of the prices $\mathcal{F}_n = \sigma(S_0, S_1, \ldots, S_N)$, we can use the martingale property $\mathbb{E}_q[\tilde{S}_N|\mathcal{F}_n] = \tilde{S}_n$:

$C_n - P_n = e^{-r(N-n)}e^{rN}\mathbb{E}_q[\tilde{S}_N|\mathcal{F}_n] - Ke^{-r(N-n)} = e^{-r(N-n)}e^{rN}\tilde{S}_n - Ke^{-r(N-n)}$

Multiplicating the exponential terms:

$C_n - P_n = e^{-r(N-n)}e^{rN}\tilde{S}_n - Ke^{-r(N-n)} = e^{rn}\tilde{S}_n - Ke^{-r(N-n)}$

But notice that $e^{rn}\tilde{S}_n$ is exactly S_n, and we conclude that:

$$C_n - P_n = S_n - Ke^{-r(N-n)}, \quad \forall n \text{ such that } 0 \leq n \leq N.$$

Which is the put-call parity relationship.

Exercise 4

Let us first show a code that prices stocks according to a T Binomial Model (the times go from $t = 0$ to $t = T$), where the interest rate is compounded daily and not continuously. That is why we use $1 + r_1$ instead of e^{r_1}. The initial stock price is set to 15€. In this case, the prices are priced up until $T = 10$ days.

The matrix stockprice represents all the stock prices up to maturity if, at each step, they can be multiplied by u (that is, the stock price goes up) or by d (that is, the stock price goes down) for each time step j, and in this way the matrix is filled.

Later on, the code also prices a call option using the risk neutral probability for up in our model and the conditional expectation in the following way. It uses the binomial model's backward induction to compute the option prices at earlier time steps until we get to $j = 0$, which represents the current price of the contract.

$C_{i,j} = \frac{1}{1+r_1}\mathbb{E}[C_{i,j+1}|\mathcal{F}_j] = \frac{1}{1+r_1}\big(q \cdot C_{i,j+1} + (1-q) \cdot C_{i+1,j+1}\big)$

Notice that the value at a given node is determined by the expected future payoffs discounted back to the present value using the interest rate r_1.

Note that the European call option is computed inductively backwards from the pay-off, which is:

$$C(S_0, \ldots, S_T) = C(S_T) = \max(S_T - K, 0)$$

In [4]:
```
import numpy as np

s = 15 # stock price at time 0 in euro
u = 1.05 # up multiplier
d = 0.79 # down multiplier
T = 10 # maturity time
K = 8 # strike price
r1 = 0.045/365 # daily interest rate; 1+r1 multiplier, not exp(r1)
q = (1+r1-d)/(u-d) # risk neutral probability for up

# Determine possible stock prices up to maturity
stockprice = np.full((T+1,T+1),np.nan)
stockprice[0,0] = s

for j in range(1,T+1):
    for i in range(0,j):
        stockprice[i,j] = stockprice[i,j-1]*u
        stockprice[i+1,j] = stockprice[i,j-1]*d
print("Stock prices:")
print(np.around(stockprice, 2)) #just for visualisation

# Determine European call option prices and conditional values
calloption = np.full((T+1,T+1),np.nan)
calloption[:,-1] = np.where(stockprice[:,-1]-K<0,0,stockprice[:,-1]-K)
for j in range(T-1,-1,-1):
    for i in range(0,j+1):
        calloption[i,j] = (q*calloption[i,j+1]+(1-q)*calloption[i+1,j+1])/(1+r1)
print("Call option prices:")
print(np.around(calloption, 2)) #just for visualisation
```

```
Stock prices:
[[15.   15.75 16.54 17.36 18.23 19.14 20.1  21.11 22.16 23.27 24.43]
 [  nan 11.85 12.44 13.06 13.72 14.4  15.12 15.88 16.67 17.51 18.38]
 [  nan   nan  9.36  9.83 10.32 10.84 11.38 11.95 12.55 13.17 13.83]
 [  nan   nan   nan  7.4   7.77  8.15  8.56  8.99  9.44  9.91 10.41]
 [  nan   nan   nan   nan  5.84  6.13  6.44  6.76  7.1   7.46  7.83]
 [  nan   nan   nan   nan   nan  4.62  4.85  5.09  5.34  5.61  5.89]
 [  nan   nan   nan   nan   nan   nan  3.65  3.83  4.02  4.22  4.43]
 [  nan   nan   nan   nan   nan   nan   nan  2.88  3.02  3.18  3.33]
 [  nan   nan   nan   nan   nan   nan   nan   nan  2.28  2.39  2.51]
 [  nan   nan   nan   nan   nan   nan   nan   nan   nan  1.8   1.89]
 [  nan   nan   nan   nan   nan   nan   nan   nan   nan   nan  1.42]]
Call option prices:
[[ 7.09  7.81  8.57  9.38 10.24 11.15 12.11 13.11 14.16 15.27 16.43]
 [  nan  4.07  4.59  5.16  5.77  6.43  7.13  7.88  8.68  9.51 10.38]
 [  nan   nan  1.89  2.21  2.58  2.99  3.46  3.98  4.55  5.17  5.83]
 [  nan   nan   nan  0.54  0.67  0.83  1.03  1.27  1.57  1.94  2.41]
 [  nan   nan   nan   nan  0.    0.    0.    0.    0.    0.    0.  ]
 [  nan   nan   nan   nan   nan  0.    0.    0.    0.    0.    0.  ]
 [  nan   nan   nan   nan   nan   nan  0.    0.    0.    0.    0.  ]
 [  nan   nan   nan   nan   nan   nan   nan  0.    0.    0.    0.  ]
 [  nan   nan   nan   nan   nan   nan   nan   nan  0.    0.    0.  ]
 [  nan   nan   nan   nan   nan   nan   nan   nan   nan  0.    0.  ]
 [  nan   nan   nan   nan   nan   nan   nan   nan   nan   nan  0.  ]]
```

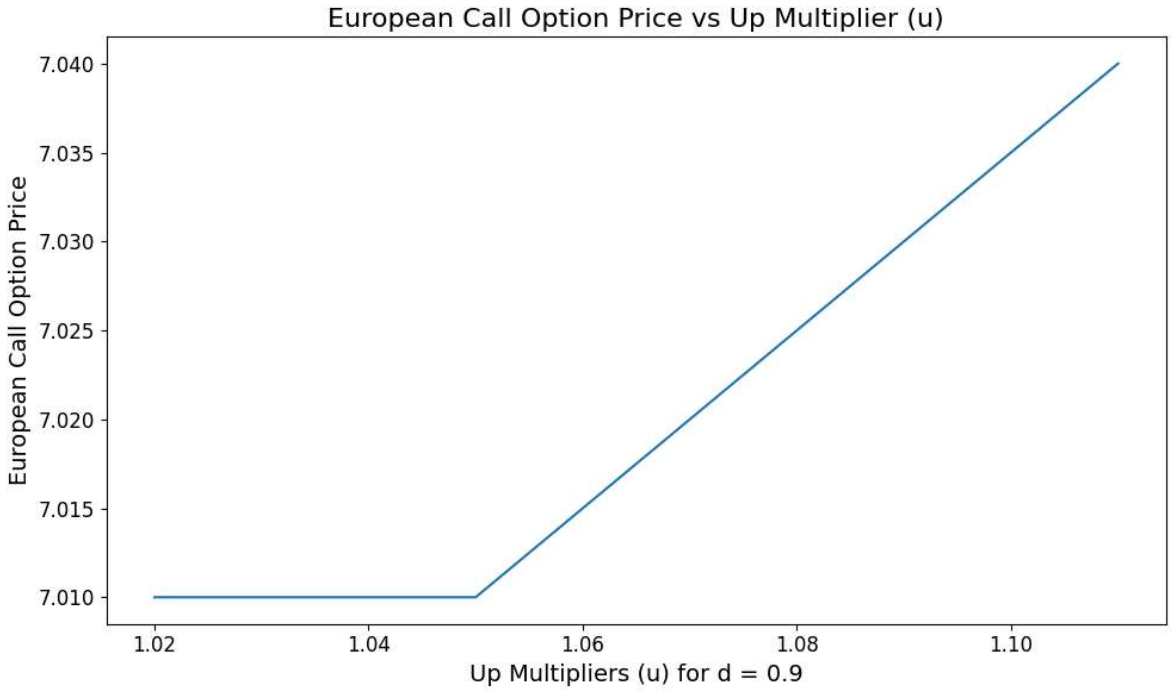
(a) Write a function that determines the value of a European call option for a user specified combination of d and u. Fix $d = 0.90$ and determine the price of the call option for various values of u, for instance, $u = 1.02, 1.03, 1.05, 1.11$. Plot these values in a graph. Repeat this for $d = 0.795$.

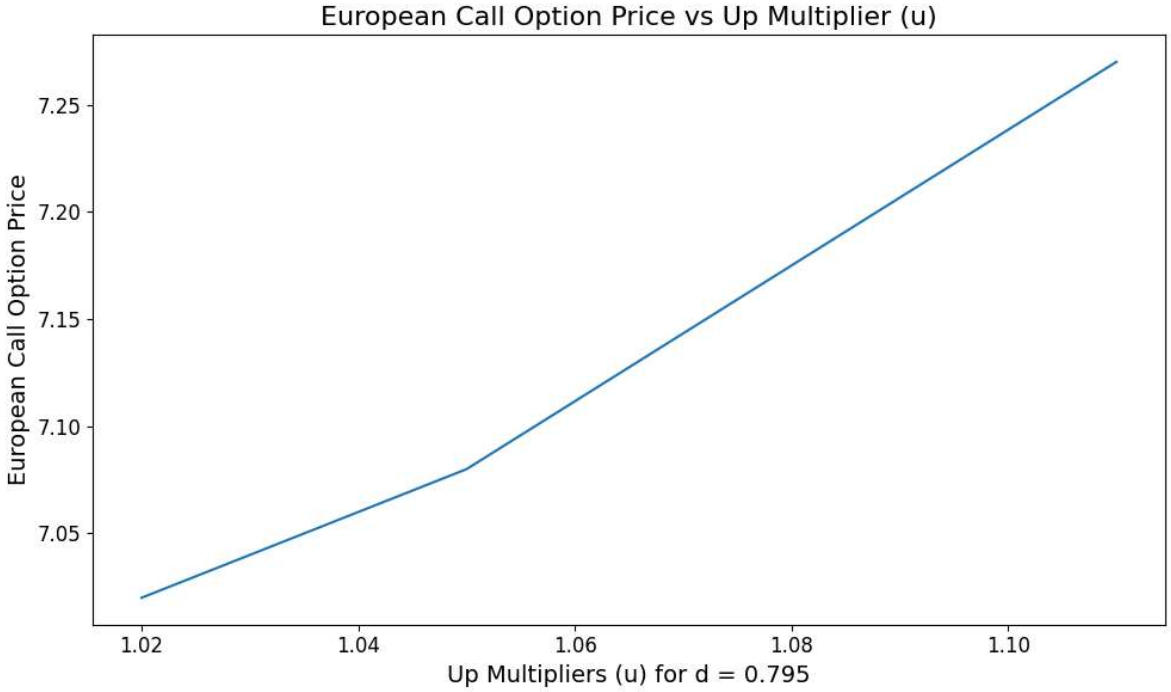
First, we program a function called $value\,european\,call(d, u)$, which, for d and u parameters that can be modified, given the same restrictions of the initial code that we showed, it computes the two matrices of stock prices and the prices of the call option at different time steps according to the same model in the initial code, but we add a line so that we return the value of the call option at current time ($calloption[0, 0]$).

In [5]:
```python
def value_european_call(d,u):
    s = 15 # stock price at time 0 in euro
    T = 10 # maturity time
    K = 8 # strike price
    r1 = 0.045/365 # daily interest rate; 1+r1 multiplier, not exp(r1)
    q = (1+r1-d)/(u-d) # risk neutral probability for up

    # Determine possible stock prices up to maturity
    stockprice = np.full((T+1,T+1),np.nan)
    stockprice[0,0] = s

    for j in range(1,T+1):
        for i in range(0,j):
            stockprice[i,j] = stockprice[i,j-1]*u
            stockprice[i+1,j] = stockprice[i,j-1]*d

    calloption = np.full((T+1,T+1),np.nan)
    calloption[:,-1] = np.where(stockprice[:,-1]-K<0,0,stockprice[:,-1]-K) # Pay
    for j in range(T-1,-1,-1):
        for i in range(0,j+1):
            calloption[i,j] = (q*calloption[i,j+1]+(1-q)*calloption[i+1,j+1])/(1
    return np.round(calloption[0,0],2) # Value of the option contract at time 0
```

Notice that the current European call option price is $calloption[0,0]$, and we round it to the two decimals for convenience. Hence, if we now want to compute the price of the call option for different values of u and d, we can plug them in at this function that we defined above.

In [6]:
```
import matplotlib.pyplot as plt

def plot_in_function_of_u(d):
    X = np.array([1.02, 1.03, 1.05, 1.11])
    Y = np.array([])
    for x in X:
        Y = np.append(Y, value_european_call(d,x))
    plt.figure(figsize=(10,6))
    plt.plot(X,Y)
    plt.title('European Call Option Price vs Up Multiplier (u)', fontsize=16)
    plt.xlabel('Up Multipliers (u) for d = ' + str(d) + '', fontsize=14)
    plt.ylabel('European Call Option Price', fontsize=14)
    plt.xticks(fontsize=12)
    plt.yticks(fontsize=12)
    plt.tight_layout()
    plt.show()
    return X,Y


X_90, Y_90 = plot_in_function_of_u(d=0.90)
X_795, Y_795 = plot_in_function_of_u(d=0.795)
```





(b) What is the observed relation between the values of u (and d) and the call option price? Is there an intuitive explanation for these relations?

First of all, notice that $u > 1$ and $d < 1$, strictly, as multipliers. The observed relation between the values of u fixed d is the following. As the up multiplier u increases, the potential future stock prices when the stock goes up rise, only in the upward movement scenario. Hence, the downward movement, fixed d, is the same, but the upward movement is greater as u increases.

As a consequence, the expected price of the call option becomes more valuable, because the underlying stock has a higher potential to increase above the strike price (in-the-money $S_T > K$).

Analogously, as the d multiplier increases, the potential future stock prices when the stock goes down rise, only in the downward movement scenario. Hence, the upward movement, fixed u, is the same, but the downward movement is not as great as d increases. As a consequence, the expected price of the call option becomes more valuable, because now when the prices go down, they don't go "down that much".

We have explained the intuition behind it, but not the mathematics. Let us show that $C_{0,0}$ increases with u and with d.

We know that, by the binomial model:

$$C_{0,0} = \frac{1}{(1+r_1)^T} \sum_{i=0}^{T} \binom{T}{i} q^i (1-q)^{T-i} \max(0, S_0 \cdot u^i d^{T-i} - K).$$

Hence, the derivative with respect to u is:

$$\frac{\delta C_{0,0}}{\delta u} = \frac{1}{(1+r_1)^T} \sum_{i=0}^{T} \binom{T}{i} q^i (1-q)^{T-i} \frac{\delta}{\delta u} \max(0, S_0 \cdot u^i d^{T-i} - K),$$

but the derivative of the maximum function can be expressed as:

$$\frac{\delta}{\delta u}\max(0, S_0 \cdot u^i \cdot d^{T-i} - K) = \begin{cases} S_0 \cdot i \cdot u^{i-1} \cdot d^{T-i} & \text{if } S_0 \cdot u^i \cdot d^{T-i} > K \\ 0 & \text{otherwise} \end{cases}$$

Notice that, when we take the derivative of the call option price $C_{0,0}$ with respect to u, we are examining how the call option price changes as we make small changes to u (sensitivity to changes in the up multiplier). Thus, if this derivative is positive, it implies that an increase in u will result in an increase in the call option price.

When is the derivative positive?

The term $S_0 \cdot i \cdot u^{i-1} \cdot d^{T-i}$ is positive if:

- $S_0 > 0$ (the initial stock price is positive)
- $i > 0$ (there is at least one upward movement)
- $u^{i-1} > 0$ (since $u > 0$)
- $d^{T-i} > 0$ (since $d > 0$)

Therefore, the contribution to the derivative is positive and, for values of u that lead to more scenarios where the option is in-the-money (because the up movement is larger), the positive contributions increase.

In a similar fashion, let us show the derivative with respect to d:
$\frac{\delta C_{0,0}}{\delta d} = \frac{1}{(1+r_1)^T}\sum_{i=0}^{T}\binom{T}{i}q^i(1-q)^{T-i}\frac{\delta}{\delta d}\max(0, S_0 \cdot u^i d^{T-i} - K)$,

but the derivative of the maximum function can be expressed as:
$$\frac{\delta}{\delta d}\max(0, S_0 \cdot u^i \cdot d^{T-i} - K) = \begin{cases} S_0 \cdot (T-i) \cdot u^i \cdot d^{T-i.1} & \text{if } S_0 \cdot u^i \cdot d^{T-i} > K \\ 0 & \text{otherwise} \end{cases}$$

Notice that, when we take the derivative of the call option price $C_{0,0}$ with respect to d, we are examining how the call option price changes as we make small changes to d (sensitivity to changes in the down multiplier). Thus, if this derivative is positive, it implies that an increase in d will result in an increase in the call option price.

When is the derivative positive?

The term $S_0 \cdot (T-i) \cdot u^i \cdot d^{T-i-1}$ is positive if:

- $S_0 > 0$ (the initial stock price is positive)
- $i > 0$ (there is at least one upward movement)
- $u^i > 0$ (since $u > 0$)
- $d^{T-i-1} > 0$ (since $d > 0$)

Therefore, the contribution to the derivative is positive and, as d increases, that leads to downward movements that are lighter, making the option more valuable since, potentially, less out-of-the-money scenarious occur ($K - S_T$).

On the other hand, if either u or d decrease, the option price would be less valuable, since, in the first case, the upward movement would be lighter, hence being "less upward". In the latter case, the option would be less valuable since the downward

movement would be heavier, meaning that the downsides are greater when the stock does not go up.

To sum up, for a call option:

- u increases $\implies$ the call option is more valuable, since the upward movements are heavier
- d increases $\implies$ the call option is more valuable, since the downward movements are lighter
- u decreases $\implies$ the call option is less valuable, since the upward movements are lighter
- d decreases $\implies$ the call option is less valuable, since the downward movements are heavier

(c) Write a function that determines the value of a put option with the same strike price K as the call option from before for a user-specified combination of d and u. Investigate how the parameters u and d influence the price of the put option.

Let us define the analogous function of the European call but now for a European put option. Notice that, since the computation of the expectation with respect to the Q measure is the same, the only line that changes is the pay-offof the European put option:

$$P(S_0, \ldots, S_T) = P(S_T) = \max(K - S_T, 0)$$

In [7]:
```
def value_european_put(d,u):
    s = 15 # stock price at time 0 in euro
    T = 10 # maturity time
    K = 8 # strike price
    r1 = 0.045/365 # daily interest rate; 1+r1 multiplier, not exp(r1)
    q = (1+r1-d)/(u-d) # risk neutral probability for up

    # Determine possible stock prices up to maturity
    stockprice = np.full((T+1,T+1),np.nan)
    stockprice[0,0] = s

    for j in range(1,T+1):
        for i in range(0,j):
            stockprice[i,j] = stockprice[i,j-1]*u
            stockprice[i+1,j] = stockprice[i,j-1]*d

    putoption = np.full((T+1,T+1),np.nan)
    putoption[:,-1] = np.where(K-stockprice[:,-1]<0,0,K-stockprice[:,-1]) # Pay-
    for j in range(T-1,-1,-1):
        for i in range(0,j+1):
            putoption[i,j] = (q*putoption[i,j+1]+(1-q)*putoption[i+1,j+1])/(1+r1
    return np.round(putoption[0,0],2) # Value of the option contract at time 0
```

Now, to see how the parameters u and d influence the price of the put option, we can perform an analysis similar to that of the call option. First, we will show how the explicit formula of P_0 is computed, and then we will analyse the partial derivatives of P_0 with respect to u and with respect to d.

Let us show that $P_0 \equiv P_{0,0}$, by the binomial model:

$P_{0,0} = \frac{1}{(1+r_1)^T} \sum_{i=0}^{T} \binom{T}{i} q^i (1-q)^{T-i} \max(0, K - S_0 \cdot u^i d^{T-i})$.

Derivative with respect to u,

Hence, the derivative with respect to u is:

$\frac{\delta P_{0,0}}{\delta u} = \frac{1}{(1+r_1)^T} \sum_{i=0}^{T} \binom{T}{i} q^i (1-q)^{T-i} \frac{\delta}{\delta u} \max(0, K - S_0 \cdot u^i d^{T-i})$,

but the derivative of the maximum function can be expressed as:

$$\frac{\delta}{\delta u} \max(0, K - S_0 \cdot u^i d^{T-i}) = \begin{cases} -S_0 \cdot i \cdot u^{i-1} \cdot d^{T-i} & \text{if } K > S_0 \cdot u^i \cdot d^{T-i} \\ 0 & \text{otherwise} \end{cases}$$

Notice that, when we take the derivative of the put option price $P_{0,0}$ with respect to u, we are examining how the put option price changes as we make small changes to u (sensitivity to changes in the up multiplier). Thus, if this derivative is negative, it implies that an increase in u will result in a decrease in the put option price.

When is the derivative with respect to u negative?

The term $S_0 \cdot i \cdot u^{i-1} \cdot d^{T-i}$ is positive if:

- $S_0 > 0$ (the initial stock price is positive)
- $i > 0$ (there is at least one upward movement)
- $u^{i-1} > 0$ (since $u > 0$)
- $d^{T-i} > 0$ (since $d > 0$)

Since we are looking at the term $-S_0 \cdot i \cdot u^{i-1} \cdot d^{T-i}$, it is negative and the contribution to the derivative is negative. Therefore, for values of u that lead to more scenarios where the option is in-the-money (since the upward movement is heavier), the negative contributions increase, hence making the option less valuable.

We would do the same reasoning analogously, since **the derivatives are the ones calculated on the previous section but scaled by -1, making them negative. Hence, the consequences go in the opposite direction.** That is:

- u increases $\implies$ the put option is less valuable, since the upward movements are heavier
- d increases $\implies$ the put option is less valuable, since the downward movements are lighter
- u decreases $\implies$ the put option is more valuable, since the upward movements are lighter
- d decreases $\implies$ the put option is more valuable, since the downward movements are heavier

(d) Using the fact that the discounted stock prices form a martingale (w.r.t. the natural filtration) under the risk neutral measure, use your code illustrate the so called put-call parity as defined in exercise 3 of this assignment.

In Exercise 3 we had the following formula

$$C_n - P_n = S_n - e^{-r(N-n)}K, \quad n = 0, \ldots, N,$$

Notice that now we have daily compounding $(1+r_1)^{N-n}$ instead of continuous compounding $e^{-r(N-n)}$.

We will now check for the put-call parity at each time-step and at each path, to show that it is always preserved in our model. Let us denote $putcallparitycheck$ for a matrix that, at the element i, j is nan if there is no path i at the time step j, 1 if it complies with the put-call parity at each time step and 0 if it does not. At time 0, the put-call parity condition is $C_0 - S0 + K\frac{1}{(1+r_1)^T} = P_0$, but the condition that we will check is:

$$(C_{i,j} - P_{i,j} - stockprice[i,j] + K/(1+r_1)^{(T-j)}) < tol,$$

at each i, j (at each path along all the time steps), with $tol = 1e^{-9}$, to ensure that we are not confusing negligibly small values with 0.

In [8]:
```
q = (1+r1-d)/(u-d) # risk neutral probability for up

putcallparitycheck = np.full((T+1,T+1),np.nan)
calloption = np.full((T+1,T+1),np.nan)
putoption = np.full((T+1,T+1),np.nan)
calloption[:,-1] = np.where(stockprice[:,-1]-K<0,0,stockprice[:,-1]-K)
putoption[:,-1] = np.where(K - stockprice[:,-1]<0,0,K - stockprice[:,-1])
tol = 1e-9
for j in range(T-1,-1,-1):
    for i in range(0,j+1):
        calloption[i,j] = (q*calloption[i,j+1]+(1-q)*calloption[i+1,j+1])/(1+r1)
        putoption[i,j] = (q*putoption[i,j+1]+(1-q)*putoption[i+1,j+1])/(1+r1)
        discount = 1/(1+r1)**(T-j)
        putcallparitycheck[i,j] = (calloption[i,j] - putoption[i,j] - stockprice

for i in range(0,T+1):
    putcallparitycheck[i,T] = (calloption[i,T] - putoption[i,T] - stockprice[i,T

print("Put-Call Parity Check:")
print(putcallparitycheck)
```

```
Put-Call Parity Check:
[[ 1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.]
 [nan  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.]
 [nan nan  1.  1.  1.  1.  1.  1.  1.  1.  1.]
 [nan nan nan  1.  1.  1.  1.  1.  1.  1.  1.]
 [nan nan nan nan  1.  1.  1.  1.  1.  1.  1.]
 [nan nan nan nan nan  1.  1.  1.  1.  1.  1.]
 [nan nan nan nan nan nan  1.  1.  1.  1.  1.]
 [nan nan nan nan nan nan nan  1.  1.  1.  1.]
 [nan nan nan nan nan nan nan nan  1.  1.  1.]
 [nan nan nan nan nan nan nan nan nan  1.  1.]
 [nan nan nan nan nan nan nan nan nan nan  1.]]
```

As we can see, the put-call parity check is preserved along the paths and time steps of the European option contract modelled with the binomial model. Let us show everything summed up in Appendix A.

Appendix A: For exercise 4, computation of the binomial tree for N = 10

In [21]:
```
import numpy as np
import matplotlib.pyplot as plt

# Plot the binomial tree
fig, ax = plt.subplots(figsize=(30, 15))
ax.set_title('N=10-Step Binomial Tree', fontsize=16)

# Plot the nodes and annotate with stock prices and option values
for j in range(T,-1,-1):
    for i in range(0,j+1):
        x = j  # horizontal position (step number)
        y = j - 2 * i  # vertical position (arbitrarily chosen for visualization
        ax.plot(x, y, 'bo', markersize=10)  # Plot node

        ax.text(x, y, f"$S_{{{i}}}={stockprice[i, j]:.2f}$"
                f"\n$CallOption_{{{i,j}}}={calloption[i, j]:.2f}$"
                f"\n$PutOption_{{{i,j}}}={putoption[i, j]:.2f}$"
                f"\n$PutCallParityCheck_{{{i,j}}}= {bool(putcallparitycheck[i, j
                ha='center', va='center', fontsize=10, bbox=dict(facecolor='whit

# Draw lines between the nodes to represent possible transitions
for j in range(T-1,-1,-1):
    for i in range(0,j+1):
        x = j  # horizontal position (step number)
        y = j - 2 * i  # vertical position (arbitrarily chosen for visualization
        # Line to up move
        ax.plot([x, x + 1], [y, y - 1], 'k-')
        # Line to down move
        ax.plot([x, x + 1], [y, y + 1], 'k-')

ax.axis('off')
plt.tight_layout()
plt.show()
```