The binomial tree model

1 Model theory

The binomial tree is a discrete-time model for the evolution of asset prices. It is an elegant tool to calculate option values and hedge parameters. Here we will discuss the assumptions to the model, provide some intuition to it and derive the main model parameters.

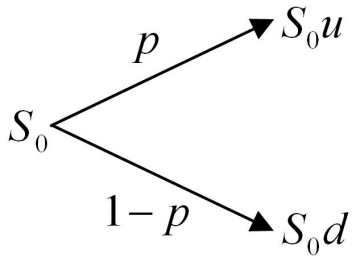


Figure 1: Stock price movements

The binomial tree is a simplified representation of the market. Let S denote the value of a stock. The main concept is that after each time period Δt, the price of the stock can make two movements, as depicted in Figure 1. We refer to these as the *up movement*, in which case $S_{t+\Delta t} = S_t \times u$ or the *down movement*, in which case $S_{t+\Delta t} = S_t \times d$, for two positive numbers $0 < d < u$. Obviously, stock price movements are in reality processes of a much higher complexity. Still, this simple model is useful in practice because it is computationally tractable and a relatively good approximation to more realistic continuous-time models if a sufficient number of periods is used.
Below we summarize the main assumptions of this model:

1. Cash invested in the money-market yields a continuously compounded interest at a constant rate r.

2. A stock price S_1 after a period of time Δt can only have two possible outcomes: $S_0 \cdot u$ or $S_0 \cdot d$ with $0 < d < e^{r\Delta t} < u$.

3. The economy is free of arbitrage.

4. There are no transaction costs for trading shares of stock.

We will use this model to introduce the concept of a risk-neutral price and a replication portfolio.

1.1 Example: Risk-neutral option pricing with a one-period tree

We start with a simple one-period example, taken from [1]. Consider a stock of which the value is denoted S_t. We assume a highly simplified market, captured by a one-period binomial tree. Suppose $S_0 = 4$, $u = 2$, $d = 0.5$ and assume that the accrued interest over Δt is given by

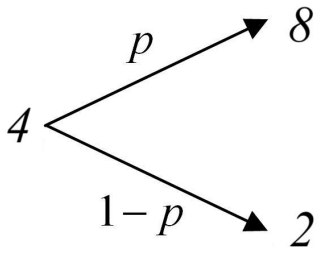


Figure 2: Example of stock price movements

$e^{r\Delta t} = 1.25$. Therefore, the two possible price moves of the stock over Δt are $S_0 \times d = 2$ and $S_0 \times u = 8$ (see Figure 2).
Secondly we consider a European call option written on one share of stock S_t. A European option is a contract which gives the holder the right, but not the obligation, to buy the underlying asset at a specified strike price on a specified future date. Suppose that the expiration of the option is after one period Δt and the strike price is $K = 5$. The central question is now: **What is the fair value f_0 of this option contract at time zero, before the price movement of the stock is known?**
We argue that the answer to this question is $f_0 = 1.20$, using a *no-arbitrage* argument. Suppose that an agent just sold the option contract for the price of 1.20. At $t = 0$ he buys $\Delta = 0.5$ of the stock. That means the agent has a portfolio with 0.5 shares of stock and a cash-position of $f_0 - \Delta \times S_0 = 1.20 - 0.5 \times 4 = -0.80$ (i.e. a debt). Now, at time $t = 1$ due to the accrual of interest, the agent has a debt of $0.8 \times e^{r\Delta t} = 0.8 \times 1.25 = 1$. For the stock value two scenarios can occur:

- **Up movement:** The buyer of the option exercises the contract, pays the strike price to the agent and receives one unit of stock in return. From the agent's perspective this means he has to buy 0.5 shares (in addition to the 0.5 shares he already had in his portfolio) at the market-price of that time. That costs him $(1 - \Delta) \times S_1 = 0.5 \times 8 = 4$ and creates a total debt of $1 + 4 = 5$. This then exactly cancels against the received strike price of $K = 5$. Hence, the agent is left with a position of zero.
- **Down movement:** The buyer of the option does not exercise the contract, because it would yield him a loss. The agent therefore sells his 0.5 shares in the market. That earns him $\Delta \times S_1 = 0.5 \times 2 = 1$, which exactly cancels against his debt of 1. Hence, the agent is again left with a position of zero.

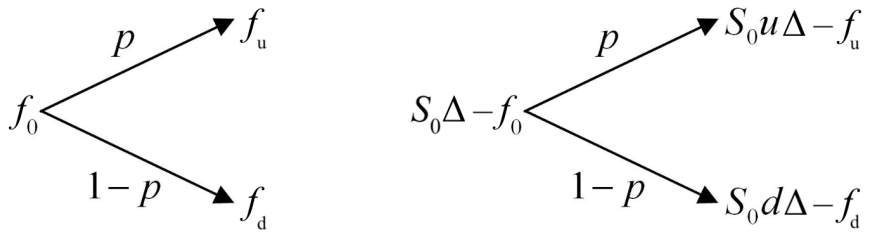
The conclusion is that an agent can set up a portfolio of assets and cash that exactly replicates the value of the option. This is called *hedging*. Should the option be sold at a higher or lower price, it means that someone can make a guaranteed profit, which we refer to as *arbitrage*. $f_0 = 1.20$ is therefore the *no-arbitrage price* or the *risk-neutral value* of the option.

1.2 Derivation binomial tree parameters

We will step by step derive the parameter values that are relevant to the binomial tree model.

1.2.1 The delta-parameter (Δ)

We start with Δ, which represents the amount of shares that need to be bought at $t = 0$ by the agent in order to set up a hedge. Figure 3a depicts the price movements of the risk-neutral



(a) The option value (b) The portfolio value

Figure 3: Price movements in the one-period binomial tree

option value. Figure 3b depicts value movements of the agent's portfolio. At time zero, the agent sells the option contract for the price f_0 and buys Δ shares of the stock at price S_0. After time Δt, the portfolio is worth $S_0u\Delta - f_u$ in case of an up movement and $S_0d\Delta - f_d$ in case of a down movement.
The key insight is that the agent wants to choose Δ such that the final value of his portfolio is independent of the price movement of the stock. The portfolio is then *risk-free*. This is the case if and only if

$$S_0u\Delta - f_u = S_0d\Delta - f_d \tag{1}$$

Solving for Δ yields

$$\boxed{\Delta = \frac{f_u - f_d}{S_0u - S_0d}} \tag{2}$$

1.2.2 The risk-neutral probability (p) and fair value (f_0)

The fair value f_0 can be determined by noting that the agent should have a zero position at maturity of the option as otherwise the agent must have obtained a loss or gain. That would imply an arbitrage since the portfolio is set up to be risk-free. At time zero, the cash-position of the agent is $f_0 - \Delta S_0$. At maturity this cash has accrued interest, by which it has the value $e^{r\Delta t}(f_0 - \Delta S_0)$. His portfolio at maturity is worth $S_0u\Delta - f_u$ in case of an up movement and $S_0d\Delta - f_d$ in case of a down movement, which by the deliberate choice of Δ will be equal. The final position of the agents is zero if his cash-position and portfolio value add up to zero at maturity. Therefore it follows that

$$\begin{aligned} & e^{r\Delta t}(f_0 - \Delta S_0) + (S_0u\Delta - f_u) = 0 \\ \implies & f_0 = S_0\Delta - e^{-r\Delta t}(S_0u\Delta - f_u) \end{aligned}$$

Having derived f_0, we aim to rewrite it in a convenient form by substituting the delta-parameter. Recall that $\Delta = \frac{f_u - f_d}{S_0 u - S_0 d}$. If we substitute that in the expression for f_0, we find:

$$\begin{aligned}
f_0 &= S_0\Delta - e^{-r\Delta t}\left(S_0 u\Delta - f_u\right) \\
&= e^{-r\Delta t}\left(S_0\Delta\left(e^{r\Delta t} - u\right) + f_u\right) \\
&= e^{-r\Delta t}\left(S_0\frac{f_u - f_d}{S_0 u - S_0 d}\left(e^{r\Delta t} - u\right) + f_u\right) \\
&= e^{-r\Delta t}\left(\left(\frac{e^{r\Delta t} - u}{u - d} + 1\right) f_u - \frac{e^{r\Delta t} - u}{u - d} f_d\right) \\
&= e^{-r\Delta t}\left(\frac{e^{r\Delta t} - d}{u - d} f_u + \frac{u - e^{r\Delta t}}{u - d} f_d\right)
\end{aligned}$$

The parameter p is defined as

$$\boxed{p := \frac{e^{r\Delta t} - d}{u - d}} \tag{3}$$

Note that the following relation holds

$$1 - p = \frac{u - d}{u - d} - \frac{e^{r\Delta t} - d}{u - d} = \frac{u - e^{r\Delta t}}{u - d}$$

Therefore we can rewrite the expression for f_0 in the following convenient form

$$\boxed{f_0 = e^{-r\Delta t}\left(p f_u + (1 - p) f_d\right)} \tag{4}$$

We refer to the quantities p and $1 - p$ as the *risk-neutral probabilities* of an up and down movement respectively. By interpreting p as a probability, the fair value of the option can be interpreted as the *expected value* of the option's pay-off. In other words, we can write:

$$f_0 = \mathbb{E}\left[e^{-r\Delta t} f_1\right] \tag{5}$$

Also note that as a consequence, the risk-neutral expectation of S_1 can shown to be equal to

$$\mathbb{E}\left[S_1\right] = p S_0 u + (1 - p) S_0 d \tag{6}$$

$$= \frac{e^{r\Delta t} - d}{u - d} S_0 u + \frac{u - e^{r\Delta t}}{u - d} S_0 d \tag{7}$$

$$= S_0 e^{r\Delta t} \tag{8}$$

1.2.3 The up and down factors (u and d)

We finalise by deriving the u and d factors. These factors should be chosen such that they represent market conform price movements. To do so, the factors are linked to the volatility of the stock price. The volatility σ is a measure for the degree of variation in the stock price over time. The derivation in this section is based on the following two assumptions:

- For small values of Δt, the variance of the stock price change is approximately $S_0^2 \sigma^2 \Delta t$.
- $u \times d = 1$.

Recall that $\mathbb{E}[S_1] = S_0 e^{r\Delta t}$, then according to the definition of the variance we have

$$\begin{aligned} \mathrm{Var}(S_1) &= \mathbb{E}\left[S_1^2\right] - \mathbb{E}[S_1]^2 \\ &= p(S_0 u)^2 + (1-p)(S_0 d)^2 - \left(S_0 e^{r\Delta t}\right)^2 \end{aligned}$$

Now substitute our derived value of p to find

$$\begin{aligned} \mathrm{Var}(S_1) &= S_0^2 \left(\frac{e^{r\Delta t} - d}{u - d} u^2 + \frac{u - e^{r\Delta t}}{u - d} d^2 - e^{2r\Delta t} \right) \\ &= S_0^2 \left(e^{r\Delta t}(u + d) - ud - e^{2r\Delta t} \right) \end{aligned}$$

The first assumption says $\mathrm{Var}(S_1) = S_0^2 \sigma^2 \Delta t$. Divide both sides by S_0^2 and substitute $d = \frac{1}{u}$, which follows from the second assumption. We can then write

$$\begin{aligned} \sigma^2 \Delta t &= e^{r\Delta t}\left(u + \frac{1}{u}\right) - 1 - e^{2r\Delta t} \\ u + \frac{1}{u} &= e^{-r\Delta t}\left(\sigma^2 \Delta t + 1 + e^{2r\Delta t}\right) \\ &= e^{-r\Delta t}\sigma^2 \Delta t + e^{-r\Delta t} + e^{r\Delta t} \end{aligned}$$

Now use Taylor expansions to approximate $e^{-r\Delta t} \approx 1 - r\Delta t$ and $e^{r\Delta t} \approx 1 + r\Delta t$. If we neglect all terms with $(\Delta t)^2$ and higher powers, we have

$$\begin{aligned} u + \frac{1}{u} &\approx (1 - r\Delta t)\sigma^2 \Delta t + (1 - r\Delta t) + (1 + r\Delta t) \\ &\approx \sigma^2 \Delta t + 2 \end{aligned}$$

This we can be rewritten as

$$u^2 - \left(\sigma^2 \Delta t + 2\right) u + 1 = 0$$

which can be solved by using the quadratic formula and again ignoring higher powers of Δt:

$$\begin{aligned} u &\approx \frac{\sigma^2 \Delta t + 2 \pm \sqrt{(\sigma^2 \Delta t + 2)^2 - 4}}{2} \\ &\approx \frac{1}{2}\sigma^2 \Delta t + 1 \pm \sigma\sqrt{\Delta t} \end{aligned}$$

The solution with the minus-sign corresponds to the d parameter $\left(= \frac{1}{u}\right)$, therefore we will discard it for now. As a final step, consider the second-order Taylor expansion of $f(x) = e^{\sigma x}$ around zero. This would yield $f(x) \approx 1 + \sigma x + \frac{1}{2}\sigma^2 x^2$. Therefore it follows that

$$\begin{aligned} u &\approx 1 + \sigma\sqrt{\Delta t} + \frac{1}{2}\sigma^2 \Delta t \\ &\approx f\left(\sqrt{\Delta t}\right) = e^{\sigma\sqrt{\Delta t}} \end{aligned}$$

Hence we conclude:

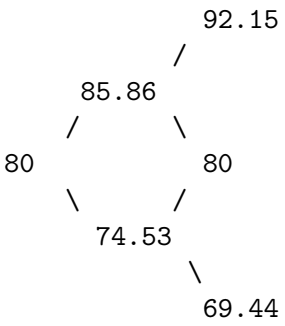
$$\boxed{u = e^{\sigma\sqrt{\Delta t}} \qquad \text{and} \qquad d = e^{-\sigma\sqrt{\Delta t}}} \tag{9}$$

2 Coding instructions

In this section, we do not discuss the theory behind the binomial tree, but provide you with a basic setup to implement your binomial tree in Python. Note that this is a setup, the code as presented here is not yet working. In order to calculate the answers, missing parts need to be completed.

2.1 Running time

You might be tempted to implement this as a tree recursion algorithm, but unfortunately this leads to sub-optimal running times $O(2^N)$. And you will quickly discover that running for larger N becomes intractable.
Therefore, we suggest a different approach. We will calculate the numbers in the binomial tree, but store them as entries of a matrix.
For example, a tree with $N = 2$ looks as follows ($S_0 = 80, \sigma = 0.1, T = 1, N = 2$):



And we can represent the matrix with the values as:

```
[ 80.      0       0    ]
[ 74.53    85.86.  0    ]
[ 69.44.   80.     92.15]
```

Thus, the matrix contains all the required values, but calculating all the values inside the matrix is only of complexity $O(N^2)$. With the generated tree, we calculate the option price.

2.2 A matrix representation

Thus, we will now build the tree as a matrix. The next piece of code is a template structure, it is not yet complete. Please code the missing parts and calculate the stock price.

```
import numpy as np

def buildTree(S, vol, T, N):
  dt = T / N

  matrix = np.zeros((N + 1, N + 1))

  u = 0 # TODO
  d = 0 # TODO

  # Iterate over the lower triangle
```

```
  for i in np.arange(N + 1): # iterate over rows
    for j in np.arange(i + 1): # iterate over columns
      # Hint: express each cell as a combination of up
      # and down moves
      matrix[i, j] = 0 # TODO

  return matrix
```

We can execute the code as follows:

```
sigma = 0.1
S = 80
T = 1.
N = 2

buildTree(S, sigma, T, N)
```

2.3 Calculating the option value

Now that we have the pricing tree, we can use this as input to compute the option value. The next piece of code is a template structure, it is not yet complete. Please code the missing parts and calculate the option price.

```
def valueOptionMatrix(tree, T, r, K, vol):

  dt = T / N

  u = 0 # TODO
  d = 0 # TODO

  p = 0 # TODO

  columns = tree.shape[1]
  rows = tree.shape[0]

  # Walk backward, we start in last row of the matrix

  # Add the payoff function in the last row
  for c in np.arange(columns):
    S = tree[rows - 1, c] # value in the matrix
    tree[rows - 1, c] = 0 # TODO

  # For all other rows, we need to combine from previous rows
  # We walk backwards, from the last row to the first row
  for i in np.arange(rows - 1)[::-1]:
    for j in np.arange(i + 1):
      down = tree[i + 1, j]
      up = tree[i + 1, j + 1]
      tree[i, j] = 0 # TODO
  return tree
```

We can execute the function as follows:

```
sigma = 0.1
S = 80
T = 1.
N = 2

K = 85
r = 0.1

tree = buildTree(S, sigma, T, N)
valueOptionMatrix(tree, T, r, K, sigma)
```

2.4 Plotting

We want to study the correctness of the implementation, thus we want to compare the solution of our algorithm with the analytical answer.
Please create a plot that has on the X-axis the number of steps in the tree (depth), and on the Y-axis the error with respect to the analytical solution.

```
# Play around with different ranges of N and step sizes.
N = np.arange(1,300)

# Calculate the option price for the correct parameters
optionPriceAnalytical = 0 # TODO

# calculate option price for each n in N
for n in N:
    treeN = buildTree(...) # TODO
    priceApproximatedly = valueOption(...) # TODO


# use matplotlib to plot the analytical value
# and the approximated value for each n
```

References

[1] S. Shreve, *Stochastic calculus for finance I: the binomial asset pricing model*. Springer Science & Business Media, 2005.