Binomial Option Pricing using the Cox-Ross-Rubinstein Model

Gabriele Alberto

1 Introduction

The purpose of this project is to implement an option pricing model based on the binomial framework introduced by Cox, Ross, and Rubinstein (1979). The program prices European and American call and put options using real market data retrieved through the `yfinance` API.

Particular attention is given to computational efficiency. Instead of constructing the full binomial tree, the implementation relies on vectorized operations and stores only one-dimensional arrays.

The model has been tested on equity data and is currently designed to operate on stocks rather than commodities. While the pricing framework could be extended to other asset classes, commodities typically involve additional factors such as storage costs or convenience yields, which are not explicitly modeled in this implementation.

2 Data Collection and Volatility Estimation

The underlying asset selected for this project is NVIDIA (NVDA). Five years of adjusted daily closing prices are downloaded in order to compute a reliable estimate of historical volatility.

Logarithmic returns are calculated as:

$$r_t = \ln\left(\frac{S_t}{S_{t-1}}\right)$$

where S_t denotes the asset price at time t.

Volatility is then annualized assuming 252 trading days:

$$\sigma = \mathrm{std}(r_t) \times \sqrt{252}$$

This value is treated as constant throughout the life of the option.

It is important to note that market data is downloaded twice within the program.

Adjusted prices are used for volatility estimation because they incorporate corporate actions such as stock splits and dividend payments, resulting in a smoother and more consistent return series.

However, the spot price is obtained from non-adjusted data to reflect the actual traded market value. Using adjusted prices for the current price could introduce small distortions in the payoff calculation.

3 Model Setup

The binomial model assumes that over a short time interval Δt, the asset price can move either upward or downward. The up and down factors are defined as:

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

This specification ensures that the tree recombines, meaning that different paths can lead to the same future price.

3.1 Risk-Neutral Probability

Under the risk-neutral framework, the expected return of the asset equals the risk-free rate adjusted for dividends. The probability of an upward movement is therefore:

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

while the probability of a downward movement is simply $1 - p$.

Future payoffs are discounted using the factor:

$$e^{-r\Delta t}$$

4 User-Defined Parameters

The model allows several inputs to be modified by the user depending on the pricing scenario.

`Variable`	**Meaning**	**Description**
`ticker`	Underlying asset	Stock used for pricing. Market data is downloaded automatically.
`s0`	Spot price	Current market price retrieved from recent non-adjusted data.
`k`	Strike price	Exercise price of the option.
`n`	Number of steps	Controls the granularity of the tree. Higher values generally improve accuracy.
`t`	Time to maturity	Expressed in days and converted internally into years.
`r`	Risk-free rate	Used for discounting and probability calculations.
`q`	Dividend yield	Continuous dividend rate applied to the underlying asset.
`opttype`	Option type	Specifies whether the option is a call or a put.
`option_style`	Exercise style	Determines whether the option is European or American.

These parameters make the model flexible and allow it to adapt to different market conditions.

5 Terminal Payoff

At maturity, the value of the option depends solely on the final asset price.

For a call option:

$$C_T = \max(S_T - K, 0)$$

For a put option:

$$P_T = \max(K - S_T, 0)$$

Rather than storing the entire price tree, the program directly computes the $n+1$ terminal nodes:

$$S_T(j) = S_0\, u^{n-j} d^j$$

where j represents the number of downward movements.

6 Backward Induction

Once terminal payoffs are known, the option price is obtained by iterating backward through the tree.

At each node, the option value equals the discounted expected value of future states:

$$V = e^{-r\Delta t}\left(pV_{up} + (1-p)V_{down}\right)$$

This process continues until the initial node is reached, producing the present value of the option.

7 American vs European Options

European options can only be exercised at maturity, so the continuation value is always used during backward induction.

American options allow early exercise. For this reason, the algorithm compares the continuation value with the immediate payoff:

$$V = \max(\text{continuation}, \text{exercise})$$

Stock prices required for this comparison are computed on the fly, avoiding the need to store the entire tree.

8 Computational Efficiency

The first version of the program constructed the binomial tree using nested `for` loops and stored the entire price matrix. While straightforward, this approach resulted in quadratic memory usage and unnecessary computational overhead.

The implementation was subsequently improved by introducing vectorized operations and storing only the nodes required at each step of the backward induction. This redesign reduced memory complexity from:

$$O(n^2) \to O(n)$$

and significantly enhanced the overall performance of the algorithm.

9 Program Output

After running the script, the program prints a concise summary of the pricing scenario.

The output includes:

- The option style (European or American).
- The current underlying price and the currency.
- The dividend yield used in the model.
- The option type (call or put).
- The strike price.
- The computed option price.

For European options, the program also performs a put-call parity check. This acts as a consistency test for the numerical implementation: if the relationship holds within a small tolerance, it provides additional confidence in the correctness of the pricing algorithm.

10 Limitations

One of the main limitations concerns the treatment of dividends. In this implementation, dividends are modeled as a continuous yield. While this assumption greatly simplifies the mathematical framework, real dividends are paid at discrete dates.

When a stock goes ex-dividend, its price typically drops by approximately the dividend amount. A continuous yield cannot perfectly capture this price jump, which may introduce pricing inaccuracies, particularly for American options where early exercise decisions are strongly influenced by dividend payments.

A more advanced approach would incorporate known dividend dates directly into the tree by adjusting the stock price at each payment. However, this would significantly increase the complexity of the model.

Despite this limitation, the binomial framework remains a reliable approximation and an effective tool for understanding option pricing mechanics.