IB9JHO Programming Assignment

2090341

May 27, 2021

Introduction

The binomial option pricing model was proposed by Sharpe in 1978 but later popularised by Cox, Ross, and Rubinstein in 1979. This is a powerful tool for valuing options especially when there is no analytical closed form solution for the security when working in continuous time.

In this project we will first recreate and investigate the binomial option pricing model for pricing simple European call and put options using C++. We will then modify our methods for evaluating European options so that we can also price American options. The key difference is that American options can be exercised early, unlike European options which cannot be exercised before the maturity time. We will then investigate how our numerical binomial methods compare to that of the analytical Black-Scholes solutions and the implications and drawbacks of the Black-Scholes model.

Option Pricing Using Binomial Trees

The Binomial Option Pricing Model is a risk-neutral method for valuing path-dependent options. This model recreates the movement of the underlying asset and option in discrete-time using a binomial lattice. The end nodes of the tree represent the specific pay-off for the option we are trying to value at maturity time T.

For a European call option the pay-off at maturity is

$$Max(S_T - K, 0) \tag{1}$$

For a European put option the pay-off at maturity is

$$Max(K - S_T, 0) \tag{2}$$

Where S_T is the value of the underlying asset at maturity time T and K is the strike price of the option. We can then illustrate the discrete paths of the lifetime of the option and underlying stock price for a simple one-period option until maturity time T, where the tree terminates. More time steps can be added to this model to create a n-period model. For a two-period model we sample the value of the stock price twice i.e. at say $\frac{T}{2}$ and at maturity time T. The one and two-period binomial trees are illustrated as follows.

Figure 1: One-Period Binomial Tree For a European Call Option

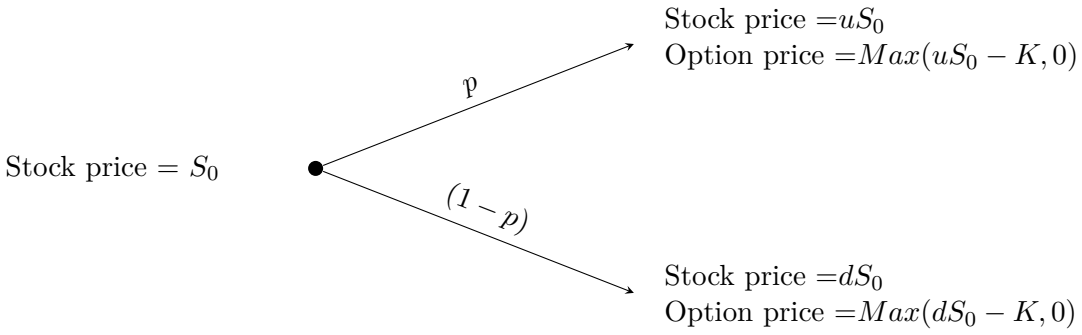
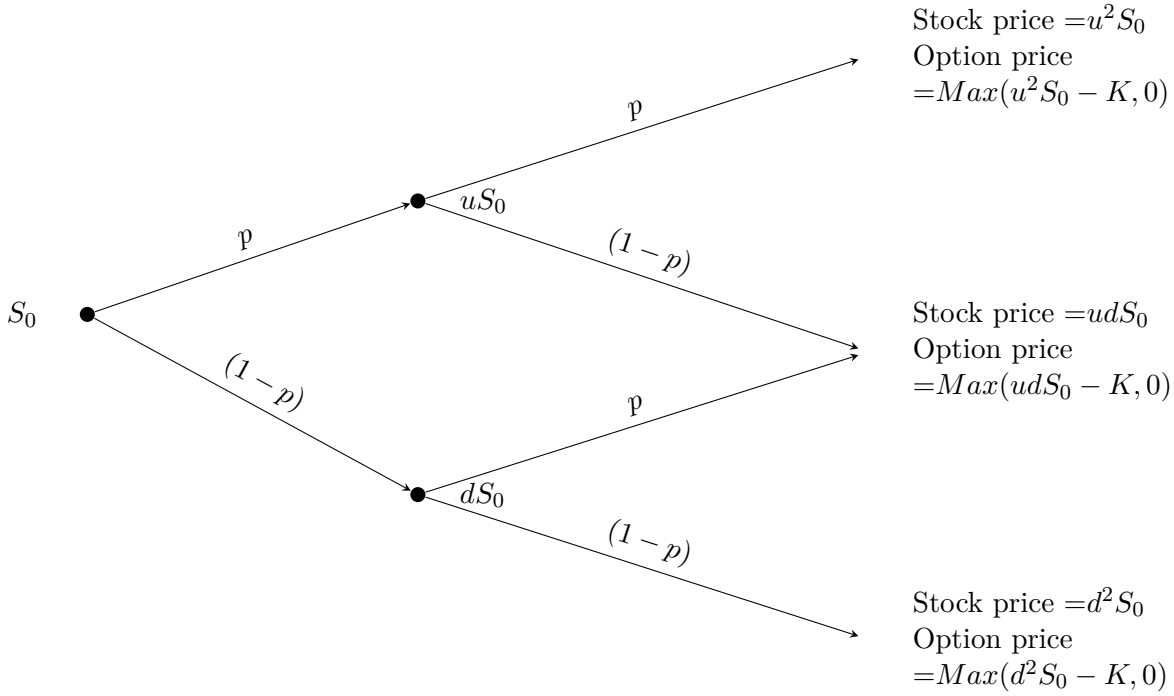


Figure 2: Two-Period Binomial Tree For a European Call Option



Here we define S_0 as the price of the stock today, u as being the growth factor by which the stock price increases in the event of a up movement in the price of the stock, with d being the decay factor by which the stock price decreases in the even of a down movement in the price of the stock, p is the probability of an up movement in the stock at each time interval.

The Cox, Ross and Rubenstein (CRR) method ensures that the tree is recombinant. This means that the stock price moves up and then moves down in the next time period then it will be the same price as if it moves down and then moves up. Under the CRR method:

$$u = e^{\sigma\sqrt{\Delta t}} \tag{3}$$

$$d = \frac{1}{u} \tag{4}$$

In order to find the fair price of the option today we must move incrementally through the tree. At each node in the tree we find the value of the option contract using a risk neutrality assumption. For this assumption we define the probability of an up movement p as.

$$p = \frac{e^{(r-q)\Delta t} - d}{u - d} \tag{5}$$

Where r is the is the risk free rate, q is the dividend yield of the stock, and Δt is the time elapsed between nodes in the tree.

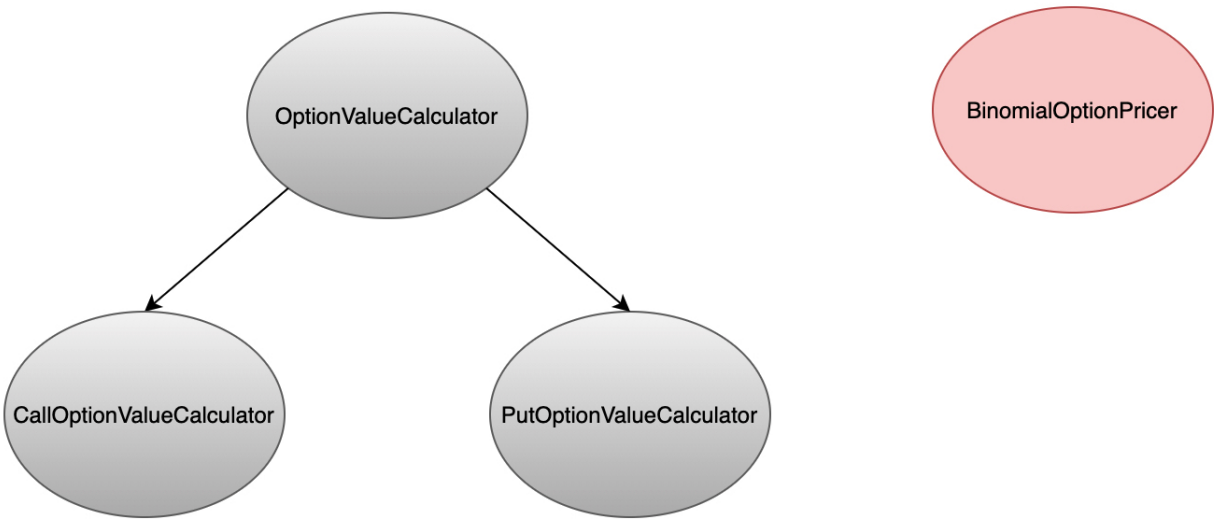
Methodology

Project Structure

In this project we will implement the binomial method for European and American call and put options using two different algorithms which yield the same result. Additionally, we will verify that our calculated solution approaches the analytical solutions for European options given by the Black-Scholes equation.

Our class structure for carrying out the analysis in C++ is as follows

Figure 3: Project Class Structure



The OptionValueCalculatorClass is an abstract class with a virtual member function that calculates the value of a vanilla option at expiry time given the price of the underlying asset at expiry time. CallOptionValueCalculator overrides the virtual member function in OptionValueCalculator so that the function values call options. The strike price for the option is stored in the object. PutOption-ValueCalculator overrides the virtual member function in OptionValueCalculator so that the function values put options. The strike price for the option is stored in the object. The arrows in this diagram indicate inheritance. BinomialOptionPricer contains all the implementation of our algorithms for valuing options.

To build our program we first created the OptionValueCalculator header and source files and stored the strike price, that will be entered as part of the constructor, as a member variable. We then add a value function in the Call/PutOptionValueCalculator source files which implements the appropriate pay-off functions. In the BinomialOptionPricer files we define a constructor that takes the arguments for the max depth we want our object to have, the growth factor and the growth probability, we then store these as member variables within the object. When creating the pricing algorithm functions one of the arguments involves us passing the OptionValueCalculator object by reference so that we can use the option valuation formulae for the corresponding call or put option.

Forward Recursion

Firstly we will implement the binomial method using forward recursion. This method starts from time-step zero and accumulates the option value from the next time-step using the recurrence relation:

$$V_i^j = e^{-r\Delta t}(pV_{i+1}^{j+1} + (1-p)V_{i+1}^j) \tag{6}$$

Where V_i^j is the option value, r is the risk-free rate, Δt is the time increment between each time-step, p is the risk-neutral probability of an upwards jump, i, is the tree depth of the current node, and j is the height of the current node. We will utilise recursion to calculate the value of the option from the first time-step at the beginning of the tree, with the recursion terminating at last time-step. At the last time-step we can calculate the value of the option using the pay-off functions defined previously for calls and puts. To expand this method for American options we need to add the condition that the option can be exercised early, hence we need to calculate the exercise value E_i^j at each node.

The exercise value for call options is

$$E_i^j = Max(S_0 u^{2j-i} - K), 0) \tag{7}$$

The exercise value for put options is

$$E_i^j = Max(K - S_0 u^{2j-i}, 0) \tag{8}$$

So now the recurrence equation now becomes

$$V_i^j = Max(e^{-r\Delta t}(pV_{i+1}^{j+1} + (1-p)V_{i+1}^j, E_i^j)) \tag{9}$$

Backward Induction

The backward induction method starts from the last time-step and computes the option value at all the base nodes using equations (1) and (2) and works backwards through the tree. To work backwards through the tree we would first calculate the expected value of the option at the nodes one time step before maturity by multiplying the value of the option during an up movement from this node by the probability of an up movement and adding this to the value of the option during a down movement from this node multiplied by the probability of a downward movement, and then discounting this by the risk free rate.

$$\text{Binomial Value } = [p \times \text{ Option up } + (1-p) \times \text{ Option down}] \times \exp(-r \times \Delta t) \tag{10}$$

which is the same calculation that we see in equation (6). We then repeat this for each node and at each time step working backwards through the tree until we get the price of the option today.

For American options, the methodology is very similar however at each node working backwards through the tree we have the option to exercise the option early thus the value of the option at each node in the tree is calculated the same way as we did in equation (9). The price of an European Call option written for a stock that does not pay dividends is always higher than its intrinsic value. Therefore, in that case, the prices of European and American call options are equal.

Black-Scholes

In order to compare the accuracy of our binomial tree algorithms we will compare the values of European call and put options obtained by the algorithms to the price of the same options obtained from the Black-Scholes. The Black-Scholes gives an analytical solution to compute the price of European call and put options. The Black-Scholes formulae for call and put options are as follows.

$$C(S_t, t) = N(d_1)S_t - N(d_2)Ke^{-r(T-t)} \tag{11}$$

$$P(S_t, t) = N(-d_2)Ke^{-r(T-t)} - N(-d_1)S_t \tag{12}$$

where

$$d_1 = \frac{1}{\sigma\sqrt{T-t}}\left[ln\left(\frac{S_t}{K}\right) + (r + \frac{\sigma^2}{2})(T-t)\right] \tag{13}$$

$$d_2 = d_1 - \sigma\sqrt{T-t} \tag{14}$$

$C(S_t, t)$ is the price of a European call option, $P(S_t, t)$ is the price of a European put option, S_t is the price of the underlying asset at time t, K is the strike price of the option, r is the annualised risk-free interest rate, σ is the volatility of the underlying asset, T is the option expiry time, $N(x)$ is the standard normal cumulative distribution function.

An issue with using the Black-Scholes is that its PDE does not have a closed form solution when trying to value American options so we cannot use it compare with our American option pricing algorithms. However an American call with no dividends is equal to the price of the equivalent European call option.

Inference

First we will investigate how our backward induction method performs followed by how our forward recursion model performs in comparison when pricing the same options for various tree depths. We will price European and American call options with current stock price, $S_0 = 90$, $K = 100$, $r = 0.05$, $\sigma = 0.45$, $T = 1$ year, a tree depth of n, $\Delta t = \frac{T}{n}$, $u = e^{\sigma\sqrt{\Delta t}}$, $d = \frac{1}{u}$, $p = \frac{e^{(r\Delta t)} - d}{u - d}$

We run our backward induction method for 44 different tree depths ranging from 1 to 9999. The prices of the European and American call and put options can be found in Table 1. Table 2 contains the execution time for the backward induction algorithm for a tree depth of n. The Black-Scholes price for the equivalent European call is 14.0388 and for the European put is 19.1617. Figure 4 and 5 illustrate the convergence of the European Call and Put Option prices of our backward induction method as n increases. Figure 6 illustrates the difference between the prices of the European and American put options.

Table 1: Backward Induction Algorithm Option Prices

Tree Depth	**Euro Call Price**	**Am Call Price**	**Euro Put Price**	**Am Put Price**
1	17.3964	17.3964	22.5193	22.5193
2	14.1177	14.1177	19.2406	20.5404
3	14.8907	14.8907	20.0136	20.4671
4	14.3967	14.3967	19.5196	20.5068
5	14.3693	14.3693	19.4923	20.1408
6	14.415	14.415	19.5379	20.3656
7	14.1464	14.1464	19.2693	19.9597
8	14.3912	14.3912	19.5142	20.3208
9	14.023	14.023	19.146	19.89
10	14.3597	14.3597	19.4827	20.2717
15	13.8511	13.8511	18.974	19.8008
20	14.2274	14.2274	19.3503	20.1013
25	13.9668	13.9668	19.0897	19.9174
30	14.1441	14.1441	19.267	20.0192
35	14.0582	14.0582	19.1812	19.9742
40	14.0873	14.0873	19.2103	19.9691
45	14.0887	14.0887	19.2117	19.9887
50	14.0455	14.0455	19.1685	19.9359
55	14.0967	14.0967	19.2197	19.988
60	14.0131	14.0131	19.1361	19.9119
65	14.095	14.095	19.218	19.9822
70	13.9871	13.9871	19.11	19.8967
75	14.0889	14.0889	19.2119	19.9743
100	14.0413	14.0413	19.1642	19.9391
125	14.0445	14.0445	19.1675	19.9338
150	14.0629	14.0629	19.1858	19.9503
175	14.0228	14.0228	19.1458	19.9203
200	14.0538	14.0538	19.1767	19.9405
250	14.038	14.038	19.161	19.9275
300	14.0274	14.0274	19.1504	19.9211
400	14.0461	14.0461	19.1691	19.934
500	14.0454	14.0454	19.1684	19.9325
650	14.033	14.033	19.1559	19.923
800	14.0421	14.0421	19.165	19.9297
1000	14.0413	14.0413	19.1642	19.9286
1250	14.0384	14.0384	19.1614	19.9266
1500	14.0413	14.0413	19.1642	19.9284
2000	14.0396	14.0396	19.1625	19.9271
2500	14.0387	14.0387	19.1616	19.9262
3000	14.04	14.04	19.1629	19.9272
4000	14.0397	14.0397	19.1626	19.9269
5000	14.0392	14.0392	19.1622	19.9265
7500	14.0387	14.0387	19.1617	19.9261
9999	14.0389	14.0389	19.1618	19.9261

Table 2: Backward Induction Algorithm Execution Times (seconds)

Tree Depth	Euro Call Time	Am Call Time	Euro Put Time	Am Put Time
1	1.91E-06	1.34E-06	9.95E-07	9.94E-07
2	5.07E-06	1.94E-06	1.68E-06	1.76E-06
3	3.14E-06	1.55E-06	1.19E-06	1.47E-06
4	4.15E-06	2.95E-06	2.37E-06	2.95E-06
5	2.36E-06	2.14E-06	1.57E-06	2.10E-06
6	3.35E-06	2.39E-06	1.78E-06	2.41E-06
7	3.55E-06	2.66E-06	1.64E-06	2.41E-06
8	2.04E-06	2.83E-06	1.68E-06	2.91E-06
9	4.25E-06	3.45E-06	2.23E-06	3.55E-06
10	3.18E-06	3.72E-06	2.40E-06	3.74E-06
15	6.24E-06	6.00E-06	3.47E-06	6.48E-06
20	5.11E-06	9.10E-06	4.39E-06	9.43E-06
25	7.39E-06	1.27E-05	5.83E-06	1.37E-05
30	7.72E-06	1.67E-05	7.30E-06	1.82E-05
35	1.01E-05	2.14E-05	8.96E-06	2.39E-05
40	2.01E-05	2.67E-05	1.07E-05	3.04E-05
45	1.31E-05	3.29E-05	1.24E-05	3.67E-05
50	1.55E-05	3.90E-05	1.46E-05	4.45E-05
55	1.69E-05	4.65E-05	1.65E-05	5.29E-05
60	2.34E-05	5.48E-05	1.96E-05	6.29E-05
65	2.29E-05	6.42E-05	2.26E-05	7.35E-05
70	3.08E-05	7.30E-05	2.58E-05	8.45E-05
75	2.81E-05	8.22E-05	2.78E-05	9.28E-05
100	4.56E-05	0.000144	4.50E-05	0.00015894
125	7.05E-05	0.00022003	6.86E-05	0.00024215
150	0.00010545	0.00031148	9.47E-05	0.00035981
175	0.00012785	0.00042697	0.00012799	0.00049291
200	0.00017728	0.0005507	0.00016065	0.00060387
250	0.00025068	0.00079533	0.00023582	0.00105988
300	0.00063998	0.00169151	0.00048688	0.00228967
400	0.00110571	0.00331332	0.00093758	0.00410281
500	0.00132517	0.00406436	0.00107902	0.00576849
650	0.00245631	0.00806726	0.00222503	0.0117642
800	0.00436195	0.0140395	0.00420563	0.0172727
1000	0.0068729	0.022628	0.00695226	0.024402
1250	0.00917582	0.035377	0.00973009	0.0438503
1500	0.0184574	0.0455895	0.014474	0.0501885
2000	0.0200671	0.0862826	0.0251164	0.0870476
2500	0.0315998	0.144781	0.0346462	0.166404
3000	0.058408	0.180133	0.0482846	0.219348
4000	0.0989094	0.237524	0.100132	0.332074
5000	0.120725	0.371941	0.160895	0.554263
7500	0.26436	0.881959	0.413186	1.20667
9999	0.452108	1.49202	0.823014	2.09846

Figure 4: Backward Induction European Call Option Prices

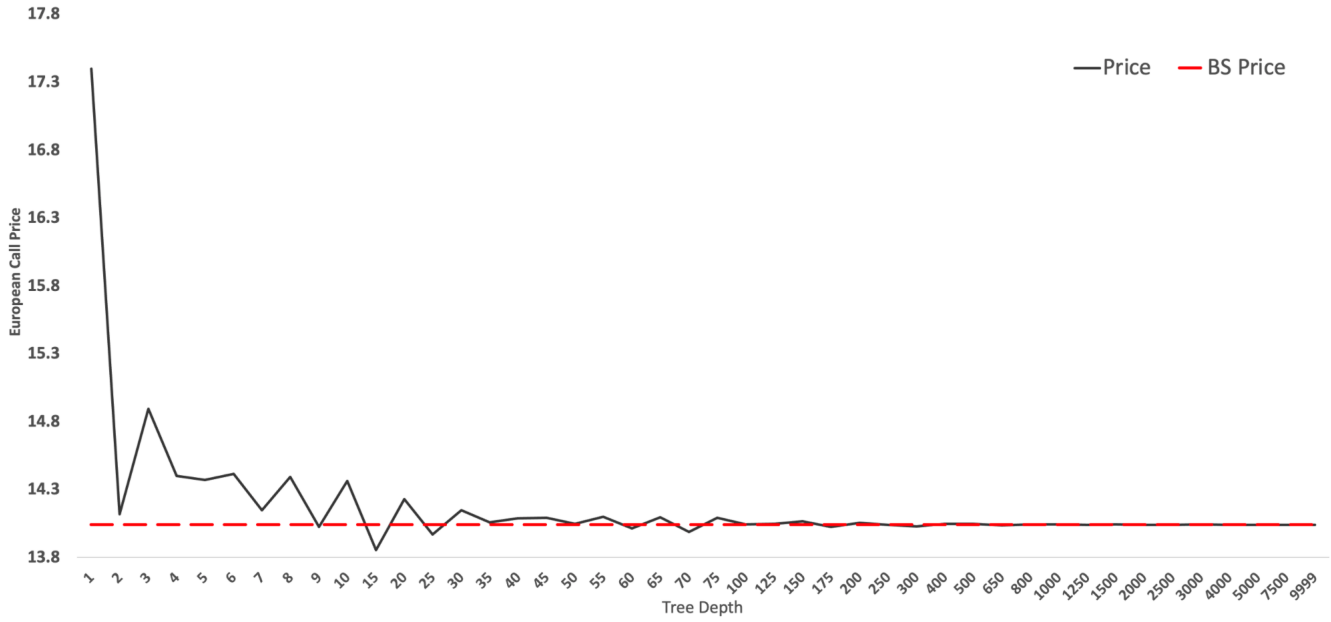


Figure 5: Backward Induction European Put Option Prices

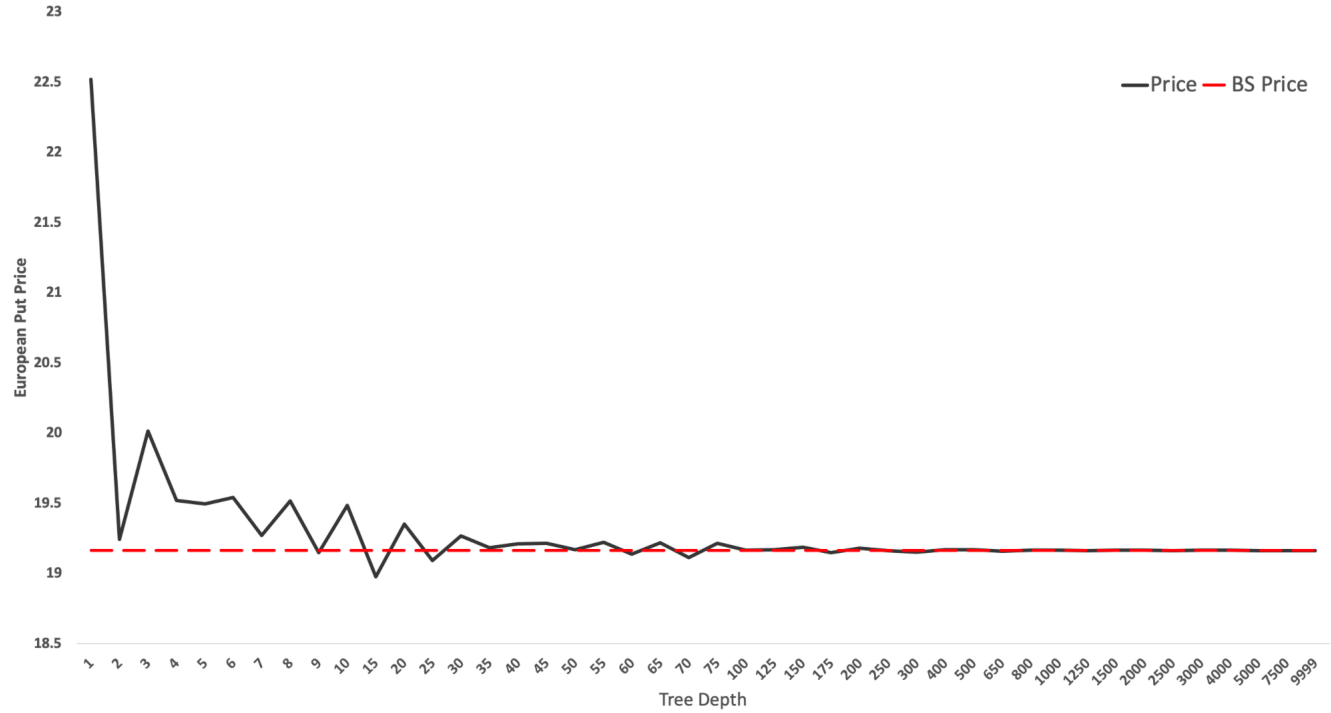
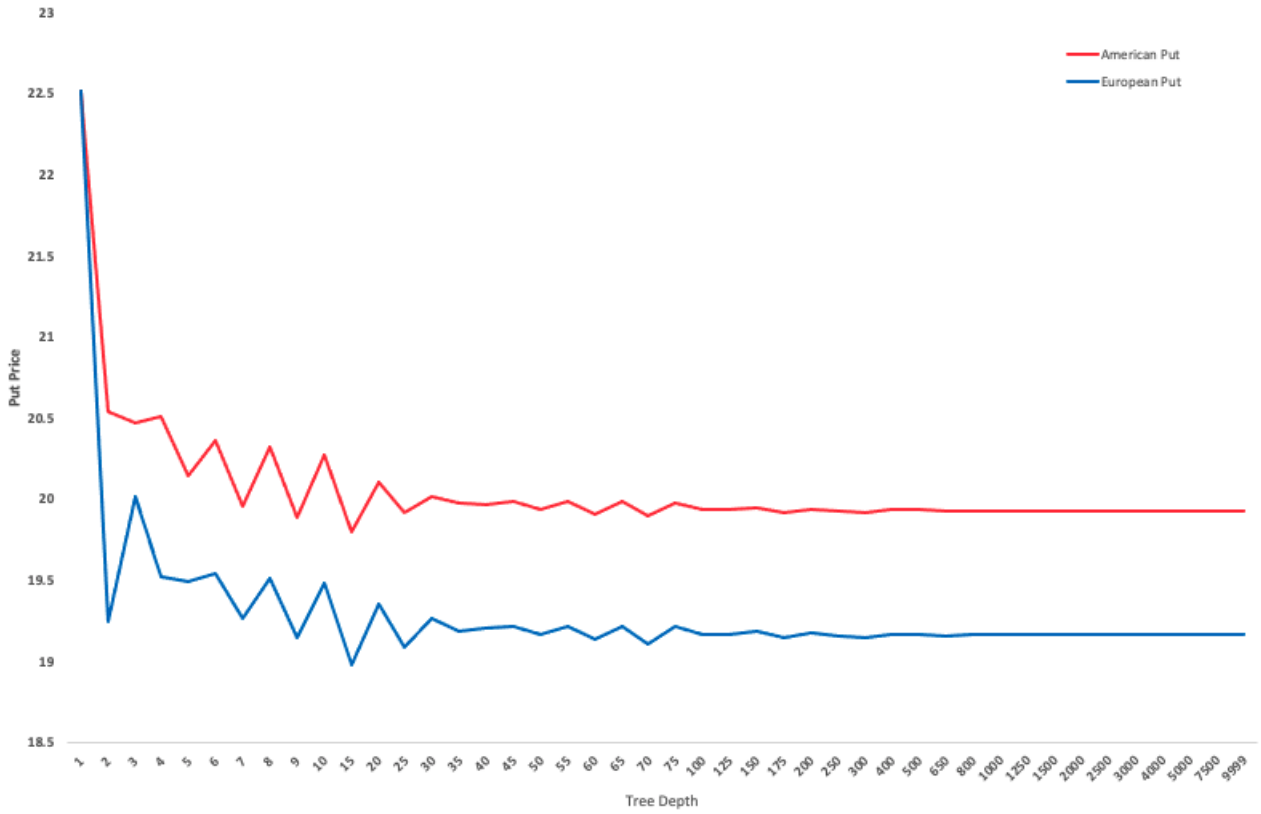


Figure 6: Backward Induction American and European Put Option Prices



Looking at Table 1 we can see how the option price converges as the tree depth gets larger. The European option prices converge towards the Black-Scholes value for the same option. From Table 1 we note how the price of the European call is equal to that of the American call as expected as the option does not pay dividends. From the tables and figures of the European option prices the prices converge to within a few cents of the Black-Scholes price when the tree depth is approximately greater than 100. With there being large fluctuations in the price for tree depths less than 30.

From Table 2 we note how the algorithm is extremely fast even with for a very large tree depth, which requires many calculations in order to work backwards through a very large tree. A tree depth of 9999 only took 2.098 seconds to calculate the American put option value.

We note how the price of the American put is greater than the price of the European put due to the opportunity to exercise the option early. We also note how the execution times for the American options in general took longer than that for the European options. This is due to the fact that at every node in the tree the algorithm must check for early exercise which is adds some extra complexity to the calculation and thus takes more computing time.

Next we will focus on the forward recursion algorithm. When testing this algorithm it was clear that it was far less efficient compared to the backward induction method, despite it obtaining the same prices. This algorithm began to take over a minute to obtain the option price over a tree depth of 30 so running the algorithm for values much greater than this would have been pointless. Table 3 contains the prices of the European and American call and put options calculated with this method. Table 4 contains the time taken in seconds for the option price to be calculated.

Table 3: Forward Recursion Option Prices

Tree Depth	Euro Call Price	Am Call Price	Euro Put Price	Am Put Price
1	17.3964	17.3964	22.5193	22.5193
2	14.1177	14.1177	19.2406	20.5404
3	14.8907	14.8907	20.0136	20.4671
4	14.3967	14.3967	19.5196	20.5068
5	14.3693	14.3693	19.4923	20.1408
6	14.415	14.415	19.5379	20.3656
7	14.1464	14.1464	19.2693	19.9597
8	14.3912	14.3912	19.5142	20.3208
9	14.023	14.023	19.146	19.89
10	14.3597	14.3597	19.4827	20.2717
15	13.8511	13.8511	18.974	19.8008
20	14.2274	14.2274	19.3503	20.1013
25	13.9668	13.9668	19.0897	19.9174
26	14.1732	14.1732	19.2962	20.0475
27	13.994	13.994	19.1169	19.9342
28	14.1581	14.1581	19.281	20.0322
29	14.0156	14.0156	19.1386	19.9477
30	14.1441	14.1441	19.267	20.0192
31	14.033	14.033	19.1559	19.9588
32	14.1311	14.1311	19.2541	20.0077

Table 4: Forward Recursion Execution Times(seconds)

Tree Depth	Euro Call Time	Am Call Time	Euro Put Time	Am Put Time
1	2.60E-07	2.58E-07	2.01E-07	2.03E-07
2	4.88E-07	4.31E-07	3.02E-07	3.68E-07
3	4.41E-07	5.59E-07	3.79E-07	5.78E-07
4	1.37E-06	1.55E-06	1.20E-06	1.61E-06
5	1.63E-05	2.03E-06	1.35E-06	1.95E-06
6	2.49E-06	3.38E-06	2.24E-06	3.45E-06
7	1.11E-05	1.40E-05	9.90E-06	1.41E-05
8	8.37E-06	1.21E-05	7.98E-06	1.20E-05
9	3.73E-05	2.26E-05	3.65E-05	2.29E-05
10	3.00E-05	0.00013424	6.20E-05	4.47E-05
15	0.00108824	0.00156231	0.00094056	0.00190535
20	0.0332176	0.0597013	0.0415951	0.0533233
25	1.03067	1.46953	0.958277	1.51133
26	2.05074	2.99012	1.98886	2.97137
27	4.03104	6.12382	4.00935	5.9602
28	7.90647	11.822	8.04586	11.7432
29	15.6822	23.2501	15.751	23.0129
30	31.4653	53.0351	37.1831	48.6253
31	63.1329	101.34	62.0927	90.0341
32	121.545	173.081	158.914	170.187

We can see from Table 3 that the prices do begin to converge similarly to that of the backwards induction method to the Black-Scholes price as the depth increases, but Table 4 tells us that for a relatively small depth of 32 steps that it takes over two minutes for the option to be priced. However for values where the tree depth is the same there is no difference in the prices given by the two algorithms as they are essentially doing the same thing.

The reason for the poor performance of the recursion algorithm is simply due to the fact it is using recursion. While the backward induction method creates an array to store the option and stock values in arrays using dynamic memory allocation as we work backwards through the tree. As the tree depth increases the complexity of the recursion algorithm increases as there are more nodes to calculate the stock and option prices for.

To improve the efficiency of the recursion algorithm it may be possible the add a helper function to aid with the recursion which would decrease the execution time as the tree depth gets larger. However even with a helper function it still may not be more efficient than the backward induction algorithm.

Conclusion

This project has shown us that algorithms that use dynamic memory allocation and vectors to store values can be extremely fast when solving problems that require many calculations, compared to using recursion. This emphasises the importance that there may be many ways to tackle a problem but depending on the use case one would have to consider the speed, accuracy, and ease of understanding of the methods being implemented. It has also shown that using classes and object orientated programming allows us to create a general framework for testing and running our algorithms, which further down the line can be changed and expanded upon without having to start from scratch. We can just simply add a new pricing function to the BinomialOptionPricing header and source files and continue as before with our current framework or even add a new class to the OptionValueCalculator family that has a new pay-off function. Specifically the project has shown us that a relatively easy method to understand how to price options can converge to the Black-Scholes price which is derived using complicated partial differential equations. However the Black-Scholes is not perfect and using it to price American options becomes requires additional and more complex work. Pricing options as we did in this project may be suitable for investors where time and accuracy may not be important, but different methods would need to be used for say, high frequency market makers where speed and accurate pricing down to the levels of individual basis points are essential.

One could expand this exercise and investigate trinomial trees, where at each time step there are three possible paths for the stock rather than just two. Trinomial methods can produce more accurate results than the binomial method if we are limiting the number of time-steps to be fairly small. Of course even trinomial trees are not realistic in practice as we know theoretically at any time point a stock can have any positive price rather than just three. We could also expand the study into other exotic types of options also such as Asian options where the pay-off depends on the average price of the underlying asset over a period of time. As a comparison we could implement Monte Carlo methods as an alternative to binomial trees to price options like those in this project and see how they compare.