MATH70119 Numerical Methods in Finance Assessed Coursework

Harrison Lam CID: 01849877
Dominic Rushton CID: 06019954

March 11, 2025

Question 1

1.1a.

To price an option with payoff $g(S_T)$, we discretise in time, and use a lattice structure to compute the time-t price $V_t(S_t)$ for all $t \in \{0, \Delta t, 2\Delta t, ..., T\}$, where $\Delta t = \frac{T}{N}$, using backward induction. Using the Kamrad–Ritchken parametrisation, for fixed $t = n\Delta t$, the stock price takes a finite number of values $\{s_k^n = S_0 u^{n-k} : k \in \{0, 1, ..., 2n\}\}$. u is computed using the given formula and we use V_k^n to denote $V_{n\Delta t}(s_k^n)$. We first compute $V_k^N = g(s_k^N)$, then we iteratively compute V_k^n using

$$V_k^n = e^{-r\Delta t}[q_u V_k^{n+1} + q_m V_{k+1}^{n+1} + q_d V_{k+2}^{n+1}], \quad k \in \{0, 1, , ..., 2n\}$$

until $n = 0$ is reached. The time-0 option value is then given by V_0^0. The process is outlined below in pseudocode.

$k \leftarrow 0$
while $k \leq 2N$ **do**
  $V_k^N \leftarrow g(s_k^N)$
  $k \leftarrow k + 1$
end while
$n \leftarrow N - 1$
while $n \geq 0$ **do**
  $k \leftarrow 0$
  while $k \leq 2n$ **do**
    $V_k^n \leftarrow e^{-r\Delta t}[q_u V_k^{n+1} + q_m V_{k+1}^{n+1} + q_d V_{k+2}^{n+1}]$
    $k \leftarrow k + 1$
  end while
  $n \leftarrow n - 1$
end while

1.1b.

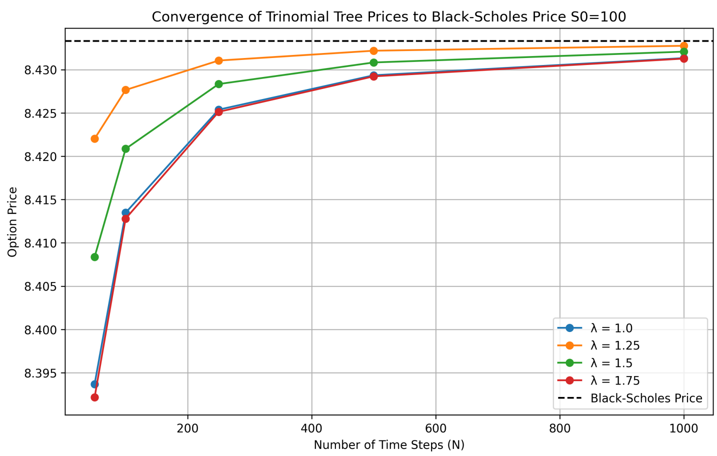

Figure 1: Price Estimation from Tree Discretisation ($S_0 = 100$)

It is observed that the error is monotonically decreasing for N larger than 50, for all values of λ. In particular, the tree pricer obtains an absolute error of < 0.005 for $N = 500$ for all λ

and consistently obtains the lowest error when $\lambda = 1.25$. We also note that our Trinomial Tree model consistently underestimates the analytical Black Scholes option prices. This is likely due to the fact the Tree approximation method only equates the first two moments of the continuous distribution at each time step, and so the discretisation fails to completely capture the right-tail of the stock price distribution (which contributes extra value in the case of a call option), leading to lower prices, particularly for larger time steps.

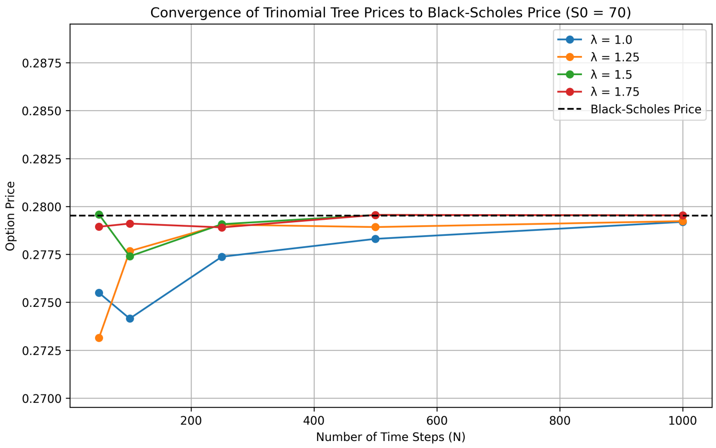

Figure 2: Price Estimation from Tree Discretisation ($S_0 = 70$)

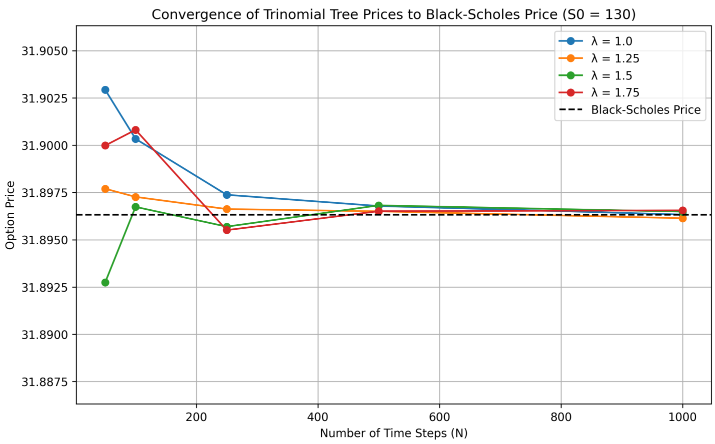

Figure 3: Price Estimation from Tree Discretisation ($S_0 = 130$)

While $S_0 = 70$ and $S_0 = 130$ do not display the same monotonic behaviour, the price estimates seem to converge well to the Black-Scholes price. The absolute errors are also noticeably smaller compared to $S_0 = 100$, likely due to the linearity of the payoff function for deep in-the-money/out-of-the-money options. While the errors for these options are similar in magnitude, the difference in price causes the deep in-the-money option to exhibit a much smaller relative error, as shown in Table 1.

1.1c.

S_0	$\lambda = 1$		$\lambda = 1.25$		$\lambda = 1.5$		$\lambda = 1.75$	
70	-0.0012	(-0.43%)	-0.0006	(-0.21%)	0.0000	(0.01%)	0.0000	(0.01%)
75	-0.0015	(-0.24%)	-0.0003	(-0.04%)	0.0003	(0.04%)	0.0005	(0.08%)
80	0.0011	(0.08%)	-0.0009	(-0.07%)	0.0005	(0.04%)	0.0003	(0.02%)
85	-0.0018	(-0.07%)	0.0009	(0.04%)	-0.0008	(-0.03%)	0.0011	(0.05%)
90	-0.0011	(-0.03%)	0.0013	(0.03%)	-0.0003	(-0.01%)	0.0006	(0.02%)
95	-0.0004	(-0.01%)	0.0017	(0.03%)	0.0001	(0.00%)	0.0007	(0.01%)
100	-0.0040	(-0.05%)	-0.0011	(-0.01%)	-0.0025	(-0.03%)	-0.0041	(-0.05%)
105	0.0022	(0.02%)	0.0016	(0.01%)	0.0016	(0.01%)	-0.0016	(-0.01%)
110	0.0025	(0.02%)	0.0016	(0.01%)	-0.0009	(-0.01%)	-0.0020	(-0.01%)
115	0.0003	(0.00%)	0.0013	(0.01%)	0.0013	(0.01%)	-0.0020	(-0.01%)
120	-0.0001	(0.00%)	0.0005	(0.00%)	0.0010	(0.00%)	0.0010	(0.00%)
125	0.0013	(0.00%)	-0.0010	(0.00%)	0.0006	(0.00%)	0.0003	(0.00%)
130	0.0005	(0.00%)	0.0002	(0.00%)	0.0005	(0.00%)	0.0002	(0.00%)

Table 1: Tree Price Absolute Error (and % Relative Error)

Consistent with the results obtained in the previous section, the tree pricer seems to have a much smaller absolute error when S_0 is far away from the strike. The absolute errors for small and large S_0 are also similar in magnitude. By averaging the absolute and relative error of each λ across all values of S_0, we observe that $\lambda = 1.5$ has the highest accuracy, with average absolute error 0.0008 and average relative error 0.015%. Therefore, we will fix $\lambda = 1.5$ in the later parts.

1.2a.

To price an American option, we have to take into account the possibility of early exercise. Instead of setting

$$V_k^n = e^{-r\Delta t}[q_u V_k^{n+1} + q_m V_{k+1}^{n+1} + q_d V_{k+2}^{n+1}]$$

we use

$$V_k^n = max\{g(s_k^n), e^{-r\Delta t}[q_u V_k^{n+1} + q_m V_{k+1}^{n+1} + q_d V_{k+2}^{n+1}]\}$$

where $g(s_k^n)$ is the exercise/intrinsic value.

1.2b.

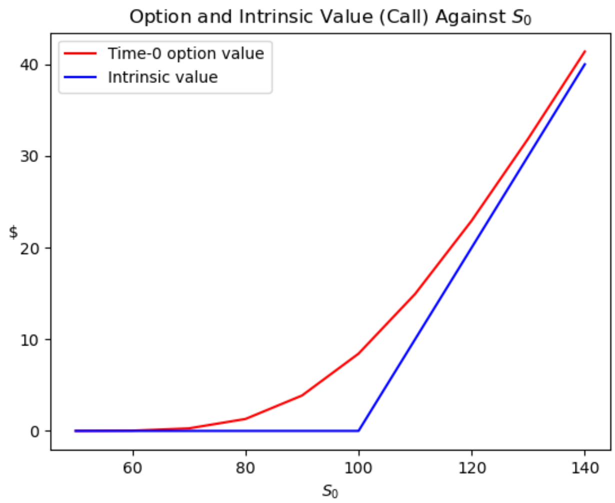

Figure 4: Time-0 and Intrinsic Value of American Calls

The time-0 option value is always greater than or equal to the intrinsic value, which is unsurprising due to the backward induction steps. In the last iteration of backward induction, we compare the exercise value $g(s_0^0)$ and the continuation value $e^{-r\Delta t}[q_u V_0^1 + q_m V_1^1 + q_d V_2^1]$, and observe that the continuation value is strictly greater than the exercise value for all S_0. Therefore, the optimal strategy is to not exercise at time-0 and wait until the option maturity.

1.2c.

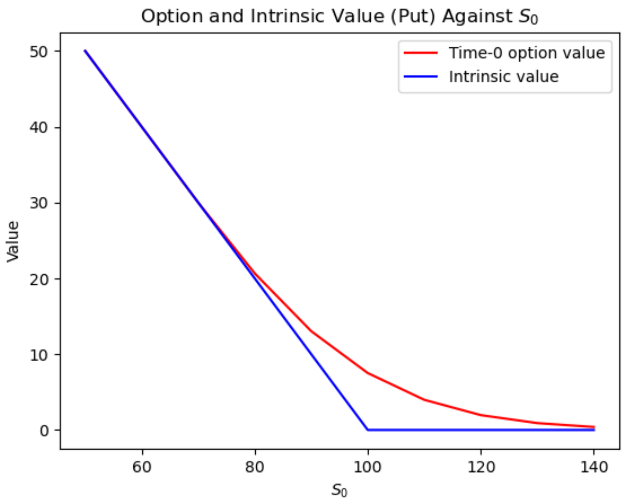

Figure 5: Time-0 and Intrinsic Value of American Puts

While the graph looks similar to Figure 4, in this case, the exercise value is greater than the continuation value when S_0 is small enough, i.e. when the option is sufficiently in-the-money. In particular, for $r = 0.01, \sigma = 0.2$, early exercise at time 0 is optimal for S_0 less than or equal to

70. When r increases or σ decreases, the threshold at which early exercise is optimal (threshold = 70 in the previous case) increases. Intuitively, when r is large and σ is small, the positive drift will cause the stock price to increase on average and dominates the fluctuations from the Brownian motion. This will reduce the payoff of the put option, implying that it has a negative time value, and exercising the option now to lock in profits is optimal.

1.3a.

The algorithm for lookback options uses a similar method to part 1. However, we now have to keep track of an additional auxiliary variable, the running maximum M_n in this case. We now consider a pair of variables (j, k), which represents the current state,

$$(S_n, M_n) = (s_k^n, m_j^n) = (S_0 u^{n-k}, S_0 u^{n-j})$$

with $k = 0, 1, ..., 2n$ and $j = 0, 1, ..., n$.
To understand how M_n evolves from m_j^n as $s_k^n \to s_{k'}^{n+1}$, from the definition of the running maximum, we have:

$$m_{j'}^{n+1} = \max(m_j^n, s_{k'}^{n+1})$$

$$S_0 u^{n+1-j'} = \max(S_0 u^{n-j}, S_0 u^{(n+1)-k'})$$

$$n + 1 - j' = \max(n - j, n + 1 - k')$$

$$j' = n + 1 - \max(n - j, n + 1 - k') = \min(k', j + 1)$$

To describe the index of the new state $s_k^n \to s_{k'}^{n+1}$ of the auxiliary variable M_n we consider $(j, k) \to (\phi(j, k'), k'))$, with ϕ being a payoff-specific shooting function. By the above reasoning, the shooting function in this context is:

$$\phi(j, k') = min\{j + 1, k'\}$$

In pseudocode,

$k \leftarrow 0$
while $k \leq 2N$ **do**
  $j \leftarrow 0$
  while $j \leq N$ **do**
    $V_{j,k}^N \leftarrow m_j^N - s_k^N$
    $j \leftarrow j + 1$
  end while
  $k \leftarrow k + 1$
end while
$n \leftarrow N - 1$
while $n \geq 0$ **do**
  $k \leftarrow 0$
  while $k \leq 2n$ **do**
    $j \leftarrow 0$
    while $j \leq n$ **do**
      $V_{j,k}^n \leftarrow e^{-r\Delta t}[q_u V_{\phi(j,k),k}^{n+1} + q_m V_{\phi(j,k+1),k+1}^{n+1} + q_d V_{\phi(j,k+2),k+2}^{n+1}]$
      $j \leftarrow j + 1$

end while
$k \leftarrow k+1$
end while
$n \leftarrow n-1$
end while

1.3b.

Type \N	2	5	10	25	50	100	250	500	1000	2500	5000	10000
European	7.947	10.253	11.736	13.264	14.115	14.751	15.340	15.646	15.866	16.063	16.164	16.235
	(0.000)	(0.000)	(0.001)	(0.002)	(0.005)	(0.021)	(0.111)	(0.574)	(3.208)	(39.180)	(461.665)	(3089.413)
American	8.063	10.385	11.869	13.395	14.246	14.883	15.472	15.778	15.999	16.196	16.297	16.368
	(0.000)	(0.000)	(0.000)	(0.002)	(0.008)	(0.035)	(0.209)	(1.029)	(5.301)	(103.334)	(700.313)	(4247.076)

Table 2: Estimated Lookback Option Prices (Computational Time in Seconds)

It seems that even for $N = 10000$, the pricer does not obtain reliable estimates, as we observe non-negligible changes going from $N = 5000$ to $N = 10000$ .The computational time is $O(N^3)$, where N is the number of discretisation steps.

2.1a.

Using a change of variables $\tau = T - t$ and $U(\tau, s) = V(t, s)$, through simple calculations we obtain $V_t = -U_\tau$ and so derive

$$U_\tau = \frac{1}{2}\sigma^2(T-\tau, s)s^2 U_{ss} + rsU_s - rU, \ , \tau < T$$

with initial condition $U(0, s) = g(s)$.

2.1b.

The $\theta-$scheme is a weighted combination (with weight θ) of the explicit scheme

$$\frac{V_k^n - V_k^{n-1}}{\Delta t} = a_k^{n-1}\frac{V_{k+1}^{n-1} - 2V_k^{n-1} + V_{k-1}^{n-1}}{\Delta x^2} + b_k^{n-1}\frac{V_{k+1}^{n-1} - V_{k-1}^{n-1}}{2\Delta x} - c_k^{n-1}V_k^{n-1}$$

and the implicit scheme

$$\frac{V_k^n - V_k^{n-1}}{\Delta t} = a_k^n\frac{V_{k+1}^n - 2V_k^n + V_{k-1}^n}{\Delta x^2} + b_k^n\frac{V_{k+1}^n - V_{k-1}^n}{2\Delta x} - c_k^n V_k^n$$

For the PDE defined above in 2.1a,
$a_k^n = \frac{1}{2}\sigma^2(T-\tau_n, s_k)s_k^2$, $b_k^n = rs_k$ and $c_k^n = r$.
The $\theta-$scheme can be written in matrix form,

$$(\mathbf{I} - \theta L^n)V^n = (\mathbf{I} + (1-\theta)L^{n-1})B^{n-1}(V^{n-1})$$

where B is the boundary operator given by $B^n(x) = [l^n(x_{min}), x_1, ..., x_{M-1}, u^n(x_{max})]$ and L^n is a matrix defined in terms of a_k^n, b_k^n and c_k^n. In this case, by fixing $x_{min} = 10$ and $x_{max} = 300$, the boundary conditions are given by $l^n(x_{min}) = 0$ and $u^n(x_{max}) = x_{max} - Ke^{-r\tau_n}$.
The $\theta-$scheme corresponds to the explicit scheme when $\theta = 0$, the implicit scheme when $\theta = 1$ and the Crank-Nicolson scheme when $\theta = 0.5$.

2.1c.

$\theta \backslash S_0$	80	85	90	95	100	105	110	115	120
0	0.7546	1.4752	2.6540	4.3780	6.7111	9.7131	13.2811	17.3152	21.7136
1/2	0.7549	1.4752	2.6535	4.3772	6.7102	9.7123	13.2806	17.3151	21.7137
1	0.7551	1.4751	2.6531	4.3765	6.7093	9.7115	13.2801	17.3149	21.7138

Table 3: Estimated Call Option Prices (Local Volatility Model)

2.2b.

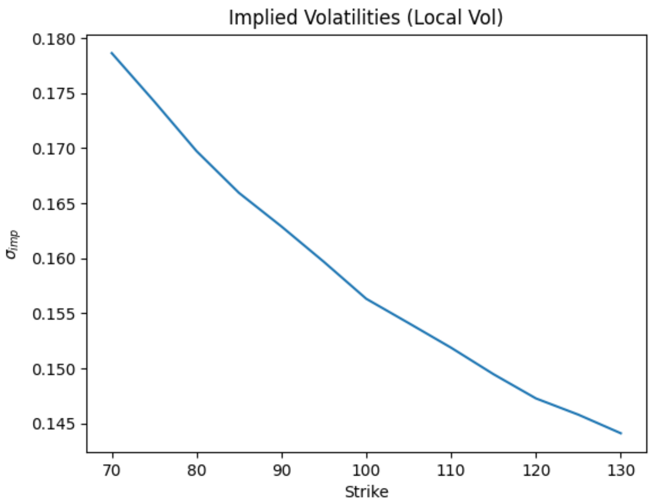

Figure 6: Implied Volatility of Calls under Local Volatility Model

When plotting implied volatilities against call prices calculated using the implicit scheme ($\theta = 1$), we observe a downward sloping curve, which is consistent with the decreasing local volatility function in s. Intuitively, for large strikes, there is only a positive payoff if the realisation of stock price process increases past the strike. However, under the local volatility model, this is less likely due to the decreasing volatility function. Therefore, it has a lower price compared to the constant volatility Black-Scholes price, and hence a lower implied volatility.

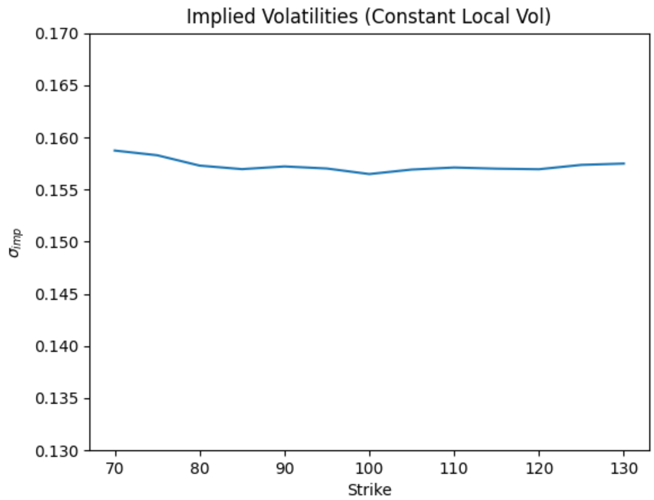

Figure 7: Implied Volatility of Calls under constant Volatility Model

Here we see implied volatility that is roughly constant, as expected.

2.3a.

By the in-out parity,

$$(K - S_t)_+ = (K - S_t)_+ \mathbb{1}_{L_T < B} + (K - S_t)_+ \mathbb{1}_{L_T \geq B}$$

since we know how to compute the price of a put option (inverting call prices from 2.1c using put-call parity), we can retrieve the price of a down-and-in put option by computing the price of a down-and-out put option. The price of a down-and-out put option follows the same PDE as 2.1a, but with different boundary conditions, i.e. $l(B) = 0$ and $u(x_{max}) = 0$.

2.3b.

Vol Model \ B_{in}	60	70	80	90
Local Volatility	0.232	1.232	3.507	5.458
Constant Volatility	0.049	0.712	3.101	5.427

Table 4: Down-and-In Barrier Put Option Prices Over Different Knock-In Levels

B_{in}	60	70	80	90
Percentage Difference	370.9	73.1	13.1	0.6

Table 5: Percentage Difference (Constant Local Vol as Baseline)

A higher Knock-In level makes it more likely the barrier is hit and we observe that our option prices increase as expected. When analysing the percentage difference in option prices for the two models, we note that when the stock price is close to the lower Knock-In levels (say $B_{in} = 60$), the local volatility function is much higher (around 22%), while at higher Knock-In levels (say $B_{in} = 90$), the local volatility function is lower (around 16%) and nearer to the 15.7% value used in the constant volatility model. Therefore, for lower barrier levels, the probability that the stock price reaches the barrier is significantly higher than the constant volatility model, while for higher barrier levels, the probability is relatively unchanged. Hence, the percentage difference in the option prices is much higher for lower barrier levels, as we observe in Table 5.