Josh Avery, Saad Rafiq

Numerical Analysis II
April 20th, 2023

A Comparative Study of Finite Difference Methods for Solving the Black-Scholes Partial Differential Equation for American Options Pricing

Introduction

Financial options are derivatives whose values are based on underlying asset prices. European and American options are two types of financial options with different characteristics. European options can only be exercised at the expiration date, while American options can be exercised at any time before the expiration date. This difference has significant implications for pricing and hedging strategies. Furthermore, the attractiveness of studying this financial instrument is enhanced by the fact that the majority of single-stock options available on exchanges such as the New York Stock Exchange are American-style (Chen, 2022).

The Black-Scholes model is a widely used mathematical model for pricing call and put options. The Black-Scholes model assumes that the underlying asset follows a geometric Brownian motion and that the market is efficient. Black and Scholes showed in their seminal paper (Black & Scholes, 1973) that the pricing distribution of the underlying asset follows a lognormal distribution. Under these assumptions, the value of a European call option can be determined by solving the Black-Scholes-Merton partial differential equation (PDE):

$$\frac{\partial \mathrm{V}}{\partial \mathrm{t}} + \frac{1}{2}\sigma^2 \mathrm{S}^2 \frac{\partial^2 \mathrm{V}}{\partial S^2} + (r - q)S\frac{\partial \mathrm{V}}{\partial \mathrm{S}} - rV = 0 \tag{1}$$

Equation 1: The Black-Scholes-Merton PDE

where V is the value of the option, t is time, S is the price of the underlying asset as a function of time, σ is the volatility of the underlying asset (which we assume is constant across time), r is the risk-free interest rate, and q is the continuous dividend yield from the underlying security.

However, the infamous Black-Scholes formula, which emerges from solving (1), assumes European options. The problem of pricing American options requires finding the optimal exercise time when the option holder should exercise the contract to maximize their profit. Without a specific exercise time constraint, an analytic solution for the Black-Scholes PDE does not currently exist (Hull, 2021).

The optimal exercise problem for American options can be formulated as a free-boundary problem, where the option holder must decide when to stop waiting and exercise the option. An American option's value is modeled as the maximum of the value gained from immediate exercise or waiting until a later time to exercise. The challenge in solving this problem is that the optimal exercise time depends on the future price of the underlying asset, which is uncertain.

Numerical methods such as finite difference methods, spectral methods, and Monte Carlo methods have been used to approximate solutions to the Black-Scholes PDE for American options, as seen in Bouchard, Chau, Manai, & Sid-Ali (2019), Song, Zhang, & Tian (2014). These methods allow for the incorporation of the possibility of early exercise into the pricing model by simulating the future prices of the underlying asset and determining the optimal exercise time at each time step. The numerical method choice depends on the characteristics of the problem, such as the complexity of the underlying asset price dynamics and the time demands on the solution.

In our project, we will compare the use of three finite difference methods to solve the problem of pricing American options. We will compare and contrast the performance of the well-known Binomial Tree model as proposed by Cox, Ross, & Rubinstein (1979), the Crank-Nicolson method (Crank & Nicolson, 1946), and an explicit Runge-Kutta scheme SERKv2 as proposed by Martin-Vaquero, Khaliq, & Kleefeld (2014). We aim to provide insights into each method's strengths and weaknesses by identifying the circumstances under which each technique is most appropriate, specifically regarding run time, numerical convergence speed, and stability conditions.

Hypothesis

We expect the Crank-Nicholson Method to have a balanced trade-off between runtime and mesh size. The Binomial Tree model is expected to have a costly trade-off between the mesh size and runtime but the most comprehensive application range (due to its stability conditions) over the other two methods. The Runge-Kutta method is expected to have a high convergence order and stability issues.

Methods

All three of the methods implemented are variants of finite-difference methods. The methods generally work by discretizing the domain into nodes and estimating the function values at each node.

The Binomial Tree Model

The Binomial Tree model models the underlying asset's price over time as a series of possible future price paths represented as a tree. Rubinstein (2000) pointed out that the binomial option pricing model is a special case of an explicit finite difference method. Due to its simplicity and versatility, the binomial tree model has been widely adopted by traders and is still used today as a standard tool for option pricing.

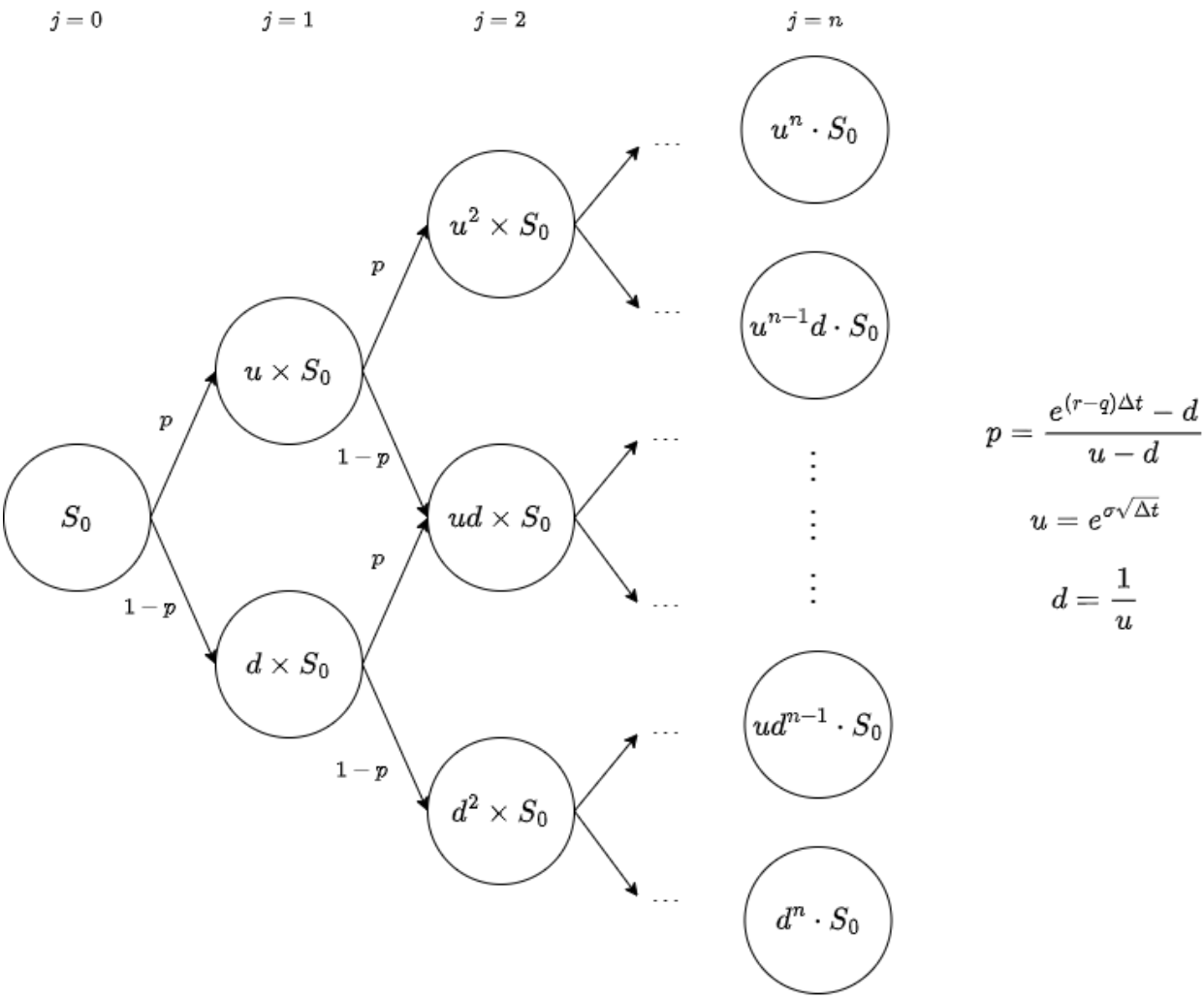

Figure 1: The Pricing Tree For the Binomial Tree Model

The model assumes the price of the asset moves up or down at each time step and calculates the expected value of the option at each node of the tree, which is given by:

$$\mathrm{V_{t-\Delta t,i}} = \mathrm{e^{-r\Delta\,t}}\left(\mathrm{p} \cdot \mathrm{V}_{t,i} + (1-p) \cdot \mathrm{V}_{t,i+1}\right) \tag{2}$$

Equation 2: Expected Value of Binomial Tree Node

where *t* is the time since the contract began, *⊿t* is the timestep, *n* is the depth of the tree, *p* is the probability that the underlying stock price increases *u* and *d* are the values the stock is expected to increase and decrease by, respectively, and S_0 is the initial price of the security.

Once the tree of prices (shown in Figure 1) is created, the option value is recursively calculated starting from the expiration date backward to the current value. The payoff for the call or put (respectively given in Equations 3 and 4) with the stock price adjusted by the pricing tree is assigned to the final layer of nodes. The option values at the previous timestep are calculated. Algorithm 1, below, describes the process in detail. This algorithm runs with a time complexity of $\mathcal{O}(n^2)$, where *n* is the height of the tree.

$$\max(\mathrm{S} \cdot e^{-rt} - \mathrm{K} \cdot e^{-qt}, 0) \tag{3}$$

Equation 3: Time-Adjusted Payoff of a Call Option

$$\max(\mathrm{K} \cdot e^{-qt} - \mathrm{S} \cdot e^{-rt}, 0) \tag{4}$$

Equation 4: Time-Adjusted Payoff of a Put Option

Algorithm 1: Binomial Tree Method of Pricing an American Put Option

$\Delta t \leftarrow T/n$
$u \leftarrow e^{\sigma\sqrt{\Delta t}}$
$p_{up} \leftarrow (u \cdot e^{-q\Delta t} - e^{-r\Delta t})/(u^2 - 1)$
$p_{down} \leftarrow e^{-r\Delta t} - p_0$
$P \leftarrow (n+1) \times (n+1)$ matrix of zeros
for $i = 0$ to n **do**
    $P_{n,i} \leftarrow \max(S \cdot u^{2i-n} - K, 0)$
end for
for $j = n-1$ **down to** 0 **do**
    for $i = 0$ to j **do**
        $P_{j,i} \leftarrow p_{up} \cdot P_{j+1,i+1} + p_{down} \cdot P_{j+1,i}$
        $exercise \leftarrow K - S \cdot u^{2i-j}$
        $P_{j,i} \leftarrow \max(P_{j,i}, exercise)$
    end for
end for
return $P_{0,0}$

Algorithm 1: Binomial Tree Algorithm for Pricing American Puts

In order for the binomial model to be valid, *p* must fall in the interval $(0,1)$ since it is a probability value. With some algebra on equation 5, one can see that *p* must conform to equation 6 for *p* to be a valid probability.

$$0 < \frac{e^{(r-q)\Delta t} - d}{u - d} < 1 \tag{5}$$

Equation 5: Restriction on p

$$\Delta t < \frac{\sigma^2}{(r-q)^2} \tag{6}$$

Equation 6: Stability Condition for the Binomial Tree Model

The Crank-Nicolson Method

The Crank-Nicolson finite difference method is a numerical technique used in option pricing to estimate the price of an option based on its underlying asset. It uses six points per approximation in a discretization of time and space to approximate the partial differential equation governing the evolution of the option price. In the context of option pricing, the Crank-Nicolson method is a popular approach for solving the Black-Scholes equation, which prices European and

American options. The method is based on a combination of explicit and implicit finite difference methods, which allows it to provide an accurate and stable solution.

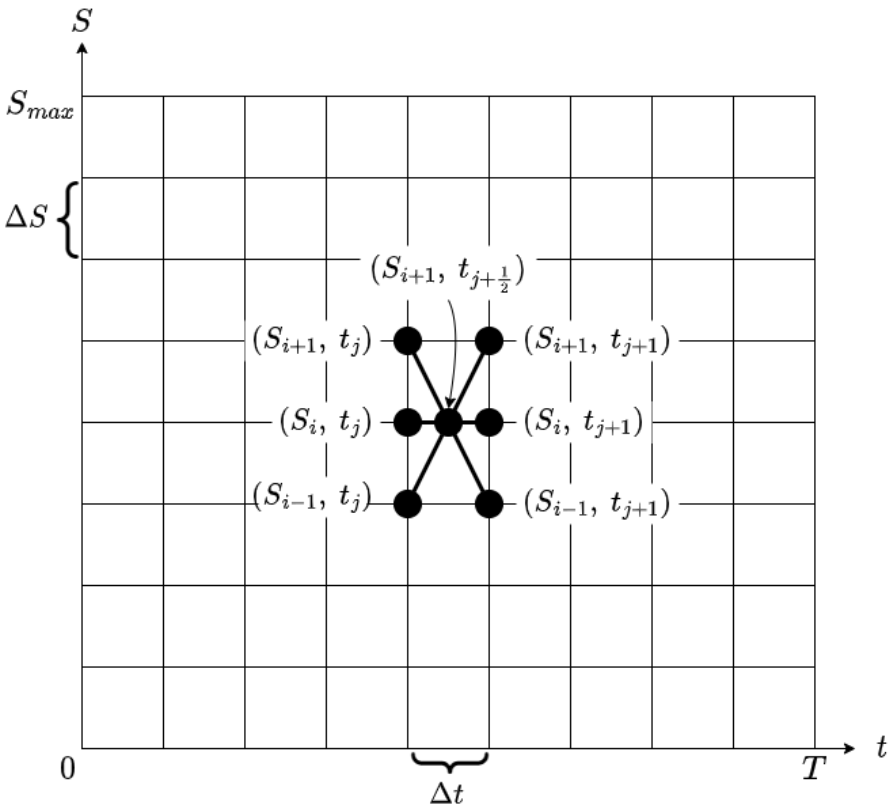

Figure 2: Crank-Nicolson Discretization

We use an adaptation of the Crank-Nicolson model, as discussed in Forsyth, Windcliff, & Vetzal (2004).

Define α_i and β_i as follows:

$$\alpha_i = \begin{cases} \frac{1}{2}\sigma^2 \frac{S_i^2}{\Delta S^2} & if\ \frac{1}{2}\sigma^2 \frac{S_i^2}{\Delta S^2} < (r-q)\frac{S_i}{2\Delta S} \\ \frac{1}{2}\sigma^2 \frac{S_i^2}{\Delta S^2} - (r-q)\frac{S_i}{\Delta S} & if\ \beta_i < 0 \\ \frac{1}{2}\sigma^2 \frac{S_i^2}{\Delta S^2} - (r-q)\frac{S_i}{2\Delta S} & otherwise \end{cases} \quad (7)$$

Equation 7: Definition of α

$$\beta_i = \begin{cases} \frac{1}{2}\sigma^2 \frac{S_i^2}{\Delta S^2} & if\ \frac{1}{2}\sigma^2 \frac{S_i^2}{\Delta S^2} < -(r-q)\frac{S_i}{2\Delta S} \\ \frac{1}{2}\sigma^2 \frac{S_i^2}{\Delta S^2} + (r-q)\frac{S_i}{\Delta S} & if\ \alpha_i < 0 \\ \frac{1}{2}\sigma^2 \frac{S_i^2}{\Delta S^2} + (r-q)\frac{S_i}{2\Delta S} & otherwise \end{cases} \quad (8)$$

Equation 8: Definition of β

Then let the matrix *A* be defined as:

$$\mathrm{A} = \begin{bmatrix} r & 0 & & 0 & \cdots & 0 \\ -\alpha_2 & r+\alpha_2+\beta_2 & -\beta_2 & & \ddots & \vdots \\ 0 & & & \ddots & & 0 \\ \vdots & \ddots & & -\alpha_{n-1} & r+\alpha_{n-1}+\beta_{n-1} & -\beta_{n-1} \\ 0 & \cdots & 0 & \gamma_{n-2} & \gamma_{n-1} & \gamma_n \end{bmatrix} \quad (9)$$

Equation 9: Definition of A

where γ_{n-2}, γ_{n-1}, and γ_n define the linear boundary condition at S_{max}:

$$\gamma_{n-2} = 0 \quad (10)$$

Equation 10: γ_{n-2}

$$\gamma_{n-1} = (r-q)\frac{S_n}{\Delta S} \quad (11)$$

Equation 11: γ_{n-1}

$$\gamma_n = r - \gamma_{n-1} \quad (12)$$

Equation 12: γ_n

Define B as:

$$B = \left(I_n + \frac{1}{2}A\Delta t\right)^{-1}\left(I_n - \frac{1}{2}A\Delta t\right) \quad (13)$$

Equation 13: Definition of B

where I_n is the n × n identity matrix. Finally, the Crank-Nicolson method's discretization is written as follows:

$$V^{(j+1)} = BV^{(j)} \quad (14)$$

Equation 14: Crank-Nicolson Finite Difference Formulation

where V^j is the vector of option values across possible stock prices at time j.

With this formulation, we have a legitimate discretization which is stable if r > 0 and when q < 0, $\Delta t < \left|\frac{1}{q}\right|$ (Forsyth, Windcliff, & Vetzal, 2004).

Algorithm 2: Crank-Nicolson Method for American Options

$\Delta t \leftarrow \frac{T}{n_t}$
$\Delta S \leftarrow \frac{S_{max}-S_{min}}{n_s}$
$first \leftarrow \frac{1}{2}\sigma^2 \frac{S^2}{\Delta S^2}$
$second \leftarrow (r-q)\frac{S}{2\Delta S}$
for $i = 1$ *to* n_s **do**
  $\alpha_i \leftarrow first - second$
  $\beta_i \leftarrow first + second$
  if $\alpha_i < 0$ **then**
    $\beta_i \leftarrow first + 2 \cdot second$
  end
  if $\beta_i < 0$ **then**
    $\alpha_i \leftarrow first + 2 \cdot second$
  end
  if $\alpha_i < 0$ **then**
    $\alpha_i \leftarrow first$
  end
  if $\beta_i < 0$ **then**
    $\beta_i \leftarrow first$
  end
end
$A \leftarrow n_S \times n_S$ *matrix of* $0s$
$A_{1,1} \leftarrow r$
for $i = 2$ ***to*** $n_s - 1$ **do**
  $A_{i,i-1} \leftarrow -\alpha_i$
  $A_{i,i} \leftarrow r + \alpha_i + \beta_i$
  $A_{i,i+1} \leftarrow -\beta_i$
end
$A_{n_S,n_S-2} \leftarrow \gamma_{n-2}$
$A_{n_S,n_S-1} \leftarrow \gamma_{n-1}$
$A_{n_S,n_S} \leftarrow \gamma_n$
$\Lambda_A \leftarrow Eigenvalues(A)$
$\Lambda_B \leftarrow \frac{1-(\frac{1}{2}\Lambda_A \Delta t)}{1+(\frac{1}{2}\Lambda_A \Delta t)}$
$\rho_B \leftarrow \max|\Lambda_B|$
if $\rho_B > 1$ **then**
  STOP
end
if $\rho_B = 1$ **then**
  $B \leftarrow (I_n + \frac{1}{2}\Delta t A)^{-1}(\frac{1}{2}\Delta t A)$
  $is\_valid \leftarrow Check\_Mult(B)$
  if $\neg is\_valid$ **then**
    STOP
  end
else
  $B \leftarrow (I_n + \frac{1}{2}\Delta t A)^{-1}(\frac{1}{2}\Delta t A)$
end
$V \leftarrow n_S \times n_t$ *matrix of* $0s$
for $i = 1$ *to* n_S **do**
  $V_{i,0} \leftarrow \max\{K - S, 0\}$
  $V_{i,n_S} \leftarrow \max\{Ke^{-rT} - Se^{-qT}, 0\}$
end
for $i = 1$ *to* $n_t - 1$ **do**
  $V_{:,i+1} \leftarrow BV_{:,i}$
  if *is_call* **then**
    $V_{:,i+1} \leftarrow \max\{V_{:,i+1}, Se^{-q(i*\Delta t)} - Ke^{-r(i\Delta t)}\}$
  else
    $V_{:,i+1} \leftarrow \max\{V_{:,i+1}, Ke^{-r(i\Delta t)} - Se^{-q(i*\Delta t)}\}$
  end
end
return V

Algorithm 2: Crank-Nicolson Algorithm for Pricing American Puts

Algorithm 2 details the implementation we used. Here are the most time-consuming portions of Algorithm 2:

- $\mathcal{O}(n_S^3)$ for calculating eigenvalues of *A*.
- $\mathcal{O}(n_S^3)$ for computing the matrix *B*.
- $\mathcal{O}(n_S n_t)$ for initializing and filling matrix *V*.
- $\mathcal{O}(n_S^4)$ for calling the Check_Mult function

The overall time complexity of Algorithm 2 is dominated by the Check_Mult function, which is $\mathcal{O}(n_S^4)$. The Check_Mult function ensures that the algebraic and geometric multiplicities of each eigenvalue of *B* are equal, which is a necessary check for stability, as mentioned by Forsyth, Windcliff, & Vetzal (2004). Therefore, the time complexity of Algorithm 2 is $\mathcal{O}(n_S^4)$.

Stabilized Explicit Runge-Kutta (SERK2v2)

The Stabilized Explicit Runge-Kutta numerical method called SERK2v2 is designed for solving the right-hand side of parabolic partial differential equations (PDEs). The method used is based on Martin-Vaquero, Khaliq, and Kleefeld (2014), where the derivation and pseudo implementation of the SERK2v2 method is defined.

The SERK2v2 method makes the Black-Scholes PDE discrete to create a set of equations for the option values at each time step and stock price. The method then solves these equations explicitly in a two-stage approach with variable coefficients that depend on the stock price and time. The discretized stock prices and time are defined as:

$$S_n = n\Delta S, \quad t_m = m\Delta T \tag{15}$$

Where $n = 0, 1, \dots, N$ and $m = 0, 1, \dots, M$. The right-hand side of the Black-Scholes PDE can then be discretized as:

$$g^{(m)}\mathrm{n} = g^{(m)}\mathrm{n} + \Delta T \left[\begin{array}{l} \frac{1}{2}\sigma^2 S_n^2 \frac{g^{(m)}n - 1 - 2g^{(m)}n + g^{(m)}n + 1}{\Delta S^2} \\ + \left(r - \frac{1}{2}\sigma^2\right) S_n \frac{g^{(m)}n + 1 - g_{n-1}^{(m)}}{2\Delta S} - r g_n^{(m)} \end{array} \right] \tag{16}$$

The SERK2v2 method proceeds by solving the right-hand side of the discretized Black-Scholes PDE for each time step m from $M - 1$ down to 0 and for each stock price S_n where $n = 1, 2, \dots, N - 1$. The boundary conditions applied at $n = 0$ and $n = N$ are:

$$g_0 = 0, \quad g_N = S_N - Ke^{-r(T - t_m)} \tag{17}$$

Where g_0 and g_N are the option values at the lower and upper stock price boundaries, respectively, and K is the option strike price.

The SERK2v2 numerical method employs a stabilization parameter α to ensure the algorithm's stability for a range of time steps and stock price step sizes. The method consists of s stages, and the stabilization parameter is defined as:

$$g_n^{(m)} = \max g_n^{(m)}, S_n - K \tag{18}$$

The early exercise constraint is applied at each stage of the SERK2v2 algorithm, ensuring that the calculated option values always satisfy the constraint and stock prices.

The implementation of the SERK2v2 algorithm is shown below. The function takes several parameters, including option values at the previous time step, the stock prices, the risk-free rate, the stock price volatility, and the strike price. Then it loops over the calculation of the Black-

Scholes equation's right-hand side and the early exercise constraint. It returns the average of the option values across all stages.

Algorithm 3 SERK2v2 Algorithm

function SERK2v2(g_prev, $stock\_prices$, r, σ, dS, dT, s_stages, $time$, m, K)
  $\alpha \leftarrow \frac{1}{0.4 \times s\_stages^2}$
  $g \leftarrow \text{zeros}(s\_stages + 1, \text{length}(stock\_prices))$
  $g[0] \leftarrow g\_prev$
  for $j \in \{1, \ldots, s\_stages\}$ **do**
    if $j \bmod 2 = 0$ **then**
      $c \leftarrow 2$
    else
      $c \leftarrow 1$
    end if
    $g\_interior \leftarrow \text{black\_scholes\_rhs}(stock\_prices, g[j-1], r, \sigma, dS, dT) \times c \times \alpha$
    $g\_interior \leftarrow \max(g\_interior, stock\_prices[1:-1] - K)$
    $g_0 \leftarrow 0$
    $g_N \leftarrow stock\_prices[-1] - K \times e^{-r \times (T - time[m])}$
    $g[j] \leftarrow \text{concatenate}([g_0, g\_interior, g_N])$
  end for
  return $\text{mean}(g, \text{axis} = 0)$
end function

Algorithm 3: SERKv2

The *black_scholes_american_call_option* function is the driver function for the SERK2v2 function. It initializes the option values matrix and sets the option values at maturity using the boundary condition. It then iterates through the time steps, calling the SERK2v2 function, storing the results, and returning V, the matrix of option values.

Algorithm 4 Black-Scholes American Call Option using SERK2v2

function BLACK_SCHOLES_AMERICAN_CALL_OPTION(S, K, T, r, σ, N, M, s_stages)
  $dS \leftarrow \frac{2 \times S}{N}$
  $dT \leftarrow \frac{T}{M}$
  $time \leftarrow \text{linspace}(0, T, M + 1)$
  $V \leftarrow \text{zeros}(N + 1, M + 1)$
  $V[:, -1] \leftarrow \max(stock\_prices - K \times e^{-r \times time}, 0)$
  for $m \in \{M - 1, \ldots, 0\}$ **do**
    $V[:, m] \leftarrow \text{SERK2v2}(V[:, m+1], stock\_prices, r, \sigma, dS, dT, s\_stages, time, m, K)$
  end for
  return V
end function

Algorithm 4: SERKv2 Pricing of an American Call Option

It can be observed that the SERK2v2 function has a time complexity of $\mathcal{O}(N)$ and the $black\_scholes\_american\_call\_options$ function has a time complexity of $\mathcal{O}(N^2)$

The parameters used are defined as follows:

- S (float): initial stock price
- K (float): strike price
- T (float): time to maturity
- r (float): risk-free interest rate
- σ (float): stock price volatility
- N (int): number of discretized stock price steps
- M (int): number of discretized time steps
- s_stages (int): number of SERK2v2 stages

By comparing the results obtained from these finite difference methods, we aim to gain insights into their relative strengths and limitations for option pricing in the financial markets.

Rationale

The pricing of options has been an important research topic in finance for several decades. The traditional approach to option pricing involves using closed-form analytical solutions, which are only available for a limited set of simple contracts. More complex contracts, like American options, cannot be priced analytically and must be solved using numerical methods. Therefore, we focus on the pricing of American options.

In this paper, we focus on finite difference methods, a common numerical technique for solving PDEs and widely used in option pricing.

The Binomial Tree and Crank-Nicolson methods are two popular finite difference methods for option pricing. These methods have been extensively studied in the literature and are widely used in practice. However, they have limitations, such as being computationally expensive and unstable for certain options.

In 2014, a stabilized explicit Runge-Kutta (SERK) scheme emerged as a more novel methodology for option pricing. In general, Runge-Kutta schemes are established, powerful finite difference methods, and it has only recently been applied to option pricing effectively. Therefore, we wanted to investigate the SERK scheme Martin-Vaquero, Khaliq, & Kleefeld proposed and compare its performance with the more established Binomial Tree and Crank-Nicolson methods.

In summary, this paper aims to study the performance of finite difference methods for pricing American options, with a particular focus on comparing the Binomial Tree, Crank-Nicolson, and SERK methods. By providing a detailed analysis of these methods, we hope to offer valuable insights for practitioners and researchers in the field of option pricing.

Results

Binomial Tree

The Binomial Tree and Crank Nicolson methods were tested on a Surface Pro 6 running Python 3.11 on Windows 10 with the following specifications:

- Intel Core i7 8650U CPU @ 1.9 GHz
  - Four cores, two threads each
  - Live clock speed: 2.4 – 2.6 GHz for each core
- 16GB of DDR3 RAM
- Caches:
  - L1-D and L1-I, 4x32KB, 8-way
  - L2 4x256KB, 4-way
  - L3 8MB, 16-way

The Binomial Tree method was tested on parameters derived from GOOG and APPL calls and puts. Table 3 shows these parameters. Table 4**Error! Reference source not found.** present the runtimes for each set of tree heights. The tree heights are 20, 100, 500, 1,000, 2,500, 5,000, 10,000, 25,000, and 50,000, which translates to timesteps of roughly bi-monthly down to every 20 minutes. Memory constraints barred us from testing with larger tree heights.

	T[1]	σ	K	r	q	S_0
GOOG[2]	2	35.9[3]	105	3.585%[4]	0%	$104.70[5]
AAPL	2	32.6	170	3.585%	5.5%	$167.63

Table 1: Binomial Tree Parameters

Tree Height (n)	**Google Call**	**Apple Call**	**Google Put**	**Apple Put**
20	0.0017359	0.0016396	0.0008454	0.0010092
100	0.023339	0.024604	0.0162228	0.0152802
500	0.5795052	0.5219235	0.4386829	0.402108
1,000	1.9802445	1.9216243	1.6476108	1.6310733
2,500	12.1569164	12.1644261	10.693373	10.4135767
5,000	47.5867264	44.4187062	43.4996731	42.2884514
10,000	179.086233	180.2901271	194.6959253	192.2900986
25,000	1130.097174	1218.819917	1179.468089	1123.494705
50,000	4709.238731	4397.227161	5117.154921	4945.517523

Table 2: Runtime (seconds) of the Binomial Tree Model Across Tree Heights

Crank-Nicolson

The Crank-Nicolson method was tested on parameters derived from GOOG and APPL calls and puts. Table 3 shows these parameters. Table 4**Error! Reference source not found.** present the runtimes for each grid arrangement. The stock increments were tested on 50¢, 25¢, 16.67¢,

[1] In years to expiration.
[2] The Google options studied are based upon Google's class-C capital stock.
[3] The 52-week averages of IV30 implied volatility as stated on Market Chameleon (2023) at the time of testing.
[4] The risk-free rate chosen is the current 10-year treasury interest rate at the time of testing (CNBC, 2023).
[5] Stock prices as stated on MarketWatch **Invalid source specified.** at the time of testing.

12.5¢, 10¢, and 1¢ intervals. The time increments tested were monthly, weekly, daily, and hourly.

	T	σ	K	r	q	S_{max}	S_{min}
GOOG	2	35.9	105	3.585%	0.01%[6]	500	0
AAPL	2	32.6	170	3.585%	5.5%	500	0

Table 3: Crank-Nicolson Parameters

Number of Time Nodes (n_t)		**Number of Stock Price Nodes (n_S)**				
		1000	**2000**	**3000**	**4000**	**5000**
Monthly	**24**	1.2450	4.9096	16.2104	36.0895	66.6167
Weekly	**104**	1.1965	5.1169	16.7891	37.1528	67.1918
Daily	**730**	1.4783	6.7505	24.4833	45.3177	84.1909
Hourly	**17,520**	7.6092	56.8880	130.5930	237.6383	360.4237

Table 4: Crank-Nicolson Runtimes (seconds) on Google Call Settings

Number of Time Nodes (n_t)		**Number of Stock Price Nodes (n_S)**				
		1000	**2000**	**3000**	**4000**	**5000**
Monthly	**24**	1.4172	5.2898	14.9086	35.1306	63.4802
Weekly	**104**	1.3650	5.3386	15.3934	36.1990	64.8750
Daily	**730**	1.7584	6.9092	19.2119	42.1976	74.3146
Hourly	**17,520**	7.5329	50.9348	118.3217	212.6308	337.5909

Table 5: Crank-Nicolson Runtimes (seconds) on Google Put Settings

Number of Time Nodes (n_t)		**Number of Stock Price Nodes (n_S)**				
		1000	**2000**	**3000**	**4000**	**5000**
Monthly	**24**	1.2574	4.8909	15.8852	36.3142	65.4320
Weekly	**104**	1.2342	5.0626	19.6893	37.4823	67.4786
Daily	**730**	1.4209	6.6422	22.3536	48.5047	79.1061
Hourly	**17,520**	7.9945	63.1015	130.9094	231.9470	341.4307

Table 6: Crank-Nicolson Runtimes (seconds) on Apple Call Settings

Number of Time Nodes (n_t)		**Number of Stock Price Nodes (n_S)**				
		1000	**2000**	**3000**	**4000**	**5000**
Monthly	**24**	1.5649	5.1809	14.9189	35.1784	62.9605
Weekly	**104**	1.5021	5.4341	15.6894	36.2442	64.8324

[6] Instability, discussed below, forced the use of 0.01% instead of the true no-dividend policy.

Daily	**730**	1.6865	6.9028	19.0056	42.0671	74.5212
Hourly	**17,520**	7.7176	50.8797	121.8095	210.8439	340.4933

Table 7: Crank-Nicolson Runtimes (seconds) on Apple Put Settings

SERK2v2

The SERK2v2 method was tested on stock data from the GOOG and APPL historical stock. The data used ranges from 2021 to 2023. Table 8 shows the parameters used in numerical experimentation. We used the 10-year U.S. treasury bond yield as the risk-free rate, *r*, and roughly based the other parameters around optimizing S=K to be as close to 0 as possible. The run time for varying stages is shown in Table 9 using the Apple stock for 20 stages. The algorithm was written in Python 3.10.6 on a system with the following specifications:

- WSL Ubuntu Version 2
- x86 architecture, Little Endian byte order
- AMD Ryzen 5 Pro 3500U
    - 4 cores (2.1GHz – 3.7GHz); 2 threads per core
    - 384KB L1
    - 2MB L2
    - 4MB L3

We additionally calculated the implied volatility σ_I produced by the predicted option values.

	T	***s_stages***	**σ**	***K***	***r***	***M***	***N***
AAPL	2	20	18.4076	140	3.585%	500	500
GOOG	2	20	13.6867	100	3.585%	500	500

Table 8: SERKv2 Parameters

Time Steps (M)	**Stock Price Steps (N)**	σ_I	**Run Time (sec)**
50	50	6.4320	0.0489
100	100	11.2777	0.1685
150	150	9.7765	0.1646
200	200	10.7519	0.2316
250	250	10.6569	0.2712
300	300	11.9730	0.3146
350	350	12.9920	0.4369
400	400	12.5641	0.4905
450	450	12.0811	0.4884
500	500	13.6867	0.5502

Table 9: SERK2v2 Results

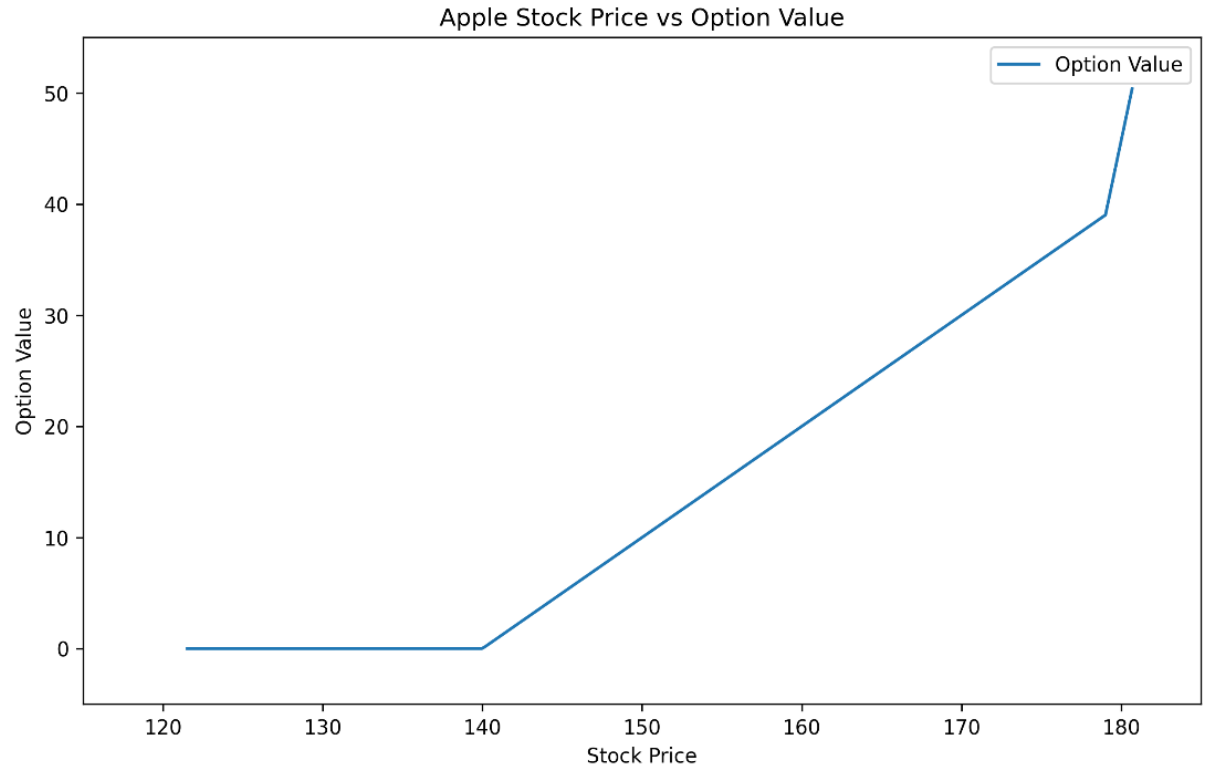

Figure 3: SERK2v2 APPL K=140 Stock Price vs Option Value at K =140

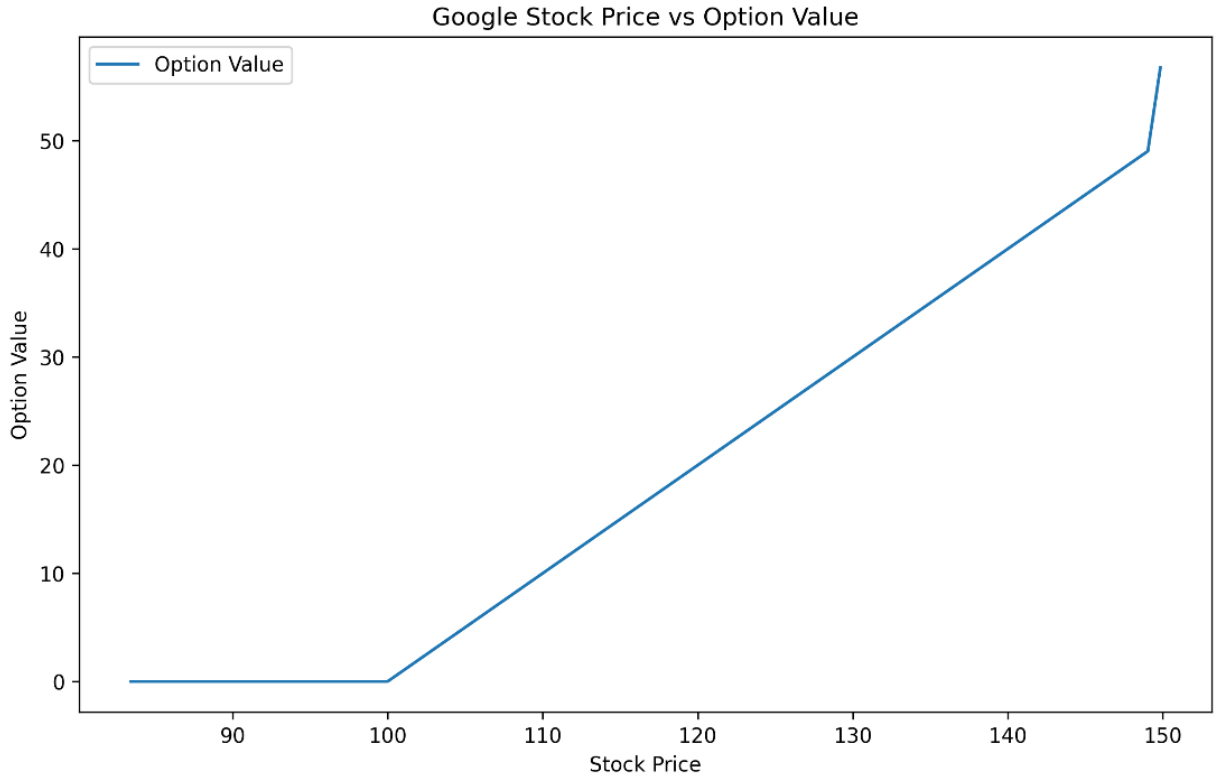

Figure 4: SERK2v2 GOOG Stock Price vs Option Value at K = 100

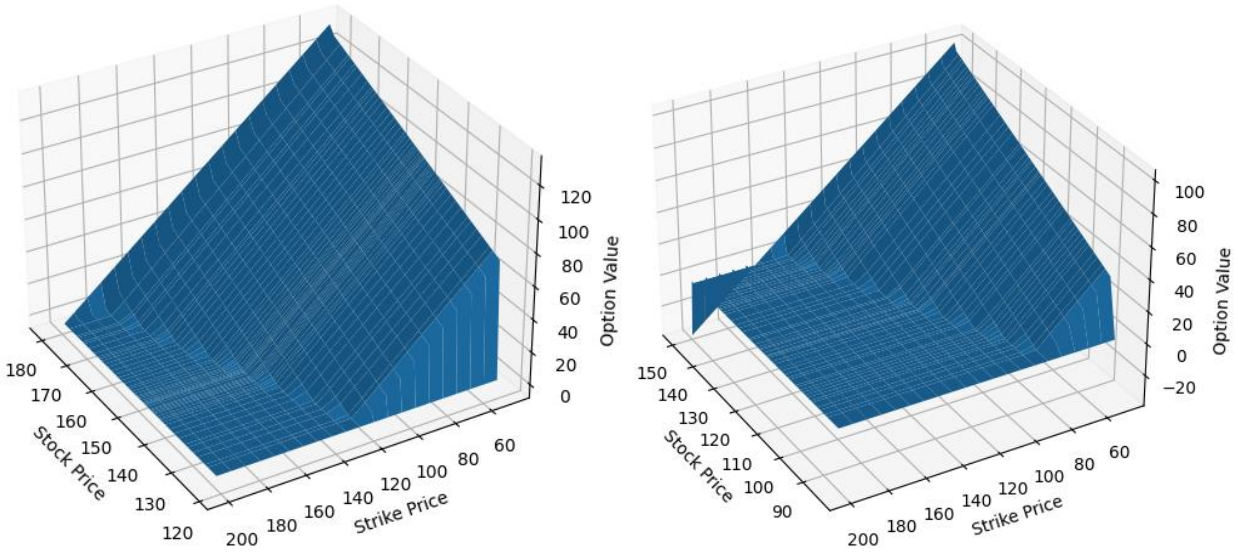

Figure 5: SERK2v2 AAPL (Left) and GOOG (Right) Stock Price and Strike Price vs. Option Value

Discussion and Conclusions

The Binomial Tree model was much slower than initially anticipated. The method's stability performed as expected; however, the algorithm's runtime proved to be a barrier to the method's usage when using a fine timestep. Perhaps the Binomial Tree would perform well for volatile options with a shorter time horizon, as this would allow for a shorter tree height.

Google's Class-C stock, which is the stock on which the Google options are based, does not pay dividends. However, in our experiments, we found that using any $q < 0.0001 \implies \|B^{n_t}\|_\infty \gg 1$, or in other words, the Crank-Nicolson method was unstable for any value of q smaller than 0.01%, which is an interesting finding. Forsyth, Windcliff, & Vetzal (2004) stated the method is stable for any $q \geq 0$, using their definition of stability; however, they did not explicitly state that this method works with American-style options. On the other hand, the early-exercise feature is the only difference in using this method to model American-Style options from what was stated in Forsyth, Windcliff, & Vetzal (2004). Checking for early exercise happens much after *B* is defined and checked for stability. Additionally, there weren't any stated restrictions on the other parameters that would cause such an instability near $q = 0$.

The Crank-Nicolson method performed much faster than the initial expectation of $\mathcal{O}(n_S^4)$, however, the method is still practically slow. The optimal use case for this method would be for options with larger time horizons that don't require cent-precision or are not highly volatile, as the algorithm's runtime scales much slower with respect to n_t than n_S.

In this implementation of the Stabilized Explicit Runge-Kutta method (SERK2v2) for American call options on Google and Apple stocks, the algorithm provided significant results for the range of stock prices. The resulting graphs are displayed as expected, demonstrating that the SERK2v2 method can handle the early exercise feature, the defining aspect of American options. The

SERK2v2 algorithm proved to be efficient and optimized as well, as varying M and N values negligibly changes the run time in the context of the data we used. It is also worth noting that the option value becomes negative for Google stock data in Figure 5 as the stock price never reaches the higher values of strike prices. This results in negative option values which are not economically meaningful or relevant and can therefore be discarded if he used in a real-time practical environment.

Future Work

It is important to note that Martin-Vaquero, Kahliq, and Kleefeld focused on multi-asset American options, whereas our implementation focuses on single-asset American call options. Theoretically, SERKv2 should be stable and work with American put options; however, more numerical testing should confirm this. Additionally, further numerical experiments should be performed comparing performance against various values of s and mesh sizes.

In the future, we plan to test the numerical approximation of the Black-Scholes PDE with different numerical methods, such as finite element and spectral methods. Another area of exploration is optimizing the investigated numerical methods, which could entail code refactoring, parallel processing, and using a lower-level language such as C++, among other things. These improvements and adjustments will allow a more accurate approximation of the Black-Scholes PDE.

Calculating errors is also a field of interest for American options; a long-term study should be performed to determine the accuracy of these models against real-time American option prices. However, when computing real errors, the assumptions that the Black-Scholes PDE holds (such as the no-arbitrage principle) are also true for these numerical methods, which have a reputation for not being realistic. Further work should be done to find numerical methods that circumvent these assumptions.

References

Black, F., & Scholes, M. (1973). The Pricing of Options and Corporate Liabilities. *Journal of Political Economy*, 637-654.

Bouchard, B., Chau, K. W., Manai, A., & Sid-Ali, A. (2019, February). MONTE-CARLO METHODS FOR THE PRICING OF AMERICAN OPTIONS: A SEMILINEAR BSDE POINT OF VIEW. *ESAIM: PROCEEDINGS AND SURVEYS, 65*, 294-308.

Chen, J. (2022, March 31). *American Option Definition, Pros & Cons, Examples*. Retrieved from Investopedia: https://www.investopedia.com/terms/a/americanoption.asp

CNBC. (2023, April 18). *U.S. 10-Year Treasury*. Retrieved from CNBC: https://www.cnbc.com/quotes/US10Y

Cox, J. C., Ross, S. A., & Rubinstein, M. (1979). Option Pricing: A Simplified Approach. *Journal of Financial Economics 7*, 229-263.

Crank, J., & Nicolson, P. (1946). A Practical Method for Numerical Evaluation of Solutions Partial Differential Equations of the Heat-Conduction Type. *Mathematical Proceedings of the Cambridge Philosophical Society*, 50-67.

Forsyth, P. A., Windcliff, H., & Vetzal, K. R. (2004). Analysis of the stability of the linear boundary condition for the Black-Scholes equation. *Journal of Computational Finance*, 65-92. doi:10.21314/JCF.2004.116

Hull, J. C. (2021). The Black-Scholes-Merton Model. In *Options, Futures, and Other Derivatives* (11th ed.). Pearson.

Market Chameleon. (2023, April 18). *AAPL Implied Volatility (IV) vs Historical Volatility (HV)*. Retrieved from Market Chameleon: https://marketchameleon.com/Overview/AAPL/IV/

Martin-Vaquero, J., Khaliq, A., & Kleefeld, B. (2014). Stabilized explicit Runge-Kutta methods for multi-asset American options. *Computers and Mathematics with Applications*, 1293-1308.

Rubinstein, M. (2000). On the Relation Between Binomial and Trinomial Option Pricing Models. *Journal of Derivatives*.

Song, H., Zhang, R., & Tian, W. (2014). Spectral Method for the Black-Scholes Model of American Options Valuation. *J. Math. Study*, 47-64.