Numerical Solution for PDE with Non-Local Boundary Condition and its application for American options pricing problem

by

Vishal Meena

Department of Mathematics and Computing

Submitted

in partial fulfillment of the requirements of the degree of Masters of technology

to the

INDIAN INSTITUTE OF TECHNOLOGY DELHI

MONTH 2023

Certificate

This is to certify that the thesis entitled **"Numerical Solution for PDE with Non-Local Boundary Condition and its application for American options pricing problem"**, submitted by **Vishal Meena** to the Indian Institute of Technology Delhi, for the award of the degree of **Masters in technology** in Mathematics and Computing , is a record of the original, bona fide research work carried out by him under our supervision and guidance. The thesis has reached the standards fulfilling the requirements of the regulations related to the award of the degree.

The results contained in this thesis have not been submitted in part or in full to any other University or Institute for the award of any degree or diploma to the best of our knowledge.

Prof. Dr. S. Chandra Sekhara Rao
Department of Mathematics,
Indian Institute of Technology Delhi.

Abstract

In this thesis, we focus on the treatment of non-local boundary conditions in numerical methods. Non-local boundary conditions arise in various mathematical and physical problems, where the behavior at a boundary is influenced by the values of the solution at different points in the domain. The presence of non-local boundary conditions poses challenges requiring complex numerical methods which are computationally expensive.

The main objective of this thesis is to develop and analyze numerical methods for solving problems with non-local boundary conditions. We aim to demonstrate the stability and accuracy of these methods in handling such conditions. Specifically, we investigate the application of these methods to the problem of pricing American options.

American option pricing involves determining the optimal exercise strategy for an option contract that can be exercised at any time before its expiration. The presence of non-local boundary conditions in this context complicates the pricing process. Therefore, we propose approaches to approximate these non-local boundary conditions using various numerical techniques.

Through extensive computational experiments, we compare the accuracy and efficiency of different approximation methods for non-local boundary conditions. We evaluate their performance and identify the most accurate method suitable for American option pricing problems. By employing this method, we can significantly reduce computational efforts while maintaining high accuracy in pricing American options.

The findings of this research contribute to the understanding and advancement of numerical methods for solving problems with non-local boundary conditions. The developed techniques offer reliable and efficient tools for addressing such conditions in practical applications, particularly in the context of American option pricing.

Contents

List of Figures

Chapter 1

Introduction

1.1 Non-local boundary conditions

Non-local boundary conditions in numerical methods for mathematics are a class of boundary conditions that involve interactions between different regions of the domain or boundary of a problem. These conditions stand in contrast to local boundary conditions, which depend solely on the values of the function and its derivatives at specific points on the boundary. Non-local boundary conditions introduce additional complexity in the numerical solution of partial differential equations (PDEs) and require specialized techniques to handle these conditions.

In many mathematical problems, particularly those involving PDEs, boundary conditions play a crucial role in determining the behavior of the solution. Local boundary conditions, such as Dirichlet, Neumann, or Robin conditions, are relatively straightforward to implement in numerical methods, as they involve only the values of the function or its derivatives at specific points on the boundary. However, non-local boundary conditions involve integrals or other non-local operators over the domain or boundary, making them more challenging to handle in numerical methods.

Non-local boundary conditions can arise in various applications, such as in problems involving long-range interactions, fractional-order PDEs, or integral equations. These conditions can be found in areas such as fluid dynamics, heat conduction, financial problems, materials science, and population dynamics, among others. The presence of non-local boundary conditions often leads to unique mathematical properties and behaviors in the solutions of the associated PDEs. To address the challenges posed by non-local boundary conditions, researchers have developed a variety of specialized numerical methods. Some of these methods include integral equation approaches, where the original PDE is reformulated as an integral equation, and the non-local boundary conditions are incorporated into the integral operator. This approach can simplify the implementation of non-local boundary conditions and facilitate the use of standard numerical techniques, such as finite element or finite difference methods, for solving the resulting integral equation.

Another approach to handling non-local boundary conditions is the use of meshless or collocation methods, which do not rely on a fixed grid or mesh for discretizing the problem domain. These methods can offer greater flexibility in handling non-local boundary conditions, as they can more easily accommodate the non-local interactions between different regions of the domain or boundary.

In some cases, non-local boundary conditions can be transformed into local boundary conditions through the use of auxiliary variables or reformulations of the problem. This can simplify the numerical treatment of the problem and allow for the use of standard numerical methods. However, such transformations may not always be possible or may introduce additional complexity in the problem formulation.

1.2 American option

The American options pricing problem refers to the mathematical challenge of determining the fair value of an American-style option. American options are a type of financial derivative contract that gives the holder the right, but not the obligation, to buy or sell an underlying asset at a predetermined price (the strike price) on or before a specified expiration date.

Unlike European options, which can only be exercised at expiration, American options can be exercised at any time prior to expiration. This flexibility introduces additional complexity into the pricing problem. The main question is when to exercise the option to maximize its value. Should the option holder exercise it immediately or wait until a later date?

The pricing problem arises because the option's value depends on several factors, including the current price of the underlying asset, the strike price, the time to expiration, the volatility of the underlying asset, and the prevailing risk-free interest rate. Determining an accurate valuation for American options requires taking into account these variables and incorporating them into a mathematical model.

1.2.1 Modeling American Options as Parabolic PDEs with Non-Local Boundary Conditions

American options are financial derivatives that grant the holder the right, but not the obligation, to buy or sell an underlying asset at a predetermined price (strike price) on or before a specified date (expiration date). The value of an American option depends on the price of the underlying asset, the strike price, the time to expiration, and other factors such as interest rates and volatility. Mathematically, American

options can be modeled as a type of 1D parabolic partial differential equation (PDE) with a non-local boundary condition. The PDE describes the evolution of the option price over time, while the boundary condition specifies the value of the option at the expiration date. The PDE for American options is typically the Black-Scholes equation, which is a parabolic PDE that describes the evolution of the option price as a function of time and the underlying asset price. The non-local boundary condition arises because the holder of an American option has the right to exercise the option at any time before expiration, which means that the boundary condition depends on the optimal exercise strategy of the holder. The non-local boundary condition is often referred to as the early exercise boundary condition, and it is a free boundary that separates the region where the option is exercised early from the region where it is held until expiration. The optimal exercise strategy depends on the underlying asset price, the time to expiration, and other factors, and it is typically determined using numerical methods such as finite difference or Monte Carlo simulations. The non-local boundary condition makes the problem of pricing American options more challenging than the problem of pricing European options, which have a simpler boundary condition. However, the ability to exercise the option early gives American options a higher value than European options, all else being equal.

1.3 Thesis Organisation

In this thesis, we explore the concept of non-local boundary conditions and their prevalence in heat conduction equations. Non-local boundary conditions arise when the value of a dependent variable at a boundary point depends on the values of the variable at other points in the domain. These conditions are commonly encountered in various physical and mathematical problems, including heat conduction equations,

which are essential for understanding heat transfer processes in various materials and systems. In Chapter 2, we present a numerical method for solving heat conduction equations with non-local boundary conditions, along with a proof of its stability. This method is based on a systematic approach that combines discretization techniques and iterative solvers to efficiently handle the non-local nature of the boundary conditions. The stability proof ensures that the numerical method converges to the correct solution and provides a solid foundation for its application in more complex problems. Chapter 3 focuses on the general stability analysis of numerical methods for solving one-dimensional parabolic problems with non-local boundary conditions. We discuss the necessary conditions for stability and provide guidelines for selecting appropriate numerical methods that guarantee accuracy in solving these problems. Chapters 4 delves into the mathematical formulation of the problem, explaining how the backward-in-time American options pricing problem can be transformed into a forward-in-time heat conduction problem. This transformation is discussed in detail, emphasizing its implications for the overall problem and the advantages it offers in terms of numerical solution techniques and in Chapter 5 the numerical method employed to solve the transformed problem efficiently is presented. The focus is on approximating the coefficients of the non-local boundary condition, which are computationally expensive to evaluate. Three different approximation methods are examined: Padé approximation, rational function approximation, and Economized Rational Approximation. The characteristics, strengths, and limitations of each method are discussed. The accuracy and computational efficiency of each method are assessed through error graph. Graphical representations of the errors help determine the best approximation method for the problem at hand. This analysis lays the foundation for selecting the most suitable approximation method for further applications in solving American option pricing problems. the chosen approximation method is applied to the transformed problem to solve for the American option

values. The numerical solution procedure is explained in detail, including the implementation of the approximation method and the computational aspects involved. Finally, real-world numerical experiments are conducted using AAPL stock to determine the option values. The experimental setup and methodology are explained, and the obtained results. This analysis provides insights into the practical applicability and reliability of the developed numerical method for real-world financial problems. By the end of this paper, we will be having a comprehensive understanding of non-local boundary conditions, their occurrence in heat conduction equations, and their applications in American option pricing. Furthermore, we will be equipped with a stable numerical method for solving these problems and the knowledge to analyze the stability of other numerical methods for one-dimensional parabolic problems with non-local boundary conditions.

Chapter 2

A Numerical Method for Heat Equation with Non-Local Boundary Condition

2.1 1D parabolic equations

1D parabolic equations are a class of partial differential equations (PDEs) that describe the evolution of a scalar quantity over time and space in one spatial dimension. These equations are characterized by their parabolic nature, which means that they exhibit a balance between diffusion and reaction processes. The most well-known example of a 1D parabolic equation is the 1D heat equation, which models the distribution of heat in a one-dimensional rod.

The general form of a 1D parabolic equation can be written as:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + f(x, t, u)$$

where $u(x,t)$ is the dependent variable representing the scalar quantity of interest (e.g., temperature, concentration), x is the spatial coordinate, t is time, and $f(x,t,u)$ is a function that represents the reaction or source term. The first term on the right-hand side, $\frac{\partial^2 u}{\partial x^2}$, represents the diffusion process, while the second term, $f(x,t,u)$, accounts for any reactions or sources that may be present.

2.2 Problem statement

Consider a 1D heat equation with nonlocal boundary conditions:

$$u_t = u_{xx} + q(t,x), \quad t \geq 0, \quad x \in [0,1], \quad u(0,x) = f(x) \tag{2.1}$$

$$u(t,0) = \int_0^1 \phi(x) u(t,x)\, dx \tag{2.2}$$

$$u(t,1) = \int_0^1 \psi(x) u(t,x)\, dx, \tag{2.3}$$

where $q(t,x)$ is the heat generation, and $f(x)$ is the initial condition.

The equation describes the evolution of temperature $u(t,x)$ over time and space in a one-dimensional rod. The first equation on the right-hand side represents the diffusion process, while the second term $q(t,x)$ accounts for any heat generation. The nonlocal boundary conditions at $x = 0$ and $x = 1$ involve integrals of the temperature over the entire domain, weighted by the functions $\phi(x)$ and $\psi(x)$, respectively.

Solving this equation requires specialized numerical methods that can handle the nonlocal boundary conditions. Some possible approaches include integral equation methods, meshless methods, or finite difference methods with appropriate modifications to handle the nonlocal terms.

2.2.1 Method of solving the given problem statement

Assume that the domain is divided into m equal intervals

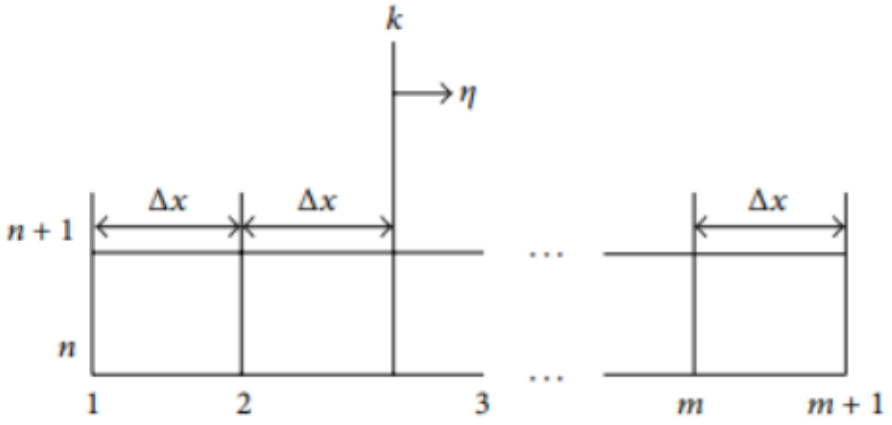

FIGURE 2.1: Finite difference mesh showing the local coordinate

For a common node k, an implicit expansion of the equation at time interval $(n+1)$

$$\frac{u_k^{n+1} - u_k^n}{\Delta t} = u_{xx} + q \tag{2.4}$$

where the value of u_k^n is known. Now, considering a transformation to the local coordinate η:

$$\eta = \frac{x - x_k}{\Delta x} \tag{2.5}$$

which implies

$$x = \eta \Delta x + x_k \tag{2.6}$$

Differentiating both sides with respect to η gives

$$\frac{\partial x}{\partial \eta} = \Delta x \tag{2.7}$$

As $\frac{\partial u}{\partial \eta} = \frac{\partial u}{\partial x}\frac{\partial x}{\partial \eta}$, differentiating again gives

$$\frac{\partial^2 u}{\partial \eta^2} = \frac{\partial^2 u}{\partial x^2}\left(\frac{\partial x}{\partial \eta}\right)^2 = \Delta x^2 u_{xx} \tag{2.8}$$

Substituting this into the original equation and using the local coordinate η gives

$$u_{\eta\eta} - \lambda^2 u = -\lambda^2 u_k^n - \Delta x^2 q_k, \tag{2.9}$$

where $\eta \in [-1, 1]$ and $\lambda^2 = \frac{\Delta x^2}{\Delta t}$. The variable q_k denotes the heat generation function $q(t, x)$ evaluated at time $(\Delta t(n+1))$ and location k. Generally, it is a function of η; however, for a first-order accurate method, the value at $x = (k-1)\Delta x$ can be used. Using Taylor series expansion, it is possible to obtain a more accurate description in terms of η that will be

$$\sum_{l=0}^{\infty} a_l \eta^l \tag{2.10}$$

for which we can solve for the exact solution. Using q_k as a local constant in η function, we can create the exact solution of $u(t, \eta)$ around every node given by

$$u(t, \eta) = c_k e^{-\lambda\eta} + d_k e^{\lambda\eta} + \zeta_k^{n+1}, \tag{2.11}$$

where

$$\zeta_k^{n+1} = u_k^n + \Delta t q_k^{n+1}. \tag{2.12}$$

We can observe from this equation that the constants of integration, c_k and d_k, are independent of time and location. For m intervals, there will be $m+1$ nodes, and hence

$m+1$ similar equations that denote the temperature profile, where $k = 1, 2, 3, \ldots, M$. Therefore, there will be $2M$ constants, c_k and d_k (where $k = 1, 2, 3, \ldots, M$), which are unknown and need to be calculated at each time interval. It is possible to introduce $2M$ conditions that are linearly independent. However, these conditions are not unique. Despite this, a set of linearly independent conditions can be used to determine the constants of integration.

The following set of conditions describes the situation:

1. The temperature distribution at $t_{n+1} = (n+1)\Delta t$ should be continuous. Imposing this condition at the mid-point between nodes gives

$$c_k e^{-\lambda(1/2)} + d_k e^{\lambda(1/2)} + \zeta_k^{n+1} c_{k+1} e^{-\lambda(1/2)} + d_{k+1} e^{\lambda(1/2)} + \zeta_{k+1}^{n+1} \tag{2.13}$$

for $k =, 2, 3, \ldots, m$.

2. The flux at t_{n+1} should be continuous. In the present case, it follows that the temperature at t_{n+1} should have a continuous first derivative. Imposing this condition at the mid-point between nodes leads to

$$c_k e^{\lambda(1/2)} - d_k e^{-\lambda(1/2)} = c_{k+1} e^{-\lambda(1/2)} -_{k+1} e^{\lambda(1/2)} \tag{2.14}$$

for $k = 1, 2, 3, \ldots, m$. The conditions furnish $2m$ linearly independent equations. The additional conditions needed are given by the boundary conditions. The solution must satisfy the boundary conditions at $x = 0$ and $x = 1$. These conditions are nonlocal and can be imposed according to

$$u(t_{n+1}, 0) = c_1 + d_1 + \zeta_1^{n+1} = \int_0^1 \phi(x) u(t_{n+1}, x) dx \tag{2.15}$$

$$u(t_{n+1}, 1) = c_M + d_M + \eta_M^{n+1} = \int_0^1 \psi(x) u(t_{n+1}, x) dx, \tag{2.16}$$

This integral can be solved using computed in terms of the solution according to convention. For the first node, the domain of integration can be chosen to be $[0, \frac{\Delta x}{2}]$. For the nodes $k = 2, 3, 4 \ldots m$. The domains can be chosen to be $\left[(K-1)\Delta x - \left(\frac{\Delta x}{2}\right), (K-1)\Delta x + \left(\frac{\Delta x}{2}\right)\right]$. For the last node, the domain can be chosen as $\left[1 - \left(\frac{\Delta x}{2}\right), 1\right]$. In terms of the local coordinates η, for the interior nodes, the individual domains of integrations are all $\left[-\frac{1}{2}, \frac{1}{2}\right]$. Similarly, the domain of integration is $\left[0, \frac{1}{2}\right]$ for the first node and $\left[-\frac{1}{2}, 0\right]$ for the last node.

Using these domains, the integral condition at $x = 0$ will be

$$\begin{aligned} c_1 + d_1 + \xi_1^{n+1} &= \int_0^1 \phi(x) u(t_{n+1}, x) dx \\ &= \Delta x \int_0^{\frac{1}{2}} \phi(\eta) \left[c_1 e^{\lambda\eta} + d_1 e^{-\lambda\eta} + \xi_1^{n+1}\right] d\eta \\ &\quad + \sum_{k=2}^{m} \Delta x \int_{-\frac{1}{2}}^{\frac{1}{2}} \phi(\eta) \left[c_k e^{\lambda\eta} + d_k e^{-\lambda\eta} + \xi_k^{n+1}\right] d\eta \\ &\quad + \Delta x \int_{-\frac{1}{2}}^{0} \phi(\eta) \left[c_M e^{\lambda\eta} + d_M e^{-\lambda\eta} + \xi_M^{n+1}\right] d\eta \end{aligned}$$

And the function $\phi(x)$ is known, so the above integrals can be evaluated in closed forms. Hence, we will get

$$\begin{aligned}(1-\Delta x\int_0^{\frac{1}{2}}\phi(\eta)e^{\lambda\eta}d\eta)c_1+(1-\Delta x\int_0^{\frac{1}{2}}\phi(\eta)e^{-\lambda\eta}d\eta)d_1\\ -\sum_{k=2}^{m}\Delta x\left(\int_{-\frac{1}{2}}^{\frac{1}{2}}\phi(\eta)e^{\lambda\eta}d\eta\right)c_k\\ -\sum_{k=2}^{m}\Delta x\left(\int_{-\frac{1}{2}}^{\frac{1}{2}}\phi(\eta)e^{-\lambda\eta}d\eta\right)d_k\\ -\Delta x\left(\int_0^{-\frac{1}{2}}\phi(\eta)e^{\lambda\eta}d\eta\right)c_M\\ -\Delta x\left(\int_0^{-\frac{1}{2}}\phi(\eta)e^{\lambda\eta}d\eta\right)d_M\\ =-\xi_1^{n+1}+\int_0^1\phi(x)\xi^{n+1}dx\end{aligned}$$

And for $x=1$, it is given by

$$\begin{aligned}c_M+d_M+\xi_M^{n+1}&=\int_0^1\psi(x)u(t_{n+1},x)dx\\ &=\Delta x\int_0^{\frac{1}{2}}\psi(\eta)\left[c_1e^{\lambda\eta}+d_1e^{-\lambda\eta}+\xi_1^{n+1}\right]d\eta\\ &\quad+\sum_{k=2}^{m}\Delta x\int_{-\frac{1}{2}}^{\frac{1}{2}}\psi(\eta)\left[c_ke^{\lambda\eta}+d_ke^{-\lambda\eta}+\xi_k^{n+1}\right]d\eta\\ &\quad+\Delta x\int_{-\frac{1}{2}}^{0}\psi(\eta)\left[c_Me^{\lambda\eta}+d_Me^{-\lambda\eta}+\xi_M^{n+1}\right]d\eta\end{aligned}$$

$\psi(x)$ is also known, so the above integral can be evaluated in closed form. Hence, we get

$$
\begin{aligned}
&- (\Delta x \int_0^{\frac{1}{2}} \psi(\eta) e^{\lambda\eta} d\eta) c_1 - (\Delta x \int_0^{\frac{1}{2}} \psi(\eta) e^{-\lambda\eta} d\eta) d_1 \\
&- \sum_{k=2}^{m} \Delta x \left(\int_{-\frac{1}{2}}^{\frac{1}{2}} \psi(\eta) e^{\lambda\eta} d\eta \right) c_k \\
&- \sum_{k=2}^{m} \Delta x \left(\int_{-\frac{1}{2}}^{\frac{1}{2}} \psi(\eta) e^{-\lambda\eta} d\eta \right) d_k \\
&+ \Delta x \left(1 - \int_0^{-\frac{1}{2}} \psi(\eta) e^{\lambda\eta} d\eta \right) c_M \\
&+ \Delta x \left(1 - \int_0^{-\frac{1}{2}} \psi(\eta) e^{\lambda\eta} d\eta \right) d_M \\
&= -\xi_1^{n+1} + \int_0^1 \psi(x) \xi^{n+1} dx
\end{aligned}
$$

The two criteria mentioned above supply the final two equations needed for the coefficient matrix to be inverted reliably.

2.3 Proof of stability

We have 2M unknown coefficients $c_k, d_k, k = 1, 2, \ldots$ M, at each time increment which can be solved with the above 2M linearly independent equations. To demonstrate the invertibility of the coefficient matrix, we choose the following case $\phi(x) = \psi(x) = 1$ Other kernels can be treated in a similar way. The appropriate way to group the above condition will be grouping them in the following order. Equation (3) which is evaluated at $\eta = \frac{1}{2}, or, x = \frac{\Delta x}{2}$ in the first row. Next, we can place (4) evaluated at $\eta = \frac{1}{2}$ in the second row. Following this pattern and evaluating both equations for every mid-point will form (2m) equations that will be placed in the

first (2m) rows of the matrix. The last two rows can be used to impose the boundary conditions given in (5) and (6). Using this manner, it is possible to group the above equations at every time interval t_{n+1} in the form of $Av^{n+1} = g^{n+1}$,

$$\begin{bmatrix} e^{\theta} & e^{-\theta} & -e^{-\theta} & -e^{\theta} & 0 & 0 & \cdots & \cdots \\ 0 & 0 & e^{\theta} & -e^{-\theta} & -e^{-\theta} & e^{\theta} & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & 0 & 0 & e^{\theta} & e^{-\theta} & -e^{-\theta} & -e^{\theta} \\ \cdots & \cdots & 0 & 0 & e^{\theta} & -e^{-\theta} & -e^{-\theta} & e^{\theta} \\ \mu_1^0 & v_1^0 & \epsilon & \epsilon & \cdots & \cdots & \mu_M^0 & v_M^1 \\ \mu_1^1 & v_1^1 & \epsilon & \epsilon & \cdots & \cdots & \mu_M^1 & v_M^1 \end{bmatrix}$$

$$\times \begin{bmatrix} c_1 \\ d_1 \\ c_2 \\ d_2 \\ c_3 \\ d_3 \\ \vdots \\ c_m \\ d_m \\ c_M \\ d_M \end{bmatrix} = \begin{bmatrix} g_1 \\ 0 \\ g_2 \\ 0 \\ g_3 \\ 0 \\ \vdots \\ g_m \\ 0 \\ \sigma \\ \delta \end{bmatrix}$$

Where $\theta = \frac{\lambda}{2}$, The unknown vector v_{n+1} contains the constants of integration at every time interval, and g_{n+1} is the known right-hand side. The constants on the

right-hand side are given by $g_k = \frac{\eta_{k+1}^{n+1} - \eta_k^{n+1}}{\Delta x}$, for $k = 1, 2, 3, \ldots$ $\sigma = (-1 + \frac{\Delta x}{2})\eta_1^{n+1} + \frac{\Delta x}{2}\sum_{k=2}^{m}\eta_k^{n+1} + \frac{\Delta x}{2}\eta_M^{n+1}$ $\delta = \frac{\Delta x}{2}\eta_1^{n+1} + \frac{\Delta x}{2}\sum_{k=2}^{m}\eta_k^{n+1} + (-1 + \frac{\Delta x}{2})\eta_M^{n+1}$

And the parameters $\mu_1^0, \mu_1^1, and v_M^0, v_M^1$ are given by

$$\begin{aligned}
\mu_1^0 &= 1 - \frac{\Delta x}{\lambda}(e^{\theta} - 1), & v_1^0 &= 1 + \frac{\Delta x}{\lambda}(e^{-\theta} - 1),\\
\mu_M^0 &= -\frac{\Delta x}{\lambda}(1 - e^{-\theta}), & v_M^0 &= \frac{\Delta x}{\lambda}(1 - e^{\theta}),\\
\epsilon &= \frac{\Delta x}{\lambda}(e^{-\theta} - e^{\theta}), & &\\
\mu_1^1 &= -\frac{\Delta x}{\lambda}(e^{\theta} - 1), & v_1^1 &= \frac{\Delta x}{\lambda}(e^{-\theta} - 1),\\
\mu_M^1 &= 1 - \frac{\Delta x}{\lambda}(1 - e^{-\theta}), & v_M^1 &= 1 + \frac{\Delta x}{\lambda}(1 - e^{\theta})
\end{aligned}$$

2.3.1 Determinant of A matrix

determinant of A matrix can be find using the block matrix determinant method, before applying this method we need to partition the above coefficient matrix according to

$$\begin{bmatrix}
D & C & 0 & 0 & \cdots & 0\\
0 & D & C & 0 & \cdots & 0\\
\vdots & \vdots & \ddots & \ddots & \ddots & \vdots\\
0 & 0 & \cdots & 0 & D & C\\
E_1 & F & \cdots & \cdots & F & E_m
\end{bmatrix}$$

or,

$$\begin{bmatrix}
\Gamma_{11} & \Gamma_{12}\\
\Gamma_{21} & \Gamma_{22}
\end{bmatrix}$$

Where the submatrices $D, C, E_1, E_m, andF$ are (2x2) matrices. The partitioned matrices are

$$\Gamma_{11} = \begin{bmatrix} D & C & 0 & 0 & \cdots & 0 \\ 0 & D & C & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & D & C \end{bmatrix}, \Gamma_{12} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ C \end{bmatrix}, \Gamma_{21} = \begin{bmatrix} E_1 & F & \cdots & F \end{bmatrix}, \text{ and } \Gamma_{22} = E_m.$$

Now, using block matrix determinant method, we get

$$\det(A) = \det(\Gamma_{11}) \det(\Gamma_{22} - \Gamma_{21}\Gamma_{11}^{-1}\Gamma_{12})$$

It is sufficient to show that the two determinants on the right-hand side in the above equation are non-zero. The matrix Γ_{11} can be inverted since it is an upper block matrix, and

$$\det(D) = -2 \neq 0.$$

In order to show that the second determinant is nonzero, we can say some $G = D^{-1}C$ and note that

$$\Gamma_{11}^{-1} = \begin{bmatrix} D^{-1} & -[GD^{-1}] & [G^2D^{-1}] & \cdots \\ 0 & D^{-1} & -[GD^{-1}] & [G^2D^{-1}] \\ \vdots & \vdots & D^{-1} & -[GD^{-1}] \\ 0 & 0 & 0 & D^{-1} \end{bmatrix}$$

where $G = -H = -\begin{bmatrix} e^{-} & 0 \\ 0 & e^{\lambda} \end{bmatrix}$, $G_l = (-1)^l H^l = -\begin{bmatrix} e^{-l\lambda} & 0 \\ 0 & e^{l\lambda} \end{bmatrix}$. Hence, the second determinant on the right-hand side can be simplified to:

$$\det(\Gamma_{22} - \Gamma_{21}\Gamma_{11}^{-1}\Gamma_{12}) = \det\left(\mathbf{E_m} + \mathbf{E_1}\mathbf{H^m} + F\sum_{j=1}^{m-1}\mathbf{H^j}\right)$$

$$= \det\left(\begin{bmatrix} {\mu^0}_{\mathbf{M}} + d_1{\mu_1}^0 & {\mathbf{v^0}}_M + d_2{\mathbf{v^1}}_1 \\ {\mu^1}_{\mathbf{M}} + d_1{\mu_1}^1 & {\mathbf{v^1}}_M + d_2{\mathbf{v^1}}_1 \end{bmatrix} + \epsilon\begin{bmatrix} c_1 & c_2 \\ c_1 & c_2 \end{bmatrix}\right)$$

where

$$\epsilon = -\frac{2\Delta x}{\lambda}\sinh(\theta) = -\frac{\Delta x}{\lambda}(e^{\theta} - e^{-\theta})$$

$$d_1 = e^{-m\lambda}, \quad d_2 = e^{m\lambda}, \quad d_1 d_2 = 1$$

$$c_1 = \sum_{l=1}^{m-1} e^{-l\lambda}, \quad c_2 = \sum_{l=1}^{m-1} e^{l\lambda}$$

We can now further simplify the above determinant of (2X2) matrices

$$= 2\sqrt{\Delta t}(e^{\theta} + e^{-\theta} - 2) + 4\sqrt{\Delta t}\sinh(\theta)\sum_{l=1}^{m-1}\sinh(l\lambda)$$

$$+ 4\sqrt{\Delta t}\sinh(\theta)\sum_{l=1}^{m-1}\sinh((m)\lambda)$$

$$+ (1 - 2\sqrt{\Delta t}(e^{\theta} - 1))e^{-m\lambda} - (1 + 2\sqrt{\Delta t}(e^{-\theta} - 1))e^{m\lambda}$$

by letting $k = m - l$ in the second summation, the above relation is further simplified to

$$= 2\sqrt{\Delta t}(e^{\theta} + e^{-\theta} - 2) + 8\sqrt{\Delta t}\sinh(\theta)\sum_{l=1}^{m-1}\sinh(l\lambda)$$

$$+(1 - 2\sqrt{\Delta t}(e^{\theta} - 1))e^{-m\lambda} - (1 + 2\sqrt{\Delta t}(e^{\theta} - 1))e^{m\lambda}$$

converting the rest of the terms to hyperbolic functions leads to

$$= (4\sqrt{\Delta t}\cosh\left(\frac{\lambda}{2}\right) + 8\sqrt{\Delta t}\sinh\frac{\lambda}{2} * \sum_{l=1}^{m-1}\sinh(l\lambda)$$

$$+4\sqrt{\Delta t}\cosh(m\lambda)) - (4\sqrt{\Delta t} + 2\sinh(m\lambda) + 4\sqrt{\Delta t}\cosh\left(\left(m+\frac{1}{2}\right)\lambda\right))$$

In order to study the stability of the method, we can rewrite the equation $A\mathbf{v}^{n+1} = \mathbf{g}^{n+1}$ in terms of the coefficients at the previous time. The temperature fields u_k^n can be written in terms of coefficients as $u_k^n = c_k^n + d_k^n + \xi_k^n$. Therefore, we can write $\mathbf{g}n+1$ in terms of the coefficients as $g_k = {\xi_{k+1}}^{n+1} - \xi_k^{n+1} = u_{k+1}^n - u_k^n + \Delta t(q_{k+1}^{n+1} - q_{k+1}^n)$. Using this, we can rewrite the equation $A\mathbf{v}^{n+1} = \mathbf{g}^{n+1}$ as $A\mathbf{v}^{n+1} = B\mathbf{v}^n + \mathbf{h}$, where A is as given before and B is given by $\det(\Gamma_{22} - \Gamma_{21}\Gamma_{11}^{-1}\Gamma_{12})$. It is important to note that for the case where the kernels in the boundary conditions are not constant, the matrix Γ_{11} remains the same (i.e., $\det(\Gamma_{11}) \neq 0$) and we just need to check for $\det(\Gamma_{22} - \Gamma_{21}\Gamma_{11}^{-1}\Gamma_{12})$. Furthermore, for a constant Δt, $\lambda = \frac{\Delta x}{\sqrt{\Delta t}}$, and the above determinant is equal to a difference of two positive monotonic functions of Δx with different slopes. Therefore, there are infinite values of Δx for which the determinant is not equal to zero.

$$Av^{n+1}= \begin{bmatrix} -1 & -1 & 1 & 1 & \cdots & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\ \cdots 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\ \cdots \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & \cdots & 0 & -1 & -1 & 0 & 0 & \cdots 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 00 & \cdots 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & \cdots \vdots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 & 0 & -1 & -1\ 1 & 1 \\ 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & \cdots \vdots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} c_1 \\ d_1 \\ c_2 \\ d_2 \\ c_3 \\ d_3 \\ \vdots \\ c_m \\ d_m \\ c_M \\ d_M \end{bmatrix} + \begin{bmatrix} g_1 \\ 0 \\ g_2 \\ 0 \\ g_3 \\ 0 \\ \vdots \\ g_m \\ 0 \\ \sigma \\ \delta \end{bmatrix}$$

2.3.2 Spectral radius

For stability the spectral radius of the matrix $A^{-1}B$ be inside a unit circle. Which is the magnitude of the largest eigenvalue for the functions of the order of the approximation m. As the mesh is refined and xß0, the magnitude of the largest eigenvalue increases. However, it should be under unit circle, and the scheme is stable. The graph of the spectral radius

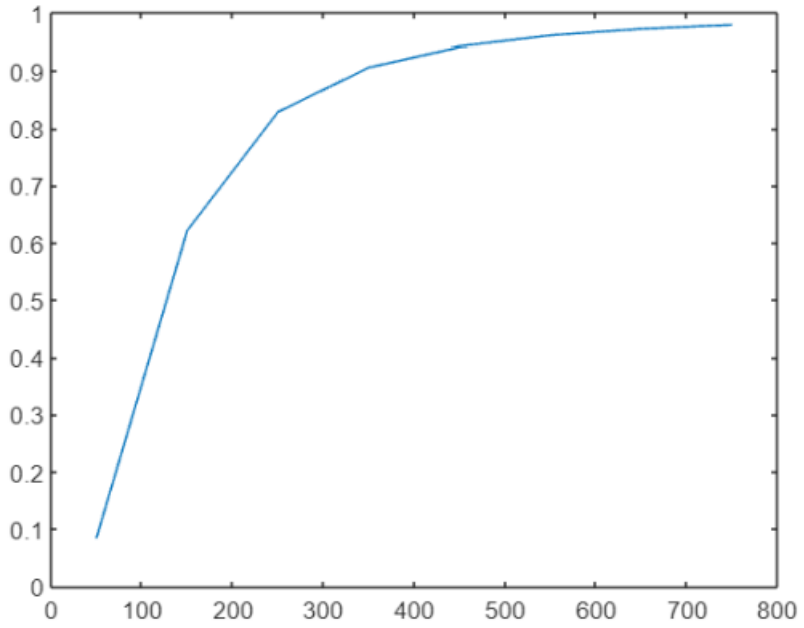

FIGURE 2.2: Norm of the largest eigenvalue of the matrix $A^{-1}B$

2.4 Numerical experiments

2.4.1 Experiment 1

Consider the heat conduction given by

$$u_t = u_{xx} + q(t,x), \quad t \geq 0, x \in [01], \quad u(0,x) = f(x), \tag{2.17}$$

$$u(t,0) = \int_0^1 \phi(x)u(t,x)dx, \quad u(t,1) = \int_0^1 \psi(x)u(t,x)dx, \tag{2.18}$$

where

$$q(t,x) = -e^{-t}\left(x(x-1) + \frac{\delta}{6(1+\delta)} + 2\right), \quad \phi(x) = -\delta, \quad \psi(x) = -\delta. \tag{2.19}$$

The exact solution is given by

$$u(t,x) = e^{-t}\left(x(x-1) + \frac{\delta}{6(1+\delta)}\right) \tag{2.20}$$

for the initial condition given by

$$f(x) = x(x-1) + \frac{\delta}{6(1+\delta)}. \tag{2.21}$$

Using the values of $\delta = 0.0144$, $\Delta t = 0.0025$, and $\Delta x = 0.0025$ for the present method, computing values of the temperature at $x = 0.32$, $x = 0.5$, and $x = 0.64$ to the exact values,

we get.

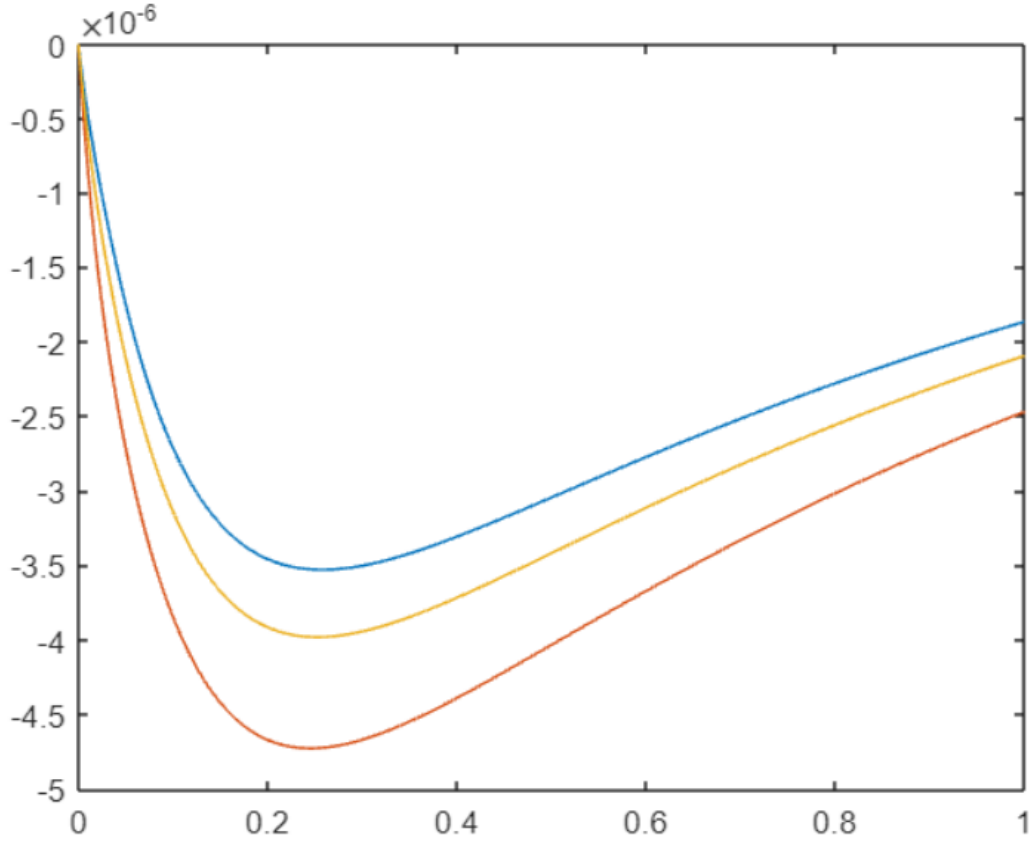

FIGURE 2.3: the error in the numerical solution at x=0.3, x=0.5 and x=0.7

We can clearly observe that the error is of 10^{-6} order Also if we try to impose the exact solution and the presented scheme on same graph

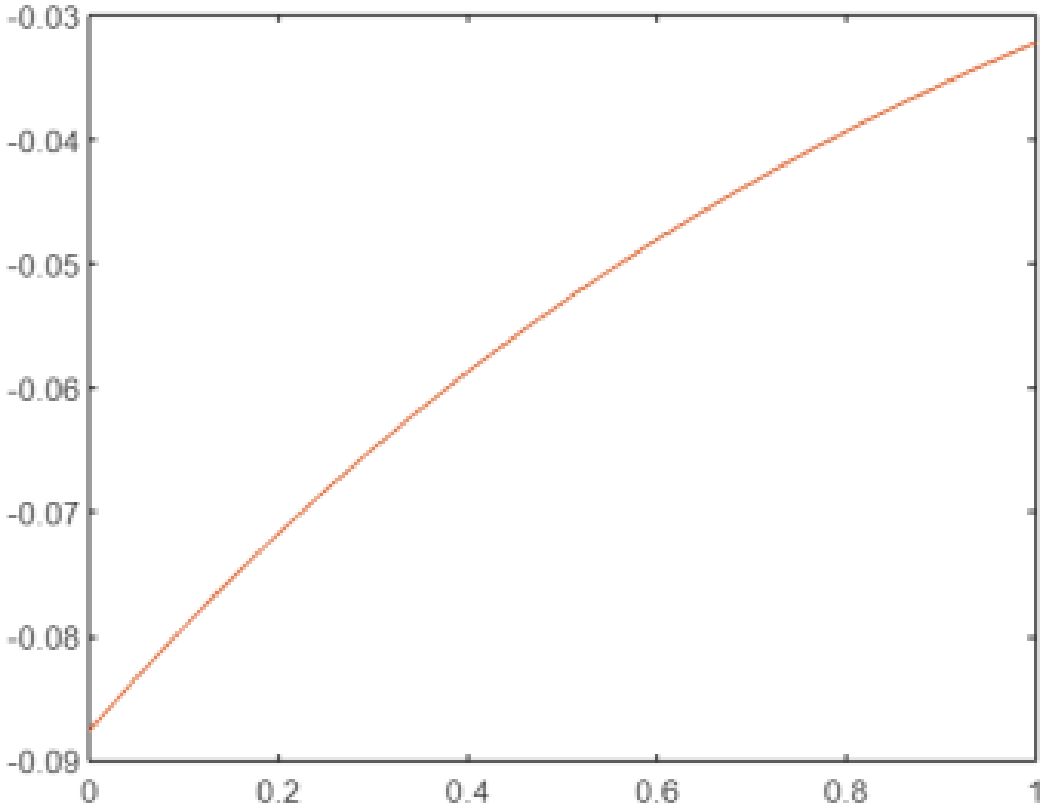

FIGURE 2.4: Imposed plot of exact solution and presented scheme solution

We can observe they both super impose which can prove the code correctness

2.4.2 Experiment 2

Consider the heat conduction given by

$$u_t = u_{xx} + q(t,x), \quad t \geq 0, x \in [0,1], \quad (0,x) = f(x), \tag{2.22}$$

$$u(t,0) = \int_0^1 \phi(x)u(t,x)dx, \quad u,1) = \int_0^1 \psi(x)u(t,x)dx, \tag{2.23}$$

where

$$q(t,x) = 2t(1+2x),\ \phi(x) = \frac{1}{2}, \quad \psi(x) = \frac{3}{2}. \tag{2.24}$$

The exact solution is given by

$$u(t,x) = (1+2x)\left(1+\frac{t}{2}\right) \tag{2.25}$$

for the initial condition given by

$$f(x) = 1 + 2x. \tag{2.26}$$

Using the values of $\delta = 0.0144$, $\Delta t = 0.0025$, and $\Delta x = 0.0025$ for the present method, computing values of the temperature at $x = 0.32$, $x = 0.5$, and $x = 0.64$ to the exact values, we get.

We can clearly observe that the error is of 10^3form

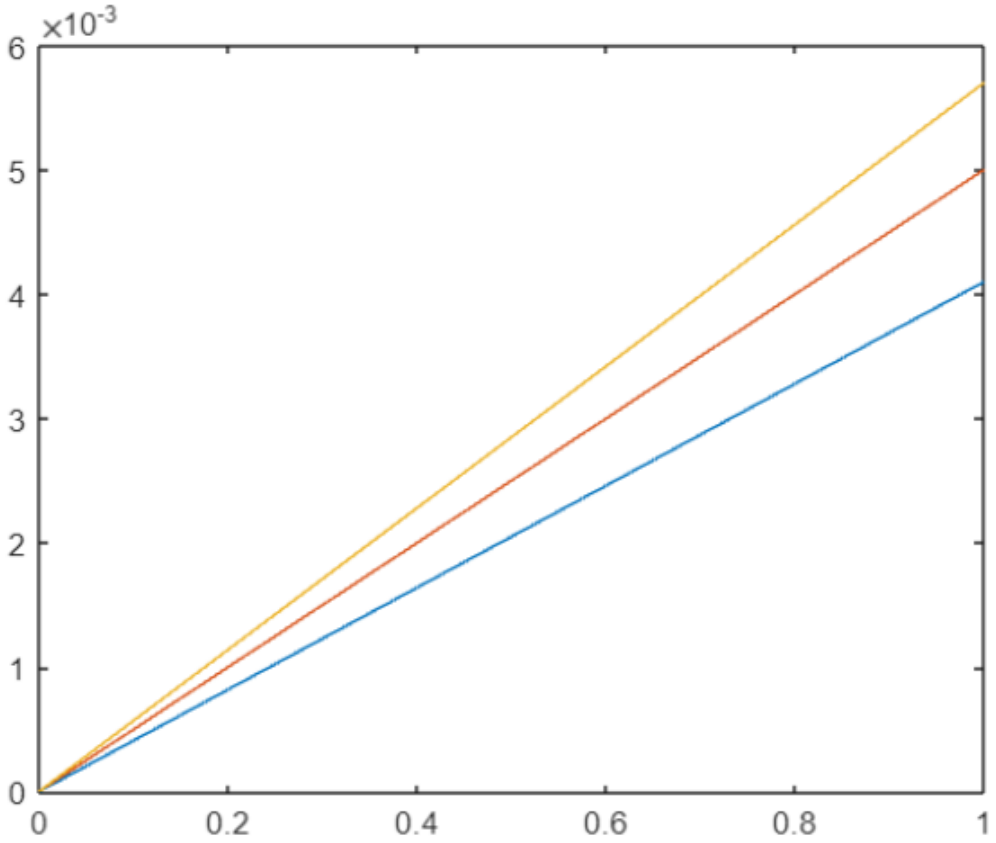

FIGURE 2.5: the error in the numerical solution at x=0.32,x=0.5 and x = 0.7

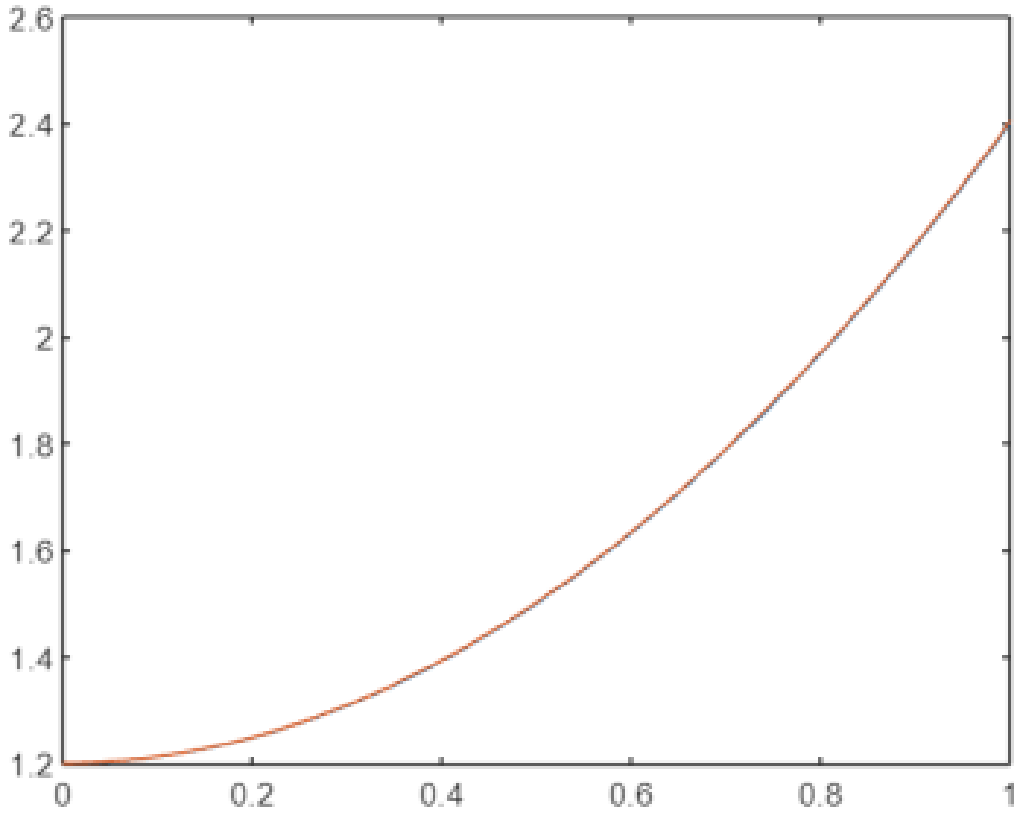

FIGURE 2.6: Imposed plot of exact solution and presented scheme solution

Chapter 3

Stability of Numerical Scheme for Parabolic Problem With Non-Local Boundary Conditions

3.1 Finite difference scheme in one-dimensional case

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + f(x,t), \quad 0 < x < 1, t > 0,$$

subject to the boundary conditions:

$$u(0,t) = \int_0^1 \alpha(x)u(x,t)dx + \mu_0(t), \quad u(1,t) = \int_0^1 \beta(x)u(x,t)dx + \mu_1(t),$$

and the initial condition:

$$u(x,0) = \phi(x).$$

In the domain $0 \leq x \leq 1$, $0 \leq t \leq T$, where $T > 0$, we introduce uniform grids Ω_h and ω_τ with the grid steps h and τ:

$$\Omega_h = x_i : x_i = ih, i = 0, 1, 2, \ldots, N, \quad h = \frac{1}{N},$$

$$\omega_\tau = t_k : t_k = k\tau, k = 0, 1, 2, \ldots, M, \quad \tau = \frac{T}{M},$$

where N and M define the grid dimensions. For the function $u = u(x, t)$, $x \in \Omega_h$, $t \in \omega_\tau$, we denote

$$u_i^k = u(x_i, t_k), \quad i = 0, 1, 2, \ldots, N, \quad k = 0, 1, 2, \ldots, M.$$

We also denote $\Lambda u_i = \frac{u_{i-1} - 2u_i + u_{i+1}}{h^2}$.

Assuming that N is an even number and $0 \leq \sigma \leq 1$ is the given weight parameter, by the 1D problem with the following weighted finite difference scheme:

$$\frac{u_i^{k+1} - u_i^k}{\tau} = \sigma \Lambda u_i^{k+1} + (1 - \sigma)\Lambda u_i^k + f_i^k, \quad (18)$$

$$u_0^{k+1} = \gamma_1 u_{N/2}^{k+1} + \mu_0(t_{k+1}), \quad u_N^{k+1} = \gamma_2 u_{N2,k+1} + \mu_1(t_{k+1}), \quad (19)$$

$$i = 1, 2, 3, \ldots, N - 1, \quad k = 0, 1, 2, \ldots, M - 1,$$

$$u_i^0 = \phi(x_i), \quad i = 0, 1, , N, \quad (20)$$

where $f_i^k = \sigma f(x_i, t_{k+1}) + (1-\sigma) f(x_i, t_k)$. For $\sigma = 0$, we have an explicit scheme (the forward Euler method). For $\sigma = 1$, we have an implicit scheme (the backward Euler method). For $\sigma = \frac{1}{2}$, one deals with a symmetric (actually, symmetry can be lost due to non-locality in boundary conditions) scheme (the Crank-Nicolson method).

3.2 Step-wise stability

We say that the numerical scheme is stepwise stable , if for fixed τ and h there exists a constant $C = C(\tau, h)$ such that

$$|u_i^k| \leq C, \quad i = 0, 1, 2, \ldots, N, \quad k = 0, 1, \ldots, (21)$$

Stepwise is certainly weaker than the uniform stability or the Lax-Richmeyer stability. We rewrite the finite difference problem in the matrix form:

$$u^{k+1} = Su^k + \tilde{f}^k, \quad (22)$$

where

$$S = (E + \tau\sigma A)^{-1}(E - \tau(1-\sigma)A), \quad (23)$$

$$\tilde{f}^k = (E + \tau\sigma A)^{-1}\bar{f}^k,$$

with $u^k = (u_1^k, u_2^k, \ldots, u_{N-1}^k)$, $\bar{f}^k = (\bar{f}_1^k, \bar{f}2^k, \ldots, \bar{f}N-1^k)$, and the matrix A of order $N-1$.

$$A = \frac{1}{h^2}\begin{pmatrix} 2 & -1 & 0 & \cdots & -\gamma_1 & \vdots & 0 & 0 \\ -1 & 2 & -1 & \vdots & \ddots & \vdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & \vdots & -1 & 2 & -1 & 0 \\ 0 & 0 & \vdots & \vdots & 0 & -1 & 2 & -1 \\ 0 & \cdots & 0 & 0 & -\gamma_2 & \vdots & -1 & 2 \end{pmatrix}$$

γ_1 and γ_2 are placed in the $\frac{N}{2}^{\text{th}}$ column of matrix A. We are using the following proposition about the stability: difference scheme (22) is stepwise stable, if absolute

values of all the eigenvalues of matrix C are less than one. Matrix A is asymmetric due to non-local conditions. Provided the matrix A has a simple structure, i.e. presents a set of $N-1$ linear independent eigenvectors, one can interpret the above definition of stability as a stability in the following norm:

$$||u||_* = ||P^{-1}u||_\infty = \max_{1\le i\le N-1} |(P^{-1}u)_i|$$

The matrix P contains linear independent eigenvectors of the matrix A. The matrix norm, compatible norm $||u||_*$, is defined as

$$||u||_* = ||P^{-1}SP||_\infty = \max_{1\le i\le N-1} \sum_{j=1}^{N-1} |\tilde{s}_{ij}| = \rho(S),$$

where $\tilde{s}_{ij}$ are the elements of the matrix $P^{-1}SP$, and $\rho(S)$ denotes the spectral radius of the matrix S.

3.2.1 Investigation of the matrix A spectrum

The spectrum of matrix A, defined by (24), the dependence of the eigenvalues of A on the value $\gamma = (\gamma_1 + \gamma_2)/2$ will be established.

$$\frac{u_{i-1} - 2u_i + u_{i+1}}{h^2} + \lambda u_i = 0, \quad i = 1, 2, \ldots, N-1, \quad (25)$$

$$u_0 = \gamma_1 u_{N/2}, \quad u_N = \gamma_2 u_{N/2}, \quad (26)$$

Few proofs of propositions on the spectrum of problem (25) and (26).

Lemma 3.1 - For all $h > 0$, the matrix A has the eigenvalue $\lambda = 0$ if and only if

$$\frac{\gamma_1 + \gamma_2}{2} = 1, \quad (27)$$

Proof. As $\lambda = 0$, the general solution of (25) is

$$u_i = c_1 + c_2 ih, \quad (28)$$

where c_1 and c_2 are arbitrary constants. By substituting expression (28) into (26) and on condition that $u_i \neq 0$, we get (27), i.e., there exists a nontrivial solution (28) if and only if equality (27) holds.

Lemma 3.2. For all $h > 0$, the condition $\gamma_1 + \gamma_2 \neq 0$ is necessary for the matrix A to have a simple spectrum. Proof. Suppose that $\gamma_1 + \gamma_2 = 0$. Then, we have

$$\begin{pmatrix} 1 & -1 & 0 & \cdots & 0 & -1 & 2 & -1 & \cdots & 0 & 0 & -1 \\ 2 & \cdots & 0 & \vdots & \vdots & \vdots & \ddots & \vdots & 0 & \cdots & -1 & 2 \\ -1 & \vdots & \vdots & \vdots & -1 & 1 & 0 & \cdots & 0 & -1 & 1 & \end{pmatrix} \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$\frac{\gamma_1 + \gamma_2}{2} > 1 \quad (3.1)$$

is necessary and sufficient for A to have only one negative eigenvalue. To prove this lemma, we start by writing equation (25) as

$$u_{i-1} - 2\left(1 - \frac{\lambda h^2}{2}\right) u_i + u_{i+1} = 0. \quad (3.2)$$

If $\lambda < 0$, then

$$cosh(\beta h) = 1 - \frac{\lambda h^2}{2} \quad (3.3)$$

we can use equation (3.3) to write the general solution of (3.2) as

$$u_i = c_1 \cosh(i\beta h) + c_2 \sinh(i\beta h). \quad (3.4)$$

Substituting expression (3.4) into (26) and demanding $u_i \neq 0$, we get equation

$$\sinh(\beta) - (\gamma_1 + \gamma_2)\sinh(\frac{\beta}{2}) = 0. \tag{3.5}$$

Solving equation (3.5) for β, we find that the condition in equation (3.1) is necessary and sufficient for A to have only one negative eigenvalue. Therefore, Lemma 3.2 is proven.

Or,

$$\sinh(\frac{\beta}{2})(2\cosh(\frac{\beta}{2}) - (\gamma_1 + \gamma_2)) = 0. \tag{3.6}$$

Since $\lambda < 0$ due to equation (3.3), we have $\sinh(\frac{\beta}{2}) \neq 0$ and hence

$$\cosh(\frac{\beta}{2}) = \frac{\gamma_1 + \gamma_2}{2}. \tag{3.7}$$

This means there exists a unique solution $\bar{\beta}$ of equation (3.7), such that $\beta > 0$, if and only if condition (3.1) holds.

Remark 3.1: Equation (3.7) has one negative root $\tilde{\beta} = -\bar{\beta}$. We can obtain the same value of λ with both $\tilde{\beta}$ and $-\bar{\beta}$, due to equation (3.4):

$$\lambda = \frac{2}{h^2}(1 - \cosh(\beta h)) = -\frac{4}{h^2}\sinh^2(\frac{\beta h}{2}). \tag{3.8}$$

Now, to deal with positive eigenvalues of the matrix A, we start with $\lambda > 0$. From $\lambda > 0$, we get $1 - \frac{\lambda h^2}{2} < 1$. Also, $|1 - \frac{\lambda h^2}{2}| \leq 1$ if $\lambda \leq \frac{4}{h^2}$. For this reason, we first find the eigenvalues of A such that $0 < \lambda \leq \frac{4}{h^2}$, under the assumption

$$\cos(\alpha h) = 1 - \frac{\lambda h^2}{2}. \tag{3.9}$$

By expression (3.2),

$$u_{i-1} - 2\cos(\alpha h)u_i + u_{i+1} = 0. \tag{3.10}$$

Its general solution is

$$u_i = c_1 \cos(i\alpha h) + c_2 \sin(i\alpha h). \tag{3.11}$$

Substituting this expression into (26) and demanding $u_i \neq 0$, we obtain the equation similar to (3.5):

$$\sin(\alpha) - (\gamma_1 + \gamma_2) \sin(\frac{\alpha}{2}) = 0. \tag{3.12}$$

Which is equivalent to two separate equations:

$$\sin(\frac{\alpha}{2}) = 0, \tag{3.13}$$

$$\cos(\frac{\alpha}{2}) = \frac{\gamma_1 + \gamma_2}{2}. \tag{3.14}$$

By putting the roots of these equations into (3.9), we obtain the expression of the eigenvalues λ. **Lemma 3.3** For all $h > 0$ independently of the value of $\frac{\gamma_1+\gamma_2}{2}$, the matrix A has eigenvalues

$$\lambda_k = \frac{4}{h^2} \sin^2(\pi k h), \quad k = 1, , \ldots, \frac{N}{2} - 1. \tag{3.15}$$

Lemma 3.4 If $|\frac{\gamma_1+\gamma_2}{2}| < 1$, then for all $h > 0$, all the eigenvalues of matrix A are real positive and different:

1. $\lambda_k = \frac{4}{h^2} \sin^2(\frac{\pi k h}{2}), \quad k = 1, 3, \ldots, \frac{N-1}{2} - 1.$
2. $\lambda_k = \frac{4}{h^2} \sin^2(\frac{\alpha_k h}{2}), \quad k = 2, 4, \ldots, \frac{N}{2}.$

where α_k are the roots of the equation $\cos(\frac{\alpha}{2}) = \frac{\gamma_1+\gamma_2}{2}$.

Lemma 3.5 If $\frac{\gamma_1+\gamma_2}{2} = 1$, then for all $h > 0$, all the eigenvalues of the matrix A are real. One eigenvalue is $\lambda_0 = 0$, and all others are positive and given by the formula

$$\lambda_k = \frac{4}{h^2}\sin^2(k\pi h), \tag{3.16}$$

where

1. With $k = 1, 3, \ldots, \frac{N}{2} - 1$ or $\frac{N}{2}$ depending on whether $N/2$ is odd or even respectively, the eigenvalues are simple.
2. With $k = 2, 4, \ldots, \frac{N}{2} - 2$ or $\frac{N}{2} - 1$ depending on whether $N/2$ is odd or even respectively, the eigenvalues are of multiplicity 3.
3. $\lambda = \frac{4}{h^2}$ if $N/2$ is even, is a simple eigenvalue.

Lemma 3.6 If $\frac{\gamma_1+\gamma_2}{2} = -1$, then for all $h > 0$, all the eigenvalues of the matrix A are real and positive, and given by the

$$\lambda_k = \frac{4}{h^2}\sin^2(k\pi h), \quad k = 1, 2, \ldots, \frac{N^2-1}{2}. \tag{3.17}$$

here

1. with $k = 1, 3, \frac{N}{2} - 1$ if $\frac{N}{2}$ is even, with $k = 1, 3, ..., \frac{N}{2} - 2$ of $\frac{N}{2}$ is odd, the eigen values λ_k are of multiplicity 3.
2. with $k = 2, 4, ...\frac{N}{2}$ if $\frac{N}{2}$ is even, with $k = 2, 4, ..., \frac{N}{2} - 1$ of $\frac{N}{2}$ is odd, the eigen values λ_k are simple.
3. $\lambda = \frac{4}{h^2}$ if $N/2$ is odd, is a simple eigenvalue.

Remark 3.2 The eigenvalues of multiplicity 3 in Lemmas 3.5 and 3.6 are multiple in the algebraic (not geometric) sense, i.e., each eigenvalue of multiplicity 2 has

the only corresponding linear eigenvector (there exist 2 adjoint linear independent vectors).

Lemma 3.7 If $\frac{\gamma_1+\gamma_2}{2} > 1$ and $N/2$ is even, then for all $h > 0$, there exists one positive eigenvalue of matrix A which is greater than $\frac{4}{h^2}$:

$$\lambda = \frac{4}{h^2}\cosh^2(\frac{\alpha h}{2}), \tag{3.18}$$

where α is the unique positive root of the equation $\cosh(\frac{\alpha}{2}) = \frac{\gamma_1+\gamma_2}{2}$.

Lemma 3.8 Suppose $\frac{\gamma_1+\gamma_2}{2} < -1$ and $N/2$ is odd, then for all $h > 0$, there exists one positive eigenvalue of matrix A which is greater than $\frac{4}{h^2}$:

$$\lambda = \frac{4}{h^2}\cosh^2(\frac{\alpha h}{2}), \tag{3.19}$$

where α is the unique positive root of the equation $\cosh(\frac{\alpha}{2}) = \frac{\gamma_1+\gamma_2}{2}$

Lemma 3.9 states that if $\frac{\gamma_1+\gamma_2}{2} < -1$, then for all $h > 0$, there exist complex eigenvalues of matrix A. number of complex eigenvalues is equal to $N/2$ or $\frac{N}{2} - 1$ depending on whether $N/2$ is even or odd respectively. These eigenvalues are given by the formula

$\lambda_k = \frac{4}{h^2}\sin^2(\frac{q_k h}{2}), (3.20)$ where $q_k = \alpha_k \pm i\beta_k$, $\alpha_k = 2(2k-1)\pi$ with $k = 1, 2, \ldots, \left[\frac{N}{4}\right]$, and β is the root of the equation $\cosh(\beta/2) = -\frac{(\gamma_1+\gamma_2)}{2}$. Similarly,

Lemma 3.10 states that if $\frac{\gamma_1+\gamma_2}{2} > 1$, then for all $h > 0$, there exist complex eigenvalues of matrix A given by the formula

$\lambda_k = \frac{4}{h^2}\sin^2(\frac{q_k h}{2})$(3.21), where $q_k = \alpha_k \pm i\beta$, $\alpha_k = 4k\pi$ with $k = 1, 2, \ldots, \left[\frac{N}{4}\right] - 1$, and β is the root of the equation $\cosh(\frac{\beta}{2}) = \frac{(\gamma_1+\gamma_2)}{2}$.

Corollary 3.1 provides a way to express the complex eigenvalues given in Lemmas 3.9 and 3.10 in a different form. Specifically, the complex eigenvalues can be expressed as

$\lambda_k = \mathrm{Re}(\lambda_k) \pm \mathrm{Im}(\lambda_k)$, (3.22) where $\mathrm{Re}(\lambda_k)$ and $\mathrm{Im}(\lambda_k)$ are given by the formulas

$$\mathrm{Re}(\lambda_k) = \frac{2}{h^2}(1 - \cos(\alpha_k h)\cosh(\beta_k h)), \tag{3.23}$$

$$\mathrm{Im}(\lambda_k) = \frac{2}{h^2}\sin(\alpha_k h)\sinh(\beta h). \tag{3.24}$$

For small enough h, one can approximately evaluate $\mathrm{Re}(\lambda_k) = \alpha_k^2 - \beta^2 + \mathcal{O}(h^2)$ and $\mathrm{Im}(\lambda_k) = 2\alpha_k\beta + \mathcal{O}(h^2)$.

3.3 Stability of difference schemes in the one-dimensional case

Now, we are in a position to state the main result about the stability of the above difference scheme (18)-(20). Note that this scheme, by the above formula (23),

defines the matrix S, the partial cases of which are

$$S = (E - \tau A), \qquad \text{if } \sigma = 0$$

$$S = (E + \tau A)^{-1}, \qquad \text{if } \sigma = 1$$

$$S = (E + \frac{\tau}{2}A)^{-1}(E - \frac{\tau}{2}A), \qquad \text{if } \sigma = \frac{1}{2}$$

Lemma 4.1 states that if $\mathrm{Re}(\lambda_k(A)) > 0$ for $k = 1, 2, \ldots, N-1$, and $\sigma = \frac{1}{2}$ or $\sigma = 1$, then the difference scheme (18)-(20) is stepwise stable for all $\sigma > 0$ and $h > 0$. **Lemma 4.2** states that the difference scheme (18)-(20) is conditionally stepwise stable if $\mathrm{Re}(\lambda_k(A)) > 0$ for $k = 1, 2, \ldots, N-1$ and $\sigma = 0$ under the following condition:

$$\tau < \max_{1 \le k \le N-1} \frac{2\mathrm{Re}(\lambda_k(A))}{(\mathrm{Re}(\lambda_k(A)))^2 + (\mathrm{Im}(\lambda_k(A)))^2} \tag{3.25}$$

Remark 4.2 notes that if all the eigenvalues $\mathrm{Re}(\lambda_k(A)) > 0$ are real positive numbers, then the inequality $\tau < \frac{h^2}{2}$ holds. However, if some of the eigenvalues of the matrix A are complex, then the inequality can be interpreted as an additional restriction on the grid step τ.

Theorem 4.1 states that there exists a number $\gamma^* > 1$ such that for all values $\frac{\gamma_1+\gamma_2}{2}$ satisfying the condition $-\gamma^* < \frac{\gamma_1+\gamma_2}{2} < 1$,
the difference scheme (18)-(20) with $\sigma = \frac{1}{2}$ or$sigma = 1$ is stepsize stable for all $\tau > 0$ and $h > 0$. The difference scheme is conditionally stepwise stable under the additional condition if $\sigma = 0$.

Remark 4.2 notes that if $h \ge \frac{1}{4}$, then $\mathrm{Re}(\lambda_k(A)) \ge 0$ and $\cos(2\pi h) \le 0$ for all values of β. As $h \to 0$, we have $\beta^* \to 2\pi$, i.e.
$\gamma^* \to \cos(\pi h) \approx -11.59195$. Similarly, one may demonstrate that if $\frac{\gamma_1+\gamma_2}{2}$ satisfying

the condition $-1 < \frac{\gamma_1+\gamma_2}{2} < \gamma^*$, then all truly complex (i.e., such that $\mathrm{Im}(\lambda_k(A)) \neq 0$) eigenvalues have the property $\mathrm{Re}(\lambda_k(A)) > 0$. Yet, Lemma 3.2 states that one real negative eigenvalue occurs when $\frac{\gamma_1+\gamma_2}{2} > 1$.

3.4 Numerical experiment

Test problems

analyzing the stability of difference schemes (18)-(20) when an exact solution of the differential problem is known. The maximum norm of the error for algorithm (18)-(20) is defined as

$$||\epsilon||_{C_h} = \max k = 01, 2, \ldots, M \max_{i=0,1,\ldots,N} |u_i^k - u(ih, k\tau)| \tag{3.26}$$

We have verified the computed error for a few different exact solutions. Namely, the functions f, μ_i, and φ have been chosen so that

$$u(x,t) = x^3 + t^3 \tag{3.27}$$

In Fig. 3.1, we have presented the impact of parameters γ_1 and γ_2 on the stability of difference schemes. The "checkerboard" plot of some matrix (discretized function in 2D) means that the values of the elements of this matrix specify the color in each pixel of the plot. We can conclude from Fig. 3.1 that stability depends only on $\bar{\gamma} = \frac{\gamma_1+\gamma_2}{2}$

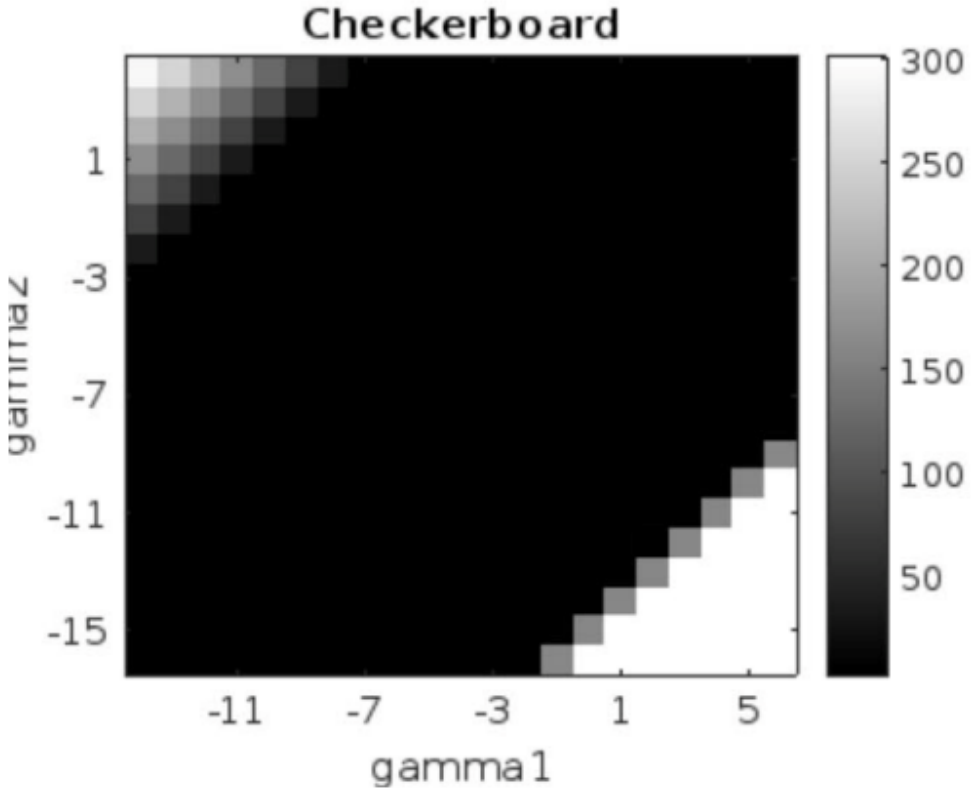

FIGURE 3.1: $\gamma_1 vs \gamma_2$

Chapter 4

Transformation of Financial Problem to Heat Problem

4.1 Black-Scholes Equation

The Black-Scholes equation models the behavior of the price of a financial derivative, such as a stock option, as a function of time and the underlying asset price. In its simplest form, the equation assumes a frictionless market with constant volatility, no dividends, and risk-free interest rates. The equation is given as:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0$$

where V is the option price as a function of the underlying asset price S and time t, σ is the volatility of the underlying asset, r is the risk-free interest rate, and the partial derivatives represent the rates of change.

This equation is a parabolic equation because it contains a second-order derivative with respect to the underlying asset price S. The term $\frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2}$ is responsible for the diffusion behavior, representing the random fluctuations in the underlying asset price.

Now, let's discuss the non-local boundary conditions that occur in the case of American options. Unlike European options, American options can be exercised at any time before expiration. This additional flexibility introduces non-local boundary conditions into the Black-Scholes equation.

In the case of American options, the holder has the right to exercise the option early if it is financially advantageous to do so. This means that the option value can depend not only on the current asset price but also on the optimal exercise strategy over a range of possible future asset prices.

To account for these non-local boundary conditions, the Black-Scholes equation needs to be solved subject to certain constraints. These constraints involve finding the optimal exercise boundary, which separates the region where it is optimal to exercise the option from the region where it is better to hold onto the option.

Determining the optimal exercise boundary is a challenging task, and analytical solutions are generally not available. Instead, numerical methods such as finite difference methods or Monte Carlo simulations are often used to solve the Black-Scholes equation with non-local boundary conditions for American options.

4.1.1 Black-Scholes Equation for American option

the analysis of an American put option being similar. The value of a call option, represented by V, relies on the current market price of the underlying asset, S, and

the remaining time t until the option expires: $V = V(S,t)$. The Black–Scholes equation is a backward-in-time parabolic equation and is defined on a time-dependent domain:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + (r - D_0) S \frac{\partial V}{\partial S} - rV = 0, \quad 0 < S < S_f(t), \quad 0 \le t < T, \quad (4.1)$$

where σ represents the annual volatility of the asset price, r is the risk-free interest rate, and T is the expiration date ($t = 0$ means "today"). We assume that dividends are paid with continuous yields of constant level $D_0 > 0$. It is essential to include dividend payments; otherwise, early exercise would not make sense for $D_0 = 0$, and the American call would be equivalent to the European options. In the equation above, $S_f(t)$ denotes the (a priori unknown) free boundary, also known as the "early exercise boundary" or "optimal exercise price." The American call option should be exercised if the asset value S is equal to or greater than $S_f(t)$ at time t; otherwise, the option should be held. The free boundary $S_f(t)$ separates the holding region ($S < S_f(t)$) from the exercise region ($S \ge S_f(t)$). The final condition ("payoff condition") at the expiry $t = T$ can be written as:

$$V(S,T) = (S - E)^+, \quad 0 \le S < S_f(T), \quad (4.2)$$

where $f^+ = \max(f, 0)$ and $E > 0$ represents the previously agreed exercise price or "strike" of the contract, and $S_f(T) = \max(E, rE/D_0)$. The "spatial" or asset-price boundary conditions at $S = 0$, and $S = S_f(t)$ are:

$$V(0,t) = 0, \quad 0 \le t \le T, \; V(S_f(t),t) \; = (S_f(t) - E)^+, \quad \frac{\partial V}{\partial S}(S_f(t),t) = 1, \quad 0 \le t \le T, \quad (4.3)$$

meaning that the option is worthless at $S = 0$. We require two conditions at the free boundary $S = S_f(t)$. One condition is necessary for solving the equation, and the other is needed for determining the position of the free boundary $S_f(t)$ itself. The first condition in the equation above ("value matching" condition) ensures the continuity of the mapping $S \to V(S,t)$, as $V(S,t) = (S-E)^+ = S - E$ in the exercise region $S \geq S_f(t)$. At $S = S_f(t)$, we also require that $V(S,t)$ touches the payoff function tangentially ("high contact condition"), i.e., the function $S \to \frac{\partial V(S,t)}{\partial S}$ should be continuous at $S = S_f(t)$. These conditions are collectively referred to as the "smooth-pasting conditions." The latter condition can be derived from an arbitrage argument . Since American options can be exercised at any time, we have the a priori bound:

$$V(S,t) \geq (S-E)^+, \quad S \geq 0, \quad 0 \leq t \leq T. \tag{4.4}$$

If $V(S,t) < (S-E)^+$ for one value $S > E$ and $t \leq T$, then purchasing a call for V and immediately exercising this option to buy the underlying asset for E (although its value is S) would result in an instantaneous risk-free profit of $S - V - E > 0$, violating the no-arbitrage principle. This reasoning, however, disregards transaction costs.

4.2 transformation of black-Scholes Equation

we demonstrate how to transform equation (4.1) into a pure diffusion equation. First, it is convenient to apply a time reversal and transform equation (4.1) into a forward-in-time equation by the change of variable $t = T - \frac{2\tau}{\sigma^2}$. The new time variable τ represents the remaining lifetime of the option (up to the scaling by $\frac{\sigma^2}{2}$).

We denote the new variables by:

$$\tilde{V}(S,\tau) = V(S,t) = V\left(S, T - \frac{2\tau}{\sigma^2}\right), \tilde{S}_f(\tau) = S_f\left(T - \frac{2\tau}{\sigma^2}\right), \tag{4.5}$$

$$\tilde{r} = \frac{2}{\sigma^2}r, \tilde{D}_0 = \frac{2}{\sigma^2}D_0, \tilde{T} = \frac{\sigma^2}{2}T. \tag{4.6}$$

The resulting forward-in-time equation reads:

$$\frac{\partial \tilde{V}}{\partial \tau} = S^2\frac{\partial^2 \tilde{V}}{\partial S^2} + (\tilde{r} - D_0')S\frac{\partial \tilde{V}}{\partial S} - \tilde{r}\tilde{V}, \quad 0 < S < \tilde{S}_f(\tau), \quad 0 \leq \tau < \tilde{T}, \tag{4.7}$$

with the initial condition:

$$\tilde{V}(S,0) = (S-E)^+, \quad 0 \leq S < \tilde{S}_f(0) = S_0. \tag{4.8}$$

And the boundary conditions are:

$$\lim_{S\to 0} \tilde{V}(S,\tau) = 0, \quad 0 \leq \tau \leq \tilde{T}, \tag{4.9}$$

$$\tilde{V}(\tilde{S}_f(\tau),\tau) = (\tilde{S}_f(\tau) - E)^+, \quad \frac{\partial \tilde{V}}{\partial S}(\tilde{S}_f(\tau),\tau) = 1, \quad 0 \leq \tau \leq \tilde{T}. \tag{4.10}$$

The right-hand side of the equation (4.7) is a well-known Euler's differential equation, and therefore it is standard practice to transform it into the heat equation. To do so, we let:

$$\alpha = -\frac{1}{2}(\tilde{r} - \tilde{D}_0 - 1), \ \beta = -\alpha^2 - \tilde{r}, \tag{4.11}$$

and use the change of variables:

$$S = Ee^x, \quad \tilde{V}(S,\tau) = Ee^{\alpha x+\beta\tau}v(x,\tau). \tag{4.12}$$

Then, problem (4.7) is equivalent to the free boundary problem for the heat equation:

$$\frac{\partial v}{\partial \tau} = \frac{\partial^2 v}{\partial x^2}, \quad -\infty < x < x_f(\tau), \quad 0 \le \tau < \tilde{T}, \tag{4.13}$$

where $x_f(\tau) = \ln(\tilde{S}_f(\tau)/E)$. The equation above is supplied with the initial condition:

$$v(x,0) = g(x,0) = \left(e^{\frac{1}{2}(\tilde{r}-\tilde{D}_0+1)x} - e^{\frac{1}{2}(\tilde{r}-\tilde{D}_0-1)x}\right)^+, \quad x < x_f(0), \tag{4.14}$$

with $x_f(0) = \ln(\max(1, r/D_0))$ and the boundary conditions:

$$\lim_{x\to-\infty} v(x,\tau) = 0, \quad 0 \le \tau \le \tilde{T}, \tag{4.15}$$

$$v(x_f(\tau),\tau) = g(x_f(\tau),\tau), \quad 0 \le \tau \le \tilde{T}, \tag{4.16}$$

$$e^{(\alpha-1)x+\beta\tau}(\alpha v(x_f(\tau),\tau) + \frac{\partial v}{\partial x}(x_f(\tau),\tau)) = 1, \quad 0 \le \tau \le \tilde{T}. \tag{4.17}$$

where:

$$g(x,\tau) = e^{-\alpha x-\beta\tau}(e^x - 1)^+. \tag{4.18}$$

It is well-known that the free boundary $S_f(t)$ is a non-decreasing function and:

$$S_f(T) \le S_f(t) \le S_f^*, \quad 0 \le t \le T, \tag{4.19}$$

with:

$$S_f^* = \frac{\sqrt{-\beta} + \alpha}{\sqrt{-\beta} + \alpha - 1} E. \tag{4.20}$$

Thus, if we set $x_f^* = \ln(S_f^*/E)$, then the free boundary $x_f(\tau)$ has the property:

$$0 \leq x_f(\tau) \leq x_f^*, \quad 0 \leq \tau \leq \tilde{T}. \tag{4.21}$$

4.3 Transformed parabolic initial boundary value problem

We note that the original Black–Scholes equation (4.7) is degenerate at $S = 0$. However, the change of variables in equation (4.12) transformed it into a uniformly parabolic initial boundary value problem (4.13).

4.3.1 Transparent Boundary Condition

The boundary problem (4.13) is posed on an unbounded and time-dependent domain $\Omega(\tau)$:

$$\Omega(\tau) = \{(x, \tau) \in \mathbb{R}^2 | x < x_f(\tau), 0 \leq \tau \leq \tilde{T}\}. \tag{4.22}$$

In the following, we briefly present the derivation of the (analytic) TBC at the artificial boundary $x = a$. For this purpose, we split the domain $\Omega(\tau)$ into the bounded time-dependent interior domain:

$$\Omega_{\text{int}}(\tau) = \{(x, \tau) \in \mathbb{R}^2 | a < x < x_f(\tau), 0 \leq \tau \leq \tilde{T}\}, \tag{4.23}$$

and the unbounded time-independent exterior domain:

$$\Omega_{\text{ext}} = \{(x, \tau) \in \mathbb{R}^2 | x < a, 0 \leq \tau \leq \tilde{T}\}. \tag{4.24}$$

4.3.2 Derivation of the Transparent Boundary Condition

Here, we determine the TBC at $x = a < 0$ such that the solution of the resulting initial boundary value problem coincides with the solution of problem (4.13) restricted to Ω_{int}. For simplicity, we assume that the initial data $v(x, 0)$ is compactly supported in the interior domain Ω_{int}, i.e., $g(x, 0) = 0$ for $x < a$. A strategy to overcome this restriction can be found in.

The analytic TBC for the heat equation has been derived by several authors. Historically, this TBC was first derived by Papadakis in the context of the Schrödinger equation. We remark that the derivation of the TBC for a parabolic convection-diffusion equation with a reaction term can be found.

For the derivation of the TBC at $x = a$, we consider the interior problem:

$$\frac{\partial v}{\partial \tau} = \frac{\partial^2 v}{\partial x^2}, \quad (x, \tau) \in \Omega_{\text{int}}(\tau), \tag{4.25}$$

$$v(x, 0) = g(x, 0), \quad a < x < x_f(0), \tag{4.26}$$

$$\frac{\partial v}{\partial x}(a, \tau) = (T_a v)(a, \tau), \quad 0 \leq \tau \leq T, \tag{4.27}$$

together with the boundary conditions (4.16), (4.17) at the free boundary $x = x_f(\tau)$. We obtain the Dirichlet-to-Neumann map T_a by solving the exterior problem:

$$\frac{\partial u}{\partial \tau} = \frac{\partial^2 u}{\partial x^2}, \quad (x,\tau) \in \Omega_{\text{ext}}, \tag{4.28}$$

$$u(x,0) = 0, \quad x < a, \tag{4.29}$$

$$u(a,\tau) = \Phi(\tau), \quad 0 \le \tau \le \tilde{T}, \quad \Phi(0) = 0, \tag{4.30}$$

$$u(-\infty,\tau) = 0, \quad 0 \le \tau \le \tilde{T}, \tag{4.31}$$

$$(T_a\Phi)(\tau) = \frac{\partial u}{\partial x}(a,\tau), \quad 0 \le \tau \le \tilde{T}. \tag{4.32}$$

The problem on the exterior domain Ω_{ext} is coupled to the problem on the interior domain Ω_{int} by the assumption that v, $\frac{\partial v}{\partial x}$ are continuous across the artificial boundary at $x = a$. One can solve (4.29) explicitly by the Laplace-method, i.e., we use the Laplace transformation of u:

$$\hat{u}(x,s) = \int_0^\infty u(x,\tau)e^{-s\tau}d\tau. \tag{4.33}$$

We set $s = \zeta + i\xi$, $\xi \in \mathbb{R}$, and $\zeta > 0$ is fixed, with the idea to later perform the limit 0. Now, the exterior problem (4.29) is transformed to:

$$\hat{u}_{xx} - s\hat{u} = 0, \quad x < a, \tag{4.34}$$

$$\hat{u}(a,s) = \hat{\Phi}(s). \tag{4.35}$$

The solution to (4.34) that decays as $x \to -\infty$ is simply $\hat{u}(x,s) = \hat{\Phi}(s)e^{\sqrt[+]{s}(x-a)}, x < a$, where $\sqrt[+]{\cdot}$ denotes branch of the square root with nonnegative real part. Consequently, the transformed TBC is:

$$\hat{u}_x(a,s) = \sqrt[+]{s}\hat{u}(a,s), \tag{4.36}$$

and after an inverse Laplace transformation, the TBC at $x = a$ reads:

$$v_x(a,\tau) = \frac{1}{\sqrt{\pi}} \int_0^\tau \frac{v_\tau(a,\xi)}{\sqrt{\tau-\xi}} d\xi. \tag{4.37}$$

We observe that (4.37) has a weakly singular kernel and is a memory-type non-local function τ, i.e., the computation of the solution at some time uses the solution at all previous times.

Remark 3.1.the solution in Ω_{ext} can also be computed with:

$$v(x,\tau) = -\frac{x-a}{2\sqrt{\pi}} \int_0^\tau \frac{e^{-\frac{(x-a)^2}{4(\tau-\xi)}} v(a,\xi)}{(\tau-\xi)^{3/2}} d\xi, \quad x < a. \tag{4.38}$$

Remark 3.2. The treatment of an American put option is completely analogous. Now, one has to consider the Black–Scholes equation (4.12) on the domain $S > S_f(t)$. The terminal condition at the expiry date $t = T$ then reads:

$$V(S,T) = (E-S)^+, \quad S > S_f(T). \tag{4.39}$$

The "spatial" boundary conditions at $S = S_f(t)$, $S \to \infty$ are given by:

$$V(S_f(t),t) = (E - S_f(t))^+, \quad \frac{\partial V}{\partial S}(S_f(t),t) = -1, \quad 0 \le t \le T, \tag{4.40}$$

$$\lim_{S\to\infty} V(S,t) = 0, \quad 0 \le t \le T. \tag{4.41}$$

Thus, the TBC has to be constructed at $x = b$ with $b > S_f(t)$, for all $0 \le t \le T$.

4.3.3 Parameters depending on Time

It is possible to derive a TBC for American call options with time-varying interest rate $r = r(t)$, dividend yield $D = D(t)$, and volatility $\sigma = \sigma(t)$. This situation is

more realistic, but the time-dependence of the parameters $r = r(t)$ and $\sigma = \sigma(t)$ is unknown and must be modeled stochastically:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2(t)S^2\frac{\partial^2 V}{\partial S^2} + (r(t) - D(t))S\frac{\partial V}{\partial S} - r(t)V = 0, \tag{4.42}$$

for $0 < S < S_f(t)$, $0 \leq t < T$. Making the substitutions:

$$\bar{S} = Se^{\alpha(t)}, \tag{4.43}$$

$$\bar{V} = Ve^{\beta(t)}, \tag{4.44}$$

$$\bar{t} = \gamma(t), \tag{4.45}$$

with

$$\alpha(t) = \int_t^T (r(\tau) - D(\tau))d\tau, \tag{4.46}$$

$$\beta(t) = \int_t^T r(\tau)d\tau, \tag{4.47}$$

$$\gamma(t) = \int_t^T \sigma^2(\tau)d\tau, \tag{4.48}$$

then (3.7) becomes

$$\frac{\partial \bar{V}}{\partial \bar{t}} = \frac{1}{2}\bar{S}^2\frac{\partial^2 \bar{V}}{\partial \bar{S}^2}, \quad 0 < \bar{S} < \bar{S}_f(\bar{t}), \quad 0 \leq \bar{t} \leq \bar{T} = \gamma(0). \tag{4.49}$$

Supplied with the initial condition $\bar{V}(\bar{S}, 0) = V(S, T)$ because $\gamma(T) = 0$. Since the right-hand side of (4.49) is again of Euler-type, one can proceed analogously to The Laplace-transformed exterior problem reads:

$$\frac{x^2}{2}\hat{u}_{xx} - s\hat{u} = 0, \quad x < a, \tag{4.50}$$

$$\hat{u}(a,s) = \hat{\Phi}(s). \tag{4.51}$$

The solution to (4.51) which decays as $x \to -\infty$ is simply:

$$\hat{u}(x,s) = \hat{\Phi}(s)\left(\frac{x}{a}\right)^{\frac{1}{2}-\frac{1}{2}\sqrt[+]{1+8s}}, \quad x < a, \tag{4.52}$$

and therefore the transformed TBC is:

$$\hat{u}_x(a,s) = a^{-1}\left(\frac{1}{2} - \sqrt{2}\sqrt[+]{s+\frac{1}{8}}\right)\hat{u}(a,s). \tag{4.53}$$

Finally, an inverse Lap transformation yields the desired TBC at $x = a$:

$$\bar{V}_x(a,\bar{t}) = \frac{\bar{V}(a,\bar{t})}{2a} - \frac{\sqrt{2}}{a\sqrt{\pi}}\int_0^{\bar{t}}\left(\frac{\partial\bar{V}}{\partial\bar{t}}(a,\xi) + \frac{\bar{V}(a,\xi)}{8}\right)e^{-(\bar{t}-\xi)/8}\frac{1}{\sqrt{\bar{t}-\xi}}d\xi. \tag{4.54}$$

Most dividend payments on an index (e.g., the Dow Jones Industrial Average (DJIA) or the Standard and Poor's 500 (SandP500)) are so frequent that they can be modeled as a continuous payment. However, companies make two or four payments per year, then one has to treat the dividend payments discretely, and the question is how to incorporate discrete dividend payments into the Black–Scholes equation. In the sequel, we briefly review the results from We assume that there is only one dividend payment during the lifetime of the option at the dividend date t_d. Neglecting other factors like taxes, the asset price S must decrease exactly by the amount of the dividend payment d_0. Thus, we have the jump condition:

$$S(t_d^+) = (1 - d_0)S(t_d^-), \tag{4.55}$$

t_d^-, t_d^+ denotes the moments just before and after t_d. This leads to the following effect on the option price:

$$V(S, t_d^-) = V((1 - d_0)S, t_d^+). \tag{4.56}$$

That is, the value of the option at S and time t_d is the same as the value immediately after the dividend date t_d but at the asset value $(1 - d_0)S$. To value a call option with one dividend payment, we solve the Black–Scholes equation from expiry $t = T$ until $t = t_d^+$ and use the relation (4.56) to compute the values at $t = t_d^-$. Finally, we continue to solve the Black–Scholes equation backward starting at $t = t_d^-$ using these values as initial data. The transparent boundary conditions need not be modified for this case.

Chapter 5

Numerical Scheme for the Transformed Problem

5.1 DTBC

Now address the question of how to discretize the analytic TBC (4.37) adequately for a chosen full discretization of (4.13), which in this example will be the Crank–Nicolson scheme. This scheme has been extremely popular for numerical solutions in finance since it is unconditionally stable and has second-order accuracy in time and space. Furthermore, it obeys a discrete maximum principle. Instead of discretizing the analytic TBC (4.37) with its singularity, our strategy is to derive the discrete TBC of the fully discretized problem. With the uniform grid points $x_j = a + j\Delta x$, $j = 0, 1, \ldots$, $\tau_n = \Delta\tau$, $n = 0, 1, \ldots$ and the approximation $v_j^{(n)} \approx v(x_j, \tau_n)$, the Crank–Nicolson scheme for solving the heat equation (4.13) is:

$$v_j^{(n+1)} - v_j^{(n)} = \rho \left(v_{j+1}^{(n+1/2)} - 2v_j^{(n+1/2)} + v_{j-1}^{(n+1/2)} \right), \tag{5.1}$$

with the abbreviation $v_j^{(n+1/2)} = (v_j^{(n+1)} + v_j^{(n)})/2$ and the parabolic mesh ratio $\rho = \Delta\tau/(\Delta x 2)$. While a uniform grid in x is necessary in the exterior domain, the interior grid may be nonuniform (e.g., logarithmic) in x. In the sequel, we present different strategies to incorporate the analytic TBC (4.37) into the finite difference scheme (5.1). now we aim to evaluate three different approaches for discretizing the TBC (4.37), which is a complex issue due to its slightly singular convolution kernel. Initially, we will examine two established discretization methods from Mayfield [] and Han and Wu [].

5.1.1 Mayfield

we can use in comparing our findings, we initially examine the ad-hoc discretization technique employed by Mayfield for the heat equation (4.13) . Based on Mayfield's approach for the Schrödinger equation, a possible method to discretize the analytic TBC (4.37) at $x = a$ in an equivalent form is given by:

$$v(a,\tau) = \frac{1}{\sqrt{\pi}} \int_0^\tau \frac{v_x(a,\xi)}{\sqrt{\tau - \xi}} d\xi \tag{5.2}$$

is

$$\int_0^{\tau_n} \frac{v_x(a,\tau_n - \xi)}{\sqrt{\xi}} d\xi \approx \frac{1}{\Delta x} \sum_{m=0}^{n-1} (v_1)^{n-m} - (v_0)^{n-m}) \int_{\tau_m}^{\tau_{m+1}} \frac{d\xi}{\xi} = \frac{2\sqrt{\Delta\tau}}{\sqrt{\Delta x}} \sum_{m=0}^{n-1} \frac{(v_1^{n-m} - v_0^{n-m})}{\sqrt{m+1} + \sqrt{m}}. \tag{5.3}$$

This results in the following discretized TBC for the heat equation:

$$v_1^n - v_0^n = \frac{\sqrt{\pi}\Delta x}{2\sqrt{\Delta\tau}} v_0^n - \sum_{m=1}^{n-1} \tilde{l}^m (v_1^{n-m} - v_0^{n-m}), \tag{5.4}$$

where the convolution coefficients are given by:

$$\tilde{l}^{(m)} = \frac{1}{\sqrt{m+1}+\sqrt{m}}. \tag{5.5}$$

5.1.2 Han and Wu

$$\int_0^{\tau_n} \frac{v_\tau(a,\xi)}{\sqrt{\tau_n-\xi}}d\xi \approx \sum_{m=0}^{n-1} v_\tau(a,\xi_m)\int_{\tau_m}^{\tau_{m+1}} \frac{d\xi}{\sqrt{\tau_n-\xi}} = 2\Delta\tau \sum_{m=0}^{n-1} \frac{v_\tau(a,\xi_m)}{\sqrt{\tau_n-\tau_{m+1}}+\sqrt{\tau_n-\tau_m}}. \tag{5.6}$$

This leads to the condition:

$$v_1^n - v_{-1}^n = \frac{4}{\sqrt{\pi}}\frac{1}{\sqrt{\rho}} \sum_{m=1}^{n} \frac{v_0^m - v_0^{m-1}}{\sqrt{n-m}+\sqrt{n-m+1}}. \tag{5.7}$$

By a purely implicit scheme to the heat equation at the artificial boundary $x_0 = a$, i.e.,

$$v_0^n - v_0^{n-1} = \rho\left(v_1^n - 2v_0^n + v_{-1}^n\right), \tag{5.8}$$

we can eliminate the fictitious value v_n^{-1} in (5.7) to obtain the discretized TBC of Han and Wu :

$$(1+2\rho+B)v_0^n - 2\rho v_1^n = (1+B)v_0^{n-1} - B\sum_{m=1}^{n-1} \tilde{l}^{n-m}\left(v_0^m - v_0^{m-1}\right), \tag{5.9}$$

with the abbreviation $B = \frac{4\sqrt{\rho}}{\sqrt{\pi}}$ and the convolution coefficients given in (5.5). On the fully discrete level, discretized TBCs such as (5.4) and (5.9) become less transparent and may result in unstable numerical scheme. Mayfield demonstrated this for a discretized TBC of the form (5.4) in the context of the Schrödinger equation.

5.1.3 Discrete TBC

To prevent numerical reflections at the artificial boundary and ensure unconditional stability of the resulting scheme, we will construct a discrete TBC in the next subsection, rather than adopting an ad-hoc discretization of the analytic TBC (4.37) as in Mayfield's approach or Han and Wu's approach . The discrete TBC entirely eliminates numerical reflections at the boundary without incurring additional computational costs compared to ad-hoc discretization strategies like (5.4) and (5.9).

5.1.4 Derivation of boundary conditions

We follow the process outlined in previous chapter at a purely discrete level to derive the Discrete TBC. We solve the discrete exterior problem, specifically equation (5.1), for $j \leq 1$. We then apply the Z-transformation for a fixed j:

$$Zv_j^{(n)} = \hat{v}_j(z) := \sum_{n=0}^{\infty} v_j^{(n)} z^{-n}, \quad |z| > R_{\hat{v_j}}, \tag{5.10}$$

where $R_{\hat{v}_j}$ denotes the convergence radius of the Laurent series. We aim to solve equation (5.1) explicitly for $j \leq 1$. Assuming the initial data $v_j^{(0)} = 0$ for $j \leq 1$, obtain the transformed exterior scheme:

$$\frac{2}{\rho}\frac{z-1}{z+1}\hat{v_j}(z) = \hat{v}_{j+1} - 2\hat{v}_j + \hat{v}_{j-1}, \quad j \leq 1. \tag{5.11}$$

The two linearly independent solutions of the resulting second-order difference equation (5.11) are given by:

$$\hat{v}_j(z) = (\nu_{1,2})^{j+1}(z), \quad j \leq 1, \tag{5.12}$$

where $\nu_{1,2}(z)$ are the solutions the quadratic equation:

$$\nu^2 - 2\left(1 + \frac{1}{\rho}\frac{z-1}{z+1}\right)\nu + 1 = 0. \tag{5.13}$$

As we seek decreasing modes when $j \to -\infty$, we require $|\nu_1| > 1$. Consequently, we obtain the Z-transformed discrete TBC:

$$\hat{v}_1(z) = \nu_1(z)\hat{v}_0(z). \tag{5.14}$$

It is now necessary to compute the inverse Z-transform of $\nu_1(z)$ to obtain the discrete TBC from (5.14). This can be done explicitly through a lengthy calculation, resulting in the discrete TBC:

$$v_1^{(n)} = l^{(n)} * v_0^{(n)} = \sum_{k=1}^{n} l^{(n-k)} v_0^{(k)}, \quad n \geq 1 \tag{5.15}$$

The convolution coefficients $l^{(n)}$ are provided. Due to the asymptotic behavior $l^{(n)} \sim 4(-1)^n/\rho$ potentially causing subtractive cancellation in (5.15), it is preferable to use the summed coefficients in the implementation:

$$s^{(n)} := l^{(n)} + l^{(n-1)}, \quad n \geq 1, \quad s^{(0)} := l^{(0)} \tag{5.16}$$

The DTBC can then be expressed as:

$$v_1^{(n)} - s^{(0)} v_0^{(n)} = \sum_{k=1}^{n-1} s^{(n-k)} v_0^{(k)} - v_1^{(n-1)}, \quad n \geq 1 \tag{5.17}$$

The convolution coefficients are given by:

$$s^{(0)} = 1 + \frac{1+\sqrt{1+2\rho}}{\rho}, \quad s^{(1)} = 1 - \frac{1}{\rho} - \frac{1}{\rho\sqrt{1+2\rho}},$$
$$s^{(n)} = -\frac{\sqrt{1+2\rho}}{\rho}\frac{\tilde{P}_{1(n)}(\mu) - \lambda^{-2}\tilde{P}_{n-2}(\mu)}{2n-1}, \quad n \geq 2, \tag{5.18}$$

where $\tilde{P}_{(n)}(\mu) := \lambda^{-n}P_n(\mu)$ denotes the "damped" Legendre polynomials ($P_0 \equiv \lambda^{-1}$, $\tilde{P}_{-1} \equiv 0$). The parameters λ, μ are given by:

$$\lambda = \frac{\sqrt{1+2\rho}}{\sqrt[+]{1-2\rho}}, \quad \mu = \frac{1}{\sqrt{1+2\rho}\,\sqrt[+]{1-2\rho}}$$

Alternatively, the convolution coefficients can be computed using the recursion formula:

$$s^{(n+1)} = \frac{2n-1}{n+1}\mu\lambda^{-1}s^{(n)} - \frac{n-2}{n+1}\lambda^{-2}s^{(n-1)}, \quad n \geq 2 \tag{5.19}$$

This can be used after calculating $s^{(n)}$, $n = 0, 1, 2$ using the formula (5.18).

In Fig. 5.1, the values of the summed coefficients $s^{(n)}$ are shown in a logarithmic plot. The rapid decay property $s^{(n)} = O(n^{-3/2})$ is evident, which motivates a simplified discrete TBC by restricting (5.17) to a convolution over the "recent past" (last M time levels):

$$v_1^{(n)} - s^{(0)}v_0^{(n)} = \sum_{k=n-M}^{n-1} s^{(n-k)}v_0^{(k)} - v_1^{(n-1)}, \quad n \geq 1 \tag{5.20}$$

It is important to mention that the stability of the resulting scheme has not been established. For a succinct examination of various discretization approaches for analytic TBCs, the development of the DTBC for a category of difference schemes applicable to a general convection-diffusion equation, and a stability demonstration for the recursion formula (5.19), we direct the reader to .

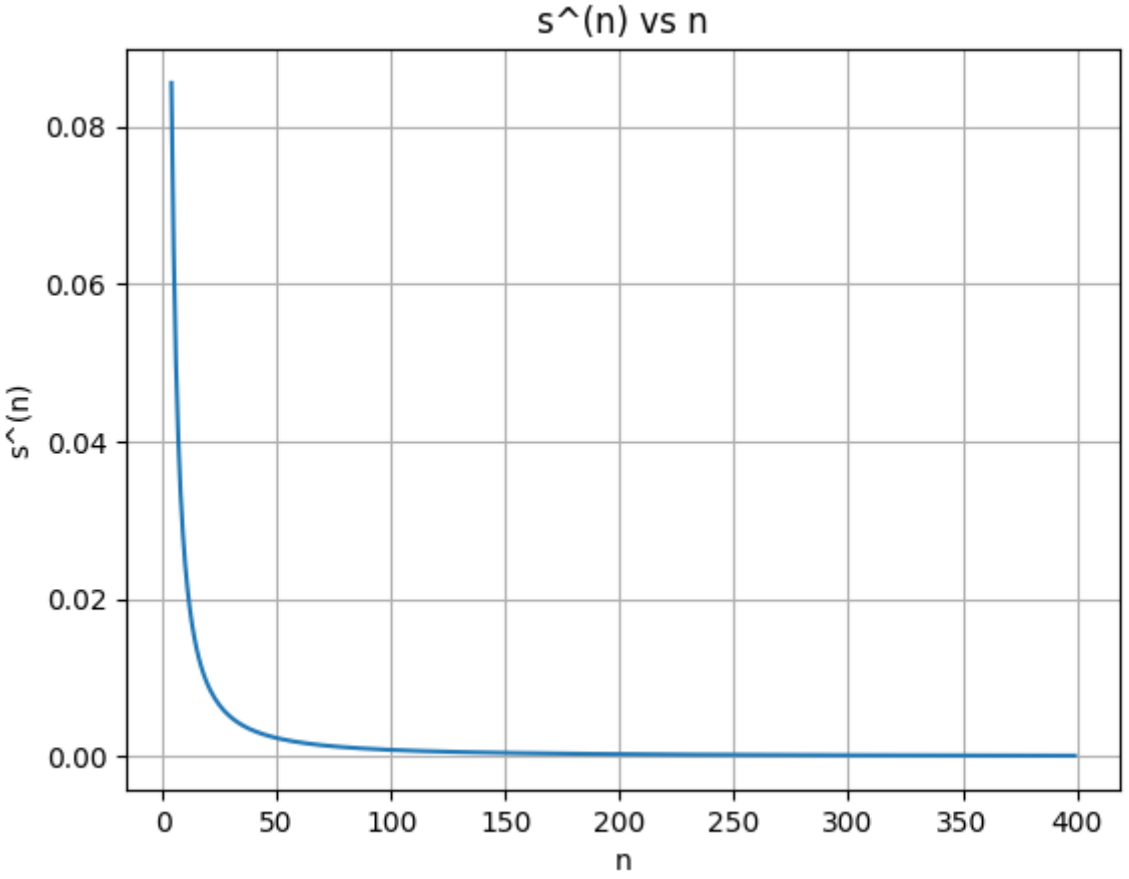

FIGURE 5.1: convolution coefficient s^n vs n

5.2 Approximation to change the non-local boundary condition

An ad-hoc implementation of the discrete convolution (5.17), with convolution coefficients $s^{(n)}$ from (5.18), still has a drawback. The boundary condition is nonlocal in time, making it computationally demanding. In fact, the evaluation of (5.17) is as costly as for the discretized TBCs (5.4), (5.9). To address this issue, we proposed the sum-of-exponentials ansatz. In the following, we briefly review this approach.

To develop a fast numerical method for calculating the discrete convolution in (5.17), we approximate the coefficients $s^{(n)}$ using the following sum of exponentials:

$$s^{(n)} \approx \tilde{s}^{(n)} := \begin{cases} s^{(n)}, & n = 0, 1 \\ \sum_{l=1}^{L} b_l q_l^{-n}, & n = 2, 3, \dots \end{cases} \tag{5.21}$$

where $L \in \mathbb{N}$ is a fixed number. Note that the approximation properties of $\tilde{s}^{(n)}$ depend on L, and the corresponding set $\{b_l, q_l\}$. In the following, we propose a deterministic method for finding $\{b_l, q_l\}$ for a fixed L.

The "split" definition of $\{\tilde{s}^{(n)}\}$ is inspired by the distinct nature of the first two coefficients in (5.18). Incorporating them into the discrete sum-of-exponential would result in less accurate approximation outcomes.

5.2.1 Rational Function Approximation

Rational Function Approximation is a technique used to approximate a given function with a rational function, which is the ratio of two polynomials. The goal is to find a rational function that closely matches the behavior of the original function over a certain interval.

Mathematically, a rational function can be expressed as:

$$R(x) = \frac{P(x)}{Q(x)}$$

where $P(x)$ and $Q(x)$ are polynomials of degree M and N respectively.

The Rational Function Approximation problem involves finding the coefficients of the numerator polynomial $P(x)$ and denominator polynomial $Q(x)$ that minimize the error between the rational function $R(x)$ and the target function $f(x)$ over a given interval.

This can be formulated as an optimization problem:

$$\min_{P(x),Q(x)} \sum_{i=1}^{n} [f(x_i) - R(x_i)]^2$$

where x_i are the data points in the given interval and n is the number of data points.

Solving this optimization problem typically involves using numerical optimization algorithms such as least squares regression or nonlinear optimization methods.

The resulting coefficients of the numerator and denominator polynomials can be used to evaluate the rational function at any desired point x to obtain the approximated value of the target function.

5.2.1.1 Approximating $s^{(n)}$ using Rational Function Approximation

To approximate the convolution coefficients $s^{(n)}$ using rational function approximation, we can express $s^{(n)}$ as a ratio of two polynomials:

$$s^{(n)} = \frac{P(n)}{Q(n)}$$

where $P(n)$ and $Q(n)$ are polynomials of degree m and n, respectively.

The goal is to find the coefficients of $P(n)$ and $Q(n)$ that best approximate the convolution coefficients $s^{(n)}$.

One common approach is to use least squares fitting to determine the coefficients. This involves minimizing the squared error between the approximated rational function and the true convolution coefficients.

Let's denote the true convolution coefficients as $s_n^{(n)}$ and the approximated coefficients as $\tilde{s}_n^{(n)}$.

The error between $s^{(n)}$ and $\tilde{s}^{(n)}$ can be defined as:

$$error = s_n^{(n)} - \tilde{s}_n^{(n)}$$

To find the coefficients of the rational function approximation, we can use the `np.polyfit()`, which performs polynomial fitting. Since we are working with rational functions, we need to rewrite the problem as a polynomial fitting by multiplying the denominator by the numerator:

$$s_n^{(n)} \cdot Q(n) - \tilde{s}_n^{(n)} \cdot P(n) = 0$$

We can rewrite this equation as:

$$s_n^{(n)} \cdot Q(n) - \tilde{s}_n^{(n)} \cdot P(n) = 0$$

$$s_n^{(n)} \cdot Q(n) = \tilde{s}_n^{(n)} \cdot P(n)$$

Now, we can use `np.polyfit()` to find the coefficients of $P(n)$ and $Q(n)$ that best approximate the convolution coefficients. The degree of $P(n)$ and $Q(n)$ can be adjusted based on the desired accuracy and complexity of the problem. the result is shown in Figure 5.2

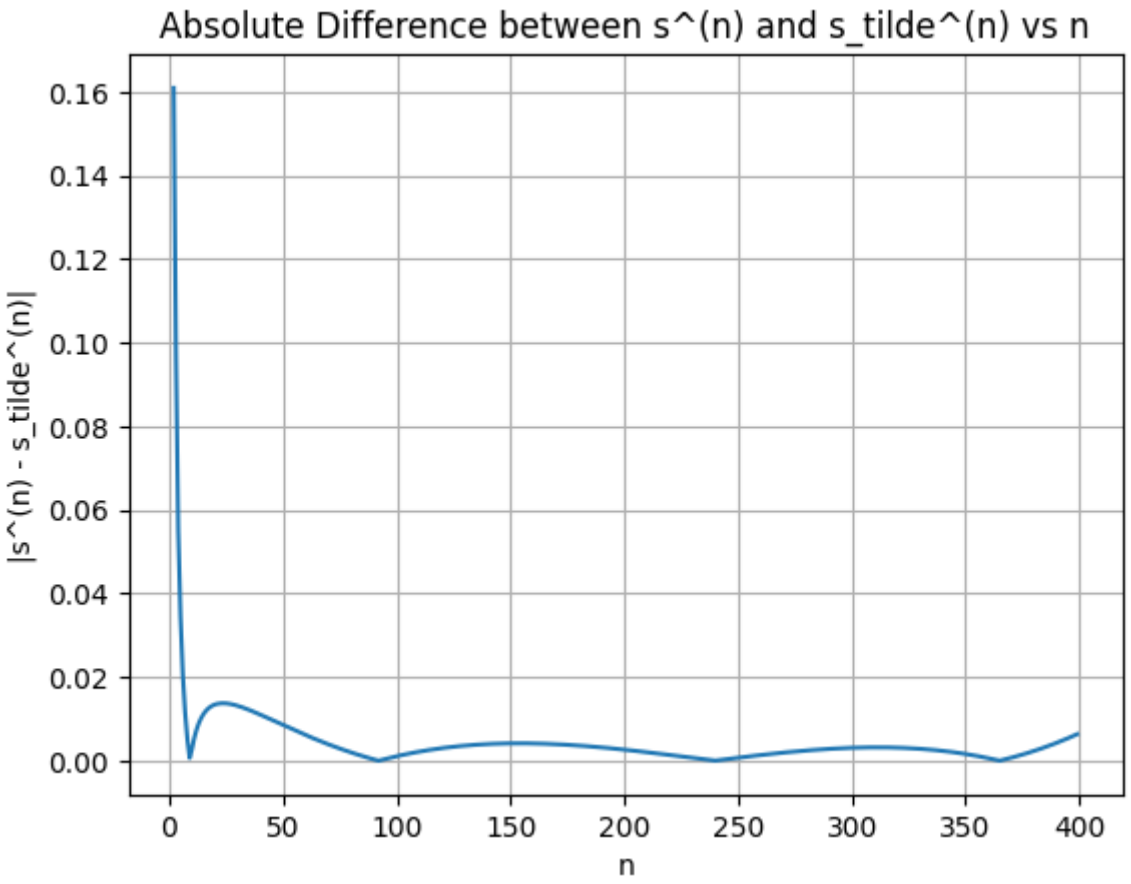

FIGURE 5.2: error $|s^{(n)} - \tilde{s}^{(n)}|$ vs n using Rational function approximation

5.2.2 Economized Rational Approximation

ERA is a method used to approximate a given function using a rational function with fewer parameters. The goal is to find the coefficients of a rational function that provides a good approximation of the original function while reducing complexity.

The rational function is expressed as the ratio of two polynomials:

$$s^{(n)} = \frac{P(n)}{Q(n)}$$

where $s^{(n)}$ represents the convolution coefficients, and $P(n)$ and $Q(n)$ are polynomials of degrees m and n respectively.

To determine the coefficients, an ERA function $f(n)$ is defined:

$$f(n) = \frac{P(n)}{Q(n)}$$

The goal is to minimize the squared error between the ERA function and the true convolution coefficients:

$$\text{minimize} \sum_{i=1}^{N} \left(s^{(n_i)} - f(n_i)\right)^2$$

where N is the number of data points available for the convolution coefficients.

The optimization problem can be solved using techniques such as least-squares fitting or optimization algorithms. The resulting coefficients of $P(n)$ and $Q(n)$ provide an approximation of the convolution coefficients $s^{(n)}$.

5.2.2.1 Approximation of $s^{(n)}$ using ERA

ERA is a technique used to approximate rational functions by reducing the degrees of the numerator and denominator polynomials. The ERA method aims to find the best approximation that minimizes the error between the true function and the approximated function. Here's the step-by-step process for using ERA to approximate the convolution coefficients $s^{(n)}$ in the given problem:

Economized Rational Approximation (ERA) is a technique used to approximate rational functions by reducing the degrees of the numerator and denominator polynomials. The process of using ERA to approximate the convolution coefficients $s^{(n)}$ in the given problem involves several steps.

Firstly, the value of ρ needs to be defined. This parameter is essential for determining the true convolution coefficients. Next, the range of n values is defined. These values represent the indices for which the convolution coefficients will be computed.

To calculate the true convolution coefficients $s^{(n)}$ for each n, a recursive formula is utilized. The formula considers previous values of $s^{(n-1)}$ and $s^{(n-2)}$ to compute the current coefficient $s^{(n)}$. After obtaining the true convolution coefficients, an ERA function is defined. This function aims to approximate the rational function by finding the best-fit coefficients of the numerator and denominator polynomials. To determine the optimal coefficients, an optimization algorithm is employed. The ERA function is fitted to the true convolution coefficients using this algorithm.

Once the coefficients are obtained, the numerator and denominator polynomials are separated. These coefficients are extracted for further evaluation. Using the obtained coefficients, the ERA function is evaluated at all values of n to calculate the approximated convolution coefficients $s^{(n)}_{\text{tilde}}$.

The next step involves computing the error between the true convolution coefficients $s^{(n)}$ and the approximated coefficients $s^{(n)}_{\text{tilde}}$. This provides insight into the accuracy of the approximation.Finally, the error is plotted on a graph, with the values of n on the x-axis and the absolute difference between $s^{(n)}$ and $s^{(n)}_{\text{tilde}}$ on the y-axis. This visual representation helps assess the quality of the approximation. By following these steps, Economized Rational Approximation can be utilized to approximate convolution coefficients $s^{(n)}$ in the given problem. the result are shown in Figure 5.3

5.2.3 Padé approximation

The Padé approximation is a rational function approximation that can be used to approximate a given function or series. It is particularly useful when the function

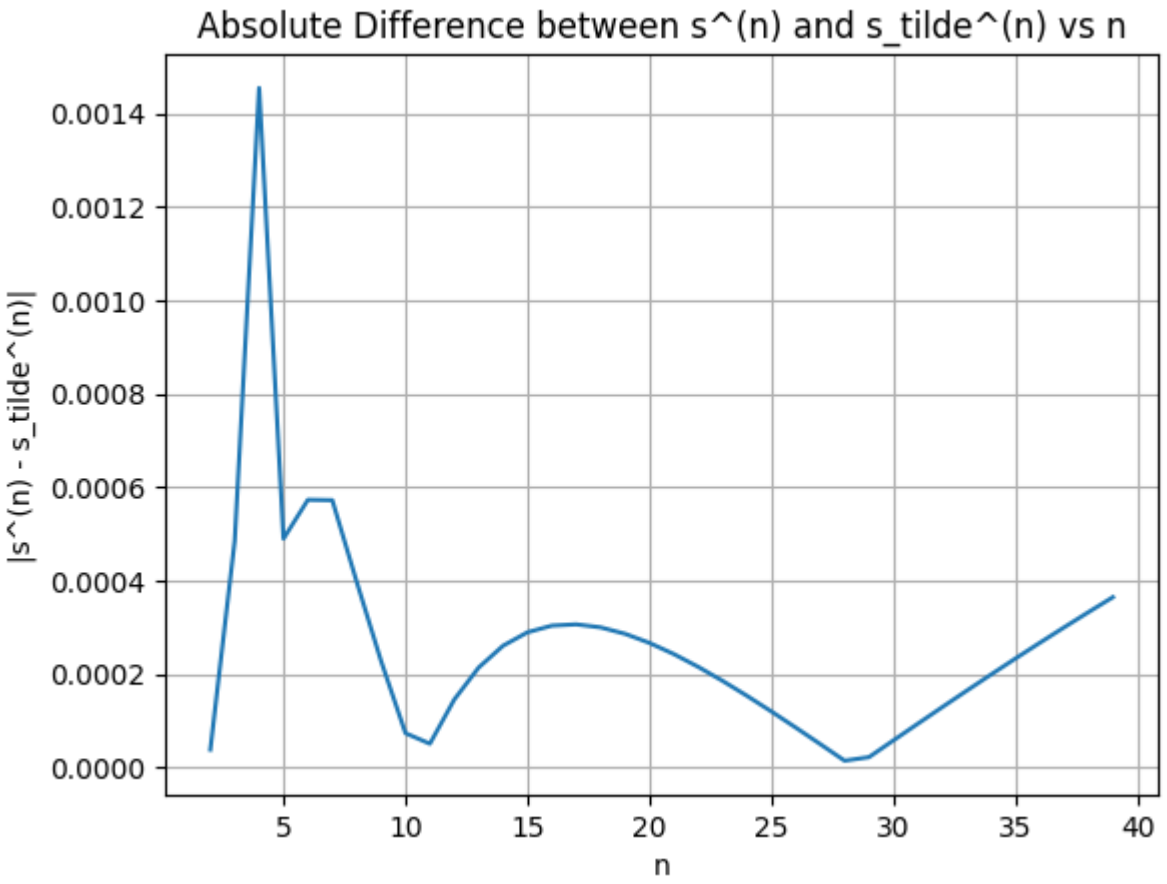

FIGURE 5.3: error $|s^{(n)} - \tilde{s}^{(n)}|$ vs n using Economized Rational Approximation

or series has singularities or when the function is not well-approximated by a polynomial. The Padé approximation is defined as follows: Given a function (f(x)) with a power series representation: $f(x) = \sum_{n=0}^{\infty} a_n x^n$ The Padé approximation of order ([M — N]) is a rational function of the form:

$$\tilde{f}(x) = \frac{P_M(x)}{Q_N(x)} = \frac{p_0 + p_1 x + \cdots + p_M x^M}{1 + q_1 x + \cdots + q_N x^N} \tag{5.22}$$

where $(P_M(x))$ is a polynomial of degree (M) and $(Q_N(x))$ is a polynomial of degree (N). The coefficients $(p_i) and (q_i)$ are chosen such that the power series expansion of ((x)) agrees with the power series expansion of (f(x)) up to the highest possible order. In other words, the first (M + N + 1) terms of the power series expansions of (f(x))

and ((x)) should be equal. The Padé approximation has several useful properties, including the ability to approximate functions with singularities and the ability to provide accurate approximations even when the function is not well-approximated by a polynomial. Additionally, Padé approximations can be used to accelerate the convergence of slowly converging series and to provide accurate approximations in regions where the original series converges slowly or not at all.

5.2.3.1 Approximation of $s^{(n)}$ using Padé Approximation

Let us fix L and consider the formal power series:

$$f(x) := s^{(2)} + s^{(3)}x + s^{(4)}x^2 + \dots, \quad |x| \leq 1. \tag{5.23}$$

If the $[L-1|L]$ Padé approximation of (5.23) exists,

$$\tilde{f}(x) := \frac{P_{L-1}(x)}{Q_L(x)}, \tag{5.24}$$

then its Taylor series

$$\tilde{f}(x) = \tilde{s}^{(2)} + \tilde{s}^{(3)}x + \tilde{s}^{(4)}x^2 + \dots \tag{5.25}$$

satisfies the conditions

$$\tilde{s}^{(n)} = s^{(n)}, \quad n = 2, 3, \dots, 2L+1, \tag{5.26}$$

due to the definition of the Padé approximation rule.

Theorem 5.1 ([5]). Let $Q_L(x)$ have L simple roots q_l with $|q_l| > 1$, $l = 1, \dots, L$. Then

$$\tilde{s}^{(n)} = \sum_{l=1}^{L} b_l q_l^{-n}, \quad n = 2, 3, \dots, \tag{5.27}$$

where

$$b_l := -\frac{P_{L-1}(q_l)}{Q'_L(q_l)} q_l \neq 0, \quad l = 1, \ldots, L. \tag{5.28}$$

It follows from (5.26) and (5.27) that the set $\{b_l, q_l\}$ defined in Theorem 5.1 can be used in (5.21) at least for $n = 2, 3, \ldots, 2L+1$. The main question now is: Is it possible to use these $\{b_l, q_l\}$ also for $n > 2L+1$? In other words, how good is the approximation $\tilde{s}^{(n)} \approx s^{(n)}$, $n > 2L+1$?

The above analysis allows us to provide the following description of the approximation to the convolution coefficients $s^{(n)}$ by the representation (5.21) if we use an $[L-1|L]$ Padé approximate for (5.23): the first $2L$ coefficients are reproduced exactly, see (5.26); however, the asymptotic behavior of $s^{(n)}$ and $\tilde{s}^{(n)}$ (as $n \to \infty$) differs significantly (algebraic versus exponential decay). A typical graph of $|s^{(n)} - \tilde{s}^{(n)}|$ versus n for $L = 5$ is shown in Fig. 5.4.

So far, we have explored the computation and approximation of the DTBC for a single fixed discretization. An appealing aspect of this method is that, after obtaining the approximate convolution coefficients $\{\tilde{s}(n)\}$ for a,

5.2.3.2 Converting for standard ρ

For a specific mesh ratio ρ, once the appropriate coefficients are obtained, it is straightforward to transform them for any other mesh ratio ρ^*.

Theorem 5.2. Consider a rational function

$$\hat{\tilde{s}}(z) := s^{(0)} + \frac{s^{(1)}}{z} + \sum_{l=1}^{L} \frac{b_l}{q_l z - 1} \tag{5.29}$$

which approximates the Z-transform of the convolution kernel $\{s(n)\}_{n=0}^{\infty}$ corresponding to a DTBC for the equation (5.1) with a given mesh ratio ρ ($\hat{\tilde{s}}$ is the Z-transform

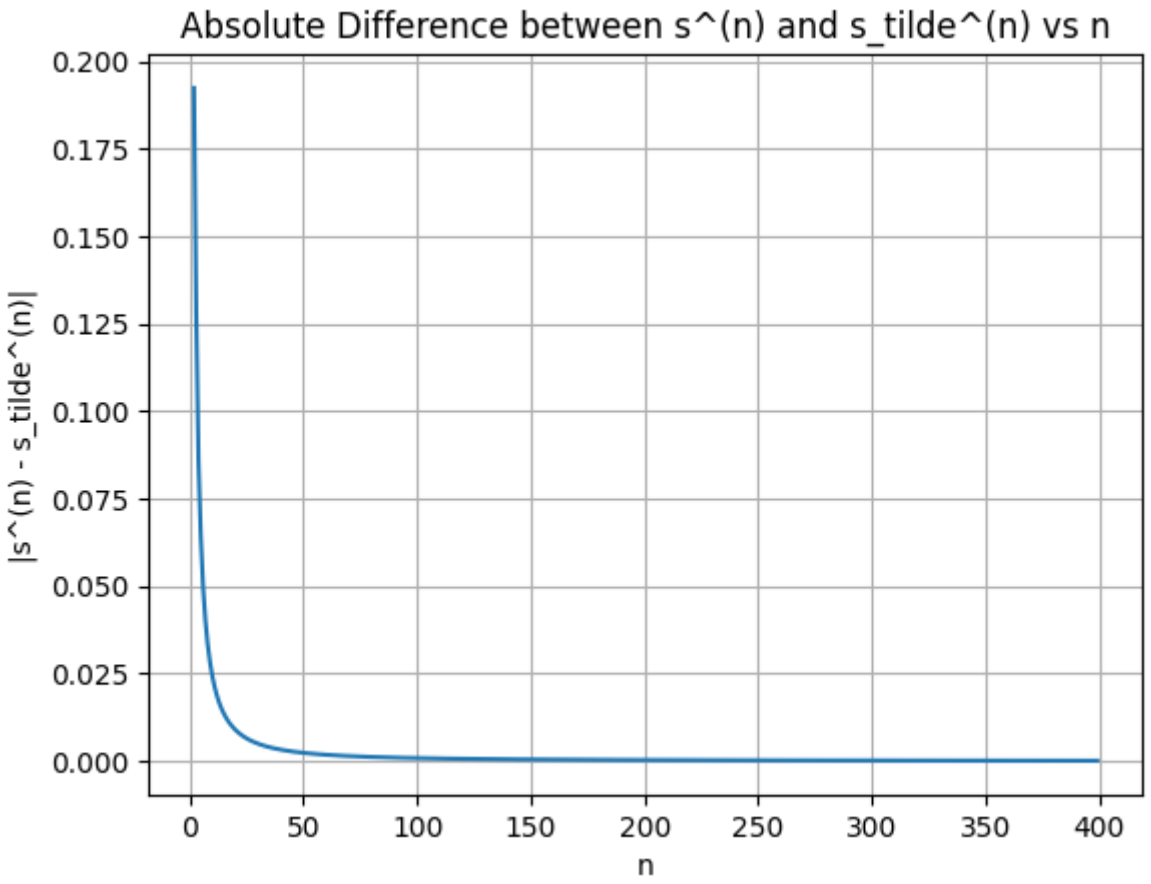

FIGURE 5.4: error $|s^{(n)} - \tilde{s}^{(n)}|$ vs n

of $\{\tilde{s}(n)\}$ from (5.21). For another mesh ratio ρ', the approximation can be taken as

$$\hat{\tilde{s}}(z) := {s_*}^{(0)} + \frac{{s_*}^{(1)}}{z} + \sum_{l=1}^{L} \frac{b_l^*}{{q_l}^* z - 1} \tag{5.30}$$

where

$${s_*}^{(0)} := \hat{\tilde{s}}\left(\frac{a}{b}\right) \quad (:= s^{(0)} \text{ if } b = 0), \tag{5.31}$$

$$b_l^* := b_l q_l \frac{a^2 - b^2}{(a - q_l b)(q_l a - b)} \frac{1 + q_l^*}{1 + q_l}, \quad q_l^* := \frac{q_l a - b}{a - q_l b}, \tag{5.32}$$

$$a := \frac{1}{\rho} + \frac{1}{\rho_*}, \quad b := \frac{1}{\rho} - \frac{1}{\rho_*}. \tag{5.33}$$

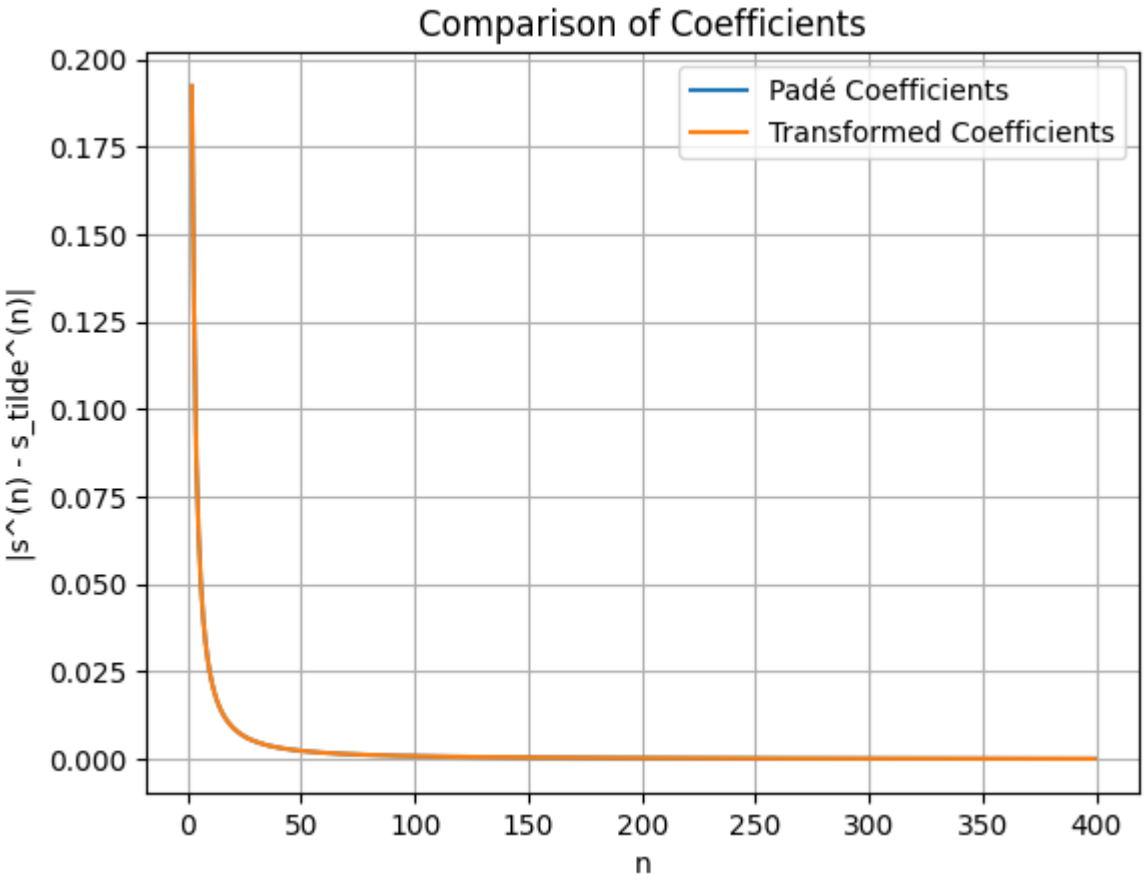

FIGURE 5.5: $\tilde{s}_*^{(n)}$ - s^n

While the Padé algorithm offers a technique for computing approximate convolution coefficients $\tilde{s}(n)$ for a fixed mesh ratio ρ, this transformation rule establishes a natural connection between different mesh ratios ρ_* (with L fixed).

Example 5.1. For $L = 5$, we computed the coefficients $\{b_l, q_l\}$ with the mesh ratio $\rho = 1$ and then applied Transformation rule 5.2 to calculate the coefficients $\{b_l^*, q_l^*\}$ for the mesh ratio $\rho_* = 0.8$. Figure 5.5 demonstrates that the resulting convolution coefficients $\tilde{s}_*^{(n)}$ are, in this example, even better approximations to the exact coefficients $s_*^{(n)}$ than the coefficients $\tilde{s}(n)$, which are obtained directly from the Padé algorithm discussed in Theorem 5.1. Consequently, the numerical solution of the corresponding heat equation is also more accurate.

5.3 Fast evaluation of convolution

Let's consider the approximation given by Eq. (5.21) for the discrete convolution kernel in the DTBC (5.17). Using these "exponential" coefficients, we can compute the convolution

$$C^{(n)} := \sum_{m=1}^{n-1} \tilde{s}^{(n-m)} v_0^{(m)}, \tag{5.34}$$

where

$$\tilde{s}^{(n)} = \sum_{l=1}^{L} b_l q_l^{-n} \tag{5.35}$$

and $|q_l| > 1$. The convolution of a discrete function $v_0^{(m)}$, with $m = 1, 2, \ldots$, and the kernel coefficients $\tilde{s}(n)$ can be calculated using recurrence formulas, which significantly reduces the numerical effort.

A simple computation provides: The value $C(n)$, from equation (5.35) for $n \geq 2$, can be expressed as

$$C(n) = \sum_{l=1}^{L} C_l^{(n)}, \tag{5.36}$$

where

$$C_l^{(1)} \equiv 0, \tag{5.37}$$

$$C_l^{(n)} = q_l^{-1} C_l^{(n-1)} + b_l q_l^{-1} v_0^{(n-1)}, \quad n = 2, 3, \ldots, l = 1, \ldots, L. \tag{5.38}$$

We note that the Pad'e approximation requires high precision ($2L-1$) digits mantissa length) to prevent a "near breakdown" due to ill-conditioned steps in the Lanczos algorithm. If these issues persist or if the absolute value of one root of the denominator is less than 1, the orders of the numerator and denominator polynomials are progressively decreased.

5.4 Numerical Treatment of the Free Boundary

We will provide a concise overview of the numerical treatment of the free boundary, denoted as $x_f(\tau)$ in equations (4.13) to (4.18).

There is currently no exact analytical formula for the free boundary profile $x_f(\tau)$ in equation (4.13). However, several authors have proposed approximate expressions for valuing American call and put options. A promising approach by Sevcovic yielded a semi-explicit formula for an American call when $r > D_0$. This was achieved by transforming equation (4.1) into a nonlinear parabolic equation on a fixed domain and applying Fourier sine and cosine transformations to derive a nonlinear singular integral equation that determines the free boundary shape. This integral equation can be effectively solved through successive iterations.

Given that the Black-Scholes equation (4.1) links $V(S,t)$ to $S_f(t)$, we opt to numerically determine the option value in conjunction with the free boundary. Numerous numerical methods have been developed for this purpose, such as the standard method which involves reformulating to a linear complementary problem and solving it using Cryer's projected SOR method . Other methods include penalty and front-fixing methods , but these alter the underlying model. An alternative approach relies on a recursive calculation of the early exercise boundary, estimating the boundary at certain points and then approximating the entire boundary using Richardson extrapolation. Explicit boundary tracking algorithms include a finite difference bisection scheme and the front-tracking strategy of Han and Wu. In this work, we will employ the latter approach, which we will briefly describe next.

Han and Wu applied the strong maximum principle for parabolic equations to the Black-Scholes equation for the derivative V_S and the equation (4.1) extended to the time-independent domain $S > 0$, known as the Jamshidian equation. The result

is a useful inequality for numerically determining the location of the free boundary $x_f(\tau)$. For a given τ, the free boundary is the unique point that satisfies both equation (4.13) and the high contact condition $V_S(S,t) = 1$, i.e., (4.18). If the boundary condition $v(x,\tau) = g(x,\tau)$ is imposed at some point $x > x_f(\tau)$, then $v(x,\tau) < g(x,\tau)$ will occur for some $x < x_f(\tau)$. To solve the Crank-Nicolson scheme (5.1), Han and Wu used the common Thomas algorithm for the resulting tridiagonal system. Once the boundary condition $v_{J+1}^{n+1} = g_{J+1}^{n+1}$, with $g_J^n = g(x_J, \tau_n)$, is given at some grid point x_{J+1}, the backward sweep of the Thomas algorithm calculates the solution v_j^{n+1} for all $0 \le j \le J$. The index J is simply the largest index such that $v_J^{n+1} \ge g_J^{n+1}$ holds.

5.5 Analysis of Stability

In this analysis, we examine the stability of the Crank-Nicolson scheme (5.1) in conjunction with the DTBC (5.17) or its approximated variant. Our primary focus is on demonstrating that the (approximated) DTBC does not compromise the unconditional stability of the underlying finite difference scheme. To this end, we consider the following problem on the half-space $j \ge 0$:

$$\begin{cases} v_j^{(n+1)} - v_j^{(n)} = \rho\left(v_{j+1}^{(n+\frac{1}{2})} - 2v_j^{(n+\frac{1}{2})} + v_{j-1}^{(n+\frac{1}{2})}\right), & j \ge 1, \\ v_j^{(0)} = g(x_j, 0), & j = 0, 1, 2, \dots \\ \text{with } v_0^{(0)} = v_1^{(0)} = 0, \\ \hat{v}_1(z) = \hat{l}(z)\hat{v}_0(z), \end{cases}$$

Where the transformed boundary kernel $\hat{\nu}(z) = \nu_1(z)$ is given by (5.14). In the following sections, we aim to bound the exponential growth of solutions to the numerical scheme for a fixed mesh ratio. We will establish an estimate of the discrete solution

We define the discrete L^2-norm as:

$$||v^{(n)}||_2^2 := \Delta x \sum_{j=1}^{\infty} |v_j^{(n)}|^2. \tag{5.39}$$

Theorem 7.1 (Growth Condition). Let the transformed boundary kernel $\hat{l}$ satisfy

$$|\hat{l}(\beta e^{i\phi})| \geq 1, \quad \forall 0 \leq \phi \leq 2\pi, \tag{5.40}$$

for some (sufficiently large) $\beta \geq 1$. Assume also that $\hat{l}(z)$ is analytic for $|z| \geq \beta$. Then, the solution of (7.1) satisfies the a-priori estimate in the discrete L^2-norm:

$$||v^{(n+1)}||_2 \leq \beta^n \left(||v^{(0)}||_2 + \sqrt{\frac{(\beta-1)\rho}{2}} ||\Delta^- v^{(0)}||_2 \right), \quad n \in \mathbb{N}_0. \tag{5.41}$$

Proof. The proof is based on a discrete energy estimate for the new variable

$$u_j^{(n)} := v_j^{(n)} \beta^{-n}, \tag{5.42}$$

which fulfills

$$\beta^{-n}(v_j^{(n+1)} \pm v_j^{(n)}) = u_j^{(n+1)} \pm u_j^{(n)} + (\beta - 1) u_j^{(n+1)},$$

and therefore satisfies

$$u_j^{(n+1)} - u_j^{(n)} = \rho\left(u_{j+1}^{(n+\frac{1}{2})} - 2u_j^{(n+\frac{1}{2})} + u_{j-1}^{(n+\frac{1}{2})}\right) + (\beta-1)\left(\frac{\rho}{2}\left(u_{j+1}^{(n+1)} - 2u_j^{(n+1)} + u_{j-1}^{(n+1)}\right) - u_j^{(n+1)}\right),$$
$$j \geq 1 \tag{5.43}$$

$$u_j^{(0)} = v_j^{(0)}, \qquad j = 0, 1, 2, \ldots \tag{5.44}$$

$$\Delta^+ \hat{u}_{(0)}(z) = (\hat{l}(\beta z) - 1)\hat{u}_{(0)}(z). \tag{5.45}$$

The transformed discrete TBC (5.45) can be written in physical space as

$$\Delta^+ u_0^{(n)} = \frac{\tilde{l}^{(n)}}{\beta^n} * u_0^{(n)} = \sum_{m=0}^{n} (\tilde{l}^{(n-m)}\beta^{m-n}) u_0^{(m)}, \tag{5.46}$$

where $\tilde{l}^{(n)} := l^{(n)} - \delta_0^n$ is given in (5.15) and $\Delta^+ u_0^{(n)} = u_1^{(n)} - u_0^{(n)}$ denotes the usual forward difference. First, we multiply (5.43) by $u_j^{(n)}/\beta$ and then by $u_j^{(n+1)}$:

$$u_j^{(n)}\left(u_j^{(n+1)} - u_j^{(n)}\right) = \rho u^{(n)} j\left(u_{j+1}^{(n+\frac{1}{2})} - 2u_j^{(n+\frac{1}{2})} + u_{j-1}^{(n+\frac{1}{2})}\right)$$
$$-\beta^{-1}(\beta-1)u_j^{(n)}\left(\frac{\rho}{2}\left(u_{j+1}^{(n)} - 2u_j^{(n)} + u_{j-1}^{(n)}\right) + u_j^{(n)}\right),$$
$$u_j^{(n+1)}\left(u_j^{(n+1)} - u_j^{(n)}\right) = \rho u_j^{(n+1)}\left(u_{j+1}^{(n+\frac{1}{2})} - 2u_j^{(n+\frac{1}{2})} + u_{j-1}^{(n+\frac{1}{2})}\right)$$
$$+(\beta-1)u_j^{(n+1)}\left(\frac{\rho}{2}\left(u_{j+1}^{(n+1)} - 2u_j^{(n+1)} + u_{j-1}^{(n+1)}\right) - u_j^{(n+1)}\right). \tag{5.47}$$

Observe that we used equation (5.43) to modify the last term of (5.47). Next, we add both equations in (5.47), sum it up for the range $j = 1, 2, \ldots$ and obtain using

the summation by parts rule:

$$\sum_{j=1}^{\infty}\Big((u_j^{(n+1)})^2-(u_j^{(n)})^2\Big)=-2\rho\sum_{j=1}^{\infty}(\Delta^- u_j^{(n+\frac{1}{2})})^2-(\beta-1)\frac{\rho}{2}\sum_{j=1}^{\infty}(\Delta^- u_j^{(n+1)})^2+\frac{\beta-1}{\beta}\frac{\rho}{2}\sum_{j=1}^{\infty}(\Delta^- u_j^{(n)})^2$$
$$-(\beta-1)\sum_{j=1}^{\infty}(u_j^{(n+1)})^2+$$
$$\frac{\beta-1}{\beta}\sum_{j=1}^{\infty}(u_j^{(n)})^2\ -\frac{\rho}{2\beta}(u_0^{(n)}+\beta u_0^{(n+1)})\Delta^+(u_0^{(n)}+\beta u_0^{(n+1)}),$$

where $\Delta^- u_j^{(n)}=u_j^{(n)}-u_{j-1}^{(n)}$ denotes the backward difference. Now summing this equation from time level $n=0$ to $n=N$ yields:

$$\begin{aligned}\beta||(u^{(N+1)})||_2^2=\beta^{-1}||(u^{(0)})||_2^2-\frac{(\beta^2-1)}{\beta}\sum_{n=1}^{N}||(u^{(n)})||_2^2-2\rho\sum_{n=0}^{N}||(\Delta^- u^{(n+\frac{1}{2})})||_2^2\\-\frac{(\beta-1)^2}{2\beta}\rho^2\sum_{n=1}^{N}||(\Delta^- u^{(n)})||_2^2+\frac{(\beta-1)}{2\beta}\rho||(\Delta^- u^{(0)})||_2^2\\-\frac{(\beta-1)}{2}\rho||(\Delta^- u^{(N+1)})||_2^2-\frac{\rho}{2\beta}\sum_{n=0}^{N}(u_0^{(n)}+\beta u_0^{(n+1)})\Delta^+(u_0^{(n)}+\beta u_0^{(n+1)}).\end{aligned}\tag{5.48}$$

Noting that $\beta\geq 1$, we obtain from (5.48) the following estimate:

$$||u^{(N+1)}||_2^2\leq\beta^{-2}||u^{(0)}||_2^2+\frac{(\beta-1)}{2\beta^2}\rho||\Delta^- u^{(0)}||_2^2-\frac{\rho}{2\beta^2}\sum_{n=0}^{N}(u_0^{(n)}+\beta u_0^{(n+1)})\Delta^+(u_0^{(n)}+\beta u_0^{(n+1)}).\tag{5.49}$$

It remains to show that the boundary-memory-term in (5.49) is of positive type. To this end, we define (for N fixed) the two sequences,

$$g(n):=\begin{cases}u_0^{(n)}+\beta u_0^{(n+1)}, & n=0,\dots,N,\\ 0, n>N,\end{cases}$$

$$f^{(n)} := \frac{\tilde{l}^{(n)}}{\beta^n} * g^{(n)} = \sum_{m=0}^{n} \frac{\tilde{l}^{(n-m)}}{\beta^{n-m}} g^{(m)}, \quad n \in \mathbb{N}_0, \tag{5.50}$$

To demonstrate that $\sum_{n=0}^{N} f^{(n)} g^{(n)} \geq 0$, we consider the Z-transform $Z\{f^{(n)}\} = \hat{f}^{(z)}$, which is analytic for $|z| > 0$ as it is a finite sum. The Z-transform $Z\{f^{(n)}\}$ satisfies $\hat{f}(z) = (\hat{l}(\beta z) - 1)\hat{g}(z)$ and is analytic for $|z| \geq 1$. By applying Plancherel's Theorem for Z-transforms, we obtain

$$\begin{aligned} \sum_{n=0}^{N} f^{(n)} g^{(n)} &= \frac{1}{2\pi} \int_0^{2\pi} \hat{f}(e^{i\phi}) \hat{} (\bar{e^{i\phi}}) \bar{g} d\phi \\ &= \frac{1}{\pi} \int_0^{\pi} \left| Re(\hat{f}(e^{i\phi}) \hat{} (\bar{e^{i\phi}} \bar{g})) \right| d\phi \\ &= \frac{1}{\pi} \int_0^{\pi} |\hat{g}(e^{i\phi})|^2 \Big| Re(\hat{l}(\beta e^{i\phi}) - 1 \Big|) d\phi, \end{aligned} \tag{5.51}$$

where we used the properties $\hat{f}(\bar{z}) = \overline{\hat{f}(z)}$ and $\hat{g}(\bar{z}) = \overline{\hat{g}(z)}$, since $f_n, g_n \in \mathbb{R}$. Using this expression for the boundary term in the inequality, we get:

$$\begin{aligned} \left| ||u_(N+1) \right| |_2 \leq \\ \beta^{-2} \left| ||u^{(0)}| \right| |_2^2 + \frac{\beta - 1}{\beta^2} \frac{\rho}{2} ||\Delta^- u^{(0)}||_2^2 \\ - \frac{\rho}{2\pi\beta^2} \int_0^{\pi} \left| (1 + \beta e^{i\phi}) \hat{u}_0(e^{i\phi}) \right|^2 \Big(Re(\hat{l}(\beta e^{i\phi})) - 1 \Big) d\phi. \end{aligned} \tag{5.52}$$

Under our assumption on $\hat{v}$, this implies

$$\left| ||u^{(N+1)}| \right| |_2 \leq \beta^{-1} \left| ||u^{(0)}| \right| |_2 + \frac{\sqrt{\beta - 1}}{\beta} \sqrt{\frac{\rho}{2}} (\left| ||\Delta^- \left| u^{(0)} \right| |_2 \right), \quad \forall N \geq 0, \tag{5.53}$$

from which the theorem's result follows.

5.6 Numerical Experiments

Now examine the two examples of American call options. We compare the numerical outcomes obtained using our newly developed (approximated) discrete TBC with those obtained using the discretized TBC (5.4) or (5.9), as well as the explicit free boundary treatment detailed in Section 5.4. As the method proved to be more effective than the projected SOR method with asymptotic boundary conditions, we will only compare our results to the approach employed by Han and Wu. Moving forward, we will use years as the time dimension and US dollars as the value dimension.

5.6.1 Experiment 1

We examine an American call option with an expiration time of $T = 0.5$ years and a dividend yield $D_0 = 0.03$. The risk-free interest rate is given by $r = 0.03$, the volatility is $\sigma = 40\%$ per annum, and the exercise price is $E = \$100$. We select a mesh ratio $\rho = 1$ and compute $N = 400$ time steps with various artificial boundary conditions at the left boundary $a = x_0 = -1.0$, which corresponds to an asset price $S = Ee^a \approx 36.79$.

Option values $V(S, 0)$ calculated using the precise discrete TBC (5.17) are depicted in Figure 5.6. It is worth noting that for all $x < a$, the option values can be computed using equation (4.38) at the final time $\tau = \tilde{T}$, specifically at $t = 0$.

An estimated maximum for the free boundary $x_f(\tau)$ was computed using equation (4.19), resulting in $x_f^* = 1.5$. Nevertheless, the actual maximum value of $x_f(\tau)$ is significantly lower, approximately 0.92. The time progression of the non-decreasing free boundary $x_f(\tau)$ can be observed in Figure 5.7.

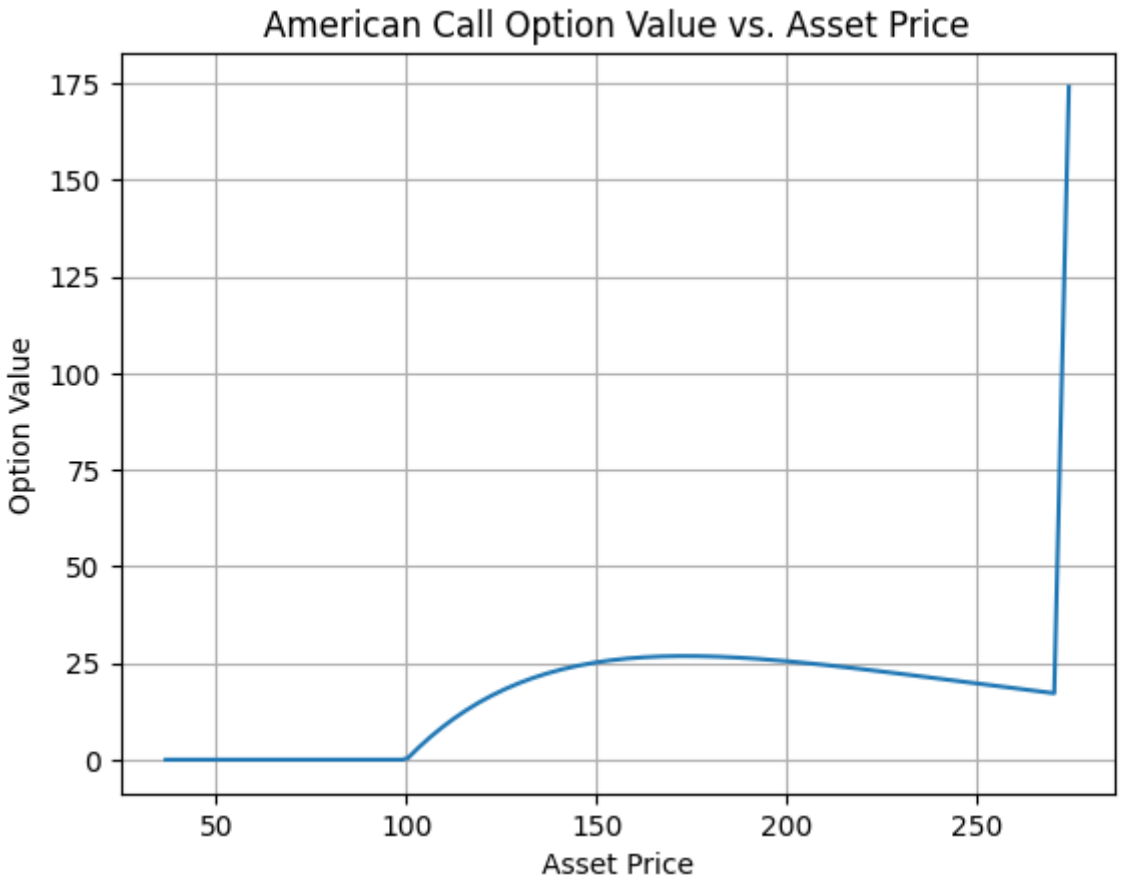

FIGURE 5.6: Option Value VS Stock Price

Next, our objective is to analyze the stability of the scheme using the approximate discrete TBC (5.21) with L = 10 exponentials. Therefore, we need to numerically verify the growth condition (5.40) necessary for stability. It was found that (5.40) holds true for all $\beta <= 1.42$. Figure 5.8 illustrates the real part of the transformed kernel $\widehat{\widetilde{\mathrm{I}}}(z)$ of the approximated DTBC on the circle $z = \beta e^{i\phi}$, where $\beta = -1.42$.

5.6.2 Experiment on American call option for Apple Inc. (AAPL)

the values in your example with real-life American call option data, we first need to identify a specific option. Let's consider an American call option for Apple Inc.

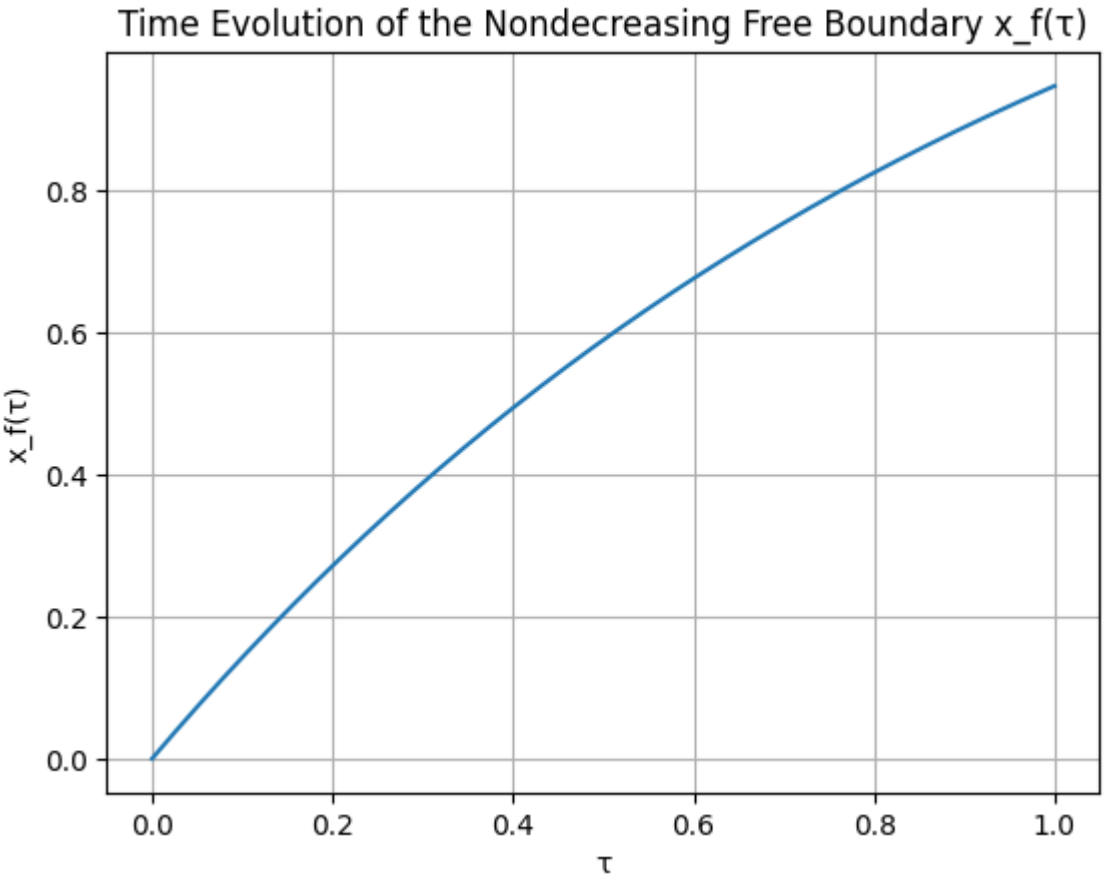

FIGURE 5.7: $x_f(\tau)$ VS τ

(AAPL) with the following parameters:

- Expiry: T = 1 year (Typically, real-life options have expiry dates ranging from one month to several years. Here, we'll consider a one-year expiry for simplicity.)
- Dividend yield: D0 = 0.006 (As of 2023, Apple's dividend yield was approximately 0.6
- Risk-free interest rate: r = 0.01 (This is an estimated value. The actual risk-free rate can vary. Typically, the 3-month U.S. Treasury bill rate is used as the risk-free rate.)

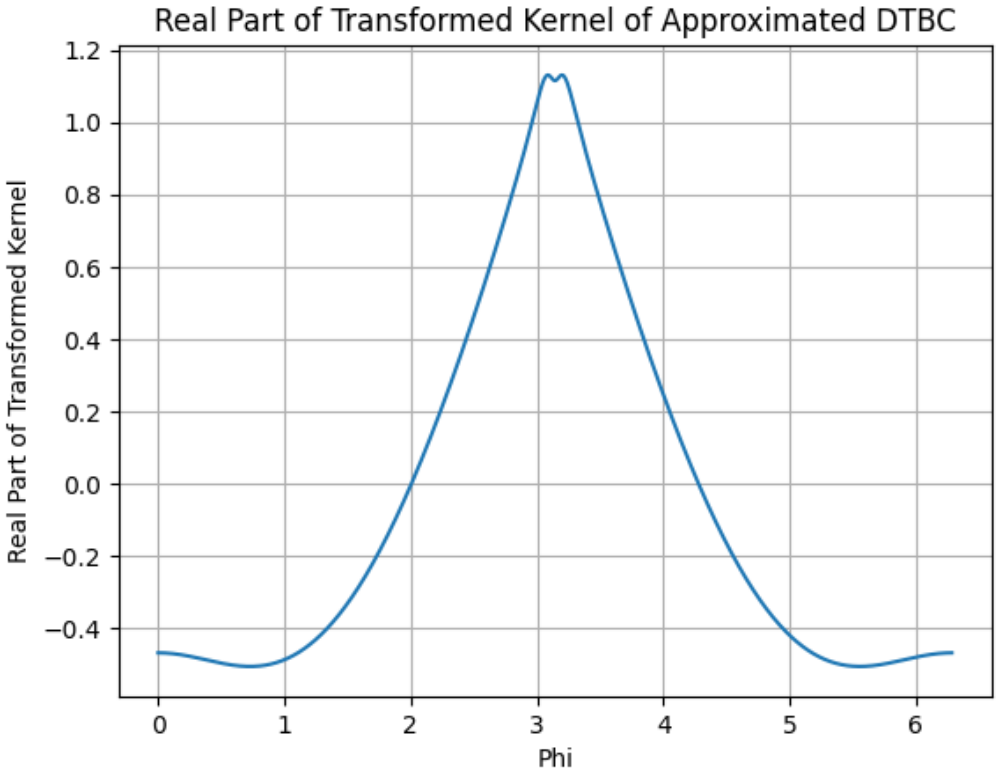

FIGURE 5.8: Real part of the transformed kernel $\widehat{\widetilde{\mathbf{l}}}(z)$ of the approximated DTBC on the circle $z = \beta e^{i\phi}$ with $\beta = -1.42$.

- Volatility: = 0.25 (This is an estimated value. The actual volatility, or standard deviation of the stock's returns, can vary significantly.)
- Exercise price: E = 150 (This is a hypothetical strike price. The actual strike price will depend on the specific option contract.)
- Mesh ratio: = 1 (This is a parameter for the numerical method used to price the option, and it doesn't change based on the specific option.)
- Number of time steps: N = 400 (This is another parameter for the numerical method, and it also doesn't change based on the specific option.)
- Left boundary: a = x0 = -1.0 (This is another parameter for the numerical method, and it also doesn't change based on the specific option.)

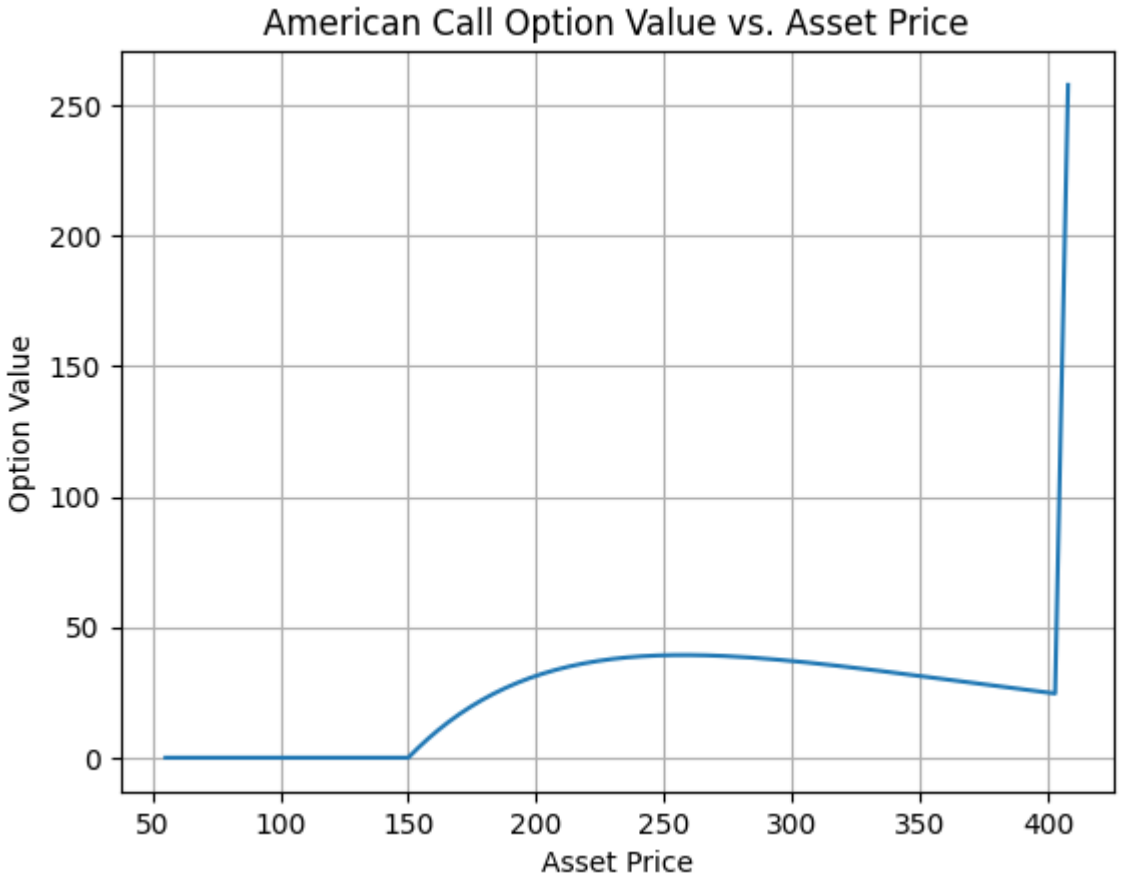

FIGURE 5.9: Option value VS stock price for AAPL

Scope for Future Work

New and improved approximation methods, it is possible to develop a significantly faster numerical method for solving the American options pricing problem with non-local boundary conditions. The current approximation techniques, such as Padé approximation, rational function approximation, and Economized Rational Approximation, have demonstrated their effectiveness in approximating the coefficients of non-local boundary conditions.

Appendix A

A Numerical Method for Heat Equation with Non-Local Boundary Condition

Code snippet A.1 spectral radius .

```
    function [U] = mtp3(M, q, phi, psi, n, U0)

delx = 1 / (M-1);
delt = 1 / n;
v = sqrt(delx*delx / delt);
theta = v/2;
eps = [
    (delx/v) * (exp(-theta) - exp(theta)),
    (delx/v) * (exp(-theta) - exp(theta)),
];
u1 = [
    1 - delx/v*(exp(theta) - 1), 1 + delx/v*(exp(-theta) - 1);
      - delx/v*(exp(theta) - 1),     delx/v*(exp(-theta) - 1);
];
```

```
um = [
        delx/v*(exp(-theta) - 1),   - delx/v*(exp(theta) - 1);
    1 + delx/v*(exp(-theta) - 1), 1 - delx/v*(exp(theta) - 1);
];

A = zeros(2*M);

K = [
    exp(theta), exp(-theta), -exp(-theta), -exp(theta);
    exp(theta), -exp(-theta), -exp(-theta), exp(theta);
];



for i = 1:M-1
    A(2*i-1:2*i, 2*i-1:2*i+2) = K;
end



A(2*M-1:2*M, 1:2) = u1;
A(2*M-1, 3:2*M-2) = eps(1);
A(2*M, 3:2*M-2) = eps(2);
A(2*M-1:2*M, 2*M-1:2*M) = um;

% disp(A);
% disp(epsilon);
% disp(u1);
% disp(um);



N = n+1;

zet = zeros(1,M);
U = zeros(N, M);
U(1, 1:M) = U0;

for n = 2:N+1
    for i = 1:M
        zet(i) = U(n-1, i) + delt * q(delt*(n-1), delx*(i-1));
    end
    g = zeros(2*M, 1);
    for i = 1:M-1
```

```
        g(2*i-1) = zet(i+1) - zet(i);
    end
    s1 = sum(phi(2:M-1).*zet(2:M-1));
    sm = sum(psi(2:M-1).*zet(2:M-1));
    g(2*M-1) = (-1 + phi(1)*delx/2) * zet(1) + delx * s1 + delx/2 * phi(M)*zet(M);
    g(2*M) = delx/2 * psi(1)*zet(1) + delx * sm + (-1 + psi(M)*delx/2) * zet(M);

    C = A\g;
    for i = 1:M
        U(n, i) = C(2*i-1) + C(2*i) + zet(i);
    end
end
end
```

LISTING A.1: spectral radius

A.0.1 Example 1

Code snippet A.3 Example 1.

```
delx = 0.0025;
%delx = 0.001;
m = 400;
% m = 1000;

n = 400;
%n = 1000;
delt = 0.0025;
%delt = 0.001;

M = m+1;
N = n+1;
v = sqrt(delx*delx / delt);
theta = v/2;
```

```
delta = 0.0144;

q = @(t, x) -exp(-t)*(x*(x-1) + delta/6/(1+delta) + 2);
psi = -delta * ones(1, M);
phi = -delta * ones(1, M);

X = linspace(0, 1, M);
U0 = X.*(X-1) + delta/6/(1+delta);

u1 = [
    1 - delx/v*(exp(theta) - 1), 1 + delx/v*(exp(-theta) - 1);
      - delx/v*(exp(theta) - 1),     delx/v*(exp(-theta) - 1);
];
U = mtp3(M, q, phi, psi, n, U0);
T = linspace(0, 1, N);
Uf = @(t, x) exp(-t).*(x*(x-1) + delta/6/(1+delta));
figure
plot(T, Uf(T, 0.1))
hold on
plot(T,U(1:N, floor(m*0.1)+1))
% size(U(1:N, 31))
% size(Uf(T, 0.3))

err03 = U(1:N, floor(m*0.32)+1).'-Uf(T, 0.32);
err05 = U(1:N, floor(m*0.5)+1).'-Uf(T, 0.5);
err07 = U(1:N,  floor(m*0.64)+1).'-Uf(T, 0.64);

size(err03);
figure
plot(T, -err03, T, -err05, T, -err07);
```

Listing A.2: Example 1

A.0.2 Example 2

Code snippet **??** Example 2.

```
delx = 0.0025;
```

```
m = 400;

n = 400;
delt = 0.0025;

M = m+1;
N = n+1;

psic = 3/2;
phic = 1/2;

q = @(t, x) 2*t*(1+2*x);
psi = psic * ones(1, M);
phi = phic * ones(1, M);

X = linspace(0, 1, M);
U0 = 1+2*X;



v = sqrt(delx*delx / delt);
c1 = phic*delx/v;
c2 = psic*delx/v;
e1 = exp(v/2);
e2 = exp(-v/2);

u1 = [
    1 - c1*(e1 - 1), 1 + c1*(e2 - 1);
      - c2*(e1 - 1),     c2*(e2 - 1);
];
um = [
        c1*(e2 - 1),   - c1*(e1 - 1);
    1 + c2*(e2 - 1), 1 - c2*(e1 - 1);
];

A = zeros(2*M);

K = [
    e1, e2, -e2, -e1;
    e1, -e2, -e2, e1;
];
```

```
for i = 1:M-1
    A(2*i-1:2*i, 2*i-1:2*i+2) = K;
end



A(2*M-1:2*M, 1:2) = u1;
A(2*M-1, 3:2*M-2) = c1 * (e2 - e1);
A(2*M, 3:2*M-2) = c2 * (e2 - e1);
A(2*M-1:2*M, 2*M-1:2*M) = um;

U = mtp4(M, q, A, phi, psi, n, U0);

T = linspace(0, 1, N);
Uf = @(t, x) (1+t.*t).*(1+2*x);
figure;
plot(T, Uf(T, 0.1));
hold on;
plot(T,U(1:N, floor(m*0.1)+1))

err03 = U(1:N, floor(m*0.32) + 1).'-Uf(T, 0.32);
err05 = U(1:N, floor(m*0.5) + 1).'-Uf(T, 0.5);
err07 = U(1:N,  floor(m*0.64) + 1).'-Uf(T, 0.64);

size(err03);
figure
plot(T, err03, T, err05, T, err07);
```

Listing A.3: Example 2

Appendix B

Stability of Numerical Scheme for Parabolic Problem With Non-Local Boundary Conditions

Code snippet B.1 for calculating $\gamma_1 vs \gamma_2$.

```
N=100;
sigma=  0.5;
tau = 0.0001;
T = 100;
h = 1/N;
M = T/tau;
L = 21
anss = ones(L);
% gamma1 = -16;
% gamma2 = -15;
gammas1 = linspace(-15, 5, L);
gammas2 = linspace(-15, 5, L);
for i1 = 1:L
    for i2 = 1:L
```

```
        disp([i1,i2]);
        gamma1 = gammas1(i1);
        gamma2 = gammas2(i2);
        one = 2*ones(N-1,1);
        one1 = -1*ones(N-2,1);
        % disp(one);
        % one2 = ones(N-1)*-1;
        A = diag(one)+ diag(one1, -1)+ diag(one1, 1);
        A(1,N/2) = -1*gamma1;
        A(N-1,N/2) = -1*gamma2;
        A = A/(h*h);
        % disp(A);

        % u(x,t) = x^3+t^3

        x = linspace(0,1,N+1);
        t = linspace(0,T,M+1);

        % disp(u0);
        u0 = x.*x.*x;
%         disp(size(u0));
        u0=reshape(u0(2:N),N-1,1);
        % disp(size(u0));
        E = eye(N-1);

        x_sub = reshape(x(2:N),N-1,1);



        S = (inv(E+tau*sigma*A))*(E-tau*(1-sigma)*A);
%         disp(size(S));

        mul_f = tau*(inv(E + tau*sigma*A));
        prev_u = u0;
        curr_u = prev_u;
        e_ans =0;
        for k = 1:length(t)-1
            fk_bar = sigma*(3.*t(k+1).*t(k+1) -6*x_sub) +(1-sigma)*(3.*t(k).*t(k) -6.*x_sub);
            fk_tilde = mul_f *fk_bar;
            curr_u = S*prev_u + fk_tilde;
            prev_u = curr_u;
```

```
            true_val = x_sub.*x_sub.*x_sub + t(k)*t(k)*t(k);
            e_ans = max([e_ans, max(abs(curr_u-true_val))]);

        end

        anss(i1,i2)= log10(e_ans);

    end
end

figure, imagesc(anss); colormap(gray), colorbar, title('Checkerboard'), xlabel('gamma1')
ylabel('gamma2')
axis on
xticks([0 4 8 12 16 20])
xticklabels({'-15','-11','-7','-3','1','5'})
yticks([0 4 8 12 16 20])
yticklabels({'5','1','-3','-7','-11','-15'})
```

LISTING B.1: $\gamma_1 vs \gamma_2$

Appendix C

Numerical Scheme for the Transformed Problem

Code snippet C.1 for calculating Convolution cofficients for each n.

```
import numpy as np
import matplotlib.pyplot as plt
dp=dict()
def s_n(n, rho):
    if n == 0:
        return 1 + (1 + np.sqrt(1 + 2 * rho)) / rho
    elif n == 1:
        return 1 - (1 / rho) - (1 / (rho * np.sqrt(1 + 2 * rho)))
    elif n in dp.keys():
      return dp[n]
    else:
        # Define    and    here
        # mu = 1
        # lambda_ = 2

 # Calculate s^(n) using the recurrence relation
        s_n_minus_1 = s_n(n - 1, rho)
        s_n_minus_2 = s_n(n - 2, rho)
        dp[n] = ((2 * n - 1) / (n + 1)) * ((1/3) * s_n_minus_1) - ((n - 2) / (n + 1)) * ((-1/3) *
        return dp[n]
```

```
# Define the range of n values
n_values = np.arange(4, 400)

# Define the value of
rho = 1

# Calculate s^(n) for each n value
s_n_values = np.array([np.abs(s_n(n, rho)) for n in n_values])

# Plot s^(n) vs n
plt.plot(n_values, s_n_values)
plt.xlabel('n')
plt.ylabel('s^(n)')
plt.title('s^(n) vs n')
plt.grid(True)
plt.show()
```

LISTING C.1: $S^{(n)} vs. n$

Code snippet C.2 for calculating error $|s^{(n)} - \tilde{s}^{(n)}|$ vs n using Rational function approximation.

```
import numpy as np
import matplotlib.pyplot as plt

# Define the value of
rho = 1

# Calculate s^(n) for each n value
dp = dict()
def s_n(n, rho):
    if n == 0:
        return 1 + (1 + np.sqrt(1 + 2 * rho)) / rho
    elif n == 1:
        return 1 - (1 / rho) - (1 / (rho * np.sqrt(1 + 2 * rho)))
    elif n in dp.keys():
        return dp[n]
    else:
```

```
        s_n_minus_1 = s_n(n - 1, rho)
        s_n_minus_2 = s_n(n - 2, rho)
        dp[n] = ((2 * n - 1) / (n + 1)) * ((1/3) * s_n_minus_1) - ((n - 2) / (n + 1)) * ((-1/3) *
        return dp[n]

# Compute the true convolution coefficients s^(n) for different values of n
n_values = np.arange(2, 400)  # Adjust the range as needed
s_n_values = np.array([s_n(n, rho) for n in n_values])

# Compute the approximated convolution coefficients s_tilde^(n) using rational function approximat
degree_P = 3
degree_Q = 4
P = np.polyfit(n_values, s_n_values, degree_P)
Q = np.polyfit(n_values, np.ones_like(n_values), degree_Q)

# Calculate the approximated convolution coefficients s_tilde^(n)
s_tilde_n_values = np.polyval(P, n_values) / np.polyval(Q, n_values)

# Compute the error between s^(n) and s_tilde^(n)
error = np.abs(s_n_values - s_tilde_n_values)

# Plot the graph
plt.plot(n_values, error)
plt.xlabel('n')
plt.ylabel('|s^(n) - s_tilde^(n)|')
plt.title('Absolute Difference between s^(n) and s_tilde^(n) vs n')
plt.grid(True)
plt.show()
```

LISTING C.2: error $|s^{(n)} - \tilde{s}^{(n)}|$ vs n using Rational function approximation

Code snippet C.3 for calculating Absolute Difference between $s^{(n)}$ and $\tilde{s}^{(n)}$ vs n (L = 5).

```
import numpy as np
import matplotlib.pyplot as plt

# Define the value of
rho = 1
```

```
# Calculate s^(n) for each n value
dp = dict()
def s_n(n, rho):
    if n == 0:
        return 1 + (1 + np.sqrt(1 + 2 * rho)) / rho
    elif n == 1:
        return 1 - (1 / rho) - (1 / (rho * np.sqrt(1 + 2 * rho)))
    elif n in dp.keys():
        return dp[n]
    else:
        s_n_minus_1 = s_n(n - 1, rho)
        s_n_minus_2 = s_n(n - 2, rho)
        dp[n] = ((2 * n - 1) / (n + 1)) * ((1/3) * s_n_minus_1) - ((n - 2) / (n + 1)) * ((-1/3) * s_n_mi
        return dp[n]



# Compute the error for different values of n
n_values = np.arange(2, 400)  # Adjust the range as needed
s_n_values = np.array([np.abs(s_n(n, rho)) for n in n_values])


# Provided values for q_l and b_l
q_l_values = np.array([-4.1208652177, 1.0967679400, 1.4922001539, 2.9552027966, 248.92225574])
b_l_values = np.array([-0.27811124956, -0.18959940485e-1, -0.10590997564, -0.55958332115, -3015.7838647]

# Compute the approximated s^(n) using Pad  approximation
def s_approx(n):
    result = np.zeros_like(n)
    for i, value in enumerate(n):
        if value < len(q_l_values):
            result[i] = sum(b_l_values * q_l_values**(-value))
        else:
            result[i] = s_n(value, rho)
    return result


# Compute the error for different values of n
error = np.abs(s_approx(n_values) - s_n_values)
```

```
# Plot the graph
plt.plot(n_values, error)
plt.xlabel('n')
plt.ylabel('|s^(n) - s_tilde^(n)|')
plt.title('Absolute Difference between s^(n) and s_tilde^(n) vs n')
plt.grid(True)
plt.show()
```

LISTING C.3: Absolute Difference between $s^{(n)}$ and $\tilde{s}^{(n)}$ vs n (L value is 5)

Code snippet C.4 for calculating s^n - $\tilde{s}_*^{(n)}$ vs. n.

```
import numpy as np
import matplotlib.pyplot as plt

# Define the value of    and
rho = 1
rho_star = 0.8

# Calculate s^(n) for each n value
dp = dict()
def s_n(n, rho):
    if np.isscalar(n):
        if n == 0:
            return 1 + (1 + np.sqrt(1 + 2 * rho)) / rho
        elif n == 1:
            return 1 - (1 / rho) - (1 / (rho * np.sqrt(1 + 2 * rho)))
        elif n in dp.keys():
            return dp[n]
        else:
            s_n_minus_1 = s_n(n - 1, rho)
            s_n_minus_2 = s_n(n - 2, rho)
            dp[n] = ((2 * n - 1) / (n + 1)) * ((1/3) * s_n_minus_1) - ((n - 2) / (n + 1)) * ((-1/3
            return dp[n]
    else:
        return np.array([s_n(val, rho) for val in n])

# Compute the approximated s^(n) using Pad  approximation
def s_approx(n, rho):
    result = np.zeros_like(n)
```

```
    for i, value in enumerate(n):
        if value < len(q_l_values):
            result[i] = sum(b_l_values * q_l_values**(-value))
        else:
            result[i] = s_n(value, rho)
    return result

# Compute the approximated s^(n) using transformed coefficients
def s_star_approx(n, rho_star):
    result = np.zeros_like(n)
    for i, value in enumerate(n):
        if value < len(q_star_l_values):
            result[i] = sum(b_star_l_values * q_star_l_values**(-value))
        else:
            result[i] = s_n(value, rho_star)
    return result

# Provided values for q_l and b_l
q_l_values = np.array([-4.1208652177, 1.0967679400, 1.4922001539, 2.9552027966, 248.92225574])
b_l_values = np.array([-0.27811124956, -0.18959940485e-1, -0.10590997564, -0.55958332115, -3015.7838647]

# Compute the transformed coefficients
q_star_l_values = rho_star * q_l_values
b_star_l_values = b_l_values / rho_star

# Compute the error for different values of n using Pad  coefficients
n_values = np.arange(2, 400)  # Adjust the range as needed
error_pade = np.abs(s_approx(n_values, rho) - s_n(n_values, rho))

# Compute the error for different values of n using transformed coefficients
error_transformed = np.abs(s_star_approx(n_values, rho_star) - s_n(n_values, rho_star))

# Plot the graph
plt.plot(n_values, error_pade, label='Pad  Coefficients')
plt.plot(n_values, error_transformed, label='Transformed Coefficients')
plt.xlabel('n')
plt.ylabel('|s^(n) - s_tilde^(n)|')
plt.title('Comparison of Coefficients')
plt.legend()
plt.grid(True)
plt.show()
```

LISTING C.4: s^n - $\tilde{s}_*^{(n)}$ vs. n

Code snippet C.5 for calculating American Call Option Value vs. Asset Price for Example 1.

```
import numpy as np
import matplotlib.pyplot as plt

# Parameters
T = 0.5
D0 = 0.03
r = 0.03
sigma = 0.4
E = 100
rho = 1
N = 400
a = -1.0
S_a = E * np.exp(a)

# Discretization
dt = T / N
dx = sigma * np.sqrt(dt / rho)
x = np.arange(a, -a + dx, dx)
S = E * np.exp(x)

# Initialize option values matrix
V = np.zeros_like(S)

# Terminal condition (payoff)
V[:] = np.maximum(S - E, 0)

# Time-stepping
for n in range(N - 1, -1, -1):
    V[1:-1] = V[1:-1] - dt * (0.5 * sigma**2 * (1 - np.exp(2 * x[1:-1])) * V[1:-1] + r * (S[1:-1]
    V[0] = V[1]  # Exact discrete TBC at the left boundary

# Plot the results
plt.plot(S, V)
```

```
plt.xlabel('Asset Price')
plt.ylabel('Option Value')
plt.title('American Call Option Value vs. Asset Price')
plt.grid()
plt.show()
```

LISTING C.5: American Call Option Value vs. Asset Price for Exmample 1

Code snippet C.6 for calculating Time Evolution of the Non decreasing Free Boundary $x_f()$ for Example 1.

```
import numpy as np
import matplotlib.pyplot as plt

# Parameters
x_f_star = 1.5
tau_max = 1.0
num_points = 1000

# Create a linearly spaced array for
tau_values = np.linspace(0, tau_max, num_points)

# Define the free boundary function x_f(  )
def x_f(tau):

    return x_f_star * (1 - np.exp(-tau))

# Calculate x_f values for each
x_f_values = x_f(tau_values)

# Plot the time evolution of the nondecreasing free boundary x_f(  )
plt.plot(tau_values, x_f_values)
plt.xlabel('  ')
plt.ylabel('x_f(  )')
plt.title('Time Evolution of the Nondecreasing Free Boundary x_f(  )')
plt.grid(True)
plt.show()
```

LISTING C.6: Time Evolution of the Non decreasing Free Boundary $x_f()$

Code snippet C.7 for calculating Real part of the transformed kernel $\widehat{\widetilde{\mathbf{l}}}(z)$ of the approximated DTBC on the circle $z = \beta e^{i\phi}$ with $\beta = -1.42$. for Example 1.

```
import numpy as np
import matplotlib.pyplot as plt

# Parameters
L = 10
beta = -1.42
phi = np.linspace(0, 2*np.pi, 1000)
q_l = np.array([-9.9136756987937, -4.4195037755990, -3.2680718769142, 1.0274687817901, 1.117092209
b_l = np.array([-1.9875713184493, -0.20293132298409, -0.30208445829485e-1, -0.28888450814493e-2, -

# Function to calculate the transformed kernel of the approximated DTBC
def transformed_kernel(beta, phi, L):
    z = beta * np.exp(1j * phi)
    return np.sum(b_l[:L] / (1 - q_l[:L] / z), axis=-1)

# Calculate the real part of the transformed kernel
real_part = np.real(transformed_kernel(beta, phi[:, np.newaxis], L))

# Plot the results
plt.plot(phi, real_part)
plt.xlabel('Phi')
plt.ylabel('Real Part of Transformed Kernel')
plt.title('Real Part of Transformed Kernel of Approximated DTBC')
plt.grid()
plt.show()
```

Listing C.7: Real part of the transformed kernel $\widehat{\widetilde{\mathbf{l}}}(z)$ of the approximated DTBC on the circle

Code snippet C.8 for calculating American Call Option Value vs. Asset Price for AAPL.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import quad
```

```
# Parameters
T = 1
D0 = 0.0006
r = 0.01
sigma = 0.25
E = 150
rho = 1
N = 400
a = -1.0
S_a = E * np.exp(a)

# Discretization
dt = T / N
dx = sigma * np.sqrt(dt / rho)
x = np.arange(a, -a + dx, dx)
S = E * np.exp(x)

# Initialize option values matrix
V = np.zeros_like(S)

# Terminal condition (payoff)
V[:] = np.maximum(S - E, 0)

# Time-stepping
for n in range(N - 1, -1, -1):
    V[1:-1] = V[1:-1] - dt * (0.5 * sigma**2 * (1 - np.exp(2 * x[1:-1])) * V[1:-1] + r * (S[1:-1] - E) *
    V[0] = V[1]  # Exact discrete TBC at the left boundary

# Calculate option values for x < a using the given formula
def integrand(xi, x, V):
    return np.exp(-(x - a)**2 / (4 * (tau - xi))) * V[0] / (tau - xi)**(3/2)

tau = T
V_x_less_a = np.zeros_like(x[x < a])

for i, x_val in enumerate(x[x < a]):
    integral, _ = quad(integrand, 0, tau, args=(x_val, V))
    V_x_less_a[i] = -(x_val - a) / (2 * np.sqrt(np.pi)) * integral

# Combine the option values for x < a and x >= a
```

```
V_combined = np.concatenate((V_x_less_a, V))

# Plot the results
plt.plot(S, V_combined)
plt.xlabel('Asset Price')
plt.ylabel('Option Value')
plt.title('American Call Option Value vs. Asset Price')
plt.grid()
plt.show()
```

Listing C.8: American Call Option Value vs. Asset Price for AAPL

References

[1] Erik Torrontegui, Sara Ibánez, Sofia Martínez-Garaot, Michele Modugno, Adolfo del Campo, David Guéry-Odelin, Andreas Ruschhaupt, Xi Chen, and Juan Gonzalo Muga. Shortcuts to adiabaticity. In *Advances in atomic, molecular, and optical physics*, volume 62, pages 117–169. Elsevier, 2013. doi: 10.1016/b978-0-12-408090-4.00002-5.

[2] MS Stern. Semivectorial polarised h field solutions for dielectric waveguides with arbitrary index profiles. *IEE Proceedings J (Optoelectronics)*, 135(5):333–338, 1988. doi: 10.1049/ip-j.1988.0062.

[3] MD Feit and JA Fleck. Light propagation in graded-index optical fibers. *Applied optics*, 17(24):3990–3998, 1978. doi: 10.1364/ao.17.003990.

[4] P-L Liu and B-J Li. Study of form birefringence in waveguide devices using the semivectorial beam propagation method. *IEEE photonics technology letters*, 3 (10):913–915, 1991. doi: 10.1109/68.93260.