Adaptive Finite Difference Methods for Valuing American Options

by

Duy Minh Dang

A thesis submitted in conformity with the requirements
for the degree of Master of Science
Graduate Department of Computer Science
University of Toronto

Abstract

Adaptive Finite Difference Methods for

Valuing American Options

Duy Minh Dang

Master of Science

Graduate Department of Computer Science

University of Toronto

2007

We develop space-time adaptive methods for valuing American options with strong emphasis on American put options. We examine the application of adaptive techniques to the Black-Scholes partial differential equation problem associated with an American put option in the context of non-uniform second-order finite differences. At certain timesteps, we obtain a redistribution of the spatial points based on a monitor function that attempts to equidistribute the error. The proposed finite difference discretization on non-uniform grids and redistribution of the spatial points lead to linear complementarity problems with $\mathcal{M}$-matrices. The Projected Successive Over-relaxation and a penalty method are considered to handle the free boundaries. We study and compare the accuracy and efficiency of the considered methods. A complete proof of convergence and uniqueness of the projected SOR method under certain conditions is also presented.

Acknowledgements

This research was supported by the Natural Science and Engineering Research Council (NSERC) of Canada, the Ontario Graduate Scholarship (OGS) Program, and the Department of Computer Science at the University of Toronto. I would like to express my special thanks to my thesis supervisor, Professor Christina Christara, for introducing me to the field of mathematical finance, and also for her support, guidance, and encouragement during my MSc program. I also would like to thank Professor Kenneth Jackson for his valuable suggestions and remarks for improvements. Finally, I would like to thank my beloved parents and my lovely Diem for their endless support, encouragement and love.

Contents

List of Tables

List of Figures

List of Notation

Symbol	*Meaning*
S	space variable (asset price)
r	risk-free interest rate
σ	volatility of the asset price
T	final time (the expiry of the option)
t	the time variable (forward)
τ	$\tau = T - t$ (backward)
ν	index for the timestep
$\nu_{\max}$	the total number of timesteps
n	the total number of spatial subintervals
$\mathbf{t}_\nu$	value of the time variable t at the νth timestep
$\boldsymbol{\tau}_\nu$	value of the time variable $\boldsymbol{\tau}$ at the νth timestep
$\Delta\mathbf{t}_\nu$ or $\Delta\boldsymbol{\tau}_\nu$	the νth timestep size
$\mathbf{S}_i^\nu$	value of the *i*th spatial grid point at the timestep ν
$\Delta_\nu \equiv \{\mathbf{S}_i^\nu\}_{i=0}^n$	the spatial partition at timestep ν
$\{h_i^\nu\}_{i=1}^n$	the spatial stepsizes at timestep ν with $h_i^\nu = \mathbf{S}_i^\nu - \mathbf{S}_{i-1}^\nu$
V	unknown function (option price)
V^ν	the solution at timestep ν
V_i^ν	value of the solution at node $(\mathbf{S}_i^\nu, \mathbf{t}_\nu)$ or $(\mathbf{S}_i^\nu, \boldsymbol{\tau}_\nu)$
$V_{k,i}^\nu$	value of the solution at $\mathbf{S}_i^k$ of spatial partition Δ_k at time $\boldsymbol{\tau}_\nu$
$\mathbf{V}^\nu$	vector of approximate values to V at the νth timestep
$\mathbf{V}_i^\nu$	approximate value to $V(\mathbf{S}_i^\nu, \mathbf{t}_\nu)$ or $V(\mathbf{S}_i^\nu, \boldsymbol{\tau}_\nu)$
$\mathbf{V}_{\Delta_k}^\nu$	vector of approximate values to V at time $\boldsymbol{\tau}_\nu$ on space partition Δ_k
$\mathbf{V}_{k,i}^\nu$	the *i*th component of $\mathbf{V}_{\Delta_k}^\nu$ ($\mathbf{V}_{k,i}^\nu \approx V(\mathbf{S}_i^k, \boldsymbol{\tau}_\nu)$)
$\mathbf{V}^{\nu,(j)}$	*j*th estimate of $\mathbf{V}^\nu$ by an iterative method

V^*	payoff function
$\mathbf{V}^{*,\nu}$	vector of payoff values on Δ_ν
$\mathbf{V}_i^{*,\nu}$	the ith component of $\mathbf{V}^{*,\nu}$
$\hat{V}(S,t)$	monitor function
$\hat{\mathbf{V}}^\nu$	vector of approximate values to $\hat{V}(S,\boldsymbol{\tau}_\nu)$
$\hat{\mathbf{V}}_i^\nu$	the ith component of $\hat{\mathbf{V}}^\nu$ ($\hat{\mathbf{V}}_i^\nu \approx \hat{V}(\mathbf{S}_i^\nu, \boldsymbol{\tau}_\nu)$
$\xi(S,t)$	grading function
$\boldsymbol{\xi}^\nu$	vector of approximate values to $\xi(S,\boldsymbol{\tau}_\nu)$ on partition Δ_ν
$\boldsymbol{\xi}'^\nu$	vector of approximate values to $\dfrac{\partial \xi}{\partial S}(S,\tau_\nu)$ on partition Δ_ν
$\boldsymbol{\xi}_i^\nu$	the ith component of $\boldsymbol{\xi}^\nu$ ($\boldsymbol{\xi}_i^\nu \approx \xi(\mathbf{S}_i^\nu, \boldsymbol{\tau}_\nu)$)
$\tilde{r}_i^\nu$	an estimate of the error for the ith subinterval $[\mathbf{S}_{i-1}^\nu, \mathbf{S}_i^\nu]$
$\tilde{r}^\nu$	the average of all $\tilde{r}_i^\nu$'s
ω^ν	relaxation factor for the PSOR method for timestep ν

Matrices are denoted by bold upper-case letters with entries denoted by corresponding bold lower-case letters with subscripts. Vectors are denoted by bold lower-case letters without subscripts.

$\mathbf{A}$	a real matrix named $\mathbf{A}$
$\mathbf{a}_{i,j}$	the *(i, j)* entry of $\mathbf{A}$, $1 \leq i, j \leq n$ with n being the dimension of $\mathbf{A}$
$\mathbf{b}$	a real vector named $\mathbf{b}$
$\mathbf{b}_i$	the *i*th component $\mathbf{b}$, $1 \leq i \leq n$ with n being the length of $\mathbf{b}$

Frequently, given a matrix $\mathbf{A}$, we define the following matrices

$\mathbf{D}$	diagonal part of $\mathbf{A}$
$\mathbf{U}$	strictly upper triangular part of $\mathbf{A}$
$\mathbf{L}$	strictly lower triangular part of $\mathbf{A}$

Chapter 1

Introduction

The evaluation of financial option contracts is of considerable importance in finance. An option is a contract between the holder and the writer that gives the right, but not an obligation, to the holder to buy or sell a certain asset by a certain time for a given price. In particular, a call option gives the holder the right to buy, whereas a put option gives the holder the right to sell its underlying asset, or briefly the underlying, for a prescribed amount, known as strike price. An important feature of such contracts is the time when the contract holders can exercise their rights. If this occurs only at the maturity date, the option is classified as a *European* option. If holders can exercise any time up to and including the maturity date, the option is said to be an *American* option.

The option premium is the price at which the option contract is traded. The premium is paid by the potential holder (buyer of the option) to the writer of the option. In return, the writer of the option is obligated to deliver the underlying asset to the option holder if the call is exercised or buy the underlying asset if the put is exercised. In any case, the writer keeps the premium whether or not the option is exercised. It is then important to determine a fair price for an option accurately.

Generally speaking, there are two basic ways to determine the price of an option: analytical methods and numerical methods. The value of a European option is given by the solution of

the Black-Scholes partial differential equation (PDE) (see, e.g. [37]). In some cases, European options can be priced using analytical formulas. In the seminal papers by Black and Scholes ([2]), and Merton ([24]), the authors derive explicit formulas for plain European options, which are written on a single underlying asset and do not pay dividends. However, most options traded on exchanges are American. For American options, the Black-Scholes model results in a free boundary problem and unfortunately, one can not find explicit closed-form solutions to the American option pricing problem in general.

Due to the non-existence of a general closed-form solution for American options, researchers and practitioners resort to numerical methods, such as lattice methods, simulation-based methods, PDE-based methods, etc. We refer the reader to the recent paper by Broadie and Detemple [4], and references therein for a review and comparison of several numerical methods for valuing American options. Here, we would like to briefly review some popular numerical methods for American option pricing problems.

Monte-Carlo simulation and lattice methods such as binomial and trinomial trees are very popular among financial institutions. When used to value an option, Monte-Carlo methods simulate the development of the underlying asset in a risk-neutral world to determine many possible path movements. The mean of expected payoffs of each path is obtained and discounted at the risk-free rate to get an estimate of the value of the derivative. However, pricing American-style options via Monte-Carlo method still remains a very challenging problem due to the existence of the free boundary ([5], [15]). The path simulation requires a forward algorithm, whereas pricing options with early exercise features generally require backward algorithms from the maturity date (i.e. the end of the path rather than the beginning). The problem arising from using simulation in pricing American-style options results from using a forward procedure to a problem that requires a backward algorithm. As a consequence, the methods will overestimate the true value of the option ([15]). This could be overcome by using a technique called Least Squares Monte Carlo derived by Longstaff and Schwartz ([22]). The major advantage of Monte-Carlo simulation is that its convergence rate is generally independent of

the number of state variables and thus can be easily adapted to accommodate complex payoffs and complex stochastic processes, multiple underlying assets, and path-dependent contracts. However, for low dimension problems, Monte-Carlo simulation approaches suffer from low efficiency due to high simulation time.

The binomial model was first introduced by Cox, Ross and Rubinstein [8]. The underlying assumption of the model is that the price of the underlying asset follows a random walk. At each timestep until maturity, it has a certain probability of moving up and down by a certain amount. The scheme eventually yields a binomial tree. To calculate the premium, one could trace the tree backwards, starting at the maturity date where the payoff is known. During the tracing process, the price of the underlying asset at each node is calculated and compared to determine whether it is more useful to hold or to execute the option. The process stops when one reaches the root where the desired price for the option is obtained. The trinomial tree approach involves a third level of price movements in the tree and is described in [17]. One major disadvantage of lattice methods is that both binomial and trinomial trees are equivalent to explicit finite difference methods, hence suffer from a temporal stepsize restriction of the form $\Delta t \leq c\Delta S^2$, for some constant c.

Partial differential equation (PDE) based approaches are very popular for problems in low dimensions because of efficiency reasons. In addition, PDE methods allow us to obtain values for all points in the spatial domain and hence the term structure of the option, i.e. the development of the option value function for each timestep, can be easily visualized. This approach is first introduced by Brennan and Schwartz in [3]. They propose a finite difference scheme incorporated with an iterative projection method to explicitly deal with the early exercise constraint.

In this thesis, we adopt this approach for the American option pricing problem on a single asset with constant volatility and interest rate. Due to the early exercise possibility, the problem in a PDE approach can be formulated as a time dependent linear complementarity problem (LCP). One approach used by many researchers is to discretize with e.g. finite differences and

reduce the problem to a sequence of discrete LCPs, one per each timestep. This formulation will be described in more detail in Chapter 3. Below we will briefly describe two popular ways of solving the LCP: relaxation methods and the penalty methods.

One common technique for the solution of the LCPs in this category is the projected successive over-relaxation method, also known as PSOR (see, for example [30], [33], [37]). This method was first proposed by Cryer ([9]) under the assumption that the underlying matrix is symmetric positive definite. In [9], a proof of the uniqueness and convergence of the PSOR solution is given under the assumption that the underlying matrix is symmetric positive definite. A refined version of this approach is presented in [19], and is based on the observation that the solution of the problem at each timestep can be obtained as a synthesis of the two independent components corresponding to the two regions of the spatial domain separated by the free boundary.

Penalty methods have been used by several authors. In [40], Zvan, Forsyth and Vetzal introduce a penalty formulation of the discretized equations that enforces the early exercise constraint. In the same paper, a proof of the uniqueness and convergence of the penalty solution is presented under the assumption that the resulting matrices are $\mathcal{M}$-matrices. A similar approach is taken by Nielsen, Skavhaug and Tveito in [25] and they introduce a penalty term in the continuous equations.

Adaptive mesh methods are widely used in the numerical solution of PDEs (see, for example, [7], [10], [12], [13] [35], [36]). These methods compute the optimal placing of a given number of discretization points so that a chosen norm of the error in the computed approximation is minimized. The ultimate goal of adaptive mesh methods is to obtain a certain level of accuracy with a smaller number of discretization points, or a higher level of accuracy with the same number of points, when compared to uniform mesh methods.

We propose a finite difference space discretization on nonuniform grids resulting in $\mathcal{M}$-matrices The grid has a fixed number of points and the locations of the grid points are determined adaptively by means of monitor functions at selected timesteps so that the positions

of grid points are well-distributed. The first-order and the second-order partial derivatives are approximated using centered finite differences. In order to obtain an $\mathcal{M}$-matrix for each timestep, a condition on the grid step sizes is enforced by the adaptive procedure. A simple timestep selector introduced in [14] is added to improve the accuracy and efficiency of the proposed method.

We integrate the adaptive mesh methods with penalty and PSOR iterative techniques for the solution of LCP at each timestep. We introduce an improved initial guess solution vector for both the penalty and PSOR methods. We give a proof of the uniqueness and convergence of the PSOR solution under the assumption that the associated matrices are $\mathcal{M}$-matrices, not necessarily symmetric. We present numerical results that demonstrate the performance of the resulting methods. By the numerical experiments, it is shown that the adaptive placement of the spatial discretization points correctly captures the behaviour of the American option pricing problem, by concentrating many more points around the exercise value (i.e., the kink point in initial conditions) on the first timestep and around the free boundary on the subsequent timesteps. The numerical experiments also show that the adaptive mesh methods outperform the uniform ones. Moreover, the improved initial guess substantially reduces the number of PSOR iterations, while it only slightly reduces the number of penalty iterations.

The thesis is structured as follows. Chapter 2 introduces adaptive mesh methods for initial value problems (IVPs). The discretization of the American option pricing problem and two iterative methods for solving the LCPs, namely the PSOR and the penalty methods, are presented in Chapter 3. In Chapter 4, we discuss an adaptive mesh method for American option pricing problems. We study the stability and convergence of the proposed method in Chapter 5. In Chapter 6, we present selected numerical results and study the efficiency of several methods. Finally, we make some concluding remarks and discuss future work in Chapter 7. A proof of the uniqueness and convergence of the PSOR solution under certain conditions is presented in the Appendix A.

Chapter 2

Adaptive Mesh Methods

In this chapter we first introduce a spatial error estimator and then briefly describe a space adaptive algorithm for initial value problems (IVPs) based on this estimator. This algorithm will be revisited in more details in the context of American option pricing in the following chapter.

2.1 Spatial error estimation

We use the idea of grading functions introduced in [6] to construct the error estimator. The adaptive techniques then relocate the nodes to equidistribute the error in some chosen norm (or semi-norm) among the subintervals of the partition. This is called *equidistribution principle* first introduced in [10] and has been used extensively by many researchers (for example, see [7], [35], [36]). Both [6] and [10] deal with two-point boundary value problems (BVPs) and the idea has been extended to parabolic IVPs ([12], [13], [35]). Here we briefly describe the basic ideas of grading functions and the equidistribution principle in the context of parabolic IVPs.

Consider a parabolic initial value problem (IVP) described by the PDE

$$\frac{\partial V}{\partial t} - p\frac{\partial^2 V}{\partial S^2} - q\frac{\partial V}{\partial S} - sV = f, \text{ for } \quad a < S < b, \quad 0 < t \leq T \tag{2.1}$$

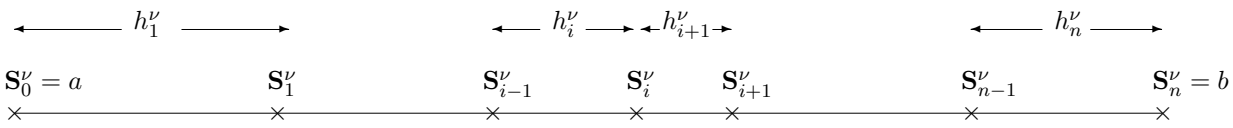


Figure 2.1: A non-uniform spatial grid at time $t = \mathbf{t}_\nu$

subject to the boundary conditions

$$V(a,t) = g_a(t), V(b,t) = g_b(t), \text{ for } \quad 0 < t \leq T \tag{2.2}$$

and the initial condition

$$V(S,0) = \gamma(S), \text{ for } \quad a \leq S \leq b \tag{2.3}$$

where the functions $p(S,t)$, $q(S,t)$, $s(S,t)$, $f(S,t)$, $g_a(t)$, $g_b(t)$, and $\gamma(t)$ are given, (a,b) is a given interval and $V(S,t)$ is an unknown function. For convenience, let $\mathcal{L}V \equiv p\dfrac{\partial^2 V}{\partial S^2} + q\dfrac{\partial V}{\partial S} + sV$, then the PDE (2.1) can be rewritten as

$$V_t = \mathcal{L}V + f. \tag{2.4}$$

We now introduce some notations that will be used in subsequent chapters of the thesis. Assume that we have a partition $\Delta_\nu \equiv \{\mathbf{S}_i^\nu\}_{i=0}^n$ at time $t = \mathbf{t}_\nu$, not necessarily uniform, with step sizes $\{h_i^\nu\}_{i=1}^n$ as shown in Figure 2.1. Let $\mathbf{V}^\nu$ denote the vector of approximate values to V at time $\mathbf{t}_\nu$ with $\mathbf{V}_i^\nu$ being an approximation to $V(\mathbf{S}_i^\nu, \mathbf{t}_\nu)$. For convenience, we denote by V^ν the solution at time $\mathbf{t}_\nu$, and let V_i^ν represent $V(\mathbf{S}_i^\nu, \mathbf{t}_\nu)$.

If the central finite difference formulas are employed for the spatial discretization, we have

$$\frac{\partial V_i^\nu}{\partial S} = a_{i1}^\nu \mathbf{V}_{i-1}^\nu + a_{i2}^\nu \mathbf{V}_i^\nu + a_{i3}^\nu \mathbf{V}_{i+1}^\nu + RM_{i1}^\nu, \tag{2.5}$$

where

$$\begin{aligned} a_{i1}^\nu &= \frac{-h_{i+1}^\nu}{h_i^\nu(h_i^\nu + h_{i+1}^\nu)}, \\ a_{i2}^\nu &= \frac{h_{i+1}^\nu - h_i^\nu}{h_i^\nu h_{i+1}^\nu}, \\ a_{i3}^\nu &= \frac{h_i^\nu}{h_{i+1}^\nu(h_i^\nu + h_{i+1}^\nu)}, \end{aligned}$$

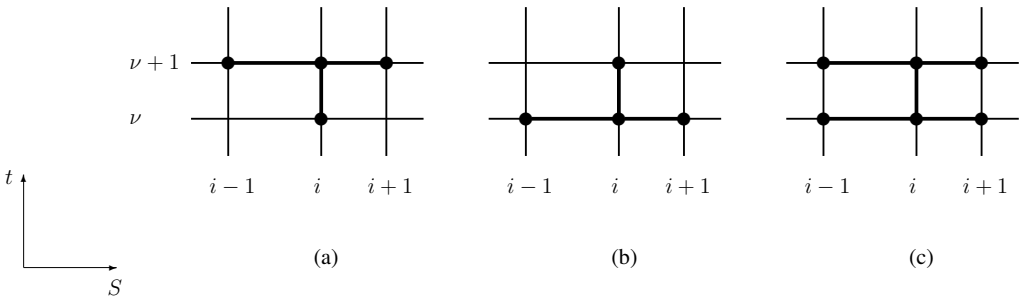


Figure 2.2: Molecules of the implicit Euler (a), explicit Euler (b), and Crank-Nicolson (c)

and

$$\frac{\partial^2 V_i^\nu}{\partial S^2} = b_{i1}^\nu \mathbf{V}_{i-1}^\nu + b_{i2}^\nu \mathbf{V}_i^\nu + b_{i3}^\nu \mathbf{V}_{i+1}^\nu + RM_{i2}^\nu, \tag{2.6}$$

where

$$\begin{aligned} b_{i1}^\nu &= \frac{2}{h_i^\nu (h_i^\nu + h_{i+1}^\nu)}, \\ b_{i2}^\nu &= \frac{-2}{h_i^\nu h_{i+1}^\nu}, \\ b_{i3}^\nu &= \frac{2}{h_{i+1}^\nu (h_i^\nu + h_{i+1}^\nu)}. \end{aligned}$$

Here, RM_{i1}^ν and RM_{i2}^ν are truncation errors whose explicit form of the first non-zero terms are $-\frac{h_{i+1}^\nu h_i^\nu}{3!}\frac{\partial^3 V_i^\nu}{\partial S^3}$ and $\frac{h_{i+1}^\nu - h_i^\nu}{3!}\frac{\partial^3 V_i^\nu}{\partial S^3}$, respectively. The timestepping is handled by the θ-scheme

$$\frac{V_i^{\nu+1} - V_i^\nu}{\Delta \mathbf{t}_\nu} = \theta \mathcal{L} V_i^{\nu+1} + (1-\theta)\mathcal{L} V_i^\nu + \theta f_i^{\nu+1} + (1-\theta) f_i^\nu, \tag{2.7}$$

where $\Delta \mathbf{t}_\nu = \mathbf{t}_{\nu+1} - \mathbf{t}_\nu$ is the time stepsize. Depending on value of θ, we have different methods as follows.

- $\theta = 1$: implicit Euler method;
- $\theta = 0$: explicit Euler method;
- $\theta = \frac{1}{2}$: the Crank-Nicolson (CN) method.

The connection scheme (stencil) of each method is illustrated in Figure (2.2).

The explicit scheme is conditionally stable with truncation error in $\mathcal{O}(\Delta \mathbf{t}_\nu)$. Both the implicit and CN schemes are unconditionally stable; however the former also has truncation error in $\mathcal{O}(\Delta \mathbf{t}_\nu)$, while the latter has truncation error in $\mathcal{O}(\Delta \mathbf{t}_\nu^2)$, which is more appealing. We focus on the CN scheme due to its second order of convergence.

The equidistribution principle attempts to find a good placement of the partition points such that some measure of the spatial error is equally distributed over the subintervals. Depending on the norm chosen, a different grading function arises, based on which the position of the partition points is computed. A grading function is of the form

$$\xi(S,t) = \frac{\int_a^S \hat{V} dS}{\int_a^b \hat{V} dS},$$

where $\hat{V}(S,t)$ is an appropriate *monitor* function. The value $\xi(\mathbf{S}_i^\nu, \mathbf{t}_\nu)$, $i = 1, \ldots, n$, of the grading function at $\mathbf{S}_i^\nu$ and $\mathbf{t}^\nu$ represents the portion of the approximate error at time $t = \mathbf{t}^\nu$ from the left endpoint of the spatial domain up to point $\mathbf{S}_i^\nu$. Appropriate quadrature rules are used to approximate the integrals. Since all monitor functions involve derivatives of high order of V, which are unknown under realistic situations, approximate values $\mathbf{V}_i^\nu$, $i = 0, \ldots, n$, are used to approximate the derivatives at time $\mathbf{t}_\nu$.

There are several choices for a monitor function. One popular choice is the arclength function ([36]), resulting in the grading function $\xi(S,t) = \dfrac{\int_a^S \sqrt{1 + (\frac{\partial V}{\partial S})^2} dS}{\int_a^b \sqrt{1 + (\frac{\partial V}{\partial S})^2} dS}$. It is suggested in [10] that, for a method with error proportional to $h^{\tilde{p}} V^{(\tilde{q})}$, where h is a stepsize and $V^{(\tilde{q})}$ is the *$\tilde{q}$-th* derivative of V with respect to S, a good grading function is

$$\xi(S,t) = \frac{\int_a^S |V^{(\tilde{q})}|^{1/\tilde{p}} dS}{\int_a^b |V^{(\tilde{q})}|^{1/\tilde{p}} dS}.$$

Ignoring higher order terms, the finite difference approximation in (2.5) is a second-order approximation and that in (2.6) is a first-order approximation. Actually, the truncation errors RM_{i1}^ν and RM_{i2}^ν can be bounded in terms of $\max(h_i^\nu)^2$ and $(h_{i+1}^\nu - h_i^\nu) + \max(h_i^\nu)^2$, respectively. Under the assumption that $h_{i+1}^\nu \approx h_i^\nu$, the truncation errors RM_{i1}^ν and RM_{i2}^ν are proportional to $\max(h_i^\nu)^2$. In this case, the (spatial) discretization error of $\mathcal{L}V$ is second order

with respect to the stepsizes, and involves $V^{(3)}$, resulting in the monitor function $\hat{V} = |V^{(3)}|^{1/2}$, and the corresponding grading function is

$$\xi(S,t) = \frac{\int_a^S |V^{(3)}|^{1/2} dS}{\int_a^b |V^{(3)}|^{1/2} dS}.$$

However, we encounter some difficulty with these grading and monitor functions. The value of the American option often oscillates and does not show convergence as grids are refined. However, if we use the monitor and grading functions $\hat{V} = |V^{(3)}|^{1/3}$ and

$$\xi(S,t) = \frac{\int_a^S |V^{(3)}|^{1/3} dS}{\int_a^b |V^{(3)}|^{1/3} dS}. \tag{2.8}$$

then the problems are resolved. These are the monitor and grading functions that we use throughout the course of experiments.

Given a grading function, the equidistribution principle requires that the partition points satisfy, for a fixed time $t = \mathbf{t}_\nu$,

$$\xi(\mathbf{S}_i^\nu, \mathbf{t}_\nu) - \xi(\mathbf{S}_{i-1}^\nu, \mathbf{t}_\nu) = \frac{\int_a^{\mathbf{S}_i^\nu} \hat{V} dS}{\int_a^b \hat{V} dS} - \frac{\int_a^{\mathbf{S}_{i-1}^\nu} \hat{V} dS}{\int_a^b \hat{V} dS} \approx \frac{1}{n},$$

or equivalently

$$\xi(\mathbf{S}_i^\nu, \mathbf{t}_\nu) = \frac{\int_a^{\mathbf{S}_i^\nu} \hat{V} dS}{\int_a^b \hat{V} dS} \approx \frac{i}{n}, \tag{2.9}$$

where $i = 1, \ldots, n-1$. Note that the two boundary points are fixed, leaving $n-1$ points to distribute.

In order to solve (2.9), one could apply the iterative scheme

$$\mathbf{S}_i^{\nu,(k+1)} = \mathbf{S}_i^{\nu,(k)} - \frac{\xi(\mathbf{S}_i^{\nu,(k)}) - \frac{i}{n}}{\xi'(\mathbf{S}_i^{\nu,(k)})}, \tag{2.10}$$

which is based on Newton's method.

2.2 Criteria for equidistribution of error

Based on the monitor function $\hat{V}(S,t)$, for a fixed time $t = \mathbf{t}_\nu$, we evaluate two quantities

$$\tilde{r}_i^\nu = \int_{\mathbf{S}_{i-1}^\nu}^{\mathbf{S}_i^\nu} \hat{V} dS, \qquad i = 1, \ldots, n, \tag{2.11}$$

$$\tilde{r}^\nu = \frac{\int_a^b \hat{V} dS}{n}, \tag{2.12}$$

using quadrature rules. It is noted that $\tilde{r}_i^\nu$ represents the estimate of the error for the *i*th subinterval $[\mathbf{S}_{i-1}^\nu, \mathbf{S}_i^\nu]$, and $\tilde{r}^\nu$ represents the average of all $\tilde{r}_i^\nu$'s. The ratio $\frac{\max_{1\le i\le n-1}\{\tilde{r}_i^\nu\}}{\tilde{r}^\nu}$ gives an indication of how well-distributed the partition is. Since we are using the equidistribution principle for the remeshing, if this ratio is too large, it follows that the maximum error estimate over the subintervals is considerably larger than the average estimate and thus the current partition is not well-distributed. At each timestep, the algorithm checks if

$$rdrift \equiv \frac{\max_{1\le i\le n-1}\{\tilde{r}_i^\nu\}}{\tilde{r}^\nu} \le \alpha, \tag{2.13}$$

where α is a small number less than 10. Typical choices for α are $\alpha = 2$ (see [35]) and $\alpha = 4$ in our experiments. That is the maximum value of $\tilde{r}_i^\nu$ must be roughly at most α times as large as the average value $\tilde{r}_i^\nu$. We consider a partition that satisfies this property to be well-distributed. Numerical experiments indicate that this criterion works reasonably well for American option pricing.

We also consider another criterion for the equidistribution of error, which is suggested in [6]. We check if

$$drift \equiv \max_{1\le i\le n-1}\{\tilde{r}_i^\nu\} - \tilde{r}^\nu \le tol, \tag{2.14}$$

where *tol* is a user chosen tolerance. In our experiments for American option pricing, the choice $tol = 10^{-1}$ works well for the set of model parameters in Table 6.1.

2.3 Algorithm Summary

In this section, we present a summary of the core code segment of a space adaptive algorithm for IVPs. This part of the code is executed when proceeding from timestep ν to timestep $\nu + 1$.

The algorithm normally works iteratively, with a stopping criterion specified in (2.13) or (2.14). In our experiments, we set the maximum number of iterations *maxit* = 1, so that at most one re-distribution of the spatial points takes place in one time step, thus the placement of the spatial points evolves as the time steps proceed. We also set the constant *smallnum* = 6, and explain below how this constant is used.

We now briefly describe the algorithm. In Line 1, we apply the standard time-stepping (usually Crank-Nicolson), using the same spatial points in steps ν and $\nu + 1$. We then calculate all quantities necessary to check the criterion (2.13) or (2.14), that decides whether a re-distribution of the points is needed (Lines 3 and 4). If the points are well-distributed, we proceed to the next time step (Lines 5 and 6). If not, the new location of the spatial points is computed using (2.9) (Lines 7 and 8). Next, we need to calculate values of the approximation at the new spatial points at the $\nu + 1$st time step. There are two ways to do this: the first, applies interpolation to values of the approximation at the current partition points at the νth time step, to compute values of the approximation at the new partition points, then applies the time-stepping procedure to compute values of the approximation at the new partition points, at the $\nu + 1$st time step; the second, simply applies interpolation to values of the approximation at the current partition points at the $\nu + 1$st time step, to compute values of the approximation at the new partition points, at the $\nu + 1$st time step. The first technique is used in the first few (*smallnum*) time steps (Lines 9, 10 and 11), while the second is used for all other time steps (Lines 12 and 13).

The choice of interpolation technique may be important under certain circumstances. In the case of European option pricing or other problems without special constraints, one can use standard interpolation techniques, such as cubic spline interpolation, to obtain the interpolated values. However, we noticed that the interpolation techniques used in valuing American op-

tions should be chosen carefully, due to the existence of the free boundary at each time step. More discussion on this will be provided in Chapter 4.

We now give the space adaptive algorithm for IVPs, which assumes that an approximation to V^ν is already computed on partition Δ_ν, and computes an approximation to $V^{\nu+1}$ on partition $\Delta_{\nu+1}$, which may be different from Δ_ν.

Algorithm 1: Brief description of space adaptive algorithm for IVPs

apply (2.7) to compute approximation to $V^{\nu+1}$ on partition $\Delta_{\nu+1} \equiv \Delta_\nu$ using the given approximation to V^ν;
for $k = 1$ to *maxit* **do**
  approximate the appropriate derivatives of $V^{\nu+1}$ and estimate the distribution of the error using (2.8);
  calculate the quantities $(\tilde{r}_i^{\nu+1})_{i=1}^{n-1}$ and $\tilde{r}^{\nu+1}$ using (2.11) and (2.12);
  check criterion (2.13) or (2.14) to decide whether a remeshing is needed;
  if (2.13) or (2.14) is satisfied **then**
    exit;
  else
    compute a new partition $\Delta_{\nu+1} \neq \Delta_\nu$ using (2.9);
    if $\nu \leq$ *smallnum* **then**
      compute new approximation to V^ν on $\Delta_{\nu+1}$ using interpolation on current approximation to V_ν;
      apply (2.7) to compute new approximation to $V^{\nu+1}$ on partition $\Delta_{\nu+1}$ using the new approximation to V^ν;
    else
      compute new approximation to $V^{\nu+1}$ on $\Delta_{\nu+1}$ using interpolation on current approximation to $V^{\nu+1}$;
    end if
  end if
end for

Chapter 3

Pricing American Options

In this chapter, we review the problem of pricing American options. We first introduce the pricing model and the formation of the American option valuation as a linear complementarity problem. We then discuss the numerical computation of American options. In particular, we present the discretization with finite differences and two iterative methods for solving the associated constrained matrix problem. Although we restrict our attention to American puts, the approach can be easily applied to American calls on dividend-paying assets.

3.1 Pricing Model

The model introduced by Black and Scholes [2] and Merton [24] was the first, and is still the most widely used, for pricing options. They observed a lognormal behavior of asset prices and derived the following partial differential equation (PDE) that describes the option's value:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0. \tag{3.1}$$

Here, $V = V(S,t)$ represents the option price; $S \geq 0$ is the underlying asset price; r is the risk-free interest rate; σ is the volatility of the underlying asset price; and $0 \leq t \leq T$ where T the expiry. In this thesis, we consider only constant volatility and interest rate. However, the approach presented here can be extended to cases where volatility and interest rate are functions

of the underlying asset price and time. Equation (3.1) is referred to as the *Black-Scholes PDE*. It is a parabolic PDE and has many solutions. To obtain a unique solution for a particular pricing model, the Black-Scholes equation must be associated with additional constraints such as final conditions, boundary conditions, or free boundary conditions. It can then can be solved backwards in time from the option expiry time $t = T$ to the present $t = 0$.

In contrast to European options which can only be exercised at the maturity date T, American options can be exercised at any time up to T. Consequently, identifying the optimal exercise strategy is an essential part of the valuation problem. For American put options, the possibility of early exercise requires that

$$V(S,t) \geq \max(E - S, 0), \quad \forall t \in [0,T], \tag{3.2}$$

otherwise an arbitrage opportunity would arise ([17], [30]).

The evaluation of an American option is associated with a free boundary value problem ([37]), and the exercise boundary curve $S_f(t)$, which varies with time, divides the *S-t* half strip $[0,\infty) \times [0,T]$ into the continuation region and the stopping region. The continuation region $\{(S,t) \in [0,\infty) \times [0,T] : V(S,t) > \max(E - S, 0)\}$ is the set of all points (S,t) where the option should be held, whereas in the stopping region $\{(S,t) \in [0,\infty) \times [0,T] : V(S,t) = \max(E - S, 0)\}$ early exercise is advisable. If $S_f(t)$ is calculated, the American put option holder should exercise the option as early as possible when $S < S_f(t)$ and hold the option otherwise.

Under the Black-Scholes framework, the price $V(S,t)$ of an American put option satisfies either

$$\left\{ \begin{array}{r} \dfrac{\partial V}{\partial t} + \dfrac{1}{2}\sigma^2 S^2 \dfrac{\partial^2 V}{\partial S^2} + rS\dfrac{\partial V}{\partial S} - rV = 0 \\ V - \max(E - S, 0) \geq 0 \end{array} \right\} \tag{3.3}$$

in the continuation region, or

$$\left\{ \begin{array}{r} \dfrac{\partial V}{\partial t} + \dfrac{1}{2}\sigma^2 S^2 \dfrac{\partial^2 V}{\partial S^2} + rS\dfrac{\partial V}{\partial S} - rV < 0 \\ V - \max(E - S, 0) = 0 \end{array} \right\} \tag{3.4}$$

in the stopping region. We also have additional boundary conditions at the free boundary $S_f(t)$

$$\begin{aligned} V(S_f(t),t) &= \max(E - S_f(t), 0) = E - S_f(t), \\ \frac{\partial V}{\partial S}(S_f(t),t) &= -1, \end{aligned} \tag{3.5}$$

which are known as smooth boundary conditions ([30]). The final condition is

$$V(S,T) = \max(E - S, 0). \tag{3.6}$$

The boundary conditions can be obtained by imposing Dirichlet boundary conditions which are given by

$$\begin{aligned} &V(0,t) = E, \\ &V(S,t) \sim 0 \quad \text{as} \quad S \to \infty. \end{aligned} \tag{3.7}$$

These conditions are based on some additional knowledge about asymptotic behavior of the solutions, which may not be available in complex contracts. For simplicity, in this thesis we consider only Dirichlet boundary conditions for American options.

Define the operator

$$\mathcal{L}V \equiv \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV \tag{3.8}$$

and let $V^* = \max(E - S, 0)$ denote the payoff function. The American put option pricing can be reformulated as the linear complementarity problem (LCP)

$$\left\{ \begin{array}{c} \dfrac{\partial V}{\partial t} + \mathcal{L}V = 0 \\ V - V^* \geq 0 \end{array} \right\} \text{ or } \left\{ \begin{array}{c} \dfrac{\partial V}{\partial t} + \mathcal{L}V < 0 \\ V - V^* = 0 \end{array} \right\}. \tag{3.9}$$

together with the final condition (3.6), boundary conditions (3.7), and smooth boundary conditions (3.5). The optimal free boundary $S_f(t)$ is automatically captured by this formulation and can be determined a-posteriori. Solutions of linear complementarity problems can be obtained by several iterative methods such as the projected successive over-relaxation method or the penalty methods. We will discuss these two methods in a later section.

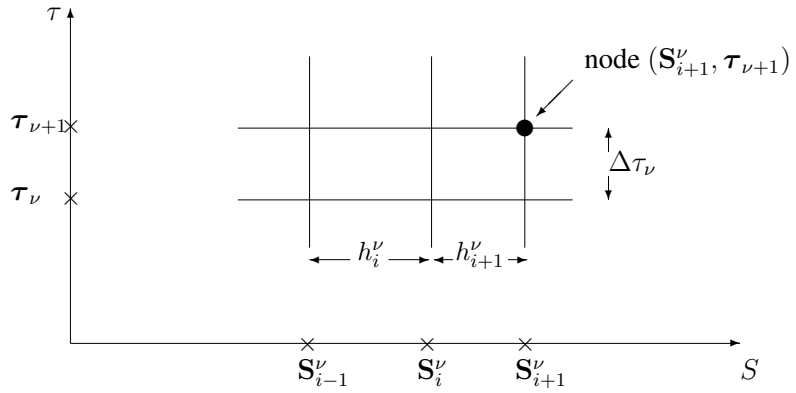


Figure 3.1: Details and notations of the finite difference grid

3.2 Discretization with Finite Differences

In order to write the LCP (3.9) in more conventional form so that we can solve it backwards in time, define $\tau = T - t$, so that it becomes

$$\left\{ \begin{array}{r} \dfrac{\partial V}{\partial \tau} - \mathcal{L}V = 0 \\ V - V^* \geq 0 \end{array} \right\} \text{ or } \left\{ \begin{array}{r} \dfrac{\partial V}{\partial \tau} - \mathcal{L}V > 0 \\ V - V^* = 0 \end{array} \right\}. \tag{3.10}$$

For the discretization of the time variable, we choose a set of grid points forward in time with respect to τ

$$\{\boldsymbol{\tau}_0, \boldsymbol{\tau}_1, \ldots, \boldsymbol{\tau}_{\nu_{\max}-1}, \boldsymbol{\tau}_{\nu_{\max}}\} \qquad \boldsymbol{\tau}_0 = 0 < \boldsymbol{\tau}_1 < \cdots < \boldsymbol{\tau}_{\nu_{\max}-1} < \boldsymbol{\tau}_{\nu_{\max}} = T \tag{3.11}$$

Define $\Delta\boldsymbol{\tau}_{\nu} = \boldsymbol{\tau}_{\nu+1} - \boldsymbol{\tau}_{\nu}$, $\nu = 0, 1, \ldots, \nu_{\max} - 1$. Usually, we use uniform time stepsize but we may make use of a non-uniform time stepsizes to speed up efficiency. The choice of spatial discretization is more complicated. The infinite domain $[0, \infty)$ must be truncated down to $[0, \mathbf{S}_{\max}]$ and the boundary condition at $S = \infty$ is replaced by the boundary condition at $S = \mathbf{S}_{\max}$. More discussion on how to choose value $\mathbf{S}_{\max}$ can be found in Chapter 6. The spatial partition on the truncated domain at time level $\boldsymbol{\tau}_{\nu}$ is denoted by Δ_{ν}. The Figure 3.1 illustrates part of the entire (S, τ) grid using the same notations introduced in previous chapter.

3.2.1 Discretization for $\mathcal{L}V$

We use non-uniform central finite difference formulas defined in (2.5) and (2.6) for the spatial discretization. Assume that we want to proceed from time step ν to time step $\nu+1$. Recall that $\mathcal{L}V \equiv \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV$. Substituting (2.5) and (2.6) into $\mathcal{L}V_i^\nu$, $i = 1, \ldots, n-1$ and ignoring the truncation errors, we obtain

$$\begin{aligned}\mathcal{L}V_i^\nu &= \frac{1}{2}\sigma^2(\mathbf{S}_i^\nu)^2(b_{i1}^\nu \mathbf{V}_{i-1}^\nu + b_{i2}^\nu \mathbf{V}_i^\nu + b_{i3}^\nu \mathbf{V}_{i+1}^\nu) + r\mathbf{S}_i^\nu(a_{i1}^\nu \mathbf{V}_{i-1}^\nu + a_{i2}^\nu \mathbf{V}_i^\nu + a_{i3}^\nu \mathbf{V}_{i+1}^\nu) - r\mathbf{V}_i^\nu \\ &= \big(\frac{1}{2}\sigma^2(\mathbf{S}_i^\nu)^2 b_{i1}^\nu + r\mathbf{S}_i^\nu a_{i1}^\nu\big)\mathbf{V}_{i-1}^\nu + \big(\frac{1}{2}\sigma^2(\mathbf{S}_i^\nu)^2 b_{i2}^\nu + r\mathbf{S}_i^\nu a_{i2}^\nu - r\big)\mathbf{V}_i^\nu \\ &\qquad + \big(\frac{1}{2}\sigma^2(\mathbf{S}_i^\nu)^2 b_{i3}^\nu + r\mathbf{S}_i^\nu a_{i3}^\nu\big)\mathbf{V}_{i+1}^\nu,\end{aligned}$$

where $a_{i1}^\nu, a_{i2}^\nu, a_{i3}^\nu$ and $b_{i1}^\nu, b_{i2}^\nu, b_{i3}^\nu$ are specified in (2.5) and (2.6), respectively. Let

$$\mathbf{m}_{i,i-1}^\nu = -\frac{1}{2}\sigma^2(\mathbf{S}_i^\nu)^2 b_{i1}^\nu - r\mathbf{S}_i^\nu a_{i1}^\nu, \tag{3.12}$$

$$\mathbf{m}_{i,i}^\nu = -\frac{1}{2}\sigma^2(\mathbf{S}_i^\nu)^2 b_{i2}^\nu - r\mathbf{S}_i^\nu a_{i2}^\nu + r, \tag{3.13}$$

$$\mathbf{m}_{i,i+1}^\nu = -\frac{1}{2}\sigma^2(\mathbf{S}_i^\nu)^2 b_{i3}^\nu - r\mathbf{S}_i^\nu a_{i3}^\nu, \tag{3.14}$$

then

$$\mathcal{L}V_i^\nu = -\mathbf{m}_{i,i-1}^\nu \mathbf{V}_{i-1}^\nu - \mathbf{m}_{i,i}^\nu \mathbf{V}_i^\nu - \mathbf{m}_{i,i+1}^\nu \mathbf{V}_{i+1}^\nu, \tag{3.15}$$

$$i = 1, 2, \ldots, n-1. \tag{3.16}$$

3.2.2 Time Discretization

Recall that under the change of variable $\tau = T - t$, the Black-Scholes PDE used for American options becomes

$$\frac{\partial V}{\partial \tau} - \mathcal{L}V = 0,$$

or equivalently

$$\frac{\partial V}{\partial \tau} = \mathcal{L}V, \tag{3.17}$$

where $\mathcal{L}V \equiv \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV$. Applying the θ-timestepping scheme (2.7) to (3.17), we obtain

$$\frac{V_i^{\nu+1} - V_i^{\nu}}{\Delta\boldsymbol{\tau}_\nu} = \theta\mathcal{L}V_i^{\nu+1} + (1-\theta)\mathcal{L}V_i^{\nu}$$

$\Longleftrightarrow$

$$V_i^{\nu+1} - \theta\Delta\boldsymbol{\tau}_\nu\mathcal{L}V_i^{\nu+1} = V_i^{\nu} + (1-\theta)\Delta\boldsymbol{\tau}_\nu\mathcal{L}V_i^{\nu}. \tag{3.18}$$

Substituting the discretization formula (3.15) for $\mathcal{L}V$ into (3.18) gives

$$\begin{aligned}\mathbf{V}_i^{\nu+1} + \theta\Delta\boldsymbol{\tau}_\nu(\mathbf{m}_{i,i-1}^{\nu+1}\mathbf{V}_{i-1}^{\nu+1} + \mathbf{m}_{i,i}^{\nu+1}\mathbf{V}_i^{\nu+1} + \mathbf{m}_{i,i+1}^{\nu+1}\mathbf{V}_{i+1}^{\nu+1}) \\ = \mathbf{V}_i^{\nu} - (1-\theta)\Delta\boldsymbol{\tau}_\nu(\mathbf{m}_{i,i-1}^{\nu}\mathbf{V}_{i-1}^{\nu} + \mathbf{m}_{i,i}^{\nu}\mathbf{V}_i^{\nu} + \mathbf{m}_{i,i+1}^{\nu}\mathbf{V}_{i+1}^{\nu}).\end{aligned} \tag{3.19}$$

Note that in equation (3.19), we have $\mathbf{m}_{i,i-1}^{\nu} = \mathbf{m}_{i,i-1}^{\nu+1}, \mathbf{m}_{i,i}^{\nu} = \mathbf{m}_{i,i}^{\nu+1}$ and $\mathbf{m}_{i,i+1}^{\nu} = \mathbf{m}_{i,i+1}^{\nu+1}$. For simplicity, we use $\mathbf{m}_{i,i-1}^{\nu}, \mathbf{m}_{i,i}^{\nu}$ and $\mathbf{m}_{i,i+1}^{\nu}$ on both sides of equation (3.19), resulting in the following one:

$$\begin{aligned}\mathbf{V}_i^{\nu+1} + \theta\Delta\boldsymbol{\tau}_\nu(\mathbf{m}_{i,i-1}^{\nu}\mathbf{V}_{i-1}^{\nu+1} + \mathbf{m}_{i,i}^{\nu}\mathbf{V}_i^{\nu+1} + \mathbf{m}_{i,i+1}^{\nu}\mathbf{V}_{i+1}^{\nu+1}) \\ = \mathbf{V}_i^{\nu} - (1-\theta)\Delta\boldsymbol{\tau}_\nu(\mathbf{m}_{i,i-1}^{\nu}\mathbf{V}_{i-1}^{\nu} + \mathbf{m}_{i,i}^{\nu}\mathbf{V}_i^{\nu} + \mathbf{m}_{i,i+1}^{\nu}\mathbf{V}_{i+1}^{\nu}).\end{aligned} \tag{3.20}$$

Writing this in matrix form, we have

$$(\mathbf{I} + \theta\Delta\boldsymbol{\tau}_\nu\mathbf{M}^{\nu})\mathbf{V}^{\nu+1} = (\mathbf{I} - (1-\theta)\Delta\boldsymbol{\tau}_\nu\mathbf{M}^{\nu})\mathbf{V}^{\nu}, \tag{3.21}$$

where $\mathbf{I}$ is the identity matrix and $\mathbf{M}^{\nu}$ is a tridiagonal matrix in $\mathbb{R}^{(n-1)\times(n-1)}$. Matrix $\mathbf{M}^{\nu}$ has the following form:

$$\mathbf{M}^{\nu} = \begin{pmatrix} \mathbf{m}_{1,1}^{\nu} & \mathbf{m}_{1,2}^{\nu} & 0 & \dots & 0 \\ \mathbf{m}_{2,1}^{\nu} & \mathbf{m}_{2,2}^{\nu} & \mathbf{m}_{2,3}^{\nu} & \dots & 0 \\ \ddots & \ddots & \ddots & \ddots & \ddots \\ \ddots & \mathbf{m}_{i,i-1}^{\nu} & \mathbf{m}_{i,i}^{\nu} & \mathbf{m}_{i,i+1}^{\nu} & \ddots \\ \ddots & \ddots & \ddots & \ddots & \ddots \\ 0 & \dots & \mathbf{m}_{n-2,n-3}^{\nu} & \mathbf{m}_{n-2,n-2}^{\nu} & \mathbf{m}_{n-2,n-1}^{\nu} \\ 0 & \dots & 0 & \mathbf{m}_{n-1,n-2}^{\nu} & \mathbf{m}_{n-1,n-1}^{\nu} \end{pmatrix},$$

with $\mathbf{m}^{\nu}_{i,i-1}, \mathbf{m}^{\nu}_{i,i}, \mathbf{m}^{\nu}_{i,i+1}$, $i = 1, \ldots, n-1$ being defined in (3.12). Note that besides constants σ and r, matrix $\mathbf{M}^{\nu}$ depends also on spatial partition at the time level $\boldsymbol{\tau}_{\nu}$.

As we mentioned earlier, we would like to use the CN method due to its second order of convergence. However, for CN method, spurious oscillations can be introduced into the solution. Even though these oscillations may be small if we look at the option value, they can be magnified when computing the option delta and gamma [42]. In this thesis, we focus on the CN scheme, but we use the implicit Euler scheme for the Rannacher smoothing technique. We will discuss the oscillatory behavior of the CN method and its remedy in Chapter 5.

Under the above discretization, the LCP (3.10) is re-formulated as a constrained matrix problem of the form

$$\left\{ \begin{array}{r} \mathbf{A}^{\nu}\mathbf{V}^{\nu+1} = \mathbf{b}^{\nu} \\ \mathbf{V}^{\nu+1} - \mathbf{V}^{*,\nu+1} \geq \mathbf{0} \end{array} \right\} \text{ or } \left\{ \begin{array}{r} \mathbf{A}^{\nu}\mathbf{V}^{\nu+1} > \mathbf{b}^{\nu} \\ \mathbf{V}^{\nu+1} - \mathbf{V}^{*,\nu+1} = \mathbf{0} \end{array} \right\}, \tag{3.22}$$

which must be solved in order to proceed from time step ν to time step $\nu + 1$. Here,

$$\mathbf{A}^{\nu} = \mathbf{I} + \theta\Delta\boldsymbol{\tau}_{\nu}\mathbf{M}^{\nu}, \qquad \mathbf{b}^{\nu} = (\mathbf{I} - (1-\theta)\Delta\boldsymbol{\tau}_{\nu}\mathbf{M}^{\nu})\mathbf{V}^{\nu}, \tag{3.23}$$

are both dependent on the space partition Δ_{ν} and time step size $\Delta\boldsymbol{\tau}_{\nu}$. We denote $\mathbf{V}^{*,\nu}$ as a vector of payoff values on spatial partition Δ_{ν}, where the ith component is

$$\mathbf{V}^{*,\nu}_{i} = \max(E - \mathbf{S}^{\nu}_{i}, 0).$$

In the next section, we will discuss some iterative methods for solving this type of problem.

3.3 Iterative Methods

Before we describe the application of the projected successive over-relaxation and the penalty method on pricing American options, we first review some background on iterative methods. We are interested in solving the linear system of equations

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \quad \mathbf{A} \in \mathbb{R}^{n\times n}, \mathbf{b}, \mathbf{x} \in \mathbb{R}^{n}. \tag{3.24}$$

With a suitable matrix $\mathbf{Q} \in \mathbb{R}^{n\times n}$, we can re-write (3.24) as

$$\mathbf{Q}\mathbf{x} = (\mathbf{Q} - \mathbf{A})\mathbf{x} + \mathbf{b}$$

$\Longleftrightarrow$

$$\mathbf{x} = \underbrace{(\mathbf{I} - \mathbf{Q}^{-1}\mathbf{A})}_{\mathbf{G}}\,\mathbf{x} + \underbrace{\mathbf{Q}^{-1}\mathbf{b}}_{\mathbf{c}},$$

resulting in the iteration scheme

$$\mathbf{x}^{(k+1)} = \mathbf{G}\mathbf{x}^{(k)} + \mathbf{c}, \tag{3.25}$$

where k is the iteration index and $\mathbf{G}$ is the *iteration matrix*. It has been proved in [34] that iteration scheme (3.25) converges if and only if $\rho(\mathbf{G}) < 1$ where $\rho(\mathbf{G})$ denotes the spectral radius of $\mathbf{G}$. Moreover, to optimize the convergence, we would like to have $\rho(\mathbf{G})$ to be as small as possible. One way is to split the matrix $\mathbf{A}$ into three parts $\mathbf{A} = \mathbf{D} + \mathbf{L} + \mathbf{U}$, where $\mathbf{D}$, $\mathbf{L}$ and $\mathbf{U}$ represent the diagonal, strictly lower triangular and strictly upper triangular parts of $\mathbf{A}$, respectively. We now introduce the *relaxation methods*.

Let $\omega > 0$ denote the *relaxation parameter*. We let $\mathbf{Q} = \frac{\mathbf{D}}{\omega} + \mathbf{L}$, which results in the iteration matrix

$$\begin{aligned}\mathbf{G} &= \mathbf{I} - \mathbf{Q}^{-1}\mathbf{A} \\ &= \mathbf{I} - (\frac{\mathbf{D}}{\omega} + \mathbf{L})^{-1}\mathbf{A}.\end{aligned}$$

This leads to the iteration scheme

$$\mathbf{x}^{(k+1)} = (\mathbf{I} - (\frac{\mathbf{D}}{\omega} + \mathbf{L})^{-1}\mathbf{A})\mathbf{x}^{(k)} + (\frac{\mathbf{D}}{\omega} + \mathbf{L})^{-1}\mathbf{b}. \tag{3.26}$$

Substituting $\mathbf{A} = \mathbf{D} + \mathbf{L} + \mathbf{U}$ into (3.26) and rearranging the resulting formula, we obtain the relaxation scheme

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \omega\mathbf{D}^{-1}(\mathbf{b} - \mathbf{L}\mathbf{x}^{(k+1)} - (\mathbf{D} + \mathbf{U})\mathbf{x}^{(k)}). \tag{3.27}$$

Note that in formula (3.27), when $\mathbf{x}_i^{(k+1)}$ is being computed, all previous components $\mathbf{x}_j^{(k+1)}$'s, $j = 1, \ldots, i-1$, are already computed and thus they can be used to compute $\mathbf{x}_i^{(k+1)}$ to speed

up the convergence. In component form, (3.27) can be written as

$$\begin{cases} \textbf{for } i = 1, \ldots, n \textbf{ do} \\ \quad \mathbf{y}_i^{(k+1)} = \dfrac{1}{\mathbf{a}_{i,i}} \Big(\mathbf{b}_i - \displaystyle\sum_{j<i} \mathbf{a}_{i,j} \mathbf{x}_j^{(k+1)} - \sum_{j>i} \mathbf{a}_{i,j} \mathbf{x}_j^{(k)} \Big) \\ \quad \mathbf{x}_i^{(k+1)} = \mathbf{x}_i^{(k)} + \omega (\mathbf{y}_i^{(k+1)} - \mathbf{x}_i^{(k)}) \\ \textbf{endfor} \end{cases} \tag{3.28}$$

Letting $\omega = 1$, this results in the Gauss-Seidel method. For $0 < \omega < 1$, the scheme is called *successive under-relaxation*, and for $1 < \omega < 2$, we obtain the *successive over-relaxation* (SOR) method.

3.3.1 Projected SOR Method

The projected SOR (PSOR) method for linear complementarity problems was first proposed by Cryer in [9]. The method is well-known and widely applied for pricing American options (see [33], [30]). In this section, we first discuss the PSOR method and then we present an algorithm which performs PSOR iteration for solving the constrained matrix problem resulting from American option pricing.

Let us now consider a generic problem of the form

$$\left\{ \begin{array}{l} \mathbf{A}\mathbf{x} = \mathbf{b} \\ \mathbf{x} \geq \mathbf{g} \end{array} \right\} \text{ or } \left\{ \begin{array}{l} \mathbf{A}\mathbf{x} > \mathbf{b} \\ \mathbf{x} = \mathbf{g} \end{array} \right\}. \tag{3.29}$$

For the solution of this problem, Cryer [9] proposes the PSOR based on modifications of the iterative scheme of the SOR method by including at each iteration the constraint $\mathbf{x} \geq \mathbf{g}$. More specifically, the PSOR method for solving problem (3.29) is

$$\begin{cases} \textbf{for } i = 1, \ldots, n \textbf{ do} \\ \quad \mathbf{y}_i^{(k+1)} = \dfrac{1}{\mathbf{a}_{i,i}} \Big(\mathbf{b}_i - \displaystyle\sum_{j<i} \mathbf{a}_{i,j} \mathbf{x}_j^{(k+1)} - \sum_{j>i} \mathbf{a}_{i,j} \mathbf{x}_j^{(k)} \Big) \\ \quad \mathbf{x}_i^{(k+1)} = \max \left(\mathbf{g}_i, \mathbf{x}_i^{(k)} + \omega (\mathbf{y}_i^{(k+1)} - \mathbf{x}_i^{(k)}) \right) \\ \textbf{endfor} \end{cases} \tag{3.30}$$

The proof of convergence for the PSOR method under the condition that matrix $\mathbf{A}$ is symmetric positive definite can be found in [9]. In our case, due to adaptivity, the underlying matrix at each timestep in our case is highly non-symmetric and thus the convergence condition in [9] does not apply. In Appendix A, we present a proof to show that if the matrix $\mathbf{A}$ belongs to a specific class of matrices and the relaxation parameter ω satisfies certain conditions, then the convergence of the PSOR method is guaranteed.

Both SOR and PSOR methods start with some initial guess $\mathbf{x}^{(0)}$ and compute successive approximations $\mathbf{x}^{(k)}$ to the solution for $k = 1, \ldots$ until some stopping criterion is satisfied. The stopping criterion we use is

$$\| \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)} \| \leq tol,$$

where $\| \ . \ \|$ could be 2-norm or ∞-norm and *tol* is a user-defined tolerance.

The relaxation parameter ω plays an important role on the convergence rate of the (P)SOR method. The optimal relaxation parameter can be determined when $\mathbf{A}$ is a positive definite matrix (see [39]). However, the matrix at each time step is not symmetric due to adaptivity, it is not easy to find the optimal ω in our case. For the numerical solutions of IVPs, a technique to approximate dynamically the optimal relaxation parameter for (P)SOR method at each time step is proposed in [37]. To explain the technique described in [37], consider three successive timesteps $\nu - 1$, ν , $\nu + 1$. Let ω^{ν} and it^{ν} denote the value of the relaxation parameter and the number of iterations required for convergence at timestep ν, respectively. First, we compare the number of iterations the method required in timesteps $\nu - 1$ and ν, which use $\omega^{\nu-1}$ and ω^{ν}, respectively. Here, $\omega^{\nu} = \omega^{\nu-1} + \Delta\omega$ with $\Delta\omega$ being a very small, constant number either greater or smaller than zero. Typically, the value of $\Delta\omega$ in absolute value is between 0.01 and 0.05. Depending on the result of that comparison we decide the value of ω^{ν} to be used in the next timestep by adjusting $\Delta\omega$ as follows.

(a) If $it^{\nu-1} < it^{\nu}$, set $\Delta\omega = -\Delta\omega$;

(b) Set $\omega^{\nu+1} = \omega^{\nu} + \Delta\omega$;

We adopt this technique in our experiments.

The initial guess also plays a role in the convergence of the (P)SOR method. The most common initial guess is $\mathbf{x}^{(0)} = \mathbf{0}$. In the context of IVPs, a better initial guess for the current time step is the approximate solution for the previous time step. However, using extrapolation on the approximations at the previous time steps could produce even a better initial guess. More discussion on this will be provided next.

We now give an implementation of the PSOR method for solving the LCP resulting from American options. Recall that the LCP is described by (3.9) and the discretization of the partial differential operator gives rise to the constrained matrix problem (3.22). We notice the similarities in form between the constrained matrix problem (3.22) and problem (3.29) and thus the PSOR iterative scheme can be used to solve the non-linear problem associated with the American option pricing. For each iteration and each component of the solution vector, the valuation involves a comparison between the value of the option that would be obtained if the holder does not exercise and the value of the option that would be obtained if the holder does exercise the option. Since it is assumed that the holder would act optimally, the larger of these two values would be chosen as the value of the option at that point. We introduce some additional notations used for the algorithm and for the rest of the thesis.

$\mathbf{V}^{\nu+1,(k)}$: kth estimate of $\mathbf{V}^{\nu+1}$;

$\mathbf{a}^{\nu}_{i,j}$: (i,j)th entry of matrix $\mathbf{A}^{\nu}$;

$\mathbf{b}^{\nu}_{i}$: ith entry of vector $\mathbf{b}^{\nu}$;

A brief description of PSOR iteration for valuing an American put is presented in Algorithm 2.

It has been noted in the literature that the convergence rate of the PSOR method deteriorates when the discretization is refined which makes PSOR a very slow method on finer grids. The convergence rate of the PSOR method depends on the initial guess and relaxation factor ω. As we mentioned earlier, we follow a technique in [37] to dynamically determine ω at each time step. With respect to the initial guess, one popular choice for the initial guess is

$$\mathbf{V}^{\nu+1,(0)} = \mathbf{V}^{\nu}, \tag{3.31}$$

which is an approximate solution at the previous time step $\boldsymbol{\tau}_\nu$. Our experimental results show that this initial guess still results in deteriorating behaviors of the convergence rate. However, deterioration of the convergence rate could be reduced if we had a better initial guess. In particular, we use linear extrapolation on $\mathbf{V}^\nu$ and $\mathbf{V}^{\nu-1}$, resulting in the initial guess

$$\mathbf{V}^{\nu+1,(0)} = \frac{(\Delta\boldsymbol{\tau}_\nu + \Delta\boldsymbol{\tau}_{\nu-1})}{\Delta\boldsymbol{\tau}_{\nu-1}}\mathbf{V}^\nu - \frac{\Delta\boldsymbol{\tau}_\nu}{\Delta\boldsymbol{\tau}_{\nu-1}}\mathbf{V}^{\nu-1}, \quad \nu = 2, \ldots, \nu_{\max}. \tag{3.32}$$

In case of constant timesteps, that is $\Delta\boldsymbol{\tau}_\nu = \Delta\boldsymbol{\tau}_{\nu-1}$, $\nu = 2, \ldots, \nu_{\max}$, the above formula reduces to

$$\mathbf{V}^{\nu+1,(0)} = 2\mathbf{V}^\nu - \mathbf{V}^{\nu-1}, \quad \nu = 2, \ldots, \nu_{\max}.$$

Experiment results show that for option pricing, this choice of initial guess produce much faster convergence and the deterioration of convergence rate on finer grids is less serious than those obtained from the choice (3.31). We provide numerical results related to this issue in Chapter 6.

Algorithm 2: Brief description of PSOR American put option constraint iteration

initialize $\mathbf{V}^{\nu+1,(0)}$;
for $k = 0, \ldots,$ until convergence **do**
  for $i = 1, \ldots, n-1$ **do**
    calculate $\widetilde{\mathbf{V}}_i^{\nu+1,(k+1)} = \frac{1}{\mathbf{a}_{i,i}^\nu}\Big(\mathbf{b}_i^\nu - \sum_{j<i}\mathbf{a}_{i,j}^\nu\mathbf{V}_j^{\nu+1,(k+1)} - \sum_{j>i}\mathbf{a}_{i,j}^\nu\mathbf{V}_j^{\nu+1,(k)}\Big)$;
    $\mathbf{V}_i^{\nu+1,(k+1)} = \max\big(\mathbf{V}_i^{*,\nu+1}, \mathbf{V}_i^{\nu+1,(k)} + \omega^{\nu+1}(\widetilde{\mathbf{V}}_i^{\nu+1,(k+1)} - \mathbf{V}_i^{\nu+1,(k)})\big)$;
  end for
  if $\| \mathbf{V}^{\nu+1,(k+1)} - \mathbf{V}^{\nu+1,(k)} \| \leq tol$ **then**
    break;
  end if
end for
$\mathbf{V}^{\nu+1} = \mathbf{V}^{\nu+1,(k+1)}$;

3.3.2 Penalty Method

While the PSOR method explicitly applies the constraints at each iteration of computing the option price, the penalty method applies the constraints implicitly, using a nonsmooth Newton iteration. As described earlier, the LCP for an American put is

$$\left\{ \begin{array}{r} \frac{\partial V}{\partial \tau} - \mathcal{L}V = 0 \\ V - V^* \geq 0 \end{array} \right\} \text{ or } \left\{ \begin{array}{r} \frac{\partial V}{\partial \tau} - \mathcal{L}V > 0 \\ V - V^* = 0 \end{array} \right\}, \tag{3.33}$$

where $V^* = \max(E-S, 0)$ is the payoff for an American put and $\mathcal{L}V \equiv \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV$. The idea of the penalty methods is to add a penalty term to the right hand side of equation $\frac{\partial V}{\partial \tau} - \mathcal{L}V = 0$, in order to ensure that the early exercise constraint is fulfilled. The penalty term should be negligible when $V > V^*$ so that it does not affect the solution of the Black-Scholes PDE, and of non-negligible size when V approaches V^*, so that V does not become less than V^*. Nielsen et al. propose a continuous penalty formulation in [25]. They define a barrier function $bar(S) = E - S$ and the penalty term by $\frac{\epsilon C}{V + \epsilon - bar(S)}$, where $0 < \epsilon \ll 1$ is a parameter and C is a positive constant. Thus the LCP is formulated as the nonlinear PDE $\frac{\partial V}{\partial \tau} - \mathcal{L}V = \frac{\epsilon C}{V+\epsilon-bar(S)}$. One should note that the penalty term is of order ϵ and hence very small when $V \gg bar(S)$ and thus the Black Scholes PDE is satisfied in this case. In addition, if $V \to bar(S)$, then the penalty term approaches C which makes $\frac{\partial V}{\partial \tau} - \mathcal{L}V = C > 0$ and thus the early exercise constraint is not violated.

A discrete penalty formulation is proposed by Zvan et al. in [40]. A penalty term is added to the discrete equations. The resulting problem can again be viewed as a non-linear PDE. The nonlinear discretized equations are solved by Newton iterations. This approach has the advantage that advanced discretization methods like flux limiters (see [41]) can be incorporated. In addition, other types of constraints such as time-dependent barriers can also fit easily into the framework. In this thesis, we follow this approach and we briefly describe it for an American put option below.

Problem (3.33) is replaced by the non-linear PDE

$$\frac{\partial V}{\partial \tau} - \mathcal{L}V = \hat{P}(V^*, V), \tag{3.34}$$

where $\hat{P}(V^*, V)$ is the penalty term defined by

$$\hat{P}(V^*, V) = \zeta \max(V^* - V, 0).$$

Here, the positive penalty parameter ζ, $\zeta \to \infty$, effectively ensures that the solution satisfies $V - V^* \geq -\epsilon$ for $0 < \epsilon \ll 1$.

We explain intuitively how this works. If $V > V^*$ then $\hat{P}(V^*, V) = 0$, hence (3.34) becomes

$$\left\{ \begin{array}{c} \dfrac{\partial V}{\partial \tau} - \mathcal{L}V = 0 \\ V - V^* > 0 \end{array} \right\}$$

On the other hand, if $-\epsilon \leq V - V^* < 0$, i.e. $\hat{P}(V^*, V) > 0$, then we have

$$\left\{ \begin{array}{l} \dfrac{\partial V}{\partial \tau} - \mathcal{L}V = \hat{P}(V^*, V) > 0 \\ \quad V - V^* \geq -\epsilon \quad \text{for} \quad 0 < \epsilon \ll 1 \end{array} \right\}.$$

Except for the penalty part, the discretization of equation (3.34) has been discussed in Section 3.2. Let $\hat{p}_i^{\nu+1}$ denote a discrete penalty term corresponding to the node $(\mathbf{S}_i^{\nu+1}, \boldsymbol{\tau}_{\nu+1})$, and recall the definition of the discrete differential operator $\mathcal{L}V_i^\nu$ in equation (3.15). The discretized version of equation (3.34) could be written as

$$\frac{V_i^{\nu+1} - V_i^\nu}{\Delta \boldsymbol{\tau}_\nu} = \theta \mathcal{L}V_i^{\nu+1} + (1-\theta)\mathcal{L}V_i^\nu + \hat{p}_i^{\nu+1}, \quad i = 1, 2, \ldots, n-1, \tag{3.35}$$

where the penalty term $\hat{p}_i^{\nu+1}$ is defined by

$$\hat{p}_i^{\nu+1} = \left\{ \begin{array}{ll} \dfrac{1}{\Delta \boldsymbol{\tau}_\nu}(V_i^{*,\nu+1} - V_i^{\nu+1})L & \text{if } V_i^{\nu+1} < V_i^{*,\nu+1}, \\ 0 & \text{otherwise.} \end{array} \right. \tag{3.36}$$

Here, L is a large positive number referred to as *penalty factor* chosen so that $L \simeq \mathcal{O}(\frac{1}{\epsilon})$. We will define L precisely in terms of convergence tolerance later.

Equation (3.35) represents a set of nonlinear equations due to the penalty term (3.36). Let us write these equations in matrix form. Let the diagonal matrix $\hat{\mathbf{P}} \in \mathbb{R}^{(n-1)\times(n-1)}$ be given by

$$\hat{\mathbf{p}}(V^{\nu+1})_{ij} = \begin{cases} L & \text{if } \mathbf{V}_i^{\nu+1} < \mathbf{V}_i^{*,\nu+1} \text{ and } i = j, \\ 0 & \text{otherwise.} \end{cases} \tag{3.37}$$

Recall equation (3.21) for the discretization of the Black-Scholes PDE. Then we can write equations (3.35)-(3.36) as

$$[\mathbf{I} + \theta\Delta\boldsymbol{\tau}_\nu \mathbf{M}^\nu + \hat{\mathbf{P}}(V^{\nu+1})]\mathbf{V}^{\nu+1} = (\mathbf{I} - (1-\theta)\Delta\boldsymbol{\tau}_\nu \mathbf{M}^\nu)\mathbf{V}^\nu + [\hat{\mathbf{P}}(V^{\nu+1})]\mathbf{V}^{*,\nu+1}, \tag{3.38}$$

where $\mathbf{I}$ and $\mathbf{M}$ are as defined in Section 3.2.2. With notations for $\mathbf{A}^\nu$ and $\mathbf{b}^\nu$ defined in (3.23), Equation (3.38) can be written as

$$[\mathbf{A}^\nu + \hat{\mathbf{P}}(V^{\nu+1})]\mathbf{V}^{\nu+1} = \mathbf{b}^\nu + [\hat{\mathbf{P}}(V^{\nu+1})]\mathbf{V}^{*,\nu+1}. \tag{3.39}$$

An iterative technique for the solution of Equation (3.38) is presented in Algorithm 3.

Let us further study Algorithm 3 to see how it works. Suppose we have that $\mathbf{V}_i^{\nu+1,(0)} = \mathbf{V}_i^\nu$ and $\mathbf{V}_i^\nu > \mathbf{V}_i^{*,\nu+1}$, i.e it is not optimal to early exercise at node $(\mathbf{S}_i^{\nu+1}, \boldsymbol{\tau}_{\nu+1})$. Since $\mathbf{V}_i^{\nu+1,(0)} = \mathbf{V}_i^\nu > \mathbf{V}_i^{*,\nu+1}$, we have that $\hat{\mathbf{p}}(V^{\nu+1})_{ii} = 0$. If after one iteration, $\mathbf{V}_i^{\nu+1,(1)} > \mathbf{V}_i^{*,\nu+1}$, then it is still not optimal to exercise, so $\hat{\mathbf{p}}(V^{\nu+1})_{ii} = 0$. If $\mathbf{V}_i^{\nu+1,(1)} < \mathbf{V}_i^{*,\nu+1}$, it is now optimal to exercise early so $\hat{\mathbf{p}}(V^{\nu+1})_{ii} = L$, and the penalty term $\hat{p}_i^{\nu+1}$ adds value to the solution at this node forcing $\mathbf{V}_i^{\nu+1,(2)}$ to increase, resulting in $\mathbf{V}_i^{\nu+1,(2)} > \mathbf{V}_i^{\nu+1,(1)}$. This is the monotone convergence property of the iterates.

Eventually, the iterates converge to a solution where either

$$\mathbf{V}_i^{\nu+1} \geq \mathbf{V}_i^{*,\nu+1} \Rightarrow \hat{\mathbf{p}}(V^{\nu+1})_{ii} = 0,$$

or

$$\begin{cases} \mathbf{V}_i^{*,\nu+1} - \mathbf{V}_i^{\nu+1} > 0 \Rightarrow \hat{\mathbf{p}}(V^{\nu+1})_{ii} > 0, \\ \mathbf{V}_i^{*,\nu+1} - \mathbf{V}_i^{\nu+1} \leq \epsilon; \qquad \epsilon \ll 1. \end{cases}$$

Therefore, the converged solution of Algorithm 3 either satisfies a discrete form of the Black-Scholes PDE with $\mathbf{V}_i^{\nu+1} > \mathbf{V}_i^{*,\nu+1}$ or $\mathbf{V}_i^{\nu+1} - \mathbf{V}_i^{*,\nu+1} \gtrsim -\epsilon$ for some ϵ, $0 < \epsilon \ll 1$. Thus, Algorithm 3 generates an approximation to the solution of the LCP.

It has been proved in [14] that the parameter L is closely related to the maximum relative error in enforcing the American constraint using the penalty method. This error is calculated by the quantity $\max_{i,\nu} \frac{\max[0, V_i^* - V_i^\nu]}{\max(1, V_i^*)}$. To ensure that this quantity is small, the penalty factor must be sufficiently large. In the stopping region, where early exercise is advisable, we have that $\mathbf{V}_i^{\nu+1} \gtrsim \mathbf{V}_i^{*,\nu+1} - \epsilon$. As $L \to \infty$, the solution is more accurate, i.e $\epsilon \to 0$ since $\epsilon \simeq \mathcal{O}(\frac{1}{L})$, but we could expect the round-off errors to dominate if $L > \frac{1}{\text{machine } \epsilon}$. In [14], it has been shown that as $L \to \infty$, in the stopping region where $\mathbf{V}_i^{*,\nu+1} \neq 0$ we have

$$\left| \frac{\mathbf{V}_i^{*,\nu+1} - \mathbf{V}_i^{\nu+1}}{\mathbf{V}_i^{*,\nu+1}} \right| \approx \frac{1}{L}.$$

Therefore, if we want an accuracy *tol* in Algorithm 3, we must have

$$L \simeq \frac{1}{tol}. \tag{3.40}$$

So L is well-defined, and cannot be arbitrarily large. It is worth noting that, in practice, one or two iterations usually suffices to obtain convergence. It is worth mentioning that through numerical experiments, we observe effects of two different initial guesses (3.31) and (3.32) on the convergence rate of the penalty method, although, in this case, the effects are not as significant as in the case of PSOR. In Chapter 6, we present selected numerical results of the penalty method related to this issue.

Algorithm 3: Penalty iteration for American options

initialize $\mathbf{V}^{\nu+1,(0)}$;

construct $\hat{\mathbf{P}}(V^{\nu+1,(0)})$ using (3.37);

for $k = 0, \ldots,$ until convergence **do**

  solve (3.38) for $\mathbf{V}^{\nu+1,(k+1)}$;

  construct $\hat{\mathbf{P}}(V^{\nu+1,(k+1)})$ using (3.37);

  if $\left[\max_i \frac{|\mathbf{V}_i^{\nu+1,(k+1)} - \mathbf{V}_i^{\nu+1,(k)}|}{\max(1, |\mathbf{V}^{\nu+1,(k+1)}|)} < tol\right]$ or $\left[\hat{\mathbf{P}}(V^{\nu+1,(k)}) = \hat{\mathbf{P}}(V^{\nu+1,(k+1)})\right]$ **then**

    break;

  end if

end for

$\mathbf{V}^{\nu+1} = \mathbf{V}^{\nu+1,(k+1)}$;

Chapter 4

Adaptive Mesh Methods for Pricing American Options

In this chapter, we illustrate how the adaptive mesh techniques based on the equidistribution principle are applied to the pricing of American options on one underlying asset. We will revisit Algorithm 1 in more detail and refer to Algorithms 2 and 3 as subroutines.

Recall that the price of an American option can be computed from the LCP

$$\left\{ \begin{array}{r} \dfrac{\partial V}{\partial \tau} - \mathcal{L}V = 0 \\ V - V^* \geq 0 \end{array} \right\} \text{ or } \left\{ \begin{array}{r} \dfrac{\partial V}{\partial \tau} - \mathcal{L}V > 0 \\ V - V^* = 0 \end{array} \right\}, \tag{4.1}$$

which can be solved on a discrete domain by applying either the PSOR method or the penalty method. The details of the discretization and resulting matrices and vectors for the LCP depend on the desired discretization methods and have been discussed in previous chapters. If we solve the non-linear problem by PSOR, then the resulting constrained matrix problem that needs to be solved in order to proceed from time step ν to time step $\nu + 1$ is described by

$$\left\{ \begin{array}{r} \mathbf{A}^{\nu}\mathbf{V}^{\nu+1} = \mathbf{b}^{\nu} \\ \mathbf{V}^{\nu+1} - \mathbf{V}^{*,\nu+1} \geq \mathbf{0} \end{array} \right\} \text{ or } \left\{ \begin{array}{r} \mathbf{A}^{\nu}\mathbf{V}^{\nu+1} > \mathbf{b}^{\nu} \\ \mathbf{V}^{\nu+1} - \mathbf{V}^{*,\nu+1} = \mathbf{0} \end{array} \right\}. \tag{4.2}$$

On the other hand, if the penalty method is used, then at each time step, we need to solve the

matrix problem

$$[\mathbf{A}^{\nu} + \hat{\mathbf{P}}(V^{\nu+1})]\mathbf{V}^{\nu+1} = \mathbf{b}^{\nu} + [\hat{\mathbf{P}}(V^{\nu+1})]\mathbf{V}^{*,\nu+1}, \tag{4.3}$$

possibly more than once, but usually at most once or twice per timestep. Here, matrix $\mathbf{A}^{\nu}$ and vector $\mathbf{b}^{\nu}$ are defined in (3.23) and the penalty matrix $\hat{\mathbf{P}}$ is defined in (3.37).

4.1 Algorithm Description

Algorithm 1 assumes that an approximation to V^{ν} is already computed on partition Δ_{ν}, and computes an approximation to $V^{\nu+1}$ on partition $\Delta_{\nu+1}$ which may be different from Δ_{ν}. This is done by using a grading function which involves high-order derivatives of V. However, we did not specify how the appropriate derivatives of V at fixed time $\boldsymbol{\tau}_{\nu+1}$ and the estimate of the distribution of the error are calculated. In this chapter, we will provide more details on these matters. As mentioner earlier, the monitor function at time τ that we are using is $\hat{V}(S,\tau) = |V^{(3)}|^{1/3}$, and the corresponding grading function is

$$\xi(S,\tau) = \frac{\int_a^S |V^{(3)}|^{1/3} dS}{\int_a^b |V^{(3)}|^{1/3} dS}, \tag{4.4}$$

where $V^{(3)}$ denotes $\dfrac{\partial^3 V}{\partial S^3}$. Since we do not have a closed-form expression for the derivative of V, we have to use approximate values $\mathbf{V}_i$, $i = 0, \ldots, n$, to approximate the third derivative by the means of finite differences. The integrals appearing in the grading function can be approximated by appropriate quadrature rules such as the midpoint or trapezoidal rule. We now introduce some new notations used in the adaptive algorithm. For easy reference, we also repeat relevant previously defined notations below.

Δ_{ν}: spatial partition at time $\boldsymbol{\tau}_{\nu}$, $\Delta_{\nu} \equiv \{\mathbf{S}_i^{\nu}\}_{i=0}^{n}$, where $\mathbf{S}_i^{\nu}$ is the *i*th spatial point at time $\boldsymbol{\tau}_{\nu}$;

$\mathbf{V}_{\Delta_k}^{\nu}$: vector of approximate values to V at time $\boldsymbol{\tau}_{\nu}$ on space partition Δ_k with components $\mathbf{V}_{k,i}^{\nu} \approx V(\mathbf{S}_i^k, \boldsymbol{\tau}_{\nu})$; for simplicity let $V_{k,i}^{\nu}$ denote $V(\mathbf{S}_i^k, \boldsymbol{\tau}_{\nu})$;

$\hat{\mathbf{V}}^{\nu}$: vector of approximate values to $\hat{V}(S, \boldsymbol{\tau}_{\nu}) = |V^{(3)}|^{1/2}$ on partition Δ_{ν}, with the ith component being $\hat{\mathbf{V}}_i^{\nu} \approx \hat{V}(\mathbf{S}_i^{\nu}, \boldsymbol{\tau}_{\nu})$;

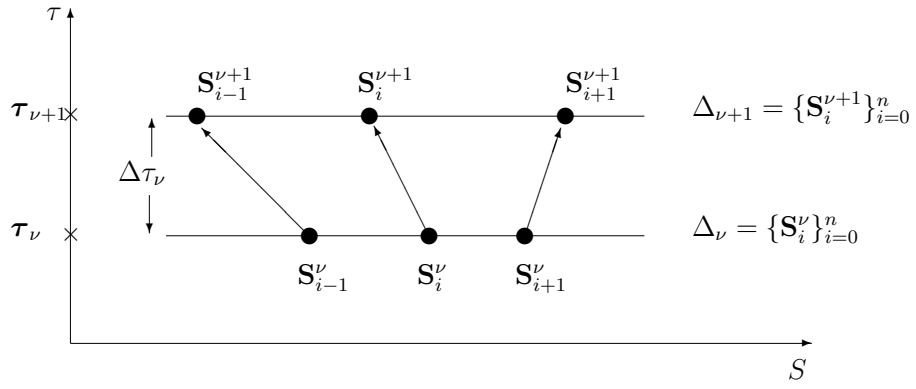


Figure 4.1: Details and notations of an adaptive step

$\boldsymbol{\xi}^\nu$ and $\boldsymbol{\xi}'^\nu$: vector of approximate values to $\xi(S, \boldsymbol{\tau}_\nu)$ and $\dfrac{\partial \xi}{\partial S}(S, \tau_\nu)$ on partition Δ_ν, respectively;

$\boldsymbol{\xi}_i^\nu$: the ith component of $\boldsymbol{\xi}^\nu$, with $\boldsymbol{\xi}_i^\nu \approx \xi(\mathbf{S}_i^\nu, \boldsymbol{\tau}_\nu)$;

We now briefly describe the algorithm. First of all, we initialize all financial parameters of the pricing problem such as E, T, σ, r as well as the grid variables including the truncated domain $[\mathbf{S}_{\min}, \mathbf{S}_{\max}]$, number of space partition points n, number of time steps $\nu_{\max}$. In addition, we also need the accuracy tolerance for the iterative method that we use to solve the LCP and initial value of relaxation parameter $\omega \in [1, 2]$ if PSOR is used.

Generally, the core calculations for each time step consist of four main parts: (I) solving the LCP, (II) checking if a new partition is needed, (III) computing the new grid, (IV) computing values on the new grid. If the current grid is well-distributed, steps (III-IV) are omitted. We now describe each part in detail.

In part (I) (Line 4), we solve the LCP (4.1) to proceed from time step ν to time step $\nu + 1$ using either the PSOR method or the penalty method. For either method, we need to construct the matrix $\mathbf{A}^\nu$ and vectors $\mathbf{b}^\nu$ and $\mathbf{V}^{*,\nu+1}$ based on the spatial partition Δ_ν and time step size $\Delta\tau_\nu$. The construction formulas for $\mathbf{A}^\nu$, $\mathbf{b}^\nu$, and $\mathbf{V}^{*,\nu+1}$ are defined in (3.23). If PSOR is used, the solution of the LCP can be obtained from Algorithm 2 with smart initial guess based on

extrapolation as defined in (3.32). If we use the penalty method, Algorithm 3 is invoked. After this step we obtain $\mathbf{V}_{\Delta_\nu}^{\nu+1}$, the vector of approximate solution at time $\boldsymbol{\tau}_{\nu+1}$ but on the same partition Δ_ν of the previous time step.

In part (II) (Lines 5 - 7), we determine whether or not the current partition Δ_ν is well-distributed for the time $\boldsymbol{\tau}_{\nu+1}$. The general idea of this procedure is explained in Algorithm 1. Here, we discuss it in more detail. First, we approximate the monitor function $\hat{V}(S, \boldsymbol{\tau}_{\nu+1})$ on partition $\Delta_{\nu+1} = \Delta_\nu$ using $\mathbf{V}_{\Delta_\nu}^{\nu+1}$ (Line 5). The FD approximation formula for the third derivative appearing in the monitor function is

$$\frac{\partial^3 V_{\nu,i}^{\nu+1}}{\partial S^3} \approx 3!\big(c_{i1}^\nu \mathbf{V}_{\nu,i-1}^{\nu+1} + c_{i2}^\nu \mathbf{V}_{\nu,i}^{\nu+1} + c_{i3}^\nu \mathbf{V}_{\nu,i+1}^{\nu+1} + c_{i4}^\nu \mathbf{V}_{\nu,i+2}^{\nu+1}\big), \tag{4.5}$$

where

$$\begin{aligned}
c_{i1}^\nu &= \frac{1}{-h_{i-1}^\nu (h_{i-1}^\nu + h_i^\nu)(h_{i-1}^\nu + h_i^\nu + h_{i+1}^\nu)},\\
c_{i2}^\nu &= \frac{1}{h_{i-1}^\nu h_i^\nu (h_i^\nu + h_{i+1}^\nu)},\\
c_{i3}^\nu &= \frac{1}{-h_i^\nu h_{i+1}^\nu (h_{i-1}^\nu + h_i^\nu)},\\
c_{i4}^\nu &= \frac{1}{h_{i+1}^\nu (h_i^\nu + h_{i+1}^\nu)(h_{i-1}^\nu + h_i^\nu + h_{i+1}^\nu)}.
\end{aligned}$$

We obtain the vector $\hat{\mathbf{V}}^{\nu+1}$ whose components are $\hat{\mathbf{V}}_i^{\nu+1} \approx |V^{(3)}(\mathbf{S}_i^{\nu+1}, \boldsymbol{\tau}_{\nu+1})|^{1/2}, i = 0, 1, \ldots, n$. Note that $\hat{\mathbf{V}}_0^{\nu+1}$ and $\hat{\mathbf{V}}_n^{\nu+1}$ are set to zero due to the fact that the values at the two boundary points are given, hence no errors. Approximating the grading function ξ and its derivative $\dfrac{\partial \xi}{\partial S}$ as well as quantities $\tilde{r}_i^{\nu+1}$ and $\tilde{r}^{\nu+1}$ (Line 6) on partition $\Delta_{\nu+1} = \Delta_\nu$ is done by employing some quadrature rules such as the midpoint or trapezoid rules. If the points of partition $\Delta_{\nu+1}$ are well-distributed (meaning either criterion (2.13) or (2.14) is satisfied), then partition $\Delta_{\nu+1} = \Delta_\nu$ and $\mathbf{V}_{\Delta_{\nu+1}}^{\nu+1} = \mathbf{V}_{\Delta_\nu}^{\nu+1}$ are accepted and we move to the next time step (Line 9). If not, the new partition $\Delta_{\nu+1} \neq \Delta_\nu$ is computed using one iteration of (2.10) (Line 11). This step is in part (III).

Once a new partition is generated, we need to compute the values of approximation on the new $\Delta_{\nu+1}$. The steps involved are in part (IV) (Lines 12-18). As mentioned in Algorithm

1, there are two ways to do this. The first is to interpolate $\mathbf{V}^{\nu}_{\Delta_\nu}$ from the old partition Δ_ν to the new partition $\Delta_{\nu+1}$ to obtain $\mathbf{V}^{\nu}_{\Delta_{\nu+1}}$ and apply the time-stepping procedure to compute $\mathbf{V}^{\nu+1}_{\Delta_{\nu+1}}$ (Lines 13-15). The second way is simply to interpolate $\mathbf{V}^{\nu+1}_{\Delta_\nu}$ from Δ_ν to $\Delta_{\nu+1}$ to obtain $\mathbf{V}^{\nu+1}_{\Delta_{\nu+1}}$ (Line 17). The output of the algorithm would be the partition at the last time step together with an approximation on that partition. The free boundary is also calculated. The Figure 4.1 illustrates a subset of the points used in steps ν and $\nu + 1$.

Algorithm 4: Adaptive algorithm to compute American option

Initialization of variables and parameters

Core calculations

for $\nu = 0, 1, \ldots, \nu_{\max} - 1$ **do**

solve the matrix problem (4.2) for $\mathbf{V}_{\Delta_\nu}^{\nu+1}$; set $\Delta_{\nu+1} = \Delta_\nu$;

construct $\hat{\mathbf{V}}^{\nu+1}$ using (4.5);

approximate $\vartheta = \int_{\mathbf{S}_{\min}}^{\mathbf{S}_{\max}} \hat{V}(S, \boldsymbol{\tau}_{\nu+1})dS$, $\boldsymbol{\xi}_i^{\nu+1} = \frac{\int_{\mathbf{S}_{\min}}^{\mathbf{S}_i^{\nu+1}} \hat{V}(S, \boldsymbol{\tau}_{\nu+1})dS}{\vartheta}$, $\boldsymbol{\xi}_i'^{\nu+1} = \frac{\hat{\mathbf{V}}_i^{\nu+1}}{\vartheta}$, $\tilde{r}_i^{\nu+1} = \int_{\mathbf{S}_{i-1}^{\nu+1}}^{\mathbf{S}_i^{\nu+1}} \hat{V}(S, \boldsymbol{\tau}_{\nu+1})dS$, and $\tilde{r}^{\nu+1} = \frac{\vartheta}{n}$;

check stopping criterion (2.13) or (2.14)

if (2.13) or (2.14) is satisfied **then**

$\Delta_{\nu+1} = \Delta_\nu$; $\mathbf{V}_{\Delta_{\nu+1}}^{\nu+1} = \mathbf{V}_{\Delta_\nu}^{\nu+1}$;

else

compute **(new)** $\Delta_{\nu+1} \equiv \{\mathbf{S}_i^{\nu+1}\}_{i=0}^n$ using $\mathbf{S}_i^{\nu+1} = \mathbf{S}_i^{\nu+1} - \frac{\boldsymbol{\xi}_i^{\nu+1} - \frac{i}{n}}{\boldsymbol{\xi}_i'^{\nu+1}}$;

if $\nu \leq$ *smallnum* **then**

compute $\mathbf{V}_{\Delta_{\nu+1}}^{\nu}$ using interpolation on $\left((\mathbf{S}_i^{\nu})_{i=0}^n, \mathbf{V}_{\Delta_\nu}^{\nu}\right)$;

construct new matrix $\mathbf{A}^\nu$, and vectors $\mathbf{b}^\nu$ and $V^{*,\nu+1}$ on the new partition $\Delta_{\nu+1}$;

solve again the matrix problem (4.2) for $\mathbf{V}_{\Delta_{\nu+1}}^{\nu+1}$ using $\mathbf{V}_{\Delta_{\nu+1}}^{\nu}$;

else

compute $\mathbf{V}_{\Delta_{\nu+1}}^{\nu+1}$ using interpolation on $\left((\mathbf{S}_i^{\nu})_{i=0}^n, \mathbf{V}_{\Delta_\nu}^{\nu+1}\right)$;

end if

end if

end for

Results

output $\Delta_{\nu_{\max}}$ and $\mathbf{V}_{\Delta_{\nu_{\max}}}^{\nu_{\max}}$;

compute and output the free boundary S_f;

Regarding the two ways of handling part (IV), note that the first technique involves solving the LCP again on the new partition $\Delta_{\nu+1}$. To do this, we need to construct the matrix $\mathbf{A}^{\nu}$, and vectors $\mathbf{b}^{\nu}$ and $V^{*,\nu+1}$ again on the new partition and apply an iterative method to solve the problem. This part is costly and thus should be avoided, if not needed. Through experiments we observe that we need to apply the first technique only for the first few time steps (*smallnum*). This produces good results without sacrificing computation time.

Normally, in the case of European options or other problems without constraints, one can use standard interpolation techniques, such as cubic spline interpolation, to obtain interpolated values. However, due to existence of the free boundary at each timestep, the interpolation techniques used in the pricing of American option must be chosen carefully. At each timestep, the free boundary point separates the spatial domain into the continuation region and the stopping region. The relationships between the option value and the payoff in the continuation and the stopping regions are expressed in (3.3) and (3.4), respectively. In order to maintain accuracy of the option valuation, interpolated option values which are obtained at Line 13 (timestep ν) or Line 17 (timestep $\nu+1$) of Algorithm 4 must satisfy these requirements. This means that these interpolated option values must be either (a) equal (within some tolerance) to the payoff in the stopping region or (b) larger than the payoff in the continuation region. With cubic spline interpolation techniques, we observe that (b) is always satisfied. However, (a) is not always met since the interpolated option values are sometimes larger than the corresponding payoff values in the stopping region. This could result in poor approximation of the free boundary point at that timestep and could affect the accuracy of the option value at the next timestep. As time evolves, this could diminish the accuracy of the option value approximation and free boundary point approximation at the last timestep. To deal with this situation, we first use standard cubic spline interpolation to obtain interpolated values. We then approximate the free boundary point at that timestep using the partition and option values available, namely $\left((\mathbf{S}_i^{\nu})_{i=0}^{n}, \mathbf{V}_{\Delta_\nu}^{\nu}\right)$ (Line 13) or $\left((\mathbf{S}_i^{\nu})_{i=0}^{n}, \mathbf{V}_{\Delta_\nu}^{\nu+1}\right)$ (Line 17). All interpolated values in the stopping region are set to the corresponding payoff values. Experiments show that this simple technique works pretty well

for American options.

Chapter 5

Stability and Convergence Analysis

In this chapter, we study the stability and convergence of the adaptive method used for pricing American options introduced in previous chapter.

The numerical solution of the pricing an American option typically consists of two parts: (1) discretizing the problem on a finite mesh and (2) computing an approximate solution to the resulting constrained matrix problem using an iterative method. As we mentioned earlier, there are various ways to discretize the problem and also there is a variety of iterative solvers that can be used for constrained matrix problems. Since the convergence of iterative solvers for constrained matrix problems can be guaranteed for specific classes of matrices it is important to investigate the properties of matrices resulting from our approach.

One of the most favorable class of matrices is the class of symmetric positive definite matrices. The stability and convergence of many solvers are guaranteed in this case. However, in our case, the underlying matrices are highly non-symmetric and due to the non-uniform grids resulting from adaptivity, it is fairly hard to enforce symmetry and at the same time preserve the accuracy of the method. Hence, we need to consider a favorable subclass of non-symmetric matrices which can guarantee the stability and convergence. The class that we will examine is the class of $\mathcal{M}$-matrices.

5.1 Preliminaries

Let $\mathbf{x}$ be a vector in $\mathbb{R}^n$ and $\mathbf{A}$ be a square matrix in $\mathbb{R}^{n\times n}$. As previously defined, we write $\mathbf{a}_{i,j}$ for the *(i, j)*th element of matrix $\mathbf{A}$ and $\mathbf{x}_i$ for the *i*th component of vector $\mathbf{x}$. We say $\mathbf{A} \geq 0$ if $\mathbf{a}_{i,j} \geq 0, \forall i, j,\ 1 \leq i, j \leq n$. Also, recall that $\mathbf{A}$ is *strictly diagonal dominant* if $|\mathbf{a}_{i,i}| > \sum_{j=1, j\neq i}^{n} |\mathbf{a}_{i,j}|$, and *diagonal dominant* if $|\mathbf{a}_{i,i}| \geq \sum_{j=1, j\neq i}^{n} |\mathbf{a}_{i,j}|$. We say matrix $\mathbf{A}$ is reducible if there is permutation $\mathbb{P}$ under which $\mathbf{A}$ has the structure

$$\begin{pmatrix} \mathbf{A}_1 & 0 \\ \mathbf{B} & \mathbf{A}_2 \end{pmatrix},$$

where $\mathbf{A}_1$ and $\mathbf{A}_2$ are square matrices. A matrix is irreducible if it is not reducible.

Definition 1. *($\mathcal{L}$-matrix,*[39]*)*

*A real matrix $\mathbf{A}$ in $\mathbb{R}^{n\times n}$ is said to be an $\mathcal{L}$-*matrix *if*

$$\mathbf{a}_{i,i} > 0, \qquad \forall i, \qquad 1 \leq i \leq n, \tag{5.1}$$

and

$$\mathbf{a}_{i,j} \leq 0, \qquad \forall i \neq j, \qquad 1 \leq i, j \leq n. \tag{5.2}$$

Definition 2. *($\mathcal{M}$-matrix,*[39]*)*

A real matrix $\mathbf{A}$ in $\mathbb{R}^{n\times n}$ is called an $\mathcal{M}$-matrix if (5.2) holds, if $\mathbf{A}$ is nonsingular, and if $\mathbf{A}^{-1} \geq \mathbf{0}$.

Definition 3. *(irreducibly (row) diagonally dominant matrix,*[32]*)*

A real matrix $\mathbf{A}$ in $\mathbb{R}^{n\times n}$ is called an irreducibly (row) diagonally dominant if it is irreducible and diagonally dominant with strict diagonal dominance in at least one row.

Note that while the $\mathcal{L}$-matrix property are easy to verify, the $\mathcal{M}$-matrix property is not, since normally we are interested in large and sparse matrices whose inverses are costly to compute and explicit formulae for the inverse elements are not available. For this reason, we are interested in sufficient conditions for $\mathcal{M}$-matrix structure. Some of important criterions of the $\mathcal{M}$-matrix structure are listed below.

Theorem 1. *A strictly diagonally dominant $\mathcal{L}$-matrix is an $\mathcal{M}$-matrix.*

Proof. A proof can be found in [16]. □

Theorem 2. *An irreducibly (row) diagonally dominant $\mathcal{L}$-matrix is an $\mathcal{M}$-matrix.*

Proof. A proof can be found in [16] or in [32]. □

With respect to the American option pricing problem, we need to solve the constrained matrix problem (3.22) at each time step. We employ either the PSOR or the penalty methods. The convergence of the penalty method on American option pricing has been proved in [14] under the condition that $\mathbf{M}^\nu$ is an $\mathcal{M}$-matrix. (Note that if $\mathbf{M}^\nu$ is an $\mathcal{M}$-matrix, so is $\mathbf{A}^\nu$). However, the convergence condition for PSOR mentioned in [9] requires $\mathbf{A}^\nu$ to be symmetric positive definite and obviously this condition is not satisfied in our case. For that reason, we would like to resort to the $\mathcal{M}$-matrix class to prove that the PSOR method for American option pricing is convergent if matrix $\mathbf{A}^\nu$ is an $\mathcal{M}$-matrix. In Appendix A, we present a proof of convergence for the PSOR method on American option pricing under certain conditions involving matrix $\mathbf{A}$ and the relaxation parameter ω.

Normally, sufficient conditions for a matrix to be an $\mathcal{M}$-matrix listed in Theorem 2 are milder than those listed in Theorem 1. Nonetheless, both theorems require the $\mathcal{L}$-matrix structure. As we shall see, the $\mathcal{L}$-matrix structure automatically results in strict diagonal dominance in our case. For this reason, we will study the sufficient conditions in Theorem 1.

5.2 $\mathcal{L}$-matrix Property

We first investigate the $\mathcal{L}$-matrix structure of $\mathbf{M}^\nu$. Recall that to have $\mathcal{L}$-matrix structure, the entries of $\mathbf{M}^\nu$ must satisfy

$$\mathbf{m}_{i,j}^\nu = \begin{cases} \leq 0 & \forall i \neq j \\ > 0 & \forall i = j \end{cases}$$

As we mentioned earlier, $\mathbf{M}^\nu$ is a tridiagonal matrix of the following form

$$\mathbf{M}^\nu = \begin{pmatrix} \mathbf{m}^\nu_{1,1} & \mathbf{m}^\nu_{1,2} & 0 & \dots & 0 \\ \mathbf{m}^\nu_{2,1} & \mathbf{m}^\nu_{2,2} & \mathbf{m}^\nu_{2,3} & \dots & 0 \\ \ddots & \ddots & \ddots & \ddots & \ddots \\ \ddots & \mathbf{m}^\nu_{i,i-1} & \mathbf{m}^\nu_{i,i} & \mathbf{m}^\nu_{i,i+1} & \ddots \\ \ddots & \ddots & \ddots & \ddots & \ddots \\ 0 & \dots & \mathbf{m}^\nu_{n-2,n-3} & \mathbf{m}^\nu_{n-2,n-2} & \mathbf{m}^\nu_{n-2,n-1} \\ 0 & \dots & 0 & \mathbf{m}^\nu_{n-1,n-2} & \mathbf{m}^\nu_{n-1,n-1} \end{pmatrix},$$

where the entries $\mathbf{m}^\nu_{i,i-1}, \mathbf{m}^\nu_{i,i}, \mathbf{m}^\nu_{i,i+1}$ are defined in (3.12).

Obviously, all super-diagonal entries $\mathbf{m}^\nu_{i,i+1}$, $i = 2, 3, \dots, n-1$, of $\mathbf{M}^\nu$ are negative and hence non-positive since $a^\nu_{i3} = \frac{h^\nu_i}{h^\nu_{i+1}(h^\nu_i + h^\nu_{i+1})}$ and $b^\nu_{i3} = \frac{2}{h^\nu_{i+1}(h^\nu_i + h^\nu_{i+1})}$ are both positive. However, sub-diagonal entries $\mathbf{m}^\nu_{i,i-1}$, $i = 2, 3, \dots, n-1$, of $\mathbf{M}^\nu$ are not guaranteed to be non-positive. The explicit formula for $\mathbf{m}^\nu_{i,i-1}$ is

$$\begin{aligned} \mathbf{m}^\nu_{i,i-1} &= -\frac{1}{2}\sigma^2(\mathbf{S}^\nu_i)^2 b^\nu_{i1} - r\mathbf{S}^\nu_i a^\nu_{i1} \\ &= -\frac{\sigma^2(\mathbf{S}^\nu_i)^2}{h^\nu_i(h^\nu_i + h^\nu_{i+1})} + \frac{h^\nu_{i+1} r\mathbf{S}^\nu_i}{h^\nu_i(h^\nu_i + h^\nu_{i+1})}. \end{aligned}$$

Note that non-positive off-diagonal in $\mathbf{M}^\nu$ could result from the convection term $\frac{\partial V}{\partial S}$ dominating the diffusion term $\frac{\partial^2 V}{\partial S^2}$. In this case, the central finite differences produce non-positive sub-diagonal entries in $\mathbf{M}^\nu$. We could fix this by using forward differences for $\frac{\partial V}{\partial S}$ instead of central differences so that $\mathbf{m}^\nu_{i,i-1} = -\frac{\sigma^2(\mathbf{S}^\nu_i)^2}{h^\nu_i(h^\nu_i + h^\nu_{i+1})}$, $i = 2, 3, \dots, n-1$, and hence non-positive. However, this modification could result in overall first order of convergence, except that if only a few nodes are modified then we could expect order of convergence between 1 and 2. For this reason, we do not follow this approach. Another approach which guarantees non-positive sub-diagonal entries for $\mathbf{M}^\nu$ is to impose conditions on the spatial step sizes so that $\mathbf{m}^\nu_{i,i-1}$, $i = 2, 3, \dots, n-1$, is non-positive, that is $\mathbf{m}^\nu_{i,i-1} \le 0$. This results in the following condition on the step sizes:

$$h^\nu_{i+1} \le \frac{\sigma^2 \mathbf{S}^\nu_i}{r}, \qquad i = 2, 3, \dots, n-1. \tag{5.3}$$

This condition imposes a restriction on points $\mathbf{S}_{i+1}^{\nu}$, $i = 2, 3, \ldots, n-1$, leaving the first two interior points $\mathbf{S}_1^{\nu}$ and $\mathbf{S}_2^{\nu}$ unrestricted. The reason is that the step size h_1^{ν} appears in the formula for $\mathbf{m}_{1,3}^{\nu}$ which is a negative super-diagonal entry and h_2^{ν} appears in the first entry $\mathbf{b}_1^{\nu}$ of $\mathbf{b}^{\nu}$ due to Dirichlet boundary conditions.

For constant spatial step sizes, if the partition point $\mathbf{S}_3^{\nu}$ satisfies (5.3), then all other points $\mathbf{S}_i^{\nu}, i > 3$, of the partition satisfy (5.3) as well. However, on a non-uniform grid, this condition must be imposed on each node of the partition. We will explain in detail later how condition (5.3) is ensured at each time step.

So far, we have discussed conditions that guarantee $\mathbf{M}^{\nu}$ to have $\mathbf{m}_{i,j}^{\nu} \leq 0, \forall i \neq j$. To ensure that $\mathbf{M}^{\nu}$ is a $\mathcal{L}$-matrix, we must check the positivity of all diagonal entries of $\mathbf{M}^{\nu}$. We have that

$$\mathbf{m}_{i,i}^{\nu} = -\frac{1}{2}\sigma^2(\mathbf{S}_i^{\nu})^2 b_{i2} - r\mathbf{S}_i^{\nu} a_{i2} + r.$$

Note that $a_{i2} = -(a_{i1} + a_{i3})$ and $b_{i2} = -(b_{i1} + b_{i3})$ are both negative. Taking into account σ, r and $\mathbf{S}_i^{\nu}$ are positive, it is obvious that $\mathbf{m}_{ii}^{\nu}$ is positive. Hence, all diagonal entries of $\mathbf{M}^{\nu}$ are positive. It follows that if condition (5.3) is ensured, then matrix $\mathbf{M}^{\nu}$ is an $\mathcal{L}$-matrix.

Now we consider matrix $\mathbf{A}^{\nu}$. Since $\mathbf{A}^{\nu} = \mathbf{I} + \theta\Delta\boldsymbol{\tau}_{\nu}\mathbf{M}^{\nu}$, it follows that under condition (5.3), $\mathbf{A}^{\nu}$ is also an $\mathcal{L}$-matrix. Note that $\mathbf{A}^{\nu}$ is also a tridiagonal matrix with non-positive off-diagonal entries and positive diagonal entries, taking into account θ and $\Delta\boldsymbol{\tau}_{\nu}$ are positive.

5.3 Diagonal Dominance

We now investigate the diagonal dominance property of matrix $\mathbf{M}^{\nu}$. Consider the ith row, $2 \leq i \leq n-2$ of matrix $\mathbf{M}^{\nu}$. Taking into account that $a_{i2} = -(a_{i1}+a_{i3})$ and $b_{i2} = -(b_{i1}+b_{i3})$,

we have

$$\begin{aligned}
|\mathbf{m}_{i,i}^{\nu}| &= |-\frac{1}{2}\sigma^2(\mathbf{S}_i^{\nu})^2 b_{i2} - r\mathbf{S}_i^{\nu} a_{i2} + r| \\
&= |\frac{1}{2}\sigma^2(\mathbf{S}_i^{\nu})^2(b_{i1}+b_{i3}) + r\mathbf{S}_i^{\nu}(a_{i1}+a_{i3}) + r| \\
&= |\frac{1}{2}\sigma^2(\mathbf{S}_i^{\nu})^2 b_{i1} + r\mathbf{S}_i^{\nu} a_{i1} + \frac{1}{2}\sigma^2(\mathbf{S}_i^{\nu})^2 b_{i3} + r\mathbf{S}_i^{\nu} a_{i3} + r| \\
&= |-\mathbf{m}_{i,i-1}^{\nu} - \mathbf{m}_{i,i+1}^{\nu} + r|.
\end{aligned}$$

As mentioned in the previous section, under condition (5.3), all off-diagonal entries $\mathbf{m}_{i,i-1}^{\nu}, \mathbf{m}_{i,i+1}^{\nu}$ of $\mathbf{M}^{\nu}$ are non-positive. It then follows that

$$\begin{aligned}
|\mathbf{m}_{i,i}^{\nu}| &= |-\mathbf{m}_{i,i-1}^{\nu} - \mathbf{m}_{i,i+1}^{\nu} + r|, \\
&= |\mathbf{m}_{i,i-1}^{\nu}| + |\mathbf{m}_{i,i+1}^{\nu}| + r, \\
&> |\mathbf{m}_{i,i-1}^{\nu}| + |\mathbf{m}_{i,i+1}^{\nu}|.
\end{aligned}$$

The first and last rows of $\mathbf{M}^{\nu}$ are obviously strictly diagonally dominant. Thus matrix $\mathbf{M}^{\nu}$ is strictly diagonally dominant under the condition (5.3) on step sizes. Under this condition, the matrix $\mathbf{M}^{\nu}$ is both strictly diagonally dominant and possesses the $\mathcal{L}$-matrix structure and thus by Theorem 1, it is an $\mathcal{M}$-matrix. Moreover, if $\mathbf{M}^{\nu}$ is strictly diagonally dominant then $\mathbf{A}^{\nu}$ is too. It follows that if (5.3) is true, $\mathbf{A}^{\nu}$ is an $\mathcal{M}$-matrix.

5.4 Implementation Issues

We have shown that under condition (5.3), the matrix $\mathbf{M}^{\nu}$ possesses the $\mathcal{L}$-matrix structure and is strictly diagonally dominant and hence it is an $\mathcal{M}$-matrix. Matrix $\mathbf{A}^{\nu}$ is also an $\mathbf{M}^{\nu}$ under this condition. However, condition (5.3) may limit the movement of points induced by the adaptive technique and hence may affect the quality of the resulting partition. In this section, we discuss some of the implementation issues related to this matter.

Note that condition (5.3) involves only the stepsizes of the space partition as well as volatility and risk-free interest rate with the latter two being constant. That means if the partition Δ_{ν}

satisfies condition (5.3) then this condition is guaranteed at all subsequent time steps until the adaptivity is invoked and a new partition is obtained. For this reason, to ensure condition (5.3) at any time step, we do as follows. We impose condition (5.3) for $\nu = 0$, that is, when we proceed from time $\boldsymbol{\tau}_0 = 0$ to time $\boldsymbol{\tau}_1$. Whenever the adaptive technique is invoked, we control the movement of points (only if we have to) so that the resulting partition satisfies the step size condition. We do not have to check condition (5.3) until the adaptive movement of points is called again. By this way, we can always guarantee that at any times $\boldsymbol{\tau}_\nu$, the matrices $\mathbf{M}^\nu$ and $\mathbf{A}^\nu$ have the $\mathcal{M}$-matrix structure.

We now discuss how to enforce condition (5.3) in the adaptive procedure. Assume that we are proceeding from time step ν to $\nu + 1$ and the adaptivity is invoked. Note that it is ensured that $\mathbf{M}^\nu$ and $\mathbf{A}^\nu$ are $\mathcal{M}$-matrices as we already explained. However $\mathbf{M}^{\nu+1}$ and $\mathbf{A}^{\nu+1}$ may not be. Recall that the new partition $\Delta_{\nu+1}$ is constructed point by point in one sweep from the left boundary to the right boundary. During the construction process, if the point $\mathbf{S}_{i+1}^{\nu+1}$ violates condition (5.3), meaning $h_{i+1}^{\nu+1} > \frac{\sigma^2 \mathbf{S}_i^{\nu+1}}{r}$, then we enforce $h_{i+1}^{\nu+1} = \frac{\sigma^2 \mathbf{S}_i^{\nu+1}}{r}$. By this way, we can ensure condition (5.3) and at the same time minimize our interference with the movement of points produced by the adaptive technique. However, there is one catch with this approach. Consider the interior point $\mathbf{S}_{n-1}^{\nu+1}$ which is close to the right boundary. Once this point is constructed, we do not have any choice for the next point $\mathbf{S}_n^{\nu+1}$ if $h_n^{\nu+1}$ fails the condition since the next point is a boundary point hence fixed. Thus we cannot impose condition (5.3) on the last step size and consequently, $\mathbf{M}^{\nu+1}$ and $\mathbf{A}^{\nu+1}$ may not be an $\mathcal{L}$-matrix since it may have a positive off-diagonal entry on the last row. In this case, they are nearly $\mathcal{M}$-matrices. However, we can always introduce a few more points so that condition (5.3) is satisfied. We monitor very carefully condition (5.3) throughout the testing and noticed that the points generated by the adaptive procedure never violated this condition and hence we never had to interfere with the adaptive technique.

5.5 Crank-Nicolson Method

In this section we would like to mention two important issues of the CN method: (a) the stability and (b) spurious oscillations and suggested remedies.

Recall the θ-timestepping method described by (3.20)

$$\mathbf{V}_i^{\nu+1} + \theta\Delta\boldsymbol{\tau}_\nu(\mathbf{m}_{i,i-1}^{\nu}\mathbf{V}_{i-1}^{\nu+1} + \mathbf{m}_{i,i}^{\nu}\mathbf{V}_i^{\nu+1} + \mathbf{m}_{i,i+1}^{\nu}\mathbf{V}_{i+1}^{\nu+1}) \\ = \mathbf{V}_i^{\nu} - (1-\theta)\Delta\boldsymbol{\tau}_\nu(\mathbf{m}_{i,i-1}^{\nu}\mathbf{V}_{i-1}^{\nu} + \mathbf{m}_{i,i}^{\nu}\mathbf{V}_i^{\nu} + \mathbf{m}_{i,i+1}^{\nu}\mathbf{V}_{i+1}^{\nu}),$$

or in matrix form (3.21)

$$(\mathbf{I} + \theta\Delta\boldsymbol{\tau}_\nu\mathbf{M}^\nu)\mathbf{V}^{\nu+1} = (\mathbf{I} - (1-\theta)\Delta\boldsymbol{\tau}_\nu\mathbf{M}^\nu)\mathbf{V}^{\nu}.$$

With $\theta = \frac{1}{2}$, we obtain the Crank-Nicolson (CN) method. Generally speaking, the numerical solution $\mathbf{V}^{\nu+1}$ will not be equal to the true solution $V^{\nu+1}$ due to (a) truncation errors introduced at each timestep by FD approximations and (b) round-off errors. In the case of an unstable method, such errors can grow without bound as the number of timesteps increases. Informally speaking, in option pricing, we would like the option price to stay finite as the number of timesteps increases ($\nu_{\max} \to \infty$) for a fixed T and a finite computational domain $[0, \mathbf{S}_{\max}]$.

We can write (3.21) as

$$\mathbf{V}^{\nu+1} = (\mathbf{I} + \theta\Delta\boldsymbol{\tau}_\nu\mathbf{M}^\nu)^{-1}(\mathbf{I} - (1-\theta)\Delta\boldsymbol{\tau}_\nu\mathbf{M}^\nu)\mathbf{V}^{\nu},$$

or

$$\mathbf{V}^{\nu+1} = \mathbf{B}\mathbf{V}^{\nu}$$

with

$$\mathbf{B} = (\mathbf{I} + \theta\Delta\boldsymbol{\tau}_\nu\mathbf{M}^\nu)^{-1}(\mathbf{I} - (1-\theta)\Delta\boldsymbol{\tau}_\nu\mathbf{M}^\nu).$$

Thus

$$\mathbf{V}^{\nu} = \mathbf{B}^{\nu}\mathbf{V}^{0},$$

where $\mathbf{V}^0$ is an initial values at time $\tau = 0$. We would like to have $\|\mathbf{V}^\nu\|_p$ bounded independently of the number of timesteps $\nu_{\max}$ and number of spatial partition points n (as

$n, \nu_{\max} \to \infty$). Here, $\|.\|_p$ denotes a vector norm with p either ∞ or the RMS-norm (where $\|x\|_{\text{RMS}} = \|x\|_2/\sqrt{n} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}$ for $x \in \mathbb{R}^n$).

In case the case the matrix $\mathbf{B}$ is symmetric, the set of eigenvectors of $\mathbf{B}$ forms a orthogonal basis for $\mathbb{R}^n$. Thus, we can decompose $\mathbf{V}^0$ and any error vector into a linear combination of those eigenvectors. In this case, the necessary and sufficient conditions for stability is that all eigenvalues of $\mathbf{B}$ are less than one in magnitude.

However, if $\mathbf{B}$ is not symmetric, like in our case, then we do not know if the eigenvectors form a basis and therefore it is hard to study the stability of the CN method using this technique. One could turn to the von Neumann method which only works for constant coefficient equations and hence requires the change of variable $x = \log(S)$.

In order to show the stability of the operator $\mathbf{B}$, we could study the boundedness of the powers of $\mathbf{B}$ (see [11]). We say that $\mathbf{B}$ is

- strictly stable if $\|\mathbf{B}^\nu\| \leq 1 \qquad \forall \nu, n > 0,$
- strongly stable if $\|\mathbf{B}^\nu\| \leq C \qquad \forall \nu, n > 0,$
- algebraically stable if $\|\mathbf{B}^\nu\| \leq C n^\alpha \nu^\beta$, where C, α, β are constant independent of ν and n. Here, n is the dimension of $\mathbf{B}$. Algebraic stability is obviously a weaker condition than either strict or strong stability. According to Lax Equivalence Theorem, strong stability is a necessary and sufficient condition for convergence for all initial data. Algebraic stability yields convergence only for certain initial data.

Under quite general conditions, it has been shown in [11] that the CN method is algebraically stable in the sense that it satisfies $\|\mathbf{B}^\nu\|_\infty \leq C n^{1/2}$. Here, $\alpha = 1/2$ and $\beta = 0$. Since both α, β are less than one, we have mild growth of errors in this case. In our case, since the matrix $\mathbf{I} + \theta \Delta \boldsymbol{\tau}_\nu \mathbf{M}^\nu$ is an $\mathcal{M}$-matrix, it is straightforward to show, using maximum principles, that $\|\mathbf{V}^{\nu+1}\|_\infty \leq \|\mathbf{V}^\nu\|_\infty$, and hence strict stability follows (see [38]).

Even though the CN method is unconditionally stable in the above sense, it is a non-dissipative scheme, meaning that high-frequency oscillations may persist for many timesteps, especially with non-smooth initial conditions. In our case, the initial condition is a hockey-

stick function with discontinuity in the first derivative at the strike and hence we would expect that oscillations can be introduced in the solution. Even though these oscillations may be small or even unnoticeable when we look at the option values, they could become magnified when computing delta and gamma. Experimental results in Chapter 6 agree with these predictions. Note that these oscillations are not the result of instability, since the oscillations do not grow excessively. Stability only deals with the limit as the $\nu_{\max}$ and n tend to infinity. Let us informally investigate a bit more on this issue to see why oscillations are possible with the CN method.

First, we consider the θ-timestepping with $\theta = 1$, i.e the implicit method. Substituting $\theta = 1$ into (3.20), we have

$$\mathbf{V}_i^{\nu+1} + \Delta\boldsymbol{\tau}_\nu(\mathbf{m}_{i,i-1}^{\nu}\mathbf{V}_{i-1}^{\nu+1} + \mathbf{m}_{i,i}^{\nu}\mathbf{V}_i^{\nu+1} + \mathbf{m}_{i,i+1}^{\nu}\mathbf{V}_{i+1}^{\nu+1}) = \mathbf{V}_i^{\nu},$$

or equivalently,

$$(1 + \Delta\boldsymbol{\tau}_\nu\mathbf{m}_{i,i}^{\nu})\mathbf{V}_i^{\nu+1} = \mathbf{V}_i^{\nu} - \Delta\boldsymbol{\tau}_\nu\mathbf{m}_{i,i-1}^{\nu}\mathbf{V}_{i-1}^{\nu+1} - \Delta\boldsymbol{\tau}_\nu\mathbf{m}_{i,i+1}^{\nu}\mathbf{V}_{i+1}^{\nu+1}. \tag{5.4}$$

Notice that $\mathbf{m}_{i,i-1}^{\nu}$ and $\mathbf{m}_{i,i+1}^{\nu}$ are non-positive, and $\mathbf{m}_{i,i}^{\nu}$ are positive. Taking into account that $\Delta\boldsymbol{\tau}_\nu$ is positive, we have that $-\Delta\boldsymbol{\tau}_\nu\mathbf{m}_{i,i-1}^{\nu}$ and $-\Delta\boldsymbol{\tau}_\nu\mathbf{m}_{i,i+1}^{\nu}$ are non-negative. It is then straightforward to prove that $\|V^{\nu+1}\| \leq \|V^{\nu}\|$. By the maximum principle, we can conclude that the implicit method is unconditionally stable, that is, there is no restriction on the timestep sizes. The details of the proof are straightforward and can readily be found in the literature. However, what we want to investigate here is the spurious behaviors of the solution. Note that from (5.4) we have,

$$\begin{aligned}\mathbf{V}_i^{\nu+1} &= \frac{\mathbf{V}_i^{\nu} - \Delta\boldsymbol{\tau}_\nu\mathbf{m}_{i,i-1}^{\nu}\mathbf{V}_{i-1}^{\nu+1} - \Delta\boldsymbol{\tau}_\nu\mathbf{m}_{i,i+1}^{\nu}\mathbf{V}_{i+1}^{\nu+1}}{1 + \Delta\boldsymbol{\tau}_\nu\mathbf{m}_{i,i}^{\nu}} \\ &\leq \Big[\frac{1 - \Delta\boldsymbol{\tau}_\nu(\mathbf{m}_{i,i-1}^{\nu} + \mathbf{m}_{i,i+1}^{\nu})}{1 + \Delta\boldsymbol{\tau}_\nu\mathbf{m}_{i,i}^{\nu}}\Big]\mathbf{V}_i^{\nu,\max},\end{aligned} \tag{5.5}$$

and

$$\mathbf{V}_i^{\nu+1} \geq \Big[\frac{1 - \Delta\boldsymbol{\tau}_\nu(\mathbf{m}_{i,i-1}^{\nu} + \mathbf{m}_{i,i+1}^{\nu})}{1 + \Delta\boldsymbol{\tau}_\nu\mathbf{m}_{i,i}^{\nu}}\Big]\mathbf{V}_i^{\nu,\min}, \tag{5.6}$$

with $\mathbf{V}_i^{\nu,\min} = \min\{\mathbf{V}_{i-1}^{\nu+1}, \mathbf{V}_i^{\nu}, \mathbf{V}_{i+1}^{\nu+1}\}$ and $\mathbf{V}_i^{\nu,\max} = \max\{\mathbf{V}_{i-1}^{\nu+1}, \mathbf{V}_i^{\nu}, \mathbf{V}_{i+1}^{\nu+1}\}$. Note that the quantity $\frac{1-\Delta\boldsymbol{\tau}_\nu(\mathbf{m}_{i,i-1}^{\nu}+\mathbf{m}_{i,i+1}^{\nu})}{1+\Delta\boldsymbol{\tau}_\nu\mathbf{m}_{i,i}^{\nu}}$ is positive and less than 1 due to the fact that $\mathbf{A}$ is strictly diagonally dominant. Relations (5.5) and (5.6) indicate that values of $\mathbf{V}_i^{\nu+1}$ are bounded by the maximum and minimum values of all neighboring nodes and hence spurious behaviors cannot occur in the discrete solution.

Now, let us consider the CN method. Substituting $\theta = \frac{1}{2}$ into (3.20), we obtain

$$\begin{aligned}\mathbf{V}_i^{\nu+1} + \frac{\Delta\boldsymbol{\tau}_\nu}{2}(\mathbf{m}_{i,i-1}^{\nu}\mathbf{V}_{i-1}^{\nu+1} + \mathbf{m}_{i,i}^{\nu}\mathbf{V}_i^{\nu+1} + \mathbf{m}_{i,i+1}^{\nu}\mathbf{V}_{i+1}^{\nu+1}) \\ = \mathbf{V}_i^{\nu} - \frac{\Delta\boldsymbol{\tau}_\nu}{2}(\mathbf{m}_{i,i-1}^{\nu}\mathbf{V}_{i-1}^{\nu} + \mathbf{m}_{i,i}^{\nu}\mathbf{V}_i^{\nu} + \mathbf{m}_{i,i+1}^{\nu}\mathbf{V}_{i+1}^{\nu}),\end{aligned}$$

or equivalently,

$$\begin{aligned}(1 + \frac{\Delta\boldsymbol{\tau}_\nu}{2}\mathbf{m}_{i,i}^{\nu})\mathbf{V}_i^{\nu+1} = -\frac{\Delta\boldsymbol{\tau}_\nu}{2}\mathbf{m}_{i,i-1}^{\nu}(\mathbf{V}_{i-1}^{\nu} + \mathbf{V}_{i-1}^{\nu+1}) - \frac{\Delta\boldsymbol{\tau}_\nu}{2}\mathbf{m}_{i,i+1}^{\nu}(\mathbf{V}_{i+1}^{\nu} + \mathbf{V}_{i+1}^{\nu+1}) \\ + (1 - \frac{\Delta\boldsymbol{\tau}_\nu}{2}\mathbf{m}_{i,i}^{\nu})\mathbf{V}_i^{\nu}.\end{aligned} \tag{5.7}$$

From (5.7), it is obvious that we cannot use the same arguments with the implicit method to bound the minimum and maximum discrete values of $\mathbf{V}_i^{\nu+1}$ by those of neighboring nodes, since the coefficient $1 - \frac{\Delta\boldsymbol{\tau}_\nu}{2}\mathbf{m}_{i,i}^{\nu}$ can be negative unless the timestep size satisfies the condition

$$\Delta\boldsymbol{\tau}_\nu \leq \frac{2}{\mathbf{m}_{i,i}^{\nu}}. \tag{5.8}$$

From here, we can see why spurious oscillations *may* occur when the CN method is used if condition (5.8) is not satisfied. If we impose condition (5.8), then oscillations are guaranteed not to occur. Note that this is *not* a necessary condition but a sufficient one: this limitation on timestep sizes is double the limitation on timestep sizes for the explicit method. However, taking into account that $\mathbf{m}_{i,i}^{\nu} \simeq \mathcal{O}(\frac{1}{h^2})$, this limitation could be severe, especially in cases of an option with very long maturity. In such a case, since we have to use very small timestep sizes, the method takes many timesteps and hence very long computation time.

We would like to use the CN method as much as possible due to its second order of convergence in time ($\mathcal{O}(\Delta\boldsymbol{\tau}_\nu)^2$). In case of European and American options, the CN method can cause difficulties such as lower order of convergence (lower than quadratic convergence as

theoretically expected) and highly oscillatory delta and gamma in the region near the strike. In order to restore the order of convergence and suppress the oscillations, we appeal to the Rannacher smoothing technique [28]: for the first *two* steps, we employ the implicit method ($\theta = 1$) and we switch to CN method for the remaining timesteps. In addition, we adopt another smoothing technique suggested in [27], that is, we choose the grid so that there is always a node at the strike E (the kink point) at each timestep. One consideration is that adaptive grids may not have a node at the strike. In this case, noticing that the option values behave linearly in the area towards the left boundary of the domain, we propose to move one point from this area to line up with the strike price. Another way to handle the convergence and oscillation issues is presented in [20] where the ratio between the timestep sizes and the spatial stepsize are kept constant and small enough.

Note that the Rannacher smoothing and having a node at the strike do not guarantee to preclude oscillation but we are certain to get second order of convergence (second order convergence does not imply no oscillations). As we shall see later in Chapter 6, in practice these methods work very well for European and American options.

Chapter 6

Numerical Results

In this chapter, we illustrate results from adaptive mesh methods for certain option pricing problems. Our primary focus is adaptive mesh methods for American option pricing described in Chapter 4. However, we start by providing numerical results of adaptive mesh methods applied to European option pricing problem to show the effectiveness of the methods on problems without constraints. We then present experimental results of the adaptive techniques on pricing of American put options.

For convenience, we denote a grid with n nodes and ν timesteps by "$n \times \nu$ grid". For example, a "2560×2560 grid" means a grid with 2560 spatial grid points and 2560 timesteps. Notations used and the statistics collected in this chapter include:

"Nodes": the number of spatial grid points.

"Timesteps": the total number of timesteps in the time dimension.

"Value": numerical value of the option at $S = E$ unless otherwise stated.

"Change": the difference in the numerical value from coarser grid.

"Ratio": the ratio of the changes on successive grids.

"Adapt. #": the number of times the adaptive techniques were invoked over all timesteps.

"Avg. ω": the average value of (P)SOR relaxation factor ω over all timesteps.

"Min." ("Max."): the minimum (maximum) number of iterations required by an iterative

Table 6.1: Model parameters for European and American options

Parameter	Value
Time to expiry T	0.25 (years)
Interest rate r	10% (0.1)
Exercise price E	100
Volatility σ	80% (0.8)
Tolerance ϵ	$1.e-07$

method over all timesteps.

"Total" ("Avg."): the total (average) number of iterations required by an iterative method over all timesteps.

For purpose of comparison of the total number of iterations between two iterative methods, Method-1 and Method-2, we define "Percentage Saved" to be

$$\frac{\text{Method-1 Total} - \text{Method-2 Total}}{\text{Method-1 Total}} \times 100\%.$$

This quantity represents the percentage of iterations saved when Method-2 is used instead of Method-1.

As we mentioned earlier, the infinite spatial domain $[0, \infty)$ must be truncated down to $[0, \mathbf{S}_{\max}]$ and the boundary condition at $S = \infty$ is replaced by the boundary condition at $S = \mathbf{S}_{\max}$. Values for $S = \mathbf{S}_{\max}$ must be chosen with care so that the effects of the truncation is minimized. For example, in case of European options, it is suggested in [18] that, with the exercise price E, $\mathbf{S}_{\max}$ can be selected as

$$\mathbf{S}_{\max} \geq E \exp(\sqrt{2\sigma^2 T |\ln(tol)|}),$$

where tol is user-defined tolerance. As discussed in [38], this choice of $\mathbf{S}_{\max}$ is somewhat pessimistic. An alternative suggested in [38] is

$$\mathbf{S}_{\max} \geq E \exp((r - \frac{\sigma^2}{2})T + \sigma\mu\sqrt{T}.$$

With $\mu = 3$, this alternative is a typical choice for European options while with μ of higher value can be used for more complicated problems.

In our experiment, we choose

$$\mathbf{S}_{\max} = \max\left(\phi E, E\exp((r - \frac{\sigma^2}{2})T + \sigma\mu\sqrt{T})\right), \tag{6.1}$$

with $\phi = 5, \mu = 3$, and $tol = 10^{-7}$. Note that the choice of $\phi = 5$ has been used in [21] for a similar American put option, whereas in [20], $\phi = 4$ is used.

With model parameters given in Table 6.1, we have

$$\begin{aligned}\mathbf{S}_{\max} &= \max\left(\phi E, E\exp((r - \frac{\sigma^2}{2})T + \sigma\mu\sqrt{T})\right)\\ &= \max\left(500, 314\right).\end{aligned}$$

We choose $\mathbf{S}_{\max} = 500$ and with this choice the strike price E is one of the grid points if uniform grids are used.

6.1 European Options

For European option pricing problems on one underlying stock and with constant interest rate and volatility, analytical formulas for option values and corresponding deltas and gammas are available (see [37]). We can compute their exact solutions and thus are able to compare the errors of their numerical solutions, giving us an objective evaluation of different numerical methods. Linear systems from European option pricing can be solved using standard LU-factorization. In MATLAB, the backslash "\" operator is a very convenient tool to solve such linear systems.

6.1.1 Delta and Gamma Valuations

For an option pricing problem, it is of practical importance to determine the delta and gamma for hedging purposes. As we mentioned earlier, although the values of the options appear

smooth, spurious oscillations can be introduced in the delta and magnified in the gamma. In this subsection, we first examine the effectiveness of the Rannacher smoothing technique as a remedy to these problems for both uniform and adaptive mesh methods. We then study the accuracy of the adaptive mesh methods on evaluating delta and gamma. Note that

$$\text{delta} = \frac{\partial V}{\partial S}, \qquad \text{gamma} = \frac{\partial^2 V}{\partial S^2},$$

and thus they can be numerically evaluated using the FD formulas (2.5) and (2.6).

Ranacher Smoothing Technique

As we noted earlier, the Rannacher smoothing technique guarantees the second order of convergence of the numerical methods, not the preclusion of oscillations. However, as we shall see later, in practice this smoothing technique works well. For illustrative purposes, we run experiments with small number of timesteps to exaggerate oscillatory behaviors of delta and gamma in the area around the strike.

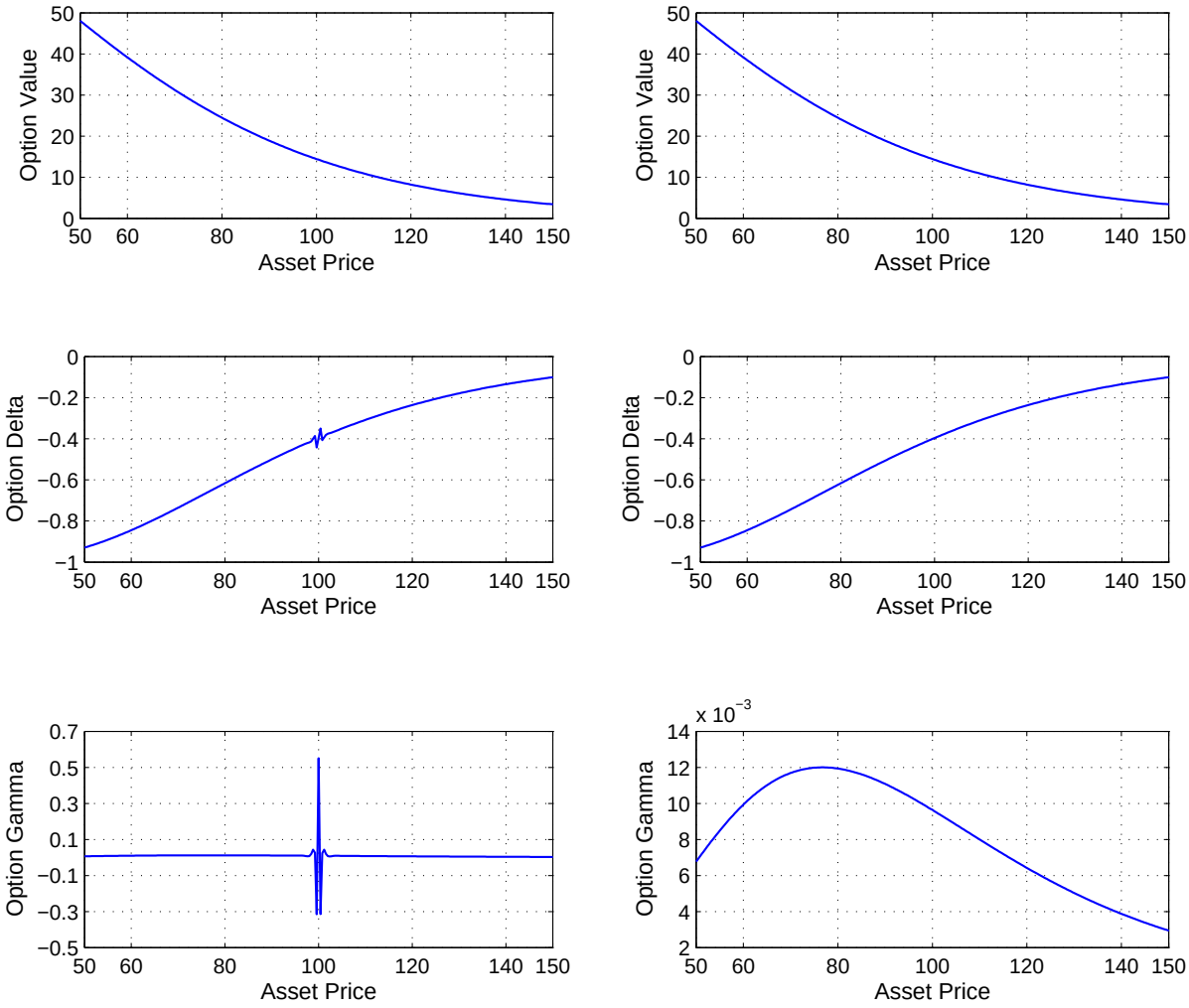
The numerical results of the European put option value, delta, and gamma obtained by uniform mesh methods are presented in Table 6.2. As expected, numerical results for the option value with no smoothing exhibit only first order of convergence (ratio 2) as the grids are refined. In addition, although numerical results of the option value are convergent, oscillations are introduced into delta (note the first three numerical values of delta and their corresponding orders of convergence). More seriously, the numerical values of gamma are divergent away from the analytical value 0.00963579.

The reason for the poor convergence and erratic behaviors of non-smoothed runs can be explained by examining the plots the option value, delta, and gamma as shown in the left side of Figure 6.1. Although the option value appears smooth, spurious oscillations are present in delta and magnified in gamma, which explains inaccurate numerical results for gamma. These numerical experiments indicate that results with no smoothing could be erratic and unreliable.

Table 6.2: Experimental results for the European put option at $S = 100$ obtained by **uniform mesh methods**. The analytical results of the option value, delta, and gamma are 14.45190585, -0.39646799, and 0.00963579, respectively.

Nodes	Time-steps	No Rannacher Smoothing			Rannacher Smoothing		
		Value	Change	Ratio	Value	Change	Ratio
		option value					
80	4	13.99245349			14.19003389		
160	8	14.22162842	0.22917492		14.38679335	0.19675946	
320	16	14.33708514	0.11545673	2.0	14.43536312	0.04856977	4.1
640	32	14.39463936	0.05755422	2.0	14.44771420	0.01235108	3.9
1280	64	14.42331694	0.02867758	2.0	14.45084925	0.00313505	3.9
2560	128	14.43762354	0.01430660	2.0	14.45164050	0.00079125	4.0
		delta					
80	4	-0.39638078			-0.39870701		
160	8	-0.39631553	0.00006525		-0.39701854	0.00168848	
320	16	-0.39636014	-0.00004462	-1.5	-0.39660691	0.00041163	4.1
640	32	-0.39640556	-0.00004542	1.0	-0.39650305	0.00010385	4.0
1280	64	-0.39643457	-0.00002901	1.6	-0.39647681	0.00002624	4.0
2560	128	-0.39645072	-0.00001615	1.8	-0.39647020	0.00000661	4.0
		gamma					
80	4	0.04296038			0.01034088		
160	8	0.07702208	0.03406169		0.00985280	-0.00048808	
320	16	0.14475634	0.06773426	0.5	0.00971313	-0.00013967	3.5
640	32	0.28004460	0.13528825	0.5	0.00966665	-0.00004648	3.0
1280	64	0.55053557	0.27049097	0.5	0.00964924	-0.00001742	2.7
2560	128	1.09147600	0.54094043	0.5	0.00964201	-0.00000723	2.4

The same problem is run with the Rannacher smoothing and numerical results are presented in Table 6.2. Plots of smoothed runs are shown on the right side of Figure 6.1. The oscillations in delta and gamma have disappeared.



(a) No Rannacher Smoothing

(b) Rannacher Smoothing

Figure 6.1: European put valued numerically using Crank-Nicolson timestepping on a **uniform** 1280×60 grid.

We now examine the Rannacher smoothing in the context of adaptive mesh techniques. The numerical results are presented in Table 6.3. Non-smoothed runs yield numerical results of extremely erratic behaviors and are very unreliable. The delta and gamma are considerably worse than those obtained by uniform mesh methods without smoothing. For example, numerical values of gamma explode as the grids are refined. However, with the Rannacher smoothing, the adaptive mesh methods produce results of higher accuracy than those obtained by uniform mesh methods with smoothing. The convergence rates are stable and there are no signs of oscillations.

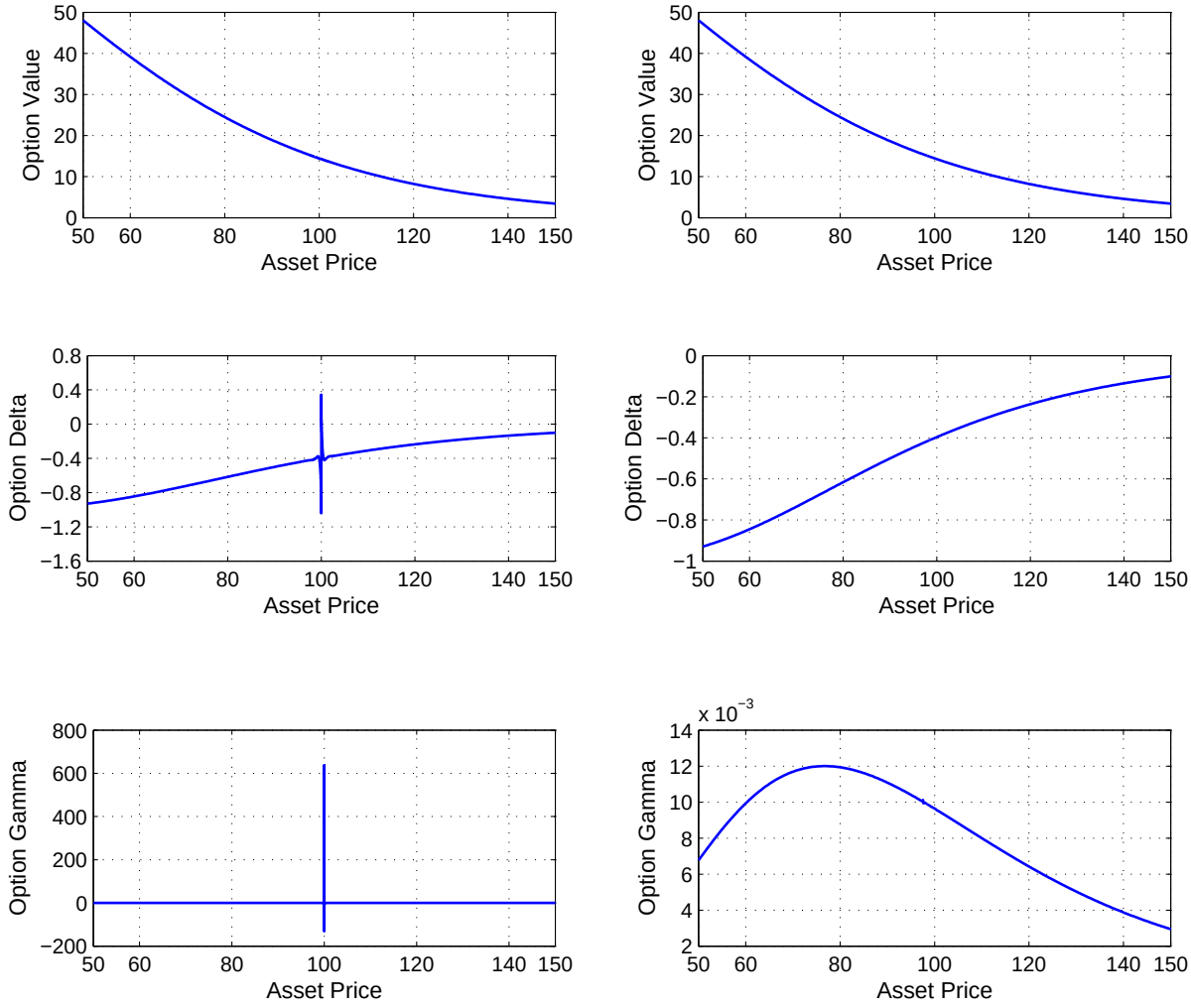
The plots of option value, delta, and gamma evaluated using adaptive mesh methods are presented in Figure 6.2. These plots agree with what is observed from other numerical results. For instance, when the Rannacher smoothing is not used, the plot of delta has a spike in the area near the strike and this spike is seriously magnified in the plot of gamma, which explains the erratic results of delta and gamma in Table 6.3. Note that in the plot of gamma, the vertical axis scale is much coarser than that of other plots, in order to make the magnitude of the oscillations in gamma more visible. With the Rannacher smoothing, the plots of delta and gamma are oscillation-free which explains the improved results in Table 6.3.

Experiments with both uniform and adaptive mesh methods show the effectiveness of the Rannacher smoothing. Note that, in Chapter 5, we discuss the effects of non-smooth payoff on the accuracy of the numerical solutions. When we apply constant timesteps with the CN method, there are two remedies suggested in [14]: (i) Rannacher smoothing, i.e. taking a small number of fully implicit timesteps (we have used *two* in our experiments) after each non-smooth initial state, then using CN timesteping thereafter; and (ii) if the initial condition is a continuous piecewise linear function, such as the payoff for a European or American option, we select the grid points so that the strike is one of the grid points. In all non-smoothed runs, we put the strike at one of the grid points. If we employ (ii) without employing (i), the solutions still exhibit linear convergence rate and delta and gamma are highly oscillatory (in Tables 6.2 and 6.3). However, using (i) eliminates all oscillations and restores quadratic convergence of

the solutions. These experimental results agree with those in [20]. We conclude that (ii) is not as useful as (i) in dealing with oscillations.

Table 6.3: Experimental results for the European put option at $S = 100$ obtained by **adaptive mesh methods**. The analytical results of the option value, delta, and gamma are 14.45190585, -0.39646799, and 0.00963579, respectively.

Nodes	Time-steps	No Rannacher Smoothing			Rannacher Smoothing		
		Value	Change	Ratio	Value	Change	Ratio
		option value					
80	4	13.49720284			14.23600445		
160	8	13.96202101	0.46481817		14.39753530	0.16153084	
320	16	14.20464193	0.24262091	1.9	14.43810113	0.04056583	4.0
640	32	14.32774073	0.12309880	2.0	14.44841082	0.01030969	3.9
1280	64	14.38969623	0.06195550	2.0	14.45102565	0.00261483	3.9
2560	128	14.42077137	0.03107515	2.0	14.45168474	0.00065909	4.0
		delta					
80	4	-0.03699890			-0.39799018		
160	16	-0.41784154	-0.38084264		-0.39688787	0.00110231	
320	32	-0.26516071	0.15268083	-2.5	-0.39656957	0.00031829	3.5
640	64	-0.35618995	-0.09102925	-1.7	-0.39649289	0.00007669	4.2
1280	128	-0.52281268	-0.16662272	0.5	-0.39647391	0.00001898	4.0
2560	256	-0.67591263	-0.15309995	1.1	-0.39646967	0.00000424	4.5
		gamma					
80	4	3.53180179			0.01023244		
160	16	14.92104252	11.38924074		0.00985197	-0.00048808	
320	32	54.14855677	39.22751424	0.3	0.00969283	-0.00013967	2.4
640	64	195.51518416	141.36662740	0.3	0.00967086	-0.00004648	7.2
1280	128	636.78231621	441.26713204	0.3	0.00964722	-0.00001742	0.9
2560	256	2113.78720093	1477.00488472	0.3	0.00963991	-0.00000723	3.2



(a) No Rannacher Smoothing (b) Rannacher Smoothing

Figure 6.2: European put valued numerically using Crank-Nicolson timestepping with **adaptive mesh methods** on a 1280×64 grid.

Adaptive Mesh Methods

We now examine the accuracy of the adaptive mesh methods in calculating the delta and gamma. In Table 6.4, numerical results of European put delta and gamma are presented. In these tests adaptive mesh methods seem to provide more accurate delta and gamma than those given by uniform mesh methods. These more accurate results are obtained with small number of adaptivity calls compared to the total number of timesteps.

Table 6.4: Experimental results for delta and gamma of the European put option at $S = 100$. Rannacher smoothing is used. The analytical values of delta and gamma are -0.39646799 and 0.00963579, respectively.

Nodes	Time steps	Uniform			Adaptive			
		Value	Change	Ratio	Value	Change	Ratio	Adapt. #
		delta						
20	80	-0.41153152			-0.40401549			5
40	160	-0.39973218	0.01179934		-0.39806386	0.00595163		6
80	320	-0.39726507	0.00246711	4.8	-0.39663459	0.00142927	4.2	10
160	640	-0.39666616	0.00059891	4.1	-0.39651054	0.00012405	11.5	15
320	1280	-0.39651747	0.00014870	4.0	-0.39647798	0.00003256	3.8	18
640	2560	-0.39648036	0.00003711	4.0	-0.39647045	0.00000753	4.3	22
1280	5120	-0.39647108	0.00000927	4.0	-0.39646857	0.00000188	4.0	33
		gamma						
20	80	0.01036238			0.00987314			5
40	160	0.00977775	-0.00058463		0.00966562	-0.00020752		6
80	320	0.00966991	-0.00010784	5.4	0.00963787	-0.00002774	7.5	10
160	640	0.00964424	-0.00002567	4.2	0.00963638	-0.00000149	18.6	15
320	1280	0.00963790	-0.00000634	4.0	0.00963590	-0.00000048	3.1	18
640	2560	0.00963632	-0.00000158	4.0	0.00963582	-0.00000008	6.1	22
1280	5120	0.00963592	-0.00000040	4.0	0.00963580	-0.00000002	4.0	33

6.1.2 Option Valuation

Since analytical solutions of European option problems are available, we can easily compare the accuracy of the adaptive mesh methods with uniform mesh methods.

In Tables 6.5 and 6.6, numerical results of the European call and put are presented, respectively. It is obvious that the adaptive mesh methods provide much more accurate results with moderate additional cost. Let us take the 2560×2560 grid as an example. On this grid, numerical results obtained by uniform mesh methods for both European call and put options are only 5-digit accurate. However, with adaptive mesh techniques, we can obtain numerical solutions of 8-digit and 9-digit accuracy for the call and put, respectively. This is significant improvement in terms of accuracy. More importantly, these accurate results are obtained by invoking the adaptive technique only a small number of times compared to the total number of time steps: 20 and 23 times out of a total number of timesteps of 2560 for European call and put, respectively.

In Table 6.7 we compare the relative errors between the adaptive mesh methods and uniform mesh methods. Relative errors are defined as

$$\text{Relative error} = \frac{|\text{approximate value} - \text{true value}|}{|\text{true value}|} \times 100\%.$$

These results again show that adaptive mesh techniques outperform uniform mesh methods in terms of accuracy.

In Figure 6.3, the true errors of the European call and put are plotted versus the asset price S. As we expected, the errors on the uniform grid are very large in the region around the strike and small elsewhere. The errors of the adaptive mesh methods are much smaller in the region around the strike price E and more well-distributed than those obtained by the uniform mesh method. This is clearly what error equidistribution is intended to achieve.

Table 6.5: Experimental results for the European call option at $S = 100$. Rannacher smoothing is used. The analytical value of the call is 16.92091465.

	Time	Uniform			Adaptive			
Nodes	steps	Value	Change	Ratio	Value	Change	Ratio	Adapt. #
40	40	16.73573950			16.86397895			3
80	80	16.87521977	0.13948027		16.91045931	0.04648037		4
160	160	16.90952630	0.03430653	4.1	16.91968136	0.00922205	5.0	7
320	320	16.91806973	0.00854342	4.0	16.92071007	0.00102871	9.0	8
640	640	16.92020355	0.00213383	4.0	16.92089772	0.00018765	5.5	11
1280	1280	16.92073688	0.00053333	4.0	16.92091285	0.00001514	12.4	14
2560	2560	16.92087021	0.00013333	4.0	16.92091453	0.00000168	9.0	20

Table 6.6: Experimental results for the European put option at $S = 100$. Rannacher smoothing is used. The analytical value of the put is 14.45190585.

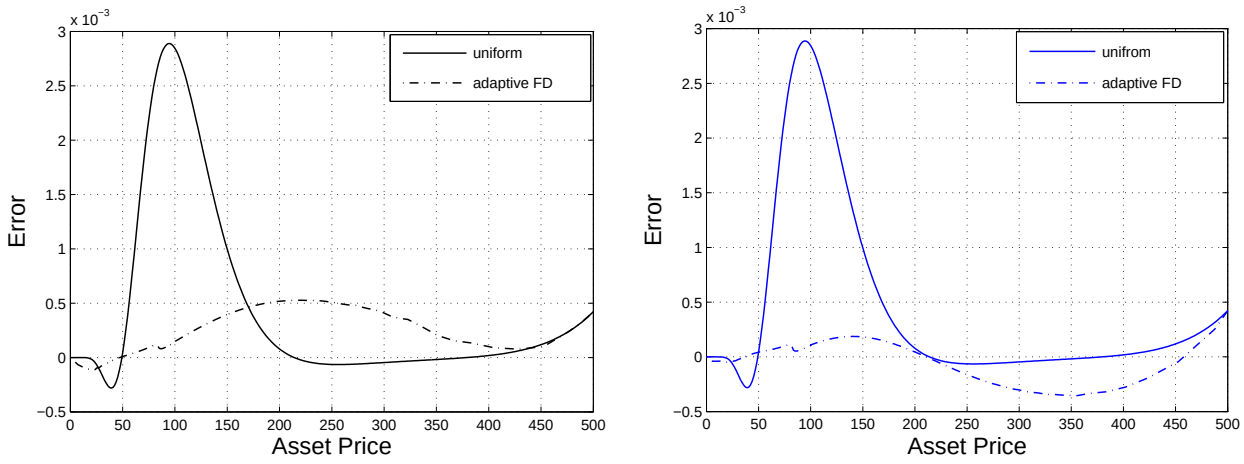
	Time	Uniform			Adaptive			
Nodes	steps	Value	Change	Ratio	Value	Change	Ratio	Adapt. #
40	40	14.26674966			14.41562094			4
80	80	14.40621571	0.13946605		14.44733426	0.03171333		5
160	160	14.44051869	0.03430298	4.1	14.45093759	0.00360332	8.8	6
320	320	14.44906122	0.00854253	4.0	14.45180252	0.00086494	4.2	9
640	640	14.45119483	0.00213360	4.0	14.45189405	0.00009153	9.5	13
1280	1280	14.45172811	0.00053328	4.0	14.45190439	0.00001034	8.9	19
2560	2560	14.45186142	0.00013331	4.0	14.45190584	0.00000145	7.2	23

Table 6.7: Errors of uniform and adaptive mesh methods applied to the European call option. The analytical value of the call is 16.92091465. Approximate values are from Table 6.5.

Nodes	Time steps	Uniform			Adaptive		
		Value	Absolute Error	Relative Error	Value	Absolute Error	Relative Error
40	40	16.73573950	0.18517515	1.0944%	16.86397895	0.05693570	0.3365%
80	80	16.87521977	0.04569488	0.2700%	16.91045931	0.01045534	0.0618%
160	160	16.90952630	0.01138835	0.0673%	16.91968136	0.00123329	0.0073%
320	320	16.91806973	0.00284492	0.0168%	16.92071007	0.00020458	0.0012%
640	640	16.92020355	0.00071110	0.0042%	16.92089772	0.00001693	0.0001%
1280	1280	16.92073688	0.00017777	0.0011%	16.92091285	0.00000180	0.0000%
2560	2560	16.92087021	0.00004444	0.0003%	16.92091453	0.00000012	0.0000%

Table 6.8: Errors of uniform and adaptive mesh methods applied to the European put option. The analytical value of the put is 14.45190585. Approximate values are from Table 6.6.

Nodes	Time steps	Uniform			Adaptive		
		Value	Absolute Error	Relative Error	Value	Absolute Error	Relative Error
40	40	14.26674966	0.18515619	1.2812%	14.41562094	0.03628491	0.2511%
80	80	14.40621571	0.04569014	0.3162%	14.44733426	0.00457159	0.0316%
160	160	14.44051869	0.01138716	0.0788%	14.45093759	0.00096826	0.0067%
320	320	14.44906122	0.00284463	0.0197%	14.45180252	0.00010333	0.0007%
640	640	14.45119483	0.00071102	0.0049%	14.45189405	0.00001180	0.0001%
1280	1280	14.45172811	0.00017774	0.0012%	14.45190439	0.00000146	0.0000%
2560	2560	14.45186142	0.00004443	0.0003%	14.45190584	0.00000001	0.0000%



(a) European call (b) European put

Figure 6.3: Observed error distribution of the European options on a 320×320 grid.

6.1.3 Initial Guesses and the SOR method

For European option pricing, the matrix problem at each timestep can be solved using a direct method (LU factorization, for instance). However, we would like to study the effects of the initial guess on the convergence rate of iterative methods in the context of non-constrained matrix problems such as those associated with European option pricing. In particular, we would like to study the convergence of the SOR method using two different initial guesses at each timesteps. As we mentioned earlier, for the iterative numerical solution at timestep $\nu + 1$, there are two initial guesses whose effects on convergence rate we would like to examine. The first initial guess is the numerical solution at the previous timestep,

$$\mathbf{V}^{\nu+1,(0)} = \mathbf{V}^{\nu},$$

as mentioned in (3.31) on page 25. The other one is based on the extrapolation of numerical solutions of the two previous timesteps,

$$\mathbf{V}^{\nu+1,(0)} = \frac{(\Delta\boldsymbol{\tau}_{\nu} + \Delta\boldsymbol{\tau}_{\nu-1})}{\Delta\boldsymbol{\tau}_{\nu-1}}\mathbf{V}^{\nu} - \frac{\Delta\boldsymbol{\tau}_{\nu}}{\Delta\boldsymbol{\tau}_{\nu-1}}\mathbf{V}^{\nu-1},$$

as mentioned in (3.32), or equivalently

$$\mathbf{V}^{\nu+1,(0)} = 2\mathbf{V}^{\nu} - \mathbf{V}^{\nu-1}$$

with constant timesteps. The two corresponding SOR methods are referred to as SOR-1 and SOR-2, respectively. As mentioned earlier, we adopt a technique given in [37] to dynamically determine the relaxation factor ω at each time step.

Table 6.9: Experimental results for the European call option at $S = 100$ obtained on uniform grids using SOR-1. The analytical value of the call is 16.92091465.

Nodes	Time steps	Value	Change	Ratio	Avg. ω	No. Iters			
						Min.	Max.	Total	Avg.
20	80	16.13223086			1.02	7	11	572	7.17
40	160	16.73676514	0.60453428		1.01	9	12	1445	9.03
80	320	16.87547343	0.13870829	4.4	1.10	11	13	3684	11.51
160	640	16.90958974	0.03411632	4.1	1.20	14	19	9290	14.52
320	1280	16.91808561	0.00849587	4.0	1.32	19	30	24990	19.52
640	2560	16.92020753	0.00212192	4.0	1.45	24	48	63454	24.79
1280	5120	16.92073788	0.00053035	4.0	1.55	31	77	162781	31.79

Table 6.10: Experimental results for the European call option at $S = 100$ obtained on uniform grids using SOR-2. The analytical value of the call is 16.92091465.

Nodes	Time steps	Value	Change	Ratio	Avg. ω	No. Iters			
						Min.	Max.	Total	Avg.
20	80	16.13223086			1.02	4	9	345	4.31
40	160	16.73676514	0.60453428		1.01	4	9	668	4.17
80	320	16.87547339	0.13870826	4.4	1.01	4	13	1320	4.13
160	640	16.90958953	0.03411614	4.1	1.01	4	19	2625	4.11
320	1280	16.91808489	0.00849536	4.0	1.01	3	30	5312	4.15
640	2560	16.92020885	0.00212396	4.0	1.13	2	48	9449	3.69
1280	5120	16.92073798	0.00052912	4.0	1.15	2	77	23987	4.68

The numerical results of the European call option obtained by SOR-1 and SOR-2 are presented in Tables 6.9 and 6.10, respectively. The numerical values of the call price from these two tables are almost identical and exhibit quadratic convergence to the analytical solution 16.92091465 as we expected. The average numbers of iterations per timesteps in Table 6.9 indicate that the convergence rate of SOR-1 deteriorates seriously as grids are refined. For instance, the average number of iterations increase by more than 300% as grids are refined from 20×80 to 1280×5120. However, it is interesting to notice that the SOR-2 method requires substantially smaller number of iterations than SOR-1 does and the convergence rate of SOR-2 remains almost the same as grids are refined. Comparisons in Table 6.11 indicate that SOR-2 is significantly more efficient than SOR-1, especially on finer grids.

From these results, we would expect that at each timestep the maximum in absolute value of initial residual (residual of an initial guess) of SOR-1 to be higher than those of SOR-2 (hence SOR-1 requires less iterations to converge). In addition, for an IVP solved by an iterative method like this case, we would normally expect that the (maximum) initial residuals will decrease as time evolves. Plots of the maximum initial residual at each timestep v.s. the timestep index of the two methods in semilogy scale are presented in Figure 6.4. These plots agree with our expectations. For each timestep, the maximum in absolute value of the initial residual of SOR-2 is smaller than that of SOR-1 and it shows decreasing trend with the increase of the timestep index. For SOR-1, the initial residual decreases at the beginning, which is expected, but once it hits a plateau, the initial residual does not decrease considerably as the initial residual of SOR-2 does.

With these numerical results, we would also expect the same behaviors from PSOR and the penalty methods used for American option pricing. In the subsequent sections, we would also like to present numerical results on the effects of these two initial guesses on the convergence rate of iterative methods for American option pricing using uniform and adaptive mesh methods.

Table 6.11: Iteration comparison between SOR-1 and SOR-2 on uniform grids for the European call. Numerical results and statistics are from Tables 6.9 and 6.10.

Nodes	Time steps	SOR-1		SOR-2		Percentage Saved
		Value	Total	Value	Total	
20	80	16.13223086	572	16.13223086	345	40%
40	160	16.73676514	1445	16.73676514	668	54%
80	320	16.87547343	3684	16.87547339	1320	64%
160	640	16.90958974	9290	16.90958953	2625	71%
320	1280	16.91808561	24990	16.91808489	5312	79%
640	2560	16.92020753	63454	16.92020885	9449	85%
1280	5120	16.92073788	162781	16.92073798	23987	86%

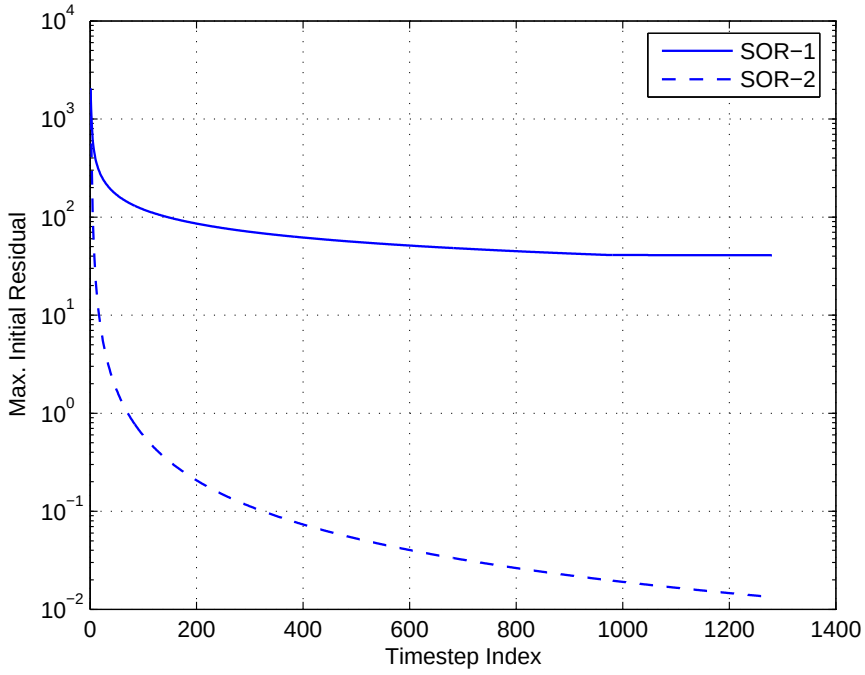


Figure 6.4: Maximum residuals of initial guess for SOR methods on the uniform 320×1280 grid.

6.2 American Options

Both PSOR and the penalty methods can be used to handle the free boundary problem from American option pricing.

In this section, we first present selected numerical results for American put option pricing using uniform mesh methods and later results obtained by adaptive mesh methods. We employ PSOR and the penalty methods to solve the associated constrained matrix problem. We then study and compare the efficiency of corresponding methods. Finally, we examine and compare the accuracy of the adaptive mesh methods in locating the free boundary point.

6.2.1 Uniform Mesh Methods Results

The purpose of this project is to study the accuracy and effectiveness of adaptive mesh methods in American option pricing. However, as we mentioned earlier, one issue we also want to investigate is the effects of initial guesses on the convergence rate of iterative methods. We have investigated this issue in the context of IVPs with non-constrained matrix problems at each timestep. The purpose of this section is to experimentally examine this issue in the context of constrained matrix problems using uniform mesh methods.

PSOR Method

It has been reported in the literature that the convergence of the PSOR method deteriorates greatly as the grids are refined. As we mentioned earlier, the convergence rate of PSOR depends strongly on the choice of relaxation parameter ω and the initial guess at each timestep. Experiments with European options indicate that the quality of the initial guess plays a role in the rate of convergence of an iterative method. In particular, SOR with initial guess (3.32) based on extrapolation

$$\mathbf{V}^{\nu+1,(0)} = \frac{(\Delta\boldsymbol{\tau}_\nu + \Delta\boldsymbol{\tau}_{\nu-1})}{\Delta\boldsymbol{\tau}_{\nu-1}}\mathbf{V}^{\nu} - \frac{\Delta\boldsymbol{\tau}_\nu}{\Delta\boldsymbol{\tau}_{\nu-1}}\mathbf{V}^{\nu-1},$$

or equivalently

$$\mathbf{V}^{\nu+1,(0)} = 2\mathbf{V}^{\nu} - \mathbf{V}^{\nu-1}$$

with constant timesteps, can greatly help reduce the deterioration as grids are refined and is considerably more efficient than SOR with initial guess (3.31)

$$\mathbf{V}^{\nu+1,(0)} = \mathbf{V}^{\nu}.$$

It is our expectation that PSOR applied to American options will behave similarly with these two initial guesses. PSOR with initial guesses (3.31) and (3.32) are hereafter referred to as PSOR-1 and PSOR-2, respectively. As in European option pricing, we follow the technique presented in [37] to dynamically determine ω at each time step.

In Tables 6.12 and 6.13, numerical results for the American put option obtained on uniform grids using PSOR-1 and PSOR-2 are presented, respectively. In both tables, we observe that the numerical results exhibit quadratic convergence as expected. Our numerical values show strong agreement with numerical solution 14.67882 from [14]. In addition, numerical results in both tables are almost identical, with differences from the 7th digit. These results indicate that both PSOR-1 and PSOR-2 converge to the same numerical solution.

Now we examine numerical results from Tables 6.12 and 6.13 more closely. The number of iterations required for convergence behaves similarly as in European case. For a particular grid, PSOR-2 requires considerably smaller total number of iterations than PSOR-1 does as we expected. The deterioration of the convergence rate of the PSOR-1 method increases as the grids are refined. For instance, the average number of iterations per timestep increases by about nearly 100% as the grids are refined from 20×80 to 1280×5120 (from 4.86 to 8.39). These deteriorating results agree with our expectations and with numerical results from [20] and [21]. On the contrary, numerical results from Table 6.13 indicate that the average number of iterations per timestep required by PSOR-2 does not increase as grids are refined. Thus, for American option pricing with uniform mesh methods, an initial guess based on extrapolation formula (3.32) helps reduce the deterioration of the convergence rate significantly.

Table 6.12: Experimental results for the American put option at $S = 100$ obtained with **uniform mesh methods** and **constant timesteps** using PSOR-1. Reference numerical solution from [14] is 14.67882.

Nodes	Time steps	Value	Change	Ratio	Avg. ω	No. Iters			
						Min.	Max.	Total	Avg.
20	80	13.83869378			1.02	4	11	389	4.86
40	160	14.47322178	0.63452800		1.02	4	11	793	4.96
80	320	14.62625295	0.15303118	4.1	1.04	4	11	1784	5.58
160	640	14.66532254	0.03906958	3.9	1.07	6	11	4152	6.49
320	1280	14.67541079	0.01008825	3.9	1.17	4	11	8878	6.94
640	2560	14.67799031	0.00257952	3.9	1.29	5	12	19454	7.60
1280	5120	14.67864721	0.00065690	3.9	1.42	5	18	42980	8.39

Table 6.13: Experimental results for the American put option at $S = 100$ obtained with **uniform mesh methods** and **constant timesteps** using using PSOR-2. Reference numerical solution from [14] is 14.67882.

Nodes	Time steps	Value	Change	Ratio	Avg. ω	No. Iters			
						Min.	Max.	Total	Avg.
20	80	13.83869377			1.02	4	9	341	4.26
40	160	14.47322179	0.63452801		1.01	4	9	663	4.16
80	320	14.62625312	0.15303133	4.1	1.01	4	10	1312	4.10
160	640	14.66532258	0.03906947	3.9	1.01	4	10	2613	4.08
320	1280	14.67541039	0.01008781	3.9	1.01	4	11	5312	4.15
640	2560	14.67799450	0.00258411	3.9	1.04	3	12	8919	3.48
1280	5120	14.67865142	0.00065692	3.9	1.18	2	18	21463	4.19

Table 6.14: Iteration comparison between PSOR-1 and PSOR-2 for **uniform mesh methods** and **constant timesteps**. Numerical results and statistics are from Tables 6.12 and 6.13.

Nodes	Time steps	PSOR-1		PSOR-2		Percentage Saved
		Value	Total	Value	Total	
20	80	13.83869378	389	13.83869377	341	12%
40	160	14.47322178	793	14.47322179	663	16%
80	320	14.62625295	1784	14.62625312	1312	26%
160	640	14.66532254	4152	14.66532258	2613	37%
320	1280	14.67541079	8878	14.67541039	5312	40%
640	2560	14.67799031	19454	14.67799450	8919	54%
1280	5120	14.67864721	42980	14.67865142	21463	50%

For comparison purposes, the total numbers of iterations required by PSOR-1 and PSOR-2 are listed in Table 6.14. We can see that as the timestep and mesh size are reduced, the percentage saved increases, making PSOR-2 more efficient than PSOR-1 on finer grids.

Since PSOR-2 requires a smaller number of iterations, we would expect that its maximum (in absolute value) initial residuals at each timestep to be smaller than that of PSOR-1. This behavior has been experimentally confirmed in the previous section on SOR applied to European option pricing problems. We compute the maximum initial residual at each timestep as follows. For an initial guess, we can roughly estimate the free boundary. As discusses earlier in Chapter 3 on page 15, the free boundary separates the domain into two regions, namely the continuation region and the stopping region. In each region, it is straight forward to estimate the maximum initial residual. The maximum initial residual over the whole domain can be approximated by the larger of the two maximum initial residuals. The plots of the maximum initial residuals of PSOR-1 and PSOR-2 versus timestep indices on a semilogy scale are presented in Figure 6.5 and the plots agree with our expectations. We can see that PSOR-2 has a

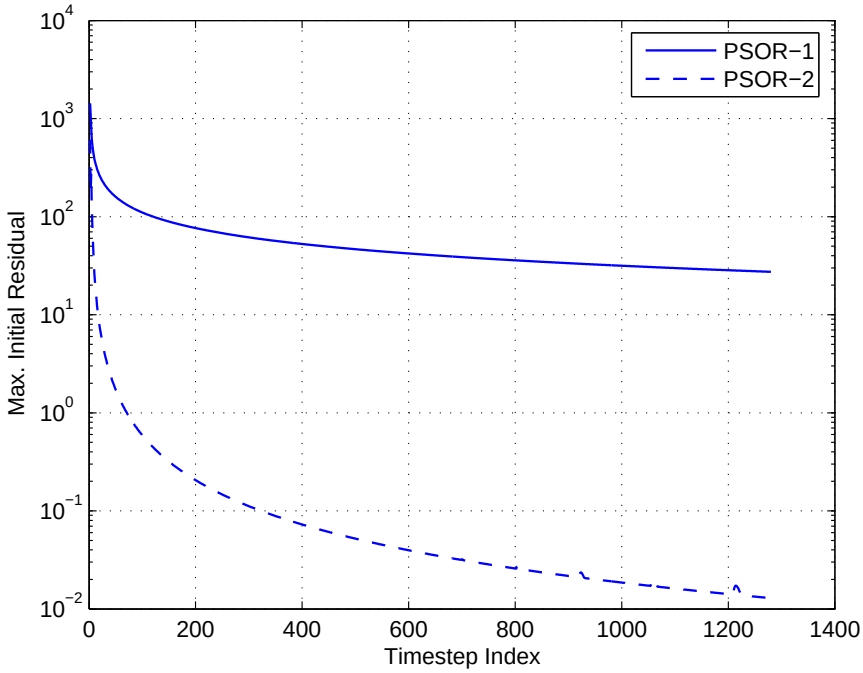


Figure 6.5: Maximum residuals of initial guess for PSOR methods on the uniform 320×1280 grid.

smaller initial residual at each timestep than PSOR-1. We can conclude that the initial guess based on extrapolation (3.32) can help to reduce significantly the convergence deterioration of PSOR as grids are refined. In the next section, we present numerical results obtained by adaptive mesh methods using the PSOR method. For efficiency reasons, we use PSOR-2 to solve the associated constrained matrix problems at each timestep in the adaptive mesh methods.

Penalty Method

In Tables 6.15 and 6.16, numerical results with the penalty method on uniform grids are presented. The penalty methods with initial guess (3.31) (previous timestep solution) and (3.32) (based on extrapolation) are referred to as PENALTY-1 and PENALTY-2, respectively. Our numerical values strongly agree with the numerical solution from [14] and with those obtained by PSOR presented in the previous section. We observe that the total number of iterations re-

quired by PENALTY-2 are smaller than those required by PENALTY-1, which is similar to the relation between the number of iterations required by (P)SOR-2 and (P)SOR-1. As we noted earlier, the penalty method converges in about one or two iterations, which is significantly faster than (P)SOR. Due to this fast convergence, the observed differences (or percentage saved) in number of iterations required for convergence between PENALTY-1 and PENALTY-2 is not as significant as those between (P)SOR-1 and (P)SOR-2.

Table 6.15: Experimental results for the American put option at $S = 100$ obtained with **uniform mesh methods** with **constant timesteps** using PENALTY-1. Reference numerical solution from [14] is 14.67882.

Nodes	Time steps	Value	Change	Ratio	No. Iters			
					Min.	Max.	Total	Avg.
20	80	13.83869377			1	2	82	1.02
40	160	14.47322179	0.63452802		1	2	164	1.02
80	320	14.62625315	0.15303136	4.1	1	2	327	1.02
160	640	14.66532280	0.03906965	3.9	1	2	655	1.02
320	1280	14.67541115	0.01008834	3.9	1	2	1309	1.02
640	2560	14.67799017	0.00257902	3.9	1	2	2618	1.02
1280	5120	14.67864926	0.00065909	3.9	1	2	5238	1.02

Table 6.16: Experimental results for the American put option at $S = 100$ obtained with **uniform mesh methods** with **constant timesteps** using PENALTY-2. Reference numerical solution from [14] is 14.67882.

Nodes	Time steps	Value	Change	Ratio	No. Iters			
					Min.	Max.	Total	Avg.
20	80	13.83869377			1	2	81	1.01
40	160	14.47322179	0.63452802		1	2	162	1.01
80	320	14.62625315	0.15303136	4.1	1	2	321	1.00
160	640	14.66532280	0.03906965	3.9	1	2	642	1.00
320	1280	14.67541115	0.01008834	3.9	1	2	1283	1.00
640	2560	14.67799017	0.00257902	3.9	1	2	2565	1.00
1280	5120	14.67864926	0.00065909	3.9	1	2	5130	1.00

6.2.2 Adaptive Mesh Methods Results

We present selected numerical results for American put option pricing obtained by adaptive mesh methods. The adaptive mesh algorithm in use is Algorithm 4 on page 37. Similar to the uniform mesh methods, the constrained matrix problem at each timestep is solved using (P)SOR or the penalty method. In this section, we present experimental results by the iterative method PENALTY-1, PENALTY-2, or PSOR-2, which are defined as previously.

Note that so far we have been comparing our numerical solutions to the solution 14.67882 from [14]. It is worth noting that this result was obtained using a fixed non-uniform grid with 1073 nodes and 554 variable timesteps. This means that the spatial grid is non-uniform and is the same at all timesteps and the timesteps are not uniform. The domain used in [14] is $[0, 1000]$ and the authors use small uniform spatial stepsizes (a total of 865 nodes out of 1073 nodes) in the region $[0, 200]$ which includes the strike price and the free boundary and larger spatial stepsizes in the remaining region which is not of significant financial interest. We would like to emphasize that this is not a purely uniform mesh method. With adaptive mesh techniques, we would expect much more accurate results than this so we need a better reference. We use extrapolation on the results provided in [14], assuming quadratic convergence, as the methods are supposed to achieve. With an accuracy requirement of 10^{-6}, our new reference solution is 14.678886. We refer to it as the "true" value.

PSOR Method (PSOR-2)

Numerical results are presented in Table 6.17. In this case, the convergence rate for PSOR deteriorates even more dramatically on finer grids than it does with uniform mesh methods. For comparison purposes, we repeat selected numerical results for PSOR-2 with uniform and adaptive mesh methods in Table 6.18. It is clear from these experimental results that adaptive mesh methods with PSOR-2 are more efficient than uniform mesh methods with PSOR-2. For instance, let us consider an accuracy requirement of 10^{-2} as an example. With adaptive mesh methods, it takes 4973 iterations on a 80×320 grid to get 14.67549376, whereas the uniform

mesh methods take 5312 iterations on a larger grid of size 320×1280 to get 14.67541039, which is less accurate. The efficiency of the adaptive mesh methods is clear.

Table 6.17: Experimental results for the American put option at $S = 100$ obtained with **adaptive mesh methods** using PSOR-2. The "true" value 14.678886 was generated with accuracy 10^{-6} based on the results in [14] and extrapolation.

Nodes	Time steps	Value	Change	Ratio	Avg. ω	No. Iters				Adap. #
						Min.	Max.	Total	Avg.	
20	80	14.58152345			1.04	4	124	824	10.30	9
40	160	14.66359198	0.08206853		1.14	4	125	1763	11.02	13
80	320	14.67549376	0.01190178	6.9	1.24	5	526	4973	15.54	13
160	640	14.67835422	0.00286046	4.2	1.41	6	1629	13718	21.43	16
320	1280	14.67877355	0.00041933	6.8	1.46	7	6439	34167	26.69	19
640	2560	14.67887154	0.00009798	4.3	1.53	8	21673	92444	36.11	31

Table 6.18: Comparison of numerical results for the American put option between **uniform mesh methods** and **adaptive mesh methods** using PSOR-2 with **constant timesteps**. Numerical results and statistics are from Tables 6.13 and 6.17. The "true" value 14.678886 was generated with accuracy 10^{-6} based on the results in [14] and extrapolation.

Nodes	Time steps	Uniform		Adaptive		
		Value	Total	Value	Total	Adapt. #
20	80	13.83869377	341	14.58152345	824	9
40	160	14.47322179	663	14.66359198	1763	13
80	320	14.62625312	1312	14.67549376	4973	13
160	640	14.66532258	2613	14.67835422	13718	16
320	1280	14.67541039	5312	14.67877355	34167	19
640	2560	14.67799450	8919	14.67887154	92444	31

Penalty Method

Numerical results are presented in Tables 6.19 and 6.20. First, we compare the results between PENALTY-1 and PENALTY-2 in the context of adaptive mesh techniques. Again, the behaviors are expected, as between (P)SOR-1 and (P)SOR-2. PENALTY-1 and PENALTY-2 are almost identical except PENALTY-1 requires more iterations to converge than PENALTY-2. These experiments again confirm that the initial guess (3.32) can help to reduce the number of iterations needed for convergence without affecting numerical values.

Table 6.19: Experimental results for the American put option at $S = 100$ obtained with **adaptive mesh methods** and **constant timesteps** using PENALTY-1. The "true" value 14.678886 was generated with accuracy 10^{-6} based on the results in [14] and extrapolation.

	Time				No. Iters				Adap.
Nodes	steps	Value	Change	Ratio	Min.	Max.	Total	Avg.	#
20	80	14.58126790			1	3	101	1.26	9
40	160	14.66379516	0.08252726		1	4	197	1.23	12
80	320	14.67562816	0.01183300	7.0	1	6	371	1.16	13
160	640	14.67837088	0.00274273	4.3	1	9	735	1.15	14
320	1280	14.67876877	0.00039789	6.9	1	12	1509	1.18	21
640	2560	14.67884555	0.00007678	5.2	1	18	3065	1.20	40
1280	5120	14.67886475	0.00001920	4.0	1	23	6468	1.26	75

Table 6.20: Experimental results for the American put option at $S = 100$ obtained with **adaptive mesh methods** and **constant timesteps** using PENALTY-2. The "true" value 14.678886 was generated with accuracy 10^{-6} based on the results in [14] and extrapolation.

	Time				No. Iters				Adap.
Nodes	steps	Value	Change	Ratio	Min.	Max.	Total	Avg.	#
20	80	14.58126790			1	3	101	1.26	9
40	160	14.66379516	0.08252726		1	4	192	1.20	12
80	320	14.67562816	0.01183300	7.0	1	6	356	1.11	13
160	640	14.67837088	0.00274273	4.3	1	9	695	1.09	14
320	1280	14.67876877	0.00039789	6.9	1	12	1398	1.09	21
640	2560	14.67884555	0.00007678	5.2	1	18	2818	1.10	40
1280	5120	14.67886475	0.00001920	4.0	1	23	5849	1.14	75

Table 6.21: Comparison of numerical results for the American put option between **uniform mesh methods** and **adaptive mesh methods** using PENALTY-2 with **constant timesteps**. Numerical results and statistics are from Tables 6.16 and 6.20 . The "true" value 14.678886 was generated with accuracy 10^{-6} based on the results in [14] and extrapolation.

	Time	Uniform			Adaptive			
Nodes	steps	Value	"true" - Value	Total	Value	"true" - Value	Total	Adapt. #
20	80	13.83869377	8.4019e-001	81	14.58126790	9.7618e-002	101	9
40	160	14.47322179	2.0566e-001	162	14.66379516	1.5091e-002	192	12
80	320	14.62625315	5.2633e-002	321	14.67562816	3.2578e-003	356	13
160	640	14.66532280	1.3563e-002	642	14.67837088	5.1512e-004	695	14
320	1280	14.67541115	3.4748e-003	1283	14.67876877	1.1723e-004	1398	21
640	2560	14.67799017	8.9583e-004	2565	14.67884555	4.0450e-005	2818	40
1280	5120	14.67864926	2.3674e-004	5130	14.67886475	2.1250e-005	5849	75

Next, we compare the results between uniform mesh methods and adaptive mesh methods. For comparison purposes, in Table 6.21 we repeat selected numerical results of uniform mesh methods and adaptive mesh methods using the penalty method PENALTY-2. We can see that the adaptive mesh methods can produce more accurate results than the uniform mesh method by invoking the adaptive algorithm only a small number of times. Although for a specific grid size, the adaptive mesh methods require more iterations, they are still significantly more efficient than the uniform mesh methods since they produce much more accurate values. For instance, if we need a 10^{-3} accuracy (we are interested in the value 14.678), the uniform mesh methods can produce the solution 14.67799017 on a 640×2560 grid with 2565 iterations; however, the adaptive mesh method can produce the numerical solution 14.67837088 with a total of only 695 iterations on a much smaller grid of size 160×640. This is a significant

improvement in efficiency and accuracy.

Timestep Selector

We would like to test the application of a simple timestep selector suggested in [14]. The timestep selector proves to work very well on both uniform grids (see [20], and [21]) and non-uniform grids (see [14]). We would like to examine whether it works well in the context of adaptive mesh methods.

This method uses only information from the current timestep to predict a suitable timestep adjustment for the next timestep. It employs a relative change criterion. Specifically, given an initial timstep $\Delta\boldsymbol{\tau}_{\nu+1}$, then the new timestep is selected so that

$$\Delta\tau_{\nu+2} = \left(\min_i \left[\frac{\texttt{dnorm}}{\frac{|V(\mathbf{S}_i,\boldsymbol{\tau}_\nu+\Delta\boldsymbol{\tau}_{\nu+1})-V(\mathbf{S}_i,\boldsymbol{\tau}_\nu)|}{\max(D,|V(\mathbf{S}_i,\boldsymbol{\tau}_\nu+\Delta\boldsymbol{\tau}_{\nu+1})|,|V(\mathbf{S}_i,\boldsymbol{\tau}_\nu)|)}} \right] \right) \Delta\boldsymbol{\tau}_{\nu+1}, \tag{6.2}$$

where dnorm is a target change (during the timestep) specified by user. The scalar D is chosen so that the method does not take an excessively large timestep in the area where the value of the option is small. Normally, for options in dollars, $D = 1$ is used. In practice, the true values $V(\mathbf{S}_i, \boldsymbol{\tau}_\nu + \Delta\boldsymbol{\tau}_{\nu+1})$ and $V(\mathbf{S}_i, \boldsymbol{\tau}_\nu)$ in (6.2) are replaced by approximate values $\mathbf{V}(\mathbf{S}_i, \boldsymbol{\tau}_\nu + \Delta\boldsymbol{\tau}_{\nu+1})$ and $\mathbf{V}(\mathbf{S}_i, \boldsymbol{\tau}_\nu)$, respectively. We choose $\Delta\boldsymbol{\tau}_0$ for the coarsest grid, and then $\Delta\boldsymbol{\tau}_0$ is divided by 4 at each grid refinement. There is no problem with $\Delta\boldsymbol{\tau}_0$ being too conservative since the subsequent timesteps will allow the timestep size to increase very rapidly. The value of dnorm is reduced by two at each grid refinement. In the following runs, we use values of $\Delta\boldsymbol{\tau}_0 = 10^{-3}$ and $\texttt{dnorm} = 0.25$ on the coarsest grid. More analysis regarding this timestep selector can be found in [14].

Numerical results of adaptive mesh methods with variable timesteps are presented in Table 6.22. The effectiveness of the timestep selector is obvious in that the total number of iterations is reduced (because of the smaller number of timesteps) and the numerical option values is more accurate. For comparison purposes, in Table 6.23, we repeat selected numerical results of PENALTY-2 with constant timesteps and variable timesteps.

Table 6.22: Experimental results for the American put option at $S = 100$ obtained with **adaptive mesh methods** and **variable timesteps** using PENALTY-2. The "true" value 14.678886 was generated with accuracy 10^{-6} based on the results in [14] and extrapolation.

Nodes	Time steps	Value	Change	Ratio	No. Iters				Adap. #
					Min.	Max.	Total	Avg.	
20	33	14.60795215			1	2	60	1.82	11
40	69	14.66648980	0.05853765		1	3	100	1.45	16
80	137	14.67694906	0.01045927	5.6	1	3	178	1.30	21
160	275	14.67867207	0.00172300	6.1	1	3	337	1.23	24
320	550	14.67885648	0.00018442	9.3	1	4	676	1.23	39
640	1100	14.67888023	0.00002375	7.8	1	9	1399	1.27	61
1280	2198	14.67888300	0.00000277	8.6	1	18	2982	1.36	115

Table 6.23: Comparison between **adaptive mesh methods** using PENALTY-2 with **constant** and with **variable** timesteps. Numerical results and statistics are from Tables 6.20 and 6.22. The "true" value 14.678886 was generated with accuracy 10^{-6} based on the results in [14] and extrapolation.

Nodes	Constant Timesteps				Variable Timesteps			
	Time steps	Value	"true" - Value	Total	Time steps	Value	"true" - Value	Total
20	80	14.58126790	9.7618e-002	101	33	14.60795215	7.0934e-002	60
40	160	14.66379516	1.5091e-002	192	69	14.66648980	1.2396e-002	100
80	320	14.67562816	3.2578e-003	356	137	14.67694906	1.9369e-003	178
160	640	14.67837088	5.1512e-004	695	275	14.67867207	2.1393e-004	337
320	1280	14.67876877	1.1723e-004	1398	550	14.67885648	2.9520e-005	676
640	2560	14.67884555	4.0450e-005	2818	1100	14.67888023	5.7700e-006	1399
1280	5120	14.67886475	2.1250e-005	5849	2198	14.67888300	3.0000e-006	2982

Grid Distribution

In Figure 6.6, mesh point distributions are presented for a 160×640 grid. We observe that the mesh points are distributed in a significantly non-uniform manner, and they follow the movements of the free boundary quite well. As we expected, points are concentrated in the region around the strike. In addition, there is another important region which is the one around the free boundary point. As mentioned in Section 3.1, the free boundary values are functions of time and hence move. At the first timestep ($\nu = 1$), the free boundary is very close to the strike and hence we observe that all the points are clustered into this region. As time evolves, the free boundary moves away (towards the left boundary) from the strike. Hence we observe that grid points fan out and points are clustered towards that direction. At the same time, the adaptive mesh methods still keep many points around the strike. At the end, we can see that mesh points are separated into two main regions: the region around the strike and the one around the free boundary. Capturing the movements of the free boundary location is essential for accurate option value approximation. More examples concerning the free boundary points are given in Section 6.2.4.

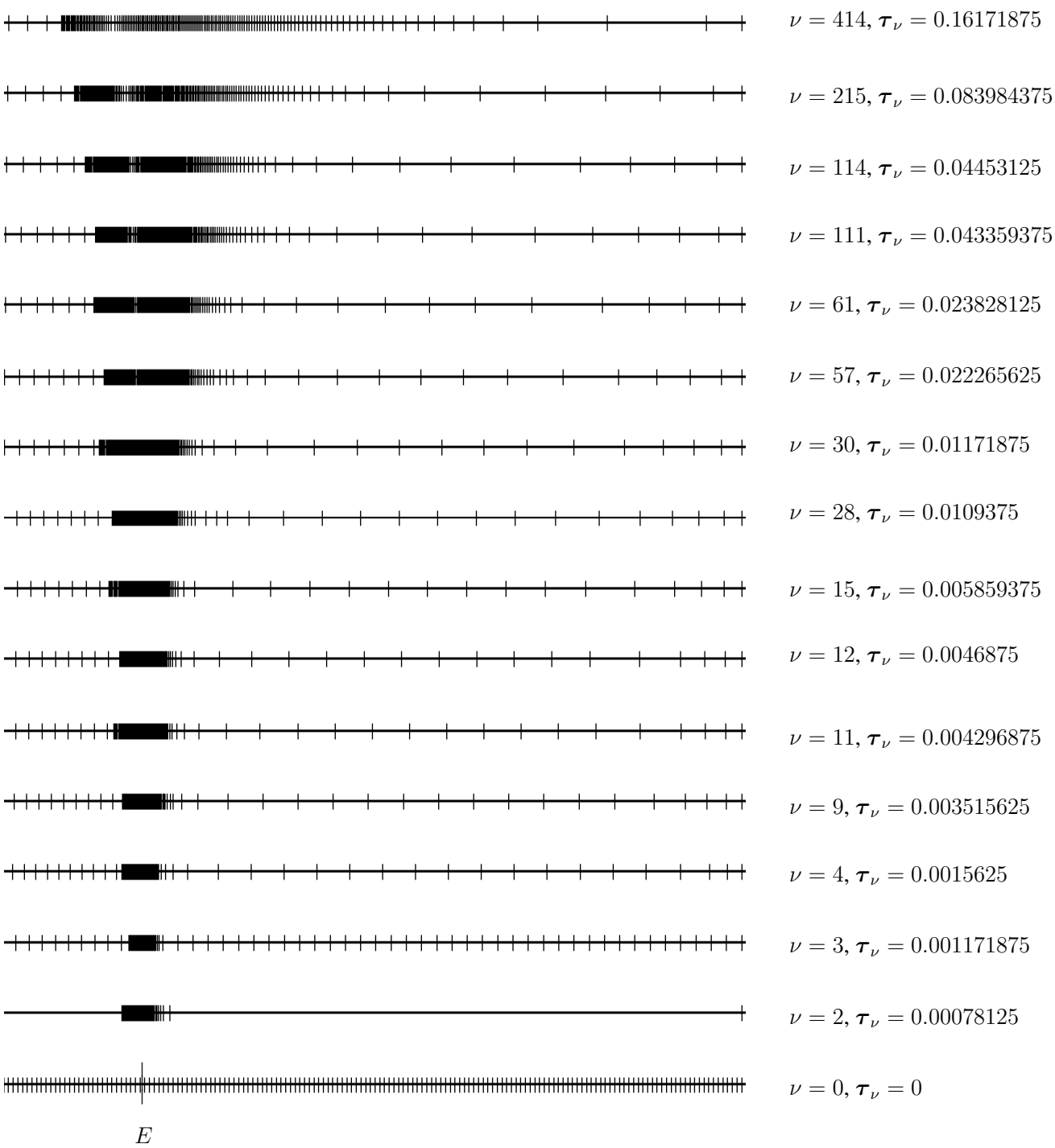


Figure 6.6: The locations of mesh points used by adaptive mesh methods for the American put on a 160×640 grid.

6.2.3 Efficiency Comparison

In this section, we would like to graphically compare the efficiency of selected methods for valuing American put options. In particular, we compare PSOR-2 and PENALTY-2 used on uniform and adaptive grids with each other. It is clear from the numerical experiments that the initial guess (3.32) based on extrapolation from previous timesteps requires fewer iterations to converge than (3.31) based on only the previous approximation. For this reason, we only compare efficiency of the methods using the initial guess (3.32).

To compare the efficiency, we plot values versus computation costs. We model the computation cost of a method by

$$\text{computation cost} = \text{Total} \times \text{Nodes},$$

where Total is the total number of iterations for all timesteps and Nodes is the number of spatial grid points as we defined earlier. We say Method-1 is more efficient than Method-2, given a computation cost, if Method-1 yields more accurate values than Method-2, or alternatively, Method-1 can attain a certain level of accuracy with less cost than Method-2. Efficiency is an important issue, especially in practical situations where time is a constraint and we want reasonably accurate results.

First, let us compare the efficiency between uniform and adaptive mesh methods. Efficiency plots of uniform and adaptive mesh methods using PSOR-2, PENALTY-1, and PENALTY-2 are presented in Figure 6.7. From this, it is very clear that adaptive mesh methods are more efficient than uniform mesh methods for any iterative method used. For example, for the same computation cost, adaptive mesh methods always yield more accurate values (closer to the "true" one). These results are expected from numerical results presented in the previous section.

Next, we compare the efficiency of adaptive mesh methods using different iterative solvers, namely adaptive mesh methods using PSOR-2, PENALTY-1, and PENALTY-2 with uniform timesteps and adaptive mesh methods using PENALTY-2 with variable timesteps. The plots are presented in Figure 6.8. From the numerical results collected, the adaptive mesh methods

using PENALTY-2 with variable timesteps are the most efficient ones. Figure 6.8 shows this fact graphically. The worst method among all these four is the adaptive mesh method with uniform timesteps and PSOR-2. Adaptive mesh methods using PENALTY-1 and PENALTY-2 are of almost the same level of efficiency, with PENALTY-2 being slightly more efficient than PENALTY-1.

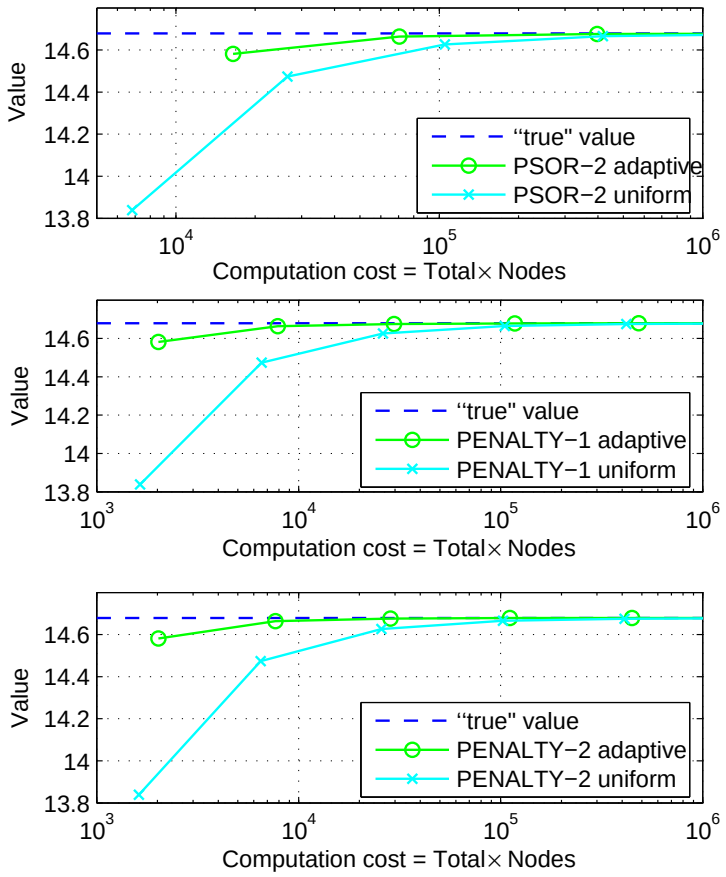


Figure 6.7: American put value at the strike price versus computational cost for uniform and adaptive mesh methods with different iterative solvers.

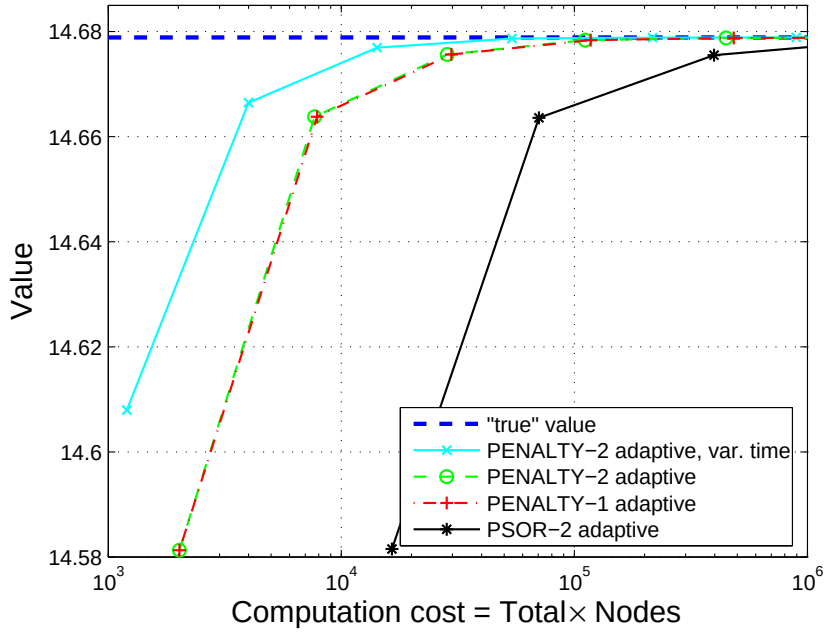


Figure 6.8: American put value at the strike price versus computation cost for adaptive mesh methods with different iterative solvers.

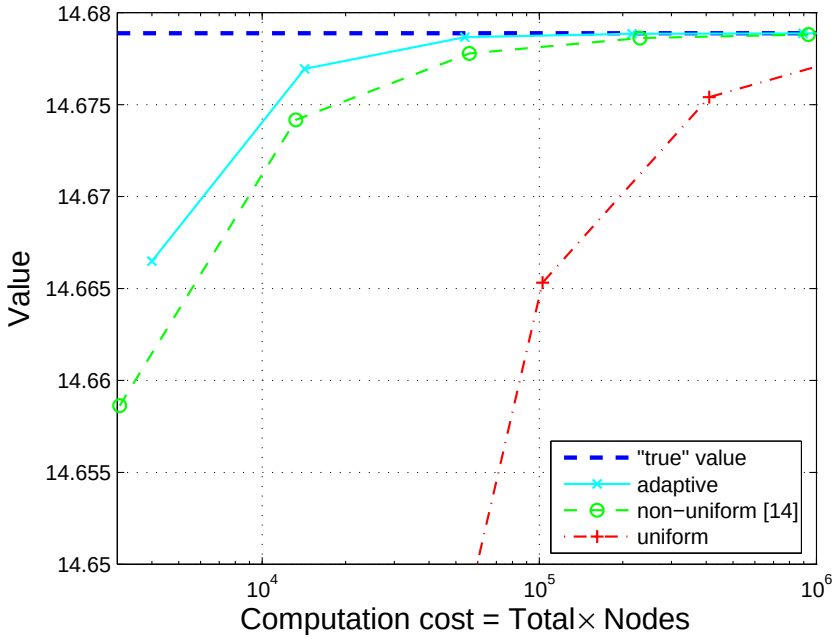


Figure 6.9: American put value at the strike price versus computational cost for various methods.

In Figure 6.9, we plot numerical values versus computation cost of three different methods: (1) adaptive mesh method using PENALTY-2 (our most efficient one), (2) (purely) uniform mesh method using PENALTY-2 implemented by us, and (3) the non-uniform mesh method as presented in [14]. Note that all three methods employ the penalty method to solve the constrained matrix problem at each timestep. It is obvious that the adaptive finite different method outperforms uniform and non-uniform mesh methods in terms of efficiency.

6.2.4 Early Exercise Boundary

An interesting problem associated with American option pricing is the analysis of the early exercise boundary and the optimal stopping time. This problem has attracted a lot of attention due to its theoretical and practical importance. At each timestep, the free boundary point is described by (3.5) on page 17. The accuracy with which we locate the free boundary has strong effects on the quality of the numerical value of the option computed. The exact analytical expression for the free boundary is not known. One standard way to locate the free boundary is to use binomial or trinomial methods. In addition, many researchers have investigated various models such as integral equations or asymptotic solutions leading to approximation for the free boundary. The purpose of this section is to demonstrate the accuracy of the adaptive mesh methods in locating the free boundary for an American put option at each timestep. We carry out a comparison between our numerical free boundary points and those obtained by several other methods, namely binomial methods, trinomial methods, integral equations, asymptotic approximations, and a method proposed by MacMillan, Barone- Adesi and Whaleys in [1] and [23] (the M.B.W approximation method). We refer the reader to the paper [31] for complete references and numerical results for these methods. Note that for both the binomial and trinomial methods a depth of 1000 subdivisions was used. In [31], reference results from the trinomial tree were computed using the software package "Option Calculator" developed by Srivastava et al. at Carnegie Mellon University.

Under the smooth condition (3.5), the free boundary point marks the change of the function from being linear to non-linear. We would expect that, with such a change, the approximation error in the region around the free boundary is higher and, hence, by the equidistribution principle, the region around the strike will be more refined, resulting in a more accurate free boundary approximation. For comparison purposes, we experiment with two different sets of parameters used in [31] listed in Tables 6.24 and 6.25. Numerical results for the free boundary locations at selected times are presented in Tables 6.26 and 6.27, respectively. Corresponding results by other methods listed in these tables can be found in Tables 1 and 2 in [31], respectively.

Table 6.24: Model parameters (I) for comparison of early exercise point in American options.

Parameter	Value
Time to expiry T	0.05 (years)
Interest rate r	10% (0.1)
Exercise price E	50
Volatility σ	40% (0.4)
Tolerance ϵ	10^{-7}

Table 6.25: Model parameters (II) for comparison of early exercise point in American options.

Parameter	Value
Time to expiry T	0.05 (years)
Interest rate r	10% (0.1)
Exercise price E	10
Volatility σ	25% (0.25)
Tolerance ϵ	10^{-7}

Method PENALTY-2 is used to solve the constrained matrix problem at each timestep.

In [31], the authors refer to approximations by binomial and trinomial methods as accurate values. In most cases, these two methods yield approximations that agree up to in the third decimal place. Hence, we also use approximations computed by these two methods as references. Numerical results obtained by uniform and adaptive mesh methods on a 200×200 grid are presented in Tables 6.26 and 6.27. First, it is obvious that the uniform mesh methods give very poor results compared to those by binomial and trinomial methods and all other methods. In most cases, uniform mesh methods underestimate the free boundary, giving results with two digits of accuracy. For instance, in both Tables 6.26 and 6.27, the approximation by the uniform mesh method at $\tau = 0.05$ underestimates by about 0.16 dollars or 16 cents compared

Table 6.26: Early exercise boundary for set of parameters listed in Table 6.24. Uniform and adaptive mesh methods are used on a 200×200 grid with PENALTY-2 and Rannacher smoothing. $\mathbf{S}_{\max} = 250$.

Time ($\tau = T - t$)	Integral Equation	Binomial Method	Trinomial Method	Asymp. Solution	M.B.W Approx.	Adap. Method	Uniform Method
0.00100 (8.76 hours)	48.3819	48.3915	48.3915	48.4176	48.4871	48.3634	47.5000
0.00500	46.8631	46.8836	46.8833	46.9771	47.0678	46.8971	46.2500
0.01000 (3.65 days)	45.8848	45.9115	45.9122	46.0773	46.1507	45.9326	45.0000
0.05000 (2.6 weeks)	42.6111	42.6681	42.6672	43.3211	43.0562	42.6425	42.5000

Table 6.27: Early exercise boundary for set of parameters listed in Table 6.25. Uniform and adaptive mesh methods are used on a 200×200 grid with PENALTY-2 and Rannacher smoothing. $\mathbf{S}_{\max} = 50$.

Time $\tau = T - t$	Integral Equation	Binomial Method	Trinomial Method	Asymp. Solution	M.B.W Approx.	Adap. Method	Uniform Method
0.001 (8.76 hours)	9.8099	9.8111	9.8111	9.8154	9.8225	9.8151	9.7500
0.005	9.6349	9.6375	9.6374	9.6533	9.6593	9.6423	9.5000
0.010 (3.65 days)	9.5232	9.5265	9.5266	9.5553	9.5546	9.5219	9.5000
0.050 (2.6 weeks)	9.1550	9.1600	9.1601	9.2865	9.2047	9.1530	9.0000

to the approximations obtained by the binomial and trinomial methods, which is significant. However, it is very interesting to note that the adaptive mesh methods capture the free boundary locations quite well as time evolves. In most cases, approximations by adaptive mesh methods agree with binomial and trinomial methods up to the first decimal place, whence the differences are in cents only. For instance at the time level $\tau = 0.05$, from Tables 6.26 and 6.27, the adaptive mesh methods yield results which are off by about 2 and 1 cents, respectively. At the time level $\tau = 0.01$, the adaptive mesh methods give results that underestimate those computed by binomial and trinomial methods by about 0.5 cents only (see Table 6.27). Compared with results by other methods considered accurate in the literature, our numerical free boundary points exhibit similar level of accuracy. It is important to note that we use only a 200×200 grid, which is of modest size.

From these numerical results, we can see that the adaptive mesh methods locate the free boundary with accuracy comparable to other methods, but with fewer points. In comparison with uniform mesh methods, adaptive mesh methods are clearly superior in accuracy in locating the free boundary. In Figures 6.10 and 6.11, profiles of the free boundary obtained by uniform and adaptive mesh methods are presented. The profiles generated by adaptive mesh methods are smooth while those by uniform mesh methods are highly non-smooth and look like a step function. Note that the free boundary is a function of time, as discussed in Section 3.1, hence moving as time evolves. Profiles of the free boundary point that look like a step function do not capture this movement properly. On the contrary, as we discussed in the section on grid distribution, the adaptive mesh methods can capture the movement of the free boundary point with time well.

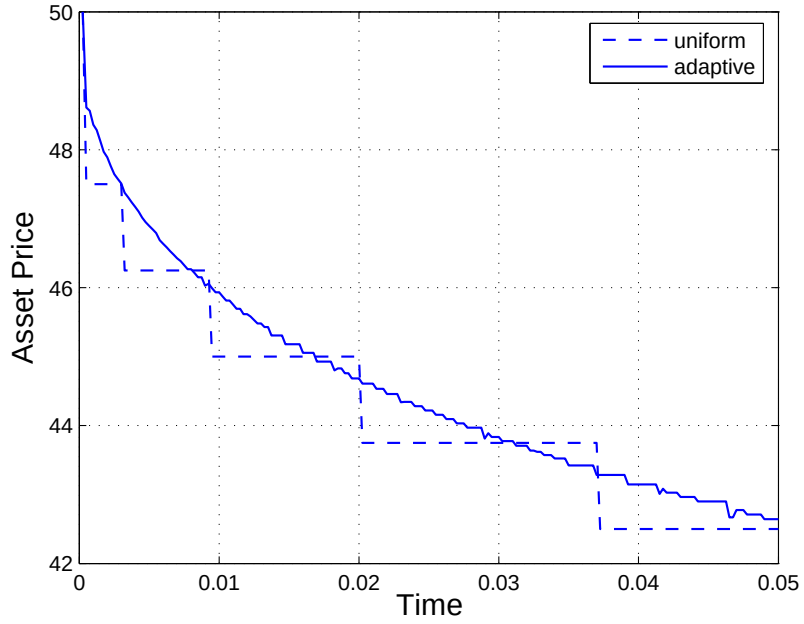


Figure 6.10: Profile of the free boundary obtained by uniform and adaptive mesh methods with set of parameters from Table 6.24 on a 200×200 grid.

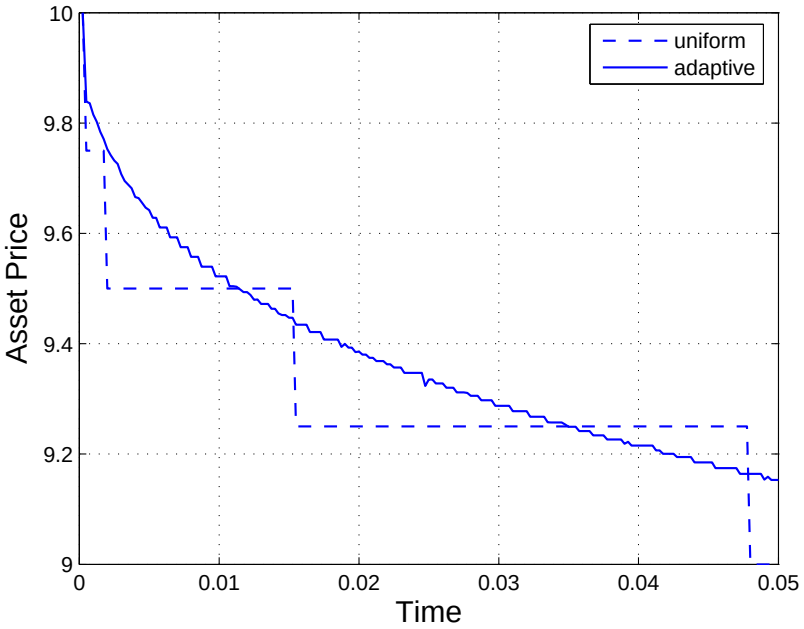


Figure 6.11: Profile of the free boundary obtained by uniform and adaptive mesh methods with set of parameters from Table 6.25 on a 200×200 grid.

Chapter 7

Conclusions and Future Work

The American option pricing problem is an important and challenging problem in finance due to the existence of the free boundary. In this paper, we have considered a PDE approach to price American options written on a single asset with constant volatility and interest rate. Although we focused the discussion on American put options, the ideas and method considered are useful for other American style options, such as American call options on dividend-paying assets, American style Asian options, or the pricing of convertible bonds with early exercise features.

Our approach is based on finite differences integrated with adaptive mesh methods which rely on grading and monitor functions to determine the distribution of the error along the spatial dimension. At certain timesteps, the adaptive techniques relocate the nodes to equidistribute the error in some chosen norm among the subintervals of the partition. We monitor the spatial discretization and redistribution of grid points so that at each timestep the resulting matrix has an $\mathcal{M}$-matrix structure. For the solution of the LCP at each timestep, we considered two iterative methods, the PSOR and penalty methods, with two different initial guesses: one is the approximation to the solution at the previous step and the other one is based of linear extrapolation of approximations to the solution at the previous two steps.

We examined the convergence rate of PSOR and the penalty methods under the effects

of the initial guesses for option pricing using both uniform and adaptive mesh methods. Experimental results show that the initial guess based on extrapolation significantly reduces the deterioration of the convergence rates of PSOR as grids are refined. The effects of the initial guesses on the convergence of the penalty method is not as significant due to the fast convergence of the method; however the penalty method with initial guess based on extrapolation still converges faster. The penalty method saves many iterations and much computation time compared to the PSOR method. Therefore, we choose to combine the penalty method with initial guess based on extrapolation with the adaptive mesh methods.

We investigated and compared the accuracy and efficiency of the adaptive mesh methods with uniform and certain prescribed non-uniform finite difference discretization methods for the evaluation of the American put options. We found that the adaptive mesh methods are superior to the other two methods in terms of accuracy and efficiency. An important feature of the American option pricing problem is the early exercise. We compared the accuracy of the adaptive mesh methods in locating the early exercise boundary with several other methods. The results are very promising: our method performs as well as any others proposed in the literature and it is significantly better than uniform mesh methods.

Several of the ideas considered in the thesis could be useful for other derivative pricing problems that can be modelled by PDEs: for example, barrier and Asian options, or control problems, such as those involving the Hamilton-Jacobi-Bellman equation. Moreover, the adaptive techniques can be combined with gridsize estimators in order to pick the number of discretization points so that the error in the approximation is below a given error tolerance. Solving multi-dimensional option pricing problems by adaptive mesh techniques is an interesting and challenging problem.

Appendix A

Convergence Proof of PSOR

We first describe the notations used in this proof. Let $|.|$ and $\|.\|_p$ denote the absolute value and the standard norm, respectively, with $p = 2$ or ∞. All matrices and vectors are real. For a real matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, let $|\mathbf{A}|$ denote the matrix obtained from $\mathbf{A}$ by replacing each element $\mathbf{a}_{i,j}$ of $\mathbf{A}$ by $|\mathbf{a}_{i,j}|$. A similar notation holds for vectors in $\mathbb{R}^n$. We say that $\mathbf{A} \leq \tilde{\mathbf{A}}$ if and only if $\mathbf{a}_{i,j} \leq \tilde{\mathbf{a}}_{i,j}$ for each pair (i, j), $1 \leq i, j \leq n$. Similarly, $\mathbf{x} \leq \tilde{\mathbf{x}}$ if and only if $\mathbf{x}_i \leq \tilde{\mathbf{x}}_i$, $1 \leq i \leq n$, where $\mathbf{x}$ and $\tilde{\mathbf{x}}$ are vectors in $\mathbb{R}^n$. Here, $\mathbf{x}_i$ denotes the ith entry of the vector $\mathbf{x}$.

Let us consider the generic constrained matrix problem (3.29)

$$\left\{ \begin{array}{l} \mathbf{Ax} = \mathbf{b} \\ \mathbf{x} \geq \mathbf{g} \end{array} \right\} \text{ or } \left\{ \begin{array}{l} \mathbf{Ax} > \mathbf{b} \\ \mathbf{x} = \mathbf{g} \end{array} \right\},$$

and the corresponding PSOR algorithm (3.30)

$$\left\{ \begin{array}{l} \textbf{for } i = 1, \ldots, n \textbf{ do} \\ \quad \mathbf{y}_i^{(k+1)} = \dfrac{1}{\mathbf{a}_{i,i}} \Big(\mathbf{b}_i - \displaystyle\sum_{j<i} \mathbf{a}_{i,j} \mathbf{x}_j^{(k+1)} - \sum_{j>i} \mathbf{a}_{i,j} \mathbf{x}_j^{(k)} \Big) \\ \quad \mathbf{x}_i^{(k+1)} = \max \left(\mathbf{g}_i, \mathbf{x}_i^{(k)} + \omega (\mathbf{y}_i^{(k+1)} - \mathbf{x}_i^{(k)}) \right) \\ \textbf{endfor} \end{array} \right.$$

This problem is the same with the constrained matrix problem (4.2) arising at each time step in American option pricing with $\mathbf{A}, \mathbf{b}$ and $\mathbf{g}$ playing the role of $\mathbf{A}^\nu, \mathbf{b}^\nu$ and the payoff $\mathbf{V}^{*,\nu+1}$,

respectively. We assume that $\mathbf{A}$ is an $\mathcal{M}$-matrix and strictly diagonally dominant. Recall that $\mathbf{A}^{\nu}$ has these properties. We will prove that under these conditions, and if the relaxation parameter ω satisfies $0 < \omega < \tilde{\omega} = 2\min_i\{\frac{\mathbf{a}_{i,i}}{\sum_j |\mathbf{a}_{i,j}|}\}$ then the PSOR algorithm (3.30) converges to the unique solution of the constrained matrix problem (3.29).

The proof is based on the idea of contractions under partial ordering introduced in Chapter 13 in [26] and organized as follows. In Lemma 1, we develop an upper bound for $|\mathbf{x}_i^{(k+1)} - \mathbf{x}_i^{(k)}|$, then in Lemma 2 a recursive bound for $|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}|$. In Lemma 3, we show that under certain conditions, the spectral radius of a relevant matrix is less than 1. This is needed in Theorem 3 that proves the convergence of the sequence generated by PSOR algorithm (3.30). In Lemma 4, we establish an equivalence between solving the constrained matrix problem (3.29) and computing a vector $\mathbf{x}$ satisfying

$$\mathbf{x}_i - \max\Big(\mathbf{g}_i, \mathbf{x}_i + \frac{\omega}{\mathbf{a}_{i,i}}(\mathbf{b}_i - \sum_j \mathbf{a}_{i,j}\mathbf{x}_j)\Big) = 0, \qquad 1 \le i \le n.$$

In Lemma 5, we prove that if matrix $\mathbf{A}$ is an $\mathcal{M}$-matrix, then the constrained matrix problem (3.29) has an unique solution. These five lemmas and the theorem will be the basis for showing that the sequence $\mathbf{x}^{(k)}$ generated by the PSOR algorithm (3.30) is convergent, that is, $\lim_{k\to\infty} \mathbf{x}^{(k)} = \tilde{\mathbf{x}}$ and the limit point $\tilde{\mathbf{x}}$ uniquely solves the constrained matrix problem (3.29).

LEMMA 1. *Assume that all diagonal entries of the matrix* $\mathbf{A}$ *in the constrained matrix problem (3.29) are positive, that is* $\mathbf{a}_{i,i} > 0$*, for* $1 \le i \le n$*. Let* $\mathbf{x}^{(k+1)}$ *and* $\mathbf{x}^{(k)}$ *be the* $(k+1)$*st and* k*th solutions generated by the PSOR algorithm 3.30. Then,*

$$|\mathbf{x}_i^{(k+1)} - \mathbf{x}_i^{(k)}| \le |1-\omega||\mathbf{x}_i^{(k)} - \mathbf{x}_i^{(k-1)}| + \frac{\omega}{\mathbf{a}_{i,i}}\Big(\sum_{j<i} |\mathbf{a}_{i,j}||\mathbf{x}_j^{(k+1)} - \mathbf{x}_j^{(k)}|\Big) + \frac{\omega}{\mathbf{a}_{i,i}}\Big(\sum_{j>i} |\mathbf{a}_{i,j}||\mathbf{x}_j^{(k)} - \mathbf{x}_j^{(k-1)}|\Big),$$

for $1 \le i \le n$.

Proof. From the PSOR algorithm (3.30), we have

$$\mathbf{x}_i^{(k+1)} = \max\Big(\mathbf{g}_i, (1-\omega)\mathbf{x}_i^{(k)} + \frac{\omega}{\mathbf{a}_{i,i}}(\mathbf{b}_i - \sum_{j<i}\mathbf{a}_{i,j}\mathbf{x}_j^{(k+1)} - \sum_{j>i}\mathbf{a}_{i,j}\mathbf{x}_j^{(k)})\Big), \tag{A.1}$$

$$\mathbf{x}_i^{(k)} = \max\Big(\mathbf{g}_i, (1-\omega)\mathbf{x}_i^{(k-1)} + \frac{\omega}{\mathbf{a}_{i,i}}(\mathbf{b}_i - \sum_{j<i}\mathbf{a}_{i,j}\mathbf{x}_j^{(k)} - \sum_{j>i}\mathbf{a}_{i,j}\mathbf{x}_j^{(k-1)})\Big). \tag{A.2}$$

Subtracting (A.2) from (A.1), we obtain [1]

$$\begin{aligned}
\mathbf{x}_i^{(k+1)} - \mathbf{x}_i^{(k)} &= \max\Big(\mathbf{g}_i, (1-\omega)\mathbf{x}_i^{(k)} + \frac{\omega}{\mathbf{a}_{i,i}}(\mathbf{b}_i - \sum_{j<i}\mathbf{a}_{i,j}\mathbf{x}_j^{(k+1)} - \sum_{j>i}\mathbf{a}_{i,j}\mathbf{x}_j^{(k)})\Big) \\
&\quad - \max\Big(\mathbf{g}_i, (1-\omega)\mathbf{x}_i^{(k-1)} + \frac{\omega}{\mathbf{a}_{i,i}}(\mathbf{b}_i - \sum_{j<i}\mathbf{a}_{i,j}\mathbf{x}_j^{(k)} - \sum_{j>i}\mathbf{a}_{i,j}\mathbf{x}_j^{(k-1)})\Big) \\
&\le \max\Big(0, (1-\omega)(\mathbf{x}_i^{(k)} - \mathbf{x}_i^{(k-1)}) - \frac{\omega}{\mathbf{a}_{i,i}}\big(\sum_{j<i}\mathbf{a}_{i,j}(\mathbf{x}_j^{(k+1)} - \mathbf{x}_j^{(k)})\big) \\
&\qquad - \frac{\omega}{\mathbf{a}_{i,i}}\big(\sum_{j>i}\mathbf{a}_{i,j}(\mathbf{x}_j^{(k)} - \mathbf{x}_j^{(k-1)})\big)\Big).
\end{aligned}$$

Then we have the following inequality [2]:

$$\begin{aligned}
\max(0, \mathbf{x}_i^{(k+1)} - \mathbf{x}_i^{(k)}) \le \max\Big(0, (1-\omega)(\mathbf{x}_i^{(k)} - \mathbf{x}_i^{(k-1)}) &- \frac{\omega}{\mathbf{a}_{i,i}}\big(\sum_{j<i}\mathbf{a}_{i,j}(\mathbf{x}_j^{(k+1)} - \mathbf{x}_j^{(k)})\big) \\
&- \frac{\omega}{\mathbf{a}_{i,i}}\big(\sum_{j>i}\mathbf{a}_{i,j}(\mathbf{x}_j^{(k)} - \mathbf{x}_j^{(k-1)})\big)\Big).
\end{aligned} \tag{A.3}$$

We can obtain a similar result for $\max(0, \mathbf{x}_i^{(k)} - \mathbf{x}_i^{(k+1)})$, that is

$$\begin{aligned}
\max(0, \mathbf{x}_i^{(k)} - \mathbf{x}_i^{(k+1)}) \le \max\Big(0, (1-\omega)(\mathbf{x}_i^{(k-1)} - \mathbf{x}_i^{(k)}) &- \frac{\omega}{\mathbf{a}_{i,i}}\big(\sum_{j<i}\mathbf{a}_{i,j}(\mathbf{x}_j^{(k)} - \mathbf{x}_j^{(k+1)})\big) \\
&- \frac{\omega}{\mathbf{a}_{i,i}}\big(\sum_{j>i}\mathbf{a}_{i,j}(\mathbf{x}_j^{(k-1)} - \mathbf{x}_j^{(k)})\big)\Big).
\end{aligned} \tag{A.4}$$

[1] It can easily be shown that for any $a, b, c \in \mathbb{R}$, $\max(a,b) - \max(a,c) \le \max(0, b-c)$.

[2] For any $a, b, c \in \mathbb{R}$, if $a \le \max(b,c)$, then $\max(a,b) \le \max(b,c)$.

Summing up (A.3) and (A.4), and taking into account that $\mathbf{a}_{i,i} > 0$, we obtain [3]

$$\begin{aligned}|\mathbf{x}_i^{(k+1)} - \mathbf{x}_i^{(k)}| \leq |(1-\omega)(\mathbf{x}_i^{(k)} - \mathbf{x}_i^{(k-1)}) - \frac{\omega}{\mathbf{a}_{i,i}}\big(\sum_{j<i} \mathbf{a}_{i,j}(\mathbf{x}_j^{(k+1)} - \mathbf{x}_j^{(k)})\big) \\ - \frac{\omega}{\mathbf{a}_{i,i}}\big(\sum_{j>i} \mathbf{a}_{i,j}(\mathbf{x}_j^{(k)} - \mathbf{x}_j^{(k-1)})\big)| \\ \leq |1-\omega||\mathbf{x}_i^{(k)} - \mathbf{x}_i^{(k-1)}| + \frac{\omega}{\mathbf{a}_{i,i}}\big(\sum_{j<i} |\mathbf{a}_{i,j}||\mathbf{x}_j^{(k+1)} - \mathbf{x}_j^{(k)}|\big) \\ + \frac{\omega}{\mathbf{a}_{i,i}}\big(\sum_{j>i} |\mathbf{a}_{i,j}||\mathbf{x}_j^{(k)} - \mathbf{x}_j^{(k-1)}|\big),\end{aligned} \tag{A.5}$$

for $1 \leq i \leq n$. □

LEMMA 2. *Let* $\mathbf{A}$ *in the constrained matrix problem (3.29) be a strictly diagonally dominant* $\mathcal{M}$*-matrix in* $\mathbb{R}^{n\times n}$ *with* $\mathbf{D}$, $\mathbf{L}$ *and* $\mathbf{U}$ *representing the diagonal, strictly lower triangular and strictly upper triangular parts of* $\mathbf{A}$*, respectively. Then,* $\mathbf{x}^{(k+1)}$ *and* $\mathbf{x}^{(k)}$ *satisfy*

$$|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}| \leq (\mathbf{I} - \omega\mathbf{D}^{-1}|\mathbf{L}|)^{-1}|\mathbf{I} - \omega\mathbf{D}^{-1}(\mathbf{A}-\mathbf{L})||\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}|,$$

where $\mathbf{x}^{(k+1)}$ *and* $\mathbf{x}^{(k)}$ *are the* $(k+1)$*st and* k*th solutions generated by the PSOR algorithm (3.30), respectively.*

Proof. First, we prove

$$(\mathbf{I} - \omega\mathbf{D}^{-1}|\mathbf{L}|)|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}| \leq |\mathbf{I} - \omega\mathbf{D}^{-1}(\mathbf{A}-\mathbf{L})||\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}|.$$

For simplicity, let

$$\mathbf{y} = (\mathbf{I} - \omega\mathbf{D}^{-1}|\mathbf{L}|)|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}|, \tag{A.6}$$

and

$$\tilde{\mathbf{y}} = |\mathbf{I} - \omega\mathbf{D}^{-1}(\mathbf{A}-\mathbf{L})||\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}|.$$

Note that $\tilde{\mathbf{y}}$ can be rewritten as

$$\begin{aligned}\tilde{\mathbf{y}} &= |\mathbf{I} - \omega\mathbf{D}^{-1}(\mathbf{A}-\mathbf{L})||\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}| \\ &= |\mathbf{I} - \omega\mathbf{D}^{-1}(\mathbf{D}+\mathbf{U})||\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}| \\ &= |(1-\omega)\mathbf{I} - \omega\mathbf{D}^{-1}\mathbf{U}||\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}|,\end{aligned}$$

[3] For any $a \in \mathbb{R}$, $\max(0,a) + \max(0,-a) = |a|$.

or equivalently,

$$\tilde{\mathbf{y}} = \big(|1-\omega|\mathbf{I} + \omega\mathbf{D}^{-1}|\mathbf{U}|\big)|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}|. \tag{A.7}$$

We have relation (A.7) since $\mathbf{U} \leq \mathbf{0}, \mathbf{D} > \mathbf{0}$ and hence $-\omega\mathbf{D}^{-1}\mathbf{U} = \omega\mathbf{D}^{-1}|\mathbf{U}| \geq \mathbf{0}$ and in addition, $(1-\omega)\mathbf{I}$ and $\omega\mathbf{D}^{-1}|\mathbf{U}|$ do not share any index of non-zero entries.

We prove that $\mathbf{y}_i \leq \tilde{\mathbf{y}}_i, \forall i, 1 \leq i \leq n$ using induction on the row index i.

$\underline{i=1}$:

We have that

$$\mathbf{y}_1 = |\mathbf{x}_1^{(k+1)} - \mathbf{x}_1^{(k)}|$$

and

$$\tilde{\mathbf{y}}_1 = |1-\omega||\mathbf{x}_1^{(k)} - \mathbf{x}_1^{(k-1)}| + \frac{\omega}{\mathbf{a}_{1,1}}\big(\sum_{j>1}|\mathbf{a}_{1,j}||\mathbf{x}_j^{(k)} - \mathbf{x}_j^{(k-1)}|\big).$$

Then $\mathbf{y}_1 \leq \tilde{\mathbf{y}}_1$ follows directly from Lemma 1 when $i = 1$.

Assume that $\mathbf{y}_i \leq \tilde{\mathbf{y}}_i$ for each i, $1 \leq i \leq \tilde{k} - 1$, where $2 \leq \tilde{k} \leq n$. Now we prove that $\mathbf{y}_{\tilde{k}} \leq \tilde{\mathbf{y}}_{\tilde{k}}$.

$\underline{i=\tilde{k}}$: We have

$$\mathbf{y}_{\tilde{k}} = -\frac{\omega}{\mathbf{a}_{\tilde{k},\tilde{k}}}\big(\sum_{j<\tilde{k}}|\mathbf{a}_{\tilde{k},j}||\mathbf{x}_j^{(k+1)} - \mathbf{x}_j^{(k)}|\big) + |\mathbf{x}_{\tilde{k}}^{(k+1)} - \mathbf{x}_{\tilde{k}}^{(k)}|,$$

and

$$\tilde{\mathbf{y}}_{\tilde{k}} = |1-\omega||\mathbf{x}_{\tilde{k}}^{(k)} - \mathbf{x}_{\tilde{k}}^{(k-1)}| + \frac{\omega}{\mathbf{a}_{\tilde{k},\tilde{k}}}\big(\sum_{j>\tilde{k}}|\mathbf{a}_{\tilde{k},j}||\mathbf{x}_j^{(k+1)} - \mathbf{x}_j^{(k)}|\big).$$

Note that we can get an upper bound for the term $|\mathbf{x}_{\tilde{k}}^{(k+1)} - \mathbf{x}_{\tilde{k}}^{(k)}|$ in $\mathbf{y}_{\tilde{k}}$ using Lemma 1 with $i = \tilde{k}$. Thus we have

$$\begin{aligned}
\mathbf{y}_{\tilde{k}} &\leq -\frac{\omega}{\mathbf{a}_{\tilde{k},\tilde{k}}}\big(\sum_{j<\tilde{k}}|\mathbf{a}_{\tilde{k},j}||\mathbf{x}_j^{(k+1)} - \mathbf{x}_j^{(k)}|\big) + |1-\omega||\mathbf{x}_{\tilde{k}}^{(k)} - \mathbf{x}_{\tilde{k}}^{(k-1)}| \\
&\quad + \frac{\omega}{\mathbf{a}_{\tilde{k},\tilde{k}}}\big(\sum_{j<\tilde{k}}|\mathbf{a}_{\tilde{k},j}||\mathbf{x}_j^{(k+1)} - \mathbf{x}_j^{(k)}|\big) + \frac{\omega}{\mathbf{a}_{\tilde{k},\tilde{k}}}\big(\sum_{j>\tilde{k}}|\mathbf{a}_{\tilde{k},j}||\mathbf{x}_j^{(k)} - \mathbf{x}_j^{(k-1)}|\big) \\
&= |1-\omega||\mathbf{x}_{\tilde{k}}^{(k)} - \mathbf{x}_{\tilde{k}}^{(k-1)}| + \frac{\omega}{\mathbf{a}_{\tilde{k},\tilde{k}}}\big(\sum_{j>\tilde{k}}|\mathbf{a}_{\tilde{k},j}||\mathbf{x}_j^{(k)} - \mathbf{x}_j^{(k-1)}|\big) \\
&= \tilde{\mathbf{y}}_{\tilde{k}}.
\end{aligned}$$

By induction, we conclude that $\mathbf{y}_i \leq \tilde{\mathbf{y}}_i, \forall i, 1 \leq i \leq n$, or equivalently

$$(\mathbf{I} - \omega\mathbf{D}^{-1}|\mathbf{L}|)|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}| \leq |\mathbf{I} - \omega\mathbf{D}^{-1}(\mathbf{A} - \mathbf{L})||\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}|.$$

Now we consider the matrix $\mathbf{I} - \omega\mathbf{D}^{-1}|\mathbf{L}|$. Since the matrix $\omega\mathbf{D}^{-1}|\mathbf{L}|$ is strictly lower triangular and $\omega\mathbf{D}^{-1}|\mathbf{L}| \geq \mathbf{0}$, it follows that $\mathbf{I} - \omega\mathbf{D}^{-1}|\mathbf{L}|$ is a unit lower triangular matrix (with non-positive off-diagonal entries). Thus $\mathbf{I} - \omega\mathbf{D}^{-1}|\mathbf{L}|$ is invertible and $(\mathbf{I} - \omega\mathbf{D}^{-1}|\mathbf{L}|)^{-1} \geq \mathbf{0}$. [4] Then $\mathbf{I} - \omega\mathbf{D}^{-1}|\mathbf{L}|$ is a monotone matrix. [5] Hence we have

$$|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}| \leq (\mathbf{I} - \omega\mathbf{D}^{-1}|\mathbf{L}|)^{-1}\big|\big(\mathbf{I} - \omega\mathbf{D}^{-1}(\mathbf{A} - \mathbf{L})\big)\big||\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}|.$$

□

In the next lemma, we will show that if $0 < \omega < \tilde{\omega} = 2\min_i\{\frac{\mathbf{a}_{i,i}}{\sum_j |\mathbf{a}_{i,j}|}\}$ then the spectral radius of matrix $(\mathbf{I} - \omega\mathbf{D}^{-1}|\mathbf{L}|)^{-1}\big|\big(\mathbf{I} - \omega\mathbf{D}^{-1}(\mathbf{A} - \mathbf{L})\big)\big|$ is less than 1.

LEMMA 3. *Let* $\mathbf{A}, \mathbf{D}, \mathbf{L}$ *and* $\mathbf{U}$ *be as in Lemma 2 and* $0 < \omega < \tilde{\omega} = 2\min_i\{\frac{\mathbf{a}_{i,i}}{\sum_j |\mathbf{a}_{i,j}|}\}$. *Then the spectral radius of matrix* $\mathbf{H} = (\mathbf{I} - \omega\mathbf{D}^{-1}|\mathbf{L}|)^{-1}|\mathbf{I} - \omega\mathbf{D}^{-1}(\mathbf{A} - \mathbf{L})|$ *is less than* 1*, that is* $\rho(\mathbf{H}) < 1$.

Proof. Note that under condition (5.3), the matrix $\mathbf{A}$ is a strictly row diagonally dominant. Thus it follows that $1 < \tilde{\omega} \leq 2$. We will prove $\|\mathbf{H}\|_\infty < 1$ instead, noting the property $\rho(\mathbf{H}) \leq \|\mathbf{H}\|_\infty$.

By matrix norm properties, we have $\|\mathbf{H}\|_\infty \equiv \max_{\mathbf{z}\neq\mathbf{0}} \dfrac{\|\mathbf{H}\mathbf{z}\|_\infty}{\|\mathbf{z}\|_\infty} = \max_{\|\mathbf{z}\|_\infty=1}\{\|\mathbf{H}\mathbf{z}\|_\infty\}$. Let $\mathbf{z}$ be an arbitrary vector in $\mathbb{R}^n$ such that $\|\mathbf{z}\|_\infty = 1$. Then it suffices to prove that $\|\mathbf{H}\mathbf{z}\|_\infty < 1$.

[4] $(\mathbf{I} - \omega\mathbf{D}^{-1}|\mathbf{L}|)^{-1} = \sum_{\hat{k}=0}^{n-1}(\omega\mathbf{D}^{-1}|\mathbf{L}|)^{\hat{k}} \geq \mathbf{0}$

[5] A square matrix $\mathbf{A}$ is called monotone if $\det(\mathbf{A}) \neq 0$ and $\mathbf{A}^{-1} \geq \mathbf{0}$. Thus, $\mathbf{A}\mathbf{x} \leq \mathbf{A}\mathbf{y}$ implies $\mathbf{x} \leq \mathbf{y}$ (see [39]). Actually, in this case $\mathbf{I} - \omega\mathbf{D}^{-1}|\mathbf{L}|$ is an $\mathcal{M}$-matrix.

First, we rewrite $\mathbf{H}$ as

$$\begin{aligned}\mathbf{H} &= (\mathbf{I} - \omega\mathbf{D}^{-1}|\mathbf{L}|)^{-1}|\mathbf{I} - \omega\mathbf{D}^{-1}(\mathbf{A} - \mathbf{L})| \\ &= \left(\mathbf{D}^{-1}(\mathbf{D} - \omega|\mathbf{L}|)\right)^{-1}\mathbf{D}^{-1}|\mathbf{D} - \omega(\mathbf{A} - \mathbf{L})| \\ &= (\mathbf{D} - \omega|\mathbf{L}|)^{-1}|\mathbf{D} - \omega(\mathbf{D} + \mathbf{U})| \\ &= (\mathbf{D} - \omega|\mathbf{L}|)^{-1}|(1-\omega)\mathbf{D} - \omega\mathbf{U})|.\end{aligned}$$

For simplicity, let $\tilde{\mathbf{h}} = \mathbf{H}\mathbf{z}$. Then,

$$\begin{aligned}(\mathbf{D} - \omega|\mathbf{L}|)\tilde{\mathbf{h}} &= |(1-\omega)\mathbf{D} - \omega\mathbf{U})|\mathbf{z} \\ &= (|1-\omega|\mathbf{D} + \omega|\mathbf{U}|)\mathbf{z}.\end{aligned} \tag{A.8}$$

In order to prove that $\|\tilde{\mathbf{h}}\|_\infty < 1$, we will show that each entry of $\tilde{\mathbf{h}}$ is less than 1 in absolute value, that is $|\tilde{\mathbf{h}}_i| < 1, i = 1, \ldots, n$.

$\underline{i = 1}$: From (A.8), we have

$$\begin{aligned}|\tilde{\mathbf{h}}_1| &= \frac{|1 - \omega|\mathbf{a}_{1,1}\mathbf{z}_1 + \omega(\sum_{j>1}|\mathbf{a}_{1,j}|\mathbf{z}_j)}{\mathbf{a}_{1,1}} \\ &\leq \frac{|1 - \omega|\mathbf{a}_{1,1}|\mathbf{z}_1| + \omega(\sum_{j>1}|\mathbf{a}_{1,j}||\mathbf{z}_j|)}{\mathbf{a}_{1,1}} \\ &\leq \frac{|1 - \omega|\mathbf{a}_{1,1} + \omega\sum_{j>1}|\mathbf{a}_{1,j}|}{\mathbf{a}_{1,1}}\end{aligned}$$

If $0 < \omega \leq 1$, it follows that

$$\begin{aligned}|\tilde{\mathbf{h}}_1| &\leq \frac{(1-\omega)\mathbf{a}_{1,1} + \omega\sum_{j>1}|\mathbf{a}_{1,j}|}{\mathbf{a}_{1,1}} \\ &= 1 - \omega + \omega\underbrace{\frac{\sum_{j>1}|\mathbf{a}_{1,j}|}{\mathbf{a}_{1,1}}}_{<1} \\ &< 1 - \omega + \omega = 1.\end{aligned}$$

If $1 < \omega < \tilde{\omega}$, it follows that

$$\begin{aligned}
|\tilde{\mathbf{h}}_1| &\leq \frac{(\omega-1)\mathbf{a}_{1,1} + \omega\sum_{j>1}|\mathbf{a}_{1,j}|}{\mathbf{a}_{1,1}} \\
&= \frac{\omega(\sum_{j>1}|\mathbf{a}_{1,j}| + \mathbf{a}_{1,1}) - \mathbf{a}_{1,1}}{\mathbf{a}_{1,1}} \\
&= \omega\frac{\sum_j |\mathbf{a}_{1,j}|}{\mathbf{a}_{1,1}} - 1 \\
&< 2 - 1 = 1,
\end{aligned}$$

for any choice of ω since $\omega < 2\min_i\{\frac{\mathbf{a}_{i,i}}{\sum_j |\mathbf{a}_{i,j}|}\}$.

Now we assume that $|\tilde{\mathbf{h}}_i| < 1$ for each i, $1 \leq i \leq \tilde{k} - 1$, where $2 \leq \tilde{k} \leq n$ We will now prove $|\tilde{\mathbf{h}}_{\tilde{k}}| < 1$. Taking into account (A.8), we have

$$-\omega\big(\sum_{j<\tilde{k}} |\mathbf{a}_{\tilde{k},j}|\tilde{\mathbf{h}}_j\big) + \mathbf{a}_{\tilde{k},\tilde{k}}\tilde{\mathbf{h}}_{\tilde{k}} = |1 - \omega|\mathbf{a}_{\tilde{k},\tilde{k}}\mathbf{z}_{\tilde{k}} + \omega\big(\sum_{j>\tilde{k}} |\mathbf{a}_{\tilde{k},j}|\mathbf{z}_j\big),$$

or equivalently,

$$\begin{aligned}
\tilde{\mathbf{h}}_{\tilde{k}} &= \frac{|1 - \omega|\mathbf{a}_{\tilde{k},\tilde{k}}\mathbf{z}_{\tilde{k}} + \omega\big(\sum_{j>\tilde{k}} |\mathbf{a}_{\tilde{k},j}|\mathbf{z}_j\big) + \omega\big(\sum_{j<\tilde{k}} |\mathbf{a}_{\tilde{k},j}|\tilde{\mathbf{h}}_j\big)}{\mathbf{a}_{\tilde{k},\tilde{k}}} \\
&\leq \frac{|1 - \omega|\mathbf{a}_{\tilde{k},\tilde{k}}|\mathbf{z}_{\tilde{k}}| + \omega\big(\sum_{j>\tilde{k}} |\mathbf{a}_{\tilde{k},j}||\mathbf{z}_j|\big) + \omega\big(\sum_{j<\tilde{k}} |\mathbf{a}_{\tilde{k},j}||\tilde{\mathbf{h}}_j|\big)}{\mathbf{a}_{\tilde{k},\tilde{k}}} \\
&\leq \frac{|1 - \omega|\mathbf{a}_{\tilde{k},\tilde{k}} + \omega\big(\sum_{j>\tilde{k}} |\mathbf{a}_{\tilde{k},j}| + \sum_{j<\tilde{k}} |\mathbf{a}_{\tilde{k},j}|\big)}{\mathbf{a}_{\tilde{k},\tilde{k}}} \\
&= \frac{|1 - \omega|\mathbf{a}_{\tilde{k},\tilde{k}} + \omega\sum_{j\neq\tilde{k}} |\mathbf{a}_{\tilde{k},j}|}{\mathbf{a}_{\tilde{k},\tilde{k}}}.
\end{aligned}$$

We can use the same argument as in the case $i = 1$, obtaining $|\tilde{\mathbf{h}}_{\tilde{k}}| < 1$. By induction, we conclude that $|\tilde{\mathbf{h}}_i| < 1, \forall i, 1 \leq i \leq n$ and thus $\rho(\mathbf{H}) < 1$. □

In the next lemma, we establish the equivalence between the constrained matrix problem (3.29) and computing $\mathbf{x} \in \mathbb{R}^n$ such that

$$\mathbf{x}_i - \max\Big(\mathbf{g}_i, \mathbf{x}_i + \frac{\omega}{\mathbf{a}_{i,i}}(\mathbf{b}_i - \sum_j \mathbf{a}_{i,j}\mathbf{x}_j)\Big) = 0, \qquad 1 \leq i \leq n.$$

LEMMA 4. *Let* $\mathbf{A}, \mathbf{D}$, $\mathbf{L}$ *and* $\mathbf{U}$ *be as in Lemma 2 and* $\omega > 0$. *Then computing a vector* $\mathbf{x}$ *satisfying*

$$\mathbf{x}_i - \max\left(\mathbf{g}_i, \mathbf{x}_i + \frac{\omega}{\mathbf{a}_{i,i}}(\mathbf{b}_i - \sum_j \mathbf{a}_{i,j}\mathbf{x}_j)\right) = 0, \qquad 1 \le i \le n,$$

is equivalent to solving the constrained matrix problem

$$\left\{ \begin{array}{l} \mathbf{Ax} = \mathbf{b} \\ \mathbf{x} \ge \mathbf{g} \end{array} \right\} \textit{ or } \left\{ \begin{array}{l} \mathbf{Ax} > \mathbf{b} \\ \mathbf{x} = \mathbf{g} \end{array} \right\}.$$

Proof. For convenience, we denote $(\mathbf{Ax}-\mathbf{b})_i$ the ith component of $\mathbf{Ax}-\mathbf{b}$, that is $(\mathbf{Ax}-\mathbf{b})_i = \sum_j \mathbf{a}_{i,j}\mathbf{x}_j - \mathbf{b}_i$.

($\Longleftarrow$)

Assume that $\left\{ \begin{array}{l} (\mathbf{Ax}-\mathbf{b})_i = 0 \\ \mathbf{x}_i \ge \mathbf{g}_i \end{array} \right\}$. We have

$$\mathbf{x}_i - \max\left(\mathbf{g}_i, \mathbf{x}_i + \frac{\omega}{\mathbf{a}_{i,i}} \underbrace{(\mathbf{b}_i - \sum_j \mathbf{a}_{i,j}\mathbf{x}_j)}_{0}\right) = \mathbf{x}_i - \underbrace{\max(\mathbf{g}_i, \mathbf{x}_i)}_{\mathbf{x}_i} = 0. \tag{A.9}$$

Assume that $\left\{ \begin{array}{l} (\mathbf{Ax}-\mathbf{b})_i > 0 \\ \mathbf{x}_i = \mathbf{g}_i \end{array} \right\}$. We have

$$\mathbf{x}_i - \max\left(\mathbf{g}_i, \mathbf{x}_i + \frac{\omega}{\mathbf{a}_{i,i}} \underbrace{(\mathbf{b}_i - \sum_j \mathbf{a}_{i,j}\mathbf{x}_j)}_{<0}\right) = \mathbf{x}_i - \mathbf{g}_i = 0. \tag{A.10}$$

($\Longrightarrow$)

First, we prove that $(\mathbf{b}-\mathbf{Ax})_i \le 0, 1 \le i \le n$ by contradiction. Assume otherwise, that for some i, $1 \le i \le n$, $(\mathbf{b}-\mathbf{Ax})_i > 0$. Note that since $\omega > 0$ and $\mathbf{a}_{i,i} > 0$, under this assumption

$$\max\left(\mathbf{g}_i, \mathbf{x}_i + \underbrace{\frac{\omega}{\mathbf{a}_{i,i}}(\mathbf{b}_i - \sum_j \mathbf{a}_{i,j}\mathbf{x}_j)}_{>0}\right) > \mathbf{x}_i.$$

Then we would have

$$\mathbf{x}_i - \max\left(\mathbf{g}_i, \mathbf{x}_i + \frac{\omega}{\mathbf{a}_{i,i}}(\mathbf{b}_i - \sum_j \mathbf{a}_{i,j}\mathbf{x}_j)\right) < 0,$$

a contradiction. Hence $(\mathbf{b} - \mathbf{A}\mathbf{x})_i \leq 0, 1 \leq i \leq n$.

We now consider all possibilities for the $\max\left(\mathbf{g}_i, \mathbf{x}_i + \frac{\omega}{\mathbf{a}_{i,i}}(\mathbf{b}_i - \sum_j \mathbf{a}_{i,j}\mathbf{x}_j)\right)$.

If

$$\mathbf{x}_i + \frac{\omega}{\mathbf{a}_{i,i}}(\mathbf{b}_i - \sum_j \mathbf{a}_{i,j}\mathbf{x}_j) \geq \mathbf{g}_i, \tag{A.11}$$

then

$$\max\left(\mathbf{g}_i, \mathbf{x}_i + \frac{\omega}{\mathbf{a}_{i,i}}(\mathbf{b}_i - \sum_j \mathbf{a}_{i,j}\mathbf{x}_j)\right) = \mathbf{x}_i + \frac{\omega}{\mathbf{a}_{i,i}}(\mathbf{b}_i - \sum_j \mathbf{a}_{i,j}\mathbf{x}_j),$$

thus

$$0 = \mathbf{x}_i - \max\left(\mathbf{g}_i, \mathbf{x}_i + \frac{\omega}{\mathbf{a}_{i,i}}(\mathbf{b}_i - \sum_j \mathbf{a}_{i,j}\mathbf{x}_j)\right) = -\frac{\omega}{\mathbf{a}_{i,i}}(\mathbf{b}_i - \sum_j \mathbf{a}_{i,j}\mathbf{x}_j)$$

whence

$$(\mathbf{A}\mathbf{x} - \mathbf{b})_i = 0.$$

In this case, since $\mathbf{b}_i - \sum_j \mathbf{a}_{i,j}\mathbf{x}_j \leq 0$, from (A.11) we have $\mathbf{x}_i \geq \mathbf{g}_i$.

If

$$\mathbf{x}_i + \frac{\omega}{\mathbf{a}_{i,i}}(\mathbf{b}_i - \sum_j \mathbf{a}_{i,j}\mathbf{x}_j) < \mathbf{g}_i, \tag{A.12}$$

then

$$0 = \mathbf{x}_i - \max\left(\mathbf{g}_i, \mathbf{x}_i + \frac{\omega}{\mathbf{a}_{i,i}}(\mathbf{b}_i - \sum_j \mathbf{a}_{i,j}\mathbf{x}_j)\right) = \mathbf{x}_i - \mathbf{g}_i,$$

or

$$\mathbf{x}_i = \mathbf{g}_i. \tag{A.13}$$

Note that we always have $\mathbf{b}_i - \sum_j \mathbf{a}_{i,j}\mathbf{x}_j \leq 0$ as proved earlier. In this case, we have $\mathbf{b}_i - \sum_j \mathbf{a}_{i,j}\mathbf{x}_j < 0$. Otherwise, if $\mathbf{b}_i - \sum_j \mathbf{a}_{i,j}\mathbf{x}_j = 0$, then it follows from (A.12) that $\mathbf{x}_i < \mathbf{g}_i$, thus contradicts with (A.13). □

LEMMA 5. *Let* $\mathbf{A}$ *be as in Lemma 2. Then the constrained matrix problem (3.29)*

$$\left\{ \begin{array}{l} \mathbf{A}\mathbf{x} = \mathbf{b} \\ \mathbf{x} \geq \mathbf{g} \end{array} \right\} \textit{ or } \left\{ \begin{array}{l} \mathbf{A}\mathbf{x} > \mathbf{b} \\ \mathbf{x} = \mathbf{g} \end{array} \right\}$$

has at most one solution.

Proof. Assume that there are two different solutions $\mathbf{y}$ and $\mathbf{z}$ to problem (3.29). We have

$$\left\{ \begin{array}{l} \mathbf{Ay} = \mathbf{b} \\ \mathbf{y} \geq \mathbf{g} \end{array} \right\} \text{ or } \left\{ \begin{array}{l} \mathbf{Ay} > \mathbf{b} \\ \mathbf{y} = \mathbf{g} \end{array} \right\} \tag{A.14}$$

and

$$\left\{ \begin{array}{l} \mathbf{Az} = \mathbf{b} \\ \mathbf{z} \geq \mathbf{g} \end{array} \right\} \text{ or } \left\{ \begin{array}{l} \mathbf{Az} > \mathbf{b} \\ \mathbf{z} = \mathbf{g} \end{array} \right\}. \tag{A.15}$$

For convenience, we denote $(\mathbf{Ax} - \mathbf{b})_i$ and $(\mathbf{x} - \mathbf{g})_i$ the ith component of $\mathbf{Ax} - \mathbf{b}$ and $\mathbf{x} - \mathbf{g}$, respectively. For an arbitrary index i, $1 \leq i \leq n$, we always have

$$\left\{ \begin{array}{l} (\mathbf{Ay} - \mathbf{b})_i = 0 \\ (\mathbf{y} - \mathbf{g})_i \geq 0 \end{array} \right\} \text{ or } \left\{ \begin{array}{l} (\mathbf{Ay} - \mathbf{b})_i > 0 \\ (\mathbf{y} - \mathbf{g})_i = 0 \end{array} \right\}.$$

Thus it follows that

$$\min\big((\mathbf{Ay} - \mathbf{b})_i, (\mathbf{y} - \mathbf{g})_i\big) = 0.$$

From (A.15), we have $(\mathbf{Az} - \mathbf{b})_i \geq 0$ and $(\mathbf{z} - \mathbf{g})_i \geq 0$. Thus it follows that

$$\begin{aligned} 0 &= \min\big((\mathbf{Ay} - \mathbf{b})_i, (\mathbf{y} - \mathbf{g})_i\big) \\ &\geq \min\big((\mathbf{Ay} - \mathbf{b})_i - (\mathbf{Az} - \mathbf{b})_i, (\mathbf{y} - \mathbf{g})_i - (\mathbf{z} - \mathbf{g})_i\big), \end{aligned}$$

or equivalently,

$$\min\big((\mathbf{A}(\mathbf{y} - \mathbf{z}))_i, (\mathbf{y} - \mathbf{z})_i\big) \leq 0. \tag{A.16}$$

We can interchange the roles of $\mathbf{y}_i$ and $\mathbf{z}_i$ and obtain

$$\min\big((\mathbf{A}(\mathbf{z} - \mathbf{y}))_i, (\mathbf{z} - \mathbf{y})_i\big) \leq \mathbf{0},$$

and hence [6]

$$\max\big((\mathbf{A}(\mathbf{y} - \mathbf{z}))_i, (\mathbf{y} - \mathbf{z})_i\big) \geq \mathbf{0}. \tag{A.17}$$

Note that relations (A.16) and (A.17) are true for any i, $1 \leq i \leq n$. For convenience, let $\mathbf{A}_i$ denote the ith row of matrix $\mathbf{A}$. Now we consider two matrices $\hat{\mathbf{A}}$ and $\tilde{\mathbf{A}}$ constructed as

[6] It can be shown that for any $a, b \in \mathbf{R}$, if $\min(a, b) \leq 0$ then $\max(-a, -b) \geq 0$.

follows:

$$\hat{\mathbf{A}}_i = \begin{cases} \mathbf{A}_i & \text{if } (\mathbf{A}(\mathbf{y}-\mathbf{z}))_i \leq 0 \\ \mathbf{I}_i & \text{if } (\mathbf{A}(\mathbf{y}-\mathbf{z}))_i > 0 \end{cases}$$

and

$$\tilde{\mathbf{A}}_i = \begin{cases} \mathbf{A}_i & \text{if } (\mathbf{A}(\mathbf{y}-\mathbf{z}))_i \geq 0 \\ \mathbf{I}_i & \text{if } (\mathbf{A}(\mathbf{y}-\mathbf{z}))_i < 0. \end{cases}$$

Now consider $\hat{\mathbf{A}}(\mathbf{y}-\mathbf{z})$. By the construction of $\hat{\mathbf{A}}$, we have that

$$(\hat{\mathbf{A}}(\mathbf{y}-\mathbf{z}))_i = \begin{cases} (\mathbf{A}(\mathbf{y}-\mathbf{z}))_i & \text{if } (\mathbf{A}(\mathbf{y}-\mathbf{z}))_i \leq 0 \\ (\mathbf{y}-\mathbf{z})_i & \text{if } (\mathbf{A}(\mathbf{y}-\mathbf{z}))_i > 0 \end{cases}$$

If $(\mathbf{A}(\mathbf{y}-\mathbf{z}))_i \leq 0$, then it is obvious that $(\hat{\mathbf{A}}(\mathbf{y}-\mathbf{z}))_i \leq 0$. If $(\mathbf{A}(\mathbf{y}-\mathbf{z}))_i > 0$, then by inequality (A.16), we must have that $(\mathbf{y}-\mathbf{z})_i \leq 0$ since if $(\mathbf{y}-\mathbf{z})_i > 0$, the inequality (A.16) does not hold. Thus, in all cases, $(\hat{\mathbf{A}}(\mathbf{y}-\mathbf{z}))_i \leq 0$ for any i, $1 \leq i \leq n$. As a result, we have $\hat{\mathbf{A}}(\mathbf{y}-\mathbf{z}) \leq \mathbf{0}$. By the construction of $\hat{\mathbf{A}}$, it is a strictly diagonally dominant $\mathcal{L}$-matrix hence an $\mathcal{M}$-matrix. It follows that $\hat{\mathbf{A}}$ is monotone and thus $\hat{\mathbf{A}}(\mathbf{y}-\mathbf{z}) \leq \mathbf{0}$ implies $\mathbf{y}-\mathbf{z} \leq \mathbf{0}$.

Using a similar line arguments, we obtain $\tilde{\mathbf{A}}(\mathbf{y}-\mathbf{z}) \geq \mathbf{0}$ and that $\tilde{\mathbf{A}}$ is monotone. Thus $\tilde{\mathbf{A}}(\mathbf{y}-\mathbf{z}) \geq \mathbf{0}$ implies $\mathbf{y}-\mathbf{z} \geq \mathbf{0}$. From $\mathbf{y}-\mathbf{z} \leq \mathbf{0}$ and $\mathbf{y}-\mathbf{z} \geq \mathbf{0}$, we obtain $\mathbf{y} = \mathbf{z}$. □

In the next theorem, we establish the convergence of the sequence $\{\mathbf{x}^{(k)}\}$ and show that the limit point of the sequence uniquely solves the constrained matrix problem (3.29).

Theorem 3. *Let* $\mathbf{A}, \mathbf{D}$, $\mathbf{L}$ *and* $\mathbf{U}$ *be as in Lemma 2 and the relaxation factor* ω *such that* $\rho(\mathbf{I}-\omega\mathbf{D}^{-1}|\mathbf{L}|)^{-1}|\mathbf{I}-\omega\mathbf{D}^{-1}(\mathbf{A}-\mathbf{L})|) < 1$. *Then the sequence* $\{\mathbf{x}^{(k)}\}$ *generated by the PSOR algorithm (3.30) converges to the unique solution* $\tilde{\mathbf{x}}$ *of the constrained matrix problem (3.29).*

Proof. By Lemma 2, we have

$$\begin{aligned} |\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}| &\leq \underbrace{(\mathbf{I}-\omega\mathbf{D}^{-1}|\mathbf{L}|)^{-1}|\big(\mathbf{I}-\omega\mathbf{D}^{-1}(\mathbf{A}-\mathbf{L})\big)|}_{\mathbf{H}} |\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}| \\ &\leq \mathbf{H}^2|\mathbf{x}^{(k-1)} - \mathbf{x}^{(k-2)}| \\ &\cdots \\ &\leq \mathbf{H}^k|\mathbf{x}^{(1)} - \mathbf{x}^{(0))}|, \end{aligned} \tag{A.18}$$

and as a result

$$\lim_{k\to\infty} |\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}| = \mathbf{0}. \tag{A.19}$$

From (A.18), we find that for any $k, m \geq 0$,

$$\begin{aligned}
|\mathbf{x}^{(k+m)} - \mathbf{x}^{(k)}| &\leq \sum_{\tilde{k}=1}^{m} |\mathbf{x}^{(k+\tilde{k})} - \mathbf{x}^{(k+\tilde{k}-1)}| \\
&\leq \sum_{\tilde{k}=1}^{m} \mathbf{H}^{\tilde{k}} |\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}| \\
&\leq (\mathbf{I}-\mathbf{H})^{-1}\mathbf{H}|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}| \\
&\leq (\mathbf{I}-\mathbf{H})^{-1}\mathbf{H}^{k}|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}| \\
&= \mathbf{H}^{k} \underbrace{(\mathbf{I}-\mathbf{H})^{-1}|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}|}_{\text{constant vector}},
\end{aligned}$$

taking into account that

$$(\mathbf{I}-\mathbf{H})^{-1} = \sum_{\tilde{k}=1}^{\infty} \mathbf{H}^{\tilde{k}} \geq \mathbf{0}, \qquad \sum_{\tilde{k}=1}^{m-1} \mathbf{H}^{\tilde{k}} \leq (\mathbf{I}-\mathbf{H})^{-1},$$

since $\mathbf{H} \geq \mathbf{0}$ and $\rho(\mathbf{H}) < 1$. Thus the sequence $\{\mathbf{x}^{(k)}\}$ is a Cauchy sequence, [7] hence converges to some limit point $\tilde{\mathbf{x}}$, [8] that is

$$\mathbf{x}^{(k)} \to \tilde{\mathbf{x}} \qquad \text{as} \qquad k \to \infty. \tag{A.20}$$

Note that from (A.19) and (A.20), we have

$$\begin{aligned}
&\mathbf{x}_i^{(k)} \to \mathbf{x}_i^{(k-1)}, \\
&\mathbf{x}_i^{(k)} \to \tilde{\mathbf{x}}_i, \\
&\qquad \text{as} \quad k \to \infty,
\end{aligned} \tag{A.21}$$

for $1 \leq i \leq n$. Substituting $\mathbf{x}_i^{(k)}$ into the PSOR algorithm (3.30) and taking the limit with

[7] A sequence $\{\mathbf{s}^{(k)}\}$ is called a Cauchy sequence if for each $\epsilon > 0$ there exists a number N such that $m, n > N$ implies $|\mathbf{s}^{(m)} - \mathbf{s}^{(n)}| < \epsilon$ (see Definition 10.8 in [29]).

[8] A sequence is a convergent sequence if and only if it is a Cauchy sequence (see Theorem 10.11 in [29]).

respect to k, we have

$$\begin{aligned}\tilde{\mathbf{x}}_i = \lim_{k\to\infty} \mathbf{x}_i^{(k)} &= \lim_{k\to\infty} \max\Big(\mathbf{g}_i, \mathbf{x}_i^{(k-1)} + \frac{\omega}{\mathbf{a}_{i,i}}(\mathbf{b}_i - \sum_{j<i} \mathbf{a}_{i,j}\mathbf{x}_j^{(k)} - \sum_{j>i} \mathbf{a}_{i,j}\mathbf{x}_j^{(k-1)} - \mathbf{a}_{i,i}\mathbf{x}_i^{(k-1)})\Big) \\ &= \max\Big(\mathbf{g}_i, \tilde{\mathbf{x}}_i + \frac{\omega}{\mathbf{a}_{i,i}}(\mathbf{b}_i - \sum_{j<i} \mathbf{a}_{i,j}\tilde{\mathbf{x}}_i - \sum_{j>i} \mathbf{a}_{i,j}\tilde{\mathbf{x}}_i - \mathbf{a}_{i,i}\tilde{\mathbf{x}}_i)\Big) \\ &= \max\Big(\mathbf{g}_i, \tilde{\mathbf{x}}_i + \frac{\omega}{\mathbf{a}_{i,i}}(\mathbf{b}_i - \sum_{j} \mathbf{a}_{i,j}\tilde{\mathbf{x}}_j)\Big),\end{aligned}$$

for $1 \le i \le n$, due to (A.21). Thus we know that the limit point $\tilde{\mathbf{x}}$ of the sequence $\{\mathbf{x}^{(k)}\}$ satisfies

$$\mathbf{x}_i = \max\Big(\mathbf{g}_i, \mathbf{x}_i + \frac{\omega}{\mathbf{a}_{i,i}}(\mathbf{b}_i - \sum_{j} \mathbf{a}_{i,j}\mathbf{x}_j)\Big),$$

or equivalently,

$$\mathbf{x}_i - \max\Big(\mathbf{g}_i, \mathbf{x}_i + \frac{\omega}{\mathbf{a}_{i,i}}(\mathbf{b}_i - \sum_{j} \mathbf{a}_{i,j}\mathbf{x}_j)\Big) = 0. \tag{A.22}$$

By Lemma 4 and (A.22), we conclude that the sequence $\{\mathbf{x}^{(k)}\}$ generated by the PSOR algorithm (3.30) converges to a solution of the constrained matrix problem (3.29). The uniqueness of the solution is guaranteed by Lemma 5.

□

Bibliography

[1] G. Barone-Adesi and R.E. Whaley. Efficient analytic approximations of American option values. *Journal of Finance*, 42:301–320, 1987.

[2] F. Black and M. Scholes. The pricing of options and corporate liabilities. *Journal of Political Economics*, 81:637–659, 1973.

[3] Michael J. Brennan and Eduardo Schwartz. The valuation of American put options. *Journal of Finance*, 32:449–462, 1977.

[4] M. Broadie and J. Detemple. Recent advances in numerical methods for pricing deriavtives securities. In *Numerical Methods in Finance*, pages 43–66. Cambridge University Press, 1997.

[5] Russel E. Caflisch and Suneal Chaudhary. Monte Carlo methods for American options. In *Proceedings of the 36th conference on Winter simulation*, pages 1656 – 1660, Washington, D.C, December 2004. Association for Computing Machinery.

[6] Graham F. Carey and Hung T. Dinh. Grading functions and mesh redistribution. *SIAM J. Numer. Anal.*, 22(5):1028–1040, 1985.

[7] C. C. Christara and K. S. Ng. Adaptive techniques for spline collocation. *Computing*, 76(3):259 – 277, 2006.

[8] J. Cox, S. Ross, and M. Rubinstein. Option pricing: a simplified approach. *Journal of Financial Economics*, 7:229–263, 1979.

[9] Colin W. Cryer. The solution of a quadratic programming problem using systematic overrelaxation. *SIAM Journal on Control and Optimization*, 9:385–392, 1971.

[10] C. de Boor. Good approximation by splines with variable knots II. Lecture notes in Mathematics, 1974.

[11] Y. d'Halluin, P. A. Forsyth, and K. Vetzal. Robust numerical methods for contingent claims under jump diffusion processes. *IMA Journal of Numerical Analysis*, 25:87–112, 2005.

[12] K. Eriksson and C. Johnson. Adaptive finite element methods for parabolic problem. I. A linear model problem. *SIAM J. Numer. Anal.*, 28:43–77, 1991.

[13] K. Eriksson and C. Johnson. Adaptive finite element methods for parabolic problem. II. Optimal error estimates in $L_\infty L_2$ and $L_\infty L_\infty$. *SIAM J. Numer. Anal.*, 32:706–740, 1995.

[14] P. A. Forsyth and K. Vetzal. Quadratic convergence for valuing American options using a penalty method. *SIAM J. Sci. Comput.*, 23(6):2095–2122, 2002.

[15] Michael C. Fu, Scott B. Laprise, Dilip B. Madan, Yi Su, and Rongwen Wu. Pricing American options: A comparison of Monte Carlo simulation approaches. *Journal of Computational Finance*, 4(3):39–88, Spring, 2001.

[16] W. Hackbush. *Iterative Solution of Large Sparse Systems of Equations*. Springer, 1993.

[17] John C. Hull. *Options, Futures, and Other Derivatives*. Prentice Hall, sixth edition, 2006.

[18] R. Kangro and R. Nicolaides. Far field boundary conditions for Black-Scholes equations. *SIAM J. Numer. Anal.*, 38(4):1357–1368, 2000.

[19] M.D. Koulisianis and T.S. Papatheodorou. Improving projected successive overrelaxation method for linear complementarity problems. *Applied Numerical Mathematics*, 45:29–40, 2003.

[20] Dongyi Li. On convergence of numerical methods for pricing convertible bonds. Master's thesis, University of Toronto, Toronto, Ontario, Canada, 2007.

[21] Lucy Xingwen Li. Pricing convertible bonds using Partial Differential Equations. Master's thesis, University of Toronto, Toronto, Ontario, Canada, 2005.

[22] F. A. Longstaff and E. S. Schwartz. Valuing American options by simulation: A simple least-squares approach. *The Review of Financial Study*, 14:113–149, 2001.

[23] L.W. MacMillan. Analytic approximation for the American put option. *Advanced Futures Options Research*, 1:119–139, 1986.

[24] Robert C. Merton. The theory of rational option pricing. *Bell Journal of Economics and Management Science*, 4:141–183, 1973.

[25] B. Nielsen, O. Skavhaug, and A. Tveito. Penalty and front-fixing methods for the numerical solution of American option problems. *Journal of Computational Finance*, 5(4):69–97, 2002.

[26] J. M. Ortega and Rheinboldt. *Iterative solution of nonlinear equations in several variables*. Academic Press, New York, London, 1970.

[27] D. M Pooley, K. R. Verzal, and P. A. Forsyth. Convergence remedies for non-smooth payoffs in option pricing. *Journal of Computational Finance*, 6:25–40, Summer, 2003.

[28] R. Rannacher. Finite element solution of diffusion problems with irregular data. *Numerische Mathematik*, 43(2):309–327, 1984.

[29] Kenneth A. Ross. *Elementary analysis : the theory of calculus*. Springer-Verlag, New York, 1980.

[30] Rudiger Seydel. *Tools for Computational Finance*. Springer, 2002.

[31] R. Stamica, D. Sevcovic, and John Chadam. The early exercise boundary for the American put near expiry: Numerical approximation. *Canada Applied Mathematics*, 7(4):427–444, 1999.

[32] J.C. Strikwerda. *Finite Difference Schemes and Partial Differential Equations*. Society for Industrial and Applied Mathematics, Philadelphia, second edition, 2004.

[33] D. Tavella and C. Randall. *Pricing financial instruments: The finite difference method*. John Wiley & Sons, Chichester, 2000.

[34] R.S Varga. *Matrix Iterative Analysis*. Springer, second edition, 2000.

[35] R. Wang, P. Keast, and P. Muir. A high-order global spatially adaptive collcation method for 1-D parabolic PDEs. *Applied Numerical Analysis*, 50:239–260, 2004.

[36] A.B. White. On selection of equidistributing meshed for two-point boundary value problems. *SIAM J. Numer. Anal.*, 16:472–502, 1979.

[37] Paul Wilmott, Sam Howison, and Jeff Dewynne. *Mathematics of Financial Derivatives*. Cambridge Univeristy Press, 1995.

[38] H. Windcliff, P. A . Forsyth, and K. R. Vetzal. Analysis of the stability of the linear boundary condition for the Black-Scholes equation. *Journal of Computational Finance*, 8:65–92, Fall, 2004.

[39] D. M. Young. *Iterative solution of large linear system*. Academic Press, New York, 1971.

[40] R. Zvan, P. A. Forsyth, and K. Vetzal. Penalty methods for American options with stochastic volatility. *Journal of Computational and Applied Mathematics*, 91:199–218, 1998.

[41] R. Zvan, P. A. Forsyth, and K. Vetzal. Robust numerical methods for PDE models of Asian options. *Journal of Computational Finance*, 2:39–78, 1998.

[42] R. Zvan, P. A. Forsyth, and K. Vetzal. Swing low, swing high. *RISK*, 11:71–75, 1998.