Thesis for the Degree of

Doctor of Philosophy

Mathematical model and numerical simulation in computational finance

by

Jeong Darae

Department of Mathematics

Graduate School

Korea University

December, 2012

김 준 석 敎 授 指 導
碩 士 學 位 論 文

Mathematical model and numerical simulation in computational finance

이 論文을 理學博士學位 論文으로 提出함.

2012 년 12 月

高麗大學校 大學院
數學科
丁 다 래

丁다래의 理學博士學位 論文
審査를 完了함.

2012년 12 月 日

委員長 김준석 (印)

委員 안인경 (印)

委員 황운재 (印)

委員 박춘재 (印)

委員 권기운 (印)

Contents

Abstract

The primary purpose of this thesis is to explore the numerical methods for computational finance.

We present an efficient and accurate finite-difference method for computing Black-Scholes partial differential equations with multi-underlying assets. We directly solve Black-Scholes equations without transformations of variables. We provide computational results showing the performance of the method for two underlying asset option pricing problems. And the finite difference methods are applied and the resulting linear system is solved by biconjugate gradient stabilized, alternating direction implicit, operator splitting, and multigrid methods. The performance of these methods is compared for two asset option problems based on two-dimensional Black-Scholes equations. Numerical results show that the operator splitting method is the most efficient among these solvers to get the same level of accuracy.

Also, we present the numerical valuation of the two-asset step-down equity-linked securities (ELS) option by using the operator-splitting method (OSM). We provide a detailed numerical algorithm and computational results showing the performance of the method for two underlying asset option pricing problems

such as cash-or-nothing and step-down ELS. Final option value of two-asset step-down ELS is obtained by a weighted average value using probability which is estimated by performing a MC simulation.

Finally, we propose two adaptive techniques for solving Black–Scholes model. By using an adaptive grid technique which is based on a far-field boundary position of the equation, we present an accurate and efficient numerical method for the Black-Scholes equations. The results show that the computational time of the new adaptive grid method is reduced substantially when compared to that of a uniform grid method. And we present an adaptive mesh refinement method which is considered grid resolutions and time steps. We computationally represent that the proposed adaptive schemes give mesh better efficiency than the standard FDM.

Acknowledgments

I would like to express my sincere gratitude to Prof. Junseok Kim for their support and for giving me the opportunity to work at the Laboratory of Scientific Computation, where I have enjoyed a friendly atmosphere and access to excellent research facilities. My appreciation also goes to Prof. Woonjae Hwang, Prof. Inkyung Ahn, Prof. Chunjae Park, and Prof. Kiwoon Kwon for serving on my committee.

Also, I offer my regards and blessings to all of those who supported me in any respect during the completion of this thesis. And I consider myself very lucky to work in my research group.

Finally, I would like to express my deepest appreciation to my parents for their support throughout my life.

Chapter 1

Introduction

1.1. Black–Scholes model

Financial options pricing model developed by Black and Scholes [6] in 1973 and extended by Merton [53]. Black, Scholes, and Merton derived a parabolic second order PDE for the valuation of an European option under the no-arbitrage assumption as well as the assumption that the price of its underlyings have log-normal distributions.

Let $s_i(t),\ i = 1, 2, ..., d$ denote the value of the underlying i-th asset and $u(\mathbf{s}, t)$ denote the price of the option. Here, $\mathbf{s} = (s_1, s_2, ..., s_d)$. In the Black–Scholes model [6], each underlying asset s_i satisfies the following stochastic differential equation:

$$ds_i(t) = \mu_i s_i(t)dt + \sigma_i s_i(t)dW_i,\ i = 1, 2, ..., d,$$

where $\mu_i,\ \sigma_i$, and $W_i(t)$ are the expected instantaneous rate of return, constant volatility, and standard Brownian motion on the underlying asset s_i, respectively. And the term dW contains the randomness which is certainly a feature of asset prices and is assumed to be a Wiener process. The Wiener processes are correlated by $\langle dW_i dW_j \rangle = \rho_{ij} dt$.

Then by using the Ito's lemma and non-arbitrage principle, the generalized Black–Scholes (BS) PDE can be obtained

$$\frac{\partial u(\mathbf{s},t)}{\partial t} + \sum_{i=1}^{d} r s_i \frac{\partial u(\mathbf{s},t)}{\partial s_i} + \frac{1}{2}\sum_{i,j=1}^{d} \rho_{ij}\sigma_i\sigma_j s_i s_j \frac{\partial^2 u(\mathbf{s},t)}{\partial s_i \partial s_j} - ru(\mathbf{s},t) = 0, \quad (1.1)$$

$$u(\mathbf{s},T) = \Lambda(\mathbf{s}), \quad (1.2)$$

where $r > 0$ is a constant riskless interest rate.

In this thesis, we use the original Black–Scholes model with two underlying assets to keep this presentation simple. However, we can easily extend the current Eq. (1.3) for more than two underlying assets [4]. Let us consider two assets case as $x = s_1$ and $y = s_2$. Let us first convert the given backward equation (1.1) to the following forward equation by a change of variable $\tau = T - t$, $u(x, y, \tau) = u(s_1, s_2, T - \tau)$

$$\frac{\partial u}{\partial \tau} = \frac{1}{2}\sigma_1^2 x^2 \frac{\partial^2 u}{\partial x^2} + \sigma_1\sigma_2\rho xy\frac{\partial^2 u}{\partial xy} + \frac{1}{2}\sigma_2^2 y^2 \frac{\partial^2 u}{\partial y^2} + rx\frac{\partial u}{\partial x} + ry\frac{\partial u}{\partial y} - ru \quad (1.3)$$

$$\text{for} \quad (x, y, \tau) \in \Omega \times (0, T],$$

$$u(x, y, 0) = \Lambda(x, y).$$

For example, for the option on the maximum of two assets, the payoff function is

$$\Lambda(x, y) = \max(\max(x, y) - K, 0)$$

with a given strike price $K > 0$; see [34].

As the financial market gets complex and diverse, there have been various types of exotic options in the market. Because it is not always possible to find the analytic solution of the Black–Scholes equation with the exotic terminal and the boundary conditions, it is necessary to apply numerical methods to obtain the values of exotic options.

Therefore methods for option pricing have been discovered and improved by many scholars. Details can be found in reference [14] which is a good review of valuation models and applications to the option pricing from its origins to the present.

Despite the real progress, however, because a closed-form solution cannot be obtained or formulas for the exact solutions are too difficult to be practically usable, numerical solution has been a natural way to solve the problem in financial engineering [33].

To obtain an approximation of the option value, option pricing problems have been solved by the simulation-based methods [21, 22, 48], the lattice methods [21, 23, 30] and by the finite difference method [13, 18, 24, 30, 42, 66, 68, 69, 75, 77], delaying the study and the application of other numerical methods like the finite elements method [2, 26, 70, 84, 85, 87] and finite volume method [29, 88], which are widely documented and used in other fields of science and engineering for decades.

In this thesis, we propose several efficient and accurate numerical methods for evaluating option problems.

1.2. Outline of the thesis

The outline of this thesis is as following.

In this chapter 2, we present an efficient and accurate finite difference method for computing Black–Scholes partial differential equations with multi-underlying assets using a multigrid solver. We provide computational results showing the performance of the method for option pricing problems with two underlying assets.

In this chapter 3, we perform a comparison study of alternating direction implicit (ADI) and operator splitting (OS) methods on multi-dimensional Black–Scholes option pricing models. The ADI method is used extensively in mathematical finance for numerically solving multi-asset option pricing problems. However, numerical results from the ADI scheme show oscillatory solution behaviors with nonsmooth payoffs or discontinuous derivatives at the exercise price with large time steps. Most option pricing problems have nonsmooth payoffs or discontinuous derivatives at the exercise price. In the ADI scheme, there are source terms which include y-derivatives when we solve x-derivative involving equations. Then, due to the nonsmooth payoffs, source term contains abrupt changes which are not in the range of implicit discrete operator and this leads to difficulty in solving the problem. On the other hand, the OS method does not contain the other variable's derivatives in the source term. We provide computational results showing the performance of the methods for two underlying asset option pricing problems. The results show that the OS method is very efficient and gives better accuracy and robustness than the ADI method with large time steps.

In this chapter 4, we perform a comparison of finite difference schemes for solving the two dimensional Black–Scholes equation. We discretise the equation in space and time, and then solve a system of linear equations using the biconjugate gradient stabilized, operator splitting, and multigrid methods. The performance is presented, and results from different schemes are compared in two asset option problems based on the two dimensional Black–Scholes equation. Numerical results indicate that the operator splitting method results in a better performance among these solvers with the same level of accuracy.

In this chapter 5, we present an accurate and efficient numerical method for solving the Black–Scholes equation. The method uses an adaptive grid technique which is based on a far-field boundary position of the equation and also the Peclet condition. The algorithm for the automatic adaptive grid generation is: First, for a given error tolerance, we determine a priori a suitable far-field boundary location using the mathematical model parameters. Second, put a uniform fine grid around the non-smooth points of the payoff such as a strike price and a non-uniform grid in the remaining regions. Numerical tests are presented to demonstrate the accuracy and efficiency of the proposed method. The results show that the computational times using the new adaptive grid method are reduced substantially when compared to those of a uniform grid method with a similar magnitude of error.

In this chapter 6, this work presents the numerical valuation of the two-asset step-down equity-linked securities (ELS) option by using the operator splitting method (OSM). The ELS is one of the most popular financial options. The value of ELS option can be modeled by a modified Black–Scholes partial differential equation. However, regardless of whether there is a closed-form solution, it is difficult and not efficient to evaluate the solution because such a solution would be represented by multiple integrations. Thus, a fast and accurate numerical algorithm is needed to value the price of the ELS option. This chapter uses a finite difference method to discretize the governing equation and applies the OSM to solve the resulting discrete equations. The OSM is very robust and accurate in evaluating finite difference discretizations. We provide a detailed numerical algorithm and computational results showing the performance of the method for two underlying asset option pricing problems such as cash-or-nothing and

step-down ELS. Final option value of two-asset step-down ELS is obtained by a weighted average value using probability which is estimated by performing a MC simulation.

In this chapter 7, we consider the adaptive multigrid method for solving the Black–Scholes equation as the numerical technique to improve the efficiency of the option pricing. Adaptive meshing is generally regarded as an indispensable tool because of reduction of the computational costs which are needed to obtain finite difference solutions. Therefore, in this chapter, the Black–Scholes equation is discretized using a Crank–Nicolson scheme on block-structured adaptively refined rectangular meshes. And the resulting discrete of equations is solved by a fast solver such as an multigrid method. Numerical simulations are implemented to confirm the efficiency of the adaptive multigrid technique. In particular, through the comparison of computational results on adaptively refined mesh and uniform mesh, we show that adaptively refined mesh solver is superior to a standard method.

Finally, Chapter 8 summarizes the results and gives some recommendations for future work.

Chapter 2

Multigrid method for Black–Scholes equations

In this chapter, we present an efficient and accurate finite difference method for computing Black–Scholes partial differential equations with multi-underlying assets using a multigrid solver. We provide computational results showing the performance of the method for option pricing problems with two underlying assets.

2.1. Introduction

The analytic solutions of Eqs. (1.1) and (1.2) for exotic options are very limited. Therefore, we need to rely on a numerical approximation. To obtain an approximation of the option value, we can compute a solution of Black–Scholes PDEs (1.1) and (1.2) using a finite difference method (FDM) [24, 66, 68, 69, 77].

We apply the FDM to the equation over a truncated finite domain. The original asymptotic infinite boundary conditions are shifted to the ends of the truncated finite domain. To avoid generating large errors in the solution due to this approximation of the boundary conditions, the truncated domain must be large enough resulting in large computational costs. In this chapter, we propose an efficient and accurate FDM to directly solve the Black–Scholes PDEs (1.1) and (1.2) without transformations of variables.

2.2. Discretization with finite differences

A finite difference method is a common numerical method that has been used by many researchers in computational finance. For an introduction to these methods we recommend the books [24, 66, 69, 68, 77]. They all introduce the concept of finite differences for option pricing and provide basic knowledge needed for a simple implementation of the method. An approach for the Black–Scholes option problem is to use an efficient solver such as the Bi-CGSTAB (Biconjugate gradient stabilized) method [60, 64, 74], GMRES (Generalized minimal residual algorithm) method [56, 63], ADI (Alternating direction implicit) method [19, 24], and the OS (Operator splitting) method [24, 38].

Let us first discretize the given computational domain $\Omega = (0, L) \times (0, M)$ as a uniform grid with a space step $h = L/N_x = M/N_y$ and a time step $\Delta t = T/N_t$. Let us denote the numerical approximation of the solution by

$$u_{ij}^n \equiv u(x_i, y_j, t^n) = u\left((i - 0.5)h, (j - 0.5)h, n\Delta t\right),$$

where $i = 1, \ldots, N_x$ and $j = 1, \ldots, N_y$.

And among several possible boundary conditions such as Neumann [24, 33], Dirichlet, linear, and PDE [24, 68] that can be used for these kinds of problems, we use a linear boundary condition on all boundaries, i.e.,

$$\frac{\partial^2 u}{\partial x^2}(0, y, \tau) = \frac{\partial^2 u}{\partial x^2}(L, y, \tau) = \frac{\partial^2 u}{\partial y^2}(x, 0, \tau) = \frac{\partial^2 u}{\partial y^2}(x, M, \tau) = 0,$$
$$\forall \tau \in [0, T], \text{ for } 0 \le x \le L, 0 \le y \le M.$$

We use a cell centered discretization since we use a linear boundary condition. By applying the implicit time scheme and centered difference for space derivatives

to Eq. (1.3), we have

$$\frac{u_{ij}^{n+1} - u_{ij}^{n}}{\Delta t} = L_{BS} u_{ij}^{n+1}, \tag{2.1}$$

where the discrete difference operator L_{BS} is defined by

$$\begin{aligned} L_{BS} u_{ij}^{n+1} = & \frac{(\sigma_1 x_i)^2}{2} \frac{u_{i-1,j}^{n+1} - 2u_{ij}^{n+1} + u_{i+1,j}^{n+1}}{h^2} + \frac{(\sigma_2 y_j)^2}{2} \frac{u_{i,j-1}^{n+1} - 2u_{ij}^{n+1} + u_{i,j+1}^{n+1}}{h^2} \\ & + \sigma_1 \sigma_2 \rho x_i y_j \frac{u_{i+1,j+1}^{n+1} + u_{i-1,j-1}^{n+1} - u_{i-1,j+1}^{n+1} - u_{i+1,j-1}^{n+1}}{4h^2} \\ & + r x_i \frac{u_{i+1,j}^{n+1} - u_{i-1,j}^{n+1}}{2h} + r y_j \frac{u_{i,j+1}^{n+1} - u_{i,j-1}^{n+1}}{2h} - r u_{ij}^{n+1}. \end{aligned}$$

2.3. A multigrid method

Multigrid methods belong to the class of fastest iterations, because their convergence rate is independent of the space step size [32]. In order to explain clearly the steps taken during a single V-cycle, we focus on a numerical solution on a 16×16 mesh. We define discrete domains, $\Omega_3,\ \Omega_2,\ \Omega_1$, and Ω_0, where

$$\Omega_k = \{(x_{k,i} = (i - 0.5)h_k, y_{k,j} = (j - 0.5)h_k) | 1 \le i, j \le 2^{k+1} \text{ and } h_k = 2^{3-k}h\}.$$

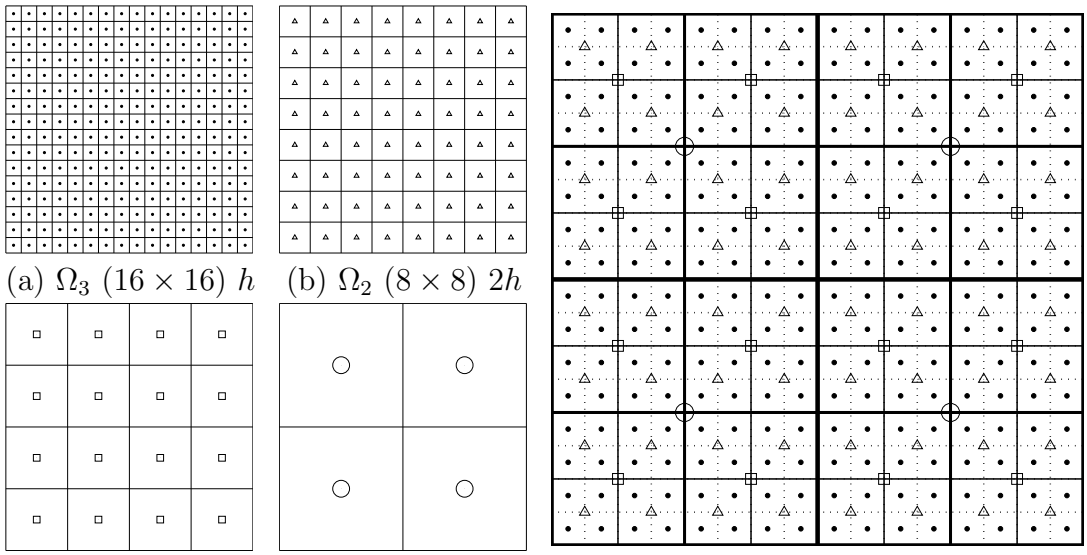
Ω_{k-1} is coarser than Ω_k by a factor of 2. The multigrid solution of the discrete BS Eq. (2.1) makes use of a hierarchy of meshes ($\Omega_3,\ \Omega_2,\ \Omega_1$, and Ω_0) created by successively coarsening the original mesh, Ω_3 as shown in Fig. 2.1.

A pointwise Gauss–Seidel relaxation scheme is used as the smoother in the multigrid method. We use a notation u_k^n as a numerical solution on the discrete domain Ω_k at time $t = n\Delta t$. The algorithm of the multigrid method for solving the discrete BS Eq. (2.1) is as follows. We rewrite the above Eq. (2.1) by

$$L_3(u_{3,ij}^{n+1}) = \phi_{3,ij}^{n} \text{ on } \Omega_3, \tag{2.2}$$

where

$$L_3(u_{3,ij}^{n+1}) = u_{3,ij}^{n+1} - \Delta t L_{BS3} u_{3,ij}^{n+1} \text{ and } \phi_{3,ij}^{n} = u_{3,ij}^{n}.$$



(a) Ω_3 (16×16) h (b) Ω_2 (8×8) $2h$

(c) Ω_1 (4×4) $4h$ (d) Ω_0 (2×2) $8h$ (e)

FIGURE 2.1. (a), (b), (c), and (d) are a sequence of coarse grids starting with $h = L/N_x$. (e) is a composition of grids, Ω_3, Ω_2, Ω_1, and Ω_0.

Given the numbers, ν_1 and ν_2, of pre- and post- smoothing relaxation sweeps, an iteration step for the multigrid method using the V-cycle is formally written as follows [72]. That is, starting an initial condition u_3^0, we want to find u_3^n for $n = 1, 2, \cdots$. Given u_3^n, we want to find the u_3^{n+1} solution that satisfies Eq. (2.1). At the very beginning of the multigrid cycle the solution from the previous time step is used to provide an initial guess for the multigrid procedure. First, let $u_3^{n+1,0} = u_3^n$.

Multigrid cycle

$$u_k^{n+1,m+1} = MGcycle(k, u_k^{n+1,m}, L_k, \phi_k^n, \nu_1, \nu_2).$$

That is, $u_k^{n+1,m}$ and $u_k^{n+1,m+1}$ are the approximations of u_k^{n+1} before and after an MGcycle. Now, define the MGcycle.

Step 1) Presmoothing

$$\bar{u}_k^{n+1,m} = SMOOTH^{\nu_1}(u_k^{n+1,m}, L_k, \phi_k^n),$$

means performing ν_1 smoothing steps with the initial approximation $u_k^{n+1,m}$, source terms ϕ_k^n, and a $SMOOTH$ relaxation operator to get the approximation $\bar{u}_k^{n+1,m}$. Here, we derive the smoothing operator in two dimensions.

Now, we derive a Gauss–Seidel relaxation operator. First, we rewrite Eq. (2.2) as

$$\begin{aligned} u_{k,ij}^{n+1} \;=\; & \left[\phi_{k,ij}^n + \Delta t \left(\frac{(\sigma_1 x_{k,i})^2}{2} \frac{u_{k,i-1,j}^{n+1} + u_{k,i+1,j}^{n+1}}{h_k^2} + \frac{(\sigma_2 y_{k,j})^2}{2} \frac{u_{k,i,j-1}^{n+1} + u_{k,i,j+1}^{n+1}}{h_k^2} \right.\right. \\ & + \sigma_1 \sigma_2 \rho x_{k,i} y_{k,j} \frac{u_{k,i+1,j+1}^{n+1} + u_{k,i-1,j-1}^{n+1} - u_{k,i-1,j+1}^{n+1} - u_{k,i+1,j-1}^{n+1}}{4h_k^2} \\ & \left.\left. + \; r x_{k,i} \frac{u_{k,i+1,j}^{n+1} - u_{k,i-1,j}^{n+1}}{2h_k} + r y_{k,j} \frac{u_{k,i,j+1}^{n+1} - u_{k,i,j-1}^{n+1}}{2h_k} \right) \right] \Big/ \\ & \left[1 + \Delta t \left(\frac{(\sigma_1 x_{k,i})^2 + (\sigma_2 y_{k,j})^2}{h_k^2} + r \right) \right]. \end{aligned} \tag{2.3}$$

Next, we replace $u_{k,\alpha\beta}^{n+1}$ in Eq. (2.3) with $\bar{u}_{k,\alpha\beta}^{n+1,m}$ if $(\alpha < i)$ or $(\alpha = i$ and $\beta \le j)$, otherwise with $u_{k,\alpha\beta}^{n+1,m}$, i.e.,

$$\begin{aligned} \bar{u}_{k,ij}^{n+1,m} \;=\; & \left[\phi_{k,ij}^n + \Delta t \left(\frac{(\sigma_1 x_{k,i})^2}{2} \frac{\bar{u}_{k,i-1,j}^{n+1,m} + u_{k,i+1,j}^{n+1,m}}{h_k^2} + \frac{(\sigma_2 y_{k,j})^2}{2} \frac{\bar{u}_{k,i,j-1}^{n+1,m} + u_{k,i,j+1}^{n+1,m}}{h_k^2} \right.\right. \\ & + \sigma_1 \sigma_2 \rho x_{k,i} y_{k,j} \frac{u_{k,i+1,j+1}^{n+1,m} + \bar{u}_{k,i-1,j-1}^{n+1,m} - \bar{u}_{k,i-1,j+1}^{n+1,m} - u_{k,i+1,j-1}^{n+1,m}}{4h_k^2} \\ & \left.\left. + \; r x_{k,i} \frac{u_{k,i+1,j}^{n+1,m} - \bar{u}_{k,i-1,j}^{n+1,m}}{2h_k} + r y_{k,j} \frac{u_{i,j+1}^{n+1,m} - \bar{u}_{k,i,j-1}^{n+1,m}}{2h_k} \right) \right] / \\ & \left[1 + \Delta t \left(\frac{(\sigma_1 x_{k,i})^2 + (\sigma_2 y_{k,j})^2}{h_k^2} + r \right) \right]. \end{aligned} \tag{2.4}$$

Therefore, in a multigrid cycle, one smooth relaxation operator step consists of solving Eq. (2.4) given above for $1 \le i \le 2^{k-3} N_x$ and $1 \le j \le 2^{k-3} N_y$.

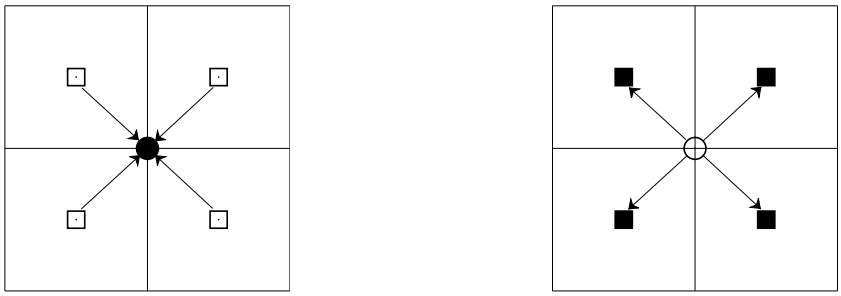
Step 2) Coarse grid correction

• Compute the defect: $\bar{d}_k^m = \phi_k^n - L_k(\bar{u}_k^{n+1,m})$.

• Restrict the defect and $\bar{u}_k^m$: $\bar{d}_{k-1}^m = I_k^{k-1}\bar{d}_k^m$

The restriction operator I_k^{k-1} maps k-level functions to $(k-1)$-level functions as shown in Fig. 2.2(a).

$$\begin{aligned} d_{k-1}(x_i, y_j) &= I_k^{k-1} d_k(x_i, y_j) \\ &= \frac{1}{4}[d_k(x_{i-\frac{1}{2}}, y_{j-\frac{1}{2}}) + d_k(x_{i-\frac{1}{2}}, y_{j+\frac{1}{2}}) + d_k(x_{i+\frac{1}{2}}, y_{j-\frac{1}{2}}) + d_k(x_{i+\frac{1}{2}}, y_{j+\frac{1}{2}})]. \end{aligned}$$



(a) (b)

FIGURE 2.2. Transfer operators: (a) restriction and (b) interpolation.

• Compute an approximate solution $\hat{u}_{k-1}^{n+1,m}$ of the coarse grid equation on Ω_{k-1}, i.e.

$$L_{k-1}(u_{k-1}^{n+1,m}) = \bar{d}_{k-1}^m. \tag{2.5}$$

If $k = 1$, we use a direct or fast iteration solver for (2.5). If $k > 1$, we solve (2.5) approximately by performing k-grid cycles using the zero grid function as an initial approximation:

$$\hat{v}_{k-1}^{n+1,m} = MGcycle(k-1, 0, L_{k-1}, \bar{d}_{k-1}^m, \nu_1, \nu_2).$$

• Interpolate the correction: $\hat{v}_k^{n+1,m} = I_{k-1}^k \hat{v}_{k-1}^{n+1,m}$. Here, the coarse values are simply transferred to the four nearby fine grid points as shown in Fig. 2.2(b),

i.e. $v_k(x_i, y_j) = I_{k-1}^k v_{k-1}(x_i, y_j) = v_{k-1}(x_{i+\frac{1}{2}}, y_{j+\frac{1}{2}})$ for the i and j odd-numbered integers.

• Compute the corrected approximation on Ω_k

$$u_k^{m,\text{ after }CGC} = \bar{u}_k^{n+1,m} + \hat{v}_k^{n+1,m}.$$

Step 3) Postsmoothing: $u_k^{n+1,m+1} = SMOOTH^{\nu_2}(u_k^{m,\text{ after }CGC}, L_k, \phi_k^n)$.

This completes the description of a MGcycle.

An illustration of the corresponding two-grid cycle is given in Fig. 2.3. For the multi-grid V-cycle, it is given in Fig. 2.4.

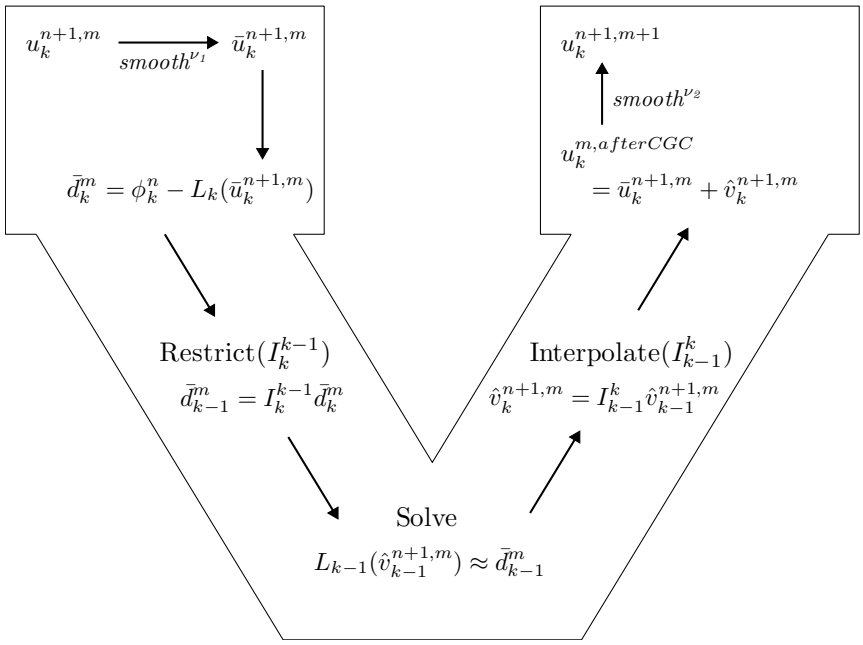


FIGURE 2.3. *MG* $(k, k-1)$ two-grid method.

2.4. Computational results

In this section, we perform a convergence test of the scheme and present several numerical experiments. Two-asset cash or nothing options can be useful

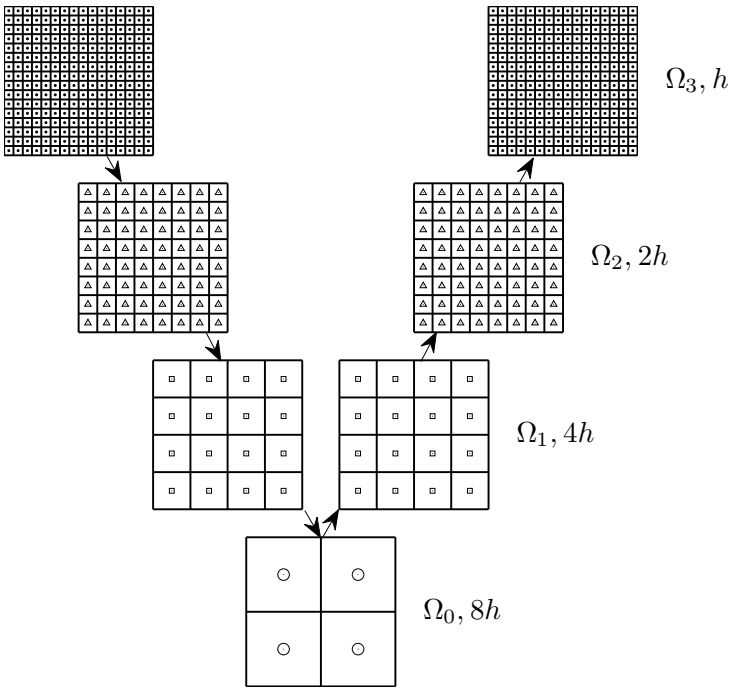


FIGURE 2.4. Schedule of grids for V-cycle.

building blocks for constructing more complex exotic option products. Let us consider a two-asset cash or nothing call option. This option pays out a fixed cash amount K if asset one, x, is above the strike X_1 and asset two, y, is above strike X_2 at expiration. The payoff is given by

$$u(x, y, 0) = \begin{cases} K & \text{if } x \geq X_1 \text{ and } y \geq X_2, \\ 0 & \text{otherwise .} \end{cases} \tag{2.6}$$

The formula for the exact value is known in [34] by

$$u(x, y, T) = Ke^{-rT} M(\alpha, \beta; \rho), \tag{2.7}$$

where

$$\alpha = \frac{\ln(x/K_1) + (r - \sigma_1^2/2)T}{\sigma_1\sqrt{T}}, \qquad \beta = \frac{\ln(y/K_2) + (r - \sigma_2^2/2)T}{\sigma_2\sqrt{T}}.$$

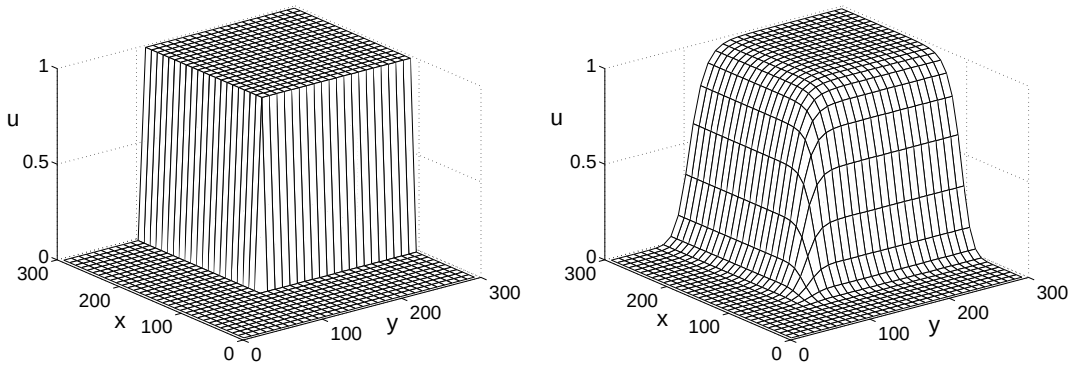
Here, $M(\alpha, \beta; \rho)$ denotes a standardized cumulative normal function where one random variable is less than α and a second random variable is less than β. The

correlation between the two variables is ρ:

$$M(\alpha, \beta; \rho) = \frac{1}{2\pi\sqrt{1-\rho^2}} \int_{-\infty}^{\alpha} \int_{-\infty}^{\beta} \exp\left[-\frac{x^2 - 2\rho xy + y^2}{2(1-\rho^2)}\right] dxdy.$$

The MATLAB code for the closed form solution of a two-asset cash or nothing call option is given in Appendix.

2.4.1. Convergence test. To obtain an estimate of the rate of convergence, we performed a number of simulations for a sample initial problem on a set of increasingly finer grids. We considered a domain, $\Omega = [0, 300] \times [0, 300]$. We computed the numerical solutions on uniform grids, $h = 300/2^n$ for $n = 5, 6, 7,$ and 8. For each case, we ran the calculation to time $T = 0.1$ with a uniform time step depending on a mesh size, $\Delta t = 0.032/2^n$. The initial condition is Eq. (2.6) with $K = 1$ and $X_1 = X_2 = 100$. The volatilities are $\sigma_1 = 0.5$ and $\sigma_2 = 0.5$. The correlation is $\rho = 0.5$, and the riskless interest rate is $r = 0.03$. Figures 2.5(a) and (b) show the initial configuration and final profile at T, respectively.



(a) (b)

FIGURE 2.5. (a) Initial condition and (b) numerical result at $T = 0.1$.

We let $\mathbf{e}$ be the error matrix with components $e_{ij} = u(x_i, y_j) - u_{ij}$. $u(x_i, y_j)$ is the analytic solution of Eq. (2.7) and u_{ij} is the numerical solution. We compute its discrete L^2 norm $\|\mathbf{e}\|_2$ is defined

$$\|\mathbf{e}\|_2 = \sqrt{\frac{1}{N_x N_y} \sum_{i=1}^{N_x} \sum_{j=1}^{N_y} e_{ij}^2}.$$

The errors and rates of convergence are given in Table 2.1. The results show that the scheme is first-order accurate.

Case	32×32	rate	64×64	rate	128×128	rate	256×256
$\|\mathbf{e}\|_2$	0.028161	0.95	0.014562	1.07	0.006928	0.96	0.003572

TABLE 2.1. L^2 norms of errors and convergence rates for u at time $T = 0.1$.

2.4.2. Multigrid performance. We investigated the convergence behavior of our multigrid(MG) method, especially mesh independence. The test problem was that of a two-asset cash or nothing call option with the convergence test parameter set. The average number of iterations per time step (see Fig. 2.6) and the CPU-time in seconds required for a solution to an identical convergence tolerance are displayed in Table 2.2. Although the number of multigrid iterations for convergence at each time step slowly increased as the mesh was refined, from a practical viewpoint, it was essentially grid independent.

Mesh	Average iterations per time step	CPU(s)
32×32	1.00	0.141
64×64	1.00	0.579
128×128	2.00	2.594
256×256	2.24	13.093

TABLE 2.2. Grid independence with an iteration convergence tolerance of 10^{-5}, $T = 0.1$, and $\Delta t = 0.001$.

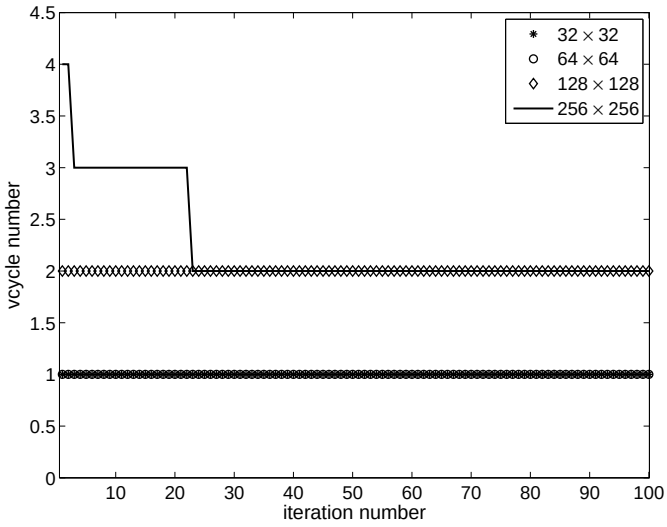


FIGURE 2.6. Number of V-cycles.

2.5. Conclusions

In this chapter, we focused on the performance of a multigrid method for option pricing problems. The numerical results showed that the total computational cost was proportional to the number of grid points. The convergence test showed that the scheme was first-order accurate since we used an implicit Euler method. In a forthcoming work, we will apply this method for multi-dimensional option problem.

Chapter 3

A comparison study of ADI and operator splitting methods on option pricing models

In this chapter, we perform a comparison study of alternating direction implicit (ADI) and operator splitting (OS) methods on multi-dimensional Black–Scholes option pricing models. The ADI method is used extensively in mathematical finance for numerically solving multi-asset option pricing problems. However, numerical results from the ADI scheme show oscillatory solution behaviors with nonsmooth payoffs or discontinuous derivatives at the exercise price with large time steps. Most option pricing problems have nonsmooth payoffs or discontinuous derivatives at the exercise price. In the ADI scheme, there are source terms which include y-derivatives when we solve x-derivative involving equations. Then, due to the nonsmooth payoffs, source term contains abrupt changes which are not in the range of implicit discrete operator and this leads to difficulty in solving the problem. On the other hand, the OS method does not contain the other variable's derivatives in the source term. We provide computational results showing the performance of the methods for two underlying asset option pricing problems. The results show that the OS method is very efficient and gives better accuracy and robustness than the ADI method with large time steps.

3.1. Introduction

In today's financial markets, options are the most common securities that are frequently bought and sold. Under the Black–Scholes partial differential equation (BS PDE) framework, various numerical methods [17, 25, 42, 54, 71] have been presented by using the finite difference method (FDM) to solve the option pricing problems [1, 16, 24, 66, 68, 69, 77]. But most option pricing problems have non-smooth payoffs or discontinuous derivatives at the exercise price. Standard finite difference schemes used to solve problems with nonsmooth payoff and large time steps do not work well because of discontinuities introduced in the source term. Moreover, these unwanted oscillations become problematic when estimating the hedging parameters, e.g., Delta and Gamma.

3.2. Numerical solutions for the ADI and OS methods

In this chapter, we focus on the two-dimensional Black–Scholes equation. Let $\mathcal{L}_{BS}$ be the operator

$$\mathcal{L}_{BS} = \frac{1}{2}\sigma_1^2 x^2 \frac{\partial^2 u}{\partial x^2} + \frac{1}{2}\sigma_2^2 y^2 \frac{\partial^2 u}{\partial y^2} + \rho\sigma_1\sigma_2 xy \frac{\partial^2 u}{\partial x \partial y} + rx\frac{\partial u}{\partial x} + ry\frac{\partial u}{\partial y} - ru.$$

Then the Black–Scholes equation can then be written as

$$\frac{\partial u}{\partial \tau} = \mathcal{L}_{BS} \quad \text{for } (x, y, \tau) \in \Omega \times (0, T], \tag{3.1}$$

where $\tau = T - t$. Originally, the option pricing problems are defined in the unbounded domain

$$\Omega \times (0, T] = \{(x, y, t) \mid x > 0, y > 0, \tau \in (0, T]\}.$$

However, we need to truncate this unbounded domain into a finite computational domain in order to solve Eq. (3.1) numerically by a finite difference method.

Therefore, we consider Eq. (3.1) on a finite domain:

$$(0, L) \times (0, M) \times (0, T] = \Omega \times (0, T],$$

where L and M are large enough so that the error in the price u is negligible. We can obtain approximate boundary conditions on the artificial boundaries $\{L\} \times [0, T]$ and $[0, T] \times \{M\}$ by assuming the asymptotic behavior of u. Let us first discretize the given computational domain $\Omega = (0, L) \times (0, M)$ as a uniform grid with a space step $h = L/N_x = M/N_y$ and a time step $\Delta\tau = T/N_t$. Here, the number of grid points is denoted by N_x, N_y, and N_t in the x, y, and t-direction, respectively. Furthermore, let us denote the numerical approximation of the solution by $u_{ij}^n \equiv u(x_i, y_j, t^n) = u\left((i - 0.5)h, (j - 0.5)h, n\Delta\tau\right)$, where $i = 1, ..., N_x$, $j = 1, ..., N_y$, and $n = 0, ..., N_t$. We use a linear boundary condition, similarly to [55, 56, 68, 89], as follows:

$$\frac{\partial^2 u}{\partial x^2}(0, y, t) = \frac{\partial^2 u}{\partial x^2}(L, y, t) = \frac{\partial^2 u}{\partial y^2}(x, 0, t) = \frac{\partial^2 u}{\partial y^2}(x, M, t) = 0,$$

for $0 \le x \le L$, $0 \le y \le M$, and $0 \le t \le T$.

3.2.1. Alternating Directions Implicit method. The main idea of the ADI method [19, 36] is to proceed in two stages, treating only one operator implicitly at each stage. First, a half-step is taken implicitly in x and explicitly in y. Then, the other half-step is taken implicitly in y and explicitly in x. The full scheme is:

$$\frac{u_{ij}^{n+\frac{1}{2}} - u_{ij}^{n}}{\Delta\tau} = \mathcal{L}_{ADI}^{x} u_{ij}^{n+\frac{1}{2}}, \tag{3.2}$$

$$\frac{u_{ij}^{n+1} - u_{ij}^{n+\frac{1}{2}}}{\Delta\tau} = \mathcal{L}_{ADI}^{y} u_{ij}^{n+1}, \tag{3.3}$$

where the discrete difference operators $\mathcal{L}_{ADI}^{x}$ and $\mathcal{L}_{ADI}^{y}$ are defined by

$$\begin{aligned}\mathcal{L}_{ADI}^{x} u_{ij}^{n+\frac{1}{2}} = {} & \frac{(\sigma_1 x_i)^2}{4}\frac{u_{i+1,j}^{n+\frac{1}{2}} - 2u_{ij}^{n+\frac{1}{2}} + u_{i-1,j}^{n+\frac{1}{2}}}{h^2} + \frac{(\sigma_2 y_j)^2}{4}\frac{u_{i,j+1}^{n} - 2u_{ij}^{n} + u_{i,j-1}^{n}}{h^2} \\ & + \frac{1}{2}\rho\sigma_1\sigma_2 x_i y_j \frac{u_{i+1,j+1}^{n} + u_{i-1,j-1}^{n} - u_{i-1,j+1}^{n} - u_{i+1,j-1}^{n}}{4h^2} \\ & + \frac{1}{2} r x_i \frac{u_{i+1,j}^{n+\frac{1}{2}} - u_{i,j}^{n+\frac{1}{2}}}{h} + \frac{1}{2} r y_j \frac{u_{ij+1}^{n} - u_{ij}^{n}}{h} - \frac{1}{2} r u_{ij}^{n+\frac{1}{2}},\end{aligned} \tag{3.4}$$

$$\begin{aligned}\mathcal{L}_{ADI}^{y} u_{ij}^{n+1} = {} & \frac{(\sigma_1 x_i)^2}{4}\frac{u_{i+1,j}^{n+\frac{1}{2}} - 2u_{ij}^{n+\frac{1}{2}} + u_{i-1,j}^{n+\frac{1}{2}}}{h^2} + \frac{(\sigma_2 y_j)^2}{4}\frac{u_{i,j+1}^{n+1} - 2u_{ij}^{n+1} + u_{i,j-1}^{n+1}}{h^2} \\ & + \frac{1}{2}\rho\sigma_1\sigma_2 x_i y_j \frac{u_{i+1,j+1}^{n+\frac{1}{2}} + u_{i-1,j-1}^{n+\frac{1}{2}} - u_{i-1,j+1}^{n+\frac{1}{2}} - u_{i+1,j-1}^{n+\frac{1}{2}}}{4h^2} \\ & + \frac{1}{2} r x_i \frac{u_{i+1,j}^{n+\frac{1}{2}} - u_{i,j}^{n+\frac{1}{2}}}{h} + \frac{1}{2} r y_j \frac{u_{ij+1}^{n+1} - u_{ij}^{n+1}}{h} - \frac{1}{2} r u_{ij}^{n+1}.\end{aligned} \tag{3.5}$$

Note that the addition of two Eqs. (3.2) and (3.3) results in Eq. (3.6).

$$\frac{u_{ij}^{n+1} - u_{ij}^{n}}{\Delta\tau} = \mathcal{L}_{ADI}^{x} u_{ij}^{n+\frac{1}{2}} + \mathcal{L}_{ADI}^{y} u_{ij}^{n+1}. \tag{3.6}$$

Algorithm ADI

- *Step 1*: The first stage of the ADI method, Eq. (3.4) can be rewritten as

$$\alpha_i u_{i-1,j}^{n+\frac{1}{2}} + \beta_i u_{ij}^{n+\frac{1}{2}} + \gamma_i u_{i+1,j}^{n+\frac{1}{2}} = f_{ij}, \tag{3.7}$$

where

$$\alpha_i = -\frac{(\sigma_1 x_i)^2}{4h^2}, \tag{3.8}$$

$$\beta_i = \frac{1}{\Delta\tau} + \frac{(\sigma_1 x_i)^2}{2h^2} + \frac{r x_i}{2h} + \frac{r}{2}, \tag{3.9}$$

$$\gamma_i = -\frac{(\sigma_1 x_i)^2}{4h^2} - \frac{r x_i}{2h}, \tag{3.10}$$

$$f_{ij} = \frac{u_{ij}^n}{\Delta\tau} + \frac{1}{4}(\sigma_2 y_j)^2 \frac{u_{i,j+1}^n - 2u_{ij}^n + u_{i,j-1}^n}{h^2} + \frac{1}{2} r y_j \frac{u_{i,j+1}^n - u_{i,j}^n}{h} + \frac{1}{2}\rho\sigma_1\sigma_2 x_i y_j \frac{u_{i+1,j+1}^n + u_{i-1,j-1}^n - u_{i-1,j+1}^n - u_{i+1,j-1}^n}{4h^2}. \tag{3.11}$$

For a fixed index j, the vector $u_{1:N_x,j}^{n+\frac{1}{2}}$ can be found by solving the tridiagonal system

$$A_x u_{1:N_x,j}^{n+\frac{1}{2}} = f_{1:N_x,j},$$

where A_x is a tridiagonal matrix constructed from Eq. (3.7) with a linear boundary condition, i.e.,

$$A_x = \begin{pmatrix} 2\alpha_1 + \beta_1 & \gamma_1 - \alpha_1 & 0 & \dots & 0 & 0 \\ \alpha_2 & \beta_2 & \gamma_2 & \dots & 0 & 0 \\ 0 & \alpha_3 & \beta_3 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \beta_{N_x-1} & \gamma_{N_x-1} \\ 0 & 0 & 0 & \dots & \alpha_{N_x} - \gamma_{N_x} & \beta_{N_x} + 2\gamma_{N_x} \end{pmatrix}.$$

Step 1 of the ADI method is then implemented in a loop over the y-direction:

for $j = 1 : N_y$

for $i = 1 : N_x$

Set α_i, β_i, γ_i, and f_{ij} by Eqs. (3.8)–(3.11)

end

Solve $A_x u_{1:N_x,j}^{n+\frac{1}{2}} = f_{1:N_x,j}$ by using the Thomas algorithm

end

- *Step 2*: The second stage of the ADI method, given by Eq. (3.5) is rewritten

$$\alpha_j u_{i,j-1}^{n+1} + \beta_j u_{ij}^{n+1} + \gamma_j u_{i,j+1}^{n+1} = g_{ij}, \tag{3.12}$$

where

$$\alpha_j = -\frac{(\sigma_2 y_j)^2}{4h^2}, \tag{3.13}$$

$$\beta_j = \frac{1}{\Delta\tau} + \frac{(\sigma_2 y_j)^2}{2h^2} + \frac{r y_j}{2h} + \frac{r}{2}, \tag{3.14}$$

$$\gamma_j = -\frac{(\sigma_2 y_j)^2}{4h^2} - \frac{r y_j}{2h}, \tag{3.15}$$

$$\begin{aligned} g_{ij} = \frac{u_{ij}^{n+\frac{1}{2}}}{\Delta\tau} + \frac{(\sigma_1 x_i)^2}{4}\frac{u_{i+1,j}^{n+\frac{1}{2}} - 2u_{ij}^{n+\frac{1}{2}} + u_{i-1,j}^{n+\frac{1}{2}}}{h^2} + \frac{1}{2} r x_i \frac{u_{i+1,j}^{n+\frac{1}{2}} - u_{i,j}^{n+\frac{1}{2}}}{h} \\ + \frac{1}{2}\rho\sigma_1\sigma_2 x_i y_j \frac{u_{i+1,j+1}^{n+\frac{1}{2}} + u_{i-1,j-1}^{n+\frac{1}{2}} - u_{i-1,j+1}^{n+\frac{1}{2}} - u_{i+1,j-1}^{n+\frac{1}{2}}}{4h^2}. \end{aligned} \tag{3.16}$$

For a fixed index i, the vector $u_{i,1:N_y}^{n+1}$ can be found by solving the tridiagonal system

$$A_y u_{i,1:N_y}^{n+1} = g_{i,1:N_y},$$

where the matrix A_y is a tridiagonal matrix,

$$A_y = \begin{pmatrix} 2\alpha_1 + \beta_1 & -\alpha_1 + \gamma_1 & 0 & \dots & 0 & 0 \\ \alpha_2 & \beta_2 & \gamma_2 & \dots & 0 & 0 \\ 0 & \alpha_3 & \beta_3 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \beta_{N_y-1} & \gamma_{N_y-1} \\ 0 & 0 & 0 & \dots & \alpha_{N_y} - \gamma_{N_y} & \beta_{N_y} + 2\gamma_{N_y} \end{pmatrix}.$$

Similarly to *Step 1*, *Step 2* is then implemented in a loop over the x-direction:

for $i = 1 : N_x$

for $j = 1 : N_y$

Set $\alpha_j,\ \beta_j,\ \gamma_j,$ and g_{ij} by Eqs. (3.13)–(3.16)

end

Solve $A_y u^{n+1}_{i,1:N_y} = g_{i,1:N_y}$ by using the Thomas algorithm

end

Execution of *Step 1* followed by *Step 2* advances the solution with a $\Delta\tau$-step in time.

3.2.2. Operator splitting method. The idea of the OS method is to divide each time step into fractional time steps with simpler operators [24, 38]). We shall introduce the basic OS scheme for the two-dimensional BS equation. The first leg is implicit in x while the second leg is implicit in y. The full scheme is:

$$\frac{u_{ij}^{n+\frac{1}{2}} - u_{ij}^{n}}{\Delta\tau} = \mathcal{L}_{OS}^{x} u_{ij}^{n+\frac{1}{2}}, \tag{3.17}$$

$$\frac{u_{ij}^{n+1} - u_{ij}^{n+\frac{1}{2}}}{\Delta\tau} = \mathcal{L}_{OS}^{y} u_{ij}^{n+1}, \tag{3.18}$$

where the discrete difference operators $\mathcal{L}_{OS}^{x}$ and $\mathcal{L}_{OS}^{y}$ are defined by

$$\begin{aligned}\mathcal{L}_{OS}^{x} u_{ij}^{n+\frac{1}{2}} &= \frac{(\sigma_1 x_i)^2}{2}\frac{u_{i-1,j}^{n+\frac{1}{2}} - 2u_{ij}^{n+\frac{1}{2}} + u_{i+1,j}^{n+\frac{1}{2}}}{h^2} + r x_i \frac{u_{i+1,j}^{n+\frac{1}{2}} - u_{ij}^{n+\frac{1}{2}}}{h} - \frac{r}{2}u_{ij}^{n+\frac{1}{2}} \\ &\quad + \frac{1}{2}\sigma_1\sigma_2\rho x_i y_j \frac{u_{i+1,j+1}^{n} + u_{i-1,j-1}^{n} - u_{i-1,j+1}^{n} - u_{i+1,j-1}^{n}}{4h^2}, \end{aligned} \tag{3.19}$$

$$\begin{aligned}\mathcal{L}_{OS}^{y} u_{ij}^{n+1} &= \frac{(\sigma_2 y_j)^2}{2}\frac{u_{i,j-1}^{n+1} - 2u_{ij}^{n+1} + u_{i,j+1}^{n+1}}{h^2} + r y_j \frac{u_{i,j+1}^{n+1} - u_{ij}^{n+1}}{h} - \frac{r}{2}u_{ij}^{n+1} \\ &\quad + \frac{1}{2}\sigma_1\sigma_2\rho x_i y_j \frac{u_{i+1,j+1}^{n+\frac{1}{2}} + u_{i-1,j-1}^{n+\frac{1}{2}} - u_{i-1,j+1}^{n+\frac{1}{2}} - u_{i+1,j-1}^{n+\frac{1}{2}}}{4h^2}. \end{aligned} \tag{3.20}$$

The OS scheme moves from the time level n to a intermediate time level $n+\frac{1}{2}$ and then to time level $n+1$. The addition of two Eqs. (3.17) and (3.18) results in Eq. (3.21).

$$\frac{u_{ij}^{n+1} - u_{ij}^{n}}{\Delta\tau} = \mathcal{L}_{OS}^{x} u_{ij}^{n+\frac{1}{2}} + \mathcal{L}_{OS}^{y} u_{ij}^{n+1}, \tag{3.21}$$

Algorithm OS

- *Step 1*: Equation (3.19) is rewritten as follows:

$$\alpha_i u_{i-1j}^{n+\frac{1}{2}} + \beta_i u_{ij}^{n+\frac{1}{2}} + \gamma_i u_{i+1j}^{n+\frac{1}{2}} = f_{ij},$$

where

$$\alpha_i = -\frac{\sigma_1^2 x_i^2}{2h^2},$$
$$\beta_i = \frac{1}{\Delta\tau} + \frac{\sigma_1^2 x_i^2}{h^2} + \frac{r x_i}{h} + \frac{r}{2},$$
$$\gamma_i = -\frac{\sigma_1^2 x_i^2}{2h^2} - \frac{r x_i}{h},$$
$$f_{ij} = \frac{1}{2}\rho\sigma_1\sigma_2 x_i y_j \frac{u^n_{i+1,j+1} - u^n_{i+1,j-1} - u^n_{i-1,j+1} + u^n_{i-1,j-1}}{4h^2} + \frac{u^n_{i,j}}{\Delta\tau}.$$

We note that in the OS method we do not have $\partial^2 u/\partial y^2$ and $\partial u/\partial y$ terms in the source f_{ij}. Then the solution procedure is same to the ADI method.

- *Step 2*: Equation (3.20) is rewritten as follows:

$$\alpha_j u^{n+1}_{ij-1} + \beta_j u^{n+1}_{ij} + \gamma_j u^{n+1}_{ij+1} = g_{ij},$$

where

$$\alpha_j = -\frac{\sigma_2^2 y_j^2}{2h^2},$$
$$\beta_j = \frac{1}{\Delta\tau} + \frac{\sigma_2^2 y_j^2}{h^2} + \frac{r y_j}{h} + \frac{r}{2},$$
$$\gamma_j = -\frac{\sigma_2^2 y_j^2}{2h^2} - \frac{r y_j}{h},$$
$$g_{ij} = \frac{1}{2}\rho\sigma_1\sigma_2 x_i y_j \frac{u^{n+\frac{1}{2}}_{i+1j+1} - u^{n+\frac{1}{2}}_{i+1j} - u^{n+\frac{1}{2}}_{ij+1} + u^{n+\frac{1}{2}}_{ij}}{4h^2} + \frac{u^{n+\frac{1}{2}}_{ij}}{\Delta\tau}.$$

We also note that in the OS method we do not have $\partial^2 u/\partial x^2$ and $\partial u/\partial x$ terms in the source g_{ij} and the solution procedure is same to the ADI method.

3.3. Numerical experiments

In this section, various examples are presented to compare the performance of the two different numerical schemes, the ADI and OS methods, for the BS equation. All computations were run in MATLAB version 7.6 [73]. The error

of the numerical solution was defined as $e_{ij} = u^e_{ij} - u_{ij}$ for $i = 1, \cdots, N_x$ and $j = 1, \cdots, N_y$, where u^e_{ij} is the exact solution and u_{ij} is the numerical solution. To compare the errors of the ADI and OS methods, we computed discrete l^2 norm $\|\mathbf{e}\|_2$ and maximum norm $\|\mathbf{e}\|_\infty$ of the error. We also used the root mean square error (RMSE) on a specific region. The RMSE is defined as

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i,j}^{N}\left(u^e_{ij} - u_{ij}\right)^2},$$

where N is the number of points on the gray region as shown in Fig. 3.1 and the region indicates the neighborhood of the exercise prices X_1 and X_2.

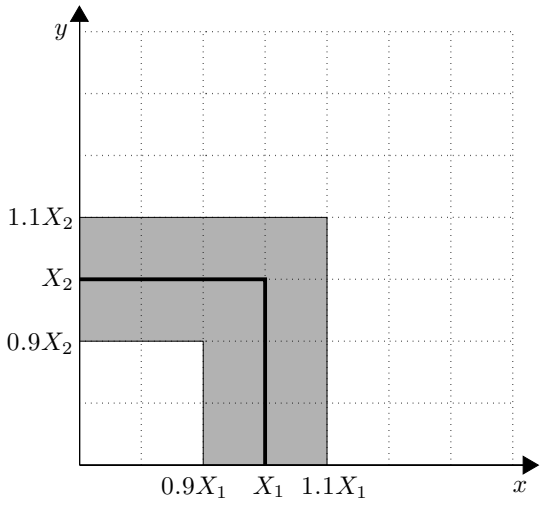


FIGURE 3.1. RMSE is calculated on the gray region.

3.3.1. Two-asset cash or nothing option. First, let us consider the two-asset cash or nothing call option [34]. Given two assets x and y, the payoff of the call option is given as

$$\Lambda(x, y) = \begin{cases} \text{Cash} & \text{if} \quad x \geq K_1 \text{ and } y \geq K_2, \\ 0 & \text{otherwise,} \end{cases}$$

where K_1 and K_2 are the strike prices of x and y, respectively (see Fig. 3.2).

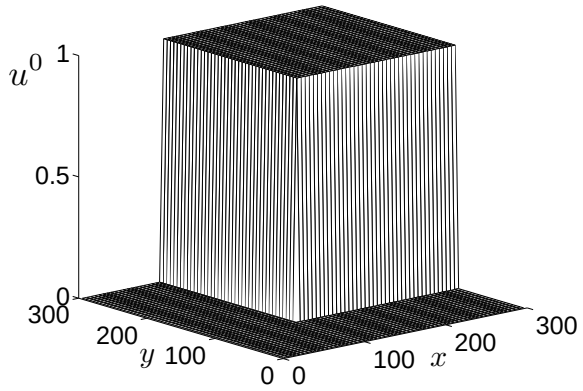


FIGURE 3.2. Payoff of a two-asset cash or nothing call option.

The exact solution is obtained from a closed form solution, which is provided in the Appendix.

The parameters used are: $\sigma_1 = \sigma_2 = 0.3$, $r = 0.03$, $\rho = 0.5$, $T = 0.5$, and $K_1 = K_2 = 100$. The computational domain is $\Omega = [0, 300] \times [0, 300]$.

As shown in Table 3.1, the ADI shows better convergence results than the OS method with relatively large space step sizes. However, with smaller space step size (equivalently with larger temporal step size) the ADI shows blowup solutions while the OS method produces convergent results.

Time $\Delta\tau$	Space h	ADI			OSM		
		$\|\mathbf{e}\|_2$	$\|\mathbf{e}\|_\infty$	RMSE	$\|\mathbf{e}\|_2$	$\|\mathbf{e}\|_\infty$	RMSE
0.05	5.0	0.000506	0.001952	0.000023	0.002411	0.010449	0.000147
0.025	2.5	0.000346	0.001266	0.000006	0.001043	0.004569	0.000028
0.0125	1.25	0.000207	0.000944	0.000003	0.000483	0.002136	0.000006
0.00625	0.625	N/A	N/A	N/A	0.000232	0.001030	0.000001

TABLE 3.1. Numerical results of a two-asset cash or nothing call option with different time step $\Delta\tau$ and space step h. Here, $\|\mathbf{e}\|_2$ and $\|\mathbf{e}\|_\infty$ are measured in a quarter of the domain, $[0, 150] \times [0, 150]$.

To investigate what made blowup solutions for the ADI scheme, we compare solutions, $u^{\frac{1}{2}}$ and u^1 and source terms, f and g generated from two methods, the ADI and OS. We used time step size, $\Delta\tau = 0.5$, and space step size, $h = 5$.

In Fig. 3.3, the first and the second columns show the numerical results at each step of the ADI and OSM for a two-asset cash or nothing option, respectively. As we can see from the figure, the numerical result of the ADI with a relatively large time step shows oscillatory solution along the lines $x = K_1$ and $y = K_2$. In Fig. 3.3(a), source terms in the first steps are shown. The source term in the ADI method exhibits oscillation around $y = K_2$ which is from the y-derivatives in the source term. On the other hand, for the OS method, we do not have the y-derivatives in the source term and solution $u^{\frac{1}{2}}$ is monotone around $y = K_2$. However, for the ADI we have oscillatory solution at the first step. After one complete time step, the result with the ADI shows a non-smooth numerical solution. However, the OS method results in a smooth numerical solution.

3.3.2. Call option on the maximum of two assets. Next, we consider a vanilla call option whose payoff is given as

$$\Lambda(x, y) = \max\{x - K_1, y - K_2, 0\}. \tag{3.22}$$

Figure 3.4 shows the payoff function (3.22).

In this case, we use the Dirichlet boundary condition at $x = L$ and $y = M$ and the linear boundary condition at $x = 0$ and $y = 0$. The parameters used are: $\sigma_1 = \sigma_2 = 0.3$, $r = 0.03$, $\rho = 0.5$, $T = 0.5$, and $K_1 = K_2 = 100$. The computational domain is $\Omega = [0, 300] \times [0, 300]$.

Table 3.2 shows the comparison of errors for the ADI and OS methods at time $T = 0.5$. The exact solutions are obtained from a closed for solution, which is

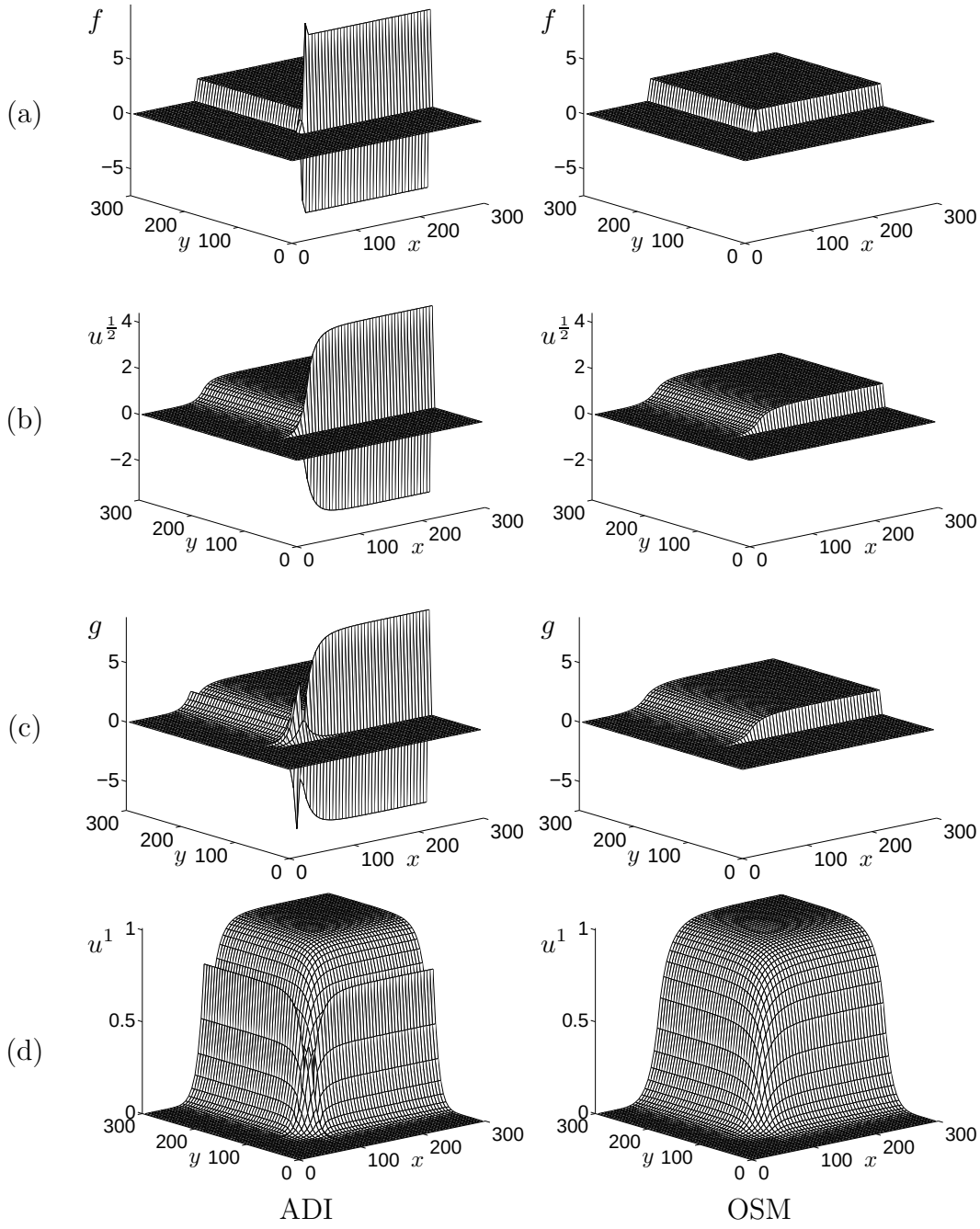


FIGURE 3.3. Numerical results of cash or nothing option using the ADI and OSM. (a) Source term f at *Step 1*, (b) solution $u^{\frac{1}{2}}$ at *Step 1*, (c) source term g at *Step 2*, and (d) solution u^1 at *Step 2*.

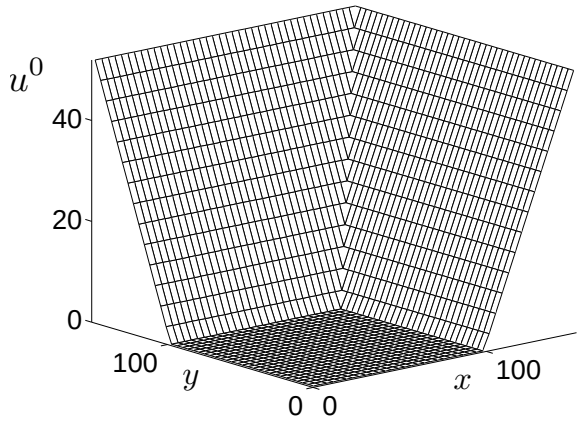


FIGURE 3.4. European call option payoff on the maximum of two assets.

Time $\Delta\tau$	Space h	ADI			OSM		
		$\|\mathbf{e}\|_2$	$\|\mathbf{e}\|_\infty$	RMSE	$\|\mathbf{e}\|_2$	$\|\mathbf{e}\|_\infty$	RMSE
0.05	5.0	0.057677	0.120848	0.006319	0.059967	0.175874	0.002286
0.025	2.5	0.027863	0.059866	0.001581	0.029001	0.085344	0.000610
0.0125	1.25	0.013713	0.029731	0.000395	0.014248	0.041703	0.000158
0.00625	0.625	N/A	N/A	N/A	0.007060	0.020569	0.000040

TABLE 3.2. Numerical results in case of European option on the maximum of two-asset with respect to the time step $\Delta\tau$ and space step h. Here, $\|\mathbf{e}\|_2$ and $\|\mathbf{e}\|_\infty$ are measured in a quarter of the domain, $[0, 150] \times [0, 150]$ and the RMSEs are evaluated in gray region which represented in Fig. 3.1.

provided in the Appendix. In the Table, the errors are similar in magnitude for the two methods until space step $h = 1.25$. However, after that, results from the ADI with smaller space steps show the blowup phenomenon of solution. On the other hand, the errors with the OS method do decrease with respect to time and space step refinements.

Figure 3.5 shows numerical results using the ADI and OS methods with $\Delta\tau = 0.5$ and $h = 3$. The first and second columns are results with the ADI and OS methods, respectively. In Fig. 3.5(a), source terms in the first steps are shown. The source term in the ADI method exhibits oscillation around $y = K_2$ which is

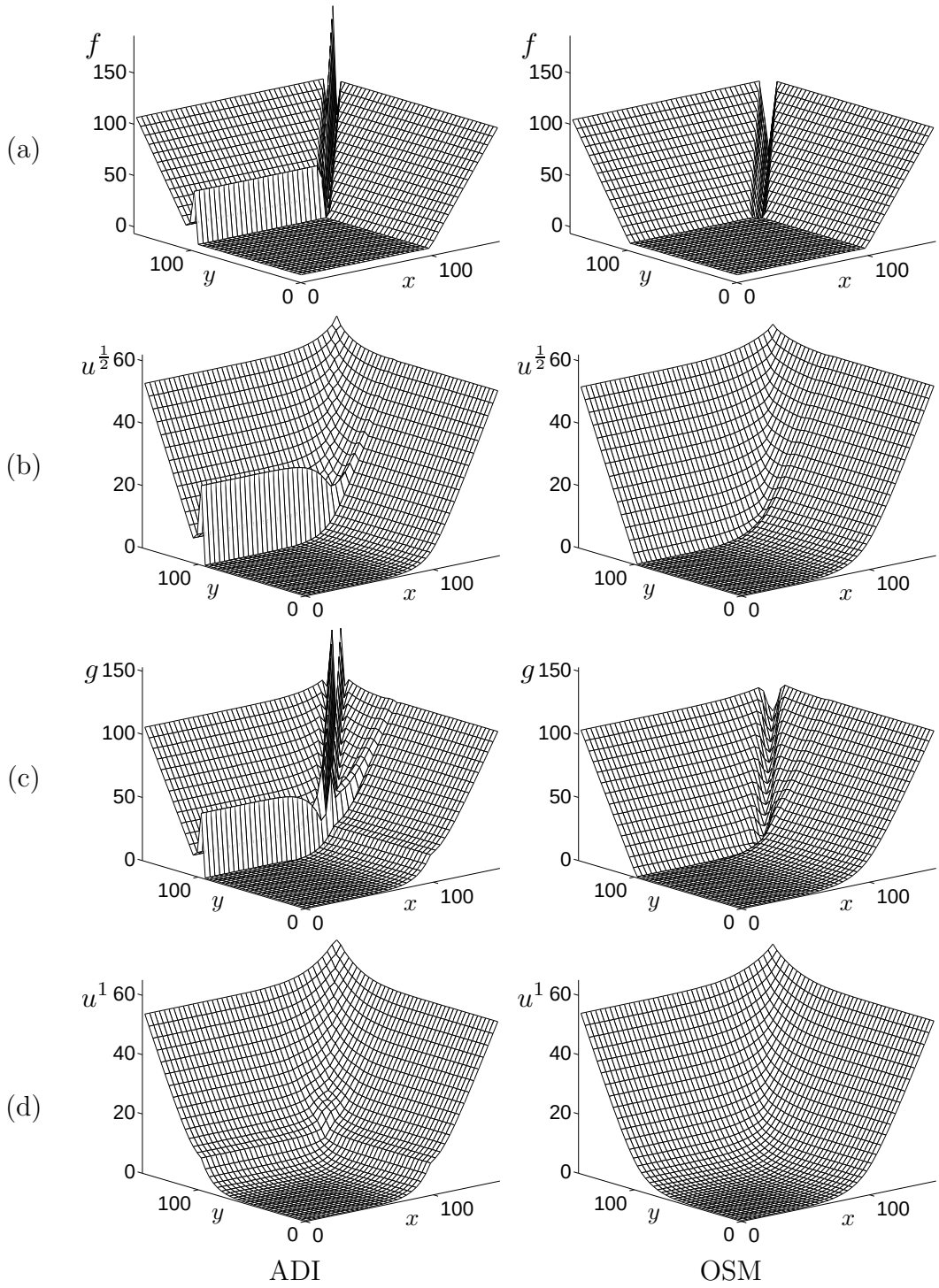


FIGURE 3.5. Numerical results using the ADI and OS methods with European call option on the maximum of two assets. (a) Source term f at *Step 1*, (b) solution $u^{\frac{1}{2}}$ at *Step 1*, (c) source term g at *Step 2*, and (d) solution u^1 at *Step 2*

from the y-derivatives in the source term. On the other hand, for the OS method, we do not have the y-derivatives in the source term and solution $u^{\frac{1}{2}}$ is smooth around $y = K_2$. After one complete time step, the result with the ADI shows a non-smooth numerical solution. However, the OS method results in a smooth numerical solution.

3.4. Conclusion

In this chapter, we performed a comparison study of alternating direction implicit and operator splitting methods on two-dimensional Black–Scholes option pricing models. However, numerical results from this scheme show oscillatory solution behaviors with nonsmooth payoffs with large time steps. Most option pricing problems have discontinuous derivatives at the exercise price. In the ADI scheme, there are source terms which include y-derivatives when we solve x-derivative involving equations. Then, due to the nonsmooth payoffs, source term contains abrupt changes which are not in the range of implicit discrete operator and this leads to difficulty in solving the problem. On the other hand, the OS method does not contain the other variable's derivatives in the source term. We provided computational results showing the performance of the methods for two underlying asset option pricing problems. The results showed that the OS method is very efficient and robust than the ADI method with large time steps.

And we provided the MATLAB code for evaluating option value with OS mehtod in Appendix.

Chapter 4

Comparison of Bi-CGSTAB, OS, and MG for 2D Black–Scholes equation

In this chapter, we perform a comparison of finite difference schemes for solving the two-dimensional Black–Scholes equation. We discretise the equation in space and time, and then solve a system of linear equations using the biconjugate gradient stabilized, operator splitting, and multigrid methods. The performance is presented, and results from different schemes are compared in two asset option problems based on the two dimensional Black–Scholes equation. Numerical results indicate that the operator splitting method results in a better performance among these solvers with the same level of accuracy.

4.1. Introduction

The finite difference methods (FDM) are very popular to approximate the solution of Black–Scholes equations (BS), see the general settings in option pricing [1, 24, 46, 51, 52, 66, 68, 77]. The FDM converts the differential equations into a system of difference equations. There have been introduced different approaches for efficient computations of the resulting linear systems, such as biconjugate gradient stabilized (Bi-CGSTAB) [40, 64, 74], operator splitting (OS) [38], and multigrid (MG) [28, 32, 59, 67, 81] methods. These solvers have been successively used in many applications such as physics, fluid dynamics, electromagnetics, and biomedical engineering. For different types of problems, different system solvers

gain advantages over the other methods, see [64]. To show the performance of the finite difference schemes for two dimensional problems, we compare the well known solvers, Bi-CGSTAB, OSM, and MG methods, for two dimensional Black–Scholes equations.

Bi-CGSTAB usually reduces the errors much faster than stationary methods and OSM can be calculated by a simple tridiagonal system of equations. Also, MG methods have been demonstrated its robustness and good convergence. There also have been other system solvers, such as ADI (Alternating Direction Method) [19], GMRES [63], however we omit the comparison in this work since GMRES and ADI methods are similar to Bi-CGSTAB and OS methods, respectively.

4.2. Numerical methods

In this chapter, we use the original Black–Scholes model (1.3) with two underlying assets by a change of variable $\tau = T - t$. And in this work, we use the following linear boundary conditions on all boundaries:

$$\frac{\partial^2 u}{\partial x^2}(0, y, \tau) = \frac{\partial^2 u}{\partial x^2}(L, y, \tau) = \frac{\partial^2 u}{\partial y^2}(x, 0, \tau) = \frac{\partial^2 u}{\partial y^2}(x, M, \tau) = 0,$$
$$\text{for } \ \forall \tau \in [0, T],\ 0 \le x \le L,\ 0 \le y \le M.$$

4.2.1. Finite difference methods. A finite difference method approximates derivatives by difference operators. FDM is a common numerical method that has been used in many application areas including finance, see [24, 66, 68, 69, 77] for a general framework of FDM for option pricing.

Let us first discretize the given computational domain $\Omega = (0, L) \times (0, M)$ as a uniform grid with a space step $h = L/N_x = M/N_y$ and a time step $\Delta t = T/N_t$. Here, N_x and N_y are the numerber of grid points, and N_t is the total number of

time steps. Let us denote the numerical approximation of the solution by

$$u_{ij}^n = u(x_i, y_j, t_n) = u\left((i-0.5)h, (j-0.5)h, n\Delta t\right),$$

where $i = 1, \ldots, N_x$, $j = 1, \ldots, N_y$, and $n = 0, 1, \ldots, N_t$. The discrete difference operator L_{BS} is defined by

$$\begin{aligned} L_{BS}u_{ij}^{n+1} = & \frac{(\sigma_1 x_i)^2}{2}\frac{u_{i-1,j}^{n+1} - 2u_{ij}^{n+1} + u_{i+1,j}^{n+1}}{h^2} + \frac{(\sigma_2 y_j)^2}{2}\frac{u_{i,j-1}^{n+1} - 2u_{ij}^{n+1} + u_{i,j+1}^{n+1}}{h^2} \\ & + \sigma_1\sigma_2\rho x_i y_j \frac{u_{i+1,j+1}^{n+1} + u_{ij}^{n+1} - u_{i,j+1}^{n+1} - u_{i+1,j}^{n+1}}{h^2} \\ & + r x_i \frac{u_{i+1,j}^{n+1} - u_{ij}^{n+1}}{h} + r y_j \frac{u_{i,j+1}^{n+1} - u_{ij}^{n+1}}{h} - r u_{ij}^{n+1}. \end{aligned} \tag{4.1}$$

Consequently, we need to solve the following system

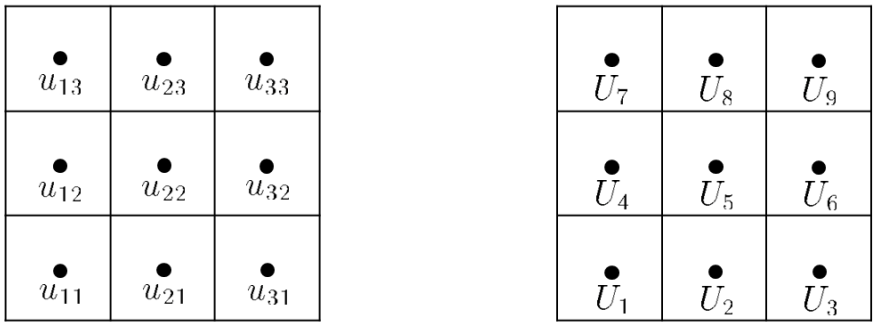
$$A\mathbf{u} = \mathbf{b}, \tag{4.2}$$

where $\mathbf{u}$ is the approximate solution of (1.3). The above system (4.2) can be solved using Bi-CGSTAB, OS, and multigrid methods. Let us first introduce each method in the following sections.

4.2.2. Bi-conjugate gradient stabilized (Bi-CGSTAB) method. The bi-conjugate gradient stabilized method (Bi-CGSTAB) was developed to solve nonsymmetric linear systems [74]. Bi-CGSTAB solves the system iteratively (4.2). First, we renumber the initial approximation u_{ij}, i.e.,

$$U_l = U_{N_x(j-1)+i} = u_{ij},$$

where $l = 1, \ldots, N_x \times N_y$, $i = 1, \ldots, N_x$, and $j = 1, \ldots, N_y$. For example, when $N_x = N_y = 3$, Fig. 4.1 shows the renumbering.



(a) (b)

FIGURE 4.1. Renumbering of U_{ij} on 3×3 grid.

Therefore, we have to solve $A\mathbf{U} = \mathbf{b}$, where $\mathbf{U}$ is the renumbered matrix, and

$$A = \begin{pmatrix} a_{11} & a_1+a_8 & 0 & a_2+a_9 & -a_3 & 0 & 0 & 0 & 0 \\ -a_1 & -2a_2+a_7 & a_8 & 0 & a_2+a_9 & -a_3 & 0 & 0 & 0 \\ 0 & -a_1-a_8 & a_{33} & 0 & a_3 & a_{36} & 0 & 0 & 0 \\ -a_2 & 0 & 0 & -2a_1+a_7 & a_1+a_8 & 0 & a_9 & -a_3 & 0 \\ 0 & -a_2 & 0 & -a_1 & a_7 & a_8 & 0 & a_9 & -a_3 \\ 0 & 0 & -a_2 & 0 & -a_1-a_8 & a_7+2a_8 & 0 & a_3 & -2a_3+a_9 \\ 0 & 0 & 0 & -a_2-a_9 & a_3 & 0 & a_{77} & a_{78} & 0 \\ 0 & 0 & 0 & 0 & -a_2-a_9 & a_3 & -a_1 & 2a_9+a_7 & -2a_3+a_8 \\ 0 & 0 & 0 & 0 & -a_3 & a_{96} & 0 & a_{98} & a_{99} \end{pmatrix}$$

Here,

$$\begin{aligned} a_1 &= \frac{(\sigma_1 x)^2}{2h^2},\ a_2 = \frac{(\sigma_2 y)^2}{2h^2},\ a_3 = \frac{\sigma_1\sigma_2\rho xy}{h^2},\ a_4 = \frac{rx}{h},\ a_5 = \frac{ry}{h},\ a_6 = -r, \\ a_7 &= \frac{1}{\Delta t} + 2(a_1+a_2) - a_3 + a_4 + a_5 - a_6,\ a_8 = -a_1 + a_3 - a_4, \\ a_9 &= -a_2 + a_3 - a_5,\ a_{11} = a_7 - 2(a_1+a_2),\ a_{33} = a_7 - 2(a_2+a_8), \\ a_{36} &= a_9 + a_2 - 2a_3,\ a_{77} = a_7 - 2(a_1-a_9),\ a_{78} = a_8 + a_1 - 2a_3, \\ a_{96} &= -a_2 + 2a_3 - a_9,\ a_{98} = -a_1 + 2a_3 - a_8,\ a_{99} = a_7 - 2(2a_3 - a_8 - a_9). \end{aligned}$$

The Bi-CGSTAB algorithm is as follows.

Bi-CGSTAB cycle

Define the maximum number of iteration $ITER$ and the error tolerance TOL

Set $\mathbf{r}^0 = \mathbf{b} - A\mathbf{U}^0, \hat{\mathbf{r}}^0 = \mathbf{r}^0, \rho^0 = \alpha = \omega^0 = 1, \mathbf{v}^0 = \mathbf{p}^0 = 0$

Set $k = 1$

While ($k \leq ITER$ & $\|\mathbf{r}^k\|_2 > TOL$)

$\rho^k = \sum_{i=1}^{N} \hat{r}_i^0 r_i^{k-1}, \beta = (\rho^k/\rho^{k-1})(\alpha/\omega^{k-1})$

$\mathbf{p}^k = \mathbf{r}^{k-1} + \beta(\mathbf{p}^{k-1} - \omega^{k-1}\mathbf{v}^{k-1})$

$\mathbf{v}^k = A\mathbf{p}^k$

$\alpha = \rho^k / \sum_{i=1}^{N} \hat{r}_j^0 v_i^k$

$\mathbf{s} = \mathbf{r}^{k-1} - \alpha\mathbf{v}^k$

$\mathbf{t} = A\mathbf{s}$

$\omega^k = \sum_{i=1}^{N} t_i s_i / \sum_{i=1}^{N} t_i^2$

$\mathbf{U}^k = \mathbf{U}^{k-1} + \alpha\mathbf{p}^k + \omega^k\mathbf{s}$

$\mathbf{r}^k = \mathbf{s} - \omega^k\mathbf{t}$

$k = k + 1$

End While

4.2.3. Operator splitting method. The operator splitting method (OS) computes the solutions in two time steps at $t_{n+\frac{1}{2}}, t_{n+1}$ with time step size Δt as follows:

$$\frac{u_{ij}^{n+1} - u_{ij}^n}{\Delta t} = \mathcal{L}_{BS}^x u_{ij}^{n+\frac{1}{2}} + \mathcal{L}_{BS}^y u_{ij}^{n+1}, \tag{4.3}$$

where the discrete difference operators $\mathcal{L}^x_{BS}$ and $\mathcal{L}^y_{BS}$ are defined by

$$\begin{aligned}\mathcal{L}^x_{BS}u_{ij}^{n+\frac{1}{2}} &= \sigma_1^2 x_i^2 \frac{u_{i-1,j}^{n+\frac{1}{2}} - 2u_{ij}^{n+\frac{1}{2}} + u_{i+1,j}^{n+\frac{1}{2}}}{2h^2} + rx_i \frac{u_{i+1,j}^{n+\frac{1}{2}} - u_{ij}^{n+\frac{1}{2}}}{h} - \lambda_2 r u_{ij}^{n+\frac{1}{2}} \\ &\quad + \lambda_1\sigma_1\sigma_2\rho x_i y_j \frac{u_{i+1,j+1}^n + u_{ij}^n - u_{i,j+1}^n - u_{i+1,j}^n}{h^2}, \end{aligned} \tag{4.4}$$

$$\begin{aligned}\mathcal{L}^y_{BS}u_{ij}^{n+1} &= \sigma_2^2 y_j^2 \frac{u_{i,j-1}^{n+1} - 2u_{ij}^{n+1} + u_{i,j+1}^{n+1}}{2h^2} + ry_j \frac{u_{i,j+1}^{n+1} - u_{ij}^{n+1}}{h} - (1-\lambda_2) r u_{ij}^{n+1} \\ &\quad + (1-\lambda_1)\sigma_1\sigma_2\rho x_i y_j \frac{u_{i+1,j+1}^{n+\frac{1}{2}} + u_{ij}^{n+\frac{1}{2}} - u_{i,j+1}^{n+\frac{1}{2}} - u_{i+1,j}^{n+\frac{1}{2}}}{h^2}, \end{aligned} \tag{4.5}$$

where $\lambda_1,\ \lambda_2 \in [0,1]$. The first leg is implicit in x-direction while the second leg is implicit in y-direction. The OS scheme moves from the time level n to a intermediate time level $n+1/2$ and then to time level $n+1$. The idea behind operator splitting is to split Eq. (1.3) into two one dimensional problems. We then solve each sub-problem by a fast direct numerical solver such as the Thomas algorithm. Thus, we consider two one-dimensional discrete equations.

$$\frac{u_{ij}^{n+\frac{1}{2}} - u_{ij}^n}{\Delta t} = \mathcal{L}^x_{BS}u_{ij}^{n+\frac{1}{2}},\ \frac{u_{ij}^{n+1} - u_{ij}^{n+\frac{1}{2}}}{\Delta t} = \mathcal{L}^y_{BS}u_{ij}^{n+1}.$$

The OSM algorithm is as follows.

Algorithm OSM

Construct a tridiagonal matrix of the form

$$A_x = \begin{pmatrix} \beta_1 & \gamma_1 & 0 & \dots & 0 & 0 \\ \alpha_2 & \beta_2 & \gamma_2 & \dots & 0 & 0 \\ 0 & \alpha_3 & \beta_3 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \beta_{N_x-1} & \gamma_{N_x-1} \\ 0 & 0 & 0 & \dots & \alpha_{N_x} & \beta_{N_x} \end{pmatrix}.$$

Here, the elements of the matrix are

$$\begin{aligned}
\beta_1 &= \frac{1}{\Delta t} + \frac{rx_1}{h} + \lambda_2 r,\ \gamma_1 = -\frac{rx_1}{h}, \\
\alpha_i &= -\frac{\sigma_1^2 x_i^2}{2h^2},\ \beta_i = \frac{1}{\Delta t} + \frac{\sigma_1^2 x_i^2}{h^2} + \frac{rx_i}{h} + \lambda_2 r,\ \gamma_i = -\frac{\sigma_1^2 x_i^2}{2h^2} - \frac{rx_i}{h}, \\
&\qquad\qquad \text{for } i = 2, \cdots, N_x - 1, \\
\alpha_{N_x} &= \frac{rx_{N_x}}{h},\ \beta_{N_x} = \frac{1}{\Delta t} - \frac{rx_{N_x}}{h} + \lambda_2 r.
\end{aligned}$$

Similarly, construct a tridiagonal matrix A_y,

$$A_y = \begin{pmatrix}
\beta_1 & \gamma_1 & 0 & \dots & 0 & 0 \\
\alpha_2 & \beta_2 & \gamma_2 & \dots & 0 & 0 \\
0 & \alpha_3 & \beta_3 & \dots & 0 & 0 \\
\vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\
0 & 0 & 0 & \dots & \beta_{N_y-1} & \gamma_{N_y-1} \\
0 & 0 & 0 & \dots & \alpha_{N_y} & \beta_{N_y}
\end{pmatrix}.$$

Here, the elements of the matrix are

$$\begin{aligned}
\beta_1 &= \frac{1}{\Delta t} + \frac{ry_1}{h} + (1-\lambda_2) r,\ \gamma_1 = -\frac{ry_2}{h}, \\
\alpha_j &= -\frac{\sigma_2^2 y_j^2}{2h^2},\ \beta_j = \frac{1}{\Delta t} + \frac{\sigma_2^2 y_j^2}{h^2} + \frac{ry_j}{h} + (1-\lambda_2) r,\ \gamma_j = -\frac{\sigma_2^2 y_j^2}{2h^2} - \frac{ry_j}{h} \\
&\qquad\qquad \text{for } j = 2, \cdots, N_y - 1, \\
\alpha_{N_y} &= \frac{ry_{N_y}}{h},\ \beta_{N_y} = \frac{1}{\Delta t} - \frac{ry_{N_y}}{h} + (1-\lambda_2) r.
\end{aligned}$$

We note that matrices A_x and A_y do not depend on solution u^n.

Step 1: Loop over the y-direction:

For $j = 1, ..., N_y$

For $i = 1, ..., N_x$

$$\mathbf{b}_y(i) = \lambda_1 \rho \sigma_1 \sigma_2 x_i y_j \frac{u^n_{i+1,j+1} - u^n_{i+1,j} - u^n_{i,j+1} + u^n_{ij}}{h^2} + \frac{u^n_{ij}}{\Delta t}.$$

end

Solve $A_x u^{n+\frac{1}{2}}(:, j) = \mathbf{b}_y$.

Apply boundary conditions.

end

Step 2: Loop over the x-direction:

For $i = 1, ..., N_x$

For $j = 1, ..., N_y$

$$\mathbf{b}_x(j) = (1 - \lambda_1) \rho \sigma_1 \sigma_2 x_i y_j \frac{u^{n+\frac{1}{2}}_{i+1,j+1} - u^{n+\frac{1}{2}}_{i+1,j} - u^{n+\frac{1}{2}}_{i,j+1} + u^{n+\frac{1}{2}}_{ij}}{h^2} + \frac{u^{n+\frac{1}{2}}_{ij}}{\Delta t}.$$

end

Solve $A_y u^{n+1}(i, :) = \mathbf{b}_x$.

Apply boundary conditions.

end

4.2.4. Multigrid method. Multigrid methods belong to the class of fastest iterations, because their convergence rate is independent of the step size h, see [32]. We define a discrete domain by

$$\Omega_k = \{(h(i - 0.5), h(j - 0.5)) \,|1 \leq i, j \leq 2^{k+1}\}.$$

Ω_{k-1} is coarser than Ω_k by factor 2. The multigrid solution of the discrete BS equation (4.6)

$$\frac{u_{ij}^{n+1} - u_{ij}^{n}}{\Delta t} = L_{BS} u_{ij}^{n+1} \tag{4.6}$$

makes use of a hierarchy of meshes created by successively coarsening the original mesh, see Fig. 4.2.

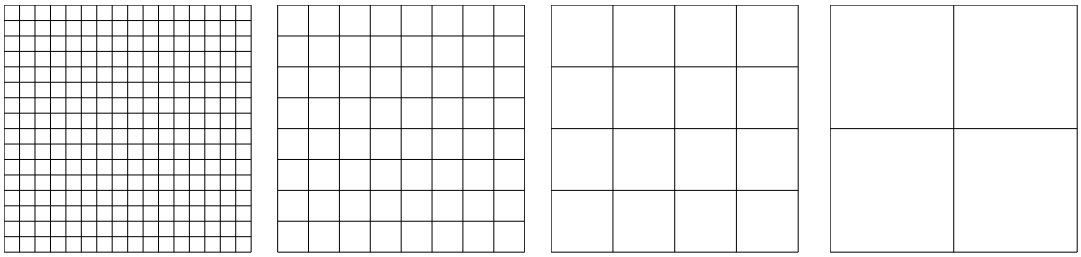


FIGURE 4.2. A sequence of coarse grids starting with h.

We use a multigrid cycle to solve the discrete system at the implicit time level. A pointwise Gauss–Seidel relaxation scheme is used as the smoother in the multigrid method. We first rewrite the above equation (4.6) by

$$L(u_{ij}^{n+1}) = u_{ij}^{n} \quad \text{for each } (i,j) \in \Omega_k, \tag{4.7}$$

where

$$L(u_{ij}^{n+1}) = u_{ij}^{n+1} - \Delta t L_{BS} u_{ij}^{n+1}.$$

Given the number ν_1 and ν_2 of pre- and post- smoothing relaxation sweeps, an iteration step for the multigrid method using the V-cycle is formally written as follows [72]. We use a notation u_k^n as a numerical solution on the discrete domain Ω_k at time $t = n\Delta t$. Given u_k^n, we want to find u_k^{n+1} solution which satisfies equation (4.6). At the very beginning of the multigrid cycle the solution from the previous time step is used to provide an initial guess for the multigrid procedure.

First, let $u_k^{n+1,0} = u_k^n$. The algorithm of the multigrid method for solving the discrete BS equation (4.6) is following:

Multigrid cycle

$$u_k^{n+1,m+1} = MGcycle(k, u_k^{n+1,m}, L_k, u_k^n, \nu_1, \nu_2).$$

Step 1) Presmoothing: perform ν_1 Gauss–Seidel relaxation steps.

$$\bar{u}_k^{n+1,m} = SMOOTH^{\nu_1}(u_k^{n+1,m}, L_k, u_k^n), \tag{4.8}$$

Step 2) Coarse grid correction

- Compute the residual on Ω_k: $\bar{d}_k^m = u_k^n - L_k(\bar{u}_k^{n+1,m})$.
- Restriction to Ω_{k-1}: $\bar{d}_{k-1}^m = I_k^{k-1}\bar{d}_k^m$, $\bar{u}_{k-1}^{n+1,m} = I_k^{k-1}\bar{u}_k^{n+1,m}$.
- Compute an approximation soultion on Ω_{k-1}:

$$L_{k-1}(u_{k-1}^{n+1,m}) = \bar{d}_{k-1}^m. \tag{4.9}$$

- Solve the equation (4.9):

$$\hat{u}_{k-1}^{n+1,m} = \begin{cases} \text{MGcycle}(k-1, \bar{u}_{k-1}^{n+1,m}, L_{k-1}, \bar{d}_{k-1}^n, \nu_1, \nu_2) & \text{for } k > 1 \\ \text{apply the smoothing procedure in (4.8)} & \text{for } k = 1. \end{cases}$$

- Interpolate the correction: $\hat{u}_k^m = I_{k-1}^k \hat{u}_{k-1}^m$.
- Compute the corrected approximation on Ω_k:

$$u_k^{m,\text{ after }CGC} = \bar{u}_k^{n+1,m} + \hat{u}_k^m.$$

Step 3) Postsmoothing: $u_k^{n+1,m+1} = SMOOTH^{\nu_2}(u_k^{m,\text{ after }CGC}, L_k, u_k^n)$.

4.3. Computational results

In this section, we compare the performance of the numerical methods (Bi-CGSTAB, OS, and MG) using CPU times. Each method is implemented using MATLAB [73] in a desktop computer.

We consider three types of two-asset cash-or-nothing options. The cash-or-nothing options are useful building blocks for constructing more complex exotic option products and they are widely traded in the real world financial market.

Case 1: A two asset cash-or-nothing call pays out a fixed cash amount K if asset one, x, is above the strike X_1 and asset two, y, is above strike X_2 at expiration. The payoff is given by

$$\Lambda(x,y) = \begin{cases} K & \text{if } x \geq X_1 \text{ and } y \geq X_2, \\ 0 & \text{otherwise .} \end{cases} \tag{4.10}$$

Case 2: A two asset cash-or-nothing put pays out a fixed cash amount K if asset one, x, is below the strike X_1 and asset two, y, is below strike X_2 at expiration. The payoff is given by

$$\Lambda(x,y) = \begin{cases} K & \text{if } x \leq X_1 \text{ and } y \leq X_2, \\ 0 & \text{otherwise .} \end{cases} \tag{4.11}$$

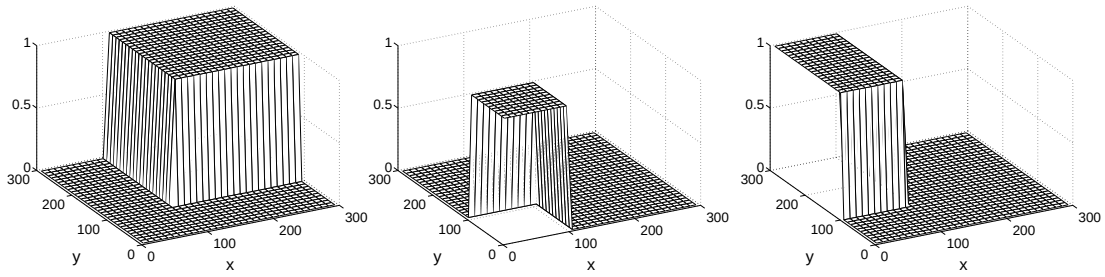
Case 3: A two asset cash-or-nothing up-down pays out a fixed cash amount K if asset one, x, is above the strike X_1 and asset two, y, is below strike X_2 at expiration. The payoff is given by

$$\Lambda(x,y) = \begin{cases} K & \text{if } x \geq X_1 \text{ and } y \leq X_2, \\ 0 & \text{otherwise .} \end{cases} \tag{4.12}$$

Fig. 4.3(a), (b), and (c) show the payoff function $\Lambda(x,y)$ for **Case 1**, **Case 2**, and **Case 3**, respectively.

The formulas published by Heynen and Kat [35] can be used to price these binary options:

$$\begin{aligned} \textbf{Case 1}: \; u(x,y,T) &= Ke^{-rT}M(\alpha,\beta;\rho), \\ \textbf{Case 2}: \; u(x,y,T) &= Ke^{-rT}M(-\alpha,-\beta;\rho), \\ \textbf{Case 3}: \; u(x,y,T) &= Ke^{-rT}M(-\alpha,\beta;-\rho), \end{aligned}$$



(a) (b) (c)

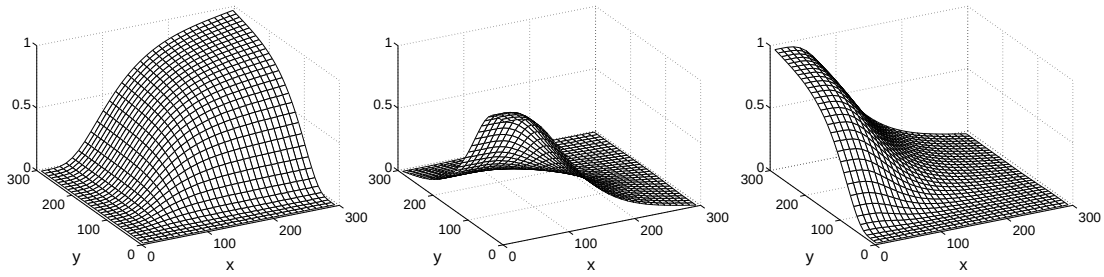
FIGURE 4.3. (a), (b), and (c) are payoff functions of **Case 1**, **Case 2**, and **Case 3**, respectively.

where $\alpha = [\ln(x/X_1)+(r-\sigma_1^2/2)T]/(\sigma_1\sqrt{T})$, $\beta = [\ln(y/X_2)+(r-\sigma_2^2/2)T]/(\sigma_2\sqrt{T})$ [34]. Here $M(\alpha, \beta; \rho)$ denotes a standardized cumulative normal function where one random variable is less than α and a second random variable is less than β. The correlation between the two variables is ρ:

$$M(\alpha, \beta; \rho) = \frac{1}{2\pi\sqrt{1-\rho^2}} \int_{-\infty}^{\alpha} \int_{-\infty}^{\beta} \exp\left[-\frac{x^2 - 2\rho xy + y^2}{2(1-\rho^2)}\right] dxdy.$$

We considered a domain, $\Omega = [0, 300] \times [0, 300]$. We computed the numerical solution on uniform grids, $h = 300/2^n$ for $n = 5, 6, 7$, and 8. For each case, we ran the calculation to time $T = 1$ with a uniform time step $\Delta t = 0.01$ with a given strike price of $X_1 = 100,\ X_2 = 100$, and cash amount $K = 1$. The volatilities are $\sigma_1 = 0.5,\ \sigma_2 = 0.5$ with a correlation $\rho = 0.5$, and the riskless interest rate $r = 0.03$. Fig. 4.4 shows the numerical solution at $T = 1$ case by case. We let $\mathbf{e}$ be the matrix with components $e_{ij} = u(x_i, y_j) - U_{ij}$ and compute its discrete l_2-norm of the error, $\|\mathbf{e}\|_2$.

We test the numerical experiments of different case with three solvers, Bi-CGSTAB, OSM and MG. To make a fair comparison of these solvers, we match the accuracy of these solvers by changing iteration parameters. Table 4.3 shows



(a) (b) (c)

FIGURE 4.4. (a), (b), and (c) are numerical solutions at time $T = 1$ of **Case 1**, **Case 2**, and **Case 3**, respectively.

the result of **Case 1**. Figure 4.5 shows the CPU time with **Case 1** and we can see l_2 errors are more or less same order to each other on each mesh size.

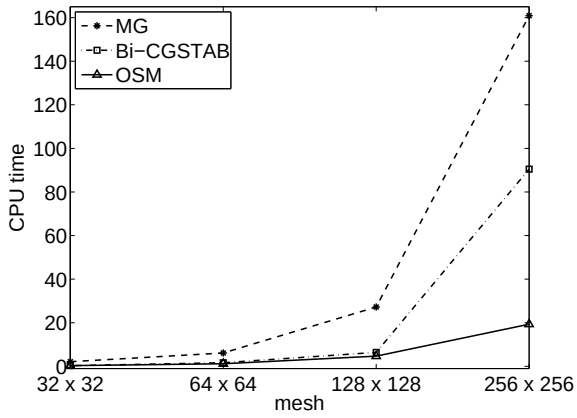


FIGURE 4.5. CPU times of **Case 1**.

In this Figs., the solid line with triangles, the dash-dot line with squares, and the dashed line with stars express OSM, BI-CGSTAB, and MG, respectively.

Next, let us check the CPU times to compare efficiency of these solvers. Table 4.3 also shows the CPU times with each method. We can confirm that OS method has a linear CPU time cost as the spatial domain is doubled in each direction. Table 4.1, Table 4.2, and Fig. 4.6 show the CPU times with **Case 2** and **Case**

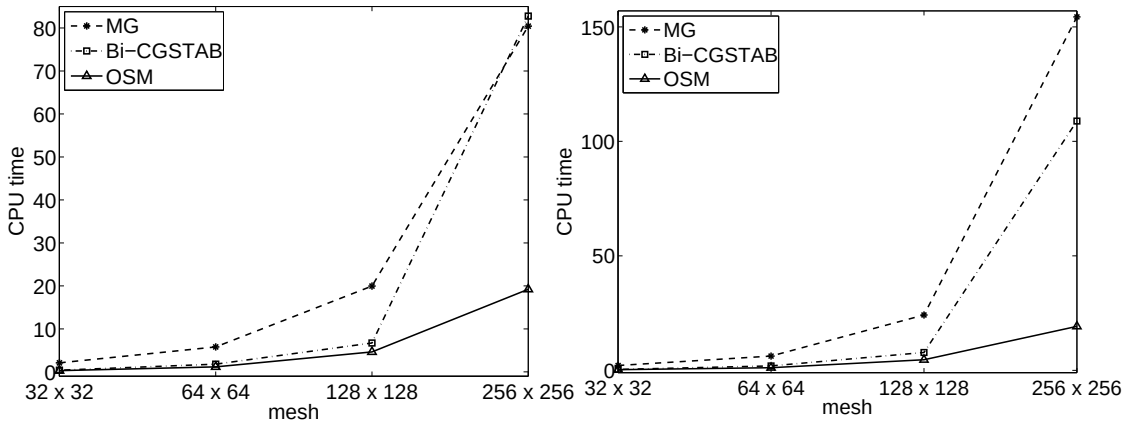


FIGURE 4.6. (a) and (b) are CPU times of **Case 2** and **Case 3**, respectively.

3. From all these results, we can confirm that OS method is faster than other methods under the same accuracy.

Mesh	Bi-CGStab	OSM	Multigrid
32×32	0.0161 (0.33)	0.0160 (0.31)	0.0150 (2.11)
64×64	0.0126 (1.67)	0.0126 (1.17)	0.0131 (6.13)
128×128	0.0078 (6.42)	0.0076 (4.67)	0.0066 (27.22)
256×256	0.0089 (90.50)	0.0087 (19.34)	0.0086 (160.90)

TABLE 4.1. **Case 1**: Comparison of l_2 error and (CPU time).

Mesh	Bi-CGStab	OSM	Multigrid
32×32	0.0130 (0.39)	0.0131 (0.30)	0.0137 (2.11)
64×64	0.0100 (1.80)	0.0099 (1.17)	0.0097 (5.81)
128×128	0.0063 (6.73)	0.0063 (4.64)	0.0060 (19.97)
256×256	0.0066 (82.78)	0.0064 (19.22)	0.0059 (80.44)

TABLE 4.2. **Case 2**: Comparison of l_2 error and (CPU time).

Mesh	Bi-CGStab	OSM	Multigrid
32×32	0.0127 (0.41)	0.0128 (0.30)	0.0124 (2.11)
64×64	0.0065 (1.94)	0.0064 (1.16)	0.0069 (6.25)
128×128	0.0070 (7.80)	0.0069 (4.63)	0.0064 (24.19)
256×256	0.0059 (108.88)	0.0058 (19.22)	0.0056 (154.38)

TABLE 4.3. **Case 3**: Comparison of l_2 error and (CPU time).

4.4. Conclusion

The main purpose of this chpater is to present the results of comparison of finite difference schemes and indicate a better performing scheme. The finite difference methods are applied to the Black–Scholes formula for stock option pricing. The large linear system, derived from the Black–Scholes equation, is solved by biconjugate gradient stabilized, operator splitting, and multigrid methods. The performance of these methods is compared for two asset option problems based on two dimensional Black–Scholes equations. For the standard finite difference schemes, we can see that the OSM leads to better performance. Moreover, the efficiency of the OSM can be increased with mesh size. Even if Bi-CGSTAB and multigrid solvers have a good accuracy but the ones still need a lot of computational times. On the other hand, operator splitting is faster than other two methods under the same accuracy.

Chapter 5

An adaptive grid generation technique depending on a far-field boundary position for the Black–Scholes equation

In this chapter, we present an accurate and efficient numerical method for solving the Black–Scholes equation. The method uses an adaptive grid technique which is based on a far-field boundary position of the equation and also the Peclet condition. The algorithm for the automatic adaptive grid generation is: First, for a given error tolerance, we determine a priori a suitable far-field boundary location using the mathematical model parameters. Second, put a uniform fine grid around the non-smooth points of the payoff such as a strike price and a non-uniform grid in the remaining regions. Numerical tests are presented to demonstrate the accuracy and efficiency of the proposed method. The results show that the computational times using the new adaptive grid method are reduced substantially when compared to those of a uniform grid method with a similar magnitude of error.

5.1. Introduction

In this chapter, we develop an accurate and efficient numerical method to solve the Black–Scholes (BS) equation with one underlying asset:

$$\frac{\partial u}{\partial t} = -\frac{1}{2}(\sigma x)^2 \frac{\partial^2 u}{\partial x^2} - rx\frac{\partial u}{\partial x} + ru, \ \forall (x,t) \in (0,\infty) \times [0,T), \tag{5.1}$$

where $u(x,t)$ represents the value of the derivative security, x is the value of the underlying security, t is the time, r is the risk-free interest rate, and σ is the

volatility of the underlying asset [6, 53]. By changing variable t to $\tau = T - t$, we can rewrite Eq. (5.1) as a parabolic partial differential equation:

$$\frac{\partial u}{\partial \tau} = \frac{1}{2}(\sigma x)^2 \frac{\partial^2 u}{\partial x^2} + rx\frac{\partial u}{\partial x} - ru, \ \forall (x, \tau) \in (0, \infty) \times (0, T]. \tag{5.2}$$

The initial condition is given by

$$u(x, 0) = p(x). \tag{5.3}$$

Taking a European call option as an example, we have the payoff function as $p(x) = \max(x - K, 0)$ with a given strike price K; see [34]. The original problem (5.1) is posed on an infinite domain $(0, \infty)$ for the space variable x. However, when we approximate the solution numerically, we need to use a finite size domain, i.e., $[0, S_{\max}]$. In [41], detailed estimates can be obtained for the error in the solution due to the truncated finite domain size.

The purpose of this work is to present an adaptive grid distribution depending on a far-field boundary position and the Peclet condition to solve the Black–Scholes equation accurately and efficiently. Our proposed algorithm is as follows: First, to generate the adaptive grid we determine a priori a suitable far-field location using estimates with a given error tolerance. Then, we put a uniform fine grid around the non-smooth points and a non-uniform grid in the remaining regions. When we choose the non-uniform grid function, we use the Peclet condition which will be defined later.

5.2. Discretization with finite differences

A finite difference method approximates continuous derivatives by difference operators. The finite difference method has been applied to pricing financial contracts for many years [68]. For more details about finite difference methods

in computational finance, we refer the reader to the books [24, 68, 69, 77] and papers [12, 13, 65].

The BS equation is discretized on a non-uniform grid defined by $x_0 = 0$ and $x_{i+1} = x_i + h_i$ for $i = 0, \cdots, N_x - 1$, where N_x is the total number of grid intervals and h_i is the grid spacing, see Fig. 5.1. And we assume $h_{N_x} = h_{N_x - 1}$.

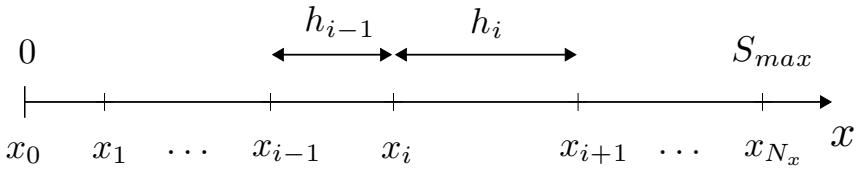


FIGURE 5.1. A non-uniform grid with space step sizes h_i.

Let us denote the numerical approximation of the solution by $u_i^n \equiv u(x_i, \tau^n)$, where $\tau^n = n\Delta\tau$, $\Delta\tau = T/N_\tau$, and N_τ is the total number of time steps. By applying an implicit scheme to Eq. (5.2), we have

$$\frac{u_i^{n+1} - u_i^n}{\Delta\tau} = \frac{\sigma^2 x_i^2}{2}\left(\frac{\partial^2 u}{\partial x^2}\right)_i^{n+1} + r x_i \left(\frac{\partial u}{\partial x}\right)_i^{n+1} - r u_i^{n+1}, \tag{5.4}$$

where first and second derivatives are defined as

$$\begin{aligned}
\left(\frac{\partial u}{\partial x}\right)_i^{n+1} &= -\frac{h_i}{h_{i-1}(h_{i-1}+h_i)} u_{i-1}^{n+1} + \frac{h_i - h_{i-1}}{h_{i-1}h_i} u_i^{n+1} + \frac{h_{i-1}}{h_i(h_{i-1}+h_i)} u_{i+1}^{n+1}, \\
\left(\frac{\partial^2 u}{\partial x^2}\right)_i^{n+1} &= \frac{2}{h_{i-1}(h_{i-1}+h_i)} u_{i-1}^{n+1} - \frac{2}{h_{i-1}h_i} u_i^{n+1} + \frac{2}{h_i(h_{i-1}+h_i)} u_{i+1}^{n+1}.
\end{aligned}$$

We used the same discretization in [56]. We can rewrite Eq. (5.4) as

$$\alpha_i u_{i-1}^{n+1} + \beta_i u_i^{n+1} + \gamma_i u_{i+1}^{n+1} = \frac{u_i^n}{\Delta\tau}, \tag{5.5}$$

where

$$\alpha_i = \frac{-\sigma^2 x_i^2 + r x_i h_i}{h_{i-1}(h_{i-1}+h_i)},$$
$$\beta_i = \frac{\sigma^2 x_i^2 - r x_i (h_i - h_{i-1})}{h_{i-1} h_i} + r + \frac{1}{\Delta\tau},$$
$$\gamma_i = \frac{-\sigma^2 x_i^2 - r x_i h_{i-1}}{h_i(h_{i-1}+h_i)}$$

The linear boundary condition is defined by $\frac{\partial^2 u}{\partial x^2}(S_{\max}, \tau) = 0,\ \forall \tau \in [0, T]$ [56, 68, 78] and we discretize it as $(u_{N_x-1}^{n+1} - 2u_{N_x}^{n+1} + u_{N_x+1}^{n+1})/h_{N_x-1}^2 = 0$. By substituting this into Eq. (5.5) we get

$$\frac{r x_{N_x}}{h_{N_x-1}} u_{N_x-1}^{n+1} + \left(\frac{1}{\Delta\tau} - \frac{r x_{N_x}}{h_{N_x-1}} + r \right) u_{N_x}^{n+1} = \frac{u_{N_x}^n}{\Delta\tau}. \tag{5.6}$$

And we can rewrite Eqs. (5.5) and (5.6) in a matrix form as

$$\begin{pmatrix} \beta_1 & \gamma_1 & 0 & \dots & 0 \\ \alpha_2 & \beta_2 & \gamma_2 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & \alpha_{N_x-1} & \beta_{N_x-1} & \gamma_{N_x-1} \\ 0 & \dots & 0 & \alpha_{N_x} & \beta_{N_x} \end{pmatrix} \begin{pmatrix} u_1^{n+1} \\ u_2^{n+1} \\ \vdots \\ u_{N_x-1}^{n+1} \\ u_{N_x}^{n+1} \end{pmatrix} = \begin{pmatrix} u_1^n/\Delta\tau \\ u_2^n/\Delta\tau \\ \vdots \\ u_{N_x-1}^n/\Delta\tau \\ u_{N_x}^n/\Delta\tau \end{pmatrix},$$

where $\alpha_{N_x} = r x_{N_x}/h_{N_x-1}$ and $\beta_{N_x} = \frac{1}{\Delta\tau} - r x_{N_x}/h_{N_x-1} + r$. We solve this tri-diagonal matrix directly by using the Thomas algorithm.

5.3. Adaptive grid generation technique

An adaptive grid generation technique is used to reduce the number of grid-points maintaining the discretization error at a prescribed level. Examples of algorithms for adaptivity in space and time can be found in [1, 20, 47, 49, 50, 56, 57, 58, 80, 83]. In [56, 57], the authors used an adaptive technique proposed in [47] based on local discretization error. Non-uniform mesh is generated by obtaining more grid-points around the discontinuity. And the grid points move in every time step for optimal distribution of them with moving grid method,

see e.g. [80]. Moreover, the computational grid is refined in blocks and the grid and time step change at every discrete time point in [50]. The authors in paper [20] developed space-time adaptive and high-order methods for valuing American options using PDE approach. In [49], the grid and time step sizes were chosen dynamically to satisfy a bound on the global error at the expiry date. In addition to this, an adaptive finite element discretization was developed in [1, 58] for American options. Also, the authors in [83] applied a discrete singular convolution algorithm with an adaptive mesh.

Now, as an extended work of adaptive grid generation techniques, we propose an improved adaptive grid technique which is based on a far-field boundary position of the BS equation and the Peclet condition for stability of solutions. For a given total number of grid points, we want to optimize the computation in terms of accuracy and efficiency by using adaptive grid techniques. First of all, we need to decide the computational domain size so that we can use a numerical method such as a finite difference method. If we want to get an accurate solution, then the domain size should be large enough. However, a large domain requires more computational resources. Therefore, we want the domain size to be as small as possible while keeping the resulting solution within a given error tolerance. Once the domain size is chosen, we adaptively distribute grid points so that we get an accurate numerical solution using a given grid function.

5.3.1. Far field boundary condition. Let u be classical solution of Eqs. (5.2) and (5.3) on an infinite domain $(0, \infty) \times (0, T]$. Assume

$$-\nu_0 + \nu_1 x \leq p(x) \leq \kappa_0 + \kappa_1 x \ \forall x \in (0,\ \infty) \tag{5.7}$$

for some $\nu_0,\ \nu_1,\ \kappa_0,\ \kappa_1 \geq 0$. Then

$$-\nu_0 e^{-r\tau} + \nu_1 x \leq u(x,\tau) \leq \kappa_0 e^{-r\tau} + \kappa_1 x \ \forall (x,\tau) \in (0,\ \infty) \times (0,T).$$

Also let w be the classical solution of Eqs. (5.2) and (5.3) on a finite domain $(0, S_{\max}) \times (0,T]$. Then at every point $(x,\tau) \in (0, S_{\max}) \times (0,T]$ satisfying

$$\ln \frac{S_{\max}}{x} \geq -(\sigma^2 - 2r)\tau, \tag{5.8}$$

we have

$$|u(x,\tau) - w(x,\tau)| \leq \|u - w\|_{L_\infty(S_{\max}\times(0,\tau))} e^{\frac{-\ln\frac{S_{\max}}{K}\left(\ln\frac{S_{\max}}{K} + \min\{0,\sigma^2-2r\}\tau\right)}{2\sigma^2\tau}}. \tag{5.9}$$

Please refer to [41] for more details about the far-field boundary conditions for the Black–Scholes equations.

5.3.2. Choice of far-field boundary position. First, we consider a European call option, whose payoff function is given as $p(x) = \max(x-K, 0)$. Let $u(x,\tau)$ and $w(x,\tau)$ be solutions of Eq. (5.2) on an infinite domain $(0,\infty)\times(0,T]$ and a finite domain $(0, S_{\max})\times(0,T]$, respectively. For the boundary condition at $x = S_{\max}$, we set $w(S_{\max},\tau) = p(S_{\max})$ for all $\tau \in (0,T)$. Since $-K + x \leq p(x) \leq x$ by Eqs. (5.7) and (5.8), we can say that $-Ke^{-r\tau} + x \leq u(x,\tau) \leq x$. Therefore we have $\sup_{\tau\in(0,T)} |u(S_{\max},\tau) - w(S_{\max},\tau)| \leq K$. Then by Eq. (5.9), for all $x \in [0,K]$, $u(x,\tau)$ and $w(x,\tau)$ satisfy the following inequality:

$$|u(x,\tau) - w(x,\tau)| \leq K \exp\left(-\frac{\ln\frac{S_{\max}}{K}\left(\ln\frac{S_{\max}}{K} + \min\{0,\sigma^2-2r\}\tau\right)}{2\sigma^2\tau}\right).$$

Therefore, if we want the error on the finite domain to be less than K/A, then $S_{\max}$ should satisfy the following inequality:

$$K \exp\left(-\frac{\ln\frac{S_{\max}}{K}\left(\ln\frac{S_{\max}}{K} + \min\{0,\sigma^2-2r\}\tau\right)}{2\sigma^2\tau}\right) \leq K/A.$$

This estimation tells us that if

$$S_{\max} \geq Ke^{-0.5\min\{0,(\sigma^2-2r)\tau\}+0.5\sqrt{(\min\{0,(\sigma^2-2r)\tau\})^2+8\sigma^2\tau\ln A}}, \tag{5.10}$$

then we can be sure that $w(x,\tau)$, the solution of the truncated domain problem, gives us a call option value that is within K/A from the correct value [41]. This estimation is essential when performing numerical approximations of infinite domain problems since we must use a finite domain in finite difference schemes. Therefore, for the accuracy and efficiency of our proposed method, we utilize this error estimation to choose the far-field boundary position.

Next, let us consider the other type of European call options. For example, a cash-or-nothing option has the payoff function $p(x) = \text{Cash}$ if $x > K$ and $p(x) = 0$ otherwise. Similar to the first case, since $\sup_{\tau\in(0,T)} |u(S_{\max},\tau) - w(S_{\max},\tau)| \leq \text{Cash}$, for all $x \in (0, S_{\max})$, $u(x,\tau)$ and $w(x,\tau)$ satisfy the following inequality:

$$|u(x,\tau) - w(x,\tau)| \leq \text{Cash}\ e^{-\frac{\ln\frac{S_{\max}}{K}\left(\ln\frac{S_{\max}}{K}+\min\{0,\sigma^2-2r\}\tau\right)}{2\sigma^2\tau}}.$$

Therefore, if we want the error on the finite domain to be less than K/A, then we need

$$S_{\max} \geq Ke^{-0.5\min\{0,(\sigma^2-2r)\tau\}+0.5\sqrt{\min\{0,(\sigma^2-2r)\tau\}^2+8\sigma^2\tau\ln(A\ \text{Cash}/K)}}. \tag{5.11}$$

5.3.3. Stability condition. In this section, we will derive the conditions under which the implicit scheme for Eq. (5.2) will not make spurious oscillations by using the idea in reference [86]. Let $k_i = \frac{1}{2}\sigma^2 x_i^2$, $a_i = rx_i$ and we rewrite Eq. (5.5) as:

$$\frac{(a_ih_i - k_i)\Delta\tau}{h_{i-1}(h_{i-1}+h_i)}u_{i-1}^{n+1} + \frac{(k_i - a_i(h_i - h_{i-1}))\Delta\tau + (1+r\Delta\tau)h_{i-1}h_i}{h_{i-1}h_i}u_i^{n+1} - \frac{(k_i + a_ih_{i-1})\Delta\tau}{h_i(h_{i-1}+h_i)}u_{i+1}^{n+1} = u_i^n. \tag{5.12}$$

Next, we substitute $u_i^{n+1} = \beta_i^{n+1}/(1+r\Delta\tau)^n$ into Eq. (5.12), where the superscript n for $(1+r\Delta\tau)$ represents an exponent. Then, we obtain

$$\begin{aligned}&\frac{(k_i - a_i(h_i - h_{i-1}))\Delta\tau + (1+r\Delta\tau)h_{i-1}h_i}{h_{i-1}h_i}\beta_i^{n+1} \\ &\quad = (1+r\Delta\tau)\,\beta_i^n + \frac{(k_i - a_i h_i)\Delta\tau}{h_{i-1}(h_{i-1}+h_i)}\beta_{i-1}^{n+1} + \frac{(k_i + a_i h_{i-1})\Delta\tau}{h_i(h_{i-1}+h_i)}\beta_{i+1}^{n+1}.\end{aligned} \tag{5.13}$$

In order for all coefficients of β in Eq. (5.13) to be positive, $k_i - h_i a_i > 0$ should be satisfied. That is, we have the Peclet condition [86]:

$$\frac{1}{h_i} > \frac{r}{\sigma^2 x_i}.$$

Now, if the Peclet condition is satisfied, then all the coefficients of β in Eq. (5.13) are positive. Let $\beta_i^{\max} = \max(\beta_i^n, \beta_{i-1}^{n+1}, \beta_{i+1}^{n+1})$, then Eq. (5.13) can be written as

$$\begin{aligned}&\frac{(k_i - a_i(h_i - h_{i-1}))\Delta\tau + (1+r\Delta\tau)h_{i-1}h_i}{h_{i-1}h_i}\beta_i^{n+1} \\ &\quad \le (1+r\Delta\tau)\,\beta_i^{\max} + \frac{(k_i - a_i h_i)\Delta\tau}{h_{i-1}(h_{i-1}+h_i)}\beta_i^{\max} + \frac{(k_i + a_i h_{i-1})\Delta\tau}{h_i(h_{i-1}+h_i)}\beta_i^{\max}.\end{aligned}$$

Therefore,

$$\beta_i^{n+1} \le \beta_i^{\max}. \tag{5.14}$$

And by a similar argument we obtain

$$\beta_i^{n+1} \ge \beta_i^{\min}, \tag{5.15}$$

where $\beta_i^{\min} = \min(\beta_i^n, \beta_{i-1}^{n+1}, \beta_{i+1}^{n+1})$. By Eqs. (5.14) and (5.15), new local maxima or minima of the numerical solution for β_i^{n+1} can not occur. Since $u_i^{n+1} = \beta_i^{n+1}/(1+r\Delta\tau)^n$, the numerical solution u_i^{n+1} does not contain oscillations if conditions (5.14) and (5.15) are satisfied.

5.3.4. Non-uniform grid generation process with the Peclet condition. The adaptive grid generation process aims to creat a grid with a uniform

fine grid around the strike price K and an increasingly large grid size as we move toward the far-field boundary. To do this, we propose a grid generating function $h(x)$ based on the Peclet condition

$$h(x) = \begin{cases} p(x - K - (m - 0.5)\bar{h})^d + \bar{h} & \text{if } x \geq K + (m - 0.5)\bar{h}, \\ p(x - K + (m - 0.5)\bar{h})^d + \bar{h} & \text{if } x \leq K - (m - 0.5)\bar{h}, \end{cases}$$

where p, d, and $\bar{h}$ are real positive numbers and m is a natural number. First, we allocate $2m$ grid points around the strike price K with a grid size of $\bar{h}$, see Fig. 5.2. Then, we start at $x_i = K + (m - 0.5)\bar{h}$ and define $x_{i+1} = x_i + h(x_i)$. We continue this procedure until we reach the point where $x_{N_x-1} \leq S_{\max} < x_{N_x}$, at this stage we reset $h_{N_x-1} = h_{N_x-2}$. Similarly, for the grid generation of left side, we start at $x_i = K - (m - 0.5)\bar{h}$ and define $x_{i-1} = x_i - h(x_i)$. We continue this process until we reach the point where $x_0 \leq 0 < x_1$. If $x_0 < 0$, then we redefine $x_0 = 0$. This procedure is described schematically in Fig. 5.2. In this figure, we also show an illustration of initial and later solutions on the adaptive grid.

Now, for numerical solutions which are free of spurious oscillations, the space step size must satisfy the following Peclet condition (5.16) [86]:

$$h_i < \frac{\sigma^2}{r} x_i. \tag{5.16}$$

In this work, we will choose a piecewise linear grid function $h(x)$ whose slope is less than σ^2/r to obtain non-oscillatory solutions. We will use the parameter $p = 0.05\sigma^2/r$ and $d = 1$ for numerical examples.

5.4. Computational results

In this section, we perform numerical experiments to test the proposed method. The main focus of these tests is on the performance of the proposed adaptive grid technique compared to the standard uniform grid method. As the benchmark

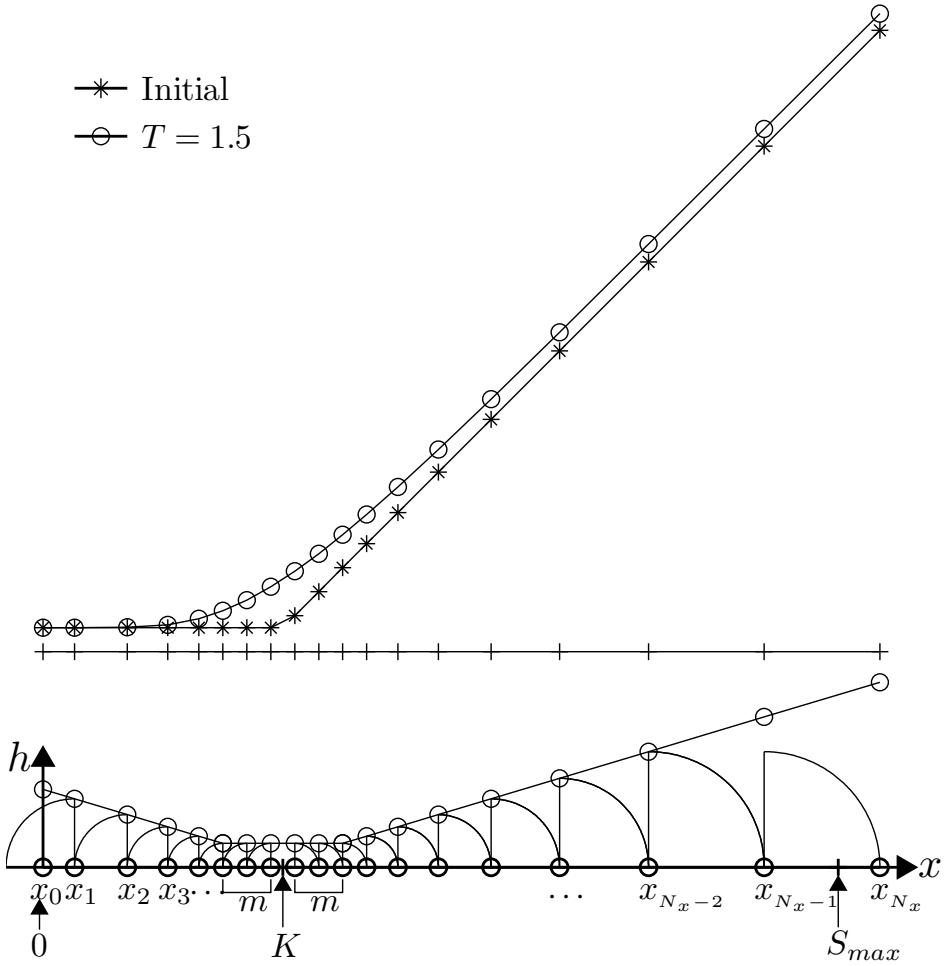


FIGURE 5.2. Construction of the adaptive grid using the function $h(x)$.

problems, we consider the European option problems for numerical examples. These problems are of great interest to academicians in the finance literature and often used to show the accuracy of a given numerical scheme [15, 27, 30].

5.4.1. Uniform grid. The effect of the computational domain size and the total time for a European vanilla call option using a uniform grid is studied. The parameters $\sigma = 0.35, r = 0.05$, and space step size $h = 1$ are used. The computational domain is $\Omega = (0, L)$. For each case, we ran the calculation up

to time T with a uniform time step of $\Delta\tau = 0.01$. The initial condition is $u(x,0) = \max(x - K, 0)$ with the strike price $K = 100$.

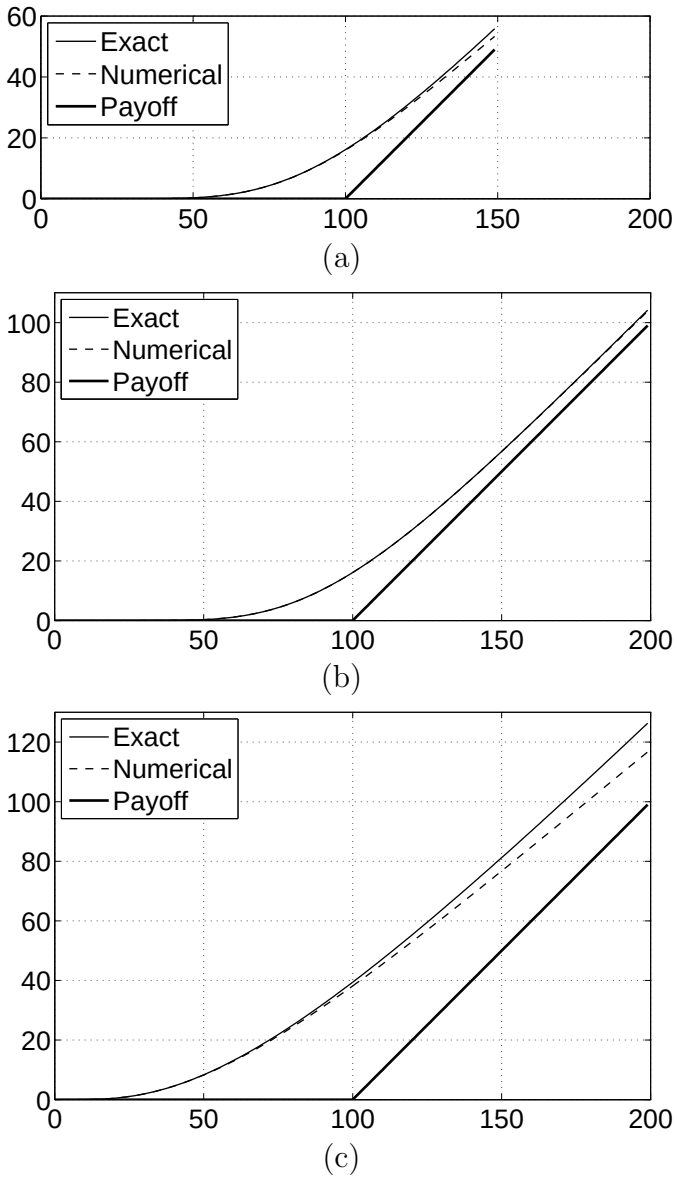


FIGURE 5.3. Initial profiles and numerical, exact results with respect to different domain sizes and times: (a) $L = 150$, $T = 1$, (b) $L = 200$, $T = 1$, and (c) $L = 200$, $T = 5$.

For this European call option, the closed form solution of the Black–Scholes equation is

$$u(x,\tau) = xN(d_1) - Ke^{-r\tau}N(d_2),\ \forall x \in [0,L],\ \forall \tau \in [0,T]$$
$$d_1 = \frac{\log(x/K) + (r + \frac{1}{2}\sigma^2)\tau}{\sigma\sqrt{\tau}},\ d_2 = d_1 - \sigma\sqrt{\tau},$$

where $N(d) = (1/\sqrt{2\pi})\int_{-\infty}^{d} \exp(-x^2/2)\,dx$ is the cumulative distribution function for the standard normal distribution [6].

Figures 5.3(a), (b), and (c) show the initial profile, numerical, and exact solutions at time T. When $L = 150$ and $T = 1$, we can observe a large deviation of numerical solution from the exact solution (see Fig. 5.3(a)). With increased domain size $L = 200$, we get a good result in Fig. 5.3(b). However, when we increase the time to $T = 5$, we again have a large deviation between the numerical and exact solutions (see Fig. 5.3(c)). This result implies that we need a large enough domain size in relation to the size of T.

5.4.2. Adaptive grid. To demonstrate the performance of the proposed adaptive grid technique with a far-field boundary condition and the Peclet condition, we compare the numerical results of both the uniform grid method and our proposed adaptive grid method. For the comparison study, we set $\sigma = 0.35,\ r = 0.05,\ \Delta\tau = 0.001$, $p = 0.05\sigma^2/r$, and $d = 1$. Also, the far-field boundary position $S_{\max}$ is set to achieve an accuracy of $K/A = 0.1$ according to the conditions (5.10) and (5.11) for each case. To compare numerical results on uniform and adaptive grids, we use the relative root mean square error (RMSE) on $[0.95K,\ 1.05K]$. Here, the relative RMSE is defined as

$$\text{Relative RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\frac{u_i - u(x_i)}{u(x_i)}\right)^2},$$

where N is the total number of points on $[0.95K,\ 1.05K]$, u_i and $u(x_i)$ are numerical and exact solutions, respectively.

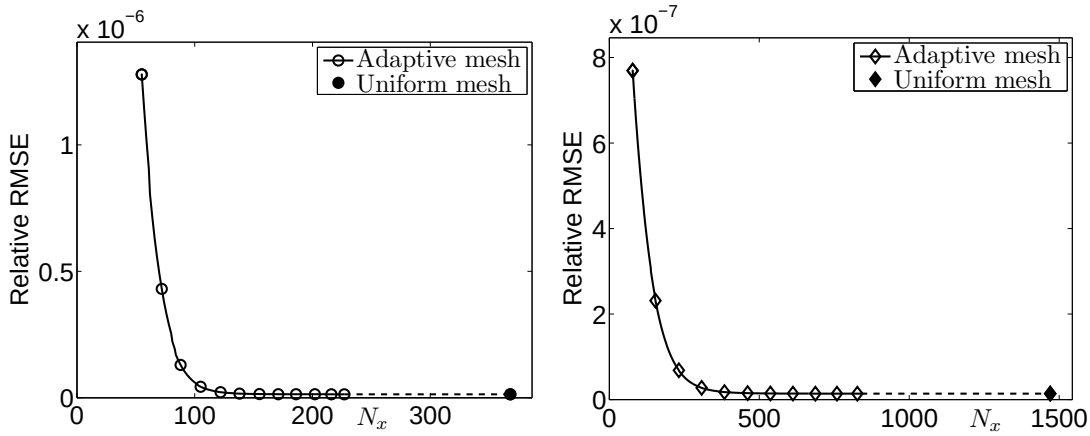
5.4.2.1. *Call option on a maximum of one asset.* The initial state is given by $u(x,0) = \max(x - 100, 0)$. Table 5.1 shows computational results such as RMSE on $[0.95K,\ 1.05K]$, relative CPU time, and grid points N_x at time $T = 1$ with adaptive and uniform grids. In this numerical test, we use ten different space steps: $\bar{h} = 2/2^m$, where $m = 0, 1, 2, \cdots, 9$. As shown in Table 5.1, the CPU times taken from the uniform grids are larger than those of the adaptive grid method. Also, the total number of grid points N_x on the adaptive grid is much smaller than on the uniform grid. Overall, the adaptive grid outperforms the uniform grid.

Case	RMSE	CPU time		N_x	
		Adaptive	Uniform	Adaptive	Uniform
$\bar{h} = 2$	1.399E-6	1	1	88	185
$\bar{h} = 1$	1.399E-6	1	1.5	149	368
$\bar{h} = 1/2$	1.399E-6	1	4.5	259	736
$\bar{h} = 1/2^2$	1.399E-6	1	5.75	461	1470
$\bar{h} = 1/2^3$	1.399E-6	1	14.50	851	2939
$\bar{h} = 1/2^4$	1.399E-6	1	24.73	1619	5877
$\bar{h} = 1/2^5$	1.399E-6	1	34.61	3137	11754
$\bar{h} = 1/2^6$	1.399E-6	1	44.45	6165	23506
$\bar{h} = 1/2^7$	1.399E-6	1	58.99	12207	47011
$\bar{h} = 1/2^8$	1.399E-6	1	78.99	24301	94022

TABLE 5.1. Comparison of relative CPU time and grid points N_x on adaptive and uniform grids at time $T = 1$ with call option on the maximum of one asset.

Figure 5.4 shows the result of the RMSE on $[0.95K,\ 1.05K]$ with different N_x at time $T = 1$. In Fig. 5.4, '$\circ$' and '$\diamond$' show the results using $\bar{h} = 1$ and 0.25, respectively and '$\bullet$' and '$\blacklozenge$' represent $\bar{h} = 1$ and 0.25 on the uniform mesh,

respectively. From these results, we see the convergence of the relative RMSE of the adaptive grid as the number of grid points around the strike price K increases.



(a) $\bar{h} = 1$ (b) $\bar{h} = 0.25$

FIGURE 5.4. Relative RMSE on $[0.95K,\ 1.05K]$ with different N_x at $T = 1$. Lines with symbols '$\circ$' and '$\diamond$' represent $\bar{h} = 1$ and 0.25 on the adaptive mesh, respectively. Also, symbols '$\bullet$' and '$\blacklozenge$' represent $\bar{h} = 1$ and 0.25 on the uniform mesh, respectively.

5.4.2.2. *Cash-or-nothing option.* Next, we perform the same comparison study using a cash-or-nothing option. The initial state is given by

$$u(x,0) = \begin{cases} Cash & \text{if } x \le K \\ 0 & \text{otherwise.} \end{cases}$$

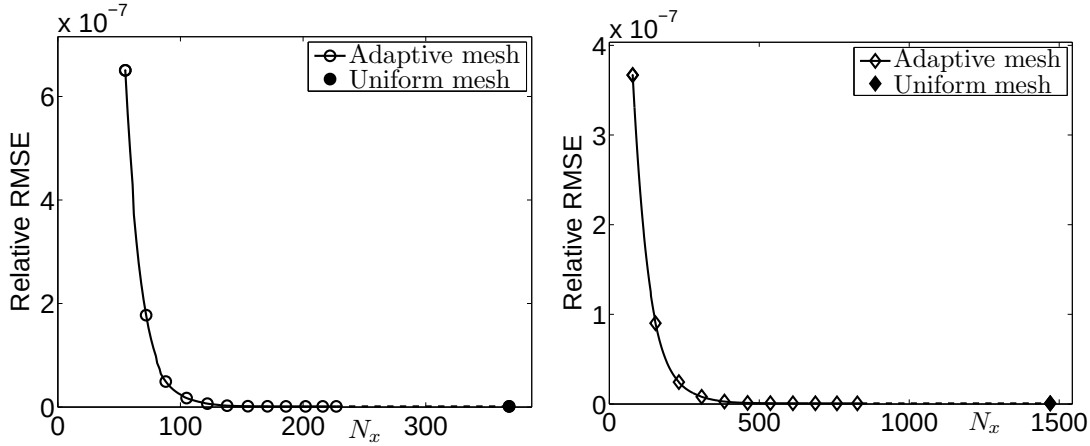
For this test, we use $Cash = 100$. Table 5.2 shows computational results such as relative CPU time, RMSE on $[0.95K, 1.05K]$ and grid points, N_x at time $T = 1$ on adaptive and uniform grids with ten different space step size $h = 2/2^m$, where $m = 0, 1, 2, \cdots, 9$. We can see that the adaptive grid technique is more efficient than uniform grid.

Figure 5.5 shows the result of the RMSE on $[0.95K,\ 1.05K]$ with different N_x at time $T = 1$. In Fig. 5.5, '$\circ$' and '$\diamond$' show the results of $\bar{h} = 1$ and 0.25 on the adaptive mesh, respectively. And symbols '$\bullet$' and '$\blacklozenge$' represent $\bar{h} = 1$ and 0.25

Case	RMSE	CPU time		N_x	
		Adaptive	Uniform	Adaptive	Uniform
$\bar{h} = 2$	1.352E-7	1	1.36	88	185
$h = 1$	7.994E-7	1	1.5	155	368
$h = 1/2$	7.994E-7	1	4.0	232	736
$h = 1/2^2$	7.994E-7	1	6.0	394	1470
$h = 1/2^3$	7.994E-7	1	18.88	710	2939
$h = 1/2^4$	7.994E-7	1	45.15	1332	5877
$h = 1/2^5$	7.994E-7	1	47.59	2564	11754
$h = 1/2^6$	7.994E-7	1	54.10	5014	23506
$h = 1/2^7$	7.994E-7	1	71.31	9904	47011
$h = 1/2^8$	7.994E-7	1	108.02	19672	94022

TABLE 5.2. Comparison of relative CPU time and grid points, N_x on adaptive and uniform grids at time $T = 1$ for a given relative RMSE tolerance with a cash-or-nothing option payoff.

on the uniform mesh, respectively. From these results, we see the convergence of the relative RMSE of the adaptive grid as the number of grid points around the strike price K increases.



(a) $\bar{h} = 1$ (b) $\bar{h} = 0.25$

FIGURE 5.5. Relative RMSE on $[0.95K,\ 1.05K]$ with different N_x at $T = 1$. Lines with symbols '$\circ$' and '$\diamond$' represent $\bar{h} = 1$ and 0.25 on the adaptive mesh, respectively. Also, symbols '$\bullet$' and '$\blacklozenge$' represent $\bar{h} = 1$ and 0.25 on the uniform mesh, respectively.

5.5. Conclusions

An accurate and efficient numerical method for solving the Black–Scholes equation was derived in this chapter. The method uses an adaptive technique which is based on a far-field boundary position of the BS equation and the Peclet condition for non-oscillatory solutions. In this chapter, we stated only the one-dimensional problems. However, this adaptive grid generation on multi-dimension is also simply extended. For example, we briefly present two-dimensional problem with a European vanilla call option using adaptive grid generating technique. The two-dimensional Black–Scholes equation is discretized on a non-uniform grid defined by $x_0 = y_0 = 0,\ x_{i+1} = x_i + h_i,\ y_{j+1} = y_j + h_j$ for $i = 0, 1, \cdots, N_x - 1$ and $j = 0, 1, \cdots, N_y - 1$ where $N_x,\ N_y$ are the total number of grid points on each x- and y-axis. Here, $S_{\max}$ is decided by the far-field boundary condition. For details, one may refer to [41]. The following Fig. 5.6 represents adaptive grid generated by such assumptions.

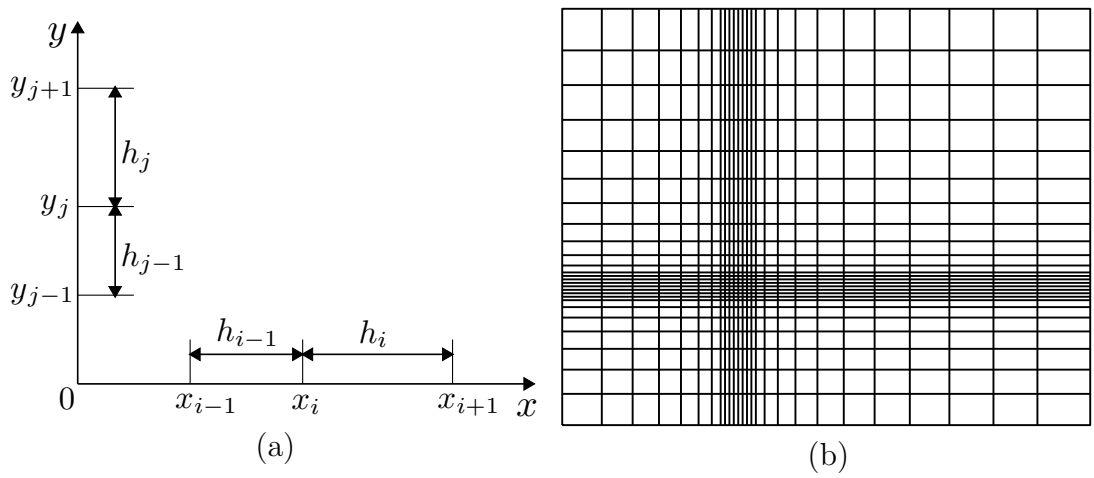


FIGURE 5.6. (a) Space step sizes h_i and h_j which are defined on two-dimensional non-uniform grid and (b) two-dimensional adaptive mesh.

And on these two-dimensional adaptive grids, we can solve the Black–Scholes equation by using the fast and accurate numerical method such as operator splitting methods [19, 24, 36, 82]. As dimension increases, the adaptive technique will decrease computational costs than on uniform grids while maintaining the accuracy. Furthermore, since the proposed adaptive grid method is based on a far-field boundary position and the Peclet condition for non-oscillatory solutions, we get efficient and accurate numerical solutions.

In this chapter, to demonstrate the accuracy and efficiency of our proposed method, numerical tests were performed. Test results show that the computational time on the adaptive grid was reduced substantially when compared to the uniform grid. From these numerical results, we confirmed the effectiveness of the proposed adaptive grid method.

Chapter 6

An operator splitting method for pricing the ELS option

This work presents the numerical valuation of the two-asset step-down equity-linked securities (ELS) option by using the operator splitting method (OSM). The ELS is one of the most popular financial options. The value of ELS option can be modeled by a modified Black–Scholes partial differential equation. However, regardless of whether there is a closed-form solution, it is difficult and not efficient to evaluate the solution because such a solution would be represented by multiple integrations. Thus, a fast and accurate numerical algorithm is needed to value the price of the ELS option. This chapter uses a finite difference method to discretize the governing equation and applies the OSM to solve the resulting discrete equations. The OSM is very robust and accurate in evaluating finite difference discretizations. We provide a detailed numerical algorithm and computational results showing the performance of the method for two underlying asset option pricing problems such as cash-or-nothing and step-down ELS. Final option value of two-asset step-down ELS is obtained by a weighted average value using probability which is estimated by performing a MC simulation.

6.1. Introduction

Equity-linked securities (ELS) are securities whose return on investment is dependent on the performance of the underlying equities linked to the securities. Since ELS were introduced to Korea in 2003, the booming world economy and

expanding financial markets have shifted funds previously focused on real estate to new investment vehicles. The ELS option represents one of the new investment vehicles in that they can be used to structure various products according to the needs of investors. We can model the value of the ELS option by a modified Black–Scholes partial differential equation (BSPDE) [1, 24, 66, 68, 69, 77]. Typically, there is no closed-form solution, and even if there were such a solution, evaluating it would be difficult because it would be represented by multiple integrations. Therefore, a fast and accurate numerical algorithm is needed to price the ELS option. We use a finite difference method to discretize the BSPDE and apply the operator splitting method (OSM) [24] to solve the resulting discrete equations. The basic idea behind the OSM is to reduce multi-dimensional equations into multiple one-dimensional problems. The OSM is very robust and accurate in evaluating finite difference discretizations.

6.2. Two-asset step-down ELS

The payoff of two-asset step-down ELS is as follows:

- Early obligatory redemption occurs and a given rate of return is paid if the value of the worst performer is greater than or equal to the prescribed exercise price on the given observation date. Here, Here the worst performer is defined as one of the two underlying assets whose value is lower than that of the other.
- If early obligatory redemptions did not occur until the maturity time, then the return is determined by the Knock-In criterion.

The basic parameters of two-asset step-down ELS are as follows:

- Maturity : T
- Face value : F
- Underlying assets at time t : $x(t)$ and $y(t)$
- Worst performer : $S_t = \min[x(t),\ y(t)]$
- Conditions for early redemption : Let N be the number of observation dates.

Observation date	t_1	t_2	$\cdots$	$t_N = T$
Exercise price	K_1	K_2	$\cdots$	K_N
Rate of return	c_1	c_2	$\cdots$	c_N

Case 1) Early obligatory redemptions happened

If the value of the worst performer S_{t_i} is greater than or equal to the exercise price K_i at time $t = t_i$, then $(1 + c_i)F$ is paid, and the contract expires.

Case 2) Early obligatory redemptions did not happen

Let D denote the Knock-In barrier level and d denote a dummy.

(i) If a Knock-in event does not occur, that is,
$m_T = \min\{S_t|\ 0 \le t \le T\} > D$, then $(1 + d)F$ is paid.

(ii) If a Knock-in event occurs, $(1 + S_T/S_0)\,F$ is paid.

We now summarize the payoff function. Let $\chi_i = \chi_{A_i}$, where χ_i denotes the characteristic function of $A_i = \{x \ge K_i \text{ and } y \ge K_i\}$. Here K_i is the exercise price at time t_i. Let $u(x, y, t)$ denote the value of the option. Generally, the

payoff function of two-asset step-down ELS is constructed as follows:

$$u(x,y,t_i) = \begin{cases} \chi_1 = 1 \quad \text{Payoff} = (1+c_1)F \\ \chi_1 = 0 \quad \begin{cases} \chi_2 = 1 \quad \text{Payoff} = (1+c_2)F \\ \chi_2 = 0 \quad \begin{cases} \chi_3 = 1 \quad \text{Payoff} = (1+c_3)F \\ \chi_3 = 0 \quad \begin{cases} \chi_4 = 1 \quad \text{Payoff} = (1+c_4)F \\ \chi_4 = 0 \quad \begin{cases} m_T > D, \text{ then} \\ \text{Payoff} = (1+d)F \\ m_T \le D, \text{ then} \\ \text{Payoff} = (1+S_T/S_0)F \end{cases} \end{cases} \end{cases} \end{cases} \end{cases}$$

In this chapter, we chose the following parameters: the reference price $K_0 = 100$, the interest rate $r = 5\%$, the volatilities of the underlying assets $\sigma_1 = 25\%, \sigma_2 = 30\%$, the total time $T = 1$ year, the face price $F = 100$, the Knock-In barrier level $D = 0.6K_0$, and the dummy rate $d = 16\%$. The other parameters are listed in Table 6.1.

Observation date	t_1	t_2	t_3	$t_4 = T$
Exercise price	$K_1 = 0.90K_0$	$K_2 = 0.85K_0$	$K_3 = 0.80K_0$	$K_4 = 0.75K_0$
Return rate	$c_1 = 5.5\%$	$c_2 = 11\%$	$c_3 = 16.5\%$	$c_4 = 22\%$

TABLE 6.1. Parameters of two-asset step-down ELS.

Figure 6.1 shows the profit-and-loss diagram of two-asset step-down ELS.

6.3. Numerical solution

In this section, we describe the numerical discretization of Eq. (1.3). We also present the operator splitting algorithm in detail.

6.3.1. Discretization. Let $\mathcal{L}_{BS}$ be the operator

$$\mathcal{L}_{BS} = \frac{1}{2}\sigma_1^2 x^2 \frac{\partial^2 u}{\partial x^2} + \frac{1}{2}\sigma_2^2 y^2 \frac{\partial^2 u}{\partial y^2} + \rho\sigma_1\sigma_2 xy \frac{\partial^2 u}{\partial x \partial y} + rx\frac{\partial u}{\partial x} + ry\frac{\partial u}{\partial y} - ru.$$

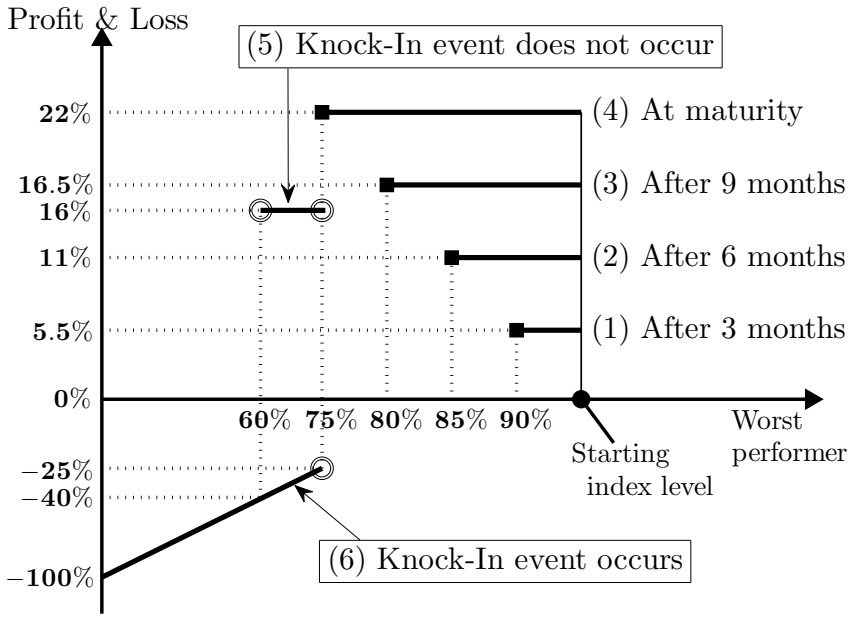


FIGURE 6.1. Profit-and-loss diagram at early redemption and maturity for two-asset step-down ELS.

Then the two-dimensional Black–Scholes equation can be rewritten as

$$\frac{\partial u}{\partial \tau} = \mathcal{L}_{BS}, \text{ for } (x, y, \tau) \in \Omega \times (0, T],$$

where $\tau = T - t$ and T is the expiration time. The original option pricing problems are defined in the unbounded domain $\{(x, y, \tau) \mid x \geq 0, y \geq 0, \tau \in [0, T]\}$. We truncate this domain into a finite computational domain $\{(x, y, \tau) \mid 0 \leq x \leq L, 0 \leq y \leq M, \tau \in [0, T]\}$, where L and M are large enough so that the error of the price u arisen by the truncation is negligible. For example, L and M can be two or three times greater than the exercise price [41]. We have the linear boundary conditions [55, 56, 68, 89] for the artificial boundaries

$$\frac{\partial^2}{\partial x^2} u(0, y, \tau) = \frac{\partial^2}{\partial x^2} u(L, y, \tau) = \frac{\partial^2}{\partial y^2} u(x, 0, \tau) = \frac{\partial^2}{\partial y^2} u(x, M, \tau) = 0,$$
$$\text{for } 0 \leq x \leq L, \ 0 \leq y \leq M \text{ and } 0 \leq \tau \leq T.$$

The numbers of grid steps are denoted by N_x, N_y, and N_τ in the x-, y- and τ-directions, respectively. We first discretize the given computational domain $\Omega = (0, L) \times (0, M)$ as a uniform grid with a space step $h = L/N_x = M/N_y$ and a time step $\Delta\tau = T/N_\tau$. Denote the numerical approximation of the solution by

$$u_{ij}^n \equiv u(x_i, y_j, \tau^n) = u\left((i-0.5)h, (j-0.5)h, n\Delta\tau\right),$$

where $i = 1, \ldots, N_x$, $j = 1, \ldots, N_y$ and $n = 0, \ldots, N_\tau$. We use a cell-centered discretization because we use the following linear boundary condition:

$$u_{0j} = 2u_{1j} - u_{2j}, u_{N_x+1,j} = 2u_{N_x,j} - u_{N_x-1,j} \text{ for } j = 1, \cdots, N_y,$$

$$u_{i0} = 2u_{i1} - u_{i2}, u_{i,N_y+1} = 2u_{i,N_y} - u_{i,N_y-1} \text{ for } i = 1, \cdots, N_x.$$

6.3.2. Operator splitting method. The basic idea behind the operator splitting method is to reduce multi-dimensional equations into multiple one-dimensional problems [24]. We introduce the basic OS scheme for the two-dimensional Black–Scholes equation as follows:

$$\frac{u_{ij}^{n+1} - u_{ij}^n}{\Delta\tau} = \mathcal{L}_{BS}^x u_{ij}^{n+\frac{1}{2}} + \mathcal{L}_{BS}^y u_{ij}^{n+1}, \tag{6.1}$$

where the discrete difference operators $\mathcal{L}_{BS}^x$ and $\mathcal{L}_{BS}^y$ are defined by

$$\begin{aligned}
\mathcal{L}_{BS}^x u_{ij}^{n+\frac{1}{2}} &= \frac{(\sigma_1 x_i)^2}{2} \frac{u_{i-1,j}^{n+\frac{1}{2}} - 2u_{ij}^{n+\frac{1}{2}} + u_{i+1,j}^{n+\frac{1}{2}}}{h^2} + r x_i \frac{u_{i+1,j}^{n+\frac{1}{2}} - u_{ij}^{n+\frac{1}{2}}}{h} - \lambda_2 r u_{ij}^{n+\frac{1}{2}} \\
&\quad + \lambda_1 \sigma_1 \sigma_2 \rho x_i y_j \frac{u_{i+1,j+1}^n + u_{ij}^n - u_{i,j+1}^n - u_{i+1,j}^n}{h^2}, \\
\mathcal{L}_{BS}^y u_{ij}^{n+1} &= \frac{(\sigma_2 y_j)^2}{2} \frac{u_{i,j-1}^{n+1} - 2u_{ij}^{n+1} + u_{i,j+1}^{n+1}}{h^2} + r y_j \frac{u_{i,j+1}^{n+1} - u_{ij}^{n+1}}{h} - (1-\lambda_2) r u_{ij}^{n+1} \\
&\quad + (1-\lambda_1) \sigma_1 \sigma_2 \rho x_i y_j \frac{u_{i+1,j+1}^{n+\frac{1}{2}} + u_{ij}^{n+\frac{1}{2}} - u_{i,j+1}^{n+\frac{1}{2}} - u_{i+1,j}^{n+\frac{1}{2}}}{h^2}.
\end{aligned}$$

The first step is implicit in the x-direction, whereas the second step is implicit in the y-direction. The OS scheme moves from the time level n to an intermediate

time level $n+\frac{1}{2}$ and then to the time level $n+1$. Through this process, the OS method is to split two problems. We then approximate each subproblem by an implicit scheme:

$$\frac{u_{ij}^{n+\frac{1}{2}} - u_{ij}^{n}}{\Delta\tau} = \mathcal{L}_{BS}^{x} u_{ij}^{n+\frac{1}{2}}, \tag{6.2}$$

$$\frac{u_{ij}^{n+1} - u_{ij}^{n+\frac{1}{2}}}{\Delta\tau} = \mathcal{L}_{BS}^{y} u_{ij}^{n+1}. \tag{6.3}$$

Note that combining two Eqs. (6.2) and (6.3) results in Eq. (6.1). The following describes an algorithm of the OS method.

Algorithm OS

- *Step 1*

  Eq. (6.2) is rewritten as follows. For each j, we have

$$\alpha_i u_{i-1j}^{n+\frac{1}{2}} + \beta_i u_{ij}^{n+\frac{1}{2}} + \gamma_i u_{i+1j}^{n+\frac{1}{2}} = f_{ij}, \tag{6.4}$$

  where

$$\alpha_i = -\frac{1}{2}\frac{\sigma_1^2 x_i^2}{h^2},\ \beta_i = \frac{1}{\Delta\tau} + \frac{\sigma_1^2 x_i^2}{h^2} + \frac{r x_i}{h} + \lambda_2 r,$$
$$\gamma_i = -\frac{1}{2}\frac{\sigma_1^2 x_i^2}{h^2} - \frac{r x_i}{h},\ \text{for } i = 1, ..., N_x$$

  and

$$f_{ij} = \lambda_1 \rho \sigma_1 \sigma_2 x_i y_j \frac{u_{i+1,j+1}^{n} - u_{i+1,j}^{n} - u_{i,j+1}^{n} + u_{ij}^{n}}{h^2} + \frac{u_{ij}^{n}}{\Delta\tau}. \tag{6.5}$$

  The first step of the OS method is then implemented in a loop over the y-direction:

for $j = 1 : N_y$

for $i = 1 : N_x$

Set f_{ij} by Eq. (6.5)

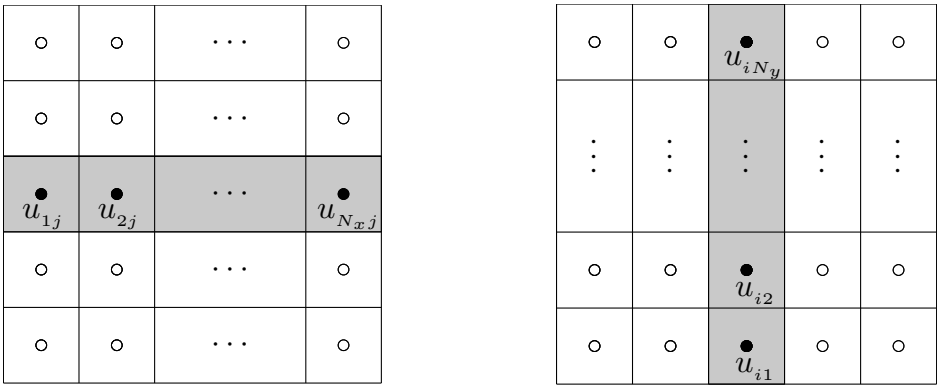
end

Solve $A_x u^{n+\frac{1}{2}}_{1:N_x,j} = f_{1:N_x,j}$ by using Thomas algorithm (see Fig. 6.2(a))

end

Here the matrix A_x is a tridiagonal matrix constructed from Eq. (6.4) with a linear boundary condition

$$A_x = \begin{pmatrix} 2\alpha_1 + \beta_1 & \gamma_1 - \alpha_1 & 0 & \dots & 0 & 0 \\ \alpha_2 & \beta_2 & \gamma_2 & \dots & 0 & 0 \\ 0 & \alpha_3 & \beta_3 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \beta_{N_x-1} & \gamma_{N_x-1} \\ 0 & 0 & 0 & \dots & \alpha_{N_x} - \gamma_{N_x} & \beta_{N_x} + 2\gamma_{N_x} \end{pmatrix}.$$



(a) Step 1 (b) Step 2

FIGURE 6.2. Two steps of the OSM.

- *Step 2*

As in *Step 1*, Eq. (6.3) is rewritten as follows:

$$\alpha_j u_{i,j-1}^{n+1} + \beta_j u_{ij}^{n+1} + \gamma_j u_{i,j+1}^{n+1} = g_{ij}, \tag{6.6}$$

where

$$\alpha_j = -\frac{1}{2}\frac{\sigma_2^2 y_j^2}{h^2},\ \beta_j = \frac{1}{\Delta\tau} + \frac{\sigma_2^2 y_j^2}{h^2} + \frac{r y_j}{h} + (1-\lambda_2)r,$$
$$\gamma_j = -\frac{1}{2}\frac{\sigma_2^2 y_j^2}{h^2} - \frac{r y_j}{h},\ \text{for } j = 1, ..., N_y$$

and

$$g_{ij} = (1-\lambda_1)\rho\sigma_1\sigma_2 x_i y_j \frac{u_{i+1,j+1}^{n+\frac{1}{2}} - u_{i+1,j}^{n+\frac{1}{2}} - u_{i,j+1}^{n+\frac{1}{2}} + u_{ij}^{n+\frac{1}{2}}}{h^2} + \frac{u_{ij}^{n+\frac{1}{2}}}{\Delta\tau}. \tag{6.7}$$

As with *Step 1*, *Step 2* is then implemented in a loop over the x-direction:

for $\quad i = 1 : N_x$

  for $\quad j = 1 : N_y$

    Set g_{ij} by Eq. (6.7)

  end

  Solve $A_y u_{i,1:N_y}^{n+1} = g_{i,1:N_y}$ by using Thomas algorithm (see Fig. 6.2(b))

end

Here A_y is tridiagonal matrix constructed from Eq. (6.6) with a linear boundary condition

$$A_y = \begin{pmatrix} 2\alpha_1 + \beta_1 & -\alpha_1 + \gamma_1 & 0 & \dots & 0 & 0 \\ \alpha_2 & \beta_2 & \gamma_2 & \dots & 0 & 0 \\ 0 & \alpha_3 & \beta_3 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \beta_{N_y-1} & \gamma_{N_y-1} \\ 0 & 0 & 0 & \dots & \alpha_{N_y} - \gamma_{N_y} & \beta_{N_y} + 2\gamma_{N_y} \end{pmatrix}.$$

6.4. Computational results

This section presents the convergence test (which determined the accuracy of the OS method) and the numerical experiments for two-asset step-down ELS.

6.4.1. Convergence test. Since the two-asset cash-or-nothing option can be useful building block for constructing more complex and exotic option products, consider the European two-asset cash-or-nothing call option [34]. Given two stock prices x and y, the payoff of the call option is

$$u(x, y, 0) = \begin{cases} \text{Cash} & \text{if} \quad x \geq K_1 \text{ and } y \geq K_2, \\ 0 & \text{otherwise,} \end{cases} \tag{6.8}$$

where K_1 and K_2 are the strike prices of x and y, respectively. The formula for the exact value of the cash-or-nothing option is known [34]. To estimate the convergence rate, we performed numerical simulations with a set of increasingly finer grids up to $T = 1$. We considered a computational domain, $\Omega = [0, 300] \times [0, 300]$. The initial condition was Eq. (6.8) with the strike prices $K_1 = K_2 = 100$ and Cash $= 1$. The volatilities were $\sigma_1 = 0.25$, $\sigma_2 = 0.3$, the correlation was $\rho = 0.5$, and the risk-free interest rate was $r = 0.05$. Also, the weighting factors were $\lambda_1 = \lambda_2 = 0.5$. The error of the numerical solution was defined as $e_{ij} = u^e_{ij} - u_{ij}$ for $i = 1, \cdots, N_x$ and $j = 1, \cdots, N_y$, where u^e_{ij} is the exact solution and u_{ij} is the numerical solution. We computed discrete l^2 norm of the error, $\|\mathbf{e}\|_2$. We also used the root mean square error (RMSE). The RMSE was defined as

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{i,j}^{N} \left(u^e_{ij} - u_{ij}\right)^2},$$

where N is the number of points on the gray region in Fig. 6.3.

Table 6.2 shows the discrete l^2 norms of the errors in a quarter of the domain, $[0, 150] \times [0, 150]$, the RMSE which is estimated in the gray region shown in Fig.

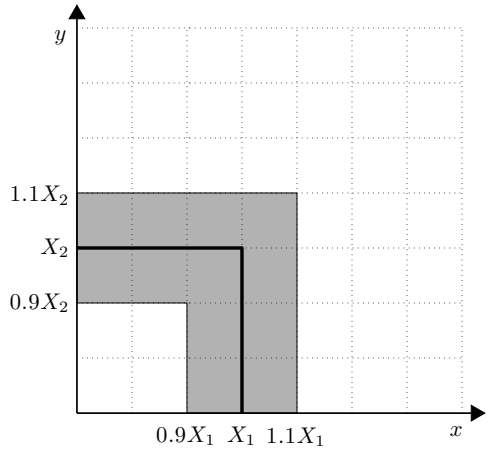


FIGURE 6.3. The gray region is part where the RMSE is estimated.

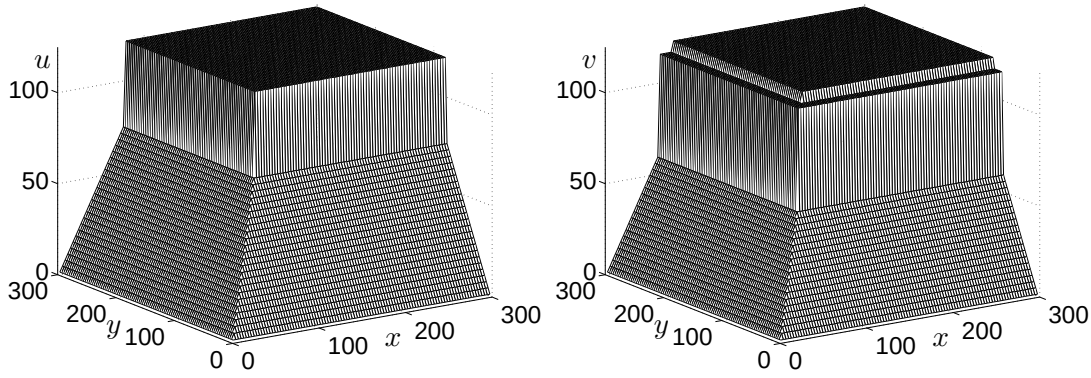
6.3 and the rates of convergence for $\|\mathbf{e}\|_2$ and RMSE. The results suggest that the scheme has first-order accuracy and the RMSE has second-order accuracy in space and time.

Mesh	h	Δt	$\|\mathbf{e}\|_2$	order	RMSE	order
128×128	2.3437	0.1000	0.005344		0.000177	
256×256	1.1719	0.0500	0.002716	0.9764	0.000053	1.7397
512×512	0.5859	0.0250	0.001335	1.0246	0.000011	2.2685
1024×1024	0.2930	0.0125	0.000679	0.9754	0.000003	1.8745

TABLE 6.2. Convergence test.

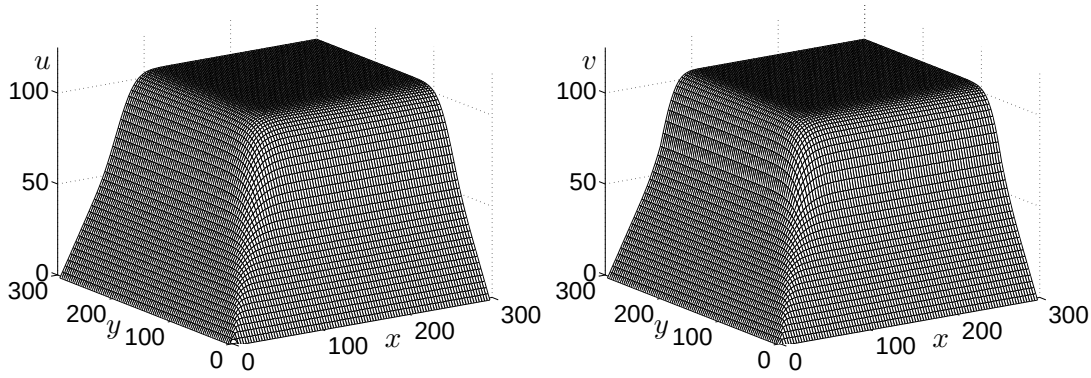
6.4.2. Numerical test of a two-asset step-down ELS. Let u and v be the solutions with payoffs which knock-in event happens and does not happen, respectively. Fig. 6.4(a) and (b) show the initial configurations of u and v, respectively.

And Fig. 6.5(a) and (b) show the final profiles of u and v, respectively, at $T = 1$ with $N_x = N_y = 100$, $K_0 = 100$, $L = 300$, and the parameters listed in Table. 6.1.



(a) Initial u (b) Initial v

FIGURE 6.4. (a) and (b) are the initial conditions for u and v, respectively.



(a) Final u (b) Final v

FIGURE 6.5. (a) and (b) are the numerical results for u and v, respectively, at $T = 1$.

The final two-asset step-down ELS price is obtained by a weighted average of u and v by each probability. By performing a Monte Carlo (MC) simulation [45] for 20000 samples, we estimated that a knock-in event occurs with a probability of approximately 0.1. Therefore, we defined the final ELS value as $0.1u + 0.9v$. Fig. 6.6 (a) shows the weighted average value $0.1u + 0.9v$, and (b) shows the overlapped contour lines of the weighted average values.

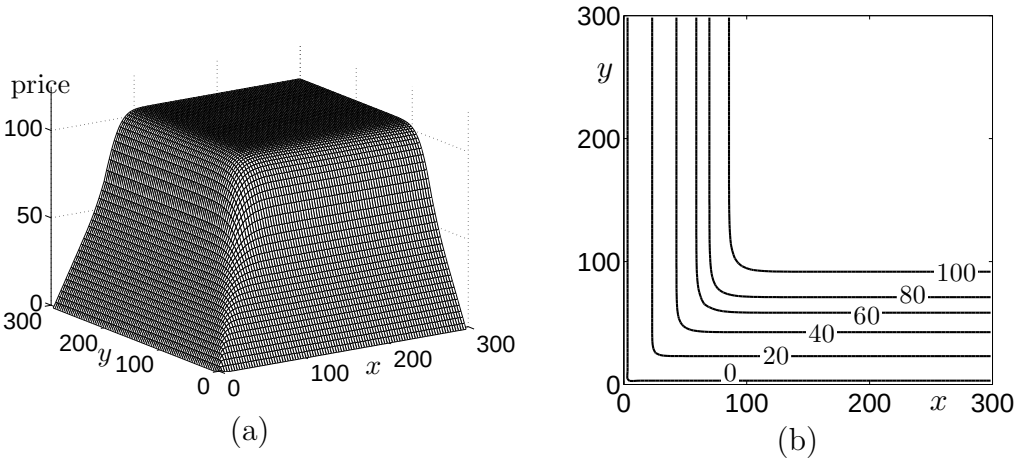
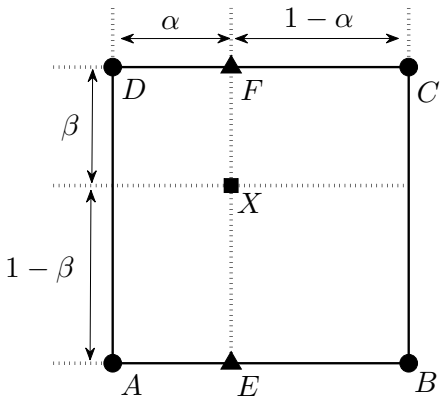


FIGURE 6.6. (a) The weighted average value $0.1u + 0.9v$ at $T = 1$. (b) The contour lines of the weighted average values.

Usually, the position of current underlying assets does not coincide with the numerical grid points. Therefore, we needed to use an interpolation method. As shown in Fig. 6.7, we obtained the numerical values at the specific point X by using the bilinear interpolation.



$$
\begin{aligned}
E &= (1-\alpha)A + \alpha B \\
F &= (1-\alpha)D + \alpha C \\
\therefore X &= (1-\beta)F + \beta E
\end{aligned}
$$

FIGURE 6.7. A diagram of the bilinear interpolation: the specific value X is obtained from the numerical solutions A, B, C, and D near the specific point X by the bilinear interpolation.

Table 6.3 shows the results for two-asset step-down ELS obtained using the OSM at the point $(100, 100)$ with different meshes and time steps.

Mesh	N_t	$v(100, 100)$	$u(100, 100)$	Weighted average $0.1u + 0.9v$
300×300	365	103.041093	101.306561	102.867640
600×600	730	103.028876	101.359551	102.861944
1200×1200	1460	103.007394	101.369623	102.843617
2400×2400	2920	102.987068	101.361671	102.824528

TABLE 6.3. Two-asset step-down ELS prices u, v, and the weighted average value $0.1u + 0.9v$ obtained using the OSM at the point $(100, 100)$ with different meshes and time steps.

Fig. 6.8 shows the two-asset step-down ELS price at position $(x, y) = (100, 100)$ obtained using the OSM and the MC simulation. The solid line is the result obtained using the OSM with a 2400×2400 mesh. The symbol lines are the results from three trial MC simulations with an increasing number of samples. Generally, MC simulations in computational finance are easy to apply than the FDM. Because results obtained using the MC simulation are affected by the distribution of random numbers, the accuracy of MC simulation can be guaranteed through many trials.

6.5. Conclusions

In this chapter, we presented a numerical algorithm for the two-asset step-down ELS option by using the OSM. We modeled the value of ELS option by using a modified Black–Scholes partial differential equation. A finite difference method was used to discretize the governing equation, and the OSM was applied to solve the resulting discrete equations. We provided a detailed numerical algorithm and computational results demonstrating the performance of the method for two underlying asset option pricing problems such as cash-or-nothing and step-down

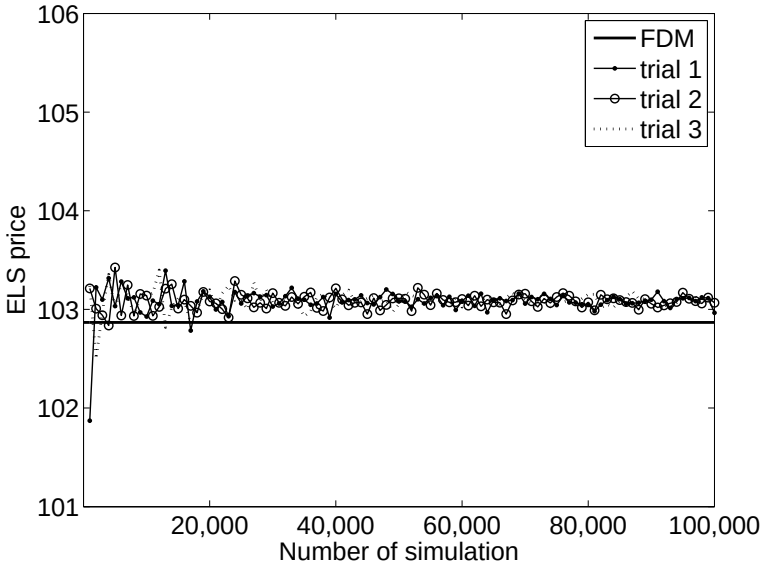


FIGURE 6.8. Two-asset step-down ELS price obtained using the OSM and the Monte-Carlo simulation versus the number of simulations.

ELS. In addition, we applied a weighted average value with a probability obtained using the MC simulation to obtain the option value of two-asset step-down ELS.

Chapter 7

An adaptive multigrid technique for option pricing under the Black–Scholes model

In this chapter, we consider the adaptive multigrid method for solving the Black–Scholes equation as the numerical technique to improve the efficiency of the option pricing. Adaptive meshing is generally regarded as an indispensable tool because of reduction of the computational costs which are needed to obtain finite difference solutions. Therefore, in this chapter, the Black–Scholes equation is discretized using a Crank–Nicolson scheme on block-structured adaptively refined rectangular meshes. And the resulting discrete of equations is solved by a fast solver such as an multigrid method. Numerical simulations are implemented to confirm the efficiency of the adaptive multigrid technique. In particular, through the comparison of computational results on adaptively refined mesh and uniform mesh, we show that adaptively refined mesh solver is superior to a standard method.

7.1. Introduction

To obtain an approximation of the option value, option pricing problems have been solved by the simulation-based methods [21, 22, 48], the lattice methods [21, 23, 30] and by the finite difference method [13, 18, 24, 30, 42, 66, 68, 69, 75, 77], delaying the study and the application of others numerical methods like the finite elements method [2, 26, 70, 84, 85, 86] and finite volume method [29, 88],

which are widely documented and used in others fields of science and engineering for decades.

In this chapter, we propose on efficient and accurate method based on multigrid method and adaptive grid refinement method as fast numerical solver.

Among the popular method in recent years, multigrid methods [31, 72, 76] are widely used for the numerical solution of PDEs. In reference [39], authors evaluated the option price by using multigrid method under Black–Scholes.

Also, adaptive time-stepping has been proposed by some researchers [89], but few researchers use space-adaptive methods. Some examples, though, can be found in Achdou and Pironneau [1] and Pironneau and Hecht in [58] who use a space-adaptive finite element method for discretization of the Black–Scholes PDE. In [56], an adaptive finite difference method is developed with full control of the local discretization error which is shown to be very efficient.

An *adaptive mesh refinement* (AMR) method is very useful to combine the two goals of good accuracy and efficiency. In many science and engineering areas, such as fluid mechanics [3], electromagnetics [62], and materials science [79], an adaptive finite difference method has been very successful.

The purpose of our work is to propose an efficient adaptive FDM to solve the Black–Scholes PDEs. We computationally show that applying an adaptive method to this problem is very efficient compared to standard FDM. We use the Crank–Nicolson method for the discretization. Other key components of the algorithm are the use of dynamic, block-structured Cartesian mesh refinement (see e.g., [8, 9]) and the use of an adaptive multigrid method [72] to solve the equations at an implicit time level. Locally refined block-structured Cartesian meshes strike a balance between grid structure and efficiency. And they are very

natural to use together with multilevel multigrid methods. We note that other multilevel multigrid algorithms have been developed as part of the CHOMBO [5] software packages. Here, we follow the framework of a block-structured multilevel adaptive technique (MLAT) developed by Brandt [11].

7.2. Discretization with finite differences

Now, let us first discretize the given computational domain $\Omega = (0, L)\times(0, M)$ as a uniform grid with a space step $h = L/N_x = M/N_y$ and a time step $\Delta t = T/N_\tau$. Here, N_x, N_y, and N_τ are the number of space and time steps, respectively.

Let us denote the numerical approximation of the solution by

$$u_{ij}^n = u(x_i, y_j, t^n) = u\left((i-0.5)h, (j-0.5)h, n\Delta t\right),$$

where $i = 1, \ldots, N_x$, $j = 1, \ldots, N_y$ and $n = 1, \ldots, N_\tau$.

By applying the Crank–Nicolson scheme to Eq. (1.3), which has an accuracy $\mathcal{O}(\Delta\tau^2 + h^2)$, we have

$$\frac{u_{ij}^{n+1} - u_{ij}^n}{\Delta\tau} = \frac{1}{2}\left(\mathcal{L}u_{ij}^{n+1} + \mathcal{L}u_{ij}^n\right),$$

where the discrete difference operator $\mathcal{L}$ is defined by

$$\begin{aligned}\mathcal{L}u_{ij}^n = {} & \frac{(\sigma_1 x_i)^2}{2}\frac{u_{i-1,j}^n - 2u_{ij}^n + u_{i+1,j}^n}{h^2} + \frac{(\sigma_2 y_j)^2}{2}\frac{u_{i,j-1}^n - 2u_{ij}^n + u_{i,j+1}^n}{h^2} \\ & + \sigma_1\sigma_2\rho x_i y_j \frac{u_{i+1,j+1}^n + u_{i-1,j-1}^n - u_{i-1,j+1}^n - u_{i+1,j-1}^n}{4h^2} \\ & + r x_i \frac{u_{i+1,j}^n - u_{i-1,j}^n}{2h} + r y_j \frac{u_{i,j+1}^n - u_{i,j-1}^n}{2h} - r u_{ij}^n.\end{aligned} \tag{7.1}$$

7.3. Numerical method

First, we rewrite the Eq. (7.1) by

$$N(u_{ij}^{n+1}) = \phi_{ij}^n, \tag{7.2}$$

where

$$N(u_{ij}^{n+1}) = u_{ij}^{n+1} - \frac{\Delta t}{2}\mathcal{L}u_{ij}^{n+1} \text{ and } \phi_{ij}^{n} = u_{ij}^{n} + \frac{\Delta t}{2}\mathcal{L}u_{ij}^{n}.$$

7.3.1. Dynamic adaptive mesh refinement method. In this section, we present an adaptive hierarchy of nested rectangular grids [3]. Both the initial creation of the grid hierarchy and the subsequent regriding operations in which the grids are dynamically changed to reflect changing solution conditions use the same procedure to create new grids [61]. Cells requiring additional refinement are identified and tagged using user-supplied criteria. The tagged cells are grouped into rectangular patches using the clustering algorithm given in Berger and Rigoutsos [10]. These rectangular patches are refined to form the grids at the next level. The process is repeated until a specified maximum level is reached. We consider a hierarchy of grids

$$\Omega_0, \ldots, \Omega_l, \Omega_{l+1}, \ldots, \Omega_{l+l^*},$$

where $\Omega_0, \ldots, \Omega_l$ are global and $\Omega_{l+1}, \ldots, \Omega_{l+l^*}$ are local grids. A typical hierarchy of grids for the solution of the problem is shown in Fig. 7.1.

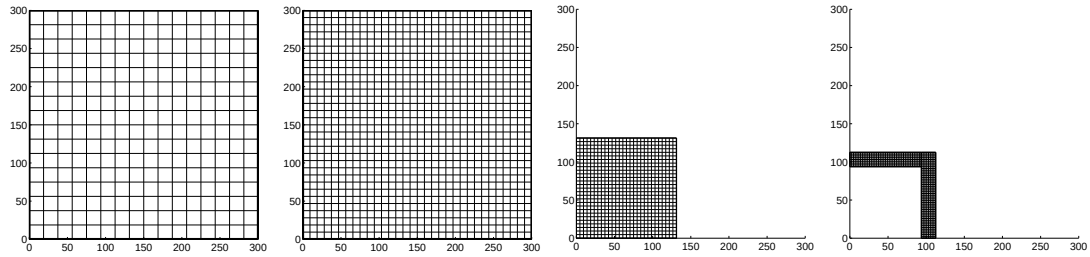


FIGURE 7.1. Hierarchy of grids. $l = 1$ and $l^* = 2$.

In this case, Ω_0 and Ω_1 are global grids ($l = 1$) and the refined grids Ω_{l+1}, Ω_{l+2} ($l^* = 2$) cover increasingly smaller subdomains as indicated in Fig. 7.2. For

instance, we can apply the refined local grids near the strike price since the values of options are not smooth near the strike price. We note that the grid refinement is automatically done by user-specified criteria.

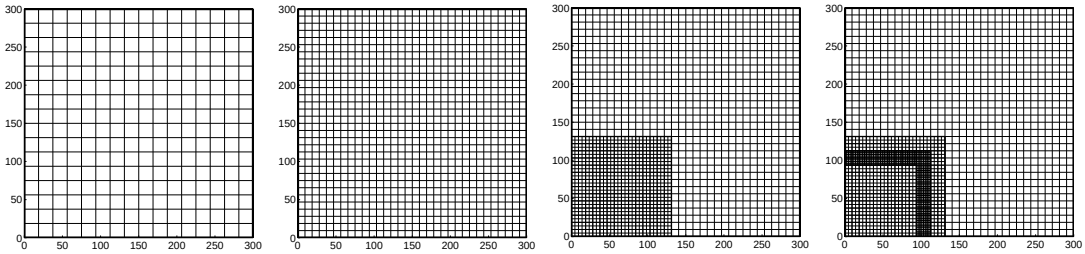


FIGURE 7.2. Composite grids corresponding to the hierarchy of grids in Fig. 7.1. $l = 1$ and $l^* = 2$.

In addition to the global and the local grids, we consider their "composition". The corresponding sequence of *composite grids* (see Fig. 7.2) is defined by

$$\hat{\Omega}_k := \Omega_k \ (k = 0, \ldots, l) \text{ and } \hat{\Omega}_{l+k} := \Omega_l \cup \bigcup_{j=1}^{k} \Omega_{l+j} \ (k = 1, \ldots, l^*).$$

We use the original multi-level adaptive technique (MLAT) proposed by Brandt [7]. We now describe an adaptive multigrid cycle. Let us use the operator in Eq. (7.2) N_k $(k = 0, 1, \ldots, l, l+1, \ldots, l+l^*)$ and the restriction and interpolation operators between Ω_k and Ω_{k-1}, I_k^{k-1}, I_{k-1}^k $(k = 1, 2, \ldots, l, l+1, \ldots, l+l^*)$ respectively. Let us assume the parameter γ (the number of smoothing iterations), and starting on the finest grid $k = l + l^*$, the calculation of a new iterate u_k^{m+1} from a given approximation u_k^m proceeds:

The details of overall steps are given in **Algorithm 1**.

Algorithm 1 Adaptive cycle

$u_k^{m+1} = adapcyc(k, u_k^m, u_{k-1}^m, N_k, \phi_k, \gamma)$**:**

1: Presmoothing

- Compute $\bar{u}_k^m$ by applying γ smoothing steps, Eq. (7.5), to u_k^m on Ω_k.

2: Coarse-grid correction

- Compute

$$\bar{u}_{k-1}^m = \begin{cases} I_k^{k-1}\bar{u}_k^m & \text{on } \Omega_{k-1} \cap \Omega_k \\ \bar{u}_{k-1}^m & \text{on } \Omega_{k-1} - \Omega_k \end{cases}$$

- Compute

$$\bar{u}_{k-1}^m = \begin{cases} I_k^{k-1}\bar{u}_k^m & \text{on } \Omega_{k-1} \cap \Omega_k \\ \bar{u}_{k-1}^m & \text{on } \Omega_{k-1} - \Omega_k \end{cases}$$

- Compute the right-hand side

$$\phi_{k-1}^n = \begin{cases} I_k^{k-1}(\phi_k^n - N_k(\bar{u}_k^m)) + N_{k-1}I_k^{k-1}\bar{u}_k^m & \text{on } \Omega_{k-1} \cap \Omega_k \\ \phi_{k-1}^n & \text{on } \Omega_{k-1} - \Omega_k \end{cases}$$

- Compute an approximate solution $\hat{w}_{k-1}^m$ of the coarse grid equation on Ω_{k-1}

$$N_{k-1}(w_{k-1}^m) = \phi_{k-1}^n. \tag{7.3}$$

If $k = 1$, employ smoothing steps.

If $k > 1$, solve Eq. (7.3) using $\bar{u}_{k-1}^m$ as an initial approximation.

$\hat{w}_{k-1}^m = adapcyc(k-1, \bar{u}_{k-1}^m, u_{k-2}^m, N_{k-1}, \phi_{k-1}, \gamma)$.

- Compute the correction $\hat{v}_{k-1}^m = \hat{w}_{k-1}^m - \bar{u}_{k-1}^m$, on $\Omega_{k-1} \cap \Omega_k$.
- Set the solution $u_{k-1}^{m+1} = \hat{w}_{k-1}^m$, on $\Omega_{k-1} - \Omega_k$.
- Interpolate the correction $\hat{v}_k^m = I_{k-1}^k \hat{v}_{k-1}^m$, on Ω_k.
- Compute the corrected approximation $u_k^{m,afterCGC} = \bar{u}_k^m + \hat{v}_k^m$, on Ω_k.
- Carry out a quadratic interpolation at the ghost points.

3: Postsmoothing

- Compute u_k^{m+1} by applying γ smoothing steps to $u_k^{m,afterCGC}$ on Ω_k.

Our implementation of this algorithm is constructed using the Chombo infrastructure [5], which has simplified the implementation of the locally adaptive

algorithm. To perform the nonlinear multilevel AMR solver, we use and modify the Chombo AMR elliptic solver. This solver is based on a linear multigrid algorithm.

7.3.2. Relaxation method in a multigrid cycle. Now we derive a Gauss-Seidel relaxation operator. First, we rewrite Eq. (7.2) as

$$\begin{aligned} u_{ij}^{n+1} &= \left[\phi_{ij}^{n} + \frac{\Delta t}{2}\left(\frac{(\sigma_1 x_i)^2}{2}\frac{u_{i-1,j}^{n+1}+u_{i+1,j}^{n+1}}{h^2} + \frac{(\sigma_2 y_j)^2}{2}\frac{u_{i,j-1}^{n+1}+u_{i,j+1}^{n+1}}{h^2}\right.\right. \\ &\quad +\sigma_1\sigma_2\rho x_i y_j \frac{u_{i+1,j+1}^{n+1}+u_{i-1,j-1}^{n+1}-u_{i-1,j+1}^{n+1}-u_{i+1,j-1}^{n+1}}{4h^2} \\ &\quad \left.\left. + rx_i\frac{u_{i+1,j}^{n+1}-u_{i-1,j}^{n+1}}{2h} + ry_j\frac{u_{i,j+1}^{n+1}-u_{i,j-1}^{n+1}}{2h}\right)\right] / \\ &\quad \left[1+\frac{\Delta t}{2}\left(\frac{(\sigma_1 x_i)^2+(\sigma_2 y_j)^2}{h^2}+r\right)\right]. \end{aligned} \tag{7.4}$$

Next, we replace u_{kl}^{n+1} in Eq. (7.4) with $\bar{u}_{kl}^{m}$ if $(k < i)$ or $(k = i$ and $l \leq j)$, otherwise with u_{kl}^{m}, i.e.,

$$\begin{aligned} \bar{u}_{ij}^{m} &= \left[\phi_{ij}^{n} + \frac{\Delta t}{2}\left(\frac{(\sigma_1 x_i)^2}{2}\frac{\bar{u}_{i-1,j}^{m}+u_{i+1,j}^{m}}{h^2} + \frac{(\sigma_2 y_j)^2}{2}\frac{\bar{u}_{i,j-1}^{m}+u_{i,j+1}^{m}}{h^2}\right.\right. \\ &\quad +\sigma_1\sigma_2\rho x_i y_j \frac{u_{i+1,j+1}^{m}+\bar{u}_{i-1,j-1}^{m}-\bar{u}_{i-1,j+1}^{m}-u_{i+1,j-1}^{m}}{4h^2} \\ &\quad \left.\left. + rx_i\frac{u_{i+1,j}^{m}-\bar{u}_{i-1,j}^{m}}{2h} + ry_j\frac{u_{i,j+1}^{m}-\bar{u}_{i,j-1}^{m}}{2h}\right)\right] / \\ &\quad \left[1+\frac{\Delta t}{2}\left(\frac{(\sigma_1 x_i)^2+(\sigma_2 y_j)^2}{h^2}+r\right)\right]. \end{aligned} \tag{7.5}$$

Therefore, in a multigrid cycle, one smooth relaxation operator step consists of solving Eq. (7.5) given above.

7.4. Computational results

In this section, several numerical experiments are on performance of the adaptive techniques and their benefit in finding accurate solutions efficiently. To demonstrate its effectiveness, we compare the total computation cost, i.e., the CPU times with uniform mesh results on a test problem on the computational domain $\Omega = (0, 1200) \times (0, 1200)$. The calculations have been performed on an IBM personal computer with 3.0GHz speed of 3.48GB RAM. It should be noticed that in our numerical experiments, we simply set $\Delta t = 1/1024$.

7.4.1. European call option. As the benchmark problem, we consider the European option problem. This problem is of great interest to academicians in the finance literature and often used to show the accuracy of a given numerical scheme [15, 27, 30].

The initial state is given as

$$u(x, y, 0) = \max[\max(x, y) - 100, 0].$$

For the parameters, we take $\sigma_1 = \sigma_2 = 0.5$, $\rho = 0.5$, and $r = 0.03$. We perform an adaptive mesh refinement every 5 time steps. The refinement is based on the range of values of u, i.e., we refine the grids if $0.3 < u < 10$. We compute this with a base 64^2 mesh with 3, 4, and 5 levels of refinements. To estimate the cost of the equivalent uniform-grid solution, we compute 1024 time steps on the equivalent 512^2, 1024^2, and 2048^2 meshes. In Fig. 7.3, (a) and (b) show the initial profile and the final configuration at time $\tau = 1$ on the adaptive mesh, respectively. We can observe fine meshes around the region of our interests which are strike prices. And Fig. 7.3(c) and (d) show magnified representations of (a) and (b), respectively.

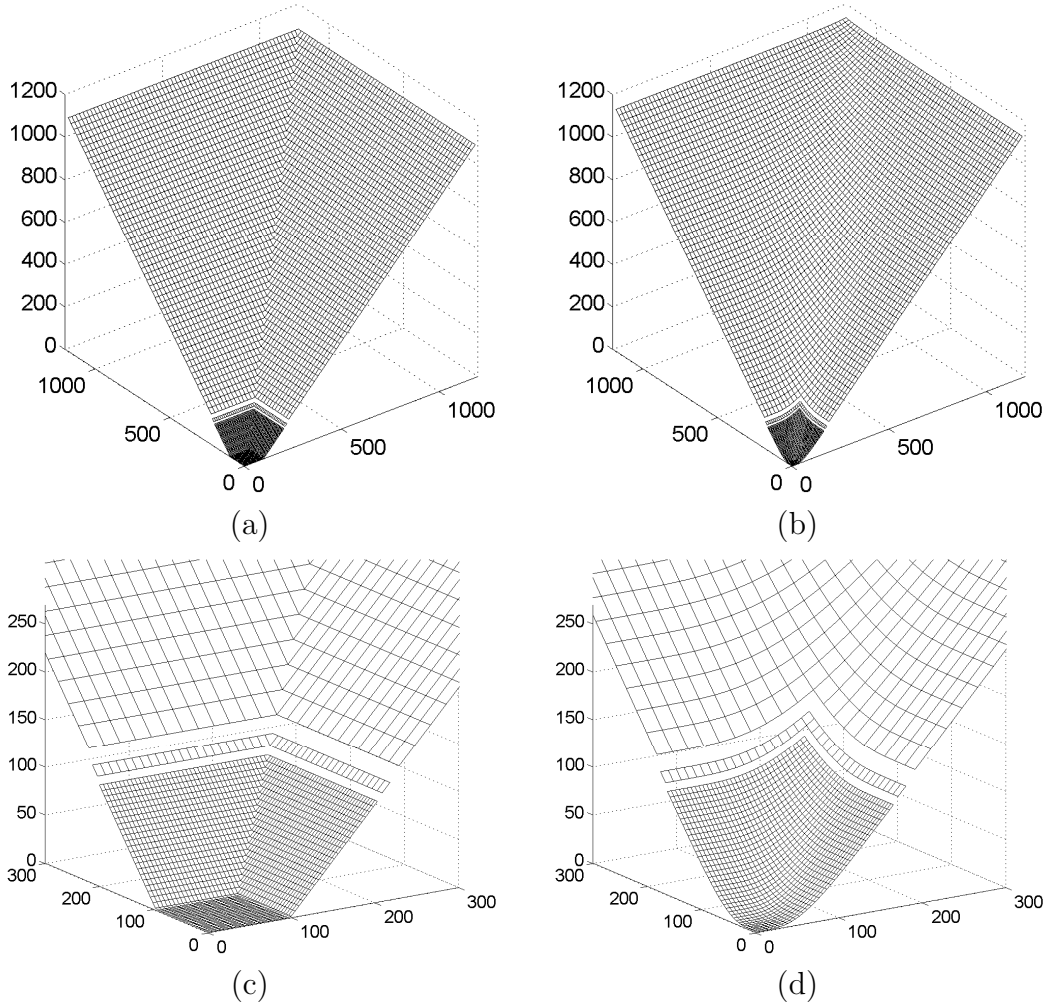


FIGURE 7.3. European call option: (a) The initial configuration at time $\tau = 0$. (b) The final configuration at time $\tau = 1$. (c) and (d) magnified representations of (a) and (b), respectively.

Next, we compare the CPU times with AMR and uniform mesh results. The computational results are shown in Table 7.1 and it is clear that AMR is efficient than the uniform mesh method. We scale CPU time with the AMR method. Here, 1 in CPU time of AMR stands for the calculation time for AMR method.

Case	Uniform mesh 512^2	AMR with base mesh size, 64^2 3 levels, effective mesh size 512^2
CPU time	68.3	1
Case	Uniform mesh 1024^2	AMR with base mesh size, 64^2 4 levels, effective mesh size 1024^2
CPU time	169.7	1
Case	Uniform mesh 2048^2	AMR with base mesh size, 64^2 5 levels, effective mesh size 2048^2
CPU time	285.3	1

TABLE 7.1. CPU time comparison between uniform mesh and AMR of European call option.

7.4.2. Cash-or-nothing option. Next, we perform the comparison with a cash-or-nothing option. The initial state is given as

$$u(x,0) = \begin{cases} Cash & \text{if } x \geq K \text{ and } y \geq K \\ 0 & \text{otherwise.} \end{cases}$$

Here, we simply set $Cash = 1$ and $K = 100$. And the other parameters and computational conditions are chosen as the same in the numerical experiment of European call option. In figure 7.4, (a) and (b) show the initial profile and the final configuration at time $\tau = 1$ on the adaptive mesh, respectively. And Fig. 7.4(c) and (d) show magnified representations of (a) and (b), respectively.

Next, the CPU times with AMR and uniform mesh results are presented in Table 7.2. As can been seen, it is clear that AMR is more robustness than the uniform mesh method.

7.5. Conclusions

In this chapter, we focused on two major aspects that we encounter when applying numerical methods to option pricing problems such that grid resolutions and time steps. We proposed a adaptive mesh refinement method to solve BS equation. We computationally showed that the proposed adaptive scheme gave

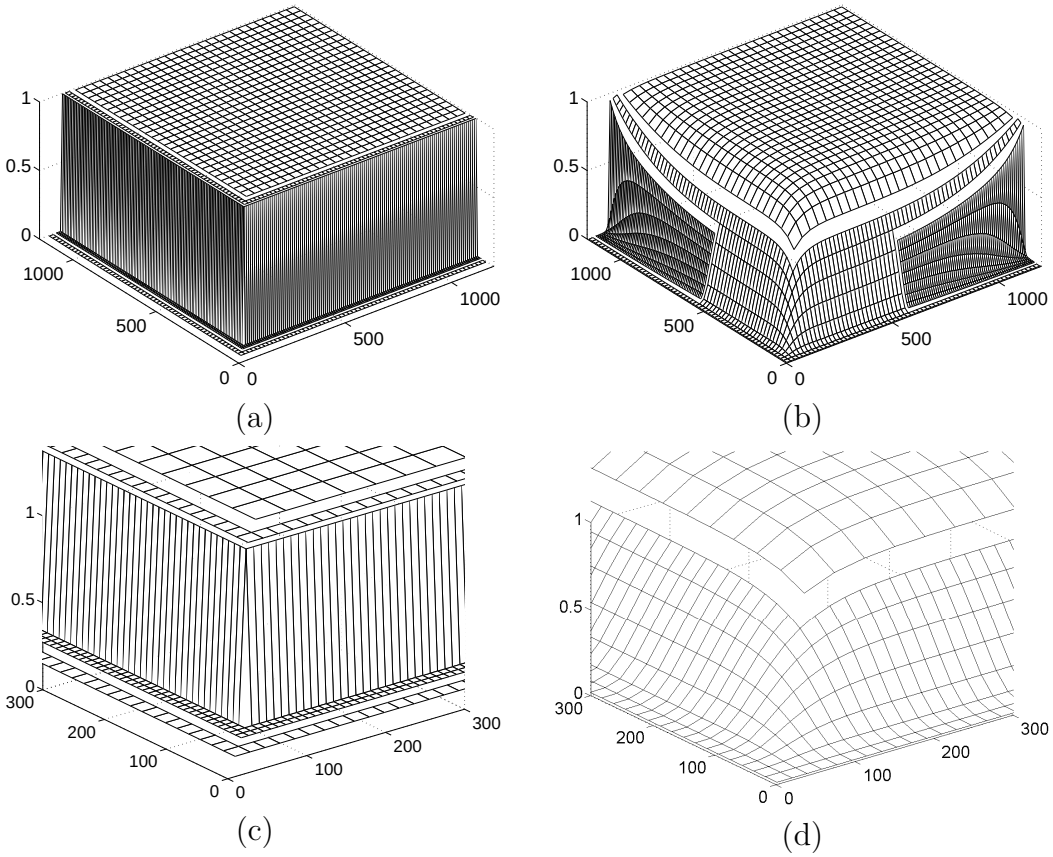


FIGURE 7.4. Cash-or-nothing option: (a) The initial configuration at time $\tau = 0$. (b) The final configuration at time $\tau = 1$. (c) and (d) magnified representations of (a) and (b), respectively.

Case	Uniform mesh 512^2	AMR with base mesh size, 64^2 3 levels, effective mesh size 512^2
CPU time	80.2	1
Case	Uniform mesh 1024^2	AMR with base mesh size, 64^2 4 levels, effective mesh size 1024^2
CPU time	167.2	1
Case	Uniform mesh 2048^2	AMR with base mesh size, 64^2 5 levels, effective mesh size 2048^2
CPU time	180.5	1

TABLE 7.2. CPU time comparison between uniform mesh and AMR of Cash-or-nothing option.

much better efficiency than the standard FDM. In particular, we showed that the use of local refinement resulted in significant savings in computational time and memory when compared to the equivalent uniform-mesh solution. Studies of these methods in higher dimensions will be the subject of future research.

Chapter 8

Conclusion

We focused on the performance of a multigrid method for option pricing problems. The numerical results showed that the total computational cost was proportional to the number of grid points. The convergence test showed that the scheme was first-order accurate since we used an implicit Euler method. In a forthcoming paper, we will investigate a switching grid method, which uses a fine mesh when the solution is not smooth and otherwise uses a coarse mesh.

And we performed a comparison study of alternating direction implicit (ADI) and operator splitting (OS) methods on multi-dimensional Black-Scholes option pricing models. ADI method has been used extensively in mathematical finance for numerically solving multi-asset option pricing problems. However, most option pricing problems have nonsmooth payoffs or discontinuous derivatives at the exercise price. ADI scheme uses source terms which include y derivatives when we solve x derivative involving equations. Then, due to the nonsmooth payoffs, source term contains abrupt changes which are not in the range of implicit discrete operator and this leads to difficulty in solving the problem. On the other hand, OS method does not contain the other variable's derivatives in the source term. We provided computational results showing the performance of the methods for two underlying asset option pricing problems. The results showed that OS method is very efficient and gives better accuracy and robustness than ADI

method in computational finance problems.

Also, The resulting linear system of BS model is solved by biconjugate gradient stabilized, operator splitting, and multigrid methods. The performance of these methods is compared for two asset option problems based on two-dimensional Black-Scholes equations. Bi-CGSTAB and multigrid solver have a good accuracy but need a lot of computing times. On the other hand, operator splitting is faster than other two methods under the same accuracy.

An accurate and efficient numerical method for the Black-Scholes equations is derived. The method uses an adaptive technique which is based on a far-field boundary position of the equation. Numerical tests were presented to demonstrate the accuracy and efficiency of the method. In particular, the computational time was reduced substantially when compared to a uniform grid.

We presented a numerical algorithm for the two-asset step-down ELS option by using the OSM.We modeled the value of ELS option by using a modified Black-Scholes partial differential equation. A finite difference method was used to discretize the governing equation, and the OSM was applied to solve the resulting discrete equations. We provided a detailed numerical algorithm and computational results demonstrating the performance of the method for two underlying asset option pricing problems such as cash-or-nothing and step-down ELS. In addition, we applied a weighted average value with a probability obtained using the MC simulation to obtain the option value of two-asset step-down ELS.

Finally, we focused on two major aspects that we encounter when applying numerical methods to option pricing problems such that grid resolutions and time steps. We proposed a adaptive mesh refinement method to solve BS equation. We computationally showed that the proposed adaptive scheme gave much better

efficiency than the standard FDM. In particular, we showed that the use of local refinement resulted in significant savings in computational time and memory when compared to the equivalent uniform-mesh solution.

Appendix: MATLAB code

1. MATLAB code for closed form of cash or nothing option

```
K=100;T=0.1;r=0.03;X1=100;X2=100;sigma1=0.5;sigma2=0.5;
rho=0.5;b1=r;b2=r;
VE=zeros(Nx,Ny);mu=[0 0];cov=[1 rho; rho 1];
for i=1:Nx
 for j=1:Ny
  y1=(log(x(i)/X1)+(b1-sigma1^2/2)*T)/(sigma1*sqrt(T));
  y2=(log(y(j)/X2)+(b2-sigma2^2/2)*T)/(sigma2*sqrt(T));
  M=mvncdf([y1 y2],mu,cov); V(i,j)=K*exp(-r*T)*M;
 end
end
[X, Y]=meshgrid(x(1:Nx),y(1:Ny)); surf(X, Y, V)
```

2. MATLAB code for closed form of max option

```
K=100;T=0.1;r=0.03;X1=100;X2=100;sigma1=0.5;sigma2=0.5;
rho=0.5;b1=r;b2=r;
VE=zeros(Nx,Ny);mu=[0 0];cov=[1 rho; rho 1];
for i=1:Nx
 for j=1:Ny
  y1=(log(x(i)/X1)+(b1-sigma1^2/2)*T)/(sigma1*sqrt(T));
  y2=(log(y(j)/X2)+(b2-sigma2^2/2)*T)/(sigma2*sqrt(T));
  M=mvncdf([y1 y2],mu,cov); V(i,j)=K*exp(-r*T)*M;
 end
end
[X, Y]=meshgrid(x(1:Nx),y(1:Ny)); surf(X, Y, V)
```

3. Operator Splitting method for BS model

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Black-Scholes 2Dim.                                                          %
%    - Operator Splitting Method                                                 %
%    - uniform mesh                                                              %
%                                                                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all; clc; close all; format compact;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%% Parameters %%%%%%%%%%%%%%%%%%%%%%%%%%%%
rho=0.5; r=0.05; sig1=0.25; sig2=0.3; K=100; L=300;
lam1=0.5;lam2=0.5; % weight_factor
cash = 1.0;

pp = 1;

dt = 0.1/2^(pp-1); Nt = 10*2^(pp-1); T = dt*Nt; Nx = 128*2^(pp-1);
Ny=Nx; h = L/Nx;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
sh = 1;

%%%%%%%%%%%%% x and y (cell center included ghost points) %%%%%%%%%%%%%
x = linspace(-0.5*h, L+0.5*h, Nx+2); y = linspace(-0.5*h, L+0.5*h,
Ny+2);

%%%%%%%%%%%%%%%%%%%%%%%%%% allocate matrix %%%%%%%%%%%%%%%%%%%%%%%%%%%
u(1:Nx+2,1:Ny+2) = 0.0; u0 = u; fx(1:Nx) = 0.0; fy = fx;

%%%%%%%%%%%%%%%%%%%%%%%%%% coefficients x %%%%%%%%%%%%%%%%%%%%%%%%%%%%
 for i=2:Nx+1
     main_x(i-1) = 1/dt + (sig1*x(i))^2/(h^2) + r*x(i)/h + lam2*r;
     sub_x(i-1) = -0.5*(sig1*x(i))^2/(h^2);
     sup_x(i-1) = -0.5*(sig1*x(i))^2/(h^2) - r*x(i)/h;
 end
main_x(1) = main_x(1) + 2.0*sub_x(1); sup_x(1) = sup_x(1) -
sub_x(1); sub_x(Nx) = sub_x(Nx) - sup_x(Nx); main_x(Nx) = main_x(Nx)
+ 2.0*sup_x(Nx);

%%%%%%%%%%%%%%%%%%%%%%%%%% coefficients y %%%%%%%%%%%%%%%%%%%%%%%%%%%%
for j=2:Ny+1
     main_y(j-1) = 1/dt + (sig2*y(j))^2/(h^2) + r*y(j)/h + (1-lam2)*r;
     sub_y(j-1) = -0.5*(sig2*y(j))^2/(h^2);
     sup_y(j-1) = -0.5*(sig2*y(j))^2/(h^2) - r*y(j)/h;
end main_y(1) = main_y(1) + 2.0*sub_y(1); sup_y(1) = sup_y(1) -
sub_y(1); sub_y(Ny) = sub_y(Ny) - sup_y(Ny); main_y(Ny) = main_y(Ny)
+ 2.0*sup_y(Ny);

%%%%%%%%%%%%%%%%%%%%%%%%%% initial condition %%%%%%%%%%%%%%%%%%%%%%%%%%
%%% cash or nothing option %%%
for i = 1:Nx+2
     for j = 1:Ny+2
         if (x(i)>=K && y(j)>=K)
             u0(i,j) = cash;
```

```
         end
      end
end u=u0;
%%%%%%%%%%%%%%%%%%%%% Linear Boundary condition %%%%%%%%%%%%%%%%%%%%%
u(1,2:Ny+1)=2*u(2,2:Ny+1)-u(3,2:Ny+1);
u(Nx+2,2:Ny+1)=2*u(Nx+1,2:Ny+1)-u(Nx,2:Ny+1);
u(1:Nx+2,1)=2*u(1:Nx+2,2)-u(1:Nx+2,3);
u(1:Nx+2,Ny+2)=2*u(1:Nx+2,Ny+1)-u(1:Nx+2,Ny);

u2 = u; start_time=cputime;
%%%%%%%%%%%%%%%%%%%%%%%%%%%% time loop %%%%%%%%%%%%%%%%%%%%%%%%%%%%
for iter = 1:Nt

    %%%%%%%%%%%%%%%%%%%%%%%% x - direction %%%%%%%%%%%%%%%%%%%%%%%%
    for j=2:Ny+1
        for i=2:Nx+1
            fy(i-1) = lam1*rho*sig1*sig2*x(i)*y(j)...
                    *(u(i+1,j+1)-u(i+1,j)-u(i,j+1)+u(i,j))/(h^2)...
                    + u(i,j)/dt;
            sor1(i-1,j-1) = fy(i-1);
        end
        u2(2:Nx+1,j)=thomas(sub_x,main_x,sup_x,fy);
    end
    %%% [case1] Linear Boundary %%%%
    u2(1,2:Ny+1)=2*u2(2,2:Ny+1)-u2(3,2:Ny+1);
    u2(Nx+2,2:Ny+1)=2*u2(Nx+1,2:Ny+1)-u2(Nx,2:Ny+1);
    u2(1:Nx+2,1)=2*u2(1:Nx+2,2)-u2(1:Nx+2,3);
    u2(1:Nx+2,Ny+2)=2*u2(1:Nx+2,Ny+1)-u2(1:Nx+2,Ny);

    %%%%%%%%%%%%%%%%%%%%%%%% y - direction %%%%%%%%%%%%%%%%%%%%%%%%
    for i=2:Nx+1
        for j=2:Ny+1
            fx(j-1) = (1-lam1)*rho*sig1*sig2*x(i)*y(j)...
                *(u2(i+1,j+1)-u2(i+1,j)-u2(i,j+1)+u2(i,j))/(h^2)...
                + u2(i,j)/dt;
            sor2(i-1,j-1) = fx(j-1);
        end
        u(i,2:Ny+1)=thomas(sub_y,main_y,sup_y,fx);
    end
    %%% [case1] Linear Boundary %%%%
    u(1,2:Ny+1)=2*u(2,2:Ny+1)-u(3,2:Ny+1);
    u(Nx+2,2:Ny+1)=2*u(Nx+1,2:Ny+1)-u(Nx,2:Ny+1);
    u(1:Nx+2,1)=2*u(1:Nx+2,2)-u(1:Nx+2,3);
    u(1:Nx+2,Ny+2)=2*u(1:Nx+2,Ny+1)-u(1:Nx+2,Ny);
```

```
end
```

Bibliography

[1] Y. Achdou and O. Pironneau, *Computational methods for option pricing*, SIAM, Philadelphia, 2005.

[2] Y. Achdou and N. Tchou, Variational analysis for the Black & Scholes equation with stochastic volatility. Math. Mod. & Numer. Anal. 36, 373–395 (2002)

[3] A.S. Almgren, J.B. Bell, P. Colella, L.H. Howell, and M.L. Welcome, A conservative adaptive projection method for the variable density incompressible Navier–Stokes equations. J. Comput. Phys. 142, 1–46 (1998)

[4] P. Amstera, C. Averbuj, P. de Napoli, and M. Mariani, A parabolic problem arising in Financial Mathematics. Nonlinear Anal. Real World Appl. **11** 759-763 (2010)

[5] Applied numerical algorithms group: The chombo framework for block-structured adaptive mesh refinement. Technical report, Lawrence Berkeley National Laboratory (2005). Available online at http://seesar.lbl.gov/ANAG/chombo/

[6] F. Black and M. Sholes, *The pricing of options and corporate liabilities*, J. Political Economy **81** (1973), no. 3, 637–659.

[7] D. Bai and A. Brandt, Local mesh refinement multilevel techniques: J. Sci. Stat. Comput. 8, 109–134 (1987)

[8] M. Berger and P. Colella, Local adaptive mesh refinement for shock hydrodynamics. J. Comput. Phys. 82, 64–84 (1989)

[9] M. Berger and J. Oliger, Adaptive mesh refinement for hyperbolic partial differential equations. J. Comput. Phys. 53, 484–512 (1984)

[10] M.J. Berger and J. Rigoutsos, An algorithm for point clustering and grid generation. Courant Inst. of Math. Sci. 21, 1278–1286 (1991)

[11] A. Brandt, Multi-level adaptive solutions to boundary-value problems. Math. Comput. 31, 333–390 (1977)

[12] M. Brennan and E. Schwartz, The valuation of American put options, J. Financ. 32 (1977) 449–462.

[13] M. Brennan and E. Schwartz, Finite difference methods and jump processes arising in the pricing of contingent claims: a synthesis, J. Financ. Quant. Anal. 13 (1978) 461–474.

[14] M. Broadie and J. Detemple, Option pricing: valuation models and applications. Manage. Sci. 50, 1145–1177 (2004)

[15] G.W. Buetow and J.S. Sochacki, The trade-off between alternative finite difference techniques used to price derivative securities, Appl. Math. Comput. 115 (2000) 177–190.

[16] H.-J. Bungartz, A. Heinecke, and D. Pflüger, S. Schraufstetter, Option pricing with a direct adaptive sparse grid approach, J. Comput. Appl. Math. 236 (2012) 3741–3750.

[17] Z. Cen and A. Le, A robust and accurate finite difference method for a generalized Black–Scholes equation, J. Comput. Appl. Math. 235 (2011) 3728–3733.

[18] M. Chawla and D. Evans, Numerical volatility in option valuation from Black–Scholes equation by finite differences. Int. J. Comput. Math. 81, 1039–1041 (2004)

[19] R.C.Y. Chin, T.A. Manteuffel, and J. de Pillis, *ADI as a preconditioning for solving the convection-diffusion equation*, SIAM J. Sci. Stat. Comput. **5** (1984), no. 2, 281-299.
[20] C. Christara and D. M. Dang, Adaptive and high-order methods for valuing American options, J. Comput. Financ. 14(4) (2011) 74–113.
[21] G. Cortazar, Simulation and Numerical Methods in Real Options Valuation. Real Options and Investment Under Uncertainty, The MIT Press, Cambridge (2004)
[22] G. Cortazar, M. Gravet, and J. Urzúa, The valuation of multidimensional American real option using the LSM simulation method. Comput. Oper. Res. 35, 113–129 (2008)
[23] J.C. Cox, S.A. Ross, and M. Rubinstein, Option pricing: a simplified approach. J. Finan. Econ. 7, 229–264 (1979)
[24] D.J. Duffy, *Finite Difference Methods in Financial Engineering : a partial differential equation approach*, John Wiley and Sons, New York, 2006.
[25] A. Eckner, Computational techniques for basic affine models of portfolio credit risk, J. Comput. Financ. 13(1) (2009) 63–97.
[26] A. Ern, S. Villeneuve, and A. Zanette, Adaptive finite element methods for local volatility European option pricing. Int. J. Theoretical Appl. Finance 7, 659–684 (2002)
[27] S. Figlewski and B. Gao, The adaptive mesh model: a new approach to efficient option pricing, J. Financ. Econ. 53 (1999) 313–351.
[28] H. Foester and K. Witsch, On efficient multigrid software for elliptic problems on rectangular domains. Math. Comput. Simul. **23** 293-298 (1981)
[29] P.A. Forsyth and K.R. Vetzal, Quadratic convergence for valuing American options using a penalty method. SIAM J. Sci. Comput. 23, 2095–2122 (2002)
[30] R. Geske and K. Shastri, Valuation by approximation: a comparison of alternative option valuation techniques, J. Financ. Quant. Anal. 20 (1985) 45–71.
[31] W. Hackbusch, Multi-grid Methods and Applications. Springer-Verlag, New York (1980)
[32] W. Hackbusch, *Iterative Solution of Large Linear Systems of Equations*, Springer, New York, 1994.
[33] H. Han and X. Wu, *A fast numerical method for the Black-Scholes equation of American options*, SIAM J. Numer. Anal., **41** (2003), 2081–2095.
[34] E.G. Haug, *The Complete Guide to Option Pricing Formulas*, MaGraw-Hill, 2007.
[35] R. Heynen and H. Kat, Brick by Brick. Risk Mag. **9** 28-31 (1996)
[36] K.J. Hout and S. Foulon, ADI finite difference schemes for option pricing in the Heston model with correlation, Int. J. Numer. Anal. Model. 7 (2010) 303–320.
[37] J.C. Hull, *Options, Futures and Others*, Prentice Hall, 2003.
[38] S. Ikonen and J. Toivanen, *Operator splitting methods for American option pricing*, Applied Mathematics Letters **17** (2004), 809-814.
[39] D. Jeong, J. Kim, and I. Wee, An accurate and efficient numerical method for the Black-Scholes equations. Commun. Korean Math. Soc. 24, 617–628 (2009)
[40] Gh. Juncu and C. Popa, Preconditioning by Gram matrix approximation for diffusion-convection-reaction equations with discontinuous coefficients. Math. Comput. Simul. **60** 487-506 (2002)
[41] R. Kangro and R. Nicolaides, *Far field boundary conditions for Black-Scholes equations*, SIAM Journal on Numerical Analysis , **38** (4) (2000), 1357–1368.
[42] A.Q.M. Khaliq, D.A. Voss, and K. Kazmi, Adaptive θ-methods for pricing American options. J. Comput. Appl. Math. 222, 210–227 (2008)
[43] Y.K. Kwok, *Mathematical Models of Financial Derivatives*, Springer, 1998.
[44] B. Lapeyre, A. Sulem, and D. Talay, Understanding Numerical Analysis for Financial Models. Cambridge University Press, (2003)

[45] K.S. Lee, Y.E. Gwong, and J.H. Shin, *Deravatives modeling I: Using MATLAB®*, A-Jin, Seoul, 2008.
[46] W. Liao and J. Zhu, An accurate and efficient numerical method for solving Black-Scholes equation in option pricing. Int. J. Math. Oper. Res. **1** 191-210 (2009)
[47] G. Linde, J. Persson, and L. von Sydow, High-order adaptive space-discretizations for the Black–Scholes equation, Int. J. Comput. Math. 86 (2006)
[48] F. Longstaff and E. Schwartz, Valuing American options by simulation: a simple least-squares approach. Rev. Financ. Stud. 14, 113–147 (2001)
[49] P. Lötstedt, J. Persson, L. von Sydow, and J. Tysk, Space-time adaptive finite difference method for European multi-asset options, Comput. Math. Appl. 53 (2007) 1159–1180.
[50] P. Lötstedt, S. Söderberg, A. Ramage, and L. Hemmingsson-Frändén, Implicit solution of hyperbolic equations with space-time adaptivity, BIT. 42 (2002) 128–153.
[51] K. Maekawa, S. Lee, T. Morimoto, and K. Kawai, Jump diffusion model with application to the Japanese stock market. Math. Comput. Simul. **78** 223-236 (2008)
[52] O. Marom and E. Momoniat, A comparison of numerical solutions of fractional diffusion models in finance. Nonlinear Anal. Real World Appl. **10** 3435-3442 (2009)
[53] R.C. Merton, *Theory of rational option pricing*, Bell J. Econ. Manag. Sci. **4** (1973), no. 1, 141–183.
[54] G.N. Milstein and M.V. Tretyakov, Numerical analysis of Monte Carlo evaluation of Greeks by finite differences, J. Comput. Financ. 8(3) (2005) 1–34.
[55] C.W. Oosterlee, *On multigrid for linear complementarity problems with application to American-style options*, Electronic Transactions on Numerical Analysis, **15** (2003), 165–185.
[56] J. Persson and L. von Sydow, *Pricing European multi-asset options using a space-time adaptive FD-method*, Comput. Visual. Sci. **10** (2007), 173–183.
[57] J. Persson and L. von Sydow, Pricing American options using a space-time adaptive finite difference method, Math. Comput. Simulat. 80 (2010) 1922–1935.
[58] O. Pironneau and F. Hecht, Mesh adaption for the Black & Scholes equations, East-West J. Numer. Math. 8 (2000) 25–35.
[59] A. Ramage and L. von Sydow, A multigrid preconditioner for an adaptive Black-Scholes solver. BIT Numer. Math. **51**, 217-233 (2011)
[60] C. Reisinger and G. Wittum, *On multigrid for anisotropic equations and variational inequalities*, Comput. Visual. Sci. **7** (2004), 189–197.
[61] C.A. Rendleman, V.E. Beckner, M. Lijewski, and W. Crutchfield, and J.B. Bell, Parallelization of structured, hierarchical adaptive mesh refinement algorithms. Comput. Visual. Sci. 3, 147–157 (2000)
[62] A.M. Roma and C.J. Garcia-Cervera, Adaptive mesh refinement for micromagnetics simulations. IEEE Trans. Magn. 42, 1648–1654 (2006)
[63] Y. Saad and M. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput. **7** (1986), 856–869.
[64] Y. Saad and H.A. van der Vorst, *Iterative solution of linear systems in the 20th century*, J. Comput. Appl. Math. **123** (2000), 1-33.
[65] E. Schwartz, The valuation of warrants: Implementing a new approach, J. Financ. Econ. 4 (1977) 79–93.
[66] R. Seydel, *Tools for Computational Finance*, Springer Verlag, Berlin, 2003.
[67] H. Sun, N. Kang, J. Zhang, and E. Carlson, A fourth-order compact difference scheme on face centered cubic grids with multigrid method for solving 2D convection diffusion equation. Math. Comput. Simul. **63**, 651-661 (2003)

[68] D. Tavella and C. Randall, *Pricing Financial Instruments-The finite difference method*, John Wiley and Sons, Inc., 2000.
[69] J. Topper, *Financial Engineering with Finite Elements*, John Wiley and Sons, New York, 2005.
[70] J. Topper, Option pricing with finite elements. in: Wilmott Magazine (2005)
[71] J. Toivanen, A high-order front-tracking finite diffrence method for pricing American options under jump-diffsion models, J. Comput. Financ. 13(3) (2010) 61–79.
[72] U. Trottenberg, C. Oosterlee, and A. Schüller, *Multigrid*, Academic press, 2001.
[73] Using MATLAB, http://www.mathworks.com/, The MathWorks Inc., Natick, MA., 1998.
[74] H.A. Van Der Vorst, *BI-CGSTAB: A fast and smoothly converging variant of BI-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput. **13** (1992), no. 2, 631-644.
[75] B.A. Wade, A.Q.M. Khaliq, M. Khaliq, J. Vigo-Aguiar, and R. Deininger, On smoothing of the Crank-Nicolson scheme and higher order schemes for pricing barrier options. J. Comput. Appl. Math. 204, 144–158 (2007)
[76] P. Wesseling, An Introduction to Multigrid Methods. John Wiley and Sons, Chichester (1991)
[77] P. Wilmott, J. Dewynne, and S. Howison, *Option Pricing : mathematical models and computation*, Oxford Financia Press, Oxford, 1993.
[78] R. Windcliff, P.A. Forsyth, and R.A. Vetzal, Analysis of the stability of the linear boundary condition for the Black–Scholes equation, J. Comp. Finan. 8 (2004) 65–92.
[79] S.M. Wise, J.S. Kim, and J.S. Lowengrub, Solving the regularized, strongly anisotropic Cahn-Hilliard equation by an adaptive nonlinear multigrid method. J. Comput. Phys. 226, 414–446 (2007)
[80] A.V. Wouwer, P. Saucez, and W.E. Schiesser, Adaptive Method of Lines, Chapman & Hall, Boca Raton, 2001.
[81] Y. Xiao, P. Zhang, and S. Shu, Algebraic multigrid methods for elastic structures with highly discontinuous coefficients. Math. Comput. Simul. **76** 249-262 (2007)
[82] N.N. Yanenko, The Method of Fractional Steps, Springer-Verlag, New York, 1971.
[83] S. Zhao and G.W. Wei, Option valuation by using discrete singular convolution, Appl. Math. Comput. 167 (2005) 383–418.
[84] C. Zhang, Pricing American Options by Adaptive Finite Element Method. Mathematics Department University of Maryland (2005)
[85] Z. Zhu and N. Stokes, A finite element platform for pricing path-dependent exotic options. in: CSIRO Mathematical & Information Sciences, Australia (1999)
[86] R. Zvan, P.A. Forsyth, and K.R. Vetzal, Robust numerical methods for PDE models of Asian options, J. Comp. Finan. 1 (1998) 39–78.
[87] R. Zvan, P.A. Forsyth, and K.R. Vetzal, A General Finite Element Approach for PDE Option Pricing Models. University of Waterloo, Canada (1998)
[88] R. Zvan, P.A. Forsyth, and K.R. Vetzal, Penalty methods for American options with stochastic volatility. J. Comput. Appl. Math. 91, 199–218 (1998)
[89] R. Zvan, K. R. Vetzal, and P.A. Forsyth, *PDE methods for pricing barrier options*, Journal of Economic Dynamics and Control , **24** (2000), 1563–1590.