Numerical methods for phase-field model and computational finance

By

Darae Jeong

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

Master of Science

in

Mathematics

in the

OFFICE OF GRADUATE STUDIES

of the

KOREA UNIVERSITY

Approved:

Committee Member 1

Committee Member 2

Committee Member 3

Committee in Charge

2010

Contents

Abstract

The primary purpose of this thesis is to explore the numerical methods for four main parts: phase-field model, Landau-Lifshitz model, computational finance, and biomathematics. This dissertation consists of published and working papers which were performed as a graduate student during last two years.

(1) Phase-field model

First, we consider an unconditionally gradient stable scheme for solving the Allen-Cahn equation representing a model for anti-phase domain coarsening in a binary mixture. The continuous problem has a decreasing total energy. We show the same property for the corresponding discrete problem by using eigenvalues of the Hessian matrix of the energy functional. We also show the pointwise boundedness of the numerical solution for the Allen-Cahn equation. We describe various numerical experiments we performed to study properties of the Allen-Cahn equation.

And we review and present details of the computational scheme and computer program for the phase-field models. The scheme is unconditionally gradient stable and is solved by an efficient and accurate multigrid method. And the program, which is written in the C language, is designed to provide the researchers with an easy and detailed guidance for the multigrid method of the phase-field models. We discuss the implementation of the code through numerical experiments. Also, we consider a numerical method for a block copolymer using Cahn-Hilliard equaiton.

Lastly, we propose a computationally efficient and fast phase-field method which uses automatic switching parameter, adaptive time step, and automatic stopping of calculation for image inpainting. The algorithm is based on an energy functional. We demonstrate the performance of our new method and compare it with a previous method.

(2) Landau-Lifshitz model

We propose an accurate and efficient numerical approach, based on a finite difference method with Crank-Nicolson time stepping, for the Landau-Lifshitz equation without damping. A nonlinear multigrid method is used for handling the nonlinearities of the resulting discrete system of equations at each time step. We show numerically that the proposed scheme has a second-order convergence in space and time.

(3) 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, 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.

(4) Biomathematics

First, we present an efficient and accurate numerical method for solving a ratio dependent predator-prey model with Turing instability. The system is discretized by a finite difference method with a semi-implicit scheme which allows much larger time step sizes than a standard explicit scheme. A proof is given for the positivity and boundedness of the finite difference solutions depending only on time step sizes.

Second, we present a mathematical model to predict the growth of cells as a powerful tool in scaffold designs, depending on its own materials. Our observation focuses on further cells' migration and growth phase beyond experiment data.

Finally, we consider the Neumann initial boundary problem for chemotaxis-systems with a logarithmic chemotactic sensitivity function and a non-diffusing chemical in a smooth bounded domain $\Omega \subset \mathbb{R}^n$, $n \geq 1$. And we present an efficient numerical method.

Acknowledgments

Throughout my graduate studies, I have been supported by many people.

First, I would like to thank my professor, Junseok Kim, for his guidance and teaching as well as the opportunity to work on fun and exciting projects. And during last two years, he always seemed to have more confidence in my abilities than I had. Once again, I appreciate to all of the efforts of my professor.

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 want to thank my family. Mom and Dad, thanks for all the love and support that you have given me. I hope that I can continue to make you proud.

Chapter 1

Introduction

This thesis deals with four main parts, phase-field model, Landau-Lifshitz model, computational finance, and biomathematics.
And this dissertation consists of published and working papers which performed as graduate student during last two years.

The published papers which consisted in this thesis are as following.
Published papers

- An unconditionally stable numerical method for the Allen-Cahn equation, Jeong-Whan Choi, Hyun Geun Lee, Darae Jung, and Junseok Kim, Physica A, Vol. 388, No. 9, pp. 1591-1606, 2009.
- Fast and automatic inpainting of binary images using a phase-field model, Darae Jung, Yibao Li, Hyun Geun Lee, and Junseok Kim, J. KSIAM, Vol. 13, No. 3, pp. 225-236, 2009.
- An accurate and efficient numerical method for the Black-Scholes equations, Darae Jung, Junseok Kim, and In-Suk Wee, Commun. Korean Math. Soc. 24, No. 4, pp. 617-628, 2009.
- A Crank-Nicolson scheme for the Landau Lifshitz equation without damping, Darae Jung and Junseok Kim, J. Comput. Appl. Math., Vol. 234, pp. 613-623, 2010.
- An operator splitting method for ELS option pricing, Darae Jung, In-Suk Wee, and Junseok Kim, J. KSIAM, Vol. 14, No. 3, pp. 175-187, 2010.

The outline of this thesis is as following.
In Chapter 2, we consider an unconditionally gradient stable scheme for solving the Allen-Cahn equation representing a model for anti-phase domain coarsening in a binary mixture. The continuous problem has a decreasing total energy. We show the same property for the corresponding discrete problem by using eigenvalues of the Hessian matrix of the energy functional. We also show the pointwise boundedness of the numerical solution for the Allen-Cahn equation. We describe various numerical experiments we performed to study properties of the Allen-Cahn equation.

In Chapter 3, we review and present details of the computational scheme and computer program for the phase-field models. The scheme is unconditionally gradient stable and is solved by an efficient and accurate multigrid method. And the program, which is written in the C language, is designed to provide the researchers with an easy and detailed guidance for the multigrid method of the phase-field models. We discuss the implementation of the code through numerical experiments.

In Chapter 4, we consider a numerical method for a block copolymer.

In Chapter 5, we propose a computationally efficient and fast phase-field method which uses automatic switching parameter, adaptive time step, and automatic stopping of calculation for image inpainting. The algorithm is based on an energy functional. We demonstrate the performance of our new method and compare it with a previous method.

In Chapter 6, we propose an accurate and efficient numerical approach, based on a finite difference method with Crank-Nicolson time stepping, for the Landau-Lifshitz equation without damping. The phenomenological Landau-Lifshitz equation describes the dynamics of ferromagnetism. The Crank-Nicolson method is very popular in the numerical schemes for parabolic equations since it is second-order accurate in time. Although widely used, the method does not always produce accurate results when it is applied to the Landau-Lifshitz equation. The objective of this article is to enumerate the problems and then to propose an accurate and robust numerical solution algorithm. A discrete scheme and a numerical solution algorithm for the Landau-Lifshitz equation are described. A nonlinear multigrid method is used for handling the nonlinearities of the resulting discrete system of equations at each time step. We show numerically that the proposed scheme has a second-order convergence in space and time.

In Chapter 7, 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.

In Chapters 8 and 9, we perform comparison of numerical methods for two-dimensional Black-Scholes equations obtained from stock option pricing. 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.

In Chapter 10, we presente 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 Chapter 11, we present an accurate and efficient numerical method for the Black-Scholes equations. The method uses an adaptive grid technique which is based on a far-field boundary position of the equation. Numerical tests are presented to demonstrate the accuracy and efficiency of the method. 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.

In Chapter 12, we present an efficient and accurate numerical method for solving a ratio dependent predator-prey model with Turing instability. The system is discretized by a finite difference method with a semi-implicit scheme which allows much larger time step sizes than a standard explicit scheme. A proof is given for the positivity and boundedness of the finite difference solutions depending only on time step sizes. Finally, we perform numerical experiments demonstrating the robustness and accuracy of the numerical solution the Turing instability. Also, we show the numerical non-constant stationary solutions with amplitudes.

In Chapter 13, we present a mathematical model to predict the growth of cells as a powerful tool in scaffold designs, depending on its own materials. The improved understanding derived from this mathematical model and its numerical analysis benefits in the fabrication of three-dimensional scaffolds that can support more confirmation of the growth of cells. Our observation focuses on further cells' migration and growth phase beyond experiment data.

Next, in Chapter 14, we consider the Neumann initial boundary problem for chemotaxis-systems with a logarithmic chemotactic sensitivity function and a non-diffusing chemical in a smooth bounded domain $\Omega \subset \mathbb{R}^n$, $n \geq 1$. And we present an efficient numerical method.

Finally, Chapter 15 summarizes the results and gives some recommendations for future work.

Chapter 2

An unconditionally gradient stable numerical method for solving the Allen-Cahn equation

We consider an unconditionally gradient stable scheme for solving the Allen-Cahn equation representing a model for anti-phase domain coarsening in a binary mixture. The continuous problem has a decreasing total energy. We show the same property for the corresponding discrete problem by using eigenvalues of the Hessian matrix of the energy functional. We also show the pointwise boundedness of the numerical solution for the Allen-Cahn equation. We describe various numerical experiments we performed to study properties of the Allen-Cahn equation.

2.1. Introduction

The Allen-Cahn (AC) equation [1] was originally introduced as a phenomenological model for anti-phase domain coarsening in a binary alloy. It has been applied to a wide range of problems such as phase transitions [10], image analysis [3, 7], the motion by mean curvature flows [8], and crystal growth [14]. An efficient and accurate numerical solution of this equation is needed to understand its dynamics. We consider an unconditionally gradient stable algorithm for the AC equation:

$$\frac{\partial c(\mathbf{x},t)}{\partial t} = -M(F'(c(\mathbf{x},t)) - \epsilon^2 \Delta c(\mathbf{x},t)),\ \mathbf{x} \in \Omega,\ 0 < t \le T, \tag{2.1}$$

where $\Omega \subset \mathbf{R}^d$ $(d = 1, 2, 3)$ is a domain. The quantity $c(\mathbf{x}, t)$ is defined to be the difference between the concentrations of the two mixtures' components. The coefficient M is a constant mobility. We take $M \equiv 1$ for convenience. The function $F(c)$ is the Helmholtz free-energy density for c. It has a double well form, i.e., $F(c) = 0.25(c^2 - 1)^2$ as in Ref. [74].

The small constant ϵ is the gradient energy coefficient related to the interfacial energy. The boundary condition is

$$\frac{\partial c}{\partial \mathbf{n}} = 0 \text{ on } \partial\Omega, \tag{2.2}$$

where $\frac{\partial}{\partial \mathbf{n}}$ denotes the normal derivative on $\partial\Omega$. The physical meaning of the condition is that the total free energy of the mixture decreases in time. The AC equation arises from the Ginzburg-Landau free energy,

$$\mathcal{E}(c) := \int_\Omega \left(F(c) + \frac{\epsilon^2}{2} |\nabla c|^2 \right) d\mathbf{x}. \tag{2.3}$$

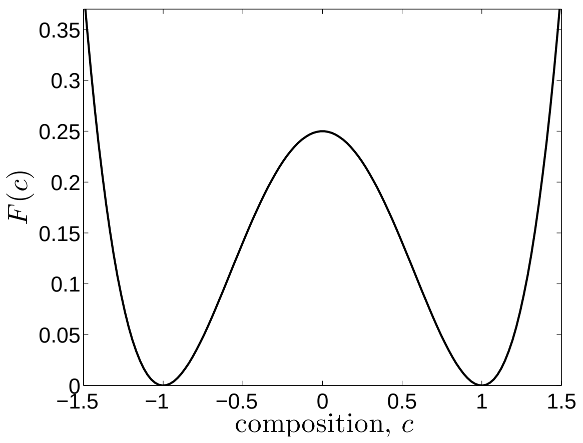


FIGURE 2.1. A double well potential, $F(c) = 0.25(c^2 - 1)^2$.

The AC equation is the L^2-gradient flow of the total free energy $\mathcal{E}(c)$. We differentiate the energy $\mathcal{E}(c)$ to get

$$\begin{aligned}\frac{d}{dt}\mathcal{E}(c) &= \int_\Omega (F'(c)c_t + \epsilon^2 \nabla c \cdot \nabla c_t) d\mathbf{x} \\ &= \int_\Omega (F'(c) - \epsilon^2 \Delta c) c_t d\mathbf{x} = -\int_\Omega (c_t)^2 d\mathbf{x} \leq 0, \end{aligned} \tag{2.4}$$

where we have used an integration by parts and the boundary condition (2.2). Therefore, the total energy is non-increasing in time; that is, the total energy is a Lyapunov functional for solutions of the AC equation. Numerical simulations of the AC equation, using explicit methods, impose severe time-step restrictions requiring the use of implicit type methods [52, 53, 11]. Ideally, we would like to use a stable integration algorithm allowing accuracy requirements rather than stability limitations to determine the integration step size. We use an unconditionally gradient stable scheme to solve the resulting discrete equations accurately and efficiently.

Eyre introduced a concept, "an unconditionally gradient stable scheme" in Refs. [52, 53]. In Eyre's papers [52, 53], he provided the idea and the theory of a scheme, but a little vaguely. We, here, clarify the proof of the scheme. We emphasize that, while the methods allow us to take arbitrarily large time steps, the accuracy of the numerical solution depends on choosing a small enough time step to resolve the dynamics [53].

This chapter is organized as follows. In Section 2.2, we briefly review a derivation of the AC equation, based on gradient dynamics, with a physically motivated functional. In Section 2.3, we describe the unconditionally gradient stable discrete scheme and its properties such as the total energy decrease and the boundedness of the numerical solution. We present the numerical results in Section 2.4. In Section 2.5, we conclude.

2.2. The Allen-Cahn equation

In this section, we review a derivation of the AC equation as a gradient flow [6, 9]. It is natural to seek a law of evolution in the form

$$\frac{\partial c}{\partial t} = -\text{grad}\mathcal{E}(c). \tag{2.5}$$

The symbol "grad" here denotes the gradient on the manifold in $L^2(\Omega)$ space. Let the domain of definition for the functional $\mathcal{E}$ be $\mathcal{D} = \{c \in H^2(\Omega) | \frac{\partial c}{\partial \mathbf{n}} = 0 \text{ on } \partial\Omega\}$. Let $c,\ v \in \mathcal{D}$. Then, we have

$$\begin{aligned}(\text{grad}\mathcal{E}(c), v)_{L^2} &= \frac{d}{d\theta}\mathcal{E}(c+\theta v)\big|_{\theta=0} = \lim_{\theta\to 0}\frac{1}{\theta}\big(\mathcal{E}(c+\theta v) - \mathcal{E}(c)\big)\\ &= \int_\Omega \big(F'(c) - \epsilon^2\Delta c\big) v d\mathbf{x} = (F'(c) - \epsilon^2\Delta c, v)_{L^2},\end{aligned}$$

where we have used an integration by parts and the boundary condition (2.2). We identify $\text{grad}\mathcal{E}(c) \equiv F'(c) - \epsilon^2\Delta c$, then Eq. (2.5) becomes the AC equation [9].

2.3. Numerical analysis

In this section, we present an unconditionally gradient stable scheme for time discretization of the AC equation. In addition, we prove a discrete version of the total free energy dissipation for any time step size, which immediately implies the stability of the numerical solution. For simplicity of exposition, we shall discretize the AC equation in one-dimensional space, i.e., $\Omega = (a, b)$. Two and three-dimensional discretizations are defined analogously.

Let N be a positive even integer, $h = (b-a)/N$ be the uniform mesh size, and $\Omega_h = \{x_i = (i-0.5)h,\ 1 \le i \le N\}$ be the set of cell-centers. Let c_i^n be approximations of $c(x_i, n\Delta t)$, where $\Delta t = T/N_t$ is the time step, T is the final time, N_t is the total number of time steps, and $\mathbf{c}^n = (c_1^n, c_2^n, \cdots, c_N^n)$. We first implement the zero Neumann boundary condition, Eq. (2.2), by requiring that for each n,

$$\nabla_h c_{\frac{1}{2}}^n = \nabla_h c_{N+\frac{1}{2}}^n = 0, \tag{2.6}$$

where the discrete differentiation operator is $\nabla_h c_{i+\frac{1}{2}}^n = (c_{i+1}^n - c_i^n)/h$. We then define a discrete Laplacian by $\Delta_h c_i = (\nabla_h c_{i+\frac{1}{2}} - \nabla_h c_{i-\frac{1}{2}})/h$ and a discrete l_2 inner product by $\langle \mathbf{c}, \mathbf{d}\rangle_h = h\sum_{i=1}^N c_i d_i$. We also define the discrete norms as $||\mathbf{c}||_h^2 = \langle \mathbf{c}, \mathbf{c}\rangle_h$ and $\|\mathbf{c}\|_\infty = \max\limits_{1\le i\le N} |c_i|$. For dissipative dynamics such as the AC equation, a discrete time stepping algorithm is defined to be *unconditionally gradient stable* if the discrete total free energy is nonincreasing for *any* size of a time step Δt.

Eyre's theorem [53] shows that an unconditionally gradient stable algorithm results for the AC equation if we can split the free energy appropriately into contractive and expansive parts,

$$\begin{aligned}\mathcal{E}(c) &= \int_a^b \left[F(c) + \frac{\epsilon^2}{2}c_x^2\right] dx \qquad (2.7)\\ &= \int_a^b \left[\frac{c^4+1}{4} + \frac{\epsilon^2}{2}c_x^2\right] dx - \int_a^b \frac{c^2}{2} dx = \mathcal{E}_c(c) - \mathcal{E}_e(c)\end{aligned}$$

and then treat the contractive part $\mathcal{E}_c(c)$ implicitly and the expansive part $-\mathcal{E}_e(c)$ explicitly. We use the nonlinearly stabilized splitting scheme [53] that involves a semi-implicit time and centered difference space discretizations of Eq. (2.1):

$$\frac{c_i^{n+1}-c_i^n}{\Delta t} = -(c_i^{n+1})^3 + c_i^n + \epsilon^2 \Delta_h c_i^{n+1} \text{ for } i=1,\cdots,N. \tag{2.8}$$

The boundary condition is $c_0 = c_1$ and $c_{N+1} = c_N$. This splitting is similar to the one employed in [13] for the time-dependent Ginzberg-Landau (TDGL) equation, which treats the TDGL equation as a heat equation with a nonlinear source term resulting in an adaptive mesh refinement scheme which is second-order in space and time.

2.3.1. The unconditionally gradient stable scheme. The main purpose of this section is to show that the scheme in Eq. (2.8) inherits characteristic properties such as a decrease in the total energy corresponding to Eq. (2.4). To show the decrease in the discrete total energy, first, we define a discrete Lyapunov functional,

$$\mathcal{E}^h(\mathbf{c}^n) = \frac{h}{4}\sum_{i=1}^{N}((c_i^n)^2-1)^2 + \frac{\epsilon^2 h}{2}\sum_{i=0}^{N}|\nabla_h c_{i+\frac{1}{2}}^n|^2 \tag{2.9}$$

for each n. It is convenient to decompose $\mathcal{E}^h(\mathbf{c}^n)$ into three parts:

$$\begin{aligned}\mathcal{E}^{(1)}(\mathbf{c}^n) &= -\frac{h}{2}\sum_{i=1}^{N}(c_i^n)^2, \qquad \mathcal{E}^{(2)}(\mathbf{c}^n) = \frac{\epsilon^2 h}{2}\sum_{i=0}^{N}\left|\nabla_h c_{i+\frac{1}{2}}^n\right|^2,\\ \mathcal{E}^{(3)}(\mathbf{c}^n) &= \frac{h}{4}\sum_{i=1}^{N}\left((c_i^n)^4+1\right).\end{aligned}$$

We define a decomposition of $\mathcal{E}^h(\mathbf{c}^n)$ as $\mathcal{E}_c^h(\mathbf{c}^n) = \mathcal{E}^{(2)}(\mathbf{c}^n) + \mathcal{E}^{(3)}(\mathbf{c}^n)$ and $\mathcal{E}_e^h(\mathbf{c}^n) = -\mathcal{E}^{(1)}(\mathbf{c}^n)$, i.e., $\mathcal{E}^h(\mathbf{c}^n) = \mathcal{E}_c^h(\mathbf{c}^n) - \mathcal{E}_e^h(\mathbf{c}^n)$. We define grad_h the variational derivative with respect c_i^n, i.e.,

$$\text{grad}_h \mathcal{E}^h(\mathbf{c}^n)_i = \frac{\delta \mathcal{E}^h(\mathbf{c}^n)}{\delta c_i^n} = (c_i^n)^3 - c_i^n - \epsilon^2 \Delta_h c_i^n. \tag{2.10}$$

We can rewrite the numerical scheme in Eq. (2.8) in terms of a gradient of the discrete total energy, i.e.,

$$\frac{c_i^{n+1}-c_i^n}{\Delta t} = -\text{grad}_h \mathcal{E}_c^h(\mathbf{c}^{n+1})_i + \text{grad}_h \mathcal{E}_e^h(\mathbf{c}^n)_i, \quad \text{for } i=1,\cdots,N. \tag{2.11}$$

The Hessian of $\mathcal{E}^{(i)}(\mathbf{c})$, denoted by $\mathbf{H}^{(i)}$, is the Jacobian of the $\text{grad}_h\mathcal{E}^{(i)}(\mathbf{c})$ and is thus given for $i = 1, 2, 3$ by

$$\left\{\mathbf{H}^{(1)}, \mathbf{H}^{(2)}, \mathbf{H}^{(3)}\right\} = \left\{J\text{grad}_h\mathcal{E}^{(1)}(\mathbf{c}), J\text{grad}_h\mathcal{E}^{(2)}(\mathbf{c}), J\text{grad}_h\mathcal{E}^{(3)}(\mathbf{c})\right\}$$

where

$$\begin{aligned}
J\mathrm{grad}_h\mathcal{E}^{(1)}(\mathbf{c}) &= -\begin{pmatrix} 1 & & & & 0 \\ & 1 & & & \\ & & \ddots & & , \\ & & & 1 & \\ 0 & & & & 1 \end{pmatrix}, \\
J\mathrm{grad}_h\mathcal{E}^{(2)}(\mathbf{c}) &= \frac{\epsilon^2}{h^2}\begin{pmatrix} 1 & -1 & & & 0 \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ 0 & & & -1 & 1 \end{pmatrix}, \\
J\mathrm{grad}_h\mathcal{E}^{(3)}(\mathbf{c}) &= 3\begin{pmatrix} c_1^2 & & & & 0 \\ & c_2^2 & & & \\ & & \ddots & & \\ & & & c_{N-1}^2 & \\ 0 & & & & c_N^2 \end{pmatrix}.
\end{aligned}$$

And we have used the boundary condition in Eq. (2.6). The eigenvalues of $\mathbf{H}^{(1)}$, $\mathbf{H}^{(2)}$, and $\mathbf{H}^{(3)}$ are

$$\lambda_k^{(1)} = -1,\ \lambda_k^{(2)} = \frac{4\epsilon^2}{h^2}\sin^2\frac{(k-1)\pi}{2N},\ \lambda_k^{(3)} = 3c_k^2 \ \text{ where } k = 1, 2, \cdots, N. \tag{2.12}$$

Note that $\lambda_k^{(1)}$ is negative and that $\lambda_k^{(2)}$ and $\lambda_k^{(3)}$ are non-negative. Let $\mathbf{v}_k$ for $k = 1, \cdots, N$ be the orthonormal eigenvector of $\mathbf{H}^{(2)}$ corresponding to the eigenvalue $\lambda_k^{(2)}$ and let $\mathbf{v}_{k,l} = \cos((k-1)\pi(2l-1)/(2N))$ for $l = 1, \cdots, N$ denote the lth component of $\mathbf{v}_k$. We can take the orthonormal eigenvector of $\mathbf{H}^{(1)}$ corresponding to the eigenvalue $\lambda_k^{(1)}$, the same as $\mathbf{v}_k$. Let $\lambda_1^e, \lambda_2^e, \ldots, \lambda_N^e$ be eigenvalues of $J(\mathrm{grad}_h\mathcal{E}_e^h) = -\mathbf{H}^{(1)}$; i.e.,

$$\lambda_k^e = -\lambda_k^{(1)} = 1,\ k = 1, 2, \cdots, N. \tag{2.13}$$

We can expand $\mathbf{c}^{n+1} - \mathbf{c}^n$ in a basis of eigenvectors $\mathbf{v}_k$ as follows.

$$\mathbf{c}^{n+1} - \mathbf{c}^n = \sum_{k=1}^{N} \alpha_k \mathbf{v}_k. \tag{2.14}$$

The decrease of the discrete energy functional is established in the following theorem: If $\mathbf{c}^{n+1}$ is the solution of Eq. (2.8) with a given $\mathbf{c}^n$, then

$$\mathcal{E}^h(\mathbf{c}^{n+1}) \le \mathcal{E}^h(\mathbf{c}^n). \tag{2.15}$$

Next, we prove Eq. (2.15). With an exact Taylor expansion of $\mathcal{E}^h(\mathbf{c}^n)$ about $\mathbf{c}^{n+1}$ up to the second order, we have

$$\begin{aligned}
\mathcal{E}^h(\mathbf{c}^n) &= \mathcal{E}^h(\mathbf{c}^{n+1}) + \langle \mathrm{grad}_h\mathcal{E}^h(\mathbf{c}^{n+1}), \mathbf{c}^n - \mathbf{c}^{n+1}\rangle_h \\
&\quad + \left\langle \frac{J(\mathrm{grad}_h\mathcal{E}^h)(\boldsymbol{\xi})}{2}(\mathbf{c}^n - \mathbf{c}^{n+1}), \mathbf{c}^n - \mathbf{c}^{n+1} \right\rangle_h,
\end{aligned}$$

where $\boldsymbol{\xi} = \theta \mathbf{c}^n + (1-\theta)\mathbf{c}^{n+1}$ and $0 \le \theta \le 1$. Rearranging the terms and using $\mathcal{E}^h = \mathcal{E}^{(1)} + \mathcal{E}^{(2)} + \mathcal{E}^{(3)}$, Eq. (2.11), Eq. (2.14), and the mean value theorem, we have

$$\begin{aligned}
&\mathcal{E}^h(\mathbf{c}^{n+1}) - \mathcal{E}^h(\mathbf{c}^n) \\
=\;& \Big\langle \mathrm{grad}_h \mathcal{E}^h(\mathbf{c}^{n+1}) - \frac{J(\mathrm{grad}_h \mathcal{E}^h)(\boldsymbol{\xi})}{2}(\mathbf{c}^{n+1} - \mathbf{c}^n), \mathbf{c}^{n+1} - \mathbf{c}^n \Big\rangle_h \\
\le\;& \langle \mathrm{grad}_h \mathcal{E}^h(\mathbf{c}^{n+1}) - \frac{1}{2}(\mathbf{H}^{(1)} + \mathbf{H}^{(2)})(\mathbf{c}^{n+1} - \mathbf{c}^n), \mathbf{c}^{n+1} - \mathbf{c}^n \rangle_h \\
=\;& \big\langle \mathrm{grad}_h \mathcal{E}^h(\mathbf{c}^{n+1}), \mathbf{c}^{n+1} - \mathbf{c}^n \big\rangle_h - \sum_{j,k=1}^{N} \Big\langle \frac{1}{2}(\mathbf{H}^{(1)} + \mathbf{H}^{(2)})\alpha_j \mathbf{v}_j,\ \alpha_k \mathbf{v}_k \Big\rangle_h \\
=\;& \Big\langle \mathrm{grad}_h \mathcal{E}_c^h(\mathbf{c}^{n+1}) - \mathrm{grad}_h \mathcal{E}_e^h(\mathbf{c}^{n+1}) - \frac{1}{\Delta t}(\mathbf{c}^{n+1} - \mathbf{c}^n) - \mathrm{grad}_h \mathcal{E}_c^h(\mathbf{c}^{n+1}) \\
&\quad + \mathrm{grad}_h \mathcal{E}_e^h(\mathbf{c}^n), \mathbf{c}^{n+1} - \mathbf{c}^n \Big\rangle_h \\
&- \sum_{j,k=1}^{N} \Big\langle \frac{1}{2}(\lambda_j^{(1)} + \lambda_j^{(2)})\alpha_j \mathbf{v}_j,\ \alpha_k \mathbf{v}_k \Big\rangle_h \\
=\;& -\Big\langle \mathrm{grad}_h \mathcal{E}_e^h(\mathbf{c}^{n+1}) - \mathrm{grad}_h \mathcal{E}_e^h(\mathbf{c}^n),\ \mathbf{c}^{n+1} - \mathbf{c}^n \Big\rangle_h - \frac{1}{\Delta t}\|\mathbf{c}^{n+1} - \mathbf{c}^n\|_h^2 \\
&- \sum_{j,k=1}^{N} \Big\langle \frac{1}{2}(\lambda_j^{(1)} + \lambda_j^{(2)})\alpha_j \mathbf{v}_j,\ \alpha_k \mathbf{v}_k \Big\rangle_h \\
=\;& -\sum_{j,k=1}^{N} \Big\langle \big[J(\mathrm{grad}_h \mathcal{E}_e^h) + \frac{1}{2}(\lambda_j^{(1)} + \lambda_j^{(2)})I\big]\alpha_j \mathbf{v}_j,\ \alpha_k \mathbf{v}_k \Big\rangle_h - \frac{1}{\Delta t}\|\mathbf{c}^{n+1} - \mathbf{c}^n\|_h^2 \\
=\;& -\sum_{k=1}^{N} \frac{1}{2}\big(\lambda_k^e + \lambda_k^{(2)}\big)\alpha_k^2 - \frac{1}{\Delta t}\|\mathbf{c}^{n+1} - \mathbf{c}^n\|_h^2 \le 0,
\end{aligned}$$

where we have used the fact that λ_k^e is positive and $\lambda_k^{(2)}$ is non-negative. Therefore, we have proven the decrease of the discrete total energy. This completes the proof. The theorem holds for any time step Δt; hence, the method is unconditionally gradient stable. It should be emphasized that while the methods will allow us to take arbitrarily large time steps, the accuracy of the numerical solution depends on choosing a small enough time step to resolve the fast-time-scale dynamics.

2.3.2. Boundedness of the numerical solution. Next, we show that the decrease of the discrete total energy functional implies the pointwise boundedness of the numerical solution for the AC equation. If $\mathbf{c}^n$ is a numerical solution for the discrete Eq. (2.8), then there exists a constant K, independent of n, such that

$$||\mathbf{c}^n||_\infty \le K. \qquad (2.16)$$

We prove Eq. (2.16) by a contradiction. Assume on the contrary that there is an integer n_K, dependent on K, such that $||\mathbf{c}^{n_K}||_\infty > K$ for all K. Then there is an index i $(1 \le i \le N)$ such that $|c_i^{n_K}| > K$. Let K be the largest solution of $hF(K) = \mathcal{E}^h(\mathbf{c}^0)$, i.e., $K = \sqrt{1 + 2\sqrt{\mathcal{E}^h(\mathbf{c}^0)/h}}$. Note that $K \geq 1$. Then, $F(c)$ is a strictly increasing function on (K, ∞) (see Fig. 2.2). Since the total energy is non-increasing, we have $\mathcal{E}^h(\mathbf{c}^0) = hF(K) < hF(|c_i^{n_K}|) \leq \mathcal{E}^h(\mathbf{c}^{n_K}) \leq \mathcal{E}^h(\mathbf{c}^0)$. This contradiction implies that Eq. (2.16) should be satisfied.

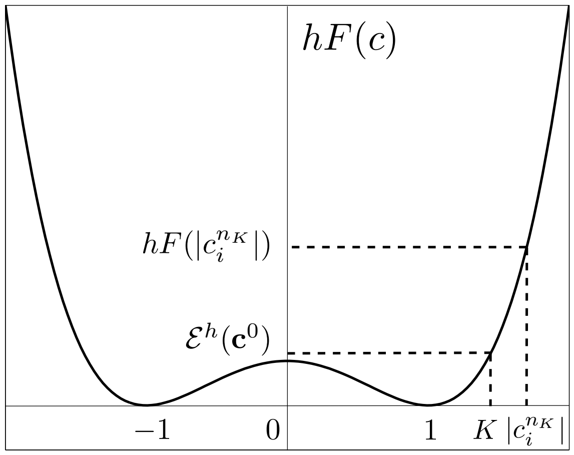


FIGURE 2.2. Graph of $hF(c)$.

2.4. Numerical experiments

In this section, we perform the following: finding relation between ϵ value and the width of transition layer, comparing the numerical equilibrium solution with the analytic equilibrium solution, investigating properties of AC equation, and checking the total energy decrease. We also implement the unconditionally gradient stable scheme in Eq. (2.8) with the recently developed adaptive mesh refinement (AMR) methodology. For detailed descriptions of the numerical method used in solving these equations with AMR, we refer to [12, 15] and the references therein. A uniform mesh solution algorithm using a nonlinear multigrid is described in subsection 2.4.1.

2.4.1. Numerical solution - a nonlinear multigrid method. In this section, we use a nonlinear Full Approximation Storage (FAS) multigrid method to solve the nonlinear discrete equation (2.8) at the implicit time level. The nonlinearity is treated using one step of Newton's iteration. A pointwise Gauss-Seidel relaxation scheme is used as the smoother in the multigrid method. See the reference text [122] for additional details and background. The algorithm of the nonlinear multigrid method for solving the discrete AC system is : First, we rewrite Eq. (2.8) as follows.

$$N(c^{n+1}) = \phi^n, \tag{2.17}$$

where $N(c^{n+1}) = c^{n+1}/\Delta t + (c^{n+1})^3 - \epsilon^2 \Delta_h c^{n+1}$ and the source term is $\phi^n = c^n/\Delta t + c^n$.

In the following description of one FAS cycle, we assume a sequence of grids Ω_k (Ω_{k-1} is coarser than Ω_k by a factor of 2). Given the number β of pre- and post-smoothing relaxation sweeps, an iteration step for the nonlinear multigrid method using the V-cycle is formally written [122]:

FAS multigrid cycle

$$c_k^{m+1} = FAScycle(k, c_k^m, N_k, \phi_k^n, \beta).$$

That is, c_k^m and c_k^{m+1} are the approximation of $c^{n+1}(x_i, y_j)$ before and after an FAScycle. Now, define the FAScycle.

Step 1) Presmoothing

$$\bar{c}_k^m = SMOOTH^{\beta}(c_k^m, N_k, \phi_k^n),$$

which means performing β smoothing steps with the initial approximation c_k^m, source term ϕ_k^n, and $SMOOTH$ relaxation operator to get the approximation $\bar{c}_k^m$. Here, we derive the smoothing operator in two dimensions. Since $(c_{ij}^{n+1})^3$ in Eq. (2.17) is nonlinear with respect to c_{ij}^{n+1}, we linearize $(c_{ij}^m)^3$ at c_{ij}^m, i.e.,

$$(c_{ij}^{n+1})^3 \approx (c_{ij}^m)^3 + 3(c_{ij}^m)^2(c_{ij}^{n+1} - c_{ij}^m).$$

After substituting this expression into (2.17), we obtain

$$\begin{aligned}\left(\frac{1}{\Delta t} + \frac{4\epsilon^2}{h^2} + 3(c_{ij}^m)^2\right) c_{ij}^{n+1} &= \phi_{ij}^n \\ +\epsilon^2 \frac{c_{i+1,j}^{n+1} + c_{i-1,j}^{n+1} + c_{i,j+1}^{n+1} + c_{i,j-1}^{n+1}}{h^2} &+ 2(c_{ij}^m)^3. \end{aligned} \tag{2.18}$$

Next, we replace $c_{\alpha\beta}^{n+1}$ in Eq. (2.18) with $\bar{c}_{\alpha\beta}^m$ if ($\alpha \le i$) or ($\alpha = i$ and $\beta \le j$); otherwise, with $c_{\alpha\beta}^m$, i.e.,

$$\begin{aligned}&\left(\frac{1}{\Delta t} + \frac{4\epsilon^2}{h^2} + 3(c_{ij}^m)^2\right) \bar{c}_{ij}^m \\ &= \phi_{ij}^n + \epsilon^2 \frac{c_{i+1,j}^m + \bar{c}_{i-1,j}^m + c_{i,j+1}^m + \bar{c}_{i,j-1}^m}{h^2} + 2(c_{ij}^m)^3. \end{aligned} \tag{2.19}$$

Step 2) Coarse grid correction

- *Compute the defect:* $\bar{d}_k^m = \phi_k^n - N_k(\bar{c}_k^m)$.
- *Restrict the defect and* $\bar{c}_k^m$*:* $\bar{d}_{k-1}^m = I_k^{k-1}(\bar{d}_k^m)$, $\bar{c}_{k-1}^m = I_k^{k-1}(\bar{c}_k^m)$.
- *Compute the right-hand side*: $\phi_{k-1}^n = \bar{d}_{k-1}^m + N_{k-1}(\bar{c}_{k-1}^m)$.
- Compute an approximate solution $\hat{c}_{k-1}^m$ of the coarse grid equation on Ω_{k-1}, i.e.

$$N_{k-1}(c_{k-1}^m) = \phi_{k-1}^n. \tag{2.20}$$

If $k = 1$, we apply the smoothing procedure in *Step1)* to obtain the approximate solution. If $k > 1$, we solve (2.20) by performing a FAS k-grid cycle using $\bar{c}_{k-1}^m$ as an initial approximation:

$$\hat{c}_{k-1}^m = \text{FAScycle}(k-1, \bar{c}_{k-1}^m, N_{k-1}, \phi_{k-1}^n, \beta).$$

- Compute the coarse grid correction (CGC): $\hat{v}_{k-1}^m = \hat{c}_{k-1}^m - \bar{c}_{k-1}^m$.
- Interpolate the correction: $\hat{v}_k^m = I_{k-1}^k \hat{v}_{k-1}^m$.

- Compute the corrected approximation on Ω_k : $c_k^{m,\text{ after } CGC} = \bar{c}_k^m + \hat{v}_k^m$.

Step 3) Postsmoothing : $c_k^{m+1} = SMOOTH^\beta(c_k^{m,\text{ after } CGC}, N_k, \phi_k^n)$.

This completes the description of a nonlinear FAScycle.

2.4.2. The relation between the ϵ value and the width of the transition layer. In our first numerical experiment, we consider the relation between the ϵ value and the width of the transition layer for the AC equation. From our choice of the total energy density Eq. (2.7) and an equilibrium profile $c(x) = \tanh(x/(\sqrt{2}\epsilon))$ on the infinite domain, the concentration field varies from -0.9 to 0.9 over a distance of about $2\sqrt{2}\epsilon\tanh^{-1}(0.9)$. Therefore, if we want this value to be about m grid points, then

$$\epsilon_m = \frac{hm}{2\sqrt{2}\tanh^{-1}(0.9)}. \tag{2.21}$$

To confirm this, we ran a simulation with the initial condition $c(x,0) = 0.01\text{rand}(x)$ on the unit domain $\Omega = (0,1)$ with $h = 1/128$, $\Delta t = 0.05$, and ϵ_4 (see the line with stars in Fig. 2.3). Here, rand(x) is a random number between -1 and 1. In Fig. 2.3, we see that the transition layer (from $c = -0.9$ to $c = 0.9$) is about 4 grid points at time $t = 10$.

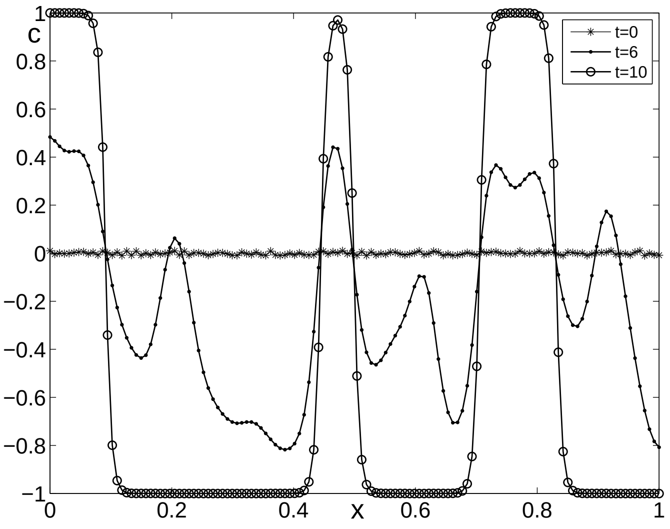


FIGURE 2.3. The evolution of an initial random distribution of concentration, $c(x,0) = 0.01\text{rand}(x)$. The concentration profile is shown at $t = 0,\ 6$, and 10.

2.4.3. An equilibrium profile. Next, we compare numerical equilibrium solutions with analytic ones. We stop the numerical computations when the discrete l_2 -norm of the difference between $(n+1)^{th}$ and n^{th} time step solutions becomes less than 10^{-10}, $\|c^{n+1} - c^n\|_h \leq 10^{-10}$ and we take c^{n+1} as a numerical equilibrium solution. The initial concentration is $c(x,0) = 0.8\tanh(x/(\sqrt{2}\epsilon_8))$ on $\Omega = (-0.5, 0.5)$. We take $h = 1/128$, ϵ_8, and $\Delta t = 0.1$. In Fig. 2.4(a), the solid, '$*$', and '$\circ$' denoted lines are the initial condition, the numerical equilibrium solution, and the analytic equilibrium solution, respectively. The numerical equilibrium profile matches well with the analytic equilibrium solution $c_{eq}^{\infty}(x) = \tanh(x/(\sqrt{2}\epsilon_8))$ on $\Omega = (-\infty, \infty)$.

In Fig. 2.4(b), numerical equilibrium solutions are shown with an initial condition $c(x,0) = 0.8\tanh(x/(\sqrt{2}\epsilon_4))$ on $\Omega = (-0.5, 0.5)$ according to various mesh sizes. Lines with the symbols, '$\diamond$', '$\circ$', '$+$', and '$\star$', denote the numerical results with 64, 128, 256, and 512 mesh sizes with ϵ_4, respectively. In this case, ϵ_4^2/h^2 is constant in Eq. (2.8); therefore, we have the same discrete equations and have almost the same values with different mesh sizes with respect to grid points from the origin.

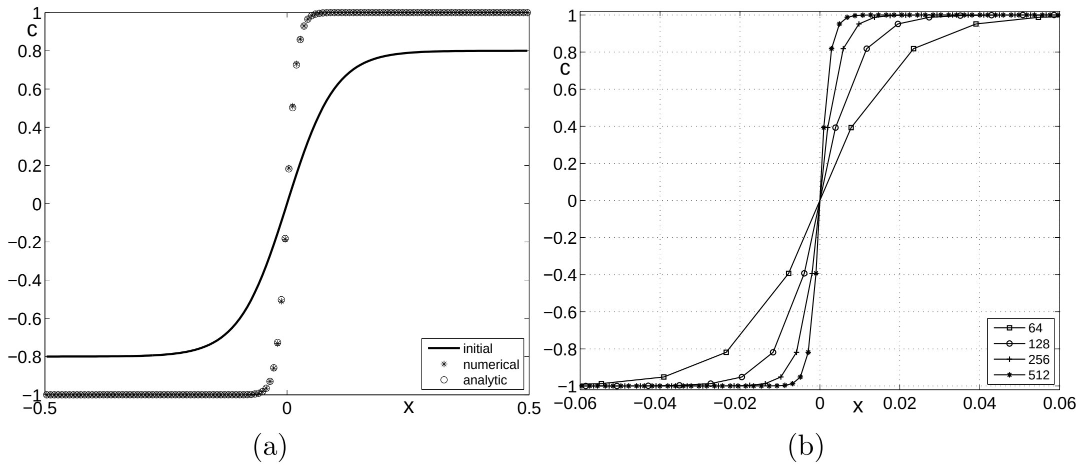


FIGURE 2.4. (a) Lines with circles and stars denote the analytic and numerical equilibrium solutions with an initial concentration $c(x,0) = 0.8\tanh(x/(\sqrt{2}\epsilon_8))$, respectively. (b) Numerical equilibrium solutions with an initial concentration $c(x,0) = 0.8\tanh(x/(\sqrt{2}\epsilon_8))$ on $\Omega = (-0.5, 0.5)$. Lines with the symbols, '$\diamond$', '$\circ$', '$+$', and '$\star$', are numerical results with 64, 128, 256, and 512 mesh sizes with ϵ_4, respectively.

2.4.4. The exact solution. The partial differential Eq. (2.1) in one dimensional space may be written as

$$c_t = -c^3 + c + \epsilon^2 \Delta c. \tag{2.22}$$

If we take the initial condition as a constant, i.e., $c(x,0) = c_0$, then, the exact solution of Eq. (2.22) is

$$c(x,t) = \frac{c_0 e^t}{\sqrt{1 + c_0^2(e^{2t} - 1)}}. \tag{2.23}$$

If $c_0 = 0.1$, then in order to find a t that satisfies $c(x,t) = 0.9$, we solve the Eq. (2.23) and get an approximate value, $t \sim 3.02257$. The initial state is taken to be $c(x,0) = 0.1$ on the computational domain $\Omega = (0,1)$ with $h = 1/32$ and ϵ_4. We set the final time $T = 3.02257$ and time step $\Delta t = T/100$. Fig. 2.5 shows an evolution of the constant concentration $c(x,t)$ with different time steps ($\Delta t = 0.0604$, $\Delta t/4$, $\Delta t/16$) up to the final time. We can see the convergence of the numerical solutions with respect to the time steps.

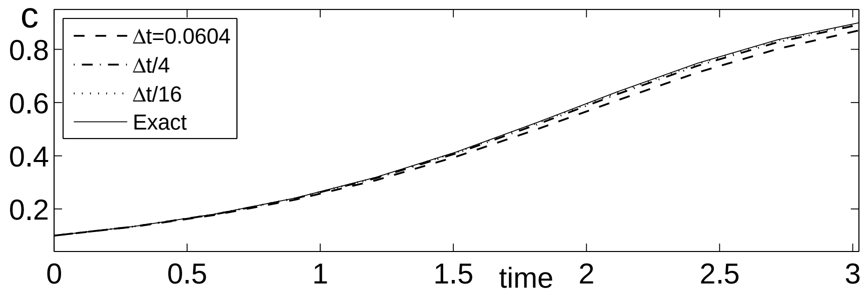


FIGURE 2.5. The evolution of the constant value of $c(x,t)$ with different time steps up to the final time, $T = 3.02257$.

2.4.5. Traveling wave solutions. We seek traveling wave solutions of Eq. (2.1) as a following form,

$$c(x,t) = \frac{1}{2}\left(1 - \tanh\frac{x - st}{2\sqrt{2}\epsilon}\right), \tag{2.24}$$

where s is the speed of the traveling wave. By substituting Eq. (2.24) into Eq. (2.1), we arrive at the following equation.

$$\frac{\sqrt{2}s - 3\epsilon}{8\epsilon}\text{sech}^2\left(\frac{x - \alpha t}{2\sqrt{2}\epsilon}\right) = 0. \tag{2.25}$$

Therefore, the speed of the traveling wave is $s = 3\epsilon/\sqrt{2}$.

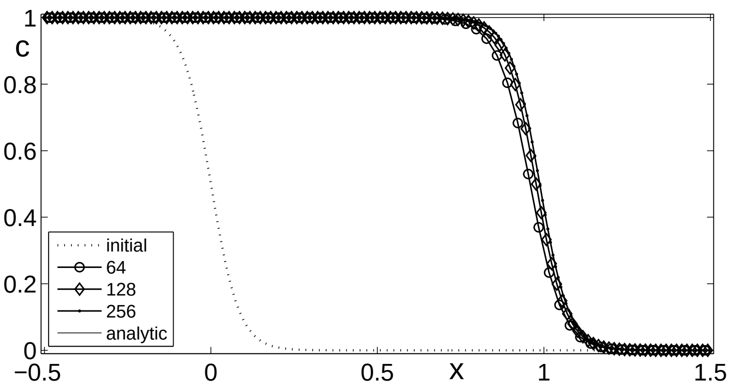


FIGURE 2.6. Numerical traveling wave solutions with an initial profile, $c(x,0) = \frac{1}{2}(1 - \tanh\frac{x}{2\sqrt{2}\epsilon})$. The final time is $T = 1/s$. The analytic solution is a solid line. Lines with symbols, 'o', '◇', and '·' represent mesh sizes of 64, 128, and 256, respectively.

In Fig. 2.6, the numerical traveling wave solutions (where symbols 'o', '$\diamond$', and '$\cdot$' represent mesh sizes of 64, 128, and 256, respectively) with an initial profile, $c(x,0) = \frac{1}{2}(1 - \tanh\frac{x}{2\sqrt{2}\epsilon})$ and on a computational domain, $\Omega = (-0.5, 1.5)$. The final time is $T = 1/s$ and the fixed time step is $\Delta t = T/400$. The analytic final profile is $c(x, 1/s) = \frac{1}{2}(1 - \tanh\frac{x-1}{2\sqrt{2}\epsilon})$. The convergence of the results with grid refinement is qualitatively evident.

To obtain a quantitative estimate of the rate of convergence, we perform a number of simulations for the same initial problem on a set of increasingly finer grids and time steps. The numerical solutions are computed on the uniform grids and time steps, $h = 2/2^n$ and $\Delta t = 5h^2$, for $n = 6, 7,$ and 8. The errors and rates of convergence are given in table 2.1. The results suggest that the scheme is indeed second order accurate in space and first order in time.

TABLE 2.1. Convergence results.

Case	64	rate	128	rate	256
l_2	1.0709e-2	1.9938	2.6887e-3	2.0140	6.6570e-4

In Fig. 2.7, we show numerical traveling wave solutions with an initial profile, $c(x,0) = 1$ if $x < 0.2$; $c(x,0) = 0$, otherwise. Solid, circle, diamond, and star lines are an initial profile, ϵ_4, $2\epsilon_4$, and $4\epsilon_4$, respectively. The results match well with the theoretical prediction of the speed, which depends linearly on the ϵ value.

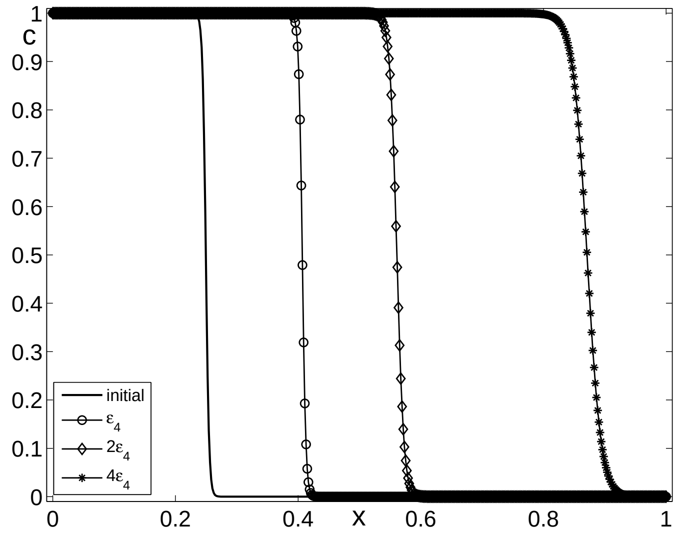


FIGURE 2.7. Numerical traveling wave solutions with an initial profile (solid line), $c(x,0) = 1$ if $x < 0.2$; $c(x,0) = 0$, otherwise. The circle, diamond, and star delineated lines represent numerical solutions with ϵ_4, $2\epsilon_4$, and $4\epsilon_4$, respectively. The computational mesh is 512, $\Delta t = 0.1$, and the final time is $T = 40$.

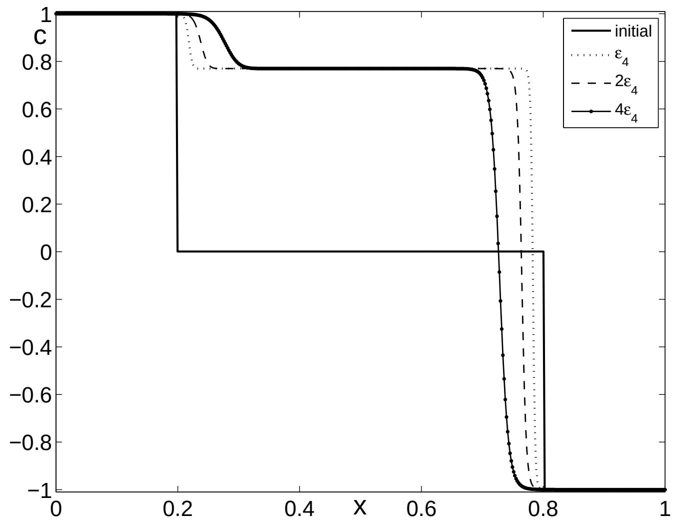


FIGURE 2.8. The evolution of the initial concentration is shown; $c(x, 0) = 1$ if $x < 0.2$ and $c(x, 0) = -1$ if $x > 0.8$; $c(x, 0) = 0.001$, otherwise with ϵ_4, $2\epsilon_4$, and $4\epsilon_4$. The computational mesh is 512, $\Delta t = 1/40$, and the final time is $T = 10$.

Fig. 2.8 shows evolution of the initial concentration, $c(x, 0) = 1$ if $x < 0.2$, $c(x, 0) = -1$ if $x > 0.8$; $c(x, 0) = 0.001$, otherwise with ϵ_4, $2\epsilon_4$, and $4\epsilon_4$. The computational mesh is 512, $\Delta t = 1/40$, and the final time is $T = 10$. The two transition layers at around $x = 0.2$ and $x = 0.8$ travels like traveling wave solution. On the other hand, in the middle in the domain behaves like a constant solution, Eq. (2.23). And this constant solution does not depend on ϵ values.

2.4.6. Metastable states. In Figs. 2.9(a) and (b) show the evolutions of the concentration $c(x, t)$ with 12 points and 13 points negative one with ϵ_7, respectively. The computational mesh is 128 and $\Delta t = 1/128$. In Fig. 2.9(c), the open circles denote ϵ_m and the number of points that have constant equilibrium solutions; while the stars indicate parameters that have non-constant equilibrium solutions.

2.4.7. Linear stability analysis. We perform a linear stability analysis around a spatially constant critical composition solution $c \equiv 0$. Linearizing the partial differential Eq. (2.8) about $c \equiv 0$ gives

$$c_t = c + \epsilon^2 \Delta c. \tag{2.26}$$

Next, we let $c = \alpha(t)\cos(k\pi x)$. Then from Eq. (2.26) we have

$$\alpha'(t)\cos(k\pi x) = \alpha(t)\cos(k\pi x) - (\epsilon k\pi)^2 \alpha(t)\cos(k\pi x). \tag{2.27}$$

By dividing Eq. (2.27) by $\cos(k\pi x)$, we obtain

$$\alpha'(t) = [1 - (\epsilon k\pi)^2]\alpha(t). \tag{2.28}$$

The solution of the ordinary differential equation (2.28) is given by

$$\alpha(t) = \alpha(0)e^{\lambda t}, \tag{2.29}$$

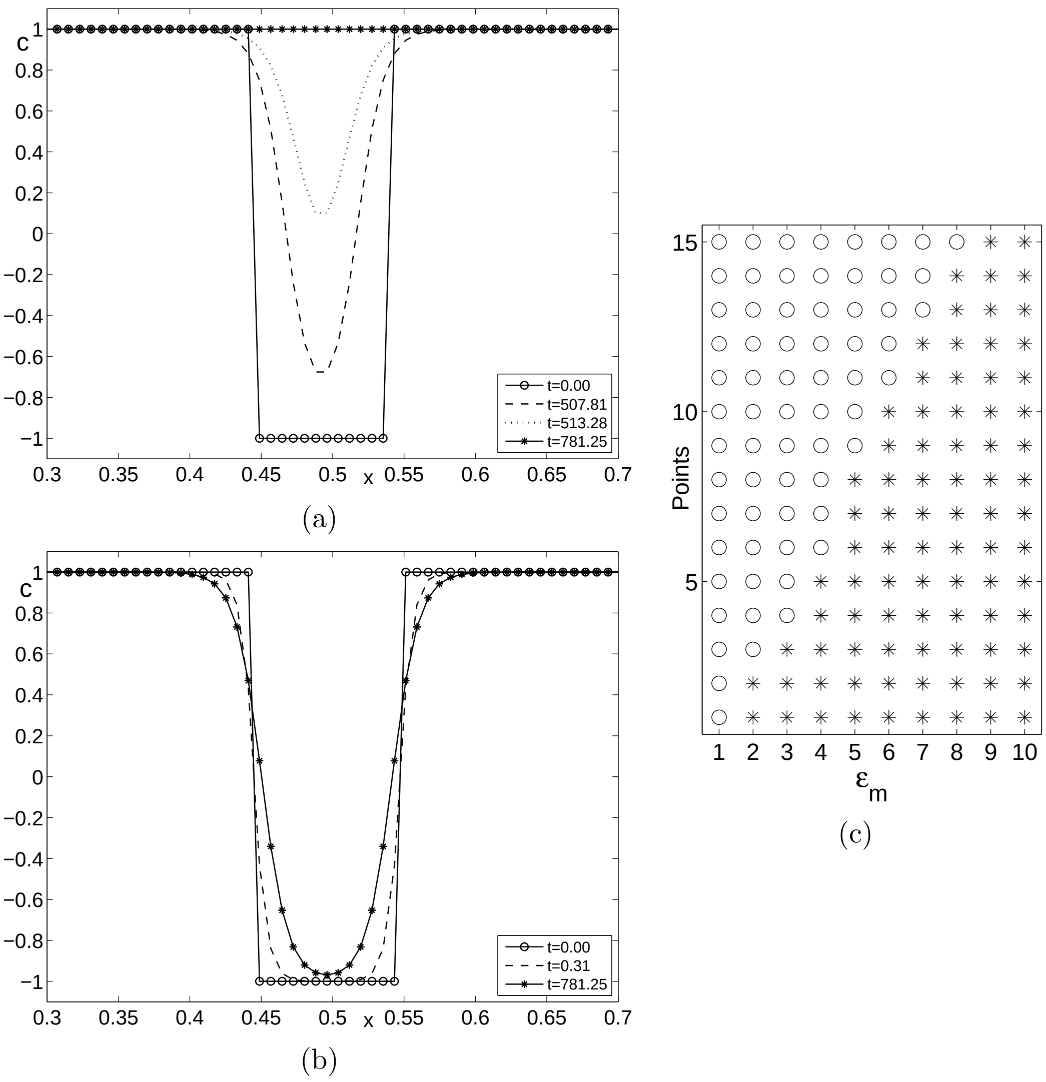


FIGURE 2.9. (a) and (b) are the evolutions of the concentration $c(x,t)$ with 12 points and 13 points negative one with ϵ_7, respectively. (c) The stars denote the pairs of ϵ_m and the number of negative one points that have constant equilibrium solutions as shown in (a), while the open circles indicate parameters that have non-constant equilibrium solutions as shown in (b).

where $\lambda = 1-(\epsilon k\pi)^2$. The numerical growth rate is defined by

$$\tilde{\lambda} = \frac{1}{T}\log\left(\frac{\max_{1\le i\le N}|c_i^{N_t}|}{\alpha(0)}\right). \tag{2.30}$$

The initial state is taken to be $c(x,0) = 0.01\cos(k\pi x)$ on the computational domain, $\Omega = (0,1)$. We use parameters such as the final time $T = 0.1$, time step $\Delta t = 0.01$,

$\alpha(0) = 0.01$, $\epsilon = 0.04$, and $N = 256$. In Fig. 2.10, numerical $\tilde{\lambda}$ ('o') and exact λ (solid line) values for different wave numbers k ($k = 0, 1, \cdots, 10$) are shown and are in good agreement.

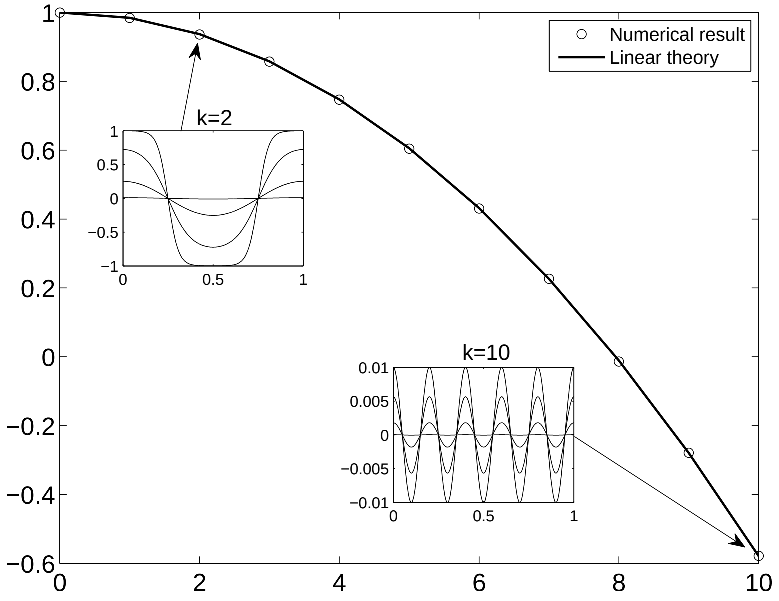


FIGURE 2.10. Numerical and exact λ values for different wave numbers k ($k = 0, 1, \cdots, 10$) and $\epsilon = 0.04$. The insets illustrate time evolutions of the initial profiles.

2.4.8. The decrease of the total energy. In order to demonstrate that the numerical scheme is unconditionally gradient stable, we consider the evolution of the discrete total energy. The initial state is taken to be $c(x, y, 0) = 0.01\text{rand}(x, y)$ on the computational domain $\Omega = (0, 1) \times (0, 1)$ with 128×128. rand(x, y) is a random number between -1 and 1. We use the simulation parameters ϵ_4 and $\Delta t = 10/128$. In Fig. 2.11, the time evolution of the non-dimensional discrete total energy $\mathcal{E}^h(\mathbf{c}^n)/\mathcal{E}^h(\mathbf{c}^0)$ is shown. Also, the inscribed small figures are the concentration fields at the indicated times. The total discrete energy is non-increasing as predicted by Eq. (2.15).

2.4.9. Mean curvature flow. After rescaling the time variable, Eq. (2.1) becomes

$$c_t = -\frac{c(c^2 - 1)}{\epsilon^2} + \Delta c. \tag{2.31}$$

It was first formally proved that, as $\epsilon \to 0$, the zero level set of c, denoted by $\Gamma_t^\epsilon := \{\mathbf{x} \in \Omega; c(\mathbf{x}, t) = 0\}$ approaches to a surface Γ_t that evolves according to the geometric law

$$V = -\kappa = -\left(\frac{1}{R_1} + \frac{1}{R_2}\right), \tag{2.32}$$

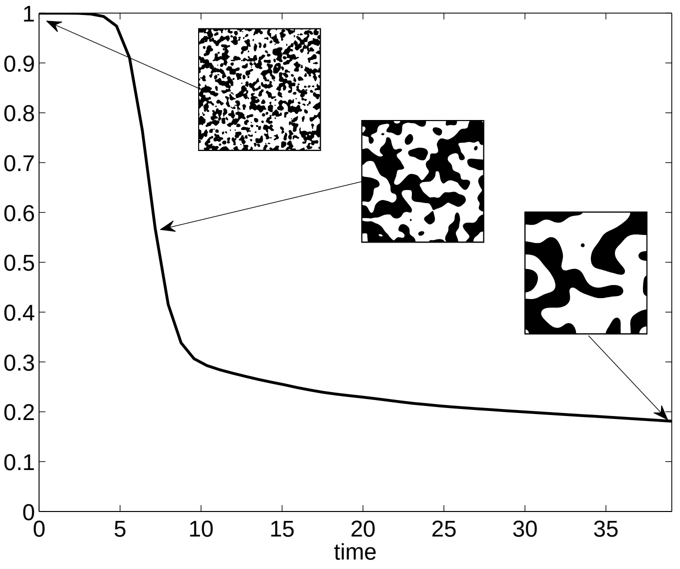


FIGURE 2.11. The time dependent non-dimensional discrete total energy $\mathcal{E}^h(\mathbf{c}^n)/\mathcal{E}^h(\mathbf{c}^0)$ of the numerical solutions with the initial data, $c(x, y, 0) = 0.01\text{rand}(x, y)$.

where V is the normal velocity of the surface Γ_t at each point, κ is its mean curvature, and R_1, R_2 are the principal radii of curvatures at the point of the surface [1]. In two dimensions, with a single radius of curvature, Eq. (2.32) becomes $V = -1/R$.

An initial condition is given with a circle centered at the center of a domain $\Omega = (0,1) \times (0,1)$. If we set the initial radius of the circle to r_0 and denote the radius at time t as $r(t)$, then Eq. (2.32) becomes $dr(t)/dt = -1/r(t)$. Its solution is given as

$$r(t) = \sqrt{r_0^2 - 2t}. \tag{2.33}$$

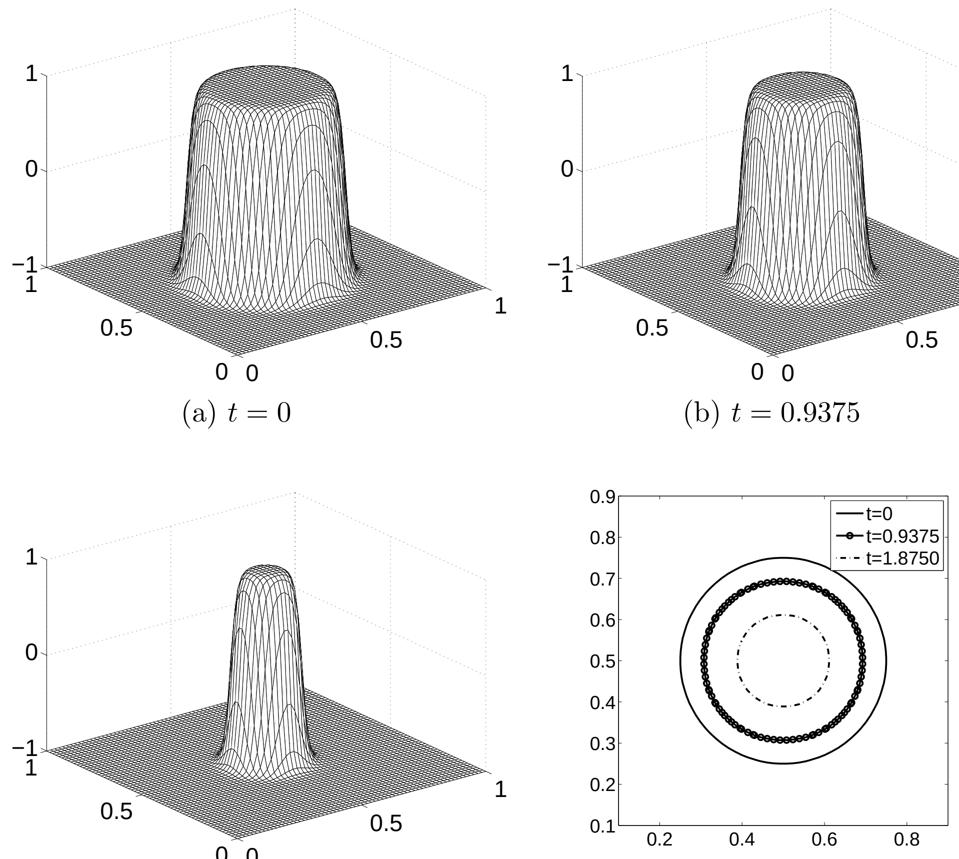
Let $r_0 = 0.25$. Then, the initial condition is

$$c(x, y, 0) = \tanh \frac{0.25 - \sqrt{(x-0.5)^2 + (y-0.5)^2}}{\sqrt{2}\epsilon}. \tag{2.34}$$

In Figs. 2.12, (a), (b), and (c) are evolutions of the initial concentration $c(x, y, 0)$, Eq. (2.34). The times are shown below each figure. (d) illustrates the zero level contour lines of (a), (b), and (c). We observe that the circle shrinks as theoretically predicted.

In Fig. 2.13, we show that as the mesh size decreases from $h = 1/64$ ('$\circ$') to $h = 1/128$ ('$\diamond$') and $h = 1/256$ ('$\cdot$'), the plot of numerical radius of the circle $r(t)$ in time becomes closer and closer to the asymptotic value (solid line) given by Eq. (2.33). We note that as mesh sizes decrease the actual value of ϵ_4 decreases also. This result confirms the theory that as $\epsilon \to 0$ then $\Gamma_t^\epsilon \to \Gamma_t$.

2.4.10. Adaptive mesh refinement. Across the spatial interfaces, the solution undergoes an $O(1)$ change over an $O(\epsilon)$ interval. If these interfaces are to be accurately resolved, a fine discretization of space is required. Therefore, an adaptive mesh refinement (AMR) of the space is necessary. In this approach, the computational mesh is



(a) $t = 0$

(b) $t = 0.9375$

(c) $t = 1.8750$

(d)

Figure 2.12. (a), (b), and (c) show evolutions of the initial concentration $c(x, y, 0)$, Eq. (2.34). The times are shown below each figure. (d) illustrates the zero level contour lines of (a), (b), and (c).

locally refined in regions where greater accuracy is desired. We use Marsha Berger's and Phillip Colella's block-structured approach; where refinement is organized in logically rectangular regions of the domain [2, 4, 5, 101, 92]. We also implement the unconditionally gradient stable scheme in Eq. (2.8) with the recently developed adaptive mesh refinement methodology. For a detailed description of the numerical method used in solving these equations with AMR, please refer to Refs. [12, 15].

In this simulation we consider the evolution of an initial state is taken to be $c(x, y, 0) = 0.01\text{rand}(x, y)$ on the computational domain $\Omega = (0, 1) \times (0, 1)$. We use a base 64×64 mesh with two levels of refinement ratios of 2. Therefore, the effective fine mesh size is 256×256. We use the simulation parameters such as ϵ_4 based on the effective fine mesh and $\Delta t = 0.1$. Fig. 2.14 shows the evolution of the concentration $c(x, y, t)$ at times $t = 100$ and 300. At $t = 300$, the mesh adapts around the zero level set.

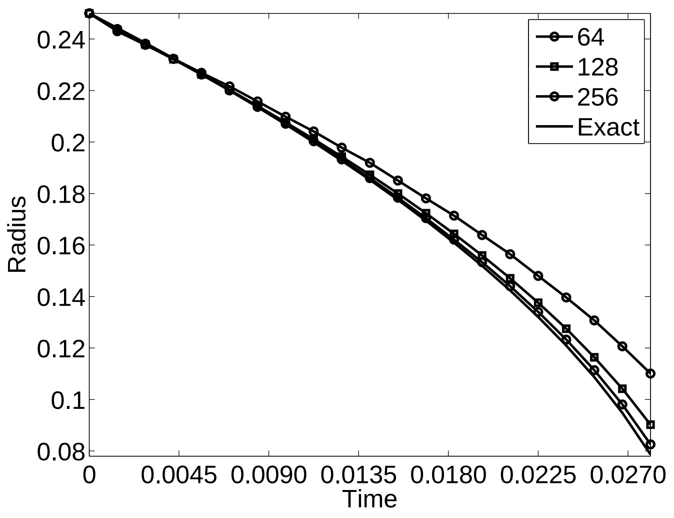


FIGURE 2.13. Mesh refinement with ϵ_4.

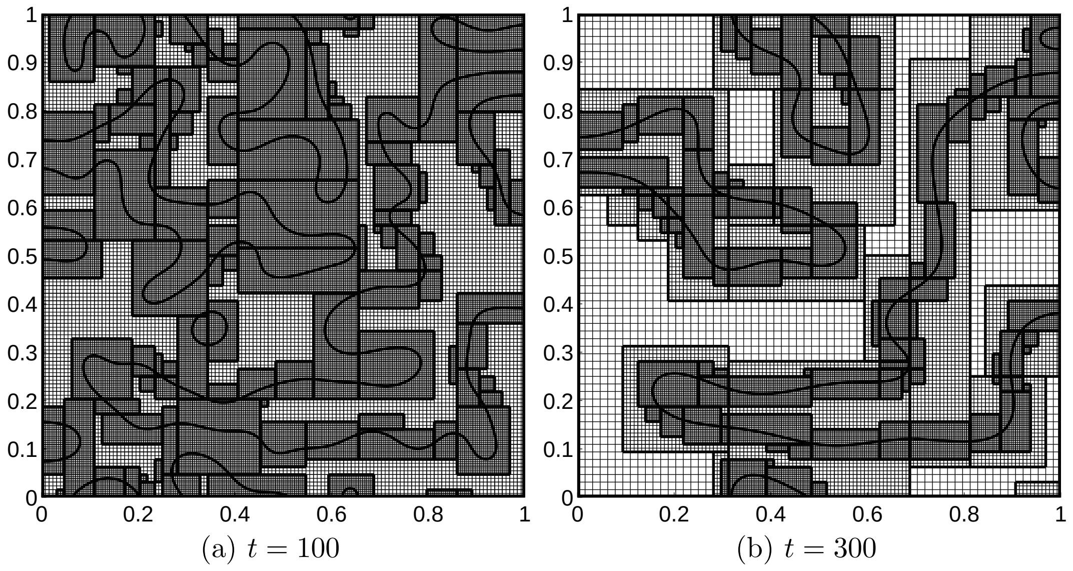


FIGURE 2.14. The evolution of the mesh and concentration $c(x, y, t) = 0$ under anti-phase domain coarsening. The times are shown below each figure. The effective fine grid resolution for 2 levels of adaptivity is 256×256.

2.5. Conclusions

In this paper, we reviewed a derivation of the AC equation as a gradient flow and showed that a numerical scheme for the AC equation is unconditionally gradient stable by using eigenvalues of the Hessian matrix of the energy functional. We also showed that the decrease of the discrete total energy functional implies the pointwise boundedness of the numerical solution for the AC equation. We investigated a variety of phenomena associated with the AC equation. We have uncovered a traveling wave solution to the AC equation and found that its speed depends linearly on the interfacial energy parameter, i.e., $s = 3\epsilon/\sqrt{2}$.

This chapter is published in Physica A, Vol. 388, No. 9, pp. 1591-1606, (2009).

Chapter 3

A computer program for the phase-field models

Phase-field model is an important methodology for modeling physical phenomena such as solidification, ferroelectrics, spinodal decomposition, and polymer blend. In this Chapter, we review and present details of the computational scheme and computer program for the phase-field models. The scheme is unconditionally gradient stable and is solved by an efficient and accurate multigrid method. And the program, which is written in the C language, is designed to provide the researchers with an easy and detailed guidance for the multigrid method of the phase-field models. We discuss the implementation of the code through numerical experiments.

3.1. Introduction

The code presented in this Chapter may be extended for various models such as the Rayleigh-Taylor instability [16], the pinch-off in liquid/liquid jets [17], thermocapillary flows [18, 19], mixing [20], contact angles and wetting phenomena [21, 92], gravity and capillary waves [22, 23, 24], and nucleation and spinodal decomposition [17, 18, 25, 26, 27].

In this Chapter, we consider an implementation of the Cahn-Hilliard (CH) equation:

$$\frac{\partial c(\mathbf{x},t)}{\partial t} = \nabla\cdot[M(c(\mathbf{x},t))\nabla\mu(c(\mathbf{x},t))],\ \mathbf{x}\in\Omega,\ 0<t\le T, \tag{3.1}$$

$$\mu(c(\mathbf{x},t)) = F'(c(\mathbf{x},t)) - \epsilon^2\Delta c(\mathbf{x},t), \tag{3.2}$$

where $\Omega\subset\mathbf{R}^d$ $(d=1,2,3)$. The quantity $c(\mathbf{x},t)$ is defined to be the difference between the concentrations of the two mixtures' components (e.g., $(m_1-m_2)/(m_1+m_2)$ where m_1 and m_2 are the masses of components 1 and 2 in a representative volume V). The CH equation was originally introduced to model spinodal decomposition and coarsening phenomena in binary alloys [46, 51]. This equation is very important in modeling and simulation of materials science applications (see Refs. [46, 51, 28, 115] and references therein). This CH equation arises from the Ginzburg-Landau free energy

$$\mathcal{E}(c) := \int_\Omega \left(F(c) + \frac{\epsilon^2}{2}|\nabla c|^2\right) d\mathbf{x},$$

where $F(c) = 0.25(c^2-1)^2$ is the Helmholtz free energy and ϵ is a positive constant. To obtain the CH equation with a variable mobility one introduces a chemical potential μ as the variational derivative of $\mathcal{E}$,

$$\mu := \frac{\delta\mathcal{E}}{\delta c} = F'(c) - \epsilon^2\Delta c$$

and defines the flux, $\mathcal{J} := -M(c)\nabla\mu$, where $M(c) \geq 0$ is a diffusional mobility. As a consequence of mass conservation, we have

$$\frac{\partial c}{\partial t} = -\nabla \cdot \mathcal{J},$$

which is the CH equation with a variable mobility. The natural and no-flux boundary conditions are

$$\frac{\partial c}{\partial n} = \mathcal{J} \cdot n = 0 \ \text{ on } \partial\Omega, \ \text{ where } n \text{ is normal to } \partial\Omega. \tag{3.3}$$

We differentiate the energy $\mathcal{E}$ and the total mass $\int_\Omega c \, d\mathbf{x}$ to get

$$\begin{aligned}\frac{d}{dt}\mathcal{E}(t) &= \int_\Omega (F'(c)c_t + \epsilon^2 \nabla c \cdot \nabla c_t) d\mathbf{x} = \int_\Omega \mu c_t \, d\mathbf{x} = \int_\Omega \mu \nabla \cdot (M(c)\nabla\mu) d\mathbf{x} \\ &= \int_{\partial\Omega} \mu M(c) \frac{\partial \mu}{\partial n} \, ds - \int_\Omega \nabla\mu \cdot (M(c)\nabla\mu) d\mathbf{x} = -\int_\Omega M(c)|\nabla\mu|^2 d\mathbf{x}\end{aligned}$$

and

$$\frac{d}{dt}\int_\Omega c \, d\mathbf{x} = \int_\Omega c_t \, d\mathbf{x} = \int_\Omega \nabla \cdot (M(c)\nabla\mu) d\mathbf{x} = \int_{\partial\Omega} M(c)\frac{\partial\mu}{\partial n} \, ds = 0,$$

where we used the no-flux boundary condition (3.3). Therefore, the total energy is non-increasing in time and the total mass is conserved.

In this Chapter, we briefly review numerical methods for the CH equation and provide details of a computational scheme and the complete C code. The code presented in this Chapter is intended to provide a comprehensive implementation guide for phase-field models. Researchers to the field of phase-field models can download the code from the webpage `http://elie.korea.ac.kr/~cfdkim/CHcode/`.

This Chapter is organized as follows. In Section 3.2, we briefly review numerical methods for the CH equation. In Section 3.3, we describe the discrete scheme and show the boundedness of its numerical solution. We present the multigrid method for the fully discrete system in Section 3.4. The numerical results are described in Section 3.5. A discussion is presented in Section 3.6. In Section 3.7, we provide the multigrid C code for solving the CH equation.

3.2. Review

In this section, we present a brief review of numerical methods for the CH equation. The CH equation with finite element [29, 30, 31, 32, 33, 34] and finite difference [35, 36, 37, 38, 39] methods have been studied intensively. For one-dimensional problems, finite Galerkin approximate solutions have been obtained by Elliott and French [29]. Also this authors considered a nonconforming finite element method for multi-dimensional problems in [30]. Elliott and Larsson [31] have obtained error bounds of finite Galerkin solutions with smooth and nonsmooth initial data. In [32], Dean et al. used a mixed finite element method to obtain approximate solutions of the CH equation. A multigrid finite element solver has been presented by Kay and Welford [33] and an adaptive finite element method was developed by Banas and Nürnberg [34]. Furihata [35] has proposed a stable and conservative finite difference scheme to solve numerically the CH equation. Choo and Chung [36] have applied a nonlinear conservative difference scheme based on the Crank-Nicolson scheme for a one-dimensional problem and Choo et al. [37]

used a nonlinear conservative difference scheme for a two-dimensional problem. Zhou and Wang [38] introduced a phase field model for the optimization of multimaterial structural topology based a modified Cahn-Hilliard theory. Kim [39] has considered a conservative nonlinear multigrid method for the CH equation with a variable mobility. Multigrid methods are generally accepted as among *the fastest numerical methods* for solving this type of partial differential equations [122]. We use a multigrid method [33, 39, 40, 41, 42] to solve the CH equation accurately and efficiently.

3.3. Numerical analysis

In this section, we present fully discrete schemes for the CH equation. We consider an unconditionally gradient stable scheme for time discretization introduced by Eyre [53, 43]. For simplicity of exposition, we shall discretize the CH equation in two-dimensional space, i.e., $\Omega = (a, b) \times (c, d)$. One and three-dimensional discretizations are defined analogously. Let N_x and N_y be positive even integers, $h = (b - a)/N_x$ be the uniform mesh size, and $\Omega_h = \{(x_i, y_j) : x_i = (i - 0.5)h,\ y_j = (j - 0.5)h,\ 1 \le i \le N_x,\ 1 \le j \le N_y\}$ be the set of cell-centers. Let c_{ij} and μ_{ij} be approximations of $c(x_i, y_j)$ and $\mu(x_i, y_j)$. We define a discrete energy functional by

$$\mathcal{E}^h(\mathbf{c}^n) = \frac{h^2}{4}\sum_{i=1}^{N_x}\sum_{j=1}^{N_y}((c_{ij}^n)^2-1)^2 + \frac{\epsilon^2 h^2}{2}\left(\sum_{i=0}^{N_x}\sum_{j=1}^{N_y}|\nabla_d c^n_{i+\frac{1}{2},j}|^2 + \sum_{i=1}^{N_x}\sum_{j=0}^{N_y}|\nabla_d c^n_{i,j+\frac{1}{2}}|^2\right).$$

By using the linearly stabilized splitting scheme [43], we present a implicit time and centered difference space discretization of Eqs. (3.1) and (3.2):

$$\frac{c_{ij}^{n+1} - c_{ij}^n}{\Delta t} = \Delta_d \mu_{ij}^{n+1}, \tag{3.4}$$

$$\mu_{ij}^{n+1} = 2c_{ij}^{n+1} - \epsilon^2 \Delta_d c_{ij}^{n+1} + f(c_{ij}^n) - 2c_{ij}^n, \tag{3.5}$$

where $f(c) = F'(c)$.

Eyre [53, 43] proved that if $\mathbf{c}^{n+1}$ is the solution of Eqs. (3.4) and (3.5) with a given $\mathbf{c}^n$, then

$$\mathcal{E}^h(\mathbf{c}^{n+1}) \le \mathcal{E}^h(\mathbf{c}^n).$$

Furihata et al. [44] have examined the boundedness of the solution of a finite difference scheme [45] using discretized Lyapunov functional. Furihata [35] has shown that the decrease of the total energy implies boundedness of discretized Sobolev norm of the solution, independent Δt and Δx, unconditionally.

We can show that boundedness of the solution of finite difference scheme using the decrease of the discrete total energy functional. This proof can be done similarly by using method in Subsec. 2.3.2 in Chapter 2. The decrease of the discrete total energy functional means that the numerical solution for the CH equation is pointwise bounded.

3.4. Numerical solution - A multigrid method

In this section, we develop a multigrid method to solve the linear discrete system (3.4) and (3.5) at the implicit time level. A pointwise Gauss-Seidel relaxation scheme is used as the smoother in the multigrid method. See the reference text [122] for additional details. We use the same notations as this reference text. Let us rewrite Eqs. (3.4) and (3.5) as follows.

$$L(c^{n+1}, \mu^{n+1}) = (\phi^n, \psi^n),$$

where the linear system operator (L) is defined as

$$L(c^{n+1}, \mu^{n+1}) = \left(\frac{c^{n+1}}{\Delta t} - \Delta_d \mu^{n+1},\ \mu^{n+1} - 2c^{n+1} + \epsilon^2 \Delta_d c^{n+1} \right)$$

and the source term is $(\phi^n, \psi^n) = (c^n/\Delta t,\ f(c^n) - 2c^n)$. In the following description of one multigrid cycle, we assume a sequence of grids Ω_k (Ω_{k-1} is coarser than Ω_k by a factor of 2). 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:

Multigrid cycle

$$\{c_k^{n+1,m+1}, \mu_k^{n+1,m+1}\} = MGcycle(k, c_k^{n+1,m}, \mu_k^{n+1,m}, L_k, \phi_k^n, \psi_k^n, \nu_1, \nu_2) \text{ on } \Omega_k \text{ grid.}$$

Here, $\{c_k^{n+1,m}, \mu_k^{n+1,m}\}$ and $\{c_k^{n+1,m+1}, \mu_k^{n+1,m+1}\}$ are the approximations of $\{c_k^{n+1}(x_i, y_j), \mu_k^{n+1}(x_i, y_j)\}$ before and after an MGcycle.

Step 1) Presmoothing

$$\{\bar{c}_k^{n+1,m}, \bar{\mu}_k^{n+1,m}\} = SMOOTH^{\nu_1}(c_k^{n+1,m}, \mu_k^{n+1,m}, L_k, \phi_k^n, \psi_k^n) \text{ on } \Omega_k \text{ grid.}$$

This means performing ν_1 smoothing steps. Let us discretize Eqs. (3.4) and (3.5) as a Gauss-Seidel type.

$$\frac{\bar{c}_{k,ij}^{n+1,m}}{\Delta t} + \frac{4\bar{\mu}_{k,ij}^{n+1,m}}{h^2} = \phi_{k,ij}^n + \frac{\bar{\mu}_{k,i-1,j}^{n+1,m} + \mu_{k,i+1,j}^{n+1,m} + \bar{\mu}_{k,i,j-1}^{n+1,m} + \mu_{k,i,j+1}^{n+1,m}}{h^2}, \tag{3.6}$$

$$-\left(2 + \frac{4\epsilon^2}{h^2}\right) \bar{c}_{k,ij}^{n+1,m} + \bar{\mu}_{k,ij}^{n+1,m} = \psi_{k,ij}^n - \epsilon^2 \frac{\bar{c}_{k,i-1,j}^{n+1,m} + c_{k,i+1,j}^{n+1,m} + \bar{c}_{k,i,j-1}^{n+1,m} + c_{k,i,j+1}^{n+1,m}}{h^2}. \tag{3.7}$$

One SMOOTH relaxation operator step consists of solving the system (3.6) and (3.7) by a 2×2 matrix inversion for each ij.

Step 2) Coarse grid correction

- Compute the defect: $(\bar{d}_{1k}^m, \bar{d}_{2k}^m) = (\phi_k^n, \psi_k^n) - L_k(\bar{c}_k^{n+1,m}, \bar{\mu}_k^{n+1,m})$.
- Restrict the defect: $(\bar{d}_{1,k-1}^m, \bar{d}_{2,k-1}^m) = I_k^{k-1}(\bar{d}_{1k}^m, \bar{d}_{2k}^m)$.
- Compute an approximate solution $\{\bar{c}_{k-1}^m, \bar{\mu}_{k-1}^m\}$ of the coarse grid equation on Ω_{k-1}:

$$L_{k-1}(c_{k-1}^{n+1,m}, \mu_{k-1}^{n+1,m}) = (\bar{d}_{1,k-1}^m, \bar{d}_{2,k-1}^m). \tag{3.8}$$

If $k = 1$, we apply the smoothing procedure in *Step 1)* to obtain the approximate solution. If $k > 1$, we solve (3.8) by performing a k-grid cycle using the zero grid function as an initial approximation:

$$\{\hat{v}_{1,k-1}^{n+1,m}, \hat{v}_{2,k-1}^{n+1,m}\} = MGcycle(k-1, \mathbf{0}, \mathbf{0}, L_{k-1}, \bar{d}_{1,k-1}^m, \bar{d}_{2,k-1}^m, \nu_1, \nu_2).$$

• Interpolate the correction: $(\hat{v}_{1k}^{n+1,m}, \hat{v}_{2k}^{n+1,m}) = I_{k-1}^{k}(\hat{v}_{1,k-1}^{n+1,m}, \hat{v}_{2,k-1}^{n+1,m})$.
• Compute the corrected approximation on Ω_k:

$$c_k^{m,\text{ after } CGC} = \bar{c}_k^{n+1,m} + \hat{v}_{1k}^{n+1,m}, \ \mu_k^{m,\text{ after } CGC} = \bar{\mu}_k^{n+1,m} + \hat{v}_{2k}^{n+1,m}.$$

Step 3) Postsmoothing:

$$\{c_k^{n+1,m+1}, \mu_k^{n+1,m+1}\}$$
$$= SMOOTH^{\nu_2}(c_k^{m,\text{ after } CGC}, \mu_k^{m,\text{ after } CGC}, L_k, \phi_k^n, \psi_k^n) \text{ on } \Omega_k \text{ grid.}$$

This completes the description of an MGcycle.

Describing an algorithm for a discrete system is one thing and actual implementation is another. Especially, using recursive function for a multigrid method is non-trivial task for the beginners. We made the code as simple as possible. Therefore, it needs a modification for accounting other features of the equations such as different boundary conditions, variable mobilities, three dimensional extension, and more than two components. These modifications can be done in a straightforward manner.

3.5. Numerical results

3.5.1. The relation between the ϵ value and the width of the transition layer. In the first numerical experiment, we consider the relation between the ϵ value and the width of the transition layer for the CH equation. From our choice of an equilibrium profile $c(x) = \tanh(x/(\sqrt{2}\epsilon))$ on the infinite domain, the concentration field varies from -0.9 to 0.9 over a distance of about $\xi = 2\sqrt{2}\epsilon \tanh^{-1}(0.9)$ (see Fig. 3.1).

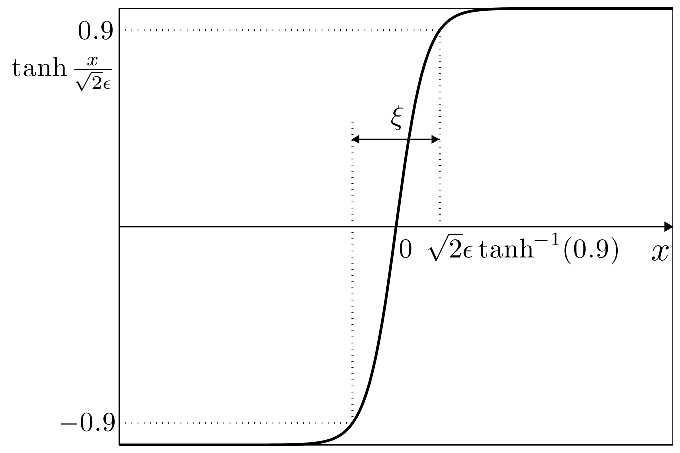


FIGURE 3.1. The concentration field varies from -0.9 to 0.9 over a distance of about $\xi = 2\sqrt{2}\epsilon \tanh^{-1}(0.9)$.

Therefore, if we want this value to be about m grid points, then

$$\epsilon_m = \frac{hm}{2\sqrt{2}\tanh^{-1}(0.9)}.$$

To confirm this, we run a simulation with the initial condition $c(x, y, 0) = 0.01\text{rand}(\ )$ on the unit domain $\Omega = (0, 1) \times (0, 1)$ with $h = 1/64$, $\Delta t = 0.05$, and ϵ_4. Here, rand( ) is a random number between -1 and 1. In Fig. 3.2, we see that the transition layer (from $c = -0.9$ to $c = 0.9$) is about 4 grid points at $t = 25$.

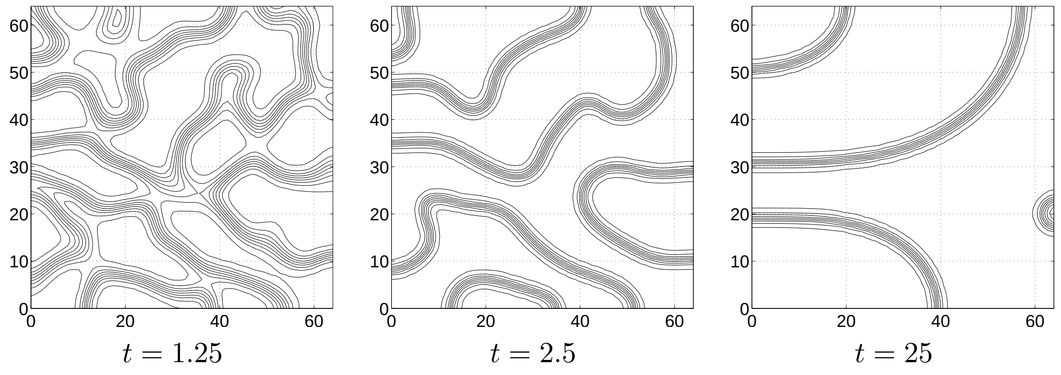


FIGURE 3.2. The evolution of an initial random concentration, $c(x, y, 0) = 0.01\text{rand}(\ )$. The concentration profile is shown at $t = 1.25$, 2.5, and 25.

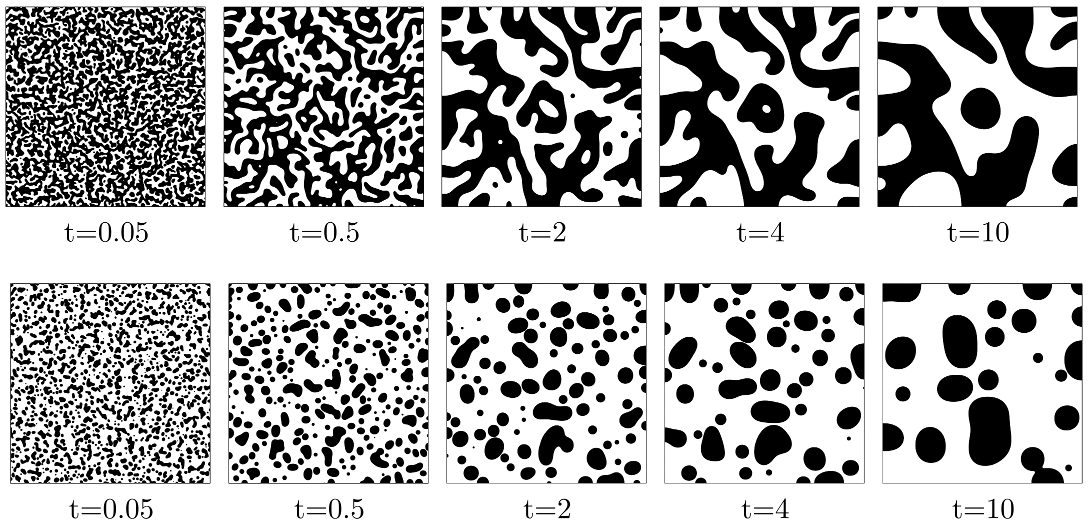


FIGURE 3.3. The evolution of the concentration $c(x, y, t)$ with different initial conditions, $c(x, y, 0) = 0.01\text{rand}(\ )$ (the top row) and $c(x, y, 0) = -0.4 + 0.1\text{rand}(\ )$ (the bottom row). The times are shown below each figure.

3.5.1.1. *Phase separation.* Next, to confirm the phase separation with different initial states, we take two different initial conditions on $\Omega = (0, 1) \times (0, 1)$, $c(x, y, 0) = 0.01\text{rand}(\ )$ and $c(x, y, 0) = -0.4 + 0.1\text{rand}(\ )$. We take the simulation parameters, $h = 1/1024$, $\Delta t = 0.01$, and ϵ_4. Fig. 3.3 shows the evolution of the concentration $c(x, y, t)$. The first and second rows represent the evolution of the concentration 50%

with the initial condition $c(x, y, 0) = 0.01\text{rand}(\ )$ and the concentration 30% with the initial condition $c(x, y, 0) = -0.4 + 0.1\text{rand}(\ )$, respectively.

3.5.1.2. *The decrease of the total energy.* In order to demonstrate that the numerical scheme inherits the energy decreasing property, we consider the evolution of the discrete total energy. In the simulation, we choose $h = 1/128$, $\Delta t = 0.01$, and ϵ_4.

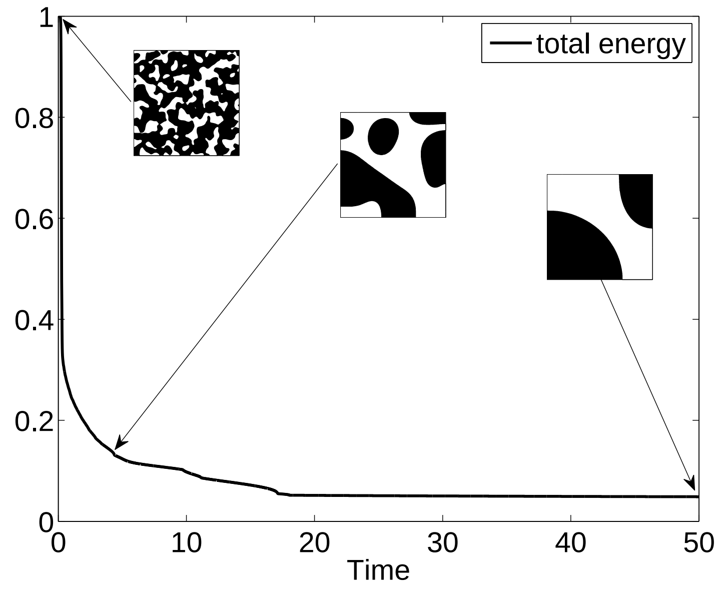


FIGURE 3.4. The time dependent non-dimensional discrete total energy $\mathcal{E}_h(t)/\mathcal{E}_h(0)$ (solid line) of the numerical solutions with the initial state $c(x, y, 0) = 0.01\text{rand}(\ )$.

In Fig. 3.4, the time evolution of the non-dimensional discrete total energy $\mathcal{E}_h(t)/\mathcal{E}_h(0)$ (solid line) of the numerical solutions with the initial state $c(x, y, 0) = 0.01\text{rand}(\ )$. As can be seen in Fig. 3.4 the energy is non-increasing. This numerical result agrees well with the total energy dissipation property.

3.6. Conclusions

In this Chapter, we reviewed various numerical methods for solving the CH equation. We described the discrete scheme and its properties, and presented the multigrid method for the fully discrete system. Also, we provided a C program code for the CH equation. We hope that the code will play an useful role in the modeling and simulation for the phase-field models. The C program source code is available from the author's homepage at `http://elie.korea.ac.kr/~cfdkim/CHcode/`.

3.7. Cahn-Hilliard C code

In this section, we present a source code written in C language for the Cahn-Hilliard equation.

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#define nx 128
#define ny 128
#define iloop for(i=1;i<=nx;i++)
#define jloop for(j=1;j<=ny;j++)
#define ijloop int i, j; iloop jloop
#define iloopt for(i=1;i<=nxt;i++)
#define jloopt for(j=1;j<=nyt;j++)
#define ijloopt int i, j; iloopt jloopt
void initialization(double **cn);
void Cahn_Hilliard(double **cn, double **cnp);
void source(double **cn, double **src_c, double **src_mu);
void vcycle(double **cnp, double **mu, double **sor_c, double **sor_mu,
            int nxf, int nyf, int ilevel);
void relax(double **cnp, double **mu, double **sor_c, double **sor_mu,
           int ilevel, int nxt, int nyt);
void defect(double **def_c, double **def_mu, double **cn, double **cnp,
            double **sor_c, double **sor_mu, int nxf, int nyf);
void LS(double **LSc, double **LSmu, double **cnp, double **mu,
        int nxt, int nyt);
void laplace(double **a, double **lap_a, int nxt, int nyt);
void restrict(double **cf, double **cc, double **muf, double **muc,
              int nxt, int nyt);
void prolong(double **cc, double **cf, double **muc, double **muf,
             int nxt, int nyt);
double **dmatrix(long nrl, long nrh, long ncl, long nch);
void zero_matrix(double **a, int nxt, int nyt);
void free_dmatrix(double **m, long nrl, long nrh, long ncl, long nch);
void mat_add(double **a, double **b, double **c, int nxt, int nyt);
void mat_sub(double **a, double **b, double **c, int nxt, int nyt);
void mat_copy(double **a, double **b, int nxt, int nyt);
double error(double **ct, double **cnp);
void print_data(double **cnp);
int n_level, c_relax;
double **tmp1, **tmp2, **ct, **mu, **sor_c, **sor_mu, xleft, xright,
       yleft, yright, h, h2, dt, epsilon, Cahn;
int main()
{
   extern int n_level, c_relax;
   extern double **tmp1, **tmp2, **ct, **mu, **sor_c, **sor_mu,
                 xleft, xright, yleft, yright, h, h2, dt, epsilon, Cahn;
   int it, max_it, print_interval, count=1;
   double **cn, **cnp;
   FILE *fphi;
   if (nx<ny)  n_level=(int)(log(nx)/log(2)+0.1);
   else        n_level=(int)(log(ny)/log(2)+0.1);
   c_relax=5, max_it=50, print_interval=max_it/2,
   xleft=0.0, xright=1.0, yleft=0.0, yright=1.0;
   h=xright/(double)nx, h2=pow(h,2), dt=h, epsilon=h,
   Cahn=pow(epsilon,2);
   tmp1=dmatrix(1, nx, 1, ny);  tmp2=dmatrix(1, nx, 1, ny);
   ct=dmatrix(1, nx, 1, ny);  mu=dmatrix(1, nx, 1, ny);
   sor_c=dmatrix(1, nx, 1, ny);  sor_mu=dmatrix(1, nx, 1, ny);
   cn=dmatrix(1, nx, 1, ny);  cnp=dmatrix(1, nx, 1, ny);

   initialization(cnp);

   fphi=fopen("phi.m","w");
   fprintf(fphi,"A=[ \n");
   fclose(fphi);

   print_data(cnp);

   for (it=1; it<=max_it; it++) {
       mat_copy(cn, cnp, nx, ny);
       Cahn_Hilliard(cn, cnp);
       if (it % print_interval==0) {
          print_data(cnp);
          printf("print out counts %d \n", count++); }
       printf("Iteration is  %d \n", it); }

   fphi = fopen("phi.m","a");
   fprintf(fphi,"]; surf(A'); shading interp \n");
   fclose(fphi);

   return 0;
}
void initialization(double **cn)
{
   ijloop {
     cn[i][j]=0.3*(0.5-rand()/(double)RAND_MAX);
     mu[i][j]=0.0; }
}
void Cahn_Hilliard(double **cn, double **cnp)
{
   int max_it=300, iter=1;
   double tol=1.0e-5, resid=1.0;
   source(cn, sor_c, sor_mu);
   mat_copy(ct, cn, nx, ny);
   while (iter <= max_it && resid > tol) {
       vcycle(cnp, mu, sor_c, sor_mu, nx ,ny, 1);
       resid=error(ct, cnp);
       mat_copy(ct, cnp, nx, ny);
       iter++; }
   printf("Error is %12.10f %d",resid,iter-1);
}
void source(double **cn, double **src_c, double **src_mu)
{
   ijloop {
       src_c[i][j]=cn[i][j]/dt;
       src_mu[i][j]=pow(cn[i][j],3)-3.0*cn[i][j];}
}
void vcycle(double **cnp, double **mu, double **sor_c, double **sor_mu,
            int nxf, int nyf, int ilevel)
{
   relax(cnp, mu, sor_c, sor_mu, ilevel, nxf, nyf);

   if (ilevel<n_level) {

   double **def_c, **def_mu, **codef_c, **codef_mu,
          **fidef_c, **fidef_mu;

      def_c=dmatrix(1, nxf/2, 1, nyf/2);
      def_mu=dmatrix(1, nxf/2, 1, nyf/2);
      fidef_c=dmatrix(1, nxf, 1, nyf);
      fidef_mu=dmatrix(1, nxf, 1, nyf);
      codef_c=dmatrix(1, nxf/2, 1, nyf/2);
      codef_mu=dmatrix(1, nxf/2, 1, nyf/2);

      defect(def_c, def_mu, cnp, mu, sor_c, sor_mu, nxf, nyf);
      zero_matrix(codef_c, nxf/2, nyf/2);
      zero_matrix(codef_mu, nxf/2, nyf/2);
      vcycle(codef_c, codef_mu, def_c, def_mu, nxf/2, nyf/2, ilevel+1);
      prolong(codef_c, fidef_c, codef_mu, fidef_mu, nxf/2, nyf/2);
      mat_add(cnp, cnp, fidef_c, nxf, nyf);
      mat_add(mu, mu, fidef_mu, nxf, nyf);
      relax(cnp, mu, sor_c, sor_mu, ilevel, nxf, nyf);
      free_dmatrix(def_c, 1, nxf/2, 1, nyf/2);
      free_dmatrix(def_mu, 1, nxf/2, 1, nyf/2);
      free_dmatrix(fidef_c, 1, nxf, 1, nyf);
      free_dmatrix(fidef_mu, 1, nxf, 1, nyf);
      free_dmatrix(codef_c, 1, nxf/2, 1, nyf/2);
      free_dmatrix(codef_mu, 1, nxf/2, 1, nyf/2);
   }
}
void restrict(double **cf, double **cc, double **muf, double **muc,
              int nxt, int nyt)
{
   ijloopt {
       cc[i][j]=0.25*(cf[2*i][2*j]+cf[2*i-1][2*j]
                      +cf[2*i][2*j-1]+cf[2*i-1][2*j-1]);
       muc[i][j]=0.25*(muf[2*i][2*j]+muf[2*i-1][2*j]
                       +muf[2*i][2*j-1]+muf[2*i-1][2*j-1]); }
}
void prolong(double **cc, double **cf, double **muc, double **muf,
             int nxt, int nyt)
{
   ijloopt {
       cf[2*i][2*j]=cf[2*i-1][2*j]
       =cf[2*i][2*j-1]=cf[2*i-1][2*j-1]=cc[i][j];
       muf[2*i][2*j]=muf[2*i-1][2*j]
       =muf[2*i][2*j-1]=muf[2*i-1][2*j-1]=muc[i][j]; }
}
void LS(double **LSc, double **LSmu, double **cnp, double **mu,
        int nxt, int nyt)
{
   double **lap_mu, **lap_c;

   lap_mu=dmatrix(1, nxt, 1, nyt);
   lap_c=dmatrix(1, nxt, 1, nyt);
   laplace(cnp, lap_c, nxt, nyt);
   laplace(mu, lap_mu, nxt, nyt);

   ijloopt {
      LSc[i][j]=cnp[i][j]/dt-lap_mu[i][j];
      LSmu[i][j]=- 2.0*cnp[i][j]+Cahn*lap_c[i][j]+mu[i][j]; }

   free_dmatrix(lap_mu, 1, nxt, 1, nyt);
   free_dmatrix(lap_c, 1, nxt, 1, nyt);
}
void defect(double **def_c, double **def_mu, double **cn, double **cnp,
            double **sor_c, double **sor_mu, int nxf, int nyf)
{
   LS(tmp1, tmp2, cn, cnp, nxf, nyf);
   mat_sub(tmp1, sor_c, tmp1, nxf, nyf);
   mat_sub(tmp2, sor_mu, tmp2, nxf, nyf);
   restrict(tmp1, def_c, tmp2, def_mu, nxf/2, nyf/2);
}
double error(double **ct, double **cnp)
{
   double value=0.0;
   ijloop {
       if (fabs(ct[i][j]-cnp[i][j]) > value)
          value=fabs(ct[i][j]-cnp[i][j]); }

   return value;
}
void relax(double **cnp, double **mu, double **sor_c, double **sor_mu,
           int ilevel, int nxt, int nyt)
{
   int iter;
   double ht2, a[4], f[2], det;
   ht2 = pow(xright/(double) nxt,2);

   for (iter=1; iter<=c_relax; iter++) {
   ijloopt {
    a[0]=1.0/dt, a[1]=0.0, a[2]=-2.0, a[3]=1.0;
    f[0]=sor_c[i][j], f[1]=sor_mu[i][j];

    if (i>1) {a[1]+=1.0/ht2,a[2]-=Cahn/ht2;
              f[0]+=mu[i-1][j]/ht2,f[1]-=Cahn*cnp[i-1][j]/ht2;}
    if (i<nxt) {a[1]+=1.0/ht2,a[2]-=Cahn/ht2;
                f[0]+=mu[i+1][j]/ht2,f[1]-=Cahn*cnp[i+1][j]/ht2;}
    if (j>1) {a[1]+=1.0/ht2,a[2]-=Cahn/ht2;
              f[0]+=mu[i][j-1]/ht2,f[1]-=Cahn*cnp[i][j-1]/ht2;}
    if (j<nyt) {a[1]+=1.0/ht2,a[2]-=Cahn/ht2;
                f[0]+=mu[i][j+1]/ht2,f[1]-=Cahn*cnp[i][j+1]/ht2;}
    det = a[0]*a[3] - a[1]*a[2];
    cnp[i][j] = (a[3]*f[0] - a[1]*f[1])/det;
    mu[i][j] = (-a[2]*f[0] + a[0]*f[1])/det;}
   }
}
void laplace(double **a, double **lap_a, int nxt, int nyt)
{
   double ht2, value;
   ht2 = pow(xright / (double) nxt, 2);

   ijloopt {
       value=0.0;
       if (i<nxt) value += a[i+1][j]-a[i][j];
       if (i>1)   value -= a[i][j]-a[i-1][j];
       if (j<nyt) value += a[i][j+1]-a[i][j];
       if (j>1)   value -= a[i][j]-a[i][j-1];
       lap_a[i][j]=value/ht2;}
}
void mat_add(double **a, double **b, double **c, int nxt, int nyt)
{
   ijloopt a[i][j] = b[i][j]+c[i][j];
}
void zero_matrix(double **a, int nxt, int nyt)
{
   ijloopt a[i][j]=0.0;
}
void mat_copy(double **a, double **b, int nxt, int nyt)
{
   ijloopt a[i][j]=b[i][j];
}
double **dmatrix(long nrl, long nrh, long ncl, long nch)
```

```
{
   double **m;
   long i, nrow=nrh-nrl+1+1, ncol=nch-ncl+1+1;
   m=(double **) malloc((nrow)*sizeof(double*));
   m+=1;
   m-=nrl;
   m[nrl]=(double *) malloc((nrow*ncol)*sizeof(double));
   m[nrl]+=1;
   m[nrl]-=ncl;
   for (i=nrl+1; i<=nrh; i++) m[i]=m[i-1]+ncol;

   return m;
}
void free_dmatrix(double **m, long nrl, long nrh, long ncl, long nch)
{
   free(m[nrl]+ncl-1);
   free(m+nrl-1);
}
void mat_sub(double **a, double **b, double **c, int nxt, int nyt)
{
   ijloopt a[i][j]=b[i][j]-c[i][j];
}
void print_data(double **cnp)
{
   int i, j;
   FILE *fp;
   fp = fopen("phi.m","a");
   iloop {
       jloop { fprintf(fp, "  %f", cnp[i][j]);}
       fprintf(fp, "\n");}
   fclose(fp);
}
```

Chapter 4

A phase-field model for a block copolymer

We consider a numerical method for a block copolymer.

4.1. Introduction

In this paper, we consider an efficient and fast finite difference method for the Cahn-Hilliard (CH) equation.

$$\begin{aligned} \frac{\partial\phi(\mathbf{x},t)}{\partial t} &= \Delta\mu(\phi(\mathbf{x},t)) - \alpha(\phi(\mathbf{x},t)-\bar{\phi}),\ \mathbf{x}\in\Omega,\ 0<t\le T, && (4.1)\\ \mu(\phi(\mathbf{x},t)) &= F'(\phi(\mathbf{x},t)) - \epsilon^2\Delta\phi(\mathbf{x},t), && (4.2)\end{aligned}$$

where $\Omega \subset \mathbf{R}^d$ $(d=1,2,3)$.

$$\bar{\phi} = \frac{1}{|\Omega|}\int_\Omega \phi d\mathbf{x}. \tag{4.3}$$

Let $\mathcal{E}_{short}(\phi)$ be a short-range part of the free energy functional

$$\mathcal{E}_{short}(\phi) := \int_\Omega \left(F(\phi) + \frac{\epsilon^2}{2}|\nabla\phi|^2 \right) d\mathbf{x},$$

where $F(\phi)$ is the Helmholtz free energy and ϵ is a positive constant. In this paper, we use the free energy in the form of $F(\phi) = \frac{1}{4}(\phi^2-1)^2$.

This Chapter is organized as follows. In section 4.2, we describe the discrete scheme and its properties. We present the descretization of the proposed scheme in section 4.3. The numerical results showing the effects of a variable mobility are described in section 4.4. A discussion is presented in section 4.5.

4.2. The Cahn-Hilliard equation

We consider a block copolymer consisting of two homopolymer blocks A and B, each having the degree of polymerization N_A and N_B, respectively. We use the model proposed by Ohta and Kawasaki[124] for the block copolymer mesophase.

In the Ohta-Kawasaki model, the free energy of the block copolymers is described as a functional of the local segment density A, B, $\rho_A(\mathbf{x})$, $\rho_B(\mathbf{x})$ at point $\mathbf{x}$. We assume that the system is incompressible; i.e., $\rho_A(\mathbf{x})+\rho_B(\mathbf{x})$ is constant. The order parameter $\phi(\mathbf{x})$ is defined as the composition difference between the A and B components:

$$\phi(\mathbf{x}) = \frac{\rho_A(\mathbf{x})-\rho_B(\mathbf{x})}{\rho_A(\mathbf{x})+\rho_B(\mathbf{x})}. \tag{4.4}$$

Let c be sufficiently smooth and satisfy $\frac{\partial c}{\partial \mathbf{n}} = \frac{\partial \Delta c}{\partial \mathbf{n}} = 0$ on $\partial\Omega$. Then, we have that for all $v \in \dot{C}_0^\infty$,

$$\begin{aligned}
\left(\frac{\delta \mathcal{E}_{short}(\phi)}{\delta\phi}, v\right)_{L^2} &= \frac{d}{d\theta}\mathcal{E}_{short}(\phi+\theta v)\big|_{\theta=0} \\
&= \lim_{\theta\to 0}\frac{1}{\theta}\big(\mathcal{E}_{short}(\phi+\theta v) - \mathcal{E}_{short}(\phi)\big) && (4.5)\\
&= \int_\Omega \big(F'(\phi) - \epsilon^2\Delta\phi\big) v d\mathbf{x}. && (4.6)\\
\frac{\delta \mathcal{E}_{short}(\phi)}{\delta\phi} &= F'(\phi) - \epsilon^2\Delta\phi. && (4.7)
\end{aligned}$$

Let $\mathcal{E}_{long}(\phi)$ be a long-range part of the free energy functional.

$$\mathcal{E}_{long}(\phi) = \frac{\alpha}{2}\int_\Omega\int_\Omega G(\mathbf{x}-\mathbf{y})(\phi(\mathbf{x})-\bar{\phi})(\phi(\mathbf{y})-\bar{\phi})d\mathbf{y}d\mathbf{x}, \tag{4.8}$$

where G is the Green's function for the Laplace equation

$$\Delta_{\mathbf{x}} G(\mathbf{x}-\mathbf{y}) = -\delta(\mathbf{x}-\mathbf{y}). \tag{4.9}$$

Let c be sufficiently smooth and satisfy $\frac{\partial c}{\partial \mathbf{n}} = \frac{\partial \Delta c}{\partial \mathbf{n}} = 0$ on $\partial\Omega$. Then, we have that for all $v \in \dot{C}_0^\infty$,

$$\begin{aligned}
\left(\frac{\delta \mathcal{E}_{long}(\phi)}{\delta\phi}, v\right)_{L^2} &= \frac{d}{d\theta}\mathcal{E}_{long}(\phi+\theta v)\big|_{\theta=0} \\
&= \lim_{\theta\to 0}\frac{1}{\theta}\big(\mathcal{E}_{long}(\phi+\theta v) - \mathcal{E}_{long}(\phi)\big) && (4.10)\\
&= \frac{\alpha}{2}\int_\Omega\int_\Omega G(\mathbf{x}-\mathbf{y})(\phi(\mathbf{x})-\bar{\phi})v(\mathbf{x}')d\mathbf{y}d\mathbf{x} && (4.11)\\
&\quad +\frac{\alpha}{2}\int_\Omega\int_\Omega G(\mathbf{x}-\mathbf{y})(\phi(\mathbf{y})-\bar{\phi})v(\mathbf{x})d\mathbf{y}d\mathbf{x} && (4.12)\\
&= \int_\Omega\left[\alpha\int_\Omega G(\mathbf{x}-\mathbf{y})(\phi(\mathbf{y})-\bar{\phi})d\mathbf{y}\right]v(\mathbf{x})d\mathbf{x}. && (4.13)
\end{aligned}$$

Therefore,

$$\frac{\delta \mathcal{E}_{long}(\phi)}{\delta\phi} = \alpha\int_\Omega G(\mathbf{x}-\mathbf{y})(\phi(\mathbf{y})-\bar{\phi})d\mathbf{y}. \tag{4.14}$$

$$\begin{aligned}
\phi_t &= \Delta\mu = \Delta\left(\frac{\delta \mathcal{E}_{short}(\phi)}{\delta\phi} + \frac{\delta \mathcal{E}_{long}(\phi)}{\delta\phi}\right) && (4.15)\\
&= \Delta(F'(\phi) - \epsilon^2\Delta\phi) + \alpha\Delta\int_\Omega G(\mathbf{x}-\mathbf{y})(\phi(\mathbf{y})-\bar{\phi})d\mathbf{y} && (4.16)\\
&= \Delta(F'(\phi) - \epsilon^2\Delta\phi) + \alpha\int_\Omega \Delta_{\mathbf{x}} G(\mathbf{x}-\mathbf{y})(\phi(\mathbf{y})-\bar{\phi})d\mathbf{y} && (4.17)\\
&= \Delta(F'(\phi) - \epsilon^2\Delta\phi) - \alpha(\phi-\bar{\phi}). && (4.18)
\end{aligned}$$

4.3. Discretization f the proposed scheme

We present a semi-implicit time and centered difference space discretization of Eqs. (4.1) and (4.2).

$$\frac{\phi_{ij}^{n+1}-\phi_{ij}^{n}}{\Delta t} = \Delta_d \mu_{ij}^{n+\frac{1}{2}}, \tag{4.19}$$

$$\mu_{ij}^{n+\frac{1}{2}} = 2\phi_{ij}^{n+1} - \epsilon^2 \Delta_d \phi_{ij}^{n+1} + f(\phi_{ij}^{n}) - 2\phi_{ij}^{n}, \tag{4.20}$$

where $f(\phi) = F'(\phi)$.

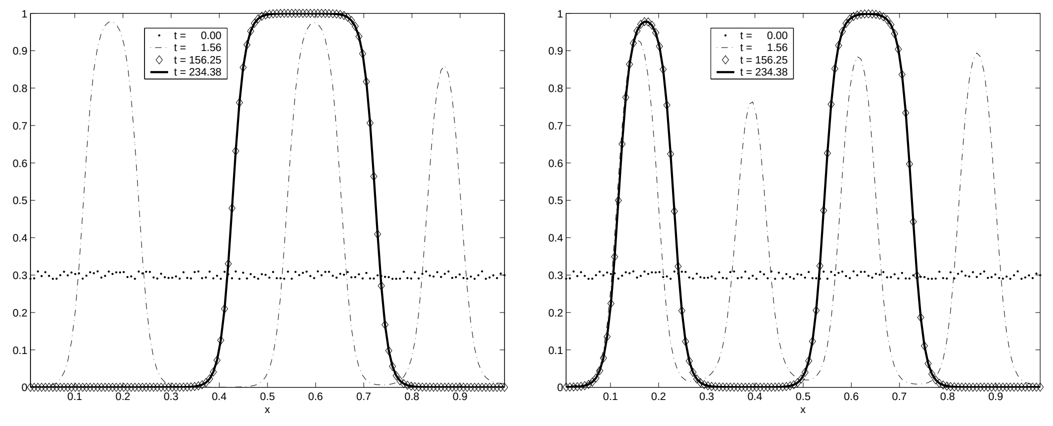
$$\phi_t = -\alpha(\phi - \bar{\phi}). \tag{4.21}$$

$$\frac{\phi^* - \phi^n}{\Delta t} = -\alpha(\phi^* - \bar{\phi}). \tag{4.22}$$

$$\phi^* = \bar{\phi} + (\phi^n - \bar{\phi})e^{-\alpha\Delta t}. \tag{4.23}$$

4.4. Numerical results

4.4.1. One space dimension. Now, we examine the evolution of a random distribution of initial concentration. We take $\epsilon = 0.009,\ h = 1/128,\ \Delta t = 0.2h,\ \Omega = (0, 1)$, and the initial state (dotted line) in Fig. 4.1 is taken to be $\phi(x, 0) = 0.3 + 0.01\text{rand}(\ )$. The random number, rand( ), is uniformly distributed between -1 and 1. Fig. 4.1 shows evolutions of the initial concentration $\phi(x, 0)$ with a constant mobility and a variable mobility from a random perturbation. Constant mobility case has only one big component, but the variable mobility case has two components.



$M(\phi) \equiv 0.25.$ $M(\phi) = \phi(1-\phi).$

FIGURE 4.1. Evolution of initial concentration $\phi(x, 0) = 0.3 + 0.01\text{rand}(\ )$.

4.4.2. Instability of perfectly oriented lamellae in compression and expansion. To study the stability of the compressed or expanded lamellae, we conducted the following simulation. We start from the initial condition

$$\phi_i^0 = A\cos(2\pi k x_i) + \text{noise term}$$

and solve Eqs. (4.19) and (4.20) numerically.

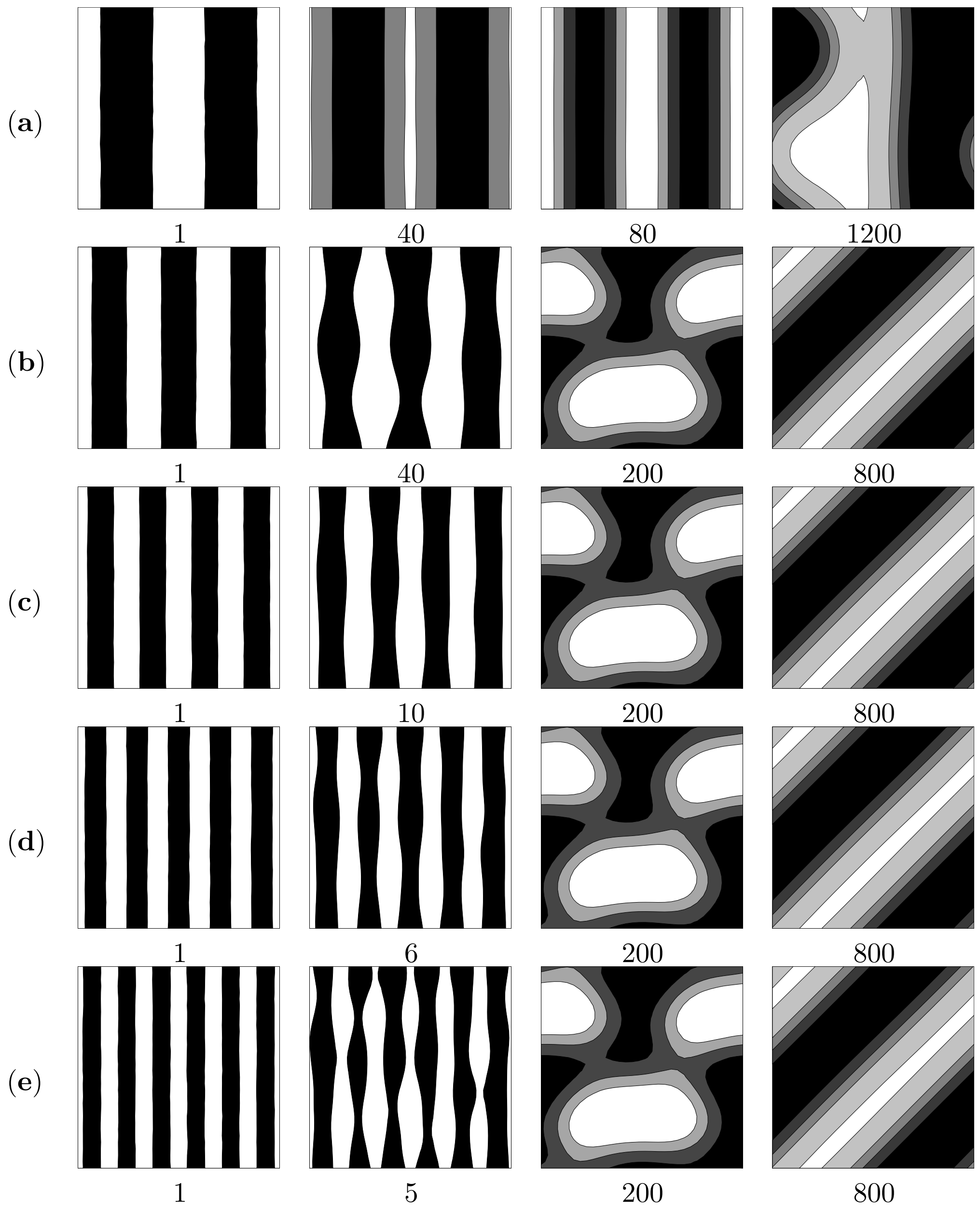


FIGURE 4.2. Relaxation of uniform lamellae of the size $L \times L = 32 \times 32$ with intial wavenumber $k = (a)2/L$, $(b) = 3/L$, $(c) = 4/L$, $(d) = 5/L$, $(e) = 6/L$. The numbers denotes the time steps.

The initial condition has two parameters A and k, but it was found that the results are almost independent of the initial amplitude A. This is because the amplitude is first optimized in a rather short time (10-30 time steps), and thereafter the relaxation of the periodicity takes place slowly. Because of the periodic boundary condition, the wave vectors can take only discrete values of k) n/L, where n is an integer. In the present simulation, we fixed the system size at L) 32 and prepare five different uniform lamellae with the wave vectors k) $2/L$, $3/L$, $4/L$, $5/L$, and $6/L$.

We take $\epsilon = 1.9298,\ h = 1.0,\ \Delta t = 1.0,\ \Omega = (0, 32) \times (0, 32)$, and the initial state in Fig. 4.2 is taken to be $\phi(x, 0) = \cos(2\pi kx) + 0.1\text{rand}(\ )$.

4.4.3. Linear stability theory. Next, to ensure that we are simulating the correct physical problem, we consider the agreement between the numerics and the results of a linear stability analysis about a constant concentration $\phi = \phi_m$. Accordingly, we look for a solution of Eqs. (4.1) and (4.2) of the form

$$\phi(x, t) = \phi_m + \sum_{k=1}^{\infty} \phi^k(t) \cos(k\pi x),$$

where $|\phi^k(t)| \ll 1$. Neglecting quadratic terms in $\phi^k(t)$, we find that $\phi^k(t)$ must solve the ordinary differential equation,

$$\frac{\mathrm{d}\phi^k}{\mathrm{d}t} = -(k\pi)^2[3{\phi_m}^2 - 1 + \epsilon^2(k\pi)^2 - \alpha]\phi^k. \tag{4.24}$$

The solution of Eq. (4.24) is

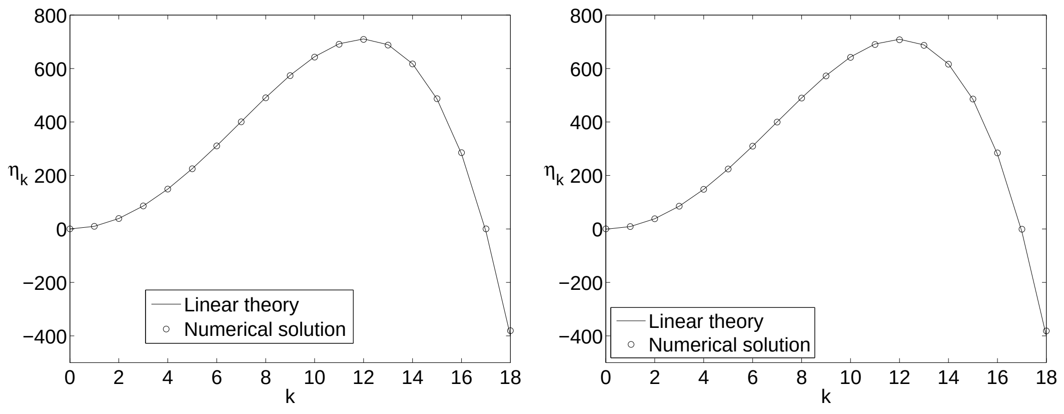
$$\phi^k(t) = \phi^k(0)\mathrm{e}^{\eta_k t},$$

where $\eta_k = -(k\pi)^2[3{\phi_m}^2 - 1 + \epsilon^2(k\pi)^2 - \alpha]$ is the growth rate.

In Fig. 4.3, the theoretical growth rate η_k is compared to that obtained from the nonlinear scheme. The numerical growth rate is defined by

$$\tilde{\eta_k} = \log\left(\frac{\max_{ij}|\phi_{ij}^n|}{\max_{ij}|\phi_{ij}^0|}\right)/t_n.$$

Here, we used $\phi_m = 0.0$, initial data $\phi_0(x) = 0.01\cos(k\pi x)$ and $\epsilon = 0.018757$, $\Delta t = 10^{-6}$, $h = 1/256$ and $t_n = 0.0001$. The graph shows that the linear analysis (solid line) and numerical solution (circle) are in good agreement.



(a) (b)

FIGURE 4.3. Growth rate for the different wave numbers k with (a) $\alpha = 0.0$ and (b) $\alpha = 1.0$.

4.5. Conclusions

We present details of the efficient computational scheme of the phase-field models for the block copolymer.

Chapter 5

Fast and automatic inpainting of binary images using a phase-field model

Image inpainting is the process of reconstructing lost or deteriorated parts of images using information from surrounding areas. We propose a computationally efficient and fast phase-field method which uses automatic switching parameter, adaptive time step, and automatic stopping of calculation. The algorithm is based on an energy functional. We demonstrate the performance of our new method and compare it with a previous method.

5.1. Introduction

Image inpainting [50, 47, 98] is the process of reconstructing lost or deteriorated parts of images using information from surrounding areas. Let $f(\mathbf{x})$, where $\mathbf{x} = (x, y)$, be a given image in a domain Ω. Let $c(\mathbf{x}, t)$ be a phase-field which is governed by the following modified Cahn-Hilliard (CH) equation [51]:

$$c_t = \Delta\mu + \lambda(\mathbf{x})(f(\mathbf{x}) - c), \tag{5.1}$$

$$\mu = F'(c) - \epsilon^2 \Delta c, \tag{5.2}$$

where $F(c) = 0.25c^2(1-c)^2$. In the examples considered here, we use binary images in which most of the pixels are either exactly black or white. Eqs. (5.1) and (5.2) are the modified CH equation, due to the added fidelity term $\lambda(\mathbf{x})(f(\mathbf{x}) - c)$ [48]. Image inpainting using phase-field methods is recently investigated by authors in [48, 49]. It is a good starting point for using partial differential equations in inpainting images, however, we found there are a couple of defects. First of all, switching parameter ϵ and stopping the calculation are done by trial and error. Furthermore, large time step Δt is more or less time step rescaling and it turns out that it is equivalent to using smaller time step than usual usage. In this Chapter, we propose a phase-field method which uses automatic varying ϵ, adaptive time step, and a stopping criterion based on energy functional.

The outline of this Chapter is the following. In Sec. 5.2, the discrete equations for the governing equations are presented. In Sec. 5.3, we present computational examples. We propose a new automatic controlled algorithm in Sec. 5.4. Finally, in Sec. 5.5, conclusions are drawn.

5.2. Discrete equations and a numerical solution

In this section, we present fully discrete schemes for the CH equation in two dimensional space, i.e., $\Omega = (a, b) \times (c, d)$.

Then, a semi-implicit time and centered difference space discretization of Eqs. (5.1) and (5.2) is

$$
\begin{aligned}
\frac{c_{ij}^{n+1} - c_{ij}^{n}}{\Delta t} &= \Delta_d \mu_{ij}^{n+\frac{1}{2}} + \lambda_{ij}(f_{ij} - c_{ij}^{n}), && (5.3)\\
\mu_{ij}^{n+\frac{1}{2}} &= \varphi(c_{ij}^{n+1}) - \frac{c^n}{4} - \epsilon^2 \Delta_d \, c_{ij}^{n+1}, && (5.4)\\
\text{where} \quad & \varphi(c) = F'(c) + \frac{c}{4}.
\end{aligned}
$$

We can rewrite Eqs. (5.3) and (5.4) as follows:

$$
\begin{aligned}
\frac{c_{ij}^{n+1} - c_{ij}^{n}}{\Delta t} &= \Delta_d \nu_{ij}^{n+1} - \frac{1}{4}\Delta_d c_{ij}^{n} + \lambda_{ij}(f_{ij} - c_{ij}^{n}), && (5.5)\\
\nu_{ij}^{n+1} &= \varphi(c_{ij}^{n+1}) - \epsilon^2 \Delta_d c_{ij}^{n+1}. && (5.6)
\end{aligned}
$$

We use nonlinear Full Approximation Storage (FAS) multigrid method to solve the nonlinear discrete system (5.5) and (5.6) at the implicit time level. The nonlinearity is treated using one step of Newton's iteration and a pointwise Gauss-Seidel relaxation scheme is used as the smoother in the multigrid method. See the reference text [122] for additional details and backgrounds.

5.3. Computational examples

In this section, we will compare the numerical scheme of the previous Bertozzi's paper [48] with our scheme. First we refer to the discrete Eq. (9) in the paper [48]

$$
\begin{aligned}
\frac{u^{n+1} - u^{n}}{\Delta t} &+ \varepsilon \Delta^2{}_d u^{n+1} - C_1 \Delta_d u^{n+1} + C_2 u^{n+1} && (5.7)\\
&= \Delta_d \left(\frac{1}{\varepsilon} \mathrm{W}'(u^n) \right) + \lambda(\mathbf{x})(f(x) - u^n) - C_1 \Delta_d u^n + C_2 u^n,
\end{aligned}
$$

where $W(u) = u^2(1-u)^2$ and the constants C_1 and C_2 are large enough so that the equation is convex for the range of u in the simulation. Next, we rewrite Eq. (5.7) as follows:

$$
\frac{u^{n+1} - u^{n}}{\frac{4\Delta t}{\varepsilon(C_2\Delta t + 1)}} = \Delta_d \left(\frac{1}{4}\mathrm{W}'(u^n) - \frac{\varepsilon^2}{4}\Delta_d u^{n+1} + \frac{\varepsilon}{4} C_1 (u^{n+1} - u^n) \right) + \frac{\varepsilon}{4}\lambda(\mathbf{x})(f(\mathbf{x}) - u^n).
$$

Table 5.1 shows that two schemes are equivalent.

TABLE 5.1. Equivalent forms of two schemes.

Bertozzi's numerical scheme
$\frac{u^{n+1}-u^n}{\frac{4\Delta t}{\varepsilon(C_2\Delta t+1)}} = \Delta_d(\frac{1}{4}\mathrm{W}'(u^n) - \frac{\varepsilon^2}{4}\Delta_d u^{n+1} + \frac{\varepsilon}{4}C_1(u^{n+1} - u^n)) + \frac{\varepsilon}{4}\lambda(\mathbf{x})(f(\mathbf{x}) - u^n)$
Our numerical scheme
$\frac{c^{n+1}-c^n}{\Delta t} = \Delta_d \left(F'(c^n) - \epsilon^2 \Delta_d c^{n+1} + \frac{1}{4}(c^{n+1} - c^n)\right) + \lambda(\mathbf{x})(f(\mathbf{x}) - c^n)$

We perform two test problems such as inpaintings of a double stripe and of a cross to show that two schemes are equivalent.

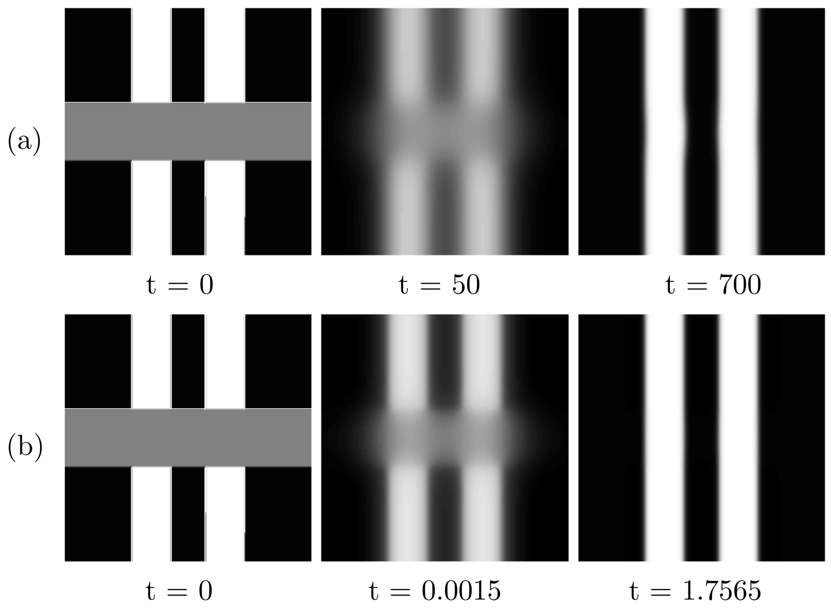


FIGURE 5.1. (a) Bertozzi's result. (b) Our result. Left column is initial data, middle column is results at iteration 50, and right column is results at iteration 700.

5.3.1. Inpainting of a double stripe. In this test problem, the computational domain $\Omega = (0, 1.28) \times (0, 1.28)$ and 128×128 mesh size are taken. The initial configurations are shown in the first column in Fig. 5.1. In the first and second rows, figures are results using the previous scheme and our proposed scheme, respectively. The gray region in the initial configuration denotes the inpainting region. In the previous scheme, we start calculations with a large $\epsilon = 0.8$ value, then switch its value to $\epsilon = 0.01$ after 50 iterations, and stop the calculation at 700 iterations. We repeat the same calculations with equivalent values which are summarized in Table 5.2. The prime notations of the parameters are from previous method and right arrow implies changing values when switching happens. By comparing two results, we see that our scheme is equivalent to the previous Bertozzi's scheme.

TABLE 5.2. Equivalent parameter values of two schemes.

Previous	Value	Current	Value
$\Delta t'$	1.0	$\Delta t = \dfrac{4\Delta t'}{\epsilon'(C_2'\Delta t' + 1)}$	$0.00003 \rightarrow 0.0027$
λ'	50000	$\lambda = \dfrac{\epsilon'}{4}\lambda'$	$10000 \rightarrow 125$
ϵ'	$0.8 \rightarrow 0.01$	$\epsilon = \dfrac{\epsilon'}{2}$	$0.4 \rightarrow 0.005$

5.3.2. Inpainting of a cross. For the second test problem, the initial configuration is a cross with an inpainting region as shown in the first column in Fig. 5.2.

The computational domain $\Omega = (0, 1.28) \times (0, 1.28)$ and 128×128 mesh size are taken. In the first and second rows, figures are results using the previous scheme and our proposed scheme, respectively. In the previous scheme, we start calculations with a large $\epsilon = 0.8$ value, switch its value to $\epsilon = 0.01$ after 300 iterations, and stop the calculation at 1000 iterations. We repeat the same calculations with equivalent values which are summarized in Table 5.3. By comparing two results, we see that our scheme is equivalent to the previous Bertozzi's scheme.

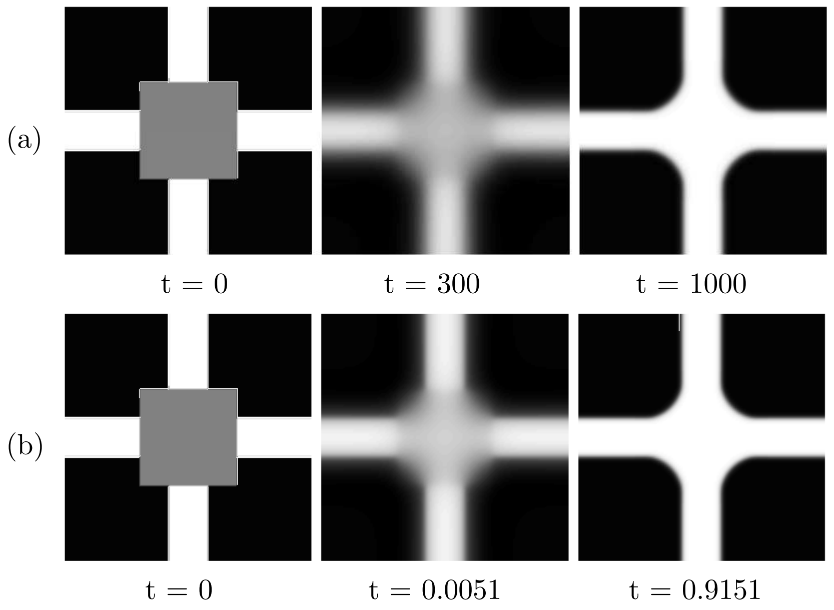


FIGURE 5.2. (a) Bertozzi's result. (b) Our result. Left column is initial data, middle column is results at iteration 300, and right column is results at iteration 1000.

TABLE 5.3. Equivalent parameter values of two schemes.

Previous	Value	Current	Value
$\Delta t'$	1.0	$\Delta t = \dfrac{4\Delta t'}{\epsilon'(C_2'\Delta t' + 1)}$	$0.000017 \rightarrow 0.0013$
λ'	100000	$\lambda = \dfrac{\epsilon'}{4}\lambda'$	$20000 \rightarrow 250$
ϵ'	$0.8 \rightarrow 0.01$	$\epsilon = \dfrac{\epsilon'}{2}$	$0.4 \rightarrow 0.005$

From these two test problems, we can conclude that two schemes are equivalent. However, in the previous algorithm, when to switch the parameters and when to stop the calculation are from trial and error. Therefore, it is our main purpose to propose an automatic switching and stopping algorithm based on an energy functional.

5.4. Proposed algorithm for automatic control

In this section, we propose an automatic switching and stopping algorithm based on an energy functional. Let us reconsider the first test problem which is the inpainting of a double stripe.

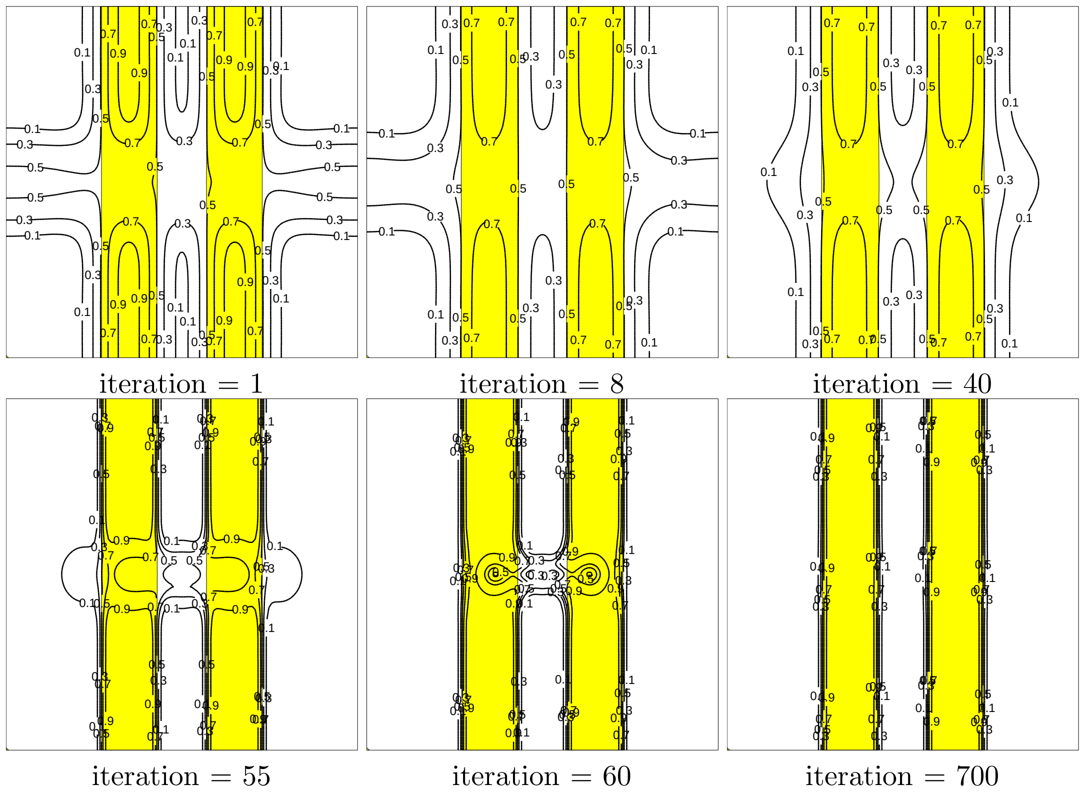


FIGURE 5.3. Temporal evolution of contour plots of the phase-field for the double stripe inpainting.

Fig. 5.3 shows the temporal evolution of contour plots of the phase-field. Around the switching time, we can observe the phase separation which means that the inpainting region separates into white and dark regions. Then we switched the ϵ parameter. Next, let us take a look at the time evolution of the energy functional. In Fig. 5.4, the energy is increased at the initial stage and it is decreased.

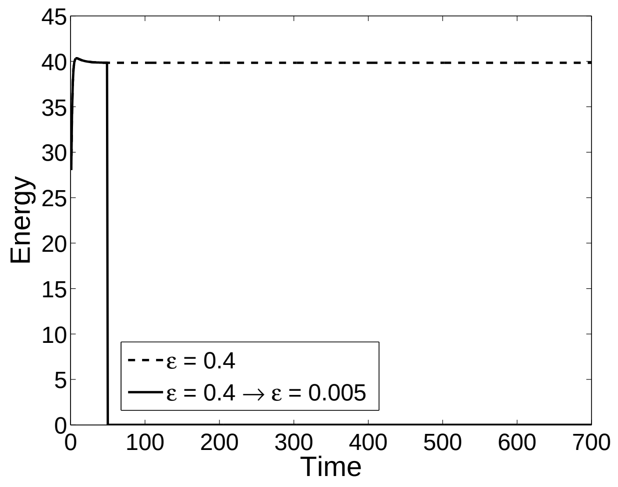


FIGURE 5.4. Temporal evolution of contour plots of the energy functional for the double stripe inpainting.

Similar phenomena in the second test problem which is the inpainting of a cross are observed. See the Figs. 5.6 and 5.5. Therefore, it is natural to monitor the energy functional for switching the parameter and stopping the calculation.

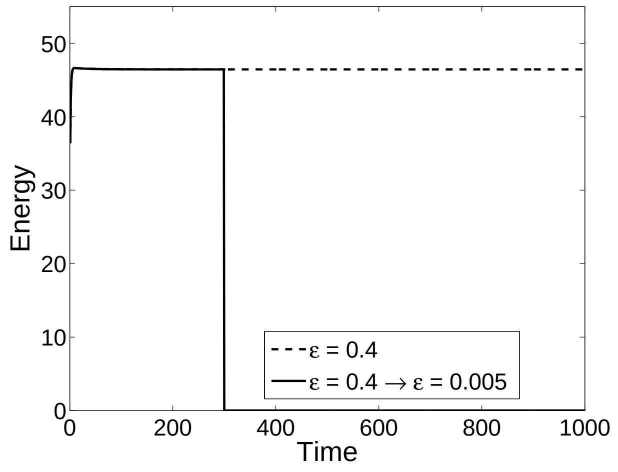


FIGURE 5.5. Temporal evolution of contour plots of the energy functional for the cross inpainting.

5.4.1. Inpainting of damaged images. Fig. 5.7(a) and (c) show the initial images of damaged double stripes and cross and Fig. 5.7(b) and (d) show the results with our proposed automatic algorithm to a double stripe and a cross inpainting problems. In the case of double stipes (see Fig. 5.7(a) and (b)), it only requires 16 iterations to recover the damaged images. Also in the other case (cross image, see Fig. 5.7(c) and (d)) we obtain the recovered image after 15 iterations. We use

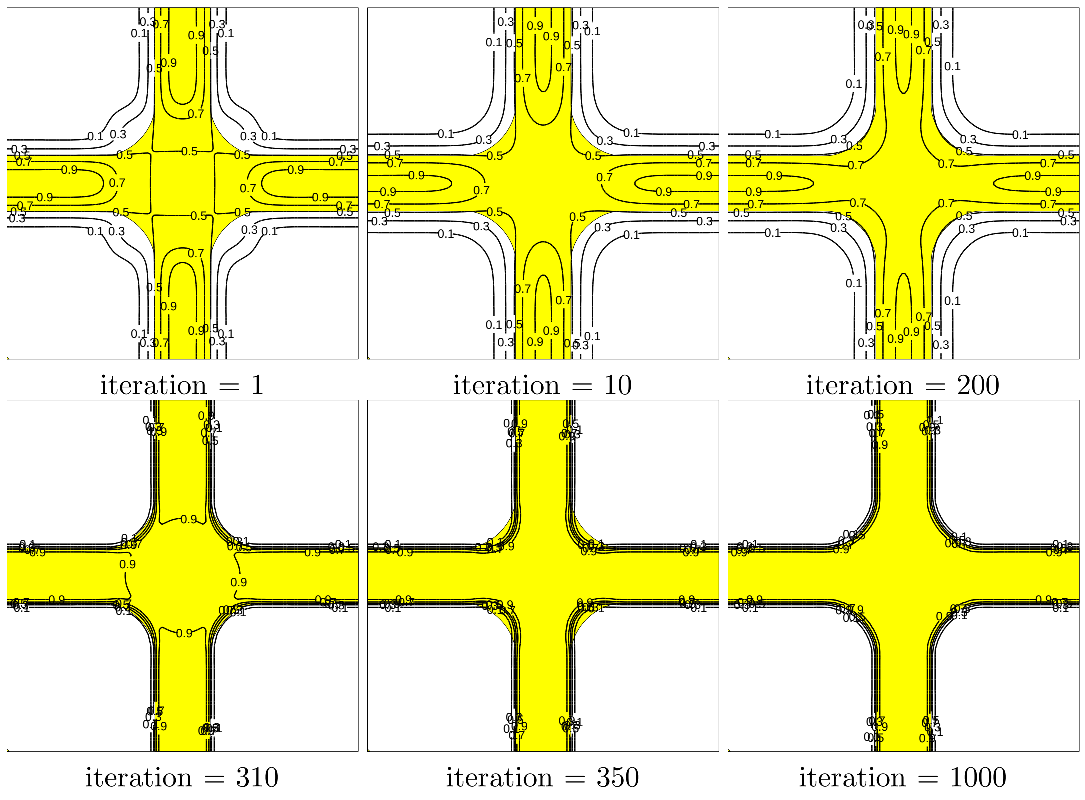


FIGURE 5.6. Temporal evolution of contour plots of the phase-field for the cross inpainting.

$\Delta t = 1/128,\ \epsilon = 0.038424, \lambda = 3/\Delta t$ and when $diff$,the difference of energies of c^{n+1} and c^n, is smaller than $tol1(= 0.08)$ is equal to 3 times, that is, at the number of iteration is 3 in both cases, we switch the parameter as $\Delta t' = 2.0\Delta t,\ \epsilon' = 0.0875\epsilon,\ \lambda' = 1.8/\Delta t'$. When $diff$ is smaller than prescribed tolerance, $tol2$ $(= 1.0E - 6)$, we stop this algorithm.

5.4.2. Inpainting of obscured text. In order to recover obscured text (see Fig. 5.8(a)), we use our inpainting algorithm. Fig. 5.8(a) shows the initial image which is obscured text by lines and Fig. 5.8(b) shows the recover result image. We use the parameter as $\Delta t = 1/128,\ \epsilon = 0.038424,\ \lambda = 3/\Delta t$ and $diff$ is smaller than $tol1$ $(= 0.08)$ is equal to 3 times, that is, at the number of iteration is 4 in this case, we switch the parameter as $\Delta t' = 1.8\Delta t,\ \epsilon' = 0.0875\epsilon,\ \lambda' = 1.8/\Delta t'$. Our automatic switching method of the modified CH equation is faster than the previous model.

5.5. Conclusions

We have shown that our automatic switching algorithm achieves faster inpainting of binary images than the previous trial and error algorithm. Therefore, inpainting region is reconstructed more efficiently and faster than previous method. The developed

The proposed automatic switching and stopping algorithm is as follows.

```
Algorithm:
Given a maximum iteration number $N$, tolerances $tol1$ and $tol2$
• Set $k = 1$, $flag = 0$.
• While ($k \leq N$) do Steps 1-4
    Step 1   Compute $c^{n+1}$ from $c^n$ by solving Eqs. (5.5) and (5.6).
    Step 2   Check the difference of energies of $c^n$ and $c^{n+1}$
             $diff = |\mathcal{E}(c^n) - \mathcal{E}(c^{n+1})|$
             If ($flag < 3$ and $diff < tol1$)
                 $flag = flag + 1$
             End
    Step 3   Switch parameters and reset data
             If ($flag = 3$)
                 If ($c^{n+1} > 0.5$)
                     $c^{n+1} = 1$
                 Else
                     $c^{n+1} = 0$
                 End
                 $\Delta t = 2\Delta t$
                 do Step 1 twice
                 $\epsilon = 0.0875\epsilon$
                 $\lambda = 1.8/\Delta t$
             End
    Step 4   Stop loop
             If ($diff < tol2$ and $flag = 3$)
                 Stop loop
             End
  End
```

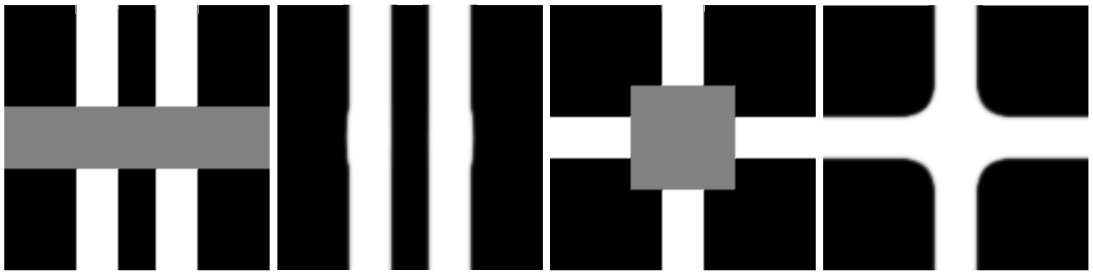

(a) Initial (b) 16 iterations (c) Initial (d) 15 iterations

FIGURE 5.7. Recovery of damaged images. The computational domain is $\Omega = (0, 1.28) \times (0, 1.28)$ and mesh size is 128×128.

automatic algorithm can be applied to calculating option pricing such as the Black-Scholes equations accurately and efficiently.

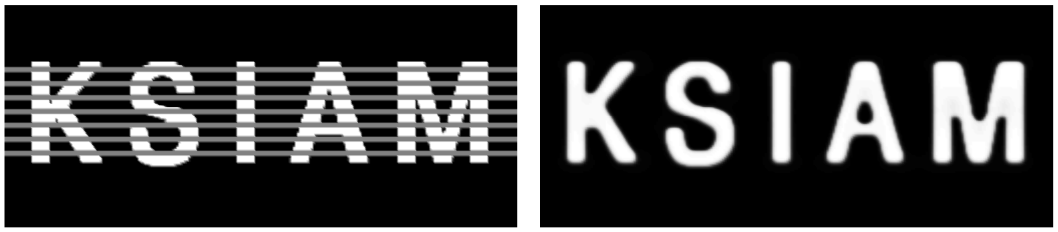

(a) Initial (b) 47 iterations

Figure 5.8. Recovery of damaged text. The computational domain is $\Omega = (0, 2.56) \times (0, 1.28)$ and mesh size is 256×128.

This chapter is published in J. KSIAM Vol.13, No. 3, 225-236, (2009).

Chapter 6

A Crank-Nicolson scheme for the Landau-Lifshitz equation without damping

An accurate and efficient numerical approach, based on a finite difference method with Crank-Nicolson time stepping, is proposed for the Landau-Lifshitz equation without damping. The phenomenological Landau-Lifshitz equation describes the dynamics of ferromagnetism. The Crank-Nicolson method is very popular in the numerical schemes for parabolic equations since it is second-order accurate in time. Although widely used, the method does not always produce accurate results when it is applied to the Landau-Lifshitz equation. The objective of this article is to enumerate the problems and then to propose an accurate and robust numerical solution algorithm. A discrete scheme and a numerical solution algorithm for the Landau-Lifshitz equation are described. A nonlinear multigrid method is used for handling the nonlinearities of the resulting discrete system of equations at each time step. We show numerically that the proposed scheme has a second-order convergence in space and time.

6.1. Introduction

The relaxation process of the magnetization distribution in a ferromagnetic material is described by the Landau-Lifshitz (LL) equation [116, 60]. Numerical analysis has played an important role in the investigation of various issues in ferromagnetic materials [85, 54, 57, 58]. Recent developments in modeling, analysis, and numerics of ferromagnetism were discussed in survey articles [55, 59]. In this paper, we consider the gyromagnetic term in the Landau-Lifshitz equation with a forcing

$$\frac{\partial \mathbf{m}(\mathbf{x},t)}{\partial t} = -\mathbf{m}(\mathbf{x},t) \times \Delta \mathbf{m}(\mathbf{x},t) + \mathbf{f}(\mathbf{x},t), \;\; \mathbf{x} \in \Omega \;\; 0 < t \leq T, \tag{6.1}$$

where $\mathbf{m}(\mathbf{x},t) = (u(\mathbf{x},t), v(\mathbf{x},t), w(\mathbf{x},t))$ is a magnetization vector field and $\Omega \subset \mathbb{R}^d$ $(d = 1, 2, 3)$ is a domain. At the domain boundary $\partial\Omega$, we will use either homogeneous Neumann or periodic boundary condition. It is obvious that the Eq. (6.1) with $\mathbf{f} \equiv 0$ has a length-preserving property during the evolution process. To see this, we do scalar multiplication of Eq. (6.1) with $\mathbf{m}$.

$$\frac{\partial \mathbf{m}}{\partial t} \cdot \mathbf{m} = -(\mathbf{m} \times \Delta \mathbf{m}) \cdot \mathbf{m} = 0. \tag{6.2}$$

Then, $\partial|\mathbf{m}|^2/\partial t = 0$, which implies $|\mathbf{m}(\mathbf{x},t)|$ is constant for all t and each $\mathbf{x}$. And we assume that $|\mathbf{m}(\mathbf{x},0)| = 1$. Let $E(\mathbf{m}(\mathbf{x},t))$ be an energy defined by $E(\mathbf{m}(t)) := \|\nabla \mathbf{m}(t)\|^2_{L^2(\Omega)}$. By taking an inner product of Eq. (6.1) with $\Delta\mathbf{m}$, we obtain

$$\frac{\partial \mathbf{m}}{\partial t} \cdot \Delta \mathbf{m} = -(\mathbf{m} \times \Delta \mathbf{m}) \cdot \Delta \mathbf{m} = 0. \tag{6.3}$$

Using homogeneous Neumann or periodic boundary conditions, from Eq. (6.3) we have

$$\begin{aligned} 0 &= \int_\Omega \frac{\partial \mathbf{m}}{\partial t} \cdot \Delta \mathbf{m} d\mathbf{x} = \int_{\partial\Omega} \frac{\partial \mathbf{m}}{\partial t} \cdot \frac{\partial \mathbf{m}}{\partial \mathbf{n}} ds - \int_\Omega \nabla \frac{\partial \mathbf{m}}{\partial t} : \nabla \mathbf{m} d\mathbf{x} \\ &= -\frac{1}{2} \frac{dE(\mathbf{m}(t))}{dt}, \end{aligned} \tag{6.4}$$

which implies that $E(\mathbf{m}(t))$ is constant and this problem has an energy conservation property. Here, $\mathbf{n}$ is a unit normal vector to $\partial\Omega$ and the operator ':' is defined as $A : B = \sum_{ij} a_{ij} b_{ij}$.

The Crank-Nicolson (CN) scheme is a popular implicit method for solving partial differential equations with second-order accuracy in time and space [61]. However, the method does not always produce accurate results when it is applied to the Landau-Lifshitz equation. It is the objective of this article to enumerate the problems and then to propose an accurate and robust numerical solution algorithm. One and two dimensional discrete schemes for the discretized Landau-Lifshitz equation with an exact solution are described and numerically solved using a nonlinear multigrid method. Also, we show the proposed scheme has a second-order convergence in space and time numerically.

The Chapter is organized as follows. In section 6.2, we describe the discrete scheme of Landau-Lifshitz equation. And we present the nonlinear multigrid method for the discrete system. In section 6.3, the numerical results showing performance of the proposed scheme are given. Conclusions are made in section 6.4.

6.2. Crank-Nicolson method

6.2.1. Discretization. For simplicity of presentation, we will describe spatial and temporal discretizations of the governing equation in one dimensional space. Two and three dimensional spaces are straight forward extensions. Let us first discretize the given computational domain $\Omega = (0, 1)$ as a uniform grid with the number of grid points N_x, a space step $h = 1/N_x$, and a time step $\Delta t = T/N_t$. Let us denote the numerical approximation of the solution by

$$\begin{aligned} \mathbf{m}_i^n &= \mathbf{m}(x_i, t^n) = (u_i^n,\ v_i^n,\ w_i^n) \\ &= (u((i-0.5)h, n\Delta t),\ v((i-0.5)h, n\Delta t),\ w((i-0.5)h, n\Delta t)), \end{aligned}$$

where $i = 1, \cdots, N_x$ and $n = 0, 1, \cdots, N_t$. We use a cell centered discretization. See Fig. 6.1.

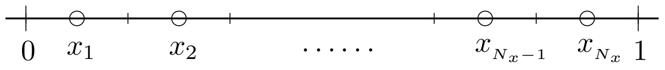


FIGURE 6.1. Cell centered grid

Neumann boundary condition is $\mathbf{m}_x(0, t) = \mathbf{m}_x(1, t) = 0$. Therefore we put the $\mathbf{m}_0 = \mathbf{m}_1$ and $\mathbf{m}_{N_x+1} = \mathbf{m}_{N_x}$ as the boundary condition. The Crank-Nicolson scheme is given as

$$\frac{\mathbf{m}^{n+1} - \mathbf{m}^n}{\Delta t} = -\frac{\mathbf{m}^{n+1} + \mathbf{m}^n}{2} \times \Delta_h \frac{\mathbf{m}^{n+1} + \mathbf{m}^n}{2} + \mathbf{f}^{n+\frac{1}{2}}, \tag{6.5}$$

where Δ_h is the standard discretization of Δ:

$$\Delta_h \mathbf{m}_i = \frac{1}{h^2}(\mathbf{m}_{i+1} - 2\mathbf{m}_i + \mathbf{m}_{i-1}).$$

Let $E_h(\mathbf{m}^n)$ be the discrete energy defined by

$$\begin{aligned} E_h(\mathbf{m}^n) &= \frac{1}{h}\sum_{i=1}^{N_x}\left[\left(u_{i+1}^n - u_i^n\right)^2 + \left(v_{i+1}^n - v_i^n\right)^2 + \left(w_{i+1}^n - w_i^n\right)^2\right] \\ &= \frac{1}{h}\sum_{i=1}^{N_x}|\mathbf{m}_{i+1}^n - \mathbf{m}_i^n|^2. \end{aligned}$$

We note that Cimrák in [56] established a weak convergence of the approximate solutions to weak solutions of the Landau-Lifshitz equation.

6.2.2. Properties of the scheme. First, we show that the scheme (6.5) conserves the magnitude of magnetization.

$$\frac{\mathbf{m}^{n+1} - \mathbf{m}^n}{\Delta t} = -\mathbf{m}^{n+\frac{1}{2}} \times \Delta_h \mathbf{m}^{n+\frac{1}{2}}. \tag{6.6}$$

Taking an inner product of Eq. (6.6) with $\mathbf{m}^{n+1} + \mathbf{m}^n$, we obtain

$$\frac{\mathbf{m}^{n+1} - \mathbf{m}^n}{\Delta t} \cdot (\mathbf{m}^{n+1} + \mathbf{m}^n) = -\left(\mathbf{m}^{n+\frac{1}{2}} \times \Delta_h \mathbf{m}^{n+\frac{1}{2}}\right) \cdot (\mathbf{m}^{n+1} + \mathbf{m}^n) = 0,$$

hence

$$|\mathbf{m}^{n+1}|^2 = |\mathbf{m}^n|^2. \tag{6.7}$$

Second, we show the discrete energy is conserved. Forming an inner product between Eq. (6.6) and $\Delta_h(\mathbf{m}^{n+1} + \mathbf{m}^n)$, we obtain

$$\frac{\mathbf{m}^{n+1} - \mathbf{m}^n}{\Delta t} \cdot \Delta_h(\mathbf{m}^{n+1} + \mathbf{m}^n) = -\left(\mathbf{m}^{n+\frac{1}{2}} \times \Delta_h \mathbf{m}^{n+\frac{1}{2}}\right) \cdot \Delta_h(\mathbf{m}^{n+1} + \mathbf{m}^n) = 0.$$

It follows that

$$(\mathbf{m}^{n+1} - \mathbf{m}^n) \cdot \Delta_h(\mathbf{m}^{n+1} + \mathbf{m}^n) = 0. \tag{6.8}$$

Summation Eq. (6.8) over $i = 1, \cdots, N_x$ leads to

$$\sum_{i=1}^{N_x}(\mathbf{m}_i^{n+1} - \mathbf{m}_i^n) \cdot \Delta_h(\mathbf{m}_i^{n+1} + \mathbf{m}_i^n) = 0. \tag{6.9}$$

Using Eq. (6.7) and periodic boundary condition, Eq. (6.9) becomes

$$\sum_{i=1}^{Nx}\left(2\mathbf{m}_i^{n+1} \cdot \mathbf{m}_{i+1}^{n+1} - 2\mathbf{m}_i^n \cdot \mathbf{m}_{i+1}^n\right) = 0. \tag{6.10}$$

Now, we get the following energy conservation:

$$\begin{aligned}
&E(\mathbf{m}^{n+1}) - E(\mathbf{m}^n) \\
&= \frac{1}{h}\sum_{i=1}^{Nx}\left(|\mathbf{m}_{i+1}^{n+1} - \mathbf{m}_i^{n+1}|^2 - |\mathbf{m}_{i+1}^n - \mathbf{m}_i^n|^2\right) \\
&= \frac{1}{h}\sum_{i=1}^{Nx}\left(|\mathbf{m}_{i+1}^{n+1}|^2 - |\mathbf{m}_{i+1}^n|^2 + |\mathbf{m}_i^{n+1}|^2 - |\mathbf{m}_i^n|^2 - 2\mathbf{m}_i^{n+1}\cdot\mathbf{m}_{i+1}^{n+1} + 2\mathbf{m}_i^n\cdot\mathbf{m}_{i+1}^n\right) \\
&= 0,
\end{aligned}$$

where we have used Eqs. (6.7) and (6.10).

Finally, we show that the truncation error of the scheme is second order in time and space. Let u, v, w be the exact solution of the partial differential equation (6.1) without a forcing term. Then, the local truncation error of the first component of the equations is

$$\begin{aligned}
T_i^{n+\frac{1}{2}} \quad = \quad & \frac{u_i^{n+1} - u_i^n}{\Delta t} + \frac{v_i^{n+1} + v_i^n}{2}\Delta_h\left(\frac{w_i^{n+1} + w_i^n}{2}\right) \\
& -\frac{w_i^{n+1} + w_i^n}{2}\Delta_h\left(\frac{v_i^{n+1} + v_i^n}{2}\right) \\
& = (u_t)_i^{n+\frac{1}{2}} + O(\Delta t^2) + (v_i^{n+\frac{1}{2}} + O(\Delta t^2))((w_{xx})_i^{n+\frac{1}{2}} + O(h^2) + O(\Delta t^2)) \\
& -(w_i^{n+\frac{1}{2}} + O(\Delta t^2))((v_{xx})_i^{n+\frac{1}{2}} + O(h^2) + O(\Delta t^2)) \\
& = (u_t + vw_{xx} - wv_{xx})_i^{n+\frac{1}{2}} + O(h^2) + O(\Delta t^2).
\end{aligned}$$

Since u, v, w is the solution of the differential equation so

$$(u_t + vw_{xx} - wv_{xx})_i^{n+\frac{1}{2}} = 0.$$

Therefore the principal part of the local truncation error is

$$T_i^{n+\frac{1}{2}} = O(h^2) + O(\Delta t^2).$$

For the second and third components of the equations, we get same results.

6.2.3. Solving the nonlinear system - a nonlinear multigrid method. Multigrid methods are generally accepted as among the fastest numerical methods for solving this type of partial differential equations. Since the scheme (6.5) is nonlinear, we use a nonlinear full approximation storage (FAS) multigrid method to solve the nonlinear discrete system (6.5) at the implicit time level. A pointwise Gauss-Seidel relaxation scheme is used as the smoother in the multigrid method. See the reference text [122] for additional details and background. The algorithm of the nonlinear multigrid method for solving the discrete equation is : First, let us rewrite Eq. (6.5) as

$$N(\mathbf{m}^{n+1}) = \boldsymbol{\phi}^n, \tag{6.11}$$

where the nonlinear system operator (N) is defined as

$$N(\mathbf{m}^{n+1}) = \mathbf{m}^{n+1} + \frac{\Delta t}{2}(\mathbf{m}\times\Delta_h\mathbf{m})^{n+1}$$

and the source term is

$$\boldsymbol{\phi}^n = \mathbf{m}^n - \frac{\Delta t}{2}(\mathbf{m} \times \Delta_h \mathbf{m})^n + \Delta t \mathbf{f}^{n+\frac{1}{2}}.$$

In the following description of one FAS cycle, we assume a sequence of grids Ω_k (Ω_{k-1} is coarser than Ω_k by factor 2). Given the number β of pre- and post-smoothing relaxation sweeps, an iteration step for the nonlinear multigrid method using the V-cycle is formally written as follows [122]:

FAS multigrid cycle

$$\mathbf{m}^{m+1,k} = FAScycle(k, \mathbf{m}^{m,k}, N_k, \boldsymbol{\phi}^{n,k}, \beta).$$

That is, $\mathbf{m}^{m,k}$ and $\mathbf{m}^{m+1,k}$ are the approximation of $\mathbf{m}^{n+1,k}$ before and after an FAS-cycle. If Ω_k is the finest mesh and $\|\mathbf{m}^{m+1,k} - \mathbf{m}^{m,k}\|_\infty <$ tol , then we let $\mathbf{m}^{n+1} = \mathbf{m}^{m+1,k}$. Now, we define the FAScycle.

Step 1) Presmoothing

$$\bar{\mathbf{m}}^{m,k} = SMOOTH^{\beta}(\mathbf{m}^{m,k}, N_k, \boldsymbol{\phi}^{n,k}), \tag{6.12}$$

which means performing β smoothing steps with the initial approximation $\mathbf{m}^{m,k}$, source term $\boldsymbol{\phi}^{n,k}$, and $SMOOTH$ relaxation operator to get the approximation $\bar{\mathbf{m}}^{m,k}$. In its component form, Eq. (6.11) becomes

$$\begin{pmatrix} u_i^{n+1} \\ v_i^{n+1} \\ w_i^{n+1} \end{pmatrix} + \frac{\Delta t}{2} \begin{pmatrix} v_i \Delta_h w_i - w_i \Delta_h v_i \\ w_i \Delta_h u_i - u_i \Delta_h w_i \\ u_i \Delta_h v_i - v_i \Delta_h u_i \end{pmatrix}^{n+1} = \boldsymbol{\phi}_i^n \quad \text{for } i = 1, \cdots, N_x. \tag{6.13}$$

The main idea of the proposed scheme is a cancelation. Note that

$$\begin{aligned} v_i \Delta_h w_i - w_i \Delta_h v_i &= v_i \frac{w_{i-1} - 2w_i + w_{i+1}}{h^2} - w_i \frac{v_{i-1} - 2v_i + v_{i+1}}{h^2} \\ &= v_i \frac{w_{i-1} + w_{i+1}}{h^2} - w_i \frac{v_{i-1} + v_{i+1}}{h^2} \\ &= v_i \tilde{\Delta}_h w_i - w_i \tilde{\Delta}_h v_i \text{ for } i = 1, \cdots, N_x, \end{aligned} \tag{6.14}$$

where $\tilde{\Delta}_h w_i = (w_{i-1} + w_{i+1})/h^2$. Similarly, we have

$$w_i \Delta_h u_i - u_i \Delta_h w_i = w_i \tilde{\Delta}_h u_i - u_i \tilde{\Delta}_h w_i, \tag{6.15}$$

$$u_i \Delta_h v_i - v_i \Delta_h u_i = u_i \tilde{\Delta}_h v_i - v_i \tilde{\Delta}_h u_i. \tag{6.16}$$

This cancelation stabilizes the scheme. By Eqs. (6.14), (6.15), and (6.16) we rewrite the above equation.

$$A_i \begin{pmatrix} u_i^{n+1} \\ v_i^{n+1} \\ w_i^{n+1} \end{pmatrix} = \begin{pmatrix} \alpha_i \\ \beta_i \\ \gamma_i \end{pmatrix},$$

where

$$A_i = \begin{pmatrix} 1 & \frac{\Delta t}{2} \tilde{\Delta}_h w_i^{n+1} & -\frac{\Delta t}{2} \tilde{\Delta}_h v_i^{n+1} \\ -\frac{\Delta t}{2} \tilde{\Delta}_h w_i^{n+1} & 1 & \frac{\Delta t}{2} \tilde{\Delta}_h u_i^{n+1} \\ \frac{\Delta t}{2} \tilde{\Delta}_h v_i^{n+1} & -\frac{\Delta t}{2} \tilde{\Delta}_h u_i^{n+1} & 1 \end{pmatrix} = \begin{pmatrix} 1 & c & -b \\ -c & 1 & a \\ b & -a & 1 \end{pmatrix}$$

and $(\alpha_i,\ \beta_i,\ \gamma_i)^T$ is the right hand side term in Eq. (6.13). Then using Cramer's rule, we obtain

$$(u_i^{n+1},\ v_i^{n+1},\ w_i^{n+1}) = 1/|A_i|(|A_{i,1}|,\ |A_{i,2}|,\ |A_{i,3}|),\ i = 1, \cdots, N_x$$

where $A_{i,j}$ is obtained by replacing the jth column of A_i with $(\alpha_i,\ \beta_i,\ \gamma_i)^T$.

$$\begin{aligned}
|A_i| &= 1 + a^2 + b^2 + c^2,\\
|A_{i,1}| &= \alpha_i(1 + a^2) - \beta_i(c - ab) + \gamma_i(ac + b),\\
|A_{i,2}| &= \alpha_i(ab + c) + \beta_i(1 + b^2) - \gamma_i(a - bc),\\
|A_{i,3}| &= \alpha_i(ac - b) + \beta_i(a + bc) + \gamma_i(1 + c^2).
\end{aligned}$$

We can rewrite Eq. (6.11) as a matrix form:

$$\begin{pmatrix} 1 & c & -b \\ -c & 1 & a \\ b & -a & 1 \end{pmatrix} \mathbf{m}_i^{n+1} = \boldsymbol{\phi}_i^n, \tag{6.17}$$

$$\text{where } a = \frac{\Delta t}{2}\tilde{\Delta}_h u_i^{n+1},\ b = \frac{\Delta t}{2}\tilde{\Delta}_h v_i^{n+1},\ \text{and } c = \frac{\Delta t}{2}\tilde{\Delta}_h w_i^{n+1}.$$

To derive a Gauss-Seidel type iteration, we replace $\mathbf{m}_\alpha^{n+1}$ in the Eq. (6.17) with $\bar{\mathbf{m}}_\alpha^{m,k}$ if $\alpha \le i$, otherwise with $\mathbf{m}_\alpha^{m,k}$, i.e.,

$$\begin{pmatrix} 1 & c & -b \\ -c & 1 & a \\ b & -a & 1 \end{pmatrix} \bar{\mathbf{m}}_i^{m,k} = \boldsymbol{\phi}_i^n, \tag{6.18}$$

where

$$a = \frac{\Delta t}{2}\frac{\bar{u}_{i-1,j}^{m,k} + u_{i+1,j}^{m,k}}{h^2},\ b = \frac{\Delta t}{2}\frac{\bar{v}_{i-1,j}^{m,k} + v_{i+1,j}^{m,k}}{h^2},\ c = \frac{\Delta t}{2}\frac{\bar{w}_{i-1,j}^{m,k} + w_{i+1,j}^{m,k}}{h^2}.$$

Step 2) Coarse grid correction

- Compute the defect: $\bar{\boldsymbol{d}}^{m,k} = \boldsymbol{\phi}^{n,k} - N_k(\bar{\mathbf{m}}^{m,k})$.
- Restrict the defect and $\bar{\mathbf{m}}^{m,k}$: $\bar{\boldsymbol{d}}^{m,k-1} = I_k^{k-1}(\bar{\boldsymbol{d}}^{m,k}),\ \bar{\mathbf{m}}^{m,k-1} = I_k^{k-1}(\bar{\mathbf{m}}^{m,k})$.

The restriction operator I_k^{k-1} maps k-level functions to $(k-1)$-level functions.

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

- Compute the right-hand side: $\boldsymbol{\phi}^{n,k-1} = \bar{\boldsymbol{d}}^{m,k-1} + N_{k-1}(\bar{\mathbf{m}}^{m,k-1})$.
- Compute an approximate solution $\hat{\mathbf{m}}^{m,k-1}$ of the coarse grid equation on Ω_{k-1}, i.e.

$$N_{k-1}(\mathbf{m}^{m,k-1}) = \boldsymbol{\phi}^{n,k-1}. \tag{6.19}$$

If $k = 1$, we apply the smoothing procedure in (6.12) to obtain the approximate solution. If $k > 1$, we solve (6.19) by performing a FAS k-grid cycle using $\bar{\mathbf{m}}^{m,k-1}$ as an initial approximation:

$$\hat{\mathbf{m}}^{m,k-1} = \text{FAScycle}(k-1, \bar{\mathbf{m}}^{m,k-1}, N_{k-1}, \boldsymbol{\phi}^{n,k-1}, \beta).$$

• Compute the coarse grid correction (CGC): $\hat{\boldsymbol{v}}^{m,k-1} = \hat{\mathbf{m}}^{m,k-1} - \bar{\mathbf{m}}^{m,k-1}$.
• Interpolate the correction: $\hat{\boldsymbol{v}}^{m,k} = I^k_{k-1}\hat{\boldsymbol{v}}^{m,k-1}$.

• Compute the corrected approximation on Ω_k: $\mathbf{m}^{m,\text{after } CGC,k} = \bar{\mathbf{m}}^{m,k} + \hat{\boldsymbol{v}}^{m,k}$. *Step 3) Postsmoothing*: $\mathbf{m}^{m+1,k} = SMOOTH^{\beta}(\mathbf{m}^{m, \text{ after } CGC,k}, N_k, \boldsymbol{\phi}^{n,k})$. This completes the description of a nonlinear FAScycle. After we get a solution after one FAScycle, using an updated source term, we repeatedly perform iterations until the numerical solution converges.

Now we consider the scheme with no cancelation. We rewrite the equation (6.13).

$$A_i \begin{pmatrix} u_i^{n+1} \\ v_i^{n+1} \\ w_i^{n+1} \end{pmatrix} = \boldsymbol{\phi}_i^n \quad \text{for } i = 1, \cdots, N_x.$$

To discuss the stability of the Crank-Nicolson scheme, we compute the charateristic polynomial of A_i.

$$\det(A_i - \lambda I) = (1-\lambda)^3 + (1-\lambda)(a^2+b^2+c^2).$$

The three eigenvalues of A_i are

$$\lambda_1 = 1,\ \lambda_2 = 1 + i\sqrt{a^2+b^2+c^2}, \text{ and } \lambda_3 = 1 - i\sqrt{a^2+b^2+c^2}.$$

Thus, the three eigenvalues of A_i^{-1} are

$$\gamma_1 = 1/\lambda_1,\ \gamma_2 = 1/\lambda_2, \text{ and } \gamma_3 = 1/\lambda_3.$$

The absolute values of three eigenvalues of A_i^{-1} are

$$|\gamma_1| = 1,\ |\gamma_2| = |\gamma_3| = \frac{1}{\sqrt{1+a^2+b^2+c^2}} \le 1.$$

Without cancelation, a, b, and c are small compared to 1, on the other hand, with cancelation $a^2+b^2+c^2 \approx O(1/h^4)$. Therefore, $1/\sqrt{1+a^2+b^2+c^2} \approx O(h^2) \ll 1$ and this makes the iterations stable.

6.3. Numerical results

In this section we perform numerical experiments with exact solutions to verify the second-order accuracy of the proposed scheme in time and space. Without the forcing term, we also show the energy conservation property.

6.3.1. One space dimension.

6.3.1.1. *Convergence test.* We consider one-dimensional Landau-Lifshitz equation with a source:

$$\mathbf{m}_t = -\mathbf{m} \times \mathbf{m}_{xx} + \mathbf{f} \text{ on } \Omega = (0,1). \tag{6.20}$$

An exact solution of Eq. (6.20) is

$$\mathbf{m}^e = \begin{pmatrix} u^e \\ v^e \\ w^e \end{pmatrix} = \begin{pmatrix} \cos(x^2(1-x)^2)\sin(t) \\ \sin(x^2(1-x)^2)\sin(t) \\ \cos(t) \end{pmatrix}.$$

In its component form, the forcing term $\mathbf{f} = \mathbf{m}^e_t + \mathbf{m}^e \times \mathbf{m}^e_{xx}$ can be calculated as follows.

$$\mathbf{f} = \begin{pmatrix} \cos(X)\cos(t) + [(X')^2\sin(X) - X''\cos(X)]\sin(t)\cos(t) \\ \sin(X)\cos(t) - [(X')^2\cos(X) + X''\sin(X)]\sin(t)\cos(t) \\ -\sin(t) + X''\sin^2(t) \end{pmatrix},$$

where $X = x^2(1-x)^2$. Now, we will solve Eq. (6.20) with an initial condition $\mathbf{m}(x,0) = (0,\ 0,\ 1)$ and zero Neumann boundary condition; i.e., $\mathbf{m}_x = \mathbf{0}$ at $\partial\Omega = \{0,1\}$. We define the numerical error $\mathbf{e}^n_i = \mathbf{m}^n_i - \mathbf{m}^e(x_i, t^n)$ for $i = 1, 2, \cdots, N_x$. The discrete l_2-norm and the maximum norm are defined as

$$\|\mathbf{e}^n\|_{l_2} = \sqrt{\sum_{1\le i\le N_x} \frac{\mathbf{e}^n_i \cdot \mathbf{e}^n_i}{3N_x}} \text{ and } \|\mathbf{e}^n\|_\infty = \max_{1\le i\le N_x} \sqrt{\mathbf{e}^n_i \cdot \mathbf{e}^n_i}.$$

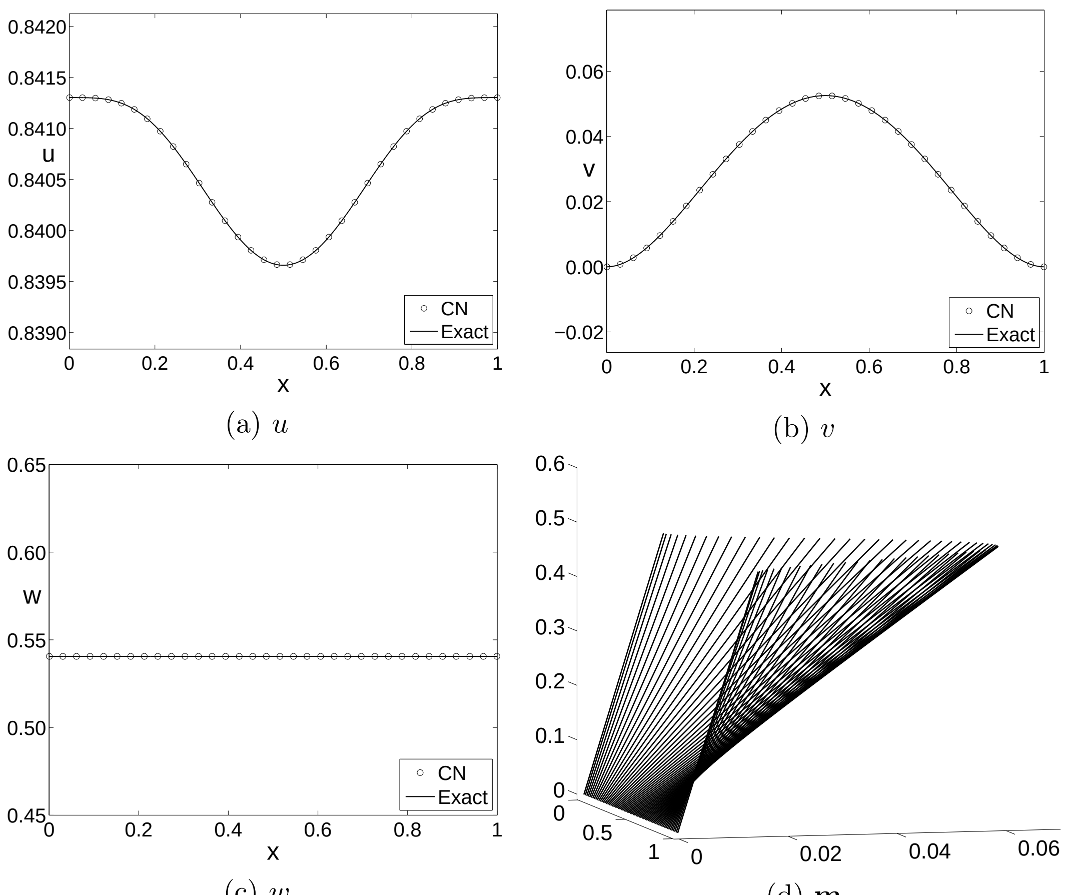


FIGURE 6.2. Magnetization distribution for one dimensional equation: (a) u, (b) v, (c) w, and (d) $\mathbf{m}$ at $T = 1$.

To obtain an estimate of the convergence rate, we performed a number of simulations on a set of increasingly finer grids. We computed the numerical solutions on uniform grids, $h = 1/2^n$ for $n = 6, 7, 8, 9,$ and 10. For each case, we ran the calculation to time $T = 1$ with a time step $\Delta t = 0.32h$. Fig. 6.2 shows the numerical results for the one dimensional equation: (a) u, (b) v, (c) w, and (d) $\mathbf{m}$ at $T = 1$. The numerical results agree very well with the exact solution. The errors and rates of convergence are given in Table 6.1. The results suggest that the scheme is indeed second-order accurate in space and time. Also, the convergence results imply the preservation of the length of the vector field, $\mathbf{m}$.

case	64	rate	128	rate	256	rate	512	rate	1024
$\|\mathbf{e}^n\|_{l_2}$	6.5E-5	1.99	1.6E-5	1.99	4.1E-6	2.00	1.0E-6	2.00	2.5E-7
$\|\mathbf{e}^n\|_{\infty}$	1.1E-4	1.99	2.7E-5	1.99	6.7E-6	2.00	1.7E-6	1.99	4.2E-7

TABLE 6.1. The l_2 and maximum norms and convergence rates with space step $h = 1/N_x$, time step $\Delta t = 0.32h$, total time $T = 1$, and an iteration convergence tolerance of 10^{-10}.

We calculate the stability constraint for the proposed scheme and take a similar test problem in [62]. We try to find the maximum Δt corresponding to different spatial grid sizes h so that stable solutions can be computed up to $T = 1$. The results are shown in Table 6.2 and we obtain stable solutions for all five mesh sizes. The results indicate that the proposed CN scheme is practically unconditionally stable because time step constraint from accuracy concern is more restrictive than one from stability of the numerical scheme.

mesh size	$h = 1/64$	$h = 1/128$	$h = 1/256$	$h = 1/512$	$h = 1/1024$
time step	$\Delta t \geq h$	$\Delta t \geq h$	$\Delta t \geq h$	$\Delta t \geq h$	$\Delta t \geq h$

TABLE 6.2. Stability constraint of Δt for the proposed scheme.

6.3.1.2. *Energy conservation.* Next, we investigate the energy conservation property when $\mathbf{f} \equiv \mathbf{0}$. An exact solution of the equation is

$$\begin{aligned} u^e(x,t) &= \sin(\alpha)\cos(kx + tk^2\cos(\alpha)), \\ v^e(x,t) &= \sin(\alpha)\sin(kx + tk^2\cos(\alpha)), \\ w^e(x,t) &= \cos(\alpha). \end{aligned}$$

We see that the function $\mathbf{m}(x,t) = (u^e(x,t), v^e(x,t), w^e(x,t))$ satisfies the Eq. (6.1). Now we apply an initial condition to the Eq. (6.1)

$$(u_0, v_0, w_0) = (\sin(\alpha)\cos(kx), \sin(\alpha)\sin(kx), \cos(\alpha)),$$

where $\alpha = \pi/4$, $k = 2\pi$, and a periodic boundary condition is applied; i.e., $\mathbf{m}_0 = \mathbf{m}_{N_x}$ and $\mathbf{m}_{N_x+1} = \mathbf{m}_1$.

Theoretically, this energy is constant irrespective of time. Now we confirm that numerically. Fig. 6.3 shows a numerical result for the evolution of the discrete energy up to time $T = 1$ by the proposed scheme with $N_x = 64$, $h = 1/N_x$, and $\Delta t = 0.32h$. We can see that the conservation of energy holds. Fig. 6.4 shows magnetization distribution for one dimensional equation:(a) u, (b) v, (c) w, and (d) $\mathbf{m}$ at $T = 1$ without forcing term and periodic boundary condition.

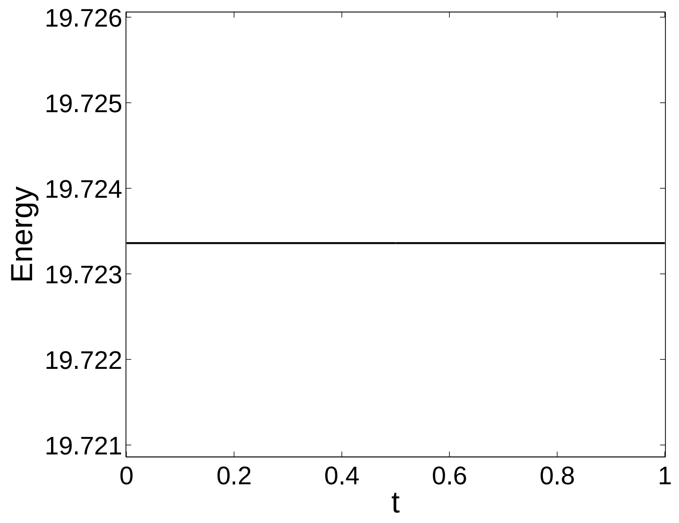


FIGURE 6.3. A temporal evolution of the discrete energy of the numerical solution.

We also calculated the rate of convergence. The errors and rates of convergence are given in Table 6.3. The results suggest that the scheme is indeed second-order accurate in space and time.

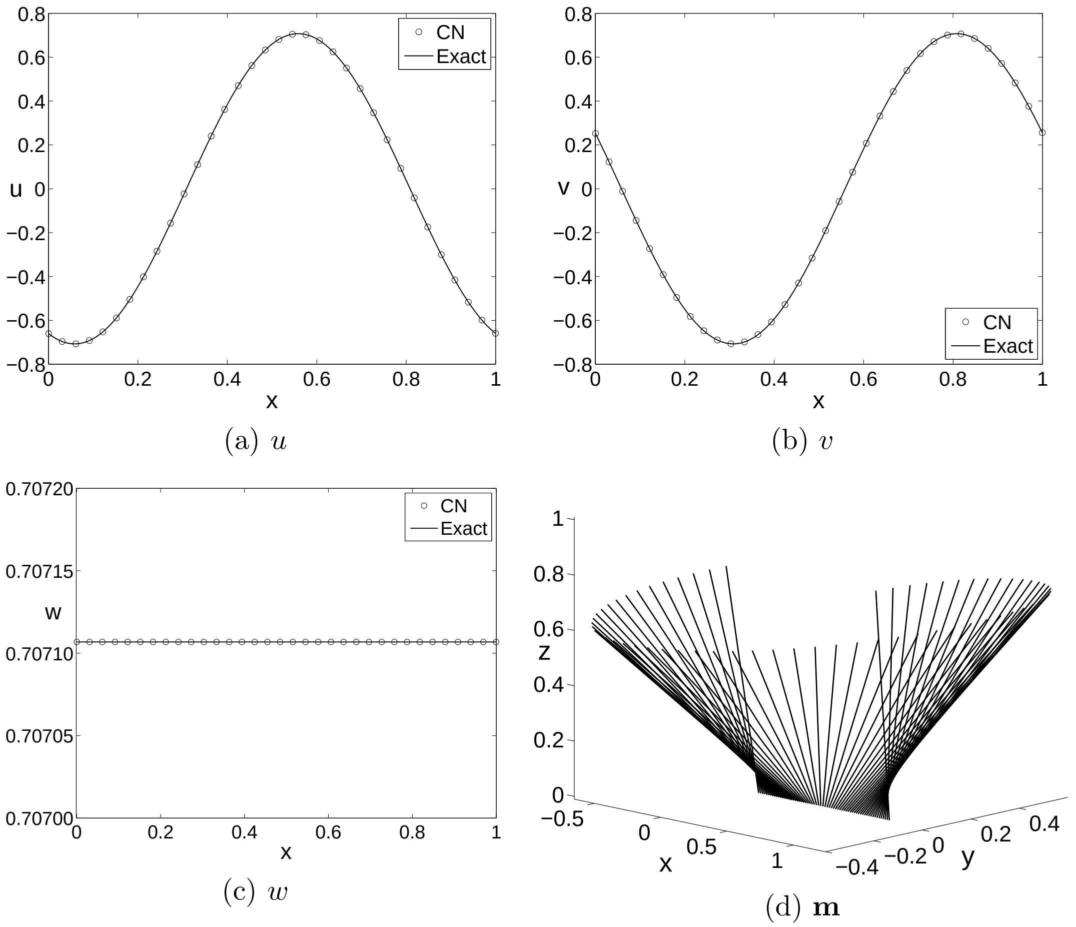
case	64	rate	128	rate	256	rate	512	rate	1024
$\|\mathbf{e}^n\|_{l_2}$	3.2E-2	1.99	8.0E-3	1.99	2.0E-3	1.99	5.0E-4	1.99	1.2E-4
$\|\mathbf{e}^n\|_{\infty}$	5.5E-2	1.99	1.4E-2	1.99	3.5E-3	1.99	8.6E-4	1.99	2.2E-4

TABLE 6.3. The l_2 and maximum norms and convergence rates with space step $h = 1/N_x$, time step $\Delta t = 0.32h$, total time $T = 1$, and an iteration convergence tolerance of 10^{-10}.

6.3.2. Two space dimensions. In this section we perform two-dimensional numerical experiments with exact solutions to verify the second-order accuracy of the proposed scheme in time and space. Without the forcing term, we also show the energy conservation property.

6.3.2.1. *Convergence test.* The two-dimensional equation on $\Omega = (0,1)\times(0,1)$ with zero Neumann boundary conditions is considered. An exact solution is

$$\mathbf{m}^e = \begin{pmatrix} u^e \\ v^e \\ w^e \end{pmatrix} = \begin{pmatrix} \cos(x^2(1-x)^2y^2(1-y)^2)\sin(t) \\ \sin(x^2(1-x)^2y^2(1-y)^2)\sin(t) \\ \cos(t) \end{pmatrix}.$$



(a) u (b) v (c) w (d) $\mathbf{m}$

FIGURE 6.4. Magnetization distribution for one dimensional equation:(a) u, (b) v, (c) w, and (d) $\mathbf{m}$ at $T = 1$ without forcing term and periodic boundary condition.

The forcing term $\mathbf{f} = (f_1,\ f_2,\ f_3) = \mathbf{m}^e_t + \mathbf{m}^e \times \Delta \mathbf{m}^e$ can be calculated as follows.

$$\begin{aligned}
f_1 &= \cos(XY)\cos(t) \\
&\quad +[(Y^2(X')^2 + X^2(Y')^2)\sin(XY) - (YX'' + XY'')\cos(XY)]\sin(t)\cos(t), \\
f_2 &= \sin(XY)\cos(t) \\
&\quad -[(Y^2(X')^2 + X^2(Y')^2)\cos(XY) + (YX'' + XY'')\sin(XY)]\sin(t)\cos(t), \\
f_3 &= -\sin(t) + (YX'' + XY'')\sin^2(t),
\end{aligned}$$

where $X = x^2(1-x)^2$, $Y = y^2(1-y)^2$, and prime denotes a derivative of functions with respect to its argument variable. Now, we will solve the following equation with an initial condition $\mathbf{m}(x, y, 0) = (0,\ 0,\ 1)$.

$$\mathbf{m}_t = -\mathbf{m} \times (\mathbf{m}_{xx} + \mathbf{m}_{yy}) + \mathbf{f} \tag{6.21}$$

with zero Neumann boundary condition

$$\mathbf{m}_x(0,y,t) = \mathbf{m}_x(1,y,t) = \mathbf{m}_y(x,0,t) = \mathbf{m}_y(x,1,t) = \mathbf{0}.$$

We let $\mathbf{e}_{ij}^n = \mathbf{m}_{ij}^n - \mathbf{m}^e(x_i, y_j, t^n)$ for $i = 1, 2, \cdots, N_x$ and $j = 1, 2, \cdots, N_y$. The discrete l_2-norm and the maximum norm are defined as

$$\|\mathbf{e}^n\|_{l_2} = \sqrt{\sum_{1\le i\le N_x}\sum_{1\le j\le N_y}\frac{\mathbf{e}_{ij}^n\cdot\mathbf{e}_{ij}^n}{3N_xN_y}} \text{ and } \|\mathbf{e}^n\|_\infty = \max_{1\le i\le N_x}\max_{1\le j\le N_y}\sqrt{\mathbf{e}_{ij}^n\cdot\mathbf{e}_{ij}^n}.$$

We performed a number of simulations on a set of increasingly finer grids to calculate the rate of convergence. The results are shown in Table 6.4. The l_2 and the maximum norms and convergence rates for $\mathbf{m}$ of CN scheme with space step $h = 1/N_x$, time step $\Delta t = 0.32h$, total time $T = 0.01$, $\alpha = \pi/24$, and an iteration convergence tolerance of 10^{-10}. The results suggest that the scheme is indeed second-order accurate in space and time.

case	64^2	rate	128^2	rate	256^2	rate	512^2	rate	1024^2
$\|\mathbf{e}^n\|_{l_2}$	6.6E-9	1.99	1.6E-9	1.99	4.1E-10	1.99	1.0E-10	2.00	2.6E-11
$\|\mathbf{e}^n\|_\infty$	1.3E-8	1.99	3.4E-9	1.98	8.4E-10	2.00	2.1E-10	2.00	5.3E-11

TABLE 6.4. The l_2 and the maximum norms and convergence rates for $\mathbf{m}$ of CN scheme with space step $h = 1/N_x$, time step $\Delta t = 0.32h$, total time $T = 0.01$, and an iteration convergence tolerance of 10^{-10}.

6.3.2.2. *Energy conservation.* We consider Eq. (6.1) in its component form with the source term $\mathbf{f} \equiv 0$ on the two-dimensional unit domain. An exact solution of the equation is

$$\begin{aligned} u^e(x,y,t) &= \sin(\alpha)\cos(k(x+y) + 2tk^2\cos(\alpha)), \\ v^e(x,y,t) &= \sin(\alpha)\sin(k(x+y) + 2tk^2\cos(\alpha)), \\ w^e(x,y,t) &= \cos(\alpha), \end{aligned}$$

where $\alpha = \pi/24$, $k = 2\pi$. We see that the function $\mathbf{m}(x,y,t) = (u^e(x,y,t), v^e(x,y,t), w^e(x,y,t))$ satisfies the Eq. (6.1). Now we apply an initial conditions to Eq. (6.1)

$$(u_0, v_0, w_0) = (\sin(\alpha)\cos(k(x+y)), \sin(\alpha)\sin(k(x+y)), \cos(\alpha))$$

and a periodic boundary condition is applied; i.e.,

$$\begin{aligned} \mathbf{m}_{0,j} &= \mathbf{m}_{N_x,j},\ \mathbf{m}_{N_x+1,j} = \mathbf{m}_{1,j} \text{ for } 1 \le j \le N_y, \\ \mathbf{m}_{i,0} &= \mathbf{m}_{i,N_y},\ \mathbf{m}_{i,N_y+1} = \mathbf{m}_{i,1} \text{ for } 1 \le i \le N_x. \end{aligned}$$

Table 6.5 shows the l_2 and maximum norms and convergence rates with space step $h = 1/N_x$, time step $\Delta t = 0.32h$, total time $T = 0.01$, $\alpha = \pi/24$, and an iteration convergence tolerance of 10^{-10}. The results suggest that the scheme is second-order accurate in space and time.

Fig. 6.5 shows the orthogonal projection of the vector field $0.1\mathbf{m}$ of the numerical solution onto the xy plane at (a) $T = 0$ and (b) $T = 0.1$. We observe that the solution at $T = 0.1$ is a translation of the initial configuration without changing its magnitude. Here we scaled $\mathbf{m}$ as $0.1\mathbf{m}$ for a visual clarity.

case	64^2	rate	128^2	rate	256^2	rate	512^2	rate	1024^2
$\|\mathbf{e}^n\|_{l_2}$	7.8E-4	1.97	2.0E-4	1.99	5.0E-5	1.99	1.3E-5	1.99	3.1E-6
$\|\mathbf{e}^n\|_{\infty}$	1.4E-3	1.97	3.4E-4	1.99	8.7E-5	1.99	2.2E-5	1.99	5.4E-6

TABLE 6.5. The l_2 and maximum norms and convergence rates with space step $h = 1/N_x$, time step $\Delta t = 0.32h$, total time $T = 0.01$, $\alpha = \pi/24$, and an iteration convergence tolerance of 10^{-10}.

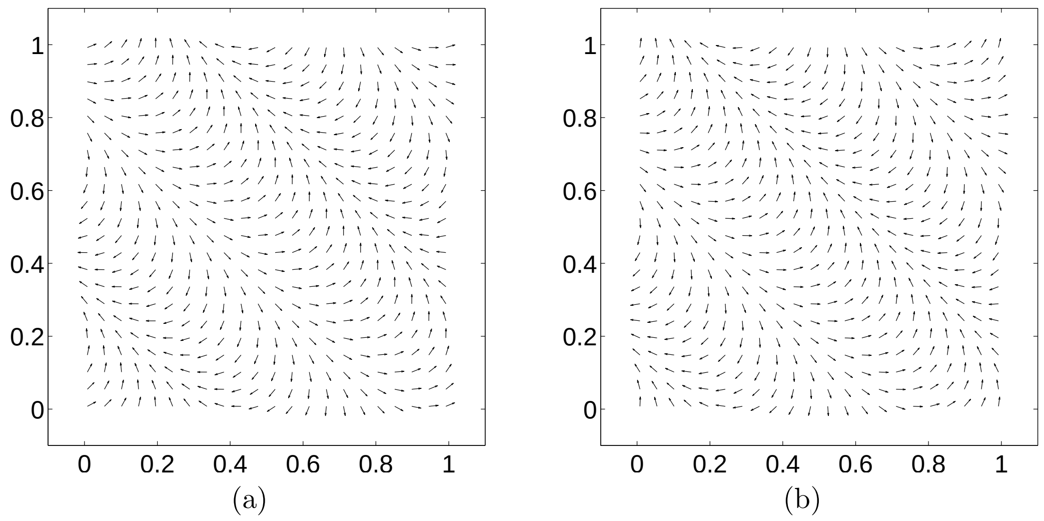


FIGURE 6.5. 2D Landau-Lifshitz equation with periodic boundary condition. (a) initial vector field (b) the numerical solution at $T = 0.1$.

Next, we test an energy conservation property. We define a discrete energy as

$$E(\mathbf{m}^n) = \sum_{i=1}^{N_x}\sum_{j=1}^{N_y}\Big[\left(u^n_{i+1,j} - u^n_{ij}\right)^2 + \left(v^n_{i+1,j} - v^n_{ij}\right)^2 + \left(w^n_{i+1,j} - w^n_{ij}\right)^2$$
$$+ \left(u^n_{i,j+1} - u^n_{ij}\right)^2 + \left(v^n_{i,j+1} - v^n_{ij}\right)^2 + \left(w^n_{i,j+1} - w^n_{ij}\right)^2\Big].$$

Theoretically, this energy is constant irrespective of time. Now we confirm that numerically. Fig. 6.6 shows the time evolution of the energy $E(\mathbf{m}^n)$ with $N_x = N_y = 64$, $h = 1/N_x$, $\Delta t = 0.32h$, $T = 0.01$, and $\alpha = \pi/24$. As expected from (6.4), the energy is constant throughout the evolution.

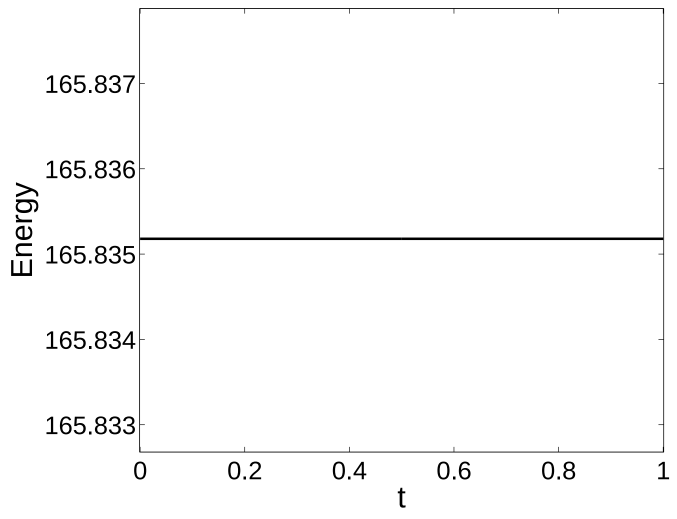


FIGURE 6.6. A temporal evolution of the discrete energy of the numerical solution.

6.4. Conclusion

In this Chapter, we have proposed a Crank-Nicolson time-stepping procedure for LL equation which has a second-order convergence in time and space. We overcame the difficulties with CN scheme associated with LL equation by a cancelation. We used a nonlinear multigrid method for handling the nonlinearities of the discrete system at each time step. We validated our numerical algorithm by various numerical experiments. We tested the second-order convergence and an energy conservation of the proposed scheme. We also showed that the time step restriction for the stability is less restrictive than the accuracy. As future research, the full version of Landau-Liftshitz equation will be investigated.

This chapter is published in Journal of Computational and Applied Mathematics, Vol. 234, pp. 613-623, (2010).

Chapter 7

An accurate and efficient numerical method for Black-Scholes model

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.

7.1. Introduction

Black and Scholes [85], and Merton [75] derived a parabolic second order partial differential equation (PDE) for the value $u(\mathbf{s}, t)$ of an option on stocks. We propose a finite difference method to solve the generalized multi-asset Black-Scholes PDE. Let $s_i, i = 1, 2, ..., n$ denote the price of the underlying i-th asset and $u(s_1, s_2, ..., s_n, t)$ denote the value of the option. The prices s_i of the underlying assets are described by geometric Brownian motions

$$ds_i = \mu_i s_i dt + \sigma_i s_i dW_i, \quad i = 1, 2, ..., n,$$

where μ_i and σ_i denote a constant expected rate of return and a constant volatility of the i-th asset, respectively. Here, W_i is the standard Brownian motion. Let ρ_{ij} denote the correlation coefficient between two Brownian motions W_i and W_j where

$$dW_i dW_j = \rho_{ij} dt, \quad i, j = 1, 2, ..., n, i \neq j.$$

Then, the no arbitrage principle leads to the following generalized n-asset Black-Scholes equation [63, 65, 95]:

$$\frac{\partial u(\mathbf{s}, t)}{\partial t} + \frac{1}{2} \sum_{i,j=1}^{n} \sigma_i \sigma_j \rho_{ij} s_i s_j \frac{\partial^2 u(\mathbf{s}, t)}{\partial s_i \partial s_j} + r \sum_{i=1}^{n} s_i \frac{\partial u(\mathbf{s}, t)}{\partial s_i} = ru(\mathbf{s}, t), \tag{7.1}$$
$$\text{for } (\mathbf{s}, t) = (s_1, s_2, \ldots, s_n, t) \in \mathbf{R}_+^n \times [0, T)$$

where $r > 0$ is a constant riskless interest rate. The final condition is the payoff function $u_T(\mathbf{s})$ at expiry T

$$u(\mathbf{s}, T) = u_T(\mathbf{s}). \tag{7.2}$$

The analytic solutions of Eqs. (7.1) and (7.2) 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 (7.1) and (7.2) using a finite difference method (FDM) [86, 92, 93, 94, 95].

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. The purpose of our work is to propose an efficient and accurate FDM to directly solve the Black-Scholes PDEs (7.1) and (7.2) without transformations of variables.

The outline of this Chapter is the following. In Sec. 7.2 we formulate the Black-Scholes (BS) partial differential equation with two underlying assets. In Sec. 7.3, we focus on the details of a multigrid solver for the BS equation. In Sec. 7.4, we present the results of numerical experiments. We draw conclusions in Sec. 7.5.

7.2. The Black-Scholes model

We use a Black-Scholes model with two underlying assets to keep this presentation simple. However, we can easily extend the current method for more than two underlying assets. Let us consider the computational domain $\Omega = (0, L) \times (0, M)$ for the two assets case. Let $x = s_1$ and $y = s_2$. Then from the change of variable $\tau = T - t$, we obtain an initial value problem:

$$\frac{\partial u}{\partial \tau} = \frac{1}{2}(\sigma_1 x)^2 \frac{\partial^2 u}{\partial x^2} + \frac{1}{2}(\sigma_2 y)^2 \frac{\partial^2 u}{\partial y^2} + \sigma_1 \sigma_2 \rho x y \frac{\partial^2 u}{\partial x \partial y} + r x \frac{\partial u}{\partial x} + r y \frac{\partial u}{\partial y} - r u, \tag{7.3}$$

with $(x, y, \tau) \in \Omega \times (0, T]$ and an initial condition $u(x, y, 0) = u_T(x, y)$ for $(x, y) \in \Omega$. There are several possible boundary conditions such as Neumann [86, 64], Dirichlet, linear, and PDE [86, 94] that can be used for these kinds of problems. In this work, we use a linear boundary condition on all boundaries, i.e.,

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

7.3. A numerical solution

7.3.1. 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 [86, 92, 93, 94, 95]. 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 [66, 77, 79], GMRES (Generalized minimal residual algorithm) method [91, 76], ADI (Alternating direction implicit) method [81, 86], and the OS (Operator splitting) method [86, 69].

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 $j = 1, \ldots, N_y$. 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. (7.3), we have

$$\frac{u_{ij}^{n+1}-u_{ij}^{n}}{\Delta t} \;=\; L_{BS}u_{ij}^{n+1}, \tag{7.4}$$

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

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

7.3.2. A multigrid method. Multigrid methods belong to the class of fastest iterations, because their convergence rate is independent of the space step size [115]. 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. (7.4) 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. 7.1. 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. (7.4) is as follows. We rewrite the above Eq. (7.4) by

$$L_3(u_{3,ij}^{n+1})=\phi_{3,ij}^{n} \text{ on } \Omega_3, \tag{7.5}$$

where

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

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 [122]. 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. (7.4). 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 $u_3^{n+1,0}=u_3^n$.

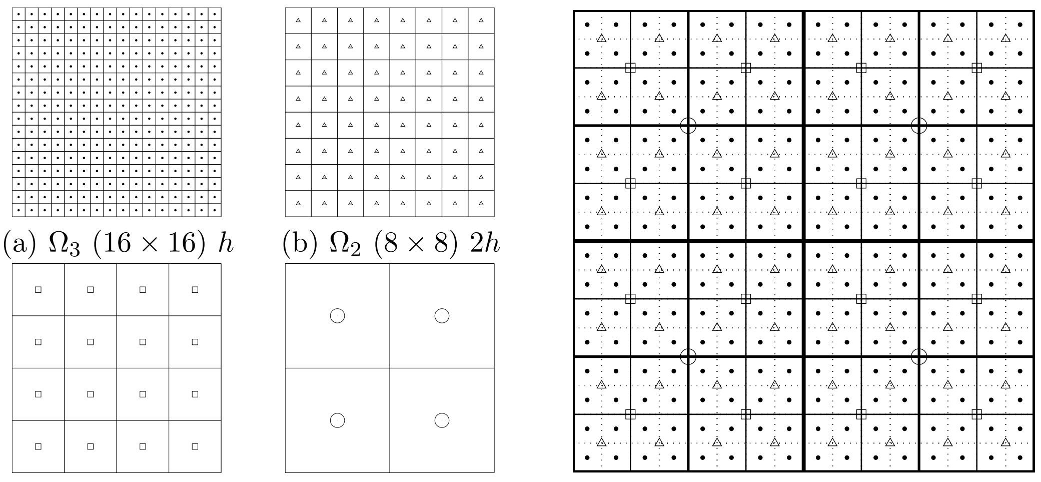
Multigrid cycle

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

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 MG-cycle. Now, define the MGcycle.

Step 1) Presmoothing

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



(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 7.1. (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.

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. (7.5) as

$$\begin{aligned} u_{k,ij}^{n+1} &= \Bigg[\phi_{k,ij}^n + \Delta t\Bigg(\frac{(\sigma_1 x_{k,i})^2}{2}\frac{u_{k,i-1,j}^{n+1}+u_{k,i+1,j}^{n+1}}{h_k^2} \\ &\quad + \frac{(\sigma_2 y_{k,j})^2}{2}\frac{u_{k,i,j-1}^{n+1}+u_{k,i,j+1}^{n+1}}{h_k^2} \\ &\quad + \sigma_1\sigma_2\rho x_{k,i} y_{k,j}\frac{u_{k,i+1,j+1}^{n+1}+u_{k,i-1,j-1}^{n+1}-u_{k,i-1,j+1}^{n+1}-u_{k,i+1,j-1}^{n+1}}{4h_k^2} \\ &\quad + r x_{k,i}\frac{u_{k,i+1,j}^{n+1}-u_{k,i-1,j}^{n+1}}{2h_k} + r y_{k,j}\frac{u_{k,i,j+1}^{n+1}-u_{k,i,j-1}^{n+1}}{2h_k}\Bigg)\Bigg] \Bigg/ \\ &\quad \Bigg[1+\Delta t\Bigg(\frac{(\sigma_1 x_{k,i})^2+(\sigma_2 y_{k,j})^2}{h_k^2}+r\Bigg)\Bigg]. \end{aligned} \tag{7.6}$$

Next, we replace $u_{k,\alpha\beta}^{n+1}$ in Eq. (7.6) 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.,

$$\begin{aligned}
\bar{u}_{k,ij}^{n+1,m} &= \Bigg[\phi_{k,ij}^n + \Delta t \Bigg(\frac{(\sigma_1 x_{k,i})^2}{2} \frac{\bar{u}_{k,i-1,j}^{n+1,m} + u_{k,i+1,j}^{n+1,m}}{h_k^2} \\
&+ \frac{(\sigma_2 y_{k,j})^2}{2} \frac{\bar{u}_{k,i,j-1}^{n+1,m} + u_{k,i,j+1}^{n+1,m}}{h_k^2} \\
&+ \sigma_1\sigma_2\rho x_{k,i} y_{k,j} \frac{u_{k,i+1,j+1}^{n+1,m} + \bar{u}_{k,i-1,j-1}^{n+1,m} - \bar{u}_{k,i-1,j+1}^{n+1,m} - u_{k,i+1,j-1}^{n+1,m}}{4h_k^2} \\
&+ r x_{k,i} \frac{u_{k,i+1,j}^{n+1,m} - \bar{u}_{k,i-1,j}^{n+1,m}}{2h_k} + r y_{k,j} \frac{u_{i,j+1}^{n+1,m} - \bar{u}_{k,i,j-1}^{n+1,m}}{2h_k} \Bigg) \Bigg] / \\
&\Bigg[1 + \Delta t \Bigg(\frac{(\sigma_1 x_{k,i})^2 + (\sigma_2 y_{k,j})^2}{h_k^2} + r \Bigg) \Bigg]. \qquad (7.7)
\end{aligned}$$

Therefore, in a multigrid cycle, one smooth relaxation operator step consists of solving Eq. (7.7) given above for $1 \leq i \leq 2^{k-3}N_x$ and $1 \leq j \leq 2^{k-3}N_y$. *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. 7.2(a).

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

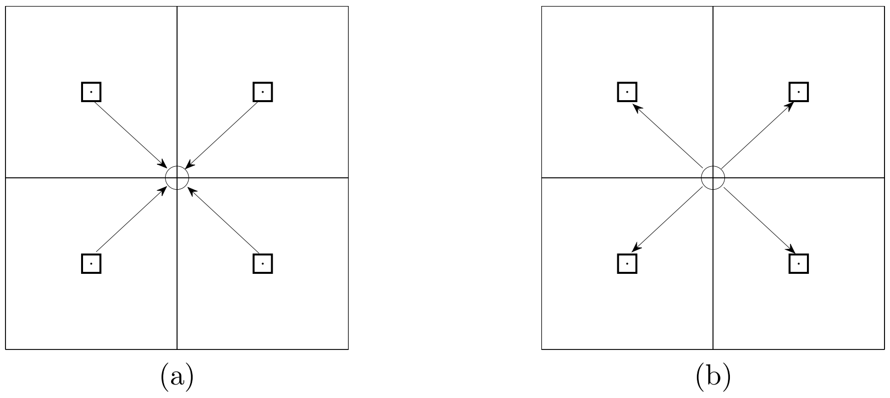


FIGURE 7.2. Transfer operators : (a) restriction and (b) interpolation.

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

$$L_{k-1}(u_{k-1}^{n+1,m}) = \bar{d}_{k-1}^m. \qquad (7.8)$$

If $k = 1$, we use a direct or fast iteration solver for Eq. (7.8). If $k > 1$, we solve Eq. (7.8) 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. 7.2(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. 7.3. For the multi-grid V-cycle, it is given in Fig. 9.4.

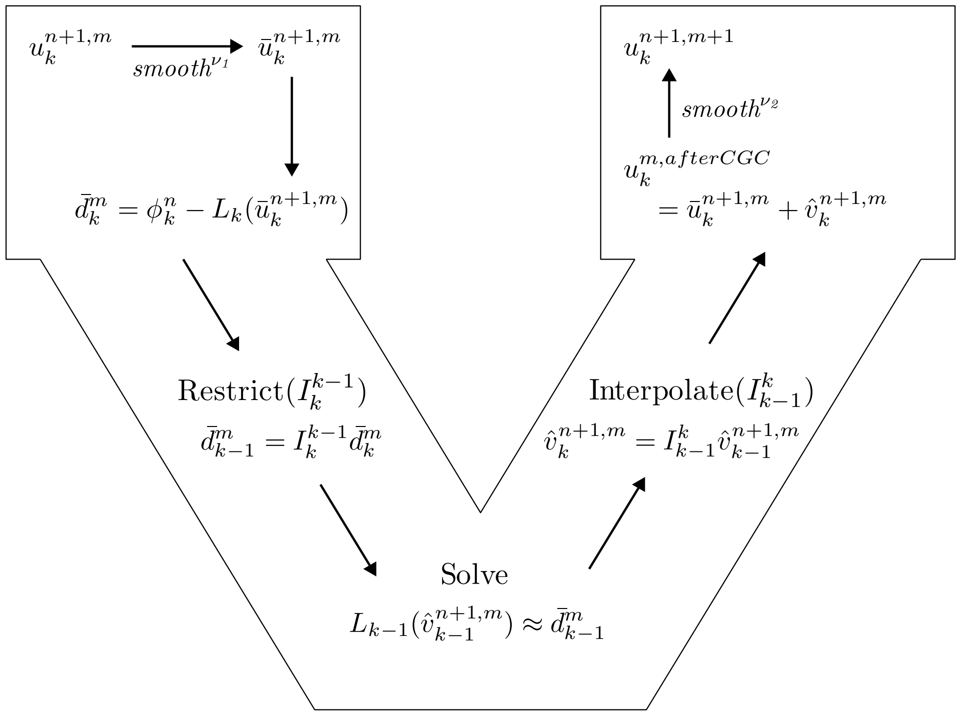

$u_k^{n+1,m}$
$smooth^{\nu_1}$
$\bar{u}_k^{n+1,m}$
$\bar{d}_k^m = \phi_k^n - L_k(\bar{u}_k^{n+1,m})$
Restrict(I_k^{k-1})
$\bar{d}_{k-1}^m = I_k^{k-1}\bar{d}_k^m$
Solve
$L_{k-1}(\hat{v}_{k-1}^{n+1,m}) \approx \bar{d}_{k-1}^m$
Interpolate(I_{k-1}^k)
$\hat{v}_k^{n+1,m} = I_{k-1}^k \hat{v}_{k-1}^{n+1,m}$
$u_k^{m,afterCGC}$
$= \bar{u}_k^{n+1,m} + \hat{v}_k^{n+1,m}$
$smooth^{\nu_2}$
$u_k^{n+1,m+1}$

FIGURE 7.3. The $MG\ (k, k-1)$ two-grid method.

7.4. 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 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

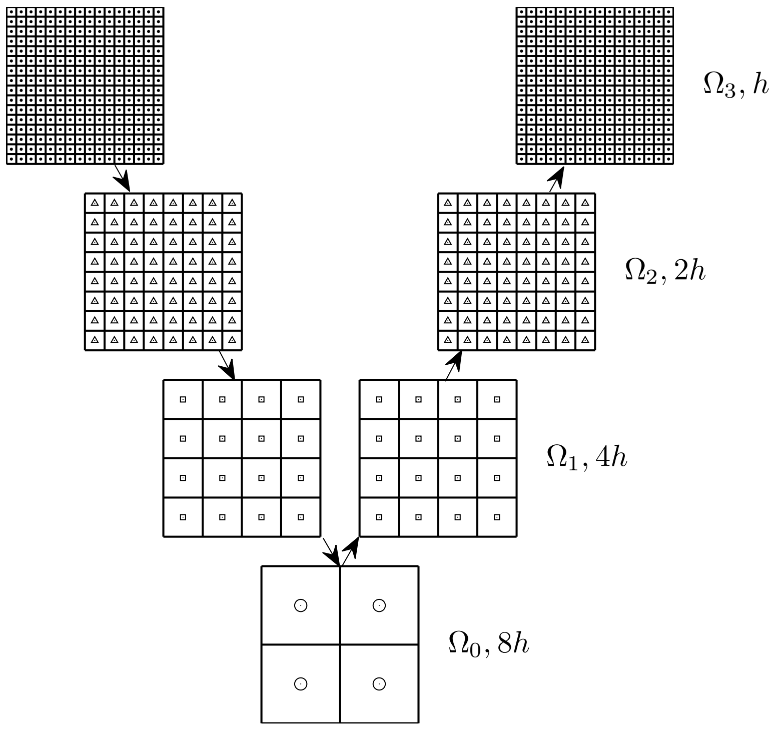


FIGURE 7.4. Schedule of grids for V-cycle.

above the strike X_1 and asset two, y, is above strike X_2 at expiration. The payoff is given by

$$u(x,y,0) = \begin{cases} K & \text{if } x \geq X_1 \text{ and } y \geq X_2, \\ 0 & \text{otherwise .} \end{cases} \tag{7.9}$$

The formula for the exact value is known in [87] by

$$u(x,y,T) = Ke^{-rT}M(\alpha,\beta;\rho), \tag{7.10}$$

where

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

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:

$$M(\alpha,\beta;\rho) = \frac{1}{2\pi\sqrt{1-\rho^2}}\int_{-\infty}^{\alpha}\int_{-\infty}^{\beta} \exp\left[-\frac{x^2-2\rho xy+y^2}{2(1-\rho^2)}\right] dxdy. \tag{7.11}$$

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

```
L=300; K=1; T=0.1; r=0.03; sigma1=0.5; sigma2=0.5; rho=0.5;
```

```
N=64; h=L/N; S=linspace(h/2,L-h/2,N); VE=zeros(N,N); mu=[0 0];

for i=1:N
      for j=1:N
         y1 = (log(S(i)/K)+(b1-sigma1^2/2)*T)/(sigma1*sqrt(T));
         y2 = (log(S(j)/K)+(b2-sigma2^2/2)*T)/(sigma2*sqrt(T));
          X = [y1 y2];
        cov = [1 rho; rho 1];
          M = mvncdf(X,mu,cov);
     V(i,j) = K*exp(-r*T)*M;
    end
end

[X, Y] = meshgrid(S); surf(X, Y, V)
```

7.4.1. 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. (7.9) 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 $r = 0.03$. Figs. 7.5 (a) and (b) show the initial configuration and final profile at T, respectively.

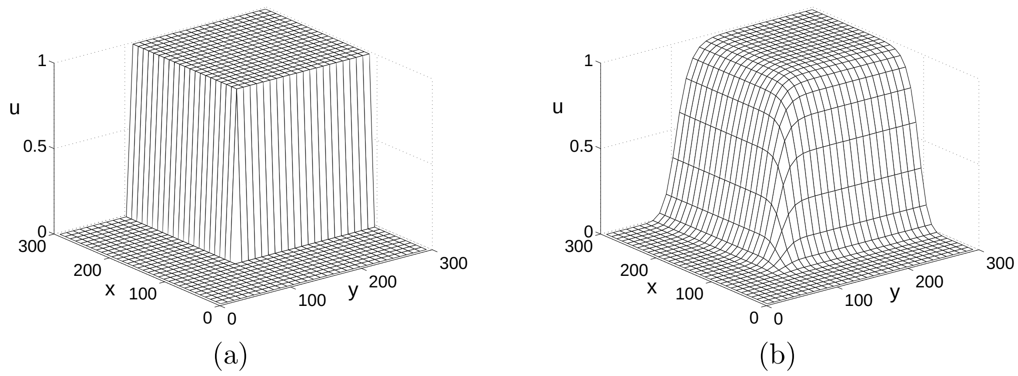


FIGURE 7.5. (a) The initial condition and (b) numerical result at $T = 0.1$.

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. (9.15) 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 7.1. The results show that the scheme is first-order accurate.

Table 7.1. The L^2 norms of errors and convergence rates for u at time $T = 0.1$.

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

7.4.2. Multigrid performance. We investigated the convergence behavior of our 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. 7.6) and the CPU-time in seconds required for a solution to an identical convergence tolerance are displayed in Table 7.2. 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.

7.5. Conclusions

In this paper, 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.

This chapter is published in Communications of the Korean Mathematical Society, Vol. 24, No. 4, pp. 617-628, (2009).

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

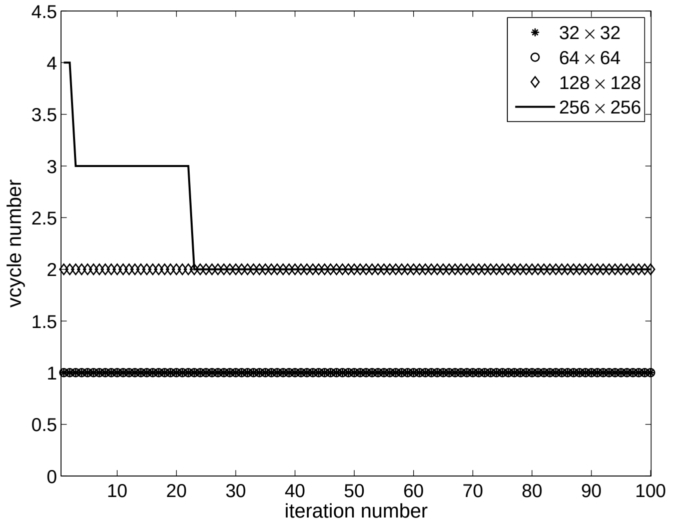


FIGURE 7.6. Number of V-cycles.

Chapter 8

Comparison of numerical methods (Bi-CGSTAB, OS, and MG) for 2D Black-Scholes equations

In this Chapter, we perform comparison of numerical methods for two-dimensional Black-Scholes equations obtained from stock option pricing. The finite difference methods are applied and the resulting linear system 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. Numerical results show that the operator splitting method is the most efficient among these solvers to get the same level of accuracy.

8.1. Introduction

Black and Sholes [85] and Merton [75] derived a Black-Scholes partial differential equation 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. 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 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.

The finite difference methods (FDM) are very popular to approximate the solution of Black-Scholes equations (B-S), see the general settings in option pricing [82, 86, 71, 72, 73, 92, 94, 95]. 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) [70, 77, 79], operator splitting (OS) [69], and multigrid (MG) [67, 115, 78, 80] 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 [77]. Therefore in this work, we compare the performance of the solvers, Bi-CGSTAB, OSM, and multigrid methods, for two-dimensional Black-Scholes equations obtained from option pricing in finance. There have been other system solvers, such as ADI (Alternating Direction Method) [81], GMRES (Generalized Minimum Residual) [76], however we omit the comparison in this work since GMRES and ADI methods are similar to Bi-CGSTAB and OS methods, respectively.

The outline of the Chapter is as follows. In Section 8.2 we first set up the problem to price stock options. In Section 8.3 we describe the general setting of numerical strategies and explain different solvers of linear system. In Section 8.4 we show the comparison of the numerical experiments between the solvers and the conclusions are drawn in Section 8.5.

8.2. Black-Scholes Equations

$$\frac{\partial V}{\partial t} + \sum_{i=1}^{n} r s_i \frac{\partial V}{\partial s_i} + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \sigma_{ij}^2 \rho_{ij} s_i s_j \frac{\partial^2 V}{\partial s_i \partial s_j} - rV = 0. \tag{8.1}$$

8.3. Numerical methods

8.3.1. 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 [86, 92, 94, 93, 95] 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

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

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

$$\begin{aligned} L_{BS}U_{ij}^{n+1} \quad = \quad & \frac{(\sigma_1 x_i)^2}{2}\frac{U_{i-1,j}^{n+1} - 2U_{ij}^{n+1} + U_{i+1,j}^{n+1}}{h^2} \\ & + \frac{(\sigma_2 y_j)^2}{2}\frac{U_{i,j-1}^{n+1} - 2U_{ij}^{n+1} + U_{i,j+1}^{n+1}}{h^2} \\ & + \sigma_1\sigma_2\rho x_i y_j \frac{U_{i+1,j+1}^{n+1} + U_{ij}^{n+1} - U_{i,j+1}^{n+1} - U_{i+1,j}^{n+1}}{h^2} \\ & + r x_i \frac{U_{i+1,j}^{n+1} - U_{ij}^{n+1}}{h} + r y_j \frac{U_{i,j+1}^{n+1} - U_{ij}^{n+1}}{h} - rU_{ij}^{n+1}. \end{aligned} \tag{8.2}$$

The two-dimensional Black-Scholes equation can then be written in reversed time form as

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

where $\tau = T - t$ is the time left to maturity T. Consequently, we need to solve the following system

$$A\mathbf{U} = \mathbf{b}, \tag{8.3}$$

where $\mathbf{U}$ is the approximate solution of (7.3). The above system (8.3) can be solved using Bi-CGSTAB, OS, and multigrid methods. Let us first introduce each method in the following sections.

8.3.2. Bi-CGSTAB. The bi-conjugate gradient stabilized method (Bi-CGSTAB) was developed to solve nonsymmetric linear systems [79]. Bi-CGSTAB solves iteratively the system (8.3). First, we renumber the initial approximation U_{ij}, i.e.,

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

where $l = 1, \dots, N_x \times N_y$, $i = 1, \dots, N_x$, and $j = 1, \dots, N_y$. For example, when $N_x = N_y = 3$. Fig. 8.1 shows the renumbering.

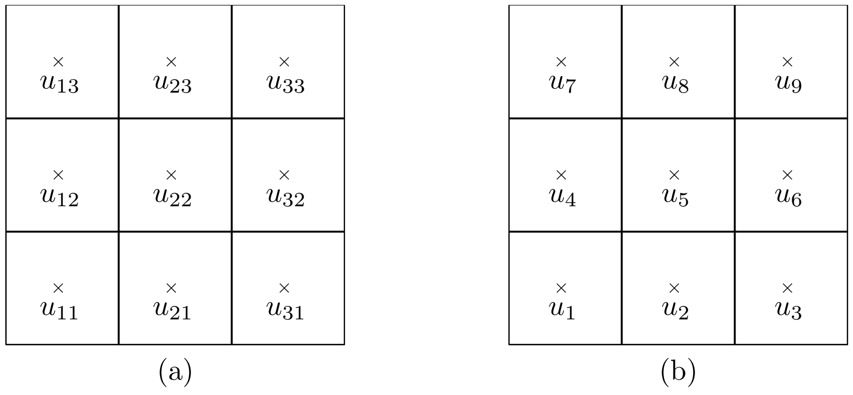


FIGURE 8.1. Renumbering of U_{ij} on 3×3 grid.

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

where

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

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
```

8.3.3. 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:

$$\frac{U_{ij}^{n+1} - U_{ij}^n}{\Delta t} = \mathcal{L}_{BS}^x U_{ij}^{n+\frac{1}{2}} + \mathcal{L}_{BS}^y U_{ij}^{n+1}, \tag{8.4}$$

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

$$\begin{aligned} \mathcal{L}_{BS}^x U_{ij}^{n+\frac{1}{2}} &= \sigma_1^2 x_i^2 \frac{U_{i-1,j}^{n+\frac{1}{2}} - 2U_{ij}^{n+\frac{1}{2}} + U_{i+1,j}^{n+\frac{1}{2}}}{2h^2} \\ &+ \lambda_1 \sigma_1 \sigma_2 \rho x_i y_j \frac{U_{i+1,j+1}^n + U_{ij}^n - U_{i,j+1}^n - U_{i+1,j}^n}{h^2} \\ &+ r x_i \frac{U_{i+1,j}^{n+\frac{1}{2}} - U_{ij}^{n+\frac{1}{2}}}{h} - \lambda_2 r U_{ij}^{n+\frac{1}{2}}, \end{aligned} \tag{8.5}$$

$$\begin{aligned}
\mathcal{L}^y_{BS}U^{n+1}_{ij} &= \sigma_2^2 y_j^2 \frac{U^{n+1}_{i,j-1} - 2U^{n+1}_{ij} + U^{n+1}_{i,j+1}}{2h^2} \\
&\quad + (1-\lambda_1)\sigma_1\sigma_2\rho x_i y_j \frac{U^{n+\frac{1}{2}}_{i+1,j+1} + U^{n+\frac{1}{2}}_{ij} - U^{n+\frac{1}{2}}_{i,j+1} - U^{n+\frac{1}{2}}_{i+1,j}}{h^2} \\
&\quad + r y_j \frac{U^{n+1}_{i,j+1} - U^{n+1}_{ij}}{h} - (1-\lambda_2) r U^{n+1}_{ij}, \qquad (8.6)
\end{aligned}$$

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. (7.3) 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.

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

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

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

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

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

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
```

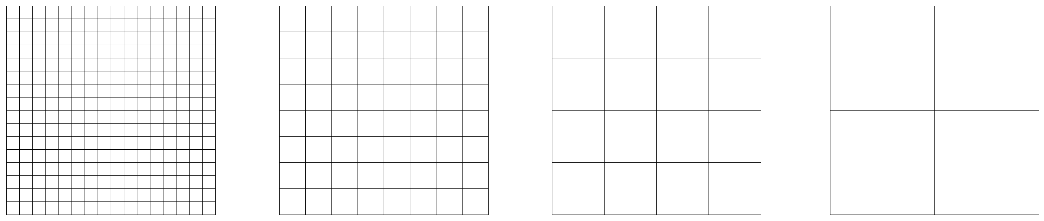
8.3.4. Multigrid method. Multigrid methods belong to the class of fastest iterations, because their convergence rate is independent of the step size h, see [115]. We define a discrete domain by

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

Ω_{k-1} is coarser than Ω_k by factor 2. The multigrid solution of the discrete B-S equation (8.7)

$$\frac{U^{n+1}_{ij} - U^n_{ij}}{\Delta t} = L_{BS} U^{n+1}_{ij} \tag{8.7}$$

makes use of a hierarchy of meshes created by successively coarsening the original mesh, see Fig. 8.2.



Ω_3 (16×16)
Ω_2 (8×8)
Ω_1 (4×4)
Ω_0 (2×2)

FIGURE 8.2. 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 (8.7) by

$$L(U_{ij}^{n+1}) = U_{ij}^n \quad \text{for each } (i,j) \in \Omega_k, \tag{8.8}$$

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 [122]. 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 (8.7). 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 B-S equation (8.7) 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.

$$\bar{U}_k^{n+1,m} = SMOOTH^{\nu_1}(U_k^{n+1,m}, L_k, U_k^n), \tag{8.9}$$

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}:

$$L_{k-1}(U_{k-1}^{n+1,m}) = \bar{d}_{k-1}^m. \tag{8.10}$$

- Solve the Eq. (8.10):

$$\hat{U}_{k-1}^{n+1,m} = \begin{cases} \text{MGcycle}(k-1, \bar{U}_{k-1}^{n+1,m}, L_{k-1}, \bar{d}_{k-1}^n, \nu_1, \nu_2) & \text{for } k > 1 \\ \text{apply the smoothing procedure in (8.9)} & \text{for } k = 1. \end{cases}$$

- 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: $U_k^{n+1,m+1} = SMOOTH^{\nu_2}(U_k^{m,\text{ after }CGC}, L_k, U_k^n)$.

8.4. 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 [74] 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

$$\Lambda(x,y) = \begin{cases} K & \text{if } \ x \geq X_1 \text{ and } y \geq X_2, \\ 0 & \text{otherwise .} \end{cases} \tag{8.11}$$

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

$$\Lambda(x,y) = \begin{cases} K & \text{if } \ x \leq X_1 \text{ and } y \leq X_2, \\ 0 & \text{otherwise .} \end{cases} \tag{8.12}$$

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

$$\Lambda(x,y) = \begin{cases} K & \text{if } \ x \geq X_1 \text{ and } y \leq X_2, \\ 0 & \text{otherwise .} \end{cases} \tag{8.13}$$

Fig. 8.3(a), (b), and (c) show the payoff function $\Lambda(x,y)$ for *Case 1*, *Case 2*, and *Case 3*, respectively.

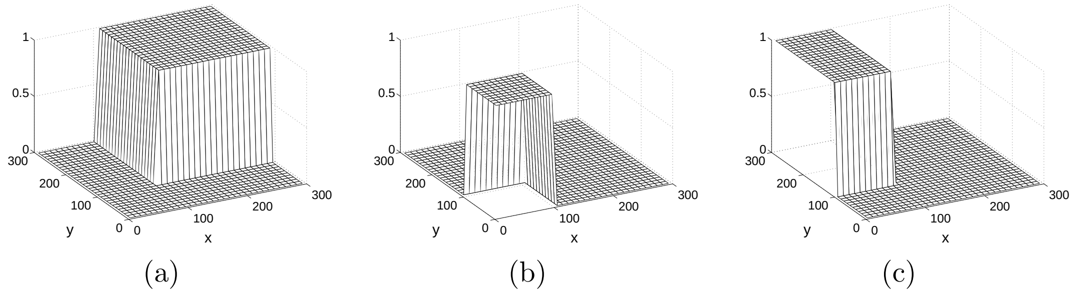


FIGURE 8.3. (a), (b), and (c) are payoff functions of *Case 1*, *Case 2*, and *Case 3*, respectively.

The formulas published by Heynen and Kat [68] can be used to price these binary options:

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

where $\alpha = [\ln(x/X_1) + (r - \sigma_1^2/2)T]/(\sigma_1\sqrt{T})$, $\beta = [\ln(y/X_2) + (r - \sigma_2^2/2)T]/(\sigma_2\sqrt{T})$ [87]. Here $M(\alpha,\beta;\rho)$ denotes a standardized cumulative normal function, Eq. (7.11).

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. 8.4 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, $\|\mathbf{e}\|_2$.

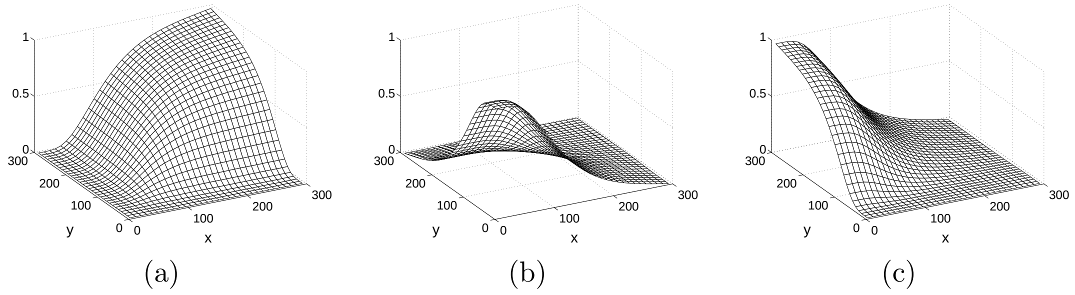


FIGURE 8.4. (a), (b), and (c) are numerical solutions at time $T = 1$ of *Case 1*, *Case 2*, and *Case 3*, respectively.

We test the numerical experiments of different case with three solvers, Bi-CGSTAB, OSM and MG. For fair comparison of these solvers, we match the accuracy of these solvers by changing iteration parameters. The Table 8.1 shows the result of *Case 1*. Fig. 8.5 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.

TABLE 8.1. *Case 1*: Comparison of l_2 error and (CPU time).

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 8.2. *Case 2*: 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 8.3. *Case 3*: 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)

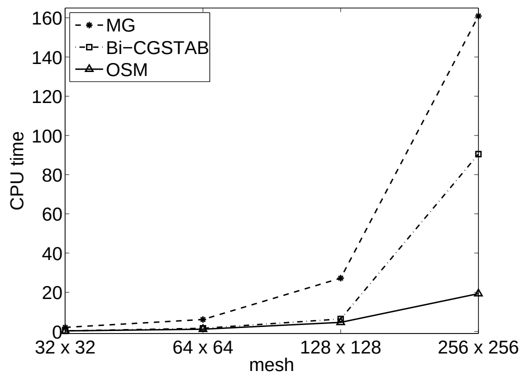
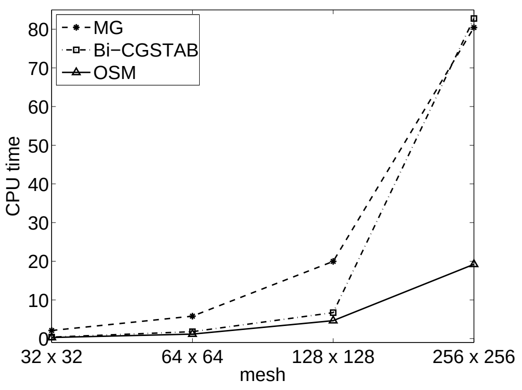


FIGURE 8.5. CPU times of *Case 1*.

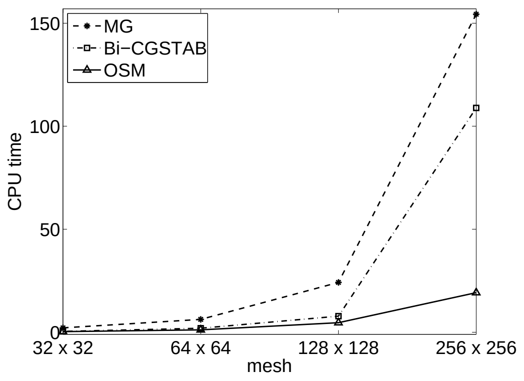
Next, let us check the CPU times to compare efficiency of these solvers. Table 8.1 also shows the CPU times with each method. We can confirm that OS method has a linear CPU time cost as the spatial domain is doubled in each direction. Table 8.2, Table 8.3 and Fig. 8.6 show the CPU times with *Case 2* and *Case 3*. From all these results, we can confirm that OS method faster than other methods under the same accuracy.

8.5. Conclusions

The finite difference methods are applied to the Black-Scholes equations obtained from stock option pricing. The resulting linear system 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-dimentional 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.



(a)



(b)

FIGURE 8.6. (a) and (b) are CPU times of *Case 2* and *Case 3*, respectively.

Chapter 9

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. ADI method is 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 in a two dimensional space. 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. Therefore, OS scheme does not present those problematic source terms. We provide computational results showing the performance of the methods for two underlying asset option pricing problems. The results show that OS method is very efficient and gives better accuracy and robustness than ADI method.

9.1. 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 (see [82, 86, 92, 94, 93, 95]), various numerical methods have been presented by using the finite difference method (FDM) to solve the option pricing problems. But most option pricing problems have nonsmooth payoffs or discontinuous derivatives at the exercise price. Standard finite difference schemes used to solve problems with nonsmooth payoff do not work well for discrete options because of discontinuities introduced in the source term. Moreover, these unwanted oscillations become worse when estimating the hedging parameters, e.g., Delta and Gamma. Alternating direction implicit (ADI) method is used extensively in mathematical finance for numerically solving multi-asset Black-Scholes option pricing models. However, most option pricing problems have nonsmooth payoffs at the exercise price. ADI scheme uses source terms which include y derivatives when we solve x derivative involving equations in a two dimensional space. 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, operator splitting (OS) scheme does not present those problematic source terms. The main contribution of this Chapter is to show the limitation of ADI method in computational finance and suggest the use of OS method instead for accuracy and robust option pricing.

This Chapter is organized as follows. In Section 9.2, we present numerical solution algorithms for ADI and OS methods. In Section 9.3, we show several numerical results by ADI and OS methods. Then we draw the conclusions in Section 9.4.

9.2. Numerical solutions for ADI and OS methods

We focus on the two-dimensional Black-Scholes equation. Let $\mathcal{L}_{BS}$ be the operator

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

The two-dimensional Black-Scholes equation can then be written in reversed time form as

$$\frac{\partial u}{\partial \tau} = \mathcal{L}_{BS} \quad \text{for } (x, y, \tau) \in \Omega \times (0, T], \tag{9.1}$$

where $\tau = T - t$ is the time left to maturity T.

9.2.1. Alternating Directions Implicit (ADI) method.

The main idea of the ADI method (see [81]) is to proceed in two stages, treating only one operator implicitly at each stage. First, a half-step is taken implicitly for the mixed derivative term and the spatial derivative in the x-direction, then explicitly for the spatial derivative in the y-direction. Then, a half-step is taken implicitly for the mixed derivative term and the spatial derivative in the y-direction and explicitly for the spatial derivative in the x-direction. Therefore, the equations for the ADI method are written as follows:

$$\frac{u_{ij}^{n+1} - u_{ij}^{n}}{\Delta\tau} = \mathcal{L}_{ADI}^{x} u_{ij}^{n+\frac{1}{2}} + \mathcal{L}_{ADI}^{y} u_{ij}^{n+1}, \tag{9.2}$$

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

$$\begin{aligned}
\mathcal{L}_{ADI}^{x} u_{ij}^{n+\frac{1}{2}} &= \frac{(\sigma_1 x_i)^2}{4} \frac{u_{i+1,j}^{n+\frac{1}{2}} - 2u_{ij}^{n+\frac{1}{2}} + u_{i-1,j}^{n+\frac{1}{2}}}{h^2} \\
&\quad + \frac{(\sigma_2 y_j)^2}{4} \frac{u_{i,j+1}^{n} - 2u_{ij}^{n} + u_{i,j-1}^{n}}{h^2} \\
&\quad + \frac{1}{2} r x_i \frac{u_{i+1,j}^{n+\frac{1}{2}} - u_{ij}^{n+\frac{1}{2}}}{h} + \frac{1}{2} r y_j \frac{u_{i,j+1}^{n} - u_{ij}^{n}}{h} \\
&\quad + \frac{1}{2}\rho\sigma_1\sigma_2 x_i y_j \frac{u_{i+1,j+1}^{n} + u_{ij}^{n} - u_{i,j+1}^{n} - u_{i+1,j}^{n}}{h^2} - \frac{1}{2} r u_{ij}^{n+\frac{1}{2}},
\end{aligned} \tag{9.3}$$

$$\begin{aligned}
\mathcal{L}^y_{ADI}u^{n+1}_{ij} &= \frac{(\sigma_1 x_i)^2}{4}\frac{u^{n+\frac{1}{2}}_{i+1,j}-2u^{n+\frac{1}{2}}_{ij}+u^{n+\frac{1}{2}}_{i-1,j}}{h^2} \\
&+\frac{(\sigma_2 y_j)^2}{4}\frac{u^{n+1}_{i,j+1}-2u^{n+1}_{ij}+u^{n+1}_{i,j-1}}{h^2} \\
&+\frac{1}{2}rx_i\frac{u^{n+\frac{1}{2}}_{i+1,j}-u^{n+\frac{1}{2}}_{ij}}{h}+\frac{1}{2}ry_j\frac{u^{n+1}_{i,j+1}-u^{n+1}_{ij}}{h} \\
&+\frac{1}{2}\rho\sigma_1\sigma_2 x_i y_j\frac{u^{n+\frac{1}{2}}_{i+1,j+1}+u^{n+\frac{1}{2}}_{ij}-u^{n+\frac{1}{2}}_{i,j+1}-u^{n+\frac{1}{2}}_{i+1,j}}{h^2}-\frac{1}{2}ru^{n+1}_{ij}. \quad (9.4)
\end{aligned}$$

Then we approximate each sub-problem by a semi-implicit scheme.

$$\frac{u^{n+\frac{1}{2}}_{ij}-u^n_{ij}}{\Delta\tau} = \mathcal{L}^x_{ADI}u^{n+\frac{1}{2}}_{ij}, \quad (9.5)$$

$$\frac{u^{n+1}_{ij}-u^{n+\frac{1}{2}}_{ij}}{\Delta\tau} = \mathcal{L}^y_{ADI}u^{n+1}_{ij}. \quad (9.6)$$

Note that the addition of two Eqs. (9.5) and (9.6) results in Eq. (9.2).
Algorithm ADI

- *Step 1*
  The first stage of the ADI method, Eq. (9.3) is rewritten

$$\alpha_i u^{n+\frac{1}{2}}_{i-1,j}+\beta_i u^{n+\frac{1}{2}}_{ij}+\gamma_i u^{n+\frac{1}{2}}_{i+1,j}=f_{ij}, \quad (9.7)$$

where

$$\begin{aligned}
\alpha_i &= -\frac{(\sigma_1 x_i)^2}{4h^2},\ \beta_i=\frac{1}{\Delta\tau}+\frac{(\sigma_1 x_i)^2}{2h^2}+\frac{rx_i}{2h}+\frac{r}{2}, \\
\gamma_i &= -\frac{(\sigma_1 x_i)^2}{4h^2}-\frac{rx_i}{2h}, \quad \text{for } i=2,...,N_x-1.
\end{aligned}$$

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

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

where A_x is a tridiagonal matrix constructed from Eq. (9.7) with a linear boundary condition

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

and $f_{1:N_x,j}$ is an N_x-dimensional vector with components

$$\begin{aligned} f_{ij} &= \frac{u_{ij}^n}{\Delta\tau} + \frac{1}{4}(\sigma_2 y_j)^2 \frac{u_{i,j+1}^n - 2u_{ij}^n + u_{i,j-1}^n}{h^2} + \frac{1}{2} r y_j \frac{u_{i,j+1}^n - u_{ij}^n}{h} \\ &\quad + \frac{1}{2}\rho\sigma_1\sigma_2 x_i y_j \frac{u_{i+1,j+1}^n + u_{ij}^n - u_{i,j+1}^n - u_{i+1,j}^n}{h^2}. \end{aligned} \tag{9.8}$$

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 $f_{ij}$ by Eq. (9.8)
  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. (9.4) is rewritten

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

where

$$\begin{aligned} \alpha_j &= -\frac{(\sigma_2 y_j)^2}{4h^2}, \ \beta_j = \frac{1}{\Delta\tau} + \frac{(\sigma_2 y_j)^2}{2h^2} + \frac{r y_j}{2h} + \frac{r}{2}, \\ \gamma_j &= -\frac{(\sigma_2 y_j)^2}{4h^2} - \frac{r y_j}{2h}, \quad \text{for } j = 2, ..., N_y - 1. \end{aligned}$$

For a fixed index i, the vector $u_{i,1:N_y}^{n+1}$ may 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 given as

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

and $g_{i,1:N_y}$ is an N_y-dimensional vector with components

$$\begin{aligned} g_{ij} &= \frac{u_{ij}^{n+\frac{1}{2}}}{\Delta\tau} + \frac{(\sigma_1 x_i)^2}{4} \frac{u_{i+1,j}^{n+\frac{1}{2}} - 2u_{ij}^{n+\frac{1}{2}} + u_{i-1,j}^{n+\frac{1}{2}}}{h^2} + \frac{1}{2} r x_i \frac{u_{i+1,j}^{n+\frac{1}{2}} - u_{ij}^{n+\frac{1}{2}}}{h} \\ &\quad + \frac{1}{2}\rho\sigma_1\sigma_2 x_i y_j \frac{u_{i+1,j+1}^{n+\frac{1}{2}} + u_{ij}^{n+\frac{1}{2}} - u_{i,j+1}^{n+\frac{1}{2}} - u_{i+1,j}^{n+\frac{1}{2}}}{h^2}. \end{aligned} \tag{9.9}$$

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 $g_{ij}$ by Eq. (9.9)
  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.

9.2.2. Operator splitting (OS) method. The idea of OS method (see [86, 83]) is to divide each time step into fractional time steps with simpler operators. We shall introduce the basic OS scheme for the two-dimensional BS equation. The basic idea behind OS method is to replace a two-dimensional scheme as

$$\frac{u^{n+1}_{ij} - u^n_{ij}}{\Delta\tau} = \mathcal{L}^x_{OS} u^{n+\frac{1}{2}}_{ij} + \mathcal{L}^y_{OS} u^{n+1}_{ij}, \tag{9.10}$$

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

$$\begin{aligned}
\mathcal{L}^x_{OS} u^{n+\frac{1}{2}}_{ij} &= \frac{(\sigma_1 x_i)^2}{2} \frac{u^{n+\frac{1}{2}}_{i-1,j} - 2u^{n+\frac{1}{2}}_{ij} + u^{n+\frac{1}{2}}_{i+1,j}}{h^2} \\
&+\frac{1}{2}\sigma_1\sigma_2\rho x_i y_j \frac{u^n_{i+1,j+1} + u^n_{ij} - u^n_{i,j+1} - u^n_{i+1,j}}{h^2} \\
&+r x_i \frac{u^{n+\frac{1}{2}}_{i+1,j} - u^{n+\frac{1}{2}}_{ij}}{h} - \frac{1}{2} r u^{n+\frac{1}{2}}_{ij},
\end{aligned}$$

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

We then approximate each sub-problem by a semi-implicit scheme.

$$\frac{u^{n+\frac{1}{2}}_{ij} - u^n_{ij}}{\Delta\tau} = \mathcal{L}^x_{OS} u^{n+\frac{1}{2}}_{ij}, \tag{9.11}$$

$$\frac{u^{n+1}_{ij} - u^{n+\frac{1}{2}}_{ij}}{\Delta\tau} = \mathcal{L}^y_{OS} u^{n+1}_{ij}. \tag{9.12}$$

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 $n + \frac{1}{2}$ and then to time level $n+1$. Note that the addition of two Eqs. (9.11) and (9.12) results in Eq. (9.10).

Algorithm OS

- *Step 1*
  Eq. (9.11) is rewritten as follows:
$$\alpha_i u_{i-1,j}^{n+\frac{1}{2}} + \beta_i u_{ij}^{n+\frac{1}{2}} + \gamma_i u_{i+1,j}^{n+\frac{1}{2}} = f_{ij},$$
  where
$$\begin{aligned}\alpha_i &= -\frac{1}{2}\frac{(\sigma_1 x_i)^2}{h^2}, \ \beta_i = \frac{1}{\Delta\tau} + \frac{(\sigma_1 x_i)^2}{h^2} + \frac{r x_i}{h} + \frac{1}{2}r,\\ \gamma_i &= -\frac{1}{2}\frac{(\sigma_1 x_i)^2}{h^2} - \frac{r x_i}{h}, \qquad \text{for } i = 2, ..., N_x - 1.\end{aligned}$$
  The first step of the OS method is then implemented in a loop over the y-direction for a fixed index i:

```
for   $j = 1 : N_y$
   for   $i = 1 : N_x$
         Set $f_{ij}$ by Eq. (9.13)
   end
   Solve $A_x u_{1:N_x,j}^{n+\frac{1}{2}} = f_{1:N_x,j}$ by using the Thomas algorithm
end
```

  Here A_x is a tridiagonal matrix,
$$A_x = \begin{pmatrix} 2\alpha_1+\beta_1 & \gamma_1-\alpha_1 & 0 & \dots & 0 & 0\\ \alpha_2 & \beta_2 & \gamma_2 & \dots & 0 & 0\\ 0 & \alpha_3 & \beta_3 & \dots & 0 & 0\\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots\\ 0 & 0 & 0 & \dots & \beta_{N_x-1} & \gamma_{N_x-1}\\ 0 & 0 & 0 & \dots & \alpha_{N_x}-\gamma_{N_x} & \beta_{N_x}+2\gamma_{N_x}\end{pmatrix}$$
  and f_{ij} is an N_y-dimensional vector with components
$$f_{ij} = \frac{1}{2}\rho\sigma_1\sigma_2 x_i y_j \frac{u_{i+1,j+1}^n - u_{i+1,j}^n - u_{i,j+1}^n + u_{ij}^n}{h^2} + \frac{u_{ij}^n}{\Delta\tau}. \tag{9.13}$$

- *Step 2*

  As before, Eq. (9.12) is rewritten as follows:
$$\alpha_j u_{i,j-1}^{n+1} + \beta_j u_{ij}^{n+1} + \gamma_j u_{i,j+1}^{n+1} = g_{ij},$$
  where
$$\begin{aligned}\alpha_j &= -\frac{1}{2}\frac{(\sigma_2 y_j)^2}{h^2}, \ \beta_j = \frac{1}{\Delta\tau} + \frac{(\sigma_2 y_j)^2}{h^2} + \frac{r y_j}{h} + \frac{1}{2}r,\\ \gamma_j &= -\frac{1}{2}\frac{(\sigma_2 y_j)^2}{h^2} - \frac{r y_j}{h}, \qquad \text{for } j = 2, ..., N_y - 1.\end{aligned}$$

By analogy, this step of the OS method is implemented in a loop over the x-direction:

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

where the matrix A_y is tridiagonal,

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

and $g_{i,1:N_y}$ is an N_y-dimensional vector with components

$$g_{ij} = \frac{1}{2}\rho\sigma_1\sigma_2 x_i y_j \frac{u_{i+1,j+1}^{n+\frac{1}{2}} - u_{i+1,j}^{n+\frac{1}{2}} - u_{i,j+1}^{n+\frac{1}{2}} + u_{ij}^{n+\frac{1}{2}}}{h^2} + \frac{u_{ij}^{n+\frac{1}{2}}}{\Delta\tau}. \qquad (9.14)$$

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 $g_{ij}$ by Eq. (9.14)
  end
  Solve $A_y u_{i,1:N_y}^{n+1} = g_{i,1:N_y}$ by using the Thomas algorithm
end
```

These two steps consist of one time step evolution for OS method.

9.3. Numerical experiments

In this section, various examples are presented to demonstrate the effectiveness and efficiency of the two different methods, ADI and OS methods, for two-dimensional cases. All computations are performed using uniform meshes for space and time on a 2.53 GHz Intel PC with 2GB RAM. Computer programs are written in MATLAB program.

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

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

where N is the number of points on the gray region in Fig. 9.1.

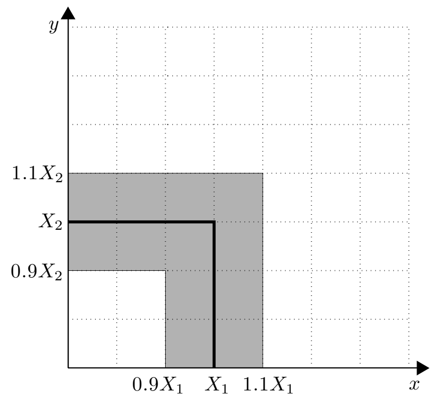


FIGURE 9.1. The gray color region is the region where the RMSE is estimated.

9.3.1. European call option on the maximum of two assets. First, we consider a simple case of a vanilla call option. The payoff for this problem is based on the best of the two assets x and y and is given as:

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

where K_1 and K_2 are the strike prices of underlying assets x and y, respectively. Fig. 9.2 shows European call option payoff on the maximum of two assets with $K_1 = K_2 = 100$.

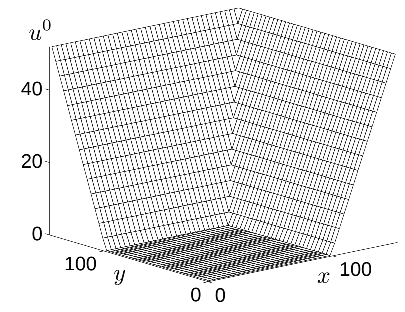


FIGURE 9.2. European option payoff on the maximum of two assets with $K_1 = K_2 = 100$.

We use the Dirichlet boundary condition at $x = L$ and $y = M$ and the linear boundary condition at $x = 0$ and $y = 0$. For example, as shown in Fig. 9.3 we use $a = 2b - c$ for the linear boundary condition and $\alpha = 2D - \beta = 2(M - K_2 \exp(-r\tau)) - \beta$ for Dirichlet boundary condition.

The errors are computed using the exact solution for the call option on the maximum of two assets. The formula for the exact value is known in [87] by

$$\begin{aligned} u(x, y, T) &= x\mathcal{M}(\alpha_1, d; \rho_1) + y\mathcal{M}(\alpha_2, -d + \sigma\sqrt{T}; \rho_2) \\ &\quad -Ke^{-rT}[1 - \mathcal{M}(-\alpha_1 + \sigma_1\sqrt{T}, -\alpha_2 + \sigma_2\sqrt{T}; \rho)], \end{aligned}$$

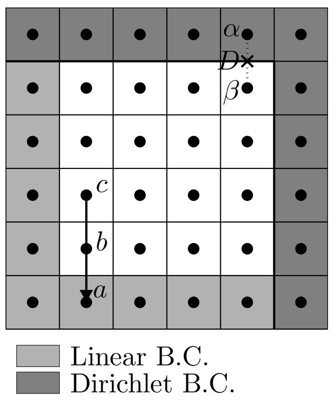


FIGURE 9.3. Boundary conditions.

where

$$d = \frac{\ln(x/y)+(\sigma^2/2)T}{\sigma\sqrt{T}},\ \alpha_1 = \frac{\ln(x/K)+(r+\sigma_1^2/2)T}{\sigma_1\sqrt{T}},\ \alpha_2 = \frac{\ln(y/K)+(r+\sigma_2^2/2)T}{\sigma_2\sqrt{T}},$$
$$\sigma = \sqrt{\sigma_1^2+\sigma_2^2-2\rho\sigma_1\sigma_2},\ \rho_1 = \frac{\sigma_1-\rho\sigma_2}{\sigma},\ \rho_2 = \frac{\sigma_2-\rho\sigma_1}{\sigma}.$$

Here $\mathcal{M}(a,b;\rho)$ denotes a standardized cumulative normal function, Eq. (7.11).

The MATLAB code for the closed form solution of a call option on the maximum of two assets is given as following.

```
clear;K=100;T=0.5;r=0.03;X1=100;X2=100;sigma1=0.3;sigma2=0.3;rho=0.5;
Nx=30;Ny=30;L=300;M=300;h=L/Nx;
x=linspace(h/2,L-h/2,Nx);y=linspace(h/2,M-h/2,Ny);
mu=[0 0]; sigma=sqrt(sigma1^2+sigma2^2-rho*sigma1*sigma2);
rho1=(sigma-rho*sigma2)/sigma; rho2=(sigma-rho*sigma1)/sigma;
cov=[1 rho; rho 1]; cov1=[1 rho1; rho1 1]; cov2=[1 rho2; rho2 1];
for i=1:Nx
 for j=1:Ny
  y1=(log(x(i)/X1)+(r+sigma1^2/2)*T)/(sigma1*sqrt(T));
  y2=(log(y(j)/X2)+(r+sigma2^2/2)*T)/(sigma2*sqrt(T));
  d=(log(x(i)/y(j))+(sigma^2/2)*T)/(sigma*sqrt(T));
  M1=mvncdf([y1 d],mu,cov1); M2=mvncdf([y2 -d+sigma*sqrt(T)],mu,cov2);
  M3=mvncdf([-y1+sigma1*sqrt(T) -y2+sigma2*sqrt(T)],mu,cov);
  V(i,j)=x(i)*M1+x(j)*M2-K*exp(-r*T)*(1-M3);
 end
end
[xx,yy]=meshgrid(x,y);surf(xx,yy,V)
```

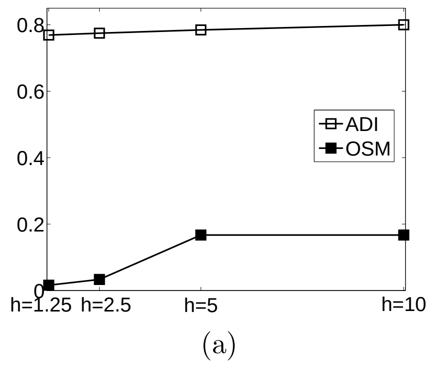
The following parameters have been used. Time to expiry, $T = 0.5$, strike prices, $K_1 = K_2 = 100$, risk-free interest rate, $r = 0.03$, volatilities, $\sigma_1 = \sigma_2 = 0.3$, correlation coefficient, $\rho = 0.5$, and domain size $[0,300] \times [0,300]$ are used.

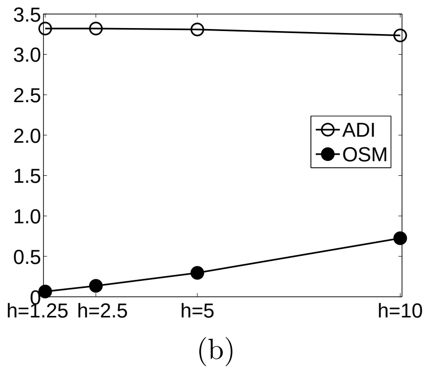
Table 9.1 and Fig. 9.4 show the comparison of errors for ADI and OS methods at time $T = 0.5$. The errors $\|\mathbf{e}\|_2$ and $\|\mathbf{e}\|_\infty$ with ADI method do not decrease with increasing number of grid points and reduced time steps. On the other hand, the errors

Time	Space	ADI			OSM		
$\Delta\tau$	h	$\|\mathbf{e}\|_2$	$\|\mathbf{e}\|_\infty$	RMSE	$\|\mathbf{e}\|_2$	$\|\mathbf{e}\|_\infty$	RMSE
0.10000	10.000	0.799995	3.234941	0.103232	0.167018	0.724133	0.032905
0.05000	5.000	0.784638	3.307959	0.042493	0.072011	0.294831	0.007333
0.02500	2.500	0.774776	3.319771	0.018888	0.033195	0.134448	0.001670
0.01250	1.250	0.769149	3.319762	0.008845	0.015901	0.064073	0.000395

TABLE 9.1. $\|\mathbf{e}\|_2$ and $\|\mathbf{e}\|_\infty$ are measured in a quarter of the domain, $[0, 150] \times [0, 150]$. RMSE is measured in the region as shown in Fig. 9.1.

with OS method do decrease linearly and smaller than errors from ADI method. It can be seen that for ADI method, RMSE results exhibit $O(h) + O(\Delta\tau)$ convergence. On the other hand, for OS method, RMSE results exhibit $O(h^2) + O(\Delta\tau^2)$ convergence.





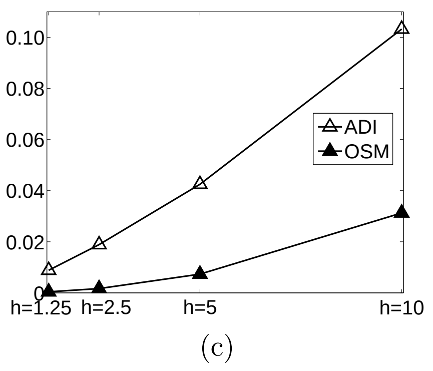


FIGURE 9.4. Comparison with (a) $\|\mathbf{e}\|_2$, (b) $\|\mathbf{e}\|_\infty$, and (c) RMSE of ADI and OSM. Lines with symbols, '□', '○', and '△' represent results with ADI and symbols, '■','●', and '▲' represent results with OSM.

Fig. 9.5 shows numerical results using ADI and OS methods for European call option on the maximum of two assets with a relatively large time step $\Delta\tau = 0.5$. The first and the second columns are results from ADI and OS methods, respectively. The first row (a) shows source term f at *Step 1*, Eq. (9.8). The result from ADI shows oscillation at the strike price $y = K$ due to the y-directional derivatives in the source term. Both ADI and OS schemes show oscillations at $x = y$ due to the cross

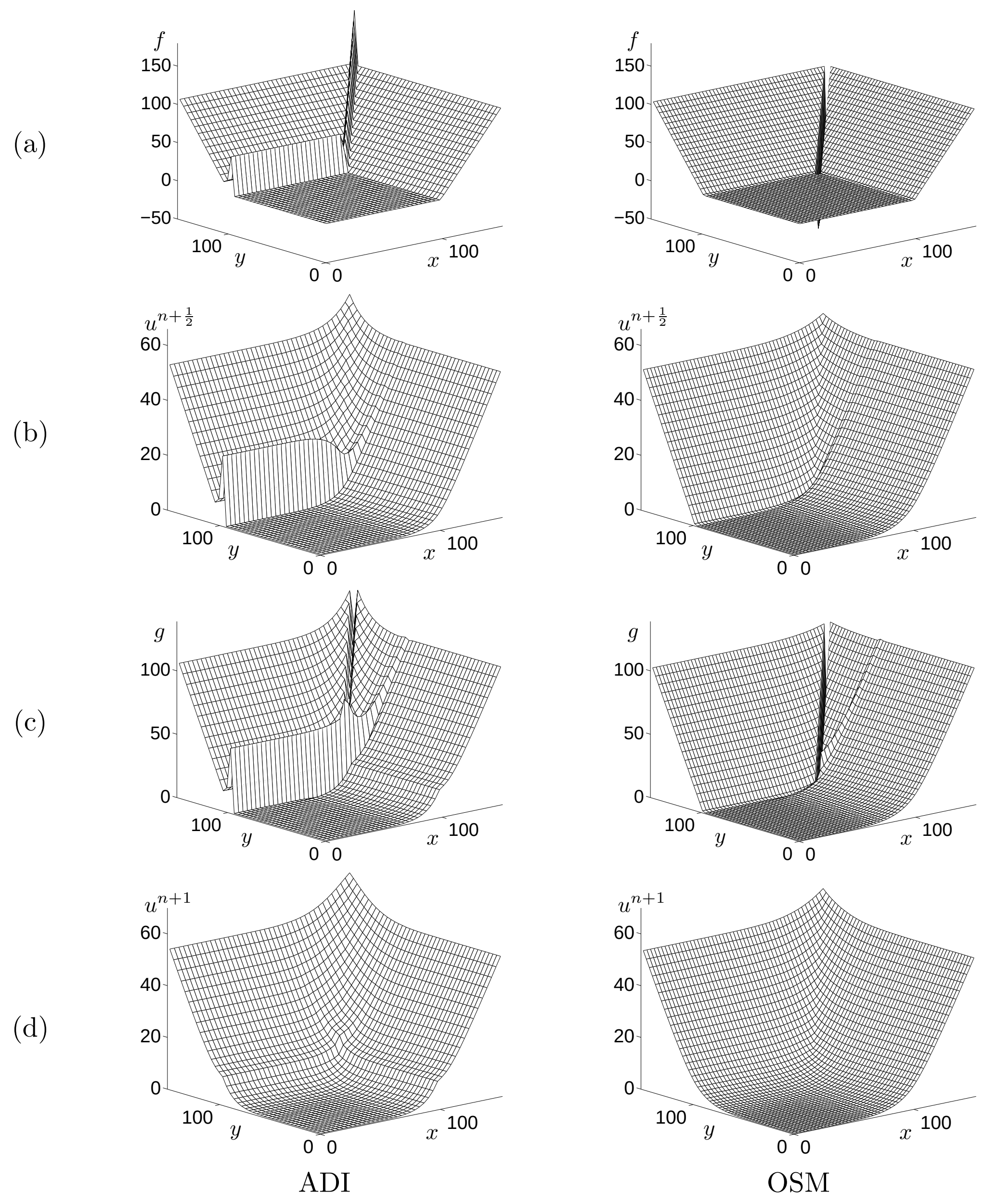


FIGURE 9.5. Comparison of numerical results of ADI and OS methods on the European call option on the maximum of two assets. (a) Source term f at *Step 1*, (b) solution $u^{n+\frac{1}{2}}$ after *Step 1*, (c) source term g at *Step 2*, and (d) solution u^{n+1} after *Step 2*.

derivatives. Intermediate solution results (b) demonstrate that these oscillations in the diagonal position do not make problem for both schemes. In Fig. 9.5(c), we observe that there is discontinuous profile at the strike price $x = K$ in the case of ADI. On the other hand, we see smooth profile with OS method. Finally, solution u^{n+1} after *Step*

2 suggests that OS method performs better than ADI with a large time step size (see Fig. 9.5(d)).

9.3.2. Cash or nothing option. Next, let us consider the European two-asset cash or nothing call option (see [87]). Given two stock prices x and y, the payoff of the call option is

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

where K_1 and K_2 are the strike prices of x and y, respectively (see Fig. 9.6).

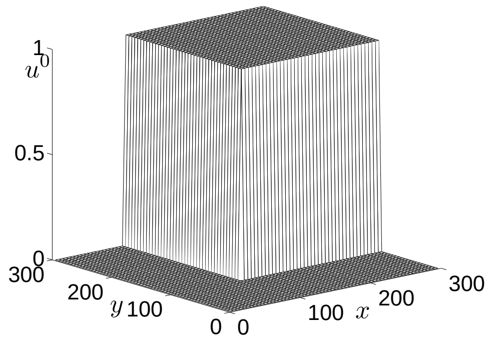


FIGURE 9.6. Payoff of a two-asset cash or nothing option on a computational domain $[0, 300] \times [0, 300]$.

The formula for the exact value is known in [87] by

$$u(x,y,T) = Ke^{-rT}M(\alpha, \beta; \rho), \tag{9.15}$$

where

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

The MATLAB code for the closed form solution of a two-asset cash or nothing call option is given in Sec. 7.4 in Chapter 7. The following parameters have been used. Time to expiry, $T = 0.5$, cash, $K = 1$, strike prices, $K_1 = K_2 = 100$, risk-free interest rate, $r = 0.03$, volatilities, $\sigma_1 = \sigma_2 = 0.3$, correlation coefficient, $\rho = 0.5$, and domain size $[0, 300] \times [0, 300]$ are used.

Time	Space	ADI			OSM		
$\Delta\tau$	h	$\|\mathbf{e}\|_2$	$\|\mathbf{e}\|_\infty$	RMSE	$\|\mathbf{e}\|_2$	$\|\mathbf{e}\|_\infty$	RMSE
0.10000	10.000	0.010196	0.041681	0.002564	0.004887	0.019115	0.000689
0.05000	5.000	0.009769	0.041316	0.001239	0.002043	0.007660	0.000126
0.02500	2.500	0.009530	0.040797	0.000610	0.000960	0.003310	0.000030
0.01250	1.250	0.009410	0.040564	0.000303	0.000469	0.001710	0.000008

TABLE 9.2. $\|\mathbf{e}\|_2$ and $\|\mathbf{e}\|_\infty$ are measured in a quarter of the domain, $[0, 150] \times [0, 150]$. RMSE is measured in the region as shown in Fig. 9.1.

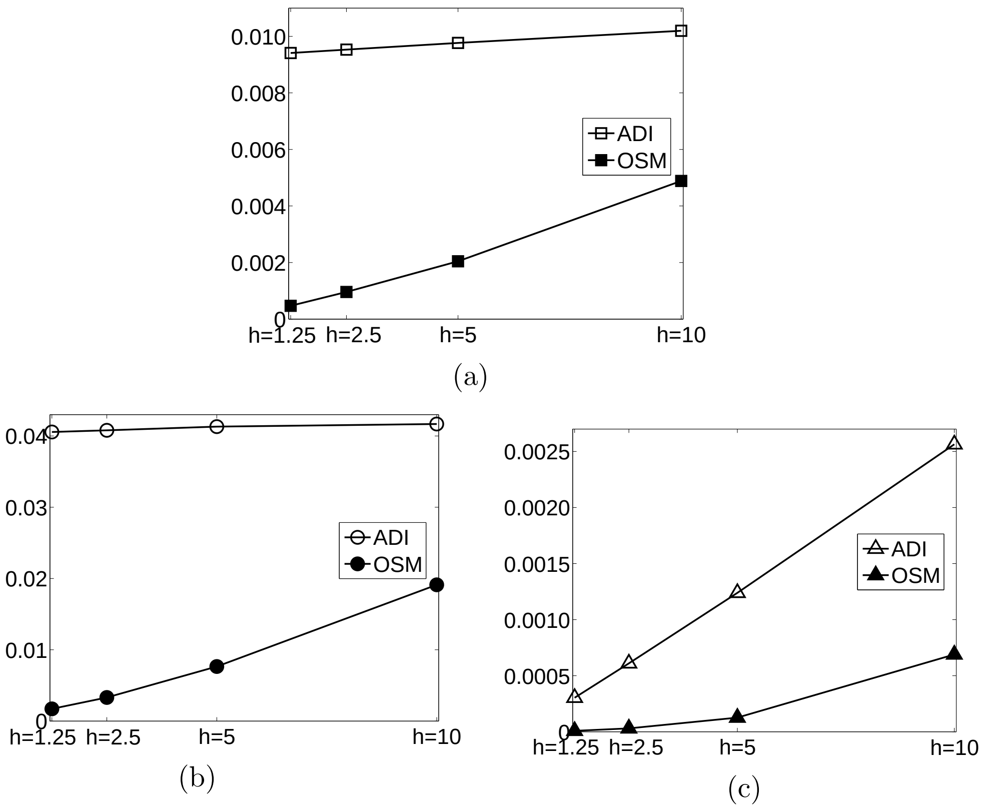


FIGURE 9.7. Comparison with (a) $\|\mathbf{e}\|_2$, (b) $\|\mathbf{e}\|_\infty$, and (c) RMSE of ADI and OSM. Lines with symbols, '□', '○', and '△' represent results with ADI and symbols,'■','●', and '▲' represent results with OSM.

Table 9.2 and Fig. 9.7 show the comparison of errors for ADI and OS methods at time $T = 0.5$. The errors $\|\mathbf{e}\|_2$ and $\|\mathbf{e}\|_\infty$ with ADI method do not decrease with increasing number of grid points and reduced time steps. On the other hand, the errors with OS method do decrease linearly and smaller then errors from ADI method. It can be seen that for ADI method, RMSE results exhibit $O(h) + O(\Delta\tau)$ convergence. On the other hand, for OS method, RMSE results exhibit $O(h^2) + O(\Delta\tau^2)$ convergence.

Similar to the result with European call option on the maximum of two assets, we have same result with two-asset cash or nothing call option as shown in Fig. 9.8.

9.4. Conclusions

In this Chapter, 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

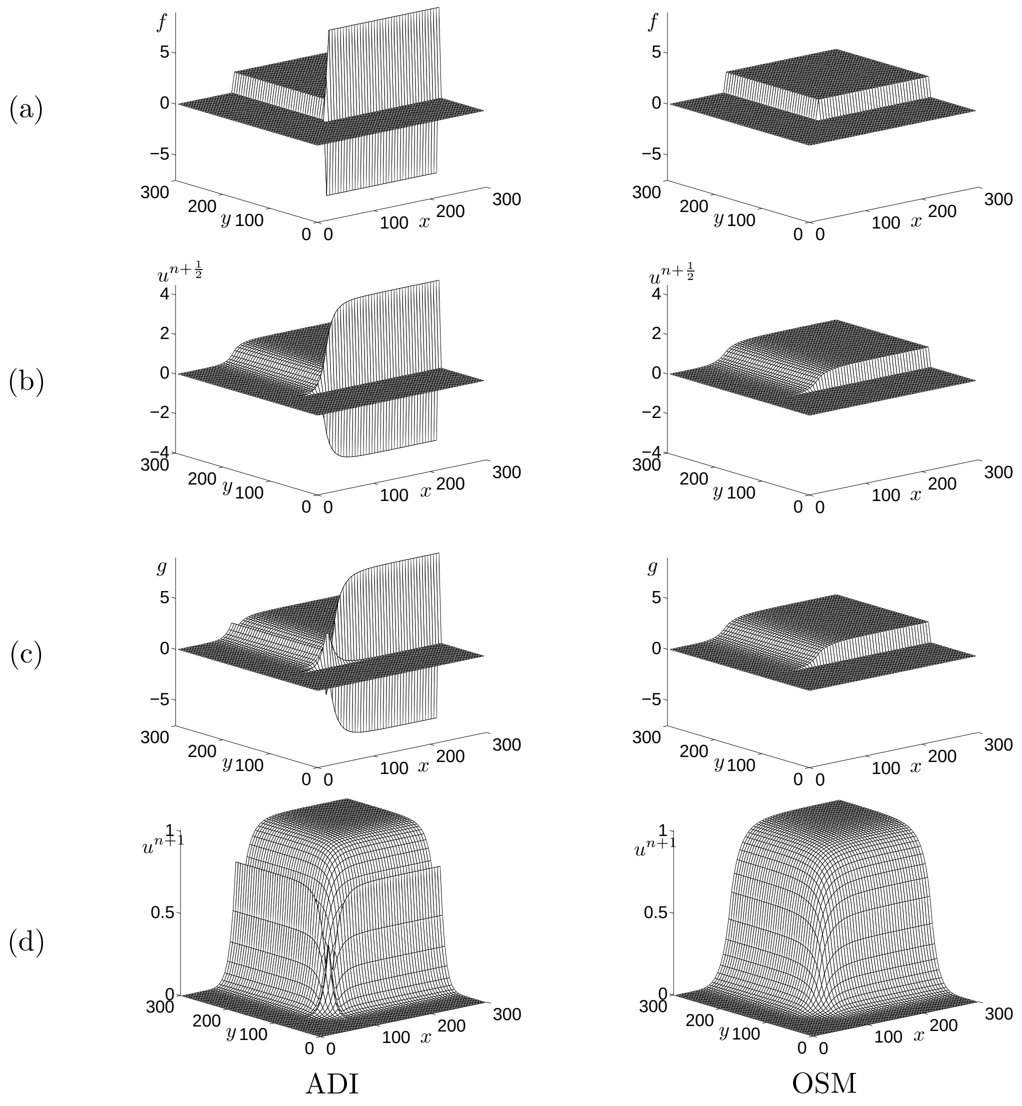


FIGURE 9.8. Comparison of numerical results of ADI and OS methods on the cash or nothing option. (a) Source term f at *Step 1*, (b) solution $u^{n+\frac{1}{2}}$ after *Step 1*, (c) source term g at *Step 2*, and (d) solution u^{n+1} after *Step 2*.

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.

Chapter 10

An operator splitting method for pricing the ELS option

This Chapter 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.

10.1. 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) [82, 86, 92, 94, 93, 95]. 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) [86, 83] 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. The rest of the Chapter is organized as follows. Section 10.2 provides a basic information and discuss the payoff of two-asset step-down ELS. Section 10.3 presents the finite difference discretizations for the BSPDE and a numerical solution algorithm using the OSM. Section 10.4 presents the computational results showing the performance of the method for option pricing problems with two underlying assets: cash-or-nothing and step-down ELS. Conclusions are presented in Section 10.5.

10.2. 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 10.1.

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 10.1. Parameters of two-asset step-down ELS.

Figure 10.1 shows the profit-and-loss diagram of two-asset step-down ELS.

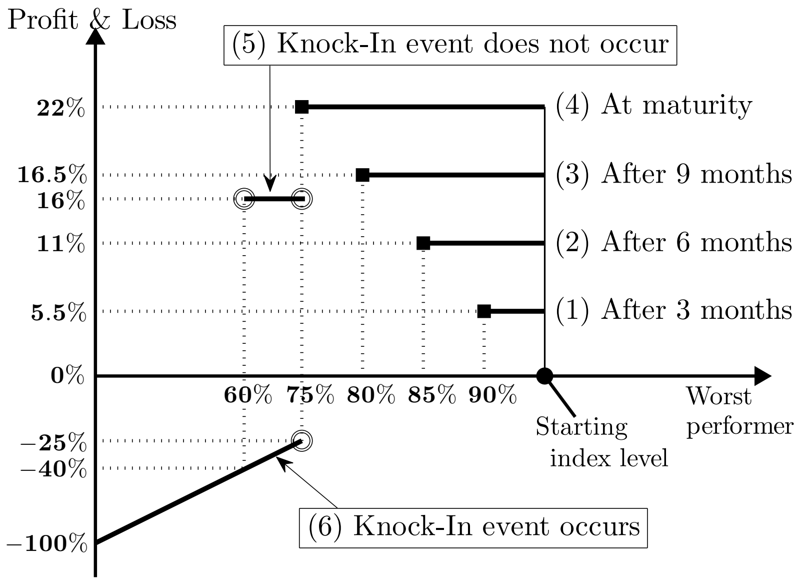


FIGURE 10.1. Profit-and-loss diagram at early redemption and maturity for two-asset step-down ELS.

10.3. Numerical solution

In this section, we describe the numerical discretization of Eq. (7.1). We also present the operator-splitting algorithm in detail.

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

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

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.

We use a cell-centered discretization because we use the following linear boundary condition:

$$u_{0j} = 2u_{1j} - u_{2j}, u_{N_x+1,j} = 2u_{N_x,j} - u_{N_x-1,j} \text{ for } j = 1, \cdots, N_y,$$
$$u_{i0} = 2u_{i1} - u_{i2}, u_{i,N_y+1} = 2u_{i,N_y} - u_{i,N_y-1} \text{ for } i = 1, \cdots, N_x.$$

10.3.2. Operator-splitting method. The basic idea behind the operator-splitting method is to reduce multi-dimensional equations into multiple one-dimensional problems [86, 83]. We introduce the basic OS scheme for the two-dimensional Black-Scholes equation as follows:

$$\frac{u_{ij}^{n+1} - u_{ij}^{n}}{\Delta\tau} = \mathcal{L}_{BS}^{x} u_{ij}^{n+\frac{1}{2}} + \mathcal{L}_{BS}^{y} u_{ij}^{n+1}, \tag{10.1}$$

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

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

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:

$$\frac{u_{ij}^{n+\frac{1}{2}} - u_{ij}^{n}}{\Delta\tau} = \mathcal{L}_{BS}^{x} u_{ij}^{n+\frac{1}{2}}, \tag{10.2}$$

$$\frac{u_{ij}^{n+1} - u_{ij}^{n+\frac{1}{2}}}{\Delta\tau} = \mathcal{L}_{BS}^{y} u_{ij}^{n+1}. \tag{10.3}$$

Note that combining two Eqs. (10.2) and (10.3) results in Eq. (10.1). The following describes an algorithm of the OS method.

Algorithm OS

- *Step 1*

  Eq. (10.2) is rewritten as follows. For each j, we have

$$\alpha_i u_{i-1j}^{n+\frac{1}{2}} + \beta_i u_{ij}^{n+\frac{1}{2}} + \gamma_i u_{i+1j}^{n+\frac{1}{2}} = f_{ij}, \tag{10.4}$$

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

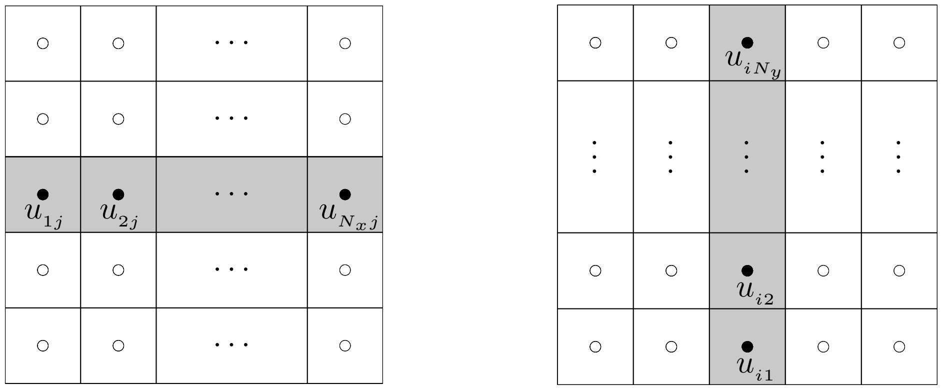
$$f_{ij} = \lambda_1 \rho \sigma_1 \sigma_2 x_i y_j \frac{u^n_{i+1,j+1} - u^n_{i+1,j} - u^n_{i,j+1} + u^n_{ij}}{h^2} + \frac{u^n_{ij}}{\Delta\tau}. \qquad (10.5)$$

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. (10.5)
    end
    Solve $A_x u^{n+\frac{1}{2}}_{1:N_x,j} = f_{1:N_x,j}$ by using Thomas algorithm (see Fig. 10.2(a))
end
```

Here the matrix A_x is a tridiagonal matrix constructed from Eq. (10.4) 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 10.2. Two steps of the OSM.

- *Step 2*

As in *Step 1*, Eq. (10.3) is rewritten as follows:

$$\alpha_j u_{i,j-1}^{n+1} + \beta_j u_{ij}^{n+1} + \gamma_j u_{i,j+1}^{n+1} = g_{ij}, \tag{10.6}$$

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

$$g_{ij} = (1-\lambda_1)\rho\sigma_1\sigma_2 x_i y_j \frac{u_{i+1,j+1}^{n+\frac{1}{2}} - u_{i+1,j}^{n+\frac{1}{2}} - u_{i,j+1}^{n+\frac{1}{2}} + u_{ij}^{n+\frac{1}{2}}}{h^2} + \frac{u_{ij}^{n+\frac{1}{2}}}{\Delta\tau}. \tag{10.7}$$

As with *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 $g_{ij}$ by Eq. (10.7)
  end
  Solve $A_y u_{i,1:N_y}^{n+1} = g_{i,1:N_y}$ by using Thomas algorithm (see Fig. 10.2(b))
end
```

Here A_y is tridiagonal matrix constructed from Eq. (10.6) with a linear boundary condition

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

10.4. 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.

10.4.1. 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 [87]. Given two stock prices x and y, the payoff of the call option is

$$u(x,y,0) = \begin{cases} \text{Cash} & \text{if} \quad x \geq K_1 \text{ and } y \geq K_2, \\ 0 & \text{otherwise,} \end{cases} \tag{10.8}$$

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 [87]. 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. (10.8) 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, $\|\mathbf{e}\|_2$.

Table 10.2 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. 9.1 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 10.2. Convergence test.

10.4.2. 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. 10.3(a) and (b) show the initial configurations of u and v, respectively.

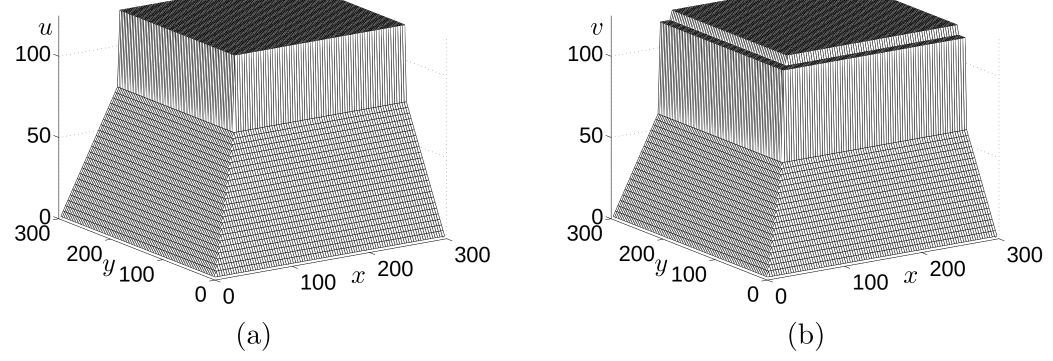
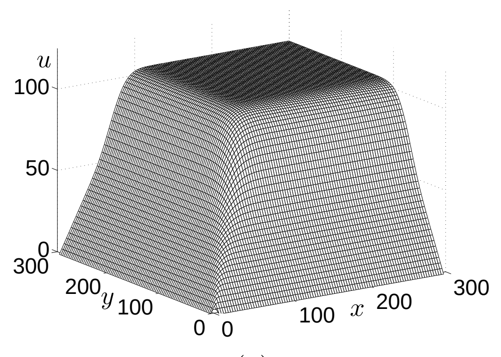
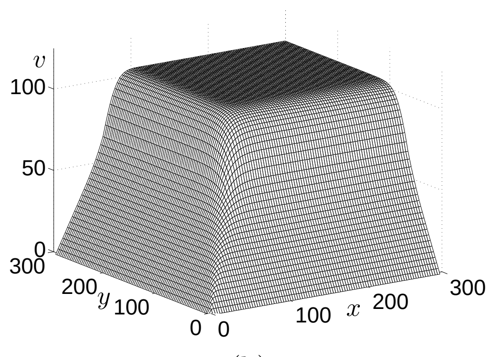


FIGURE 10.3. (a) and (b) are the initial conditions for u and v, respectively.

And Fig. 10.4(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. 10.1.

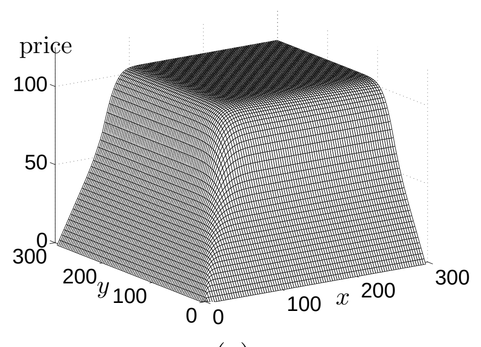
The final two-asset step-down ELS price is obtained by a weighted average of u and v by each probability. By performing a Monte Carlo (MC) simulation [84] for 20000 samples, we estimated that a knock-in event occurs with a probability of approximately 0.1. Therefore, we defined the final ELS value as $0.1u + 0.9v$. Fig. 10.5 (a) shows the weighted average value $0.1u + 0.9v$, and (b) shows the overlapped contour lines of the weighted average values.

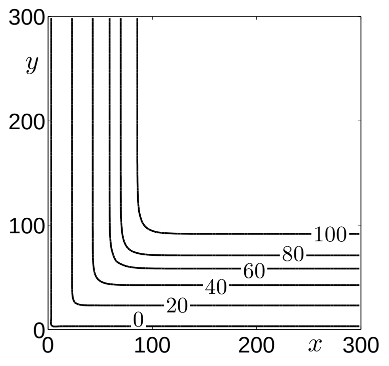




(a) (b)

FIGURE 10.4. (a) and (b) are the numerical results for u and v, respectively, at $T = 1$.





(a) (b)

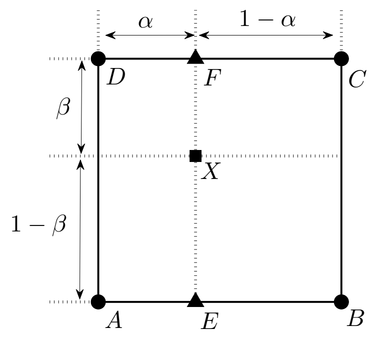
FIGURE 10.5. (a) The weighted average value $0.1u+0.9v$ at $T = 1$. (b) The contour lines of the weighted average values.

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. 10.6, we obtained the numerical values at the specific point X by using the bilinear interpolation.

Table 10.3 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 10.3. 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.



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

FIGURE 10.6. 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.

Fig. 10.7 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.

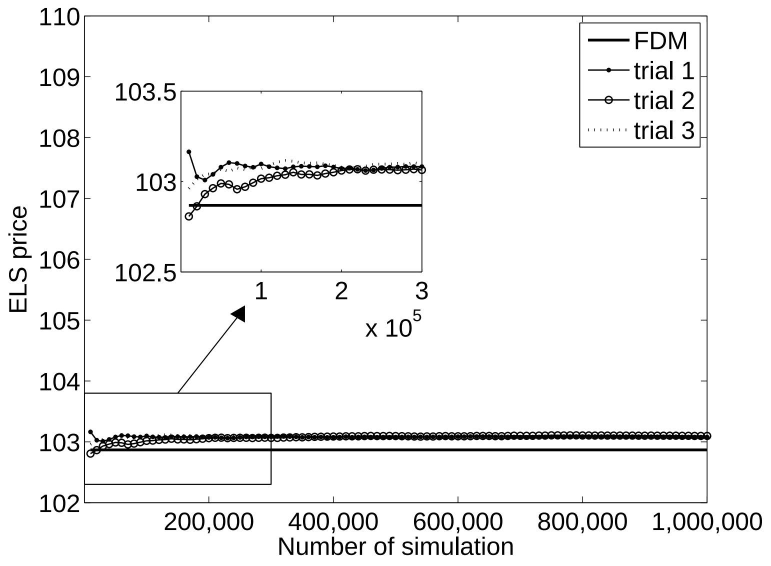


FIGURE 10.7. Two-asset step-down ELS price obtained using the OSM and the Monte-Carlo simulation versus the number of simulations.

10.5. 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 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.

This chapter is published in Journal of the Korean Society for Industrial and Applied Mathematics, Vol. 14, No. 3, pp. 175-187, (2010).

Chapter 11

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

This Chapter presents an accurate and efficient numerical method for the Black-Scholes equations. The method uses an adaptive grid technique which is based on a far-field boundary position of the equation. Numerical tests are presented to demonstrate the accuracy and efficiency of the method. 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.

11.1. Introduction

In this Chapter, we consider an accurate and efficient numerical method for the Black-Scholes equation:

$$\frac{\partial u}{\partial t} = -\frac{1}{2}(\sigma x)^2 \frac{\partial^2 u}{\partial x^2} - rx\frac{\partial u}{\partial x} + ru, \ (x, \tau) \in (0, \infty) \times (0, T], \tag{11.1}$$

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 [85]. By changing variable to $\tau = T - t$, we obtain the following equation:

$$\frac{\partial u}{\partial \tau} = \frac{1}{2}(\sigma x)^2 \frac{\partial^2 u}{\partial x^2} + rx\frac{\partial u}{\partial x} - ru \tag{11.2}$$

with an initial condition $u(x, 0) = p(x)$. For example, for the call option, the payoff function is $p(x) = \max(x - K, 0)$ with a given strike price K [87].

By extending one-dimension Black-Scholes Eq. (11.1) to two-dimensional problem, we can written as

$$\frac{\partial u}{\partial \tau} = BSu, \quad (x, y, \tau) \in (0, \infty) \times (0, \infty) \times (0, T], \tag{11.3}$$

where BS is the operator

$$BS = \frac{1}{2}\sigma_1^2 x^2 \frac{\partial^2}{\partial x^2} + \frac{1}{2}\sigma_2^2 y^2 \frac{\partial^2}{\partial y^2} + \rho\sigma_1\sigma_2 xy\frac{\partial^2}{\partial x \partial y} + rx\frac{\partial}{\partial x} + ry\frac{\partial}{\partial y} - r.$$

The partial derivatives in Eq. (11.3) are defined in the unbounded domain

$$\{(x, y, t) | x \geq 0, \ y \geq 0, \ t \in [0, T]\}.$$

We truncate this domain into a finite computational domain

$$(x, y, t) \in [0, L] \times [0, M] \times [0, T],$$

where L and M are large enough so that the error in the price u due to the truncation is negligible. On the boundary, we apply the linear boundary conditions,

The finite difference method is applied to the equation over a truncated finite domain and 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. The purpose of this work is to propose an adaptive grid distribution depending on a far-field boundary position to solve the Black-Scholes equation accurately and efficiently.

The outline of the Chapter is the following. In Sec. 11.2, we present the proposed numerical scheme. Sec. 11.3 presents the results of the numerical experiments. Conclusions are made in Sec. 11.4.

11.2. Numerical solution

11.2.1. Discretization with finite differences. The finite difference method approximates the derivatives by difference operators. This finite difference method is a common numerical method that has been used in computational finance by many authors. For an introduction to these methods we can recommend the books [86, 92, 93, 94, 95].

11.2.1.1. *One dimensional space.* 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. 11.1.

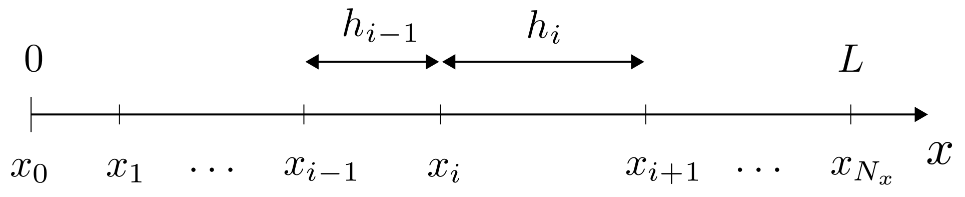


FIGURE 11.1. A non-uniform grid with space step sizes h_i.

Let us denote the numerical approximation of the solution by

$$u_i^n \equiv u(x_i, \tau^n), \text{for } i = 0, \dots, N_x \text{ and } n = 0, \dots, N_\tau,$$

where $\tau^n = n\Delta\tau$, $\Delta\tau = T/N_\tau$, and N_τ is the total number of time steps. By applying the implicit scheme to Eq. (11.2), we have

$$\frac{u_i^{n+1} - u_i^n}{\Delta\tau} = BSu_i^{n+1}, \tag{11.4}$$

where the discrete difference operator BSu_i^{n+1} is defined by

$$BSu_i^{n+1} = \frac{\sigma^2 x_i^2}{2}\left(\frac{\partial^2 u}{\partial x^2}\right)_i^{n+1} + rx_i\left(\frac{\partial u}{\partial x}\right)_i^{n+1} - ru_i^{n+1}.$$

For the first and the second derivatives we have

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

Rewrite the above Eq. (11.4) by

$$\alpha_i u_{i-1}^{n+1} + \beta_i u_i^{n+1} + \gamma_i u_{i+1}^{n+1} = \frac{u_i^n}{\Delta\tau}, \tag{11.5}$$

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

Linear boundary condition [91, 94, 96] is defined by $\frac{\partial^2 u}{\partial x^2}(L,\tau) = 0,\ \forall\tau \in [0,T]$. We discretize the linear boundary condition as

$$\frac{u_{N_x-1}^{n+1} - 2u_{N_x}^{n+1} + u_{N_x+1}^{n+1}}{h^2} = 0. \tag{11.6}$$

By substituting Eq. (11.6) in Eq. (11.4) we can get

$$\frac{rx_{N_x}}{h}u_{N_x-1}^{n+1} + \left(\frac{1}{\Delta\tau} - \frac{rx_{N_x}}{h} + r\right)u_{N_x}^{n+1} = \frac{u_{N_x}^n}{\Delta\tau}. \tag{11.7}$$

And we can rewrite this Eq. (11.7) by matrix form

$$\begin{pmatrix} \beta_1 & \gamma_1 & 0 & \dots & 0 \\ \alpha_2 & \beta_2 & \gamma_2 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & \alpha_{N_x-1} & \beta_{N_x-1} & \gamma_{N_x-1} \\ 0 & \dots & 0 & \alpha_{N_x} & \beta_{N_x} \end{pmatrix} \begin{pmatrix} u_1^{n+1} \\ u_2^{n+1} \\ \vdots \\ u_{N_x-1}^{n+1} \\ u_{N_x}^{n+1} \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_{N_x-1} \\ b_{N_x} \end{pmatrix},$$

where $\alpha_{N_x} = rN_x$, $\beta_{N_x} = \frac{1}{\Delta\tau} - rN_x + r$, and $b_{N_x} = \frac{u_{N_x}^n}{\Delta\tau}$.

As one dimensional space, the BS equation is discretized on a non-uniform grid defined by $x_0 = y_0 = 0$, $x_{i+1} = x_i + h_i$ and $y_{j+1} = y_j + h_j$ 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 intervals and h_i, h_j are the grid spacing. Fig. 11.2 (a) and (b) represent such assumptions.

The numerical approximations of the solution are denoted by

$$u_{ij}^n \equiv u(x_i, y_j, \tau^n),$$

where $i = 0, \cdots, N_x$, $j = 0, \cdots, N_y$ and $n = 0, \cdots, N_t$. By applying the implicit scheme to Eq. (11.3), we have

$$\frac{u_{ij}^{n+1} - u_{ij}^n}{\Delta\tau} = BSu_{ij}^{n+1}, \tag{11.8}$$

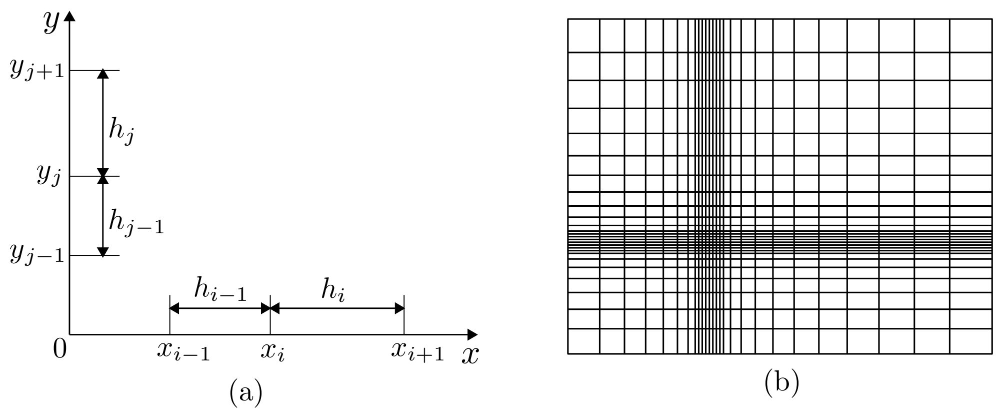


FIGURE 11.2. (a) A non-uniform grid with space step size h_i and h_j. (b) Two-dimensional adaptive mesh.

where the discrete difference operator BSu_{ij}^{n+1} is defined by

$$BSu_{ij}^{n+1} = \frac{\sigma_1^2 x_i^2}{2}\left(\frac{\partial^2 u_{ij}}{\partial x_i^2}\right)^{n+1} + \frac{\sigma_2^2 y_j^2}{2}\left(\frac{\partial^2 u_{ij}}{\partial y_j^2}\right)^{n+1} + \rho\sigma_1\sigma_2 x_i y_j\left(\frac{\partial^2 u_{ij}}{\partial x_i \partial y_j}\right)^{n+1} \quad (11.9)$$

$$+ \; r x_i\left(\frac{\partial u_{ij}}{\partial x_i}\right)^{n+1} + r y_j\left(\frac{\partial u_{ij}}{\partial y_j}\right)^{n+1} - r u_{ij}^{n+1}. \quad (11.10)$$

For the first and second derivatives we have

$$\left(\frac{\partial u}{\partial x}\right)_i^{n+1} \approx \frac{h_{i-1}}{h_i(h_{i-1}+h_i)}u_{i+1,j}^{n+1} + \frac{h_i - h_{i-1}}{h_{i-1}h_i}u_{ij}^{n+1} - \frac{h_i}{h_{i-1}(h_{i-1}+h_i)}u_{i-1,j}^{n+1},$$

$$\left(\frac{\partial^2 u}{\partial x^2}\right)_i^{n+1} \approx \frac{2}{h_i(h_{i-1}+h_i)}u_{i+1,j}^{n+1} - \frac{2}{h_{i-1}h_i}u_{ij}^{n+1} + \frac{2}{h_{i-1}(h_{i-1}+h_i)}u_{i-1,j}^{n+1},$$

$$\left(\frac{\partial^2 u}{\partial x \partial y}\right)^{n+1} \approx \frac{u_{i+1,j+1}^{n+1} - u_{i+1,j}^{n+1} - u_{i,j+1}^{n+1} + u_{ij}^{n+1}}{h_i h_j}.$$

Now, to solve the two-dimensional Black-Scholes equation, we use the basic operator splitting(OS) methods. The OS scheme is to divide each time step into fractional time steps with simpler operators. Therefore, the basic idea behind OS method is to replace a two-dimensional scheme as

$$\frac{u_{ij}^{n+1} - u_{ij}^{n}}{\Delta\tau} = \mathcal{L}_{BS}^{x} u_{ij}^{n+\frac{1}{2}} + \mathcal{L}_{BS}^{y} u_{ij}^{n+1}, \quad (11.11)$$

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

$$\begin{aligned}
\mathcal{L}^x_{BS}u_{ij}^{n+\frac{1}{2}} &= \sigma_1^2 x_i^2 \left(\frac{u_{i-1,j}^{n+\frac{1}{2}}}{h_{i-1}(h_{i-1}+h_i)} - \frac{u_{ij}^{n+\frac{1}{2}}}{h_{i-1}h_i} + \frac{u_{i+1,j}^{n+\frac{1}{2}}}{h_i(h_{i-1}+h_i)} \right) \\
&+\lambda_1\sigma_1\sigma_2\rho x_i y_j \frac{u^n_{i+1,j+1}+u^n_{ij}-u^n_{i,j+1}-u^n_{i+1,j}}{h_i h_j} \\
&+r x_i \left(-\frac{h_i u_{i-1,j}^{n+\frac{1}{2}}}{h_{i-1}(h_{i-1}+h_i)} + \frac{(h_i-h_{i-1})u_{ij}^{n+\frac{1}{2}}}{h_{i-1}h_i} + \frac{h_{i-1}u_{i+1,j}^{n+\frac{1}{2}}}{h_i(h_{i-1}+h_i)} \right) \\
&-\lambda_2 r u_{ij}^{n+\frac{1}{2}}, \qquad (11.12)
\end{aligned}$$

$$\begin{aligned}
\mathcal{L}^y_{BS}u_{ij}^{n+1} &= \sigma_2^2 y_j^2 \left(\frac{u_{i,j-1}^{n+1}}{h_{j-1}(h_{j-1}+h_j)} - \frac{u_{ij}^{n+1}}{h_{j-1}h_j} + \frac{u_{i,j+1}^{n+1}}{h_j(h_{j-1}+h_j)} \right) \\
&+(1-\lambda_1)\sigma_1\sigma_2\rho x_i y_j \frac{u^{n+\frac{1}{2}}_{i+1,j+1}+u^{n+\frac{1}{2}}_{ij}-u^{n+\frac{1}{2}}_{i,j+1}-u^{n+\frac{1}{2}}_{i+1,j}}{h_i h_j} \\
&+r y_j \left(-\frac{h_j u_{i,j-1}^{n+1}}{h_{j-1}(h_{j-1}+h_j)} + \frac{(h_j-h_{j-1})u_{ij}^{n+1}}{h_{j-1}h_j} + \frac{h_{j-1}u_{i,j+1}^{n+1}}{h_j(h_{j-1}+h_j)} \right) \\
&-(1-\lambda_2) r u_{ij}^{n+1}, \qquad (11.13)
\end{aligned}$$

where $h_i = x_{i+1} - x_i, h_j = y_{j+1} - y_j$.

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 space two one-dimensional problems. We then approximate each sub-problem by implicit or explicit schemes. Thus, we are considering of two one-dimensional discrete equations.

$$\frac{u_{ij}^{n+\frac{1}{2}} - u_{ij}^n}{\Delta\tau} = \mathcal{L}^x_{BS}u_{ij}^{n+\frac{1}{2}}, \qquad (11.14)$$

$$\frac{u_{ij}^{n+1} - u_{ij}^{n+\frac{1}{2}}}{\Delta\tau} = \mathcal{L}^y_{BS}u_{ij}^{n+1}. \qquad (11.15)$$

Note that the addition of two Eqs. (11.14) and (11.15) results in Eq. (11.11). We have the linear boundary conditions

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

Given the solution u^n, we want to find u^{n+1} which satisfies Eq. (11.11).

Algorithm

Step 1. Equation (11.14) is rewritten as follows.

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

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

```
for $j = 1, ..., N_y$
 for $i = 1, ..., N_x$
 $f_{ij} = \lambda_1 \rho \sigma_1 \sigma_2 x_i y_j \frac{u_{i+1j+1}^n - u_{i+1j}^n - u_{ij+1}^n + u_{ij}^n}{h_i h_j} + \frac{u_{ij}^n}{\Delta\tau}$
 end
 solve $A_x u_{1:Nx,j}^{n+\frac{1}{2}} = f_{1:Nx,j}$ by using a Thomas algorithm
end
```

Here the matrix A_x is a tridiagonal matrix,

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

and the elements of the matrix are

$$\begin{aligned}
\beta_1 &= \frac{1}{\Delta\tau} + \frac{\sigma_1^2 x_1^2}{h_0 h_1} + \frac{r x_1}{h_1} + \lambda_2 r - \frac{2\sigma_1^2 x_1^2}{h_0(h_0+h_1)}, \\
\gamma_1 &= -\frac{\sigma_1^2 x_1^2}{h_1(h_0+h_1)} - \frac{r x_1}{h_1} + \frac{\sigma_1^2 x_1^2}{h_0(h_0+h_1)}, \\
\alpha_i &= -\frac{\sigma_1^2 x_i^2}{h_{i-1}(h_{i-1}+h_i)}, \beta_i = \frac{1}{\Delta\tau} + \frac{\sigma_1^2 x_i^2}{h_{i-1} h_i} + \frac{r x_i}{h_i} + \lambda_2 r, \\
\gamma_i &= -\frac{\sigma_1^2 x_i^2}{h_i(h_{i-1}+h_i)} - \frac{r x_i}{h_i}, \text{ for } i = 2, ..., N_x - 1. \\
\alpha_{N_x} &= -\frac{\sigma_1^2 x_{N_x}^2}{h_{N_x-1}(h_{N_x-1}+h_{N_x})} + \frac{\sigma_1^2 x_{N_x}^2}{h_{N_x}(h_{N_x-1}+h_{N_x})} + \frac{r x_{N_x}}{h_{N_x}}, \\
\beta_{N_x} &= \frac{1}{\Delta\tau} + \frac{\sigma_1^2 x_{N_x}^2}{h_{N_x-1} h_{N_x}} - \frac{r x_{N_x}}{h_{N_x}} + \lambda_2 r - \frac{2\sigma_1^2 x_{N_x}^2}{h_{N_x}(h_{N_x-1}+h_{N_x})}.
\end{aligned}$$

11.2.2. Space adaptivity. In this section we propose an adaptive grid technique which is based on a far-field boundary position of the equation. In [91], authors used an adaptive technique based on a local discretization error. Adaptive finite element [89] and finite difference [90] methods were used for American options.

We will start with a one-dimensional problem. In the case of a European call, with boundary condition at $x = S_{\max}$ set to the payoff, for $x \in (0, K)$, $u(x,\tau)$ and $w(x,\tau)$ satisfy the following inequality:

$$\begin{aligned} &|u(x,\tau) - w(x,\tau)| \\ &\le (S_{\max} + K)\exp\left(-\frac{\ln\frac{S_{\max}}{K}\left(\ln\frac{S_{\max}}{K} + \min\{0, \sigma^2 - 2r\}\tau\right)}{2\sigma^2\tau}\right) \le \frac{K}{A} \end{aligned}$$

where K/A is the maximum truncation error. By dividing both sides by K and replacing $S_{\max}/K$ to α, we rewrite the above inequality as

$$(\alpha + 1)\exp\left(-\frac{\ln\alpha\left(\ln\alpha + (\sigma^2 - 2r)\tau\right)}{2\sigma^2\tau}\right) \le \frac{1}{A}, \tag{11.16}$$

where we assumed $\sigma^2 - 2r < 0$. By rearranging inequality (11.16), we obtain

$$(\ln\alpha)^2 + (\sigma^2 - 2r)\tau\ln\alpha \ge 2\sigma^2\tau\ln(\alpha+1) + 2\sigma^2\tau\ln A \tag{11.17}$$

Since $\ln x$ is increasing function, inequality (11.17) is rewritten as follow:

$$(\ln\alpha)^2 + (\sigma^2 - 2r)\tau\ln\alpha \ge 2\sigma^2\tau\ln(\alpha) + 2\sigma^2\tau\ln A. \tag{11.18}$$

$$(\ln\alpha)^2 - (\sigma^2 + 2r)\tau\ln\alpha - 2\sigma^2\tau\ln A \ge 0. \tag{11.19}$$

The graphs in Fig. 11.3 represent $\ln\alpha$ of the above inequalities (11.17) and (11.18) with $\sigma = 0.3$, $r = \log(1.1)$, $T = 2$, $K = 50$, $A = 100$. As you can see in Fig. 11.3, inequalities (11.17) and (11.18) have almost similar solutions. So we can determine $S_{\max}$ by using the solution of inequality (11.18) without significant difference. This estimate tells us that if

$$S_{\max} \ge K e^{\left(0.5\sigma^2 + r\right)\tau + 0.5\sqrt{(\sigma^2+2r)^2\tau^2 + 8\sigma^2\tau\ln A}},$$

then we can be sure that $w(x,\tau)$, the solution of the truncated problem, gives us a value for the call option today that is within K/A from the correct value, provided the price of the underlying asset is not greater than the exercise price K [88].

For options with two underlying assets, we have

$$S_{\max} \ge K e^{\left(0.5\sigma^2 + r\right)\tau + 0.5\sqrt{(\sigma^2+2r)^2\tau^2 + 8\sigma^2\tau\ln(2A)}},$$

The adaptive process aims at creating a grid with a uniform and fine grid around the strike price and increasing grid sizes toward the far field boundary. To do this, we define a grid function $h(x)$ as

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

where p and d are real positive numbers.

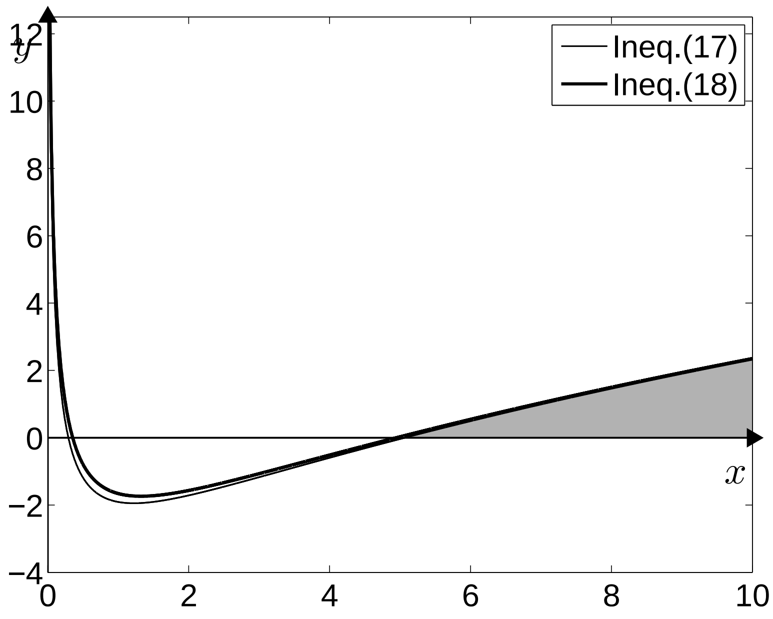


FIGURE 11.3. The graph of the inequalities. (11.17) and (11.18) with $\sigma = 0.3,\ r = \log(1.1),\ T = 2,\ K = 50,\ A = 100$.

First, we allocate $2m$ grid points around the strike price K with the grid size $\bar{h}$. We start at $x_i = K + (m - 0.5)\bar{h}$ and look at the discrete function value $h(x_i)$. And define $x_{i+1} = x_i + h(x_i)$. We proceed this process until we get $N_x - 1$ such that $x_{N_x-1} \le S_{max} < x_{N_x}$. For the left side of grid generation, we start at $x_i = K - (m - 0.5)\bar{h}$ and define $x_{i-1} = x_i + h(x_i)$. We proceed this process until we get $x_0 \le 0 < x_1$. If $x_0 < 0$, then we redefine $x_0 = 0$. This procedure is schematically described in Fig. 11.4.

11.3. Computational results

In this section, we perform numerical experiments. The main focus in the numerical tests is on the performance of the proposed adaptive grid technique over the normal uniform grid method.

For a European call option, we have the closed form solution for the Black-Scholes equation as

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

where $N(d) = (1/\sqrt{2\pi})\int_{-\infty}^{d} \exp\left(-x^2/2\right) dx$ is the standard normal distribution function [85].

11.3.1. Uniform gird. In this section, the effect and accuracy of the far-field boundary condition using European call option is studied. The parameters used are: $\sigma = 0.35, r = 0.05$, and space size $h = 1$. We considered a domain, $\Omega = (0, L)$. For

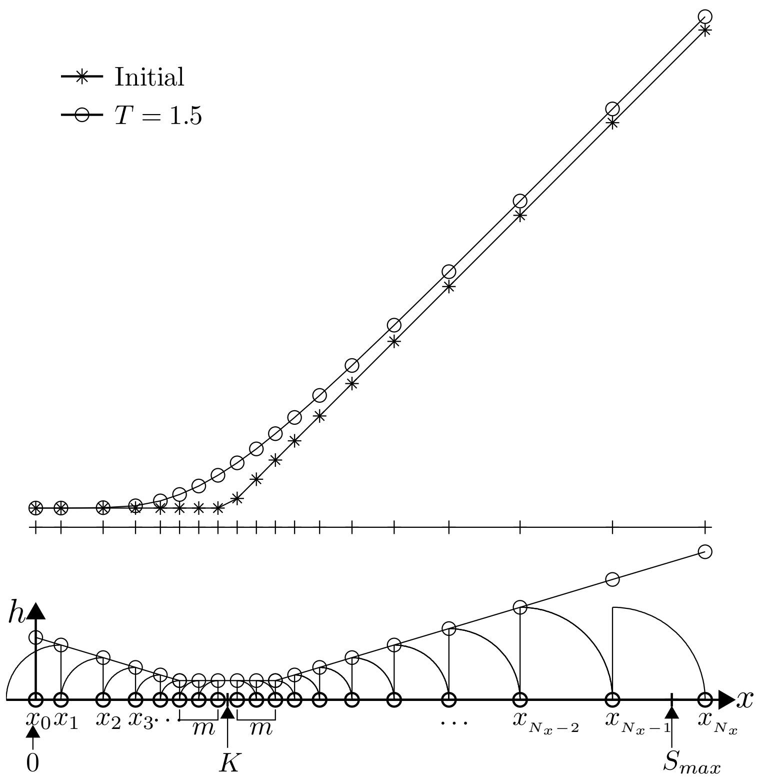


FIGURE 11.4. Constructing the adaptive grid using the function $h(x)$.

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

Fig. 11.5(a), (b), and (c) show the initial profile, numerical solutions at time T with different boundary conditions, and exact solutions on each domain $L = 150,\ 200,$ and 300, respectively. When the domain size is $L = 150$, we can observe a large deviation of numerical solutions from the exact solution. And when we increase the domain size by $L = 200$, we have a good result from all four boundary conditions. However, when we increase the time $T = 5$, we again have a large deviation. This result implies that we need a large enough domain size which depends on T.

11.3.2. Adaptive grid. Fig. 11.6 represents the result of the relative root mean square error (RMSE) on $[0.9K,\ 1.1K]$ with different N_x and time $T = 1$. For the adaptive grid generation, we use $p = 0.1$ and $d = 1$. Here, the relative RMSE is defined as

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

where N is the number of points on $[0.9K,\ 1.1K]$ and u is the exact solution. In Fig. 11.6, '∘','□', and '◇' show the result using $\bar{h} = 1,\ 0.5,$ and 0.25 on adaptive mesh, respectively and '•','■', and '♦' represent $h = 1,\ 0.5,$ and 0.25 on uniform mesh,

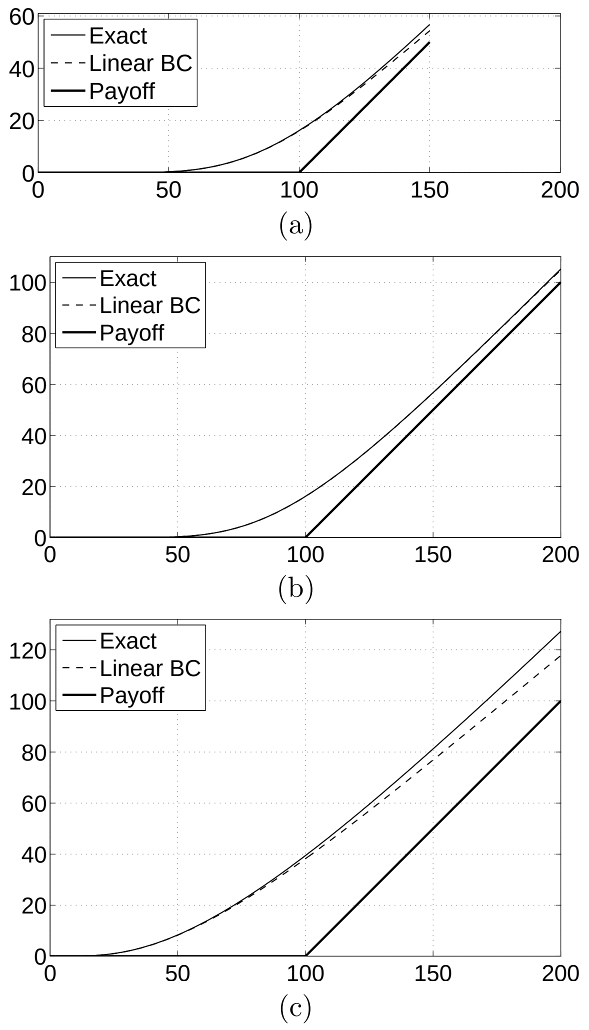


FIGURE 11.5. Different boundary conditions with domain and time (a) $L = 150$, $T = 1$ (b) $L = 200$, $T = 1$, and (c) $L = 200$, $T = 5$.

respectively. From the results, we see the convergence of the relative RMSE to the uniform grid as the number of grid points around the strike price K. We note that if we compare results from a uniform grid h and adaptive grid $\bar{h} = 0.5h$ with smaller grid points N_x, we can see the better performance of the adaptive grid technique.

Similarly, Fig. 11.7 shows the result with longer total time $T = 3$. From this result, we confirm the effectiveness of the adaptive grid method.

Table 11.1 represents computational results such as relative CPU time, RMSE on $[0.9K,\ 1.1K]$ and grid points N_x at time $T = 1$ on adaptive and uniform grids. We use a space step h in case of uniform grid and $\bar{h} = 0.5h$ in case of adaptive grid. For fair comparison test, we matched the relative RMSE of both methods. The CPU time taken from uniform grids are greater than the adaptive grid method. Also, total

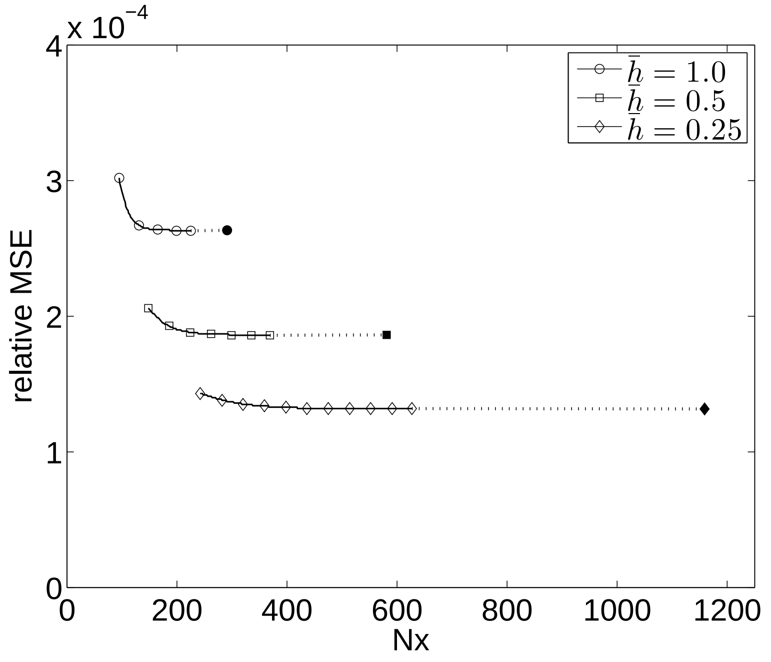


FIGURE 11.6. Relative RMSE on $[0.9K,\ 1.1K]$ with different N_x and $T = 1$. Lines with symbols, '○','□', and '◇' represent $\bar{h} = 1,\ 0.5$, and 0.25 on adaptive mesh, respectively. Also, symbols, '●','■', and '◆' represent $h = 1,\ 0.5$, and 0.25 on uniform mesh, respectively.

number of grid points N_x on adaptive grid is much smaller than uniform grid. Overall, the performance of the adaptive grid outperforms the uniform grid.

	Case	$h = 1$	$h = 0.5$	$h = 0.25$
CPU time	Adaptive	1	1	1
	Uniform	12.0021	32.3834	64.9243
RMSE	Adaptive	$2.62E-4$	$1.86E-4$	$1.13E-4$
	Uniform	$2.63E-4$	$1.86E-4$	$1.13E-4$
N_x	Adaptive	17	32	79
	Uniform	291	581	1159

TABLE 11.1. Comparison of relative CPU time, RMSE, and grid points N_x on adaptive and uniform grids at time $T = 1$. Here, $\bar{h} = 0.5h$ is used on adaptive mesh.

Similarly, Table 11.2 shows computational results such as relative CPU time, RMSE on $[0.9K, 1.1K]$ and grid points, N_x at time $T = 3$ on adaptive and uniform mesh. With this longer time, the adaptive method performs noticeably better.

To confirm the effectiveness of adaptive grid method in two-dimensional case, we get the difference with the exact and numerical values at $(x, y) = (100, 100)$ with cash-or-nothing payoff on uniform and adaptive mesh, respectively. As a result, the resulting

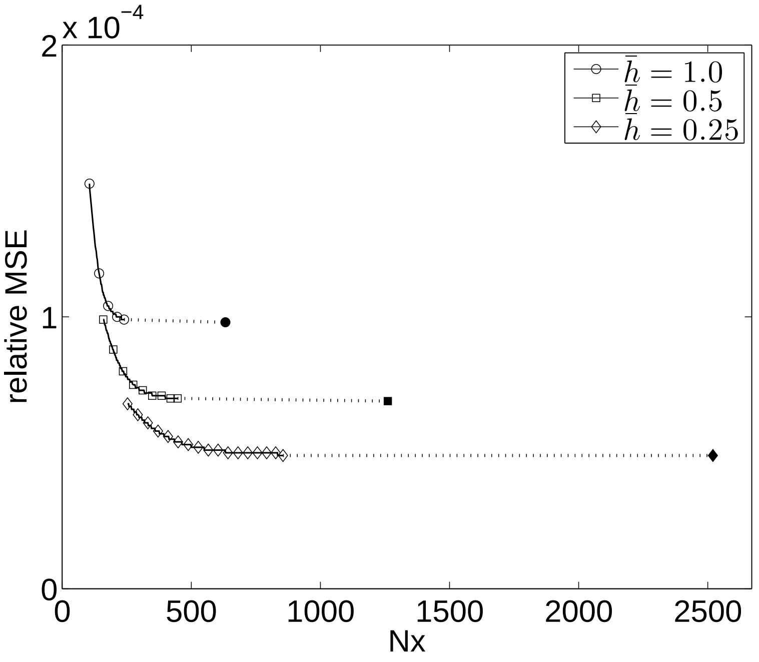


FIGURE 11.7. Relative RMSE on $[0.9K,\ 1.1K]$ with different N_x and $T = 3$. Lines with symbols, '○','□', and '◇' represent $\bar{h} = 1,\ 0.5$, and 0.25 on adaptive mesh, respectively. Also, symbols, '●','■', and '♦' represent $h = 1,\ 0.5$, and 0.25 on uniform mesh, respectively.

	Case	$h = 1$	$h = 0.5$	$h = 0.25$
CPU time	Adaptive	1	1	1
	Uniform	37.0139	102.5921	217.8538
RMSE	Adaptive	$9.81E-5$	$6.93E-5$	$4.91E-5$
	Uniform	$9.82E-5$	$6.95E-5$	$4.91E-5$
N_x	Adaptive	42	75	139
	Uniform	291	581	1159

TABLE 11.2. Comparison of relative CPU time, RMSE, and grid points, N_x on adaptive and uniform mesh at time $T = 3$. Here, $\bar{h} = 0.5h$ is used on adaptive mesh.

error using $\bar{h} = 1$ and $N_x = N_y = 79$ on adaptive mesh is 1.068E-5, but the result on uniform mesh with $h = 1$ and $N_x = N_y = 249$ is 1.837E-5

Therefore, results on the adaptive mesh are more efficient and faster than the uniform mesh on same accuracy in two-dimensional case.

11.4. Conclusions

An accurate and efficient numerical method for the Black-Scholes equations is derived in this Chapter. 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.

Chapter 12

An efficient and accurate numerical scheme for Turing instability on a predator-prey model

We present an efficient and accurate numerical method for solving a ratio dependent predator-prey model with Turing instability. The system is discretized by a finite difference method with a semi-implicit scheme which allows much larger time step sizes than a standard explicit scheme. A proof is given for the positivity and boundedness of the finite difference solutions depending only on time step sizes. Finally, we perform numerical experiments demonstrating the robustness and accuracy of the numerical solution the Turing instability. Also, we show the numerical non-constant stationary solutions with amplitudes.

12.1. Introduction

For a pair of predator and prey model, experimental search observes that the predator's growth rate is a function of the ratio of prey to predator abundance [114]. A model considering this ratio is called the ratio-dependent model which seems more appropriate in systems [97, 103, 113]. Therefore we consider a ratio-dependent model having Turing instability. In Turing instability, the equilibrium solution is asymptotically stable in the kinetic system (without diffusion), but it is unstable with diffusion term. Turing instability has been extensively investigated for biological and chemical process [107, 109, 112]. Moreover, pattern formation from the Turing instability in nonlinear complex systems is actively investigated in the fields such that social and molecular computing [105, 98]. Here we focus on the numerical scheme for a reaction diffusion dynamical systems, particulary ratio-dependent model with Turing instability.

A model having been widely used for a predator-prey model is a simple Holling Type II [102] which has a constant mortality of the predator. However, [101] introduced a more realistic model, in the model the predator mortality is an increasing function (neither a constant nor an unbounded) of the predator's abundance. We use the modified Cavani and Farkas [99, 100] ratio-dependent model with diffusion and without delay, proposed by [123].

Let $N(x,t)$ and $P(x,t)$ be the prey and predator population densities for time t and space x on the one dimensional domain $\Omega = (0,l)$, respectively. The governing equations are:

$$N_t = rN\left(1-\frac{N}{K}\right) - \frac{aNP}{mP+N} + D_1 N_{xx}, \tag{12.1}$$

$$P_t = -\frac{P(\gamma+\delta P)}{1+P} + \frac{bNP}{mP+N} + D_2 P_{xx}, \tag{12.2}$$

where a, b, r, m, γ, δ, D_1, D_2, and K are positive constants. Subscripts denote partial differentiation with respect to the variables.

The prey grows with the intrinsic growth rate r and the constant carrying capacity K. The presence of the predator reduces the prey's growth rate with a capturing rate a and the time spend capturing saturation depending on the predator density mP. And the predator's mortality is $(\gamma + \delta P)/(1 + P)$ where γ and δ are the minimal and limiting mortalities of the predator, respectively. Naturally, we assume that $0 < \gamma \leq \delta$. Also, the prey's contribution to the predator's growth rate is $bNP/(mP + N)$ where b is a conversion rate. Assume that the prey and predator diffuse by Fick's law with constant diffusions D_1 and D_2. The boundary conditions satisfy the homogeneous Neumann boundary conditions:

$$N_x(0,t) = N_x(l,t) = P_x(0,t) = P_x(l,t) = 0$$

and initial conditions are given by

$$N(x,0) > 0,\ P(x,0) > 0,\ x \in (0,l).$$

The governing Eqs. (12.1) and (12.2) can be non-dimensionalized by introducing dimensionless variables

$$\widetilde{t} = rt,\ \widetilde{N} = \frac{N}{K},\ \widetilde{P} = \frac{mP}{K},\ \widetilde{x} = \frac{x}{l}$$

and

$$\alpha = \frac{a}{mr},\ \widetilde{\gamma} = \frac{\gamma}{b},\ \widetilde{\delta} = \frac{\delta}{b},\ \epsilon = \frac{b}{r},\ \beta = \frac{K}{r},\ d_1 = \frac{D_1}{l^2 r},\ d_2 = \frac{D_2}{l^2 r}.$$

After omitting tilde notation, we get the following non-dimensional system:

$$N_t = N(1-N) - \frac{\alpha NP}{P+N} + d_1 N_{xx}, \qquad (12.3)$$

$$P_t = -\frac{\epsilon P(\gamma + \delta\beta P)}{1+\beta P} + \frac{\epsilon NP}{P+N} + d_2 P_{xx}, \qquad (12.4)$$

on the one dimensional space domain $\Omega = (0,1)$, with the positive initial conditions and the boundary conditions

$$N_x(0,t) = N_x(1,t) = P_x(0,t) = P_x(1,t) = 0. \qquad (12.5)$$

We demonstrate the numerical solutions of this system, particulary with semi-implicit scheme. Using our scheme, we prove stability depending only on a time scales and the numerical experience are efficiently and accurately performed.

This Chapter is organized as follows. In Section 2, we investigate the equilibrium stationary solution in the kinetic systems, the Turing instability conditions, and the non-constant stationary solution with small amplitudes in diffusion-reaction system with a general method [123, 99, 100, 104, 106]. In Section 3, we propose the efficient and accurate numerical scheme for the Turing instability. Moreover, we analyze and prove the positivity and boundedness of numerical solutions. In Section 4, we illustrate the numerical solutions with respect to the Turing instability region and we demonstrate numerical experiments of the non-constant stationary solution with small amplitudes as time goes. Finally, we calculate the stability constraint for an explicit scheme comparing with our semi-implicit scheme.

12.2. A model analysis

12.2.1. The model without diffusion. In the kinetic system, we get the equilibrium points by setting the derivative term as zero. Then the equilibrium points are $(0,0)$, $(1,0)$, and at least one point having positive values which is the point of intersection of the null-clines as shown in Fig. 12.1. The prey and predator's null-clines are

$$\begin{aligned} P &= H_1(N) = \frac{(1-N)N}{\alpha-1+N}, \\ P &= H_2(N) = \frac{-\gamma-\beta(\delta-1)N+\sqrt{[\gamma+\beta(\delta-1)N]^2+4\delta\beta(-\gamma+1)N}}{2\beta\delta}, \end{aligned}$$

respectively.

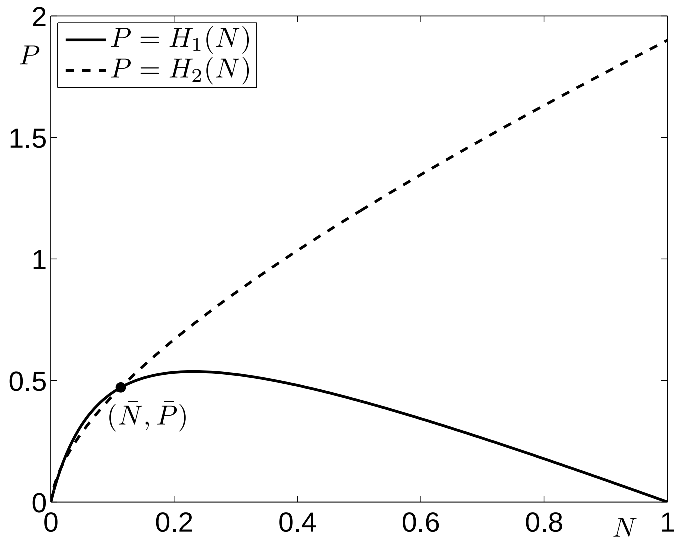


FIGURE 12.1. Null-clines of N and P with $\epsilon = 1, \alpha = 1.1, \gamma = 0.05, \beta = 1 \text{and} \delta = 0.5$.

Let $(\overline{N}, \overline{P})$ be the equilibrium point with positive values, then $(\overline{N}, \overline{P})$ is in the All*é*e effect zone [111] as shown in Fig. 12.1. Therefore there exist stable equilibrium points with positive values for the given constants. To observe the local stability near $(\overline{N}, \overline{P})$, let $u = N - \overline{N}$ and $v = P - \overline{P}$. Then with the Jacobian matrix A, the linearized kinetic system forms

$$\begin{pmatrix} u_t \\ v_t \end{pmatrix} = A \begin{pmatrix} u \\ v \end{pmatrix}, \text{ where } A = \begin{pmatrix} a_{11} & -a_{12} \\ a_{21} & -a_{22} \end{pmatrix}.$$

Therefore in the kinetic system, $(\overline{N}, \overline{P})$ is locally asymptotically stable when trace$A < 0$ and det$A > 0$, that is, $a_{11} < a_{22}$, and $a_{12}a_{21} > a_{11}a_{22}$.

12.2.2. The model with diffusion. Let the two-dimensional vector $\mathbf{u} = (N, P)^T$, the diagonal matrix D=diag(d_1, d_2), and $F = (g_1, g_2)^T$ where

$$g_1(N,P) = N(1-N) - \frac{\alpha NP}{P+N}, \; g_2(N,P) = P\left(-\frac{\gamma + \delta P}{1+P}\right) + \frac{bNP}{mP+N}.$$

Then the Eqs. (12.3)-(12.5) can be written as

$$\mathbf{u}_t = F(\mathbf{u}) + D\mathbf{u}_{xx}, \tag{12.6}$$

$$\mathbf{u}_x(0,t) = \mathbf{u}_x(1,t) = \mathbf{0}. \tag{12.7}$$

The equilibrium $\overline{\mathbf{u}} = (\overline{N}, \overline{P})$ is called Turing unstable if it is an asymptotically stable equilibrium of the kinetic system but it is unstable with diffusion term [99, 100]. The definition implies that the solution of the nonlinear system having initial values $\mathbf{u}(x,0)$ close to $\overline{\mathbf{u}}$, but does not approach to $\overline{\mathbf{u}}$ as time t goes infinity.

With a linear analysis using a method of separation variables and eigenvalue problem,

the eigenvalues are $\zeta_n = (n\pi)^2$, $n = 0, 1, 2, \cdots$ with corresponding eigenfunctions $\psi_n(x) = \cos(n\pi x)$, $n = 0, 1, 2, \cdots$ of the linearlized system of Eqs. (12.6) and (12.7) [123, 99, 100, 104].

Let $B_n = A - \psi_n D$. Then

$$\begin{aligned} \text{trace}B_n &= a_{11} - a_{22} - \zeta_n(d_1 + d_2), \\ \det B_n &= a_{12}a_{21} - a_{11}a_{22} + \zeta_n(d_1 a_{22} - d_2 a_{11}) + \zeta_n^2 d_1 d_2. \end{aligned}$$

With the stable condition in the kinetic system, the equilibrium solution $(\overline{N}, \overline{P})$ is Turing unstable when

$d_1 a_{22} < d_2 a_{11}$ and $(d_1 a_{22} - d_2 a_{11})^2 - 4d_1 d_2(a_{12}a_{21} - a_{11}a_{22}) > 0$ or there exists a positive integer k such that $\det B_k < 0$.

Here we consider the singular perturbation analysis for d_2 near the bifurcation point. The singular perturbation analysis is that it is sufficient to consider only one diffusion even though there exist two diffusions. So we fix d_1 and take d_2 as a bifurcation parameter.

For $\overline{\mathbf{u}} = (\overline{N}, \overline{P})^T$, we have $F(d_2, \overline{\mathbf{u}}) = \mathbf{0}$ for all $d_2 \in [0, \infty)$. And we let d_c be the critical value for a Turing bifurcation.

We say that $\overline{\mathbf{u}}$ undergoes a Turing bifurcation at $d_c \in [0, \infty)$ if the solution $\overline{\mathbf{u}}$ is asymptotically stable for $0 < d_2 < d_c$ and it is unstable for $d_c < d_2$, and Eq. (12.6) has non-constant stationary solution in some neighborhood of d_c. The stationary solution does not depend on time but depends on space [123, 99, 100, 104].

Therefore for all $d_2 > 0$, consider the condition for $\det B_n < 0$. When we write down $\det B_n$ as

$$\det B_n = d_2\zeta_n(\zeta_n d_1 - a_{11}) + a_{12}a_{21} - a_{11}a_{22} + \zeta_n d_1 a_{22},$$

and since ζ_n is monotonic increasing, $a_{11}/\zeta_2 \leq d_1 < a_{11}/\zeta_1$. The eigenvalues λ_i for $i = 1, 2$ of B_n satisfies $\lambda_i^2 - \text{trace}B_n\lambda_i + \det B_n = 0$. Since Turing bifurcation occurs when the real and imaginary part of λ is 0, the critical value for a Turing bifurcation d_c satisfies

$$d_c = \frac{a_{12}a_{21} - a_{11}a_{22} + \zeta_1 d_1 a_{22}}{\zeta_1(a_{11} - \zeta_1 d_1)}. \tag{12.8}$$

When $d_2 = d_c$, the eigenvalues are zero and traceB_n. Since trace$B_n < 0$, we consider the zero eigenvalue. Denote the unit eigenvector corresponding to the zero eigenvalue by $(\eta_1, \eta_2)^T$. Then Turing unstable solution of the linearlized system of Eqs. (12.6) and (12.7) forms

$$\phi(x) = \begin{pmatrix} \eta_1 \\ \eta_2 \end{pmatrix} \cos(\pi x).$$

For the non-linear Eqs. (12.6) and (12.7), by Thm. 13.4 [110], $(d_c, 0)$ is a bifurcation point and there exists a $\delta > 0$ satisfying a function $d_2(s) : (-\delta, \delta) \to R$,

$$\mathbf{u}(x,s) \;=\; \overline{\mathbf{u}} + s\phi(x)\cos(\pi x) + O(s^2) \tag{12.9}$$

as the non-constant stationary solution of the nonlinear parabolic system. If we rewrite the above equation in the component form, then

$$\begin{aligned} N(x) &= \overline{N} + s\eta_1 \cos(\pi x) + O(s^2), \\ P(x) &= \overline{P} + s\eta_2 \cos(\pi x) + O(s^2). \end{aligned}$$

By Thm. 13.4 [110], Eqs. (12.6) and (12.7) have no other stationary solution except $(\overline{N}, \overline{P})$ and Eq. (12.9).

12.3. Numerical solution

12.3.1. Proposed numerical scheme. Let us first discrete the given computational space domain $\Omega = (0,1)$ as a uniform grid with a space step $h = 1/N_x$ and a time step $\Delta t = T/N_t$. The numerical approximation to the solution $(\overline{N}, \overline{P})$ is denoted by

$$\begin{aligned} N_i^n &\equiv N(x_i, t^n) = N((i-0.5)h, n\Delta t), \\ P_i^n &\equiv P(x_i, t^n) = P((i-0.5)h, n\Delta t) \end{aligned}$$

where $i = 1, 2, \cdots, N_x$ and $n = 0, 1, \cdots, N_t$.

First, we solve Eqs. (12.3) and (12.4) using the semi-implicit scheme in time and a centered difference scheme in space. Since fully explicit schemes may have restriction of time step by diffusion term and fully implicit schemes may be expensive [108], it is efficient to use the semi-implicit scheme. We write the schemes as follows:

$$\frac{N_i^{n+1} - N_i^n}{\Delta t} = N_i^n \left(1 - N_i^n - \frac{\alpha P_i^n}{P_i^n + N_i^n}\right) + d_1 \Delta_h N_i^{n+1}, \tag{12.10}$$

$$\frac{P_i^{n+1} - P_i^n}{\Delta t} = \epsilon P_i^n \left(-\frac{\gamma + \delta\beta P_i^n}{1 + \beta P_i^n} + \frac{N_i^n}{P_i^n + N_i^n}\right) + d_2 \Delta_h P_i^{n+1} \tag{12.11}$$

$$\text{for} \quad i = 1, \cdots, N_x \text{ and } n = 0, \cdots, N_t - 1.$$

Assume that the initial condition of N satisfies $0 < N(x,0) \le 1$ and P satisfies $0 < P(x,0) \le L$ for all x and for $L = \max(1/\gamma, Q)$ for some positive $Q > 1/\gamma$. And for the boundary condition, Neumann condition holds as follows:

$$N_0^n = N_1^n, \; N_{N_x+1}^n = N_{N_x}^n, \; P_0^n = P_1^n, \; P_{N_x+1}^n = P_{N_x}^n.$$

Since we discretize the system for a linear term explicitly and a nonlinear term implicitly, it is convenient to split the scheme into following two steps.

Step 1)

$$\frac{N_i^* - N_i^n}{\Delta t} = N_i^n \left(1 - N_i^n - \frac{\alpha P_i^n}{P_i^n + N_i^n}\right), \tag{12.12}$$

$$\frac{P_i^* - P_i^n}{\Delta t} = \epsilon P_i^n \left(-\frac{\gamma + \delta\beta P_i^n}{1 + \beta P_i^n} + \frac{N_i^n}{P_i^n + N_i^n}\right). \tag{12.13}$$

Step 2)

$$\frac{N_i^{n+1} - N_i^*}{\Delta t} = d_1 \frac{N_{i-1}^{n+1} - 2N_i^{n+1} + N_{i+1}^{n+1}}{h^2}, \tag{12.14}$$

$$\frac{P_i^{n+1} - P_i^*}{\Delta t} = d_2 \frac{P_{i+1}^{n+1} - 2P_i^{n+1} + P_{i-1}^{n+1}}{h^2}. \tag{12.15}$$

12.3.2. The positiveness and boundedness of the solution under certain condition of Δt. Now we will show the positiveness and boundedness of the solution under certain condition of Δt. Suppose that $0 < N_i^n \leq 1,\ 0 < P_i^n \leq L$ for $i = 1, \cdots, N_x$ and $L = \max(1/\gamma, Q)$ for some positive Q.

Firstly, for the nonlinear term of *Step 1*, we show the positiveness and boundedness of N_i^* and P_i^* for $i = 1, \cdots, N_x$.

$$\begin{aligned} N_i^* &= N_i^n + \Delta t N_i^n \left(1 - N_i^n - \frac{\alpha P_i^n}{P_i^n + N_i^n}\right) \\ &\geq N_i^n + \Delta t N_i^n (1 - N_i^n) - \Delta t \alpha N_i^n \\ &\geq N_i^n - \Delta t \alpha N_i^n = (1 - \Delta t \alpha) N_i^n > 0 \end{aligned}$$

when $\Delta t < 1/\alpha$. And

$$\begin{aligned} N_i^* &= N_i^n + \Delta t N_i^n \left(1 - N_i^n - \frac{\alpha P_i^n}{P_i^n + N_i^n}\right) \\ &\leq N_i^n + \Delta t N_i^n (1 - N_i^n) \leq N_i^n + \Delta t (1 - N_i^n) \\ &= N_i^n (1 - \Delta t) + \Delta t \leq 1 - \Delta t + \Delta t = 1 \end{aligned}$$

when $\Delta t \leq 1$. Secondly, for P^*, we have

$$\begin{aligned} P_i^* &= P_i^n + \Delta t \epsilon P_i^n \left(-\frac{\gamma + \delta\beta P_i^n}{1 + \beta P_i^n} + \frac{N_i^n}{P_i^n + N_i^n}\right) \\ &\geq P_i^n + \Delta t \epsilon P_i^n \left(-\delta + \frac{\delta - \gamma}{1 + \beta P_i^n}\right) \geq P_i^n \left(1 - \Delta t \epsilon \frac{\gamma + \delta\beta}{1 + \beta}\right) > 0 \end{aligned}$$

when $\Delta t < (1 + \beta)/(\epsilon\gamma + \epsilon\delta\beta)$. And using $0 < \gamma \leq \delta$, we get

$$\begin{aligned} P_i^* &= P_i^n + \Delta t \epsilon P_i^n \left(-\frac{\gamma + \delta\beta P_i^n}{1 + \beta P_i^n} + \frac{N_i^n}{P_i^n + N_i^n}\right) \\ &\leq P_i^n + \Delta t \epsilon P_i^n \left(-\frac{\gamma + \gamma\beta P_i^n}{1 + \beta P_i^n} + \frac{1}{P_i^n}\right) \\ &= P_i^n - \Delta t \epsilon P_i^n \gamma + \Delta t \epsilon = P_i^n (1 - \Delta t \epsilon \gamma) + \Delta t \epsilon \\ &\leq L(1 - \Delta t \epsilon \gamma) + \Delta t \epsilon = L + \Delta t \epsilon \gamma (1/\gamma - L) \leq L \end{aligned}$$

for all Δt. By these results, for Eqs. (12.12) and (12.13), if

$$0 < N_i^n \leq 1,\ 0 < P_i^n \leq L \text{ for } i = 0, \cdots, N_x$$

and $\Delta t < \min\left(1, 1/\alpha, (1+\beta)/(\epsilon\gamma + \epsilon\delta\beta)\right)$, then

$$0 < N_i^* \leq 1,\ 0 < P_i^* \leq L \text{ for } i = 1, \cdots, N_x.$$

Next, for the linear term of *Step 2*, we need to show the boundedness and positiveness of N_i^{n+1} and P_i^{n+1} for $i = 1, \cdots, N_x$. So we need to show for $i = 1, \cdots, N_x$,

$$\begin{cases} N_i^{n+1} \geq m^* > 0 \text{ where } m^* = \min_{1\leq k\leq N_x} N_k^* \\ N_i^{n+1} \leq M^* \leq 1 \text{ where } M^* = \max_{1\leq k\leq N_x} N_k^* \\ P_i^{n+1} \geq m'^* > 0 \text{ where } m'^* = \min_{1\leq k\leq N_x} P_k^* \\ P_i^{n+1} \leq M'^* \leq L \text{ where } M'^* = \max_{1\leq k\leq N_x} P_k^*. \end{cases} \tag{12.16}$$

To show the first inequality of Eq. (12.16), assume contrary that $N_i^{n+1} < m^*$, then there exists some i, $2 \leq i \leq N_x - 1$ which satisfies $N_i^{n+1} \leq \min(N_{i-1}^{n+1}, N_{i+1}^{n+1})$. Let $\omega = d_1 \Delta t / h^2$. Then Eq. (12.14) can be written as

$$(1 + 2\omega) N_i^{n+1} = N_i^* + \omega(N_{i-1}^{n+1} + N_{i+1}^{n+1}).$$

Since $N_i^{n+1} < N_i^*$ by the assumption, we rewrite above equation as $N_i^{n+1} > (N_{i-1}^{n+1} + N_{i+1}^{n+1})/2$, and which implies that either $N_i^{n+1} > N_{i-1}^{n+1}$ or $N_i^{n+1} > N_{i+1}^{n+1}$. But since it contradicts to the assumption, we have $N_i^{n+1} \geq m^*$.

For the case of $i = 1$, assume $N_1^{n+1} < m^*$ such that $N_1^{n+1} \leq N_2^{n+1}$, then using the homogeneous Neumann boundary condition $N_0^{n+1} = N_1^{n+1}$, we have $(1+\omega)N_1^{n+1} = N_1^* + \omega N_2^{n+1}$. Therefore, we have $N_1^{n+1} > N_2^{n+1}$, which contradicts our assumption. In the case of $N_{N_x}^{n+1} < m^*$ and $N_{N_x}^{n+1} \leq N_{N_x-1}^{n+1}$, the same result holds with the same reasoning.

And we show the second inequality of Eq. (12.16). Assume contrary as $N_i^{n+1} > M^*$ such that $N_i^{n+1} \geq \max(N_{i-1}^{n+1}, N_{i+1}^{n+1})$ for some i, $2 \leq i \leq N_x - 1$. Then Eq. (12.14) is,

$$(1 + 2\omega) N_i^{n+1} = N_i^* + \omega(N_{i-1}^{n+1} + N_{i+1}^{n+1}).$$

Therefore we have $N_i^{n+1} < (N_{i-1}^{n+1} + N_{i+1}^{n+1})/2$ by the assumption, which implies that either $N_i^{n+1} < N_{i-1}^{n+1}$ or $N_i^{n+1} < N_{i+1}^{n+1}$, which contradicts to the assumption. So, we have $N_i^{n+1} \leq M^*$. For the case of $i = 1$ and N_x, we can show the inequality applying the same argument as before.

Also, using same argument the third and the fourth inequality of Eq. (12.14) can be proved, if $0 < P_i^* \leq L$ for $i = 1, \cdots, N_x$, then $0 < P_i^{n+1} \leq L$ for $i = 1, \cdots, N_x$. Collecting all the above results, we get the following theorem:

Theorem. *Let* $0 < N_i^0 \leq 1,\ 0 < P_i^0 \leq L$ *for* $i = 1, \cdots, N_x$ *where* $L = max\{1, Q\}$ *for some positive* Q. *Suppose that*

$$\Delta t \leq \min\left(1, \frac{1}{\alpha}, \frac{1+\beta}{\epsilon(\gamma + \delta\beta)}\right).$$

Then the numerical solutions N_i^{n+1} and P_i^{n+1} obtained iteratively by Eqs. (12.10) and (12.11) satisfy

$$0 < N_i^{n+1} \le 1,\ 0 < P_i^{n+1} \le L \textit{ for } i = 1, \cdots, N_x \textit{ and } n = 0, 1, \cdots .$$

12.4. Numerical results

12.4.1. Predator and prey solutions. In numerical tests, we use the same parameters $\epsilon = 1$, $\alpha = 1.1$, $\gamma = 0.05$, $\beta = 1$, and $\delta = 0.5$ on the computational domain $\Omega = (0, 1)$ [123]. Then there exist four equilibrium points, among them $(\bar{N}, \bar{P}) = (0.113585, 0.471397)$ is the unique stable equilibrium point with positive values.

In Fig. 12.2, Turing instability region is shown in the phase plane of diffusion coefficients d_1 and d_2 using Eq. (12.8).

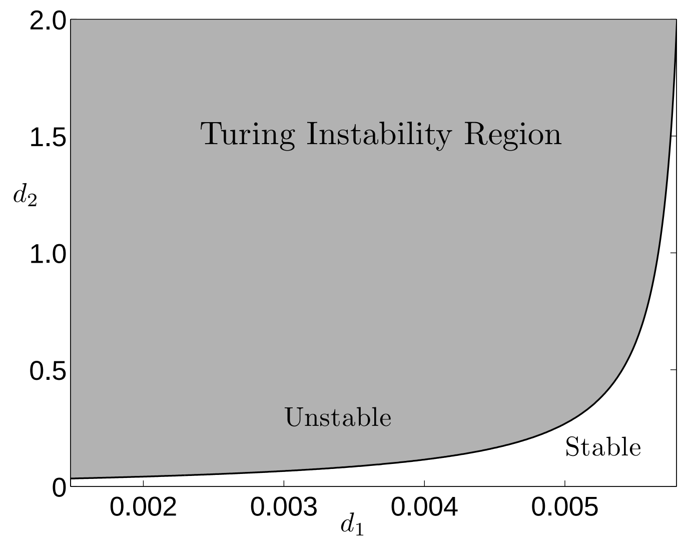


FIGURE 12.2. The relation between the prey diffusion d_1 and the predator diffusion d_2.

For diffusion coefficients, we can consider d_1 as an activator and d_2 as an inhibitor in the activator-inhibitor system. In terms of the activator-inhibitor mechanism by [106], the inhibitor must diffuse faster than the activator, that is $d_1 < d_2$ (A predator moves much quickly to catch a prey). Therefore we take small value for d_1. Let us fix $d_1 = 0.005$.

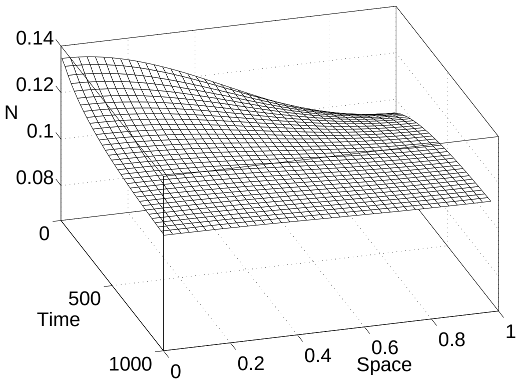
Next, with initial conditions

$$N(x, 0) = \bar{N} + 0.0214 \cos(\pi x), \tag{12.17}$$

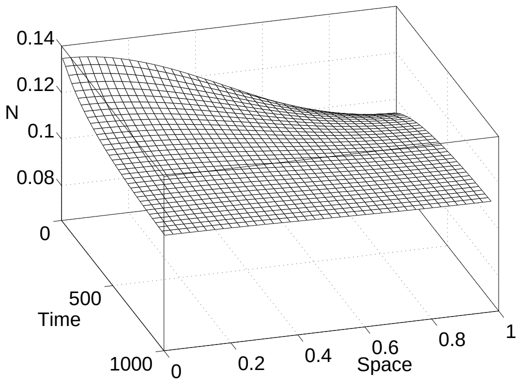
$$P(x, 0) = \bar{P} + 0.0066 \cos(\pi x), \tag{12.18}$$

we compare the numerical solutions with d_2 in the stable and unstable regions as shown in Fig. 12.3. We use the spatial mesh size $h = 0.005$ on computational domain

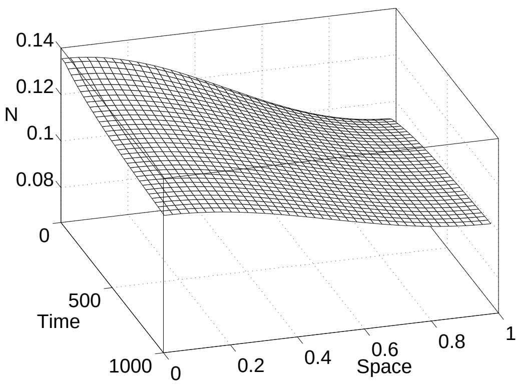
$\Omega = (0,1)$, the time step $\Delta t = 0.9$ and the total time $T = 1000$. In case of $d_2 < d_c$ ($d_2 = 0.2$ and $d_c = 0.271$), the time evolution of $N(x,t)$ and $P(x,t)$ are shown in Fig. 12.3(a) and (b), respectively. The numerical solutions converge to the equilibrium solution $\bar{N}$ and $\bar{P}$ as time iterations increase. However, in case of $d_2 > d_c$ ($d_2 = 0.32$), the time evolution of $N(x,t)$ and $P(x,t)$ are shown in Fig. 12.3(c) and (d), respectively. The numerical solutions show the deviation from the equilibrium solution $\bar{N}$ and $\bar{P}$ since d_2 is in the Turing instability region.



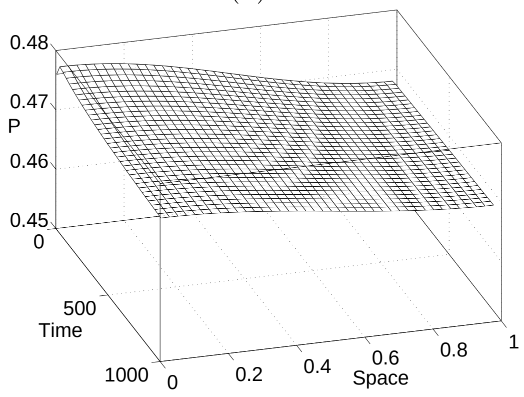
(a) N



(b) P



(c) N



(d) P

FIGURE 12.3. In case of $d_1 = 0.005$ and $d_2 = 0.2$ where d_2 is in stable region, (a) the prey solution $N(x,t)$ and (b) the predator solution $P(x,t)$ with respect to time and space. And in case of $d_1 = 0.005$ and $d_2 = 0.32$ where d_2 is in unstable region, (c) the prey solution $N(x,t)$ and (d) the predator solution $P(x,t)$ with respect to time and space.

12.4.2. Non-constant stationary solution. In this section, we investigate the prey and predator non-constant stationary solutions which have small amplitudes using the same semi-implicit scheme. Let $(\eta_1, \eta_2)^T$ be the unit eigenvector corresponding to the eigenvalue 0 of B_1. Figs. 12.4 and 12.5 show the prey and predator's non-constant stationary numerical solutions with different s's in two cases of $d_2 < d_c$ and $d_2 > d_c$, respectively. In this test, we let $s_1 = 0.002$, $s_2 = 0.004$, $s_3 = 0.008$, and $s_4 = 0.016$.

We use the spatial mesh size $h = 0.01$ on the computational domain $\Omega = (0, 1)$, with the time step $\Delta t = 0.01$ and the final time $T = 100$. Initial conditions are taken close to the non-constant stationary solutions,

$$N(x, 0) = \bar{N} + s_j\eta_1 \cos(\pi x), \qquad (12.19)$$

$$P(x, 0) = \bar{P} + s_j\eta_2 \cos(\pi x), \text{ for } j = 1, \cdots, 4. \qquad (12.20)$$

Symbols (s_1 : circle, s_2 : diamond, s_3 : square, and s_4 : triangle) represent the initial conditions.

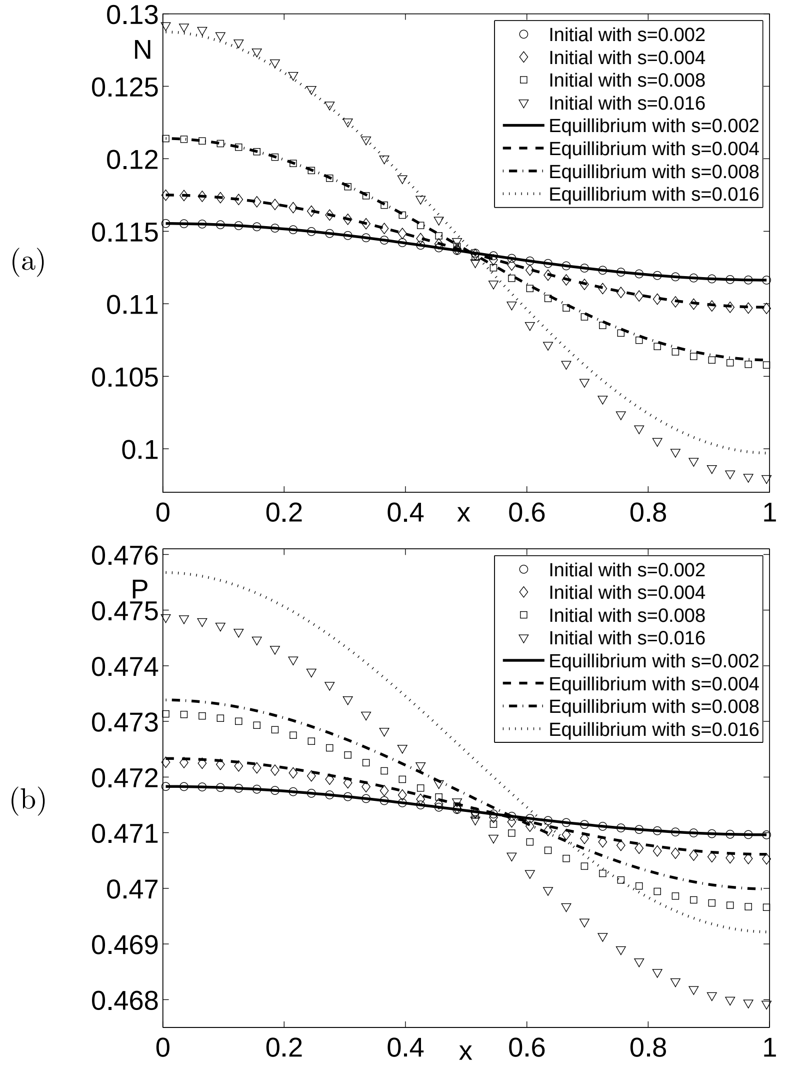
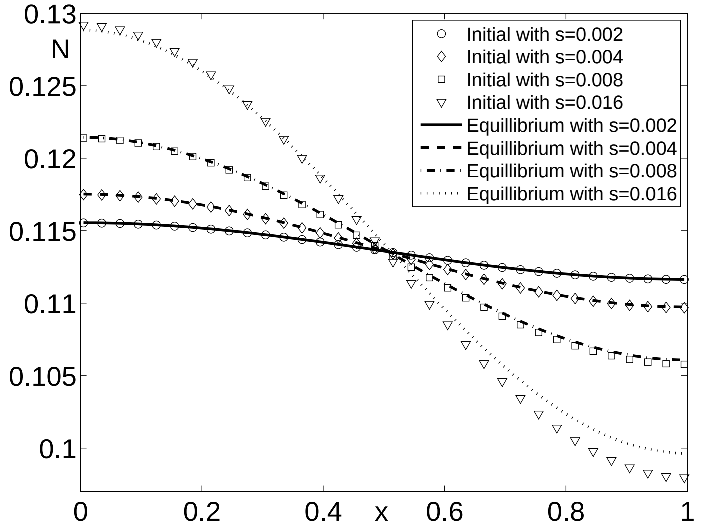


FIGURE 12.4. When $d_2 < d_c$ with varing s: (a) Prey $N(x, t)$ and (b) predator $P(x, t)$ solution pattern ($d_1 = 0.005,\ d_2 = 0.27,\ d_c = 0.271$). Here, each markers represent the initial condition depending on s.

The following test in Fig. 12.6 demonstrates the existence of the non-constant equilibrium solution after long time evolution. Fig. 12.6 (a) and (c) shows the time

(a)



(a)

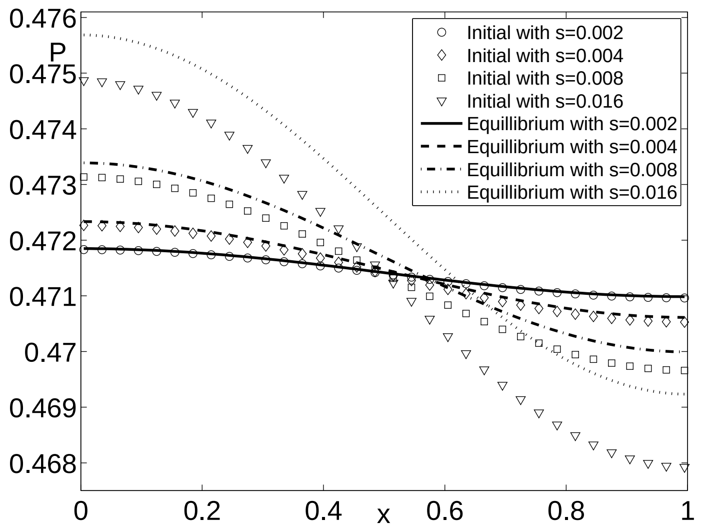


FIGURE 12.5. When $d_2 > d_c$ with varing s: (a) Prey $N(x,t)$ and (b) predator $P(x,t)$ solution pattern ($d_1 = 0.005,\ d_2 = 0.272,\ d_c = 0.271$). Here, each markers represent the initial condition depending on s.

evolution of prey and predator solutions with $d_1 = 0.005,\ d_2 = 0.27,\ d_c = 0.271$, and $s_4 = 0.016$. We use the spatial mesh size $h = 0.0025$ on the computational domain $\Omega = (0, 1)$, the time step $\Delta t = 0.01$, and the final time $T = 100$. And Fig. 12.6 (b) and (d) show prey and predator solutions at five points (x= 0.00125, 0.24875, 0.49875, 0.74875, 0.99875) depending on the time. At each point x, Fig. 12.6 shows that with small amplitude s, the pattern of solution goes to the stationary solution.

12.4.3. Effect of the amplitudes. Figs. 12.4 and 12.5 show similar results when $d_2 \in (-\delta, \delta)$, that is the value around the Turing bifurcation point d_c. Therefore in this section, we let $d_2 = d_c$ to consider the solution patterns on the bifurcation point.

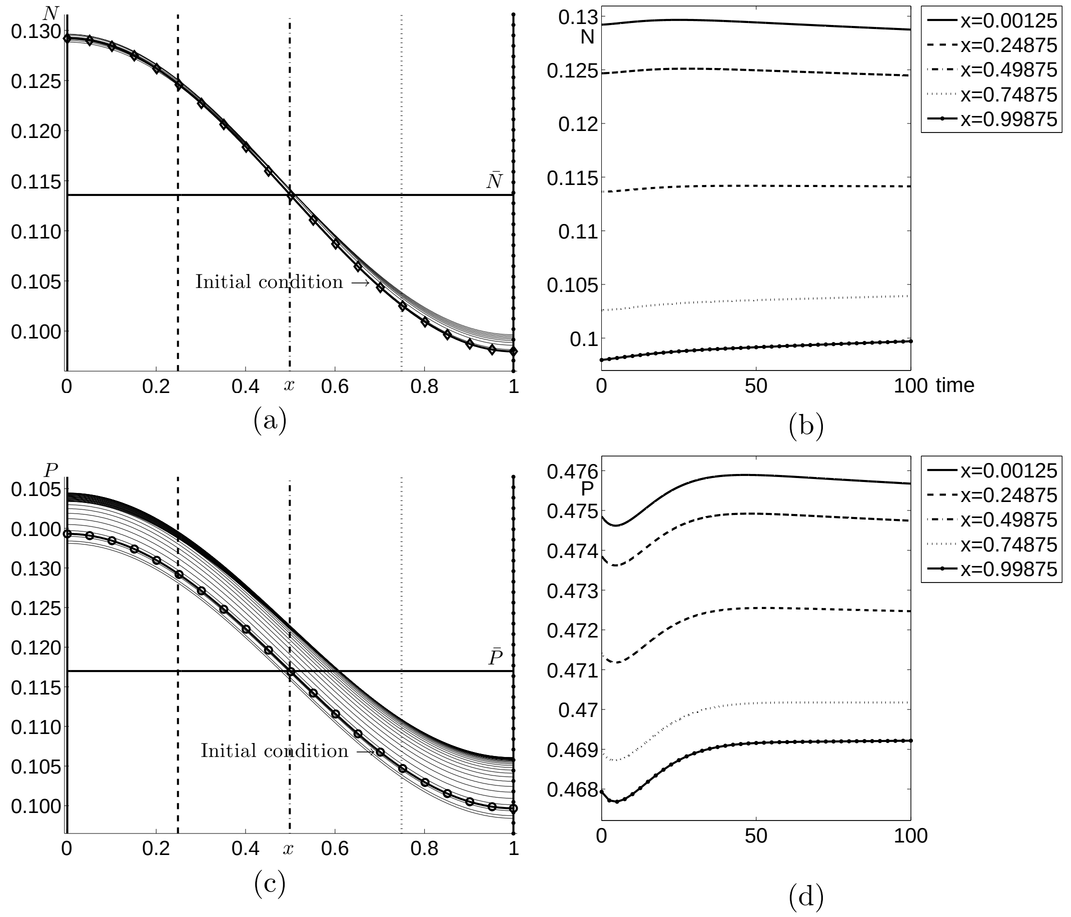


FIGURE 12.6. (a) and (c): the time evolution of prey and predator solution pattern $N(x,t),\ P(x,t)$ with $d_1 = 0.005,\ d_2 = 0.27,\ d_c = 0.271$ and $s = 0.016$. (b) and (d): prey and predator solution at five points ($x = 0.00125\ ,0.24875,\ 0.49875,\ 0.74875,\ 0.99875$) depending on the time.

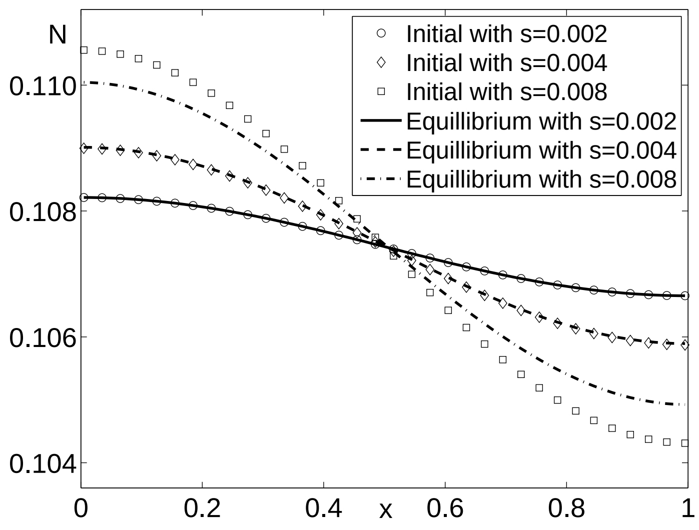
Now, to observe the effect of amplitudes on the stationary state, the following tests are presented. The procedure is repeated until the relative change with respect to a time step Δt is smaller than a tolerance ϵ, namely

$$\max\left(\frac{\|N^{n+1}-N^n\|_\infty}{\Delta t}, \frac{\|P^{n+1}-P^n\|_\infty}{\Delta t}\right) < \epsilon.$$

In this test, we use time step $\Delta t = 0.9$, space step $h = 0.01$, tolerance $\epsilon = 1.0E-6$, and the other parameters are the same values as the previous tests, with initial conditions as Eqs. (12.17) and (12.18).

Figs. 12.7 and Fig. 12.8 show the solution patterns of $N(x,t)$ and $P(x,t)$ with varying amplitudes s, respectively. These results suggest that the existence of the non-constant stationary solutions even when the s is large enough.

(a)



(a)

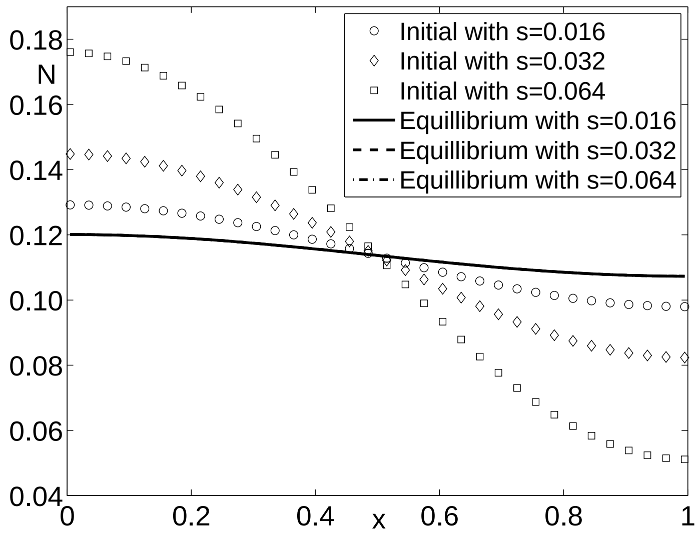
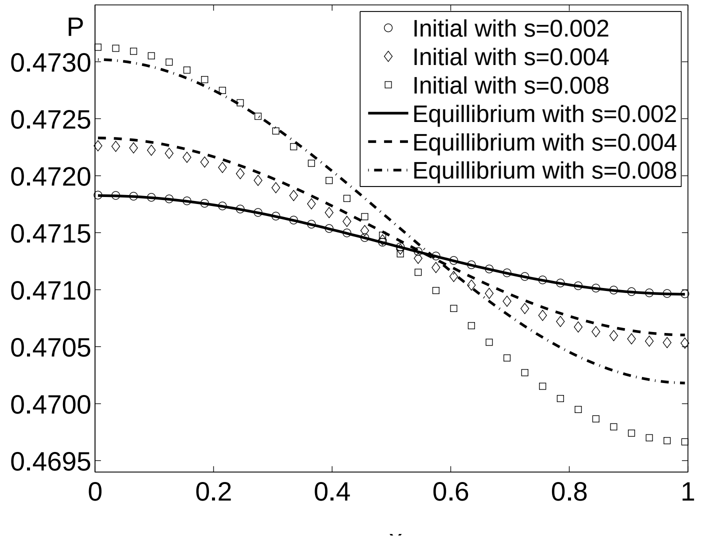


FIGURE 12.7. When $d_2 = d_c$ with different s : (a) $s = 0.002,\ 0.004,\ 0.008$ (b) $s = 0.016,\ 0.032,\ 0.064$, $N(x,t)$ solution patterns ($d_1 = 0.005,\ d_c = 0.271$). Here, each markers represents the initial condition depending on s.

Finally, we investigate the stability constraint for an explicit and the proposed scheme. We calculate the maximum Δt corresponding to different spatial grid sizes h so that stable solutions can be computed up to the total iteration 10000. We use the following parameters $d_1 = 0.005,\ d_2 = d_c = 0.271,$ and $s = 0.032$ on the computational domain $\Omega = (0,1)$ with initial conditions as Eqs. (12.19) and (12.20). As shown in Table 12.1, we obtain stable solutions for all five mesh sizes. The results indicate that the explicit scheme has a stability restriction of the time step, $\Delta t \approx O(h^2)$. Whereas

(a)



(a)

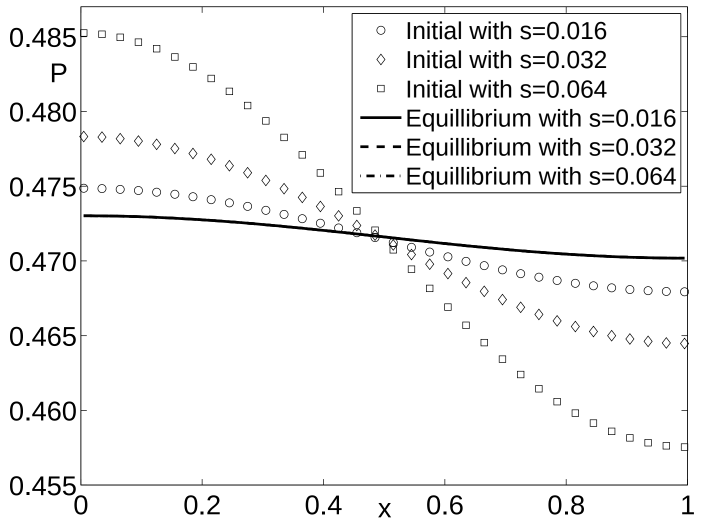


FIGURE 12.8. When $d_2 < d_c$ with different s: (a) $s = 0.002,\ 0.004,\ 0.008$ (b) $s = 0.016,\ 0.032,\ 0.064$, $P(x,t)$ solution patterns ($d_1 = 0.005,\ d_2 = 0.27,\ d_c = 0.271$). Here, each markers represents the initial condition depending on s.

the proposed scheme has only restriction of the time step, Δt given by

$$\Delta t \le \min\left(1, \frac{1}{\alpha}, \frac{1+\beta}{\epsilon(\gamma+\delta\beta)}\right) \approx 0.9091,$$

which is independent of spatial step sizes h. Therefore, the proposed semi-implicit scheme is practically more stable than the explicit scheme.

Mesh size	Explicit scheme	Proposed scheme
$h = 0.01$	$\Delta t \leq 1.853E\text{-}4$	$\Delta t \leq 1.545E\text{+}1$
$h = 0.005$	$\Delta t \leq 4.630E\text{-}5$	$\Delta t \leq 1.545E\text{+}1$
$h = 0.0025$	$\Delta t \leq 1.150E\text{-}5$	$\Delta t \leq 1.545E\text{+}1$
$h = 0.00125$	$\Delta t \leq 2.800E\text{-}6$	$\Delta t \leq 1.545E\text{+}1$
$h = 0.000625$	$\Delta t \leq 7.000E\text{-}7$	$\Delta t \leq 1.545E\text{+}1$

TABLE 12.1. Comparison of stability constraint of Δt for explicit and proposed schemes. In this test, we use the following parameters $d_1 = 0.005$, $d_2 = d_c = 0.271,$ and $s = 0.032$ on the computational domain $\Omega = (0, 1)$ up to the number of total iteration 10000.

12.5. Conclusions

We investigate the efficient and accurate finite difference scheme to find numerical solutions of the ratio-dependent Michaelis-Menten type predator-prey model. To do this, we have considered constant stable equilibrium solutions and the conditions for Turing instability of the solutions. With Turing instability occurring conditions, we have presented the semi-implicit scheme to get numerical results of the prey and the predator solutions. Since we use the semi-implicit scheme, the total time iteration is much smaller and the scheme is stable. Also, we have proved the positivity and boundedness of the prey and predator solutions only depending on time scale Δt. To show the superiority of our proposed scheme, we compare an explicit scheme's dependency of time and space with our scheme. Moreover, we have shown numerical evidence for the non-constant stationary solutions not only with small amplitudes but also large enough amplitudes. Since our proposed scheme assures the stability, we expect that the proposed scheme is useful to demonstrating the biological system numerically.

Chapter 13

Mathematical model and numerical simulation of the cell growth in scaffolds

A mathematical model to predict the growth of cells is a powerful tool in scaffold designs, depending on its own materials. The improved understanding derived from this mathematical model and its numerical analysis benefits in the fabrication of three-dimensional scaffolds that can support more confirmation of the growth of cells. Our observation focuses on further cells' migration and growth phase beyond experiment data.

13.1. Introduction

The tissue engineering covers broad fields from rehabilitating wounded patients to enhancing patient care. In particular, clinical application of bone-scaffolds can be achieved by comprehensive multidisciplinary studies. But numerous experimental and numerical studies of scaffolds have attempted to design the optimal properties, usually, structurally based models are applied to describe and analyze the mechanics of tissue-engineered human body. These techniques mainly focus on stability of scaffolds' strength and stiffness properties, while human body is not like mechanic, the analysis of body systems require much more accurate calculations to avoid fatal drawbacks. Our research explores the design of bio-scaffolds including extracellular matrix within variety of parameters affecting our bodies to suggest how best to combine variables. We expect that the parameters which simulated and analyzed will indicate better and optimal appearance of the structure and also reveal oblique space causing serious problems such as necrosis in musculoskeletal tissue.

The Chapter is organized as follows. In Section 2 some biological states are assumed for model development. In Section 3, the governing model in axisymmetric system are stated for better investigation of the movement and growth of cells in three-dimension. And we show the discretization of the model and propose hybrid numerical method in Section 4. In Section 5, we present several numerical results in order to investigate the effect of parameters the better understanding of bio-scaffolding design. We summarize our results and present future research directions in Section 6.

13.2. Model development

It is hypothesized that a limiting molecule (critical molecule denoted by C), likely oxygen, diffuses across the fluid-scaffold interfaces and is consumed by cells located near each interface. The scaffold is assumed to be stable over the time course of the experiment. The total rate of consumption of this limiting molecule is proportional to the local cell density, and the specific rate of uptake by the cells is assumed to be proportional to the local concentration of the molecules.

$$\frac{\partial C(\mathbf{x},t)}{\partial t} = D\triangle C(\mathbf{x},t) - VC(\mathbf{x},t)U(\mathbf{x},t), \tag{13.1}$$

where $\mathbf{x} = (x, y) \in \Omega = (0, L_x) \times (0, L_y)$ scaffolds' domain. Here, parameters are represented as follows:

- $C(\mathbf{x}, t)$: concentration of the critical molecule, like oxygen, glucose, etc.[mole/mL],
- D: diffusivity of critical molecule [cm^2/s],
- $U(\mathbf{x}, t)$: cell density in the tissue [cell/mL],
- V: kinetic constant for the specific consumption rate of the critical molecule [mL/s/cell].

The growth of cells in tissue culture can be modeled using the logistic law. During the growth phase, the rate of cell growth is proportional to the cell density modified by the logistic law until the maximal cell density is reached. The specific rate of cell growth is assumed proportional to the concentration of the limiting molecule:

$$\frac{\partial U(\mathbf{x},t)}{\partial t} = \lambda C(\mathbf{x},t)U(\mathbf{x},t)\left(1 - \frac{U(\mathbf{x},t)}{U_m}\right) + M\triangle U(\mathbf{x},t) \tag{13.2}$$

where

- λ: kinetic constant for the specific rate of cell growth $[ml/mole/s]$,
- U_m: maximal cell density $[cell/ml]$,
- M: Cell mobility $[cm^2/s]$,

Now, we introduce dimensionless variables $C' = C/C_0, U' = U/U_{max}$, $M' = M/M_0$, $x' = x/L_x$, $y' = y/L_y$ and $t' = t/T$, where

$$T = \frac{1}{\lambda C_0}, \quad M_0 = \frac{L^2}{T}.$$

Then the Eq. (13.1) is replaced by non-dimensional variables.

$$\frac{C_0 \partial C'}{T \partial t'} = \frac{DC_0}{L^2}\triangle' C' - C_0 U_{max} V C' U'$$

resulting from canceling out and changing variables

$$\frac{\partial C'}{\partial t'} = K\triangle' C' - RC'U' \tag{13.3}$$

where two non-dimensional values are defined as

$$K = \frac{TD}{L^2}, \quad \text{and} \quad R = TVU_{max}$$

Also we derive the dimensionless equation with regard to U'.

$$\frac{U_{max}}{T}\frac{\partial U'}{\partial t'} = \lambda C' C_0 U U_{max}\left(1 - \frac{U_{max}U'}{U_{max}}\right) + M_0 M' \frac{U_{max}\Delta U'}{L^2},$$

then

$$\frac{\partial U'}{\partial t'} = C'U'(1 - U') + M'\Delta U' \tag{13.4}$$

After dropping the prime, Eqs. (13.3) and (13.4) become

$$\frac{\partial C}{\partial t} = K\triangle C - RCU, \tag{13.5}$$

$$\frac{\partial U}{\partial t} = CU(1-U) + M\triangle U \tag{13.6}$$

Since assuming to distribute cells in a scaffold uniformly and frequently replace the resource medium related C, we set the initial conditions of C

$$C(x,y,0) = 1, \qquad U(x,y,0) = U_0/U_m \tag{13.7}$$

where U_0 is the initial cell density. In typical cells, the growth rate λC_0 is 1/1day. Thus, the characteristic time for cell growth, $T = 1/(\lambda C_0) \sim 1$day, is much longer than the characteristic time for diffusion, $T_D = L^2/D \sim 10s$.

The system of partial differential equations (13.5) and (13.6) in the axisymmetric $(r-z)$ geometry are solved using the implicit finite difference scheme for concentration coupled with the operator splitting method for cell density. So we consider only two variables; r, the radial direction and z, the axial direction.

The singularity at the origin $r = 0$ in the discrete axisymmetric concentration equation is avoided by using a staggered mesh where the nutrient concentrations and cell densities are determined at cell center grids. The corresponding discrete linear system is solved efficiently using a multigrid method [122].

The governing equations in the axisymmetric geometry are

$$C_t = K\left[\frac{1}{r}(rC_r)_r + C_{zz}\right] - RCU, \tag{13.8}$$

$$U_t = M\left[\frac{1}{r}(rU_r)_r + U_{zz}\right] + CU(1-U). \tag{13.9}$$

In Fig. 13.1, the shaded region shows the computational domain with axisymmetric system.

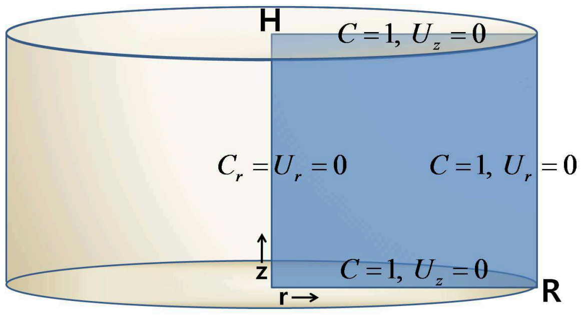


FIGURE 13.1. Computational domain.

We next specify the boundary conditions. Due to the symmetry at the column axis $r = 0$, the Neumann boundary conditions are applied, i.e., $C_r(0,z) = 0$ and $U_r(0,z) = 0$. At the rigid wall, $r = R$, $C(R,z) = 1$, $U_r(R,z) = 0$, where R is the radius of the domain. For the radial axis we also assume the following boundary conditions, i.e., $C(r,0) = C(r,H) = 1$, $U_z(r,0) = U_z(r,H) = 0$, where H is the height of the domain.

13.3. Numerical procedure

13.3.1. Discretization. Let us first discretize the given computational domain $\Omega = (0,R) \times (0,H)$ as a uniform grid with a space step $h = R/N_x = H/N_z$ and a time step $\Delta t = T/N_t$. Let us denote the numerical approximations of the solution by

$$\begin{aligned} C_{ik}^n \equiv C(x_i, z_k, t^n) &= C\left((i-0.5)h, (k-0.5)h, n\Delta t\right), \\ U_{ik}^n \equiv U(x_i, z_k, t^n) &= U\left((i-0.5)h, (k-0.5)h, n\Delta t\right), \end{aligned}$$

where $i = 0, \dots, N_x$, $k = 0, \dots, N_z$, and $n = 0, \dots, N_t$. And N_x, N_z, and N_t are the number of cells in r, z, and t directions, respectively.

13.3.2. Proposed numerical method. This Chapter considers an operator splitting method for governing equations (13.8) and (13.9). The basic idea of this method is to split a time step to two fractional time steps. These operator splitting method is easier to implement and efficient than several numerical solvers.

First, we solve Eq. (13.5) by applying the following operator splitting method as follows:

Step 1)

$$\begin{aligned} \frac{C_{ik}^* - C_{ik}^n}{\Delta t} &= K \triangle_h C_{ik}^* \qquad (13.10) \\ &= \frac{K}{h^2}\left[\frac{r_{i+\frac{1}{2},k}}{r_{ik}}(c_{i+1,k}^* - c_{ik}^*) - \frac{r_{i-\frac{1}{2},k}}{r_{ik}}(c_{ik}^* - c_{i-1,k}^*) + c_{i,k-1}^* - 2c_{ik}^* + c_{i,k+1}^*\right] \end{aligned}$$

We solve Eq. (13.10) as the implicit time scheme for C , centered difference for the second space derivatives by multigrid method [116, 122].

Then, the remaining term in Eq. (13.5) is

$$\frac{C_{ik}^{n+1} - C_{ik}^*}{\Delta t} = -RC_{ik}^* U_{ik}^n. \qquad (13.11)$$

The above Eq. (13.11) is an approximation of the equation

$$\frac{\partial C}{\partial t} = -RCU \qquad (13.12)$$

by an implicit Euler's method with an initial condition C^*. We can solve Eq. (13.12) analytically by the method of separation of variables [123] and the solution is given as

Step 2)

$$C_{ik}^{n+1} = e^{-\Delta t R C_{ik}^* U_{ik}^n}. \qquad (13.13)$$

Second, to solve Eq. (13.6), we propose the following operator splitting scheme.

Step 3)

$$\begin{aligned} \frac{U_{ik}^* - U_{ik}^n}{\Delta t} &= M \triangle_h U_{ik}^*, \qquad (13.14) \\ \frac{U_{ik}^{n+1} - U_{ik}^*}{\Delta t} &= C_{ik}^{n+1} U_{ik}^{n+1}(1 - U_{ik}^{n+1}). \qquad (13.15) \end{aligned}$$

We solve the implicit discrete Eq. (13.14) by multigrid method [116, 122]. We can consider Eq. (13.15) is an approximation of the equation

$$\frac{\partial U}{\partial t} = CU(1-U) \tag{13.16}$$

by an implicit Euler's method with an initial condition U^*. We can solve Eq. (13.16) analytically by the method of separation of variables [123] and the solution is given as

Step 4)

$$U_{ik}^{n+1} = \frac{U_{ik}^*}{U_{ik}^* + (1-U_{ik}^*)e^{-\Delta t C_{ik}^{n+1}}}.$$

As the conclusion, our proposed numerical scheme to Eqs. (13.5) and (13.6) consists of *Steps 1)*, *2)*, *3)*, and *4)* with the following boundary conditions : $C_r = U_r = 0$ at $r = 0$, $C = 1, U_r = 0$ at $r = R$, and $C = 1, U_z = 0$ at $z = 0$ and H. Figure 13.2 shows our proposed numerical algorithm schematically.

$$\begin{tikzcd}
(C^n, U^n) \arrow[d, "C_t = K\triangle C \text{ (multigrid method)}"] \\
(C^*, U^n) \arrow[d, "C_t = -RCU \text{ (analytic solution)}"] \\
(C^{n+1}, U^n) \arrow[d, "U_t = M\triangle U \text{ (multigrid method)}"] \\
(C^{n+1}, U^*) \arrow[d, "U_t = CU(1-U) \text{ (analytic solution)}"] \\
(C^{n+1}, U^{n+1})
\end{tikzcd}$$

FIGURE 13.2. A hybrid numerical method

13.4. Computational results

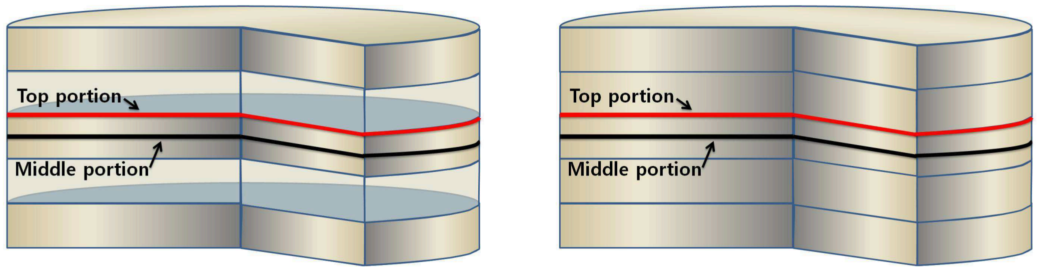
In this section, we perform several numerical experiments.

13.4.1. Initial cell seeding. Scaffolds were stacked in two patterns shown in Figure 13.6. Construct A contained five, stacked scaffolds, alternating between cell-seeded and unseeded. Construct B contained five scaffolds all seeded with cells.

The initial conditions are

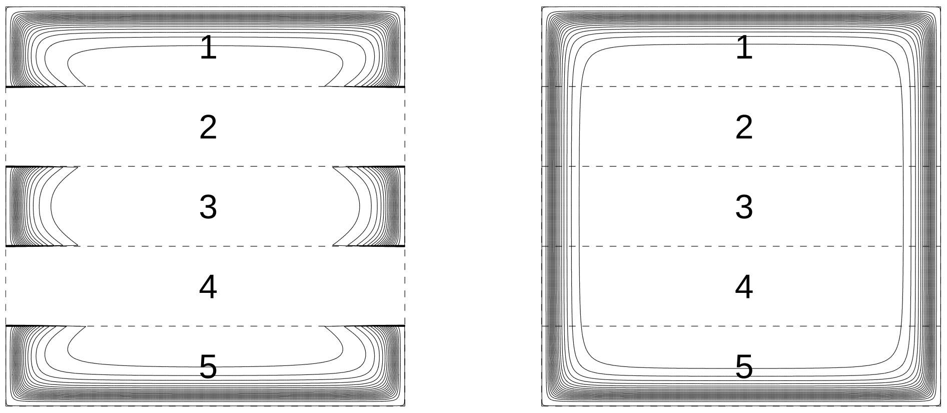
$$\begin{aligned} C(x,z,0) &= 1, \\ U(x,z,0) &= U_0/U_{max} = 0.016, \end{aligned}$$

where U_0 is the initial cell density.



(a) Contstruct A (b) Contstruct B

FIGURE 13.3. Stacking patterns for constructs A (alternating seeding) and B (uniform seeding).



(a) Contstruct A (b) Contstruct B

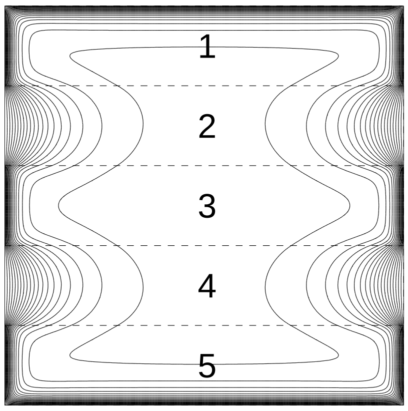
FIGURE 13.4. Numerical simulation of the cell density for the stacked scaffolds in construct A (a) and construct B (b) on day 10 with $L = 167\mu$m. Contours increase by 6.5×10^5cells/cm^3 at the fluid-scaffold boundary. The third scaffold is represented in the region between 2mm and 3mm in height.

13.4.2. Numerical simulation. In Fig. 13.6, error bar represents the experimental data of the standard deviation of measured cell density. From the experimental data, the cell density in the top and bottom portions of the third scaffold is slightly higher than that in the middle portion of the third scaffold.

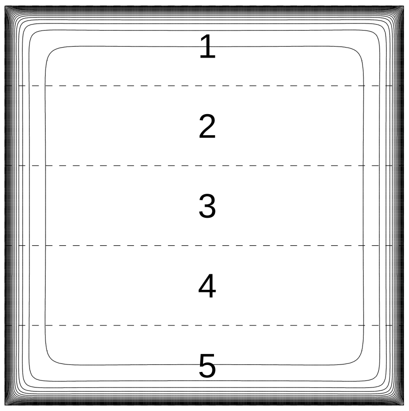
13.4.2.1. *Convergence test.* Fig. 13.9 represents the numerical results at $T = 30$ according to the time step Δt and space step h. As shown in Fig. 13.9, the numerical results, which have larger space and time step size than h and Δt, have different results in top portion for the third scaffold. Therefore, for efficient numerical tests, we use the large time step $\Delta t = 0.01$ and space step $h = 1/128$.

with $L = 167\mu m$

13.4.2.2. *Effect of K.* In this model, K represents the diffusivity of the critical molecule. To investigate the effect of K, we tested several simulations.



(a) Contstruct A



(b) Contstruct B

FIGURE 13.5. Numerical simulation of the oxygen concentration for the stacked scaffolds in construct A (a) and construct B (b) on day 10 with $L = 167\mu$m. Contours increase by 5 nmole/mL from 5 nmole/mL to 100 nmole/mL at the fluid-scaffold boundary. The third scaffold is represented in the region between 2mm and 3mm in height.

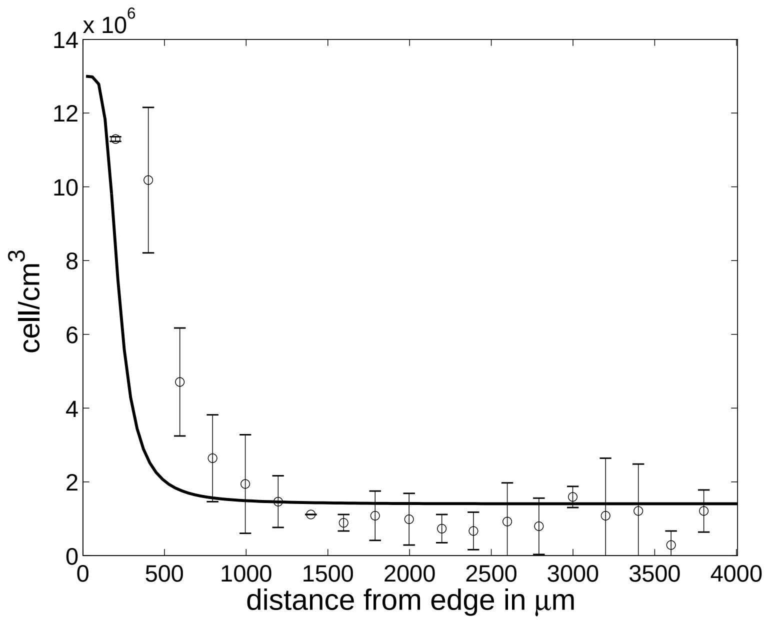
13.4.2.3. *Effect of R.* In this model, R represents the consumption rate of the critical molecule. To investigate the effect of R, we tested several simulations.

13.4.2.4. *Effect of M.* In this model, M represents the mobility rate of cell density. To investigate the effect of M, we tested several simulations.

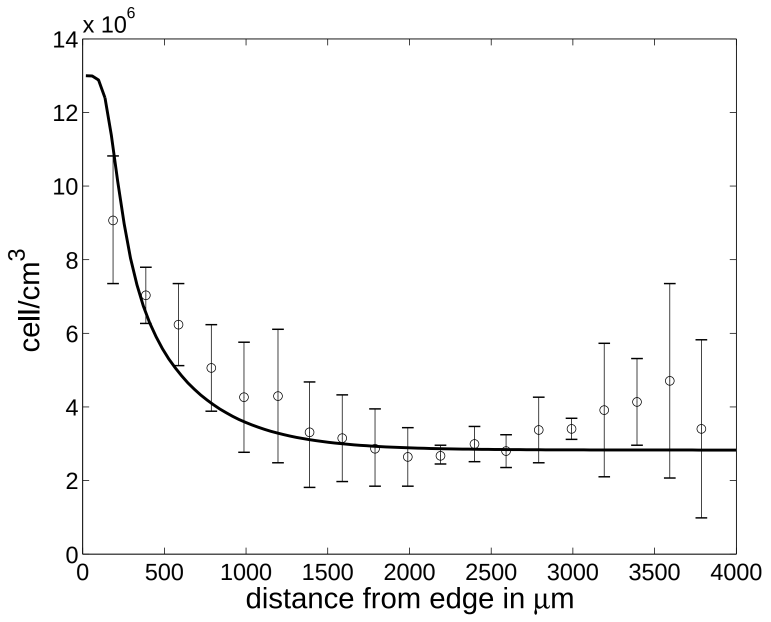
13.4.2.5. *Effect of T.* In this model, T represents the value of the time scale. To investigate the effect of T, we tested several simulations.

13.5. Conclusions

In this Chapter, we proposed mathematical and numerical models for the cell growths of three-dimensional scaffolds. We included cell migrations in our model. For the numerical solution we utilize a multigrid method to achieve improved results. Like below figures we could approximate more accurate simulated results compared with experiment results.

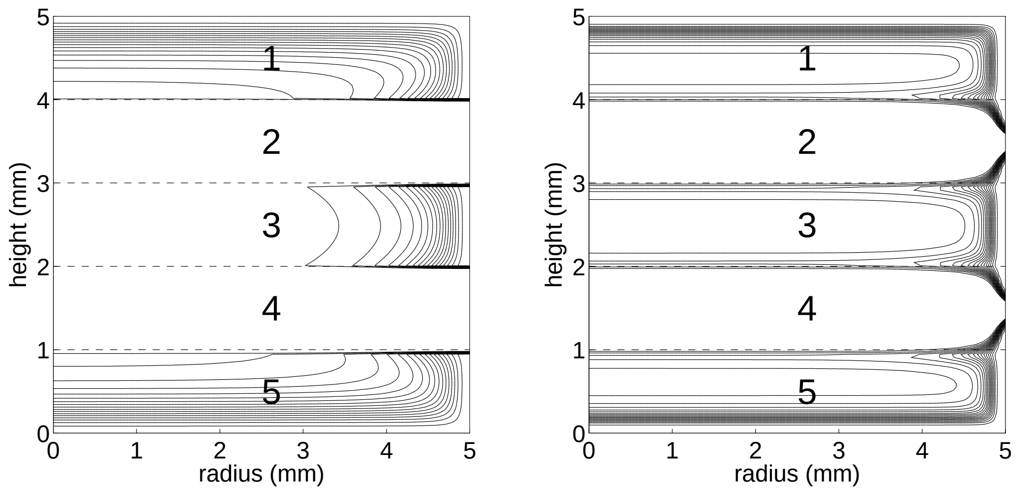


(a) Mid portion



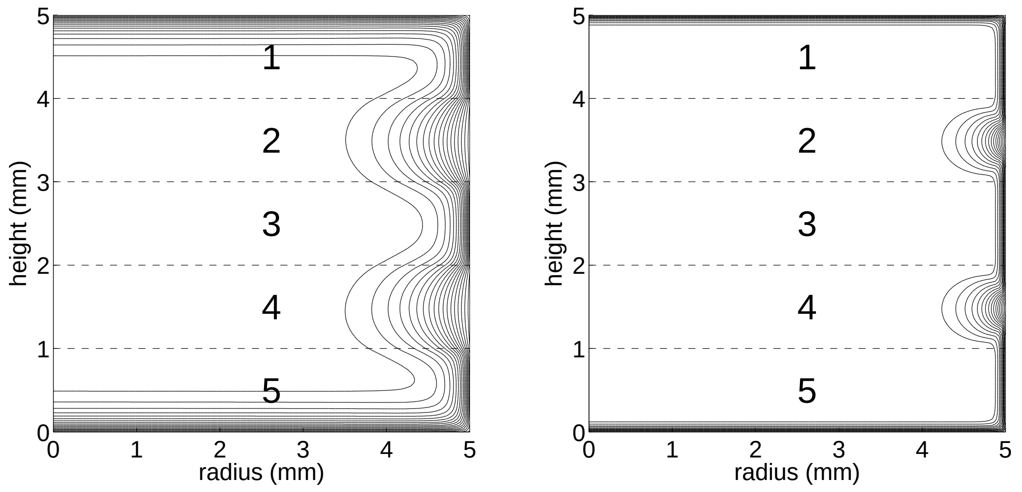
(b) Top portion

FIGURE 13.6. Measured and simulated cell density for the third scaffold in construct A on day 10. (A) Middle portions. (B) Top and bottom portions. Error bar represents the standard deviation of measured cell density. Solid line represents numerical simulation with $L = 167\mu m$, $K = 1.7$, $R = 13.2$, $\Delta t = 0.01$, $T = 30.0$ and $M = 0.001$.



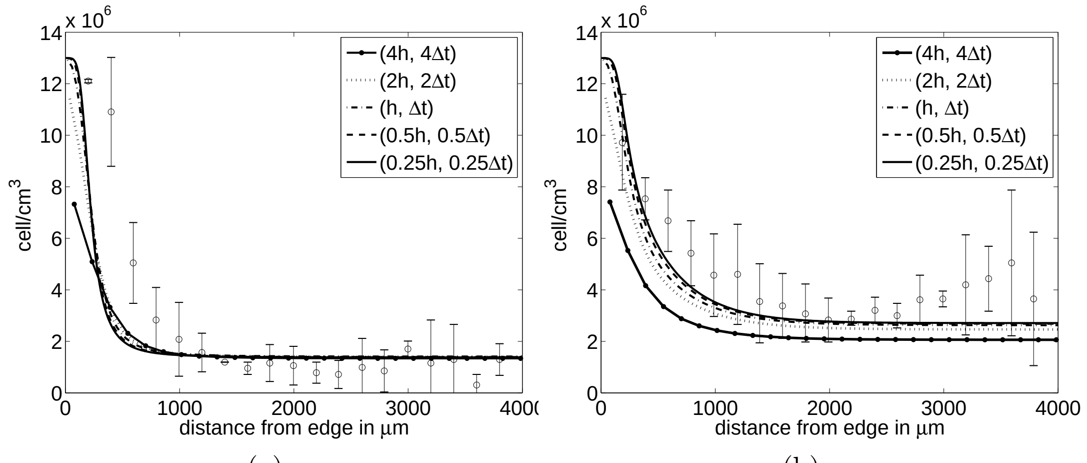
(a) previous result (b) our result

FIGURE 13.7. Numerical simulation of the cell density



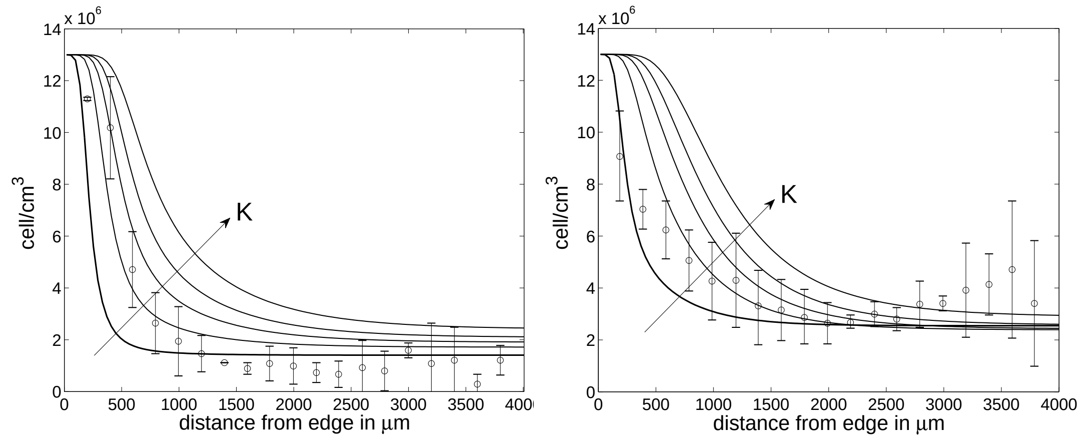
(a) previous result (b) our result

FIGURE 13.8. Numerical simulation of the concentration



(a) (b)

FIGURE 13.9. Numerical results of cell density on (a) middle and (b) top portion for the third scaffold at $T = 30$ according to the time step Δt and space step h. The following parameters used: $L = 167\mu m$, $K = 1.7,\ R = 13.2,\ \Delta t = 0.1, h = 1/128,\ T = 30.0$ and $M = 0.001$.



(a) (b)

FIGURE 13.10. Effect of varying the diffusivity of the concentration of the critical molecule on simulated cell density for the third scaffold in alternating seeding with $L = 167\mu m$. The simulated cell density for the middle portion (a) and the top and bottom portions (b) with $K = 1.7,\ 5.1,\ 8.5,\ 11.9,$and 17.0.

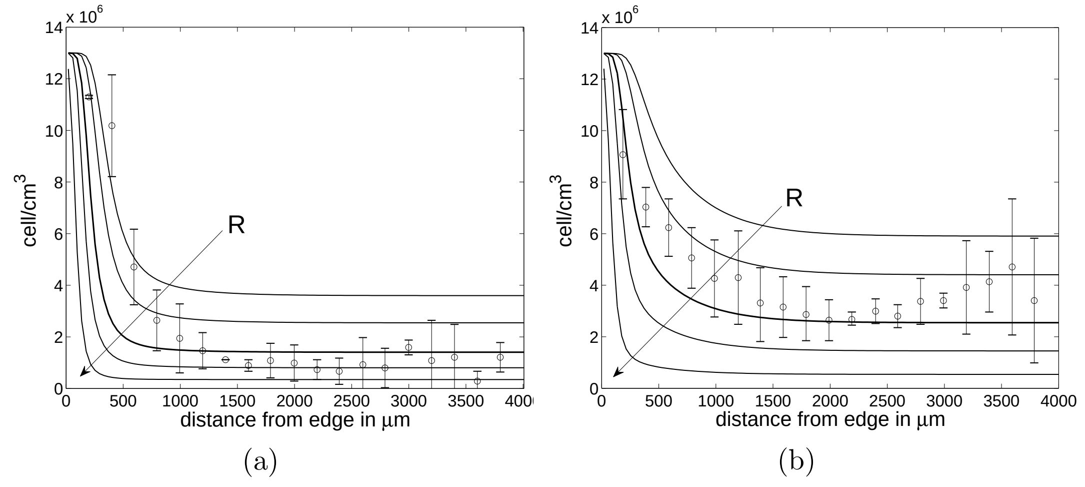


FIGURE 13.11. Effect of varying the consumption rate of the critical molecule on simulated cell density for the third scaffold in alternating seeding with $L = 167\mu m$. The simulated cell density for the middle portion (a) and the top and bottom portions (b) with $R = 4.4,\ 6.6,\ 13.2,\ 26.4$, and 105.6.

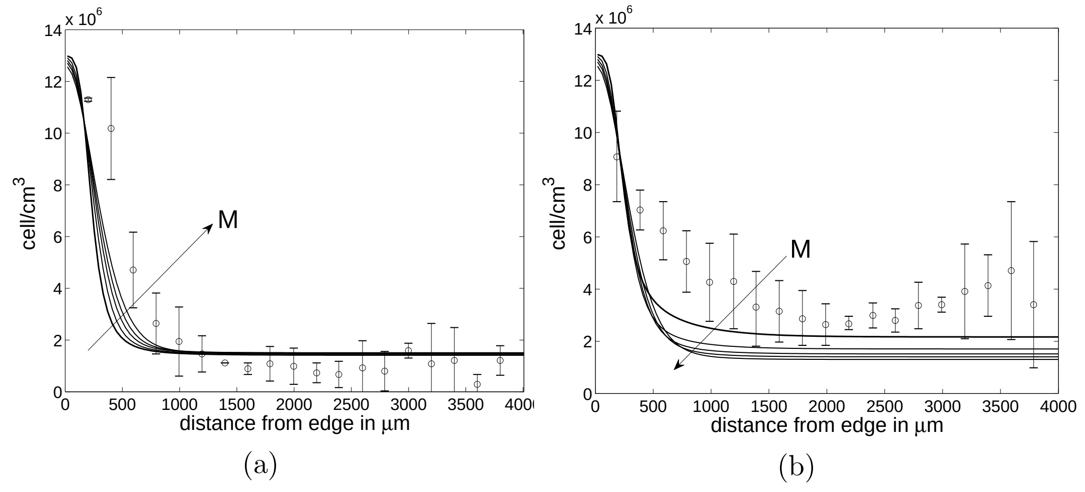
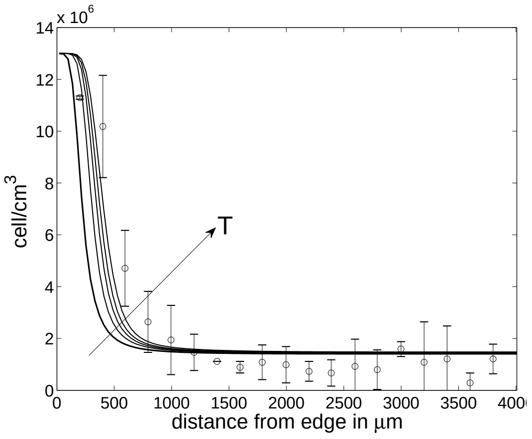
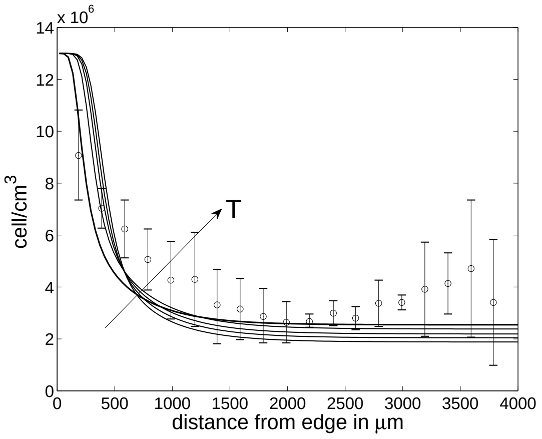


FIGURE 13.12. Effect of varying the mobility rate of cell density for the third scaffold in alternating seeding with $L = 167\mu m$. The simulated cell density for the middle portion (a) and the top and bottom portions (b) with $M = 0.001,\ 0.003,\ 0.005,\ 0.007$, and 0.01.



(a)



(b)

Figure 13.13. Effect of varying the value of the time scale for the third scaffold in alternating seeding with $L = 167\mu m$. The simulated cell density for the middle portion (a) and the top and bottom portions (b) with $T = 30,\ 90,\ 150,\ 210$, and 300.

Chapter 14

On the Keller-Segel equations arising in Mathematical Biology

We consider the Neumann initial boundary problem for a chemotaxis-systems with a logarithmic chemotactic sensitivity funtion and a non-diffusing chemical in a smooth bounded domain $\Omega \subset \mathbb{R}^n$, $n \geq 1$.

14.1. Introduction

In this Chapter, we consider a chemotaxis system with a logarithmic chemotactic sensitivity and a non-diffusing chemical,

$$\begin{cases} u_t = \Delta u - \nabla \cdot (u \nabla \log w), & x \in \Omega,\ t > 0, \\ w_t = u w^{\lambda}, & x \in \Omega,\ t > 0, \\ \frac{\partial u}{\partial \nu} = \frac{\partial w}{\partial \nu} = 0, & x \in \partial\Omega,\ t > 0, \\ u(x,0) = u_0(x), \quad w(x,0) = w_0(x) & x \in \Omega \end{cases} \tag{14.1}$$

in a smooth bounded domain $\Omega \subset \mathbb{R}^n$ for arbitrary space dimensions $n \geq 1$ and some $0 < \lambda < 1$. The PDE system in (14.1) is used in mathematical biology of lattice models for slime-trail following of myxobacteria. Here, $u(x,t)$ is the chemotatic species and $w(x,t)$ is the non-diffusive memory [117]. For the solutions of (14.1), in case $\lambda = 0$, $u(x,t)$ is uniformly bounded and $w(x,t) \to \infty$ as $t \to \infty$ for any dimension $n \geq 1$ is proved analytically in [118] and for $u_\infty(x)(:= \lim_{t\to\infty} u(x,t))$ is asymptotically a steady state for one dimension is proved analytically in [119] and in case $\lambda = 1$, finite time blow-up solutions and global solutions are shown analytically and numerically in [118, 120, 121]. And also known results, qualitative behavior of this model for one dimension proved in [119].

Similar proceeding as in [119], by setting $v = \frac{1}{1-\lambda} w^{1-\lambda}$ we obtain

$$\begin{cases} u_t = \Delta u - \frac{1}{1-\lambda} \nabla \cdot (u \nabla \log v), & x \in \Omega,\ t > 0, \\ v_t = u, & x \in \Omega,\ t > 0, \\ \frac{\partial u}{\partial \nu} = \frac{\partial v}{\partial \nu} = 0, & x \in \partial\Omega,\ t > 0, \\ u(x,0) = u_0(x), \quad v(x,0) = v_0(x) & x \in \Omega \end{cases} \tag{14.2}$$

For this transformed PDE systems, we can simplify second equation of (14.1) more easier to calculate. For example, integrating the equations of (14.2) on spatial domain Ω shows that $u(x,t)$ has mass conservation and $v(x,t)$ has increasing of mass as t becomes larger which implies similar conclusion to $u(x,t)$ and $w(x,t)$ in (14.1).

The plan of the present Chapter is as follows. In Section 2, we prove the existence of local-in-time smooth solution of (14.2) satisfying $u(x,t)$ is nonnegative and $v(x,t)$ has positive lower bound. In Section 3, we study the uniqueness and regularity of solutions of (14.2) until time goes to maxmal time T_{max} which is whether ∞ or not isn't proved yet. We suggests that blow-up in finite time or globally exist of solutions depends on dimension and initial data conditions.

14.2. Numerical Procedure

In this section, we describe the numerical procedure. Especially, we introduce a numerical algorithm for proposed operator splitting scheme. (We consider only the system in the two-dimensional case. But, in one- and three- dimensional case, we can easily apply by the two-dimensional case.)

14.2.1. Discretization. Let us first discretize the given computational domain $\Omega = (0,1) \times (0,1)$ as a uniform grid with a space step $h = 1/N_x = 1/N_y$ and a time step $\Delta t = T/N_t$. Let us denote the numerical approximations of the solution by

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

where $i = 0, \dots, N_x,\ j = 0, \dots, N_y$, and $n = 0, \dots, N_t$. And N_x, N_y, and N_t are the number of cells in x, y, and t directions, respectively.

14.2.2. Proposed numerical method. Our strategy for solving the system (14.1) is a fractional time step scheme having two parts:

(Step 1)

$$\begin{aligned} \frac{u_{ij}^{n+1} - u_{ij}^n}{\Delta t} &= \frac{u_{i+1,j}^{n+1} + u_{i-1,j}^{n+1} - 4u_{ij}^{n+1} + u_{i,j+1}^{n+1} + u_{i,j-1}^{n+1}}{h^2} \\ &- \left[\left(\frac{u_{ij}^{n+1} + u_{i+1,j}^{n+1}}{2}\right)\log\left(\frac{w_{i+1,j}^n}{w_{ij}^n}\right) - \left(\frac{u_{i-1,j}^{n+1} + u_{ij}^{n+1}}{2}\right)\log\left(\frac{w_{ij}^n}{w_{i-1,j}^n}\right)\right]/h^2 \\ &- \left[\left(\frac{u_{ij}^{n+1} + u_{i,j+1}^{n+1}}{2}\right)\log\left(\frac{w_{i,j+1}^n}{w_{ij}^n}\right) - \left(\frac{u_{i,j-1}^{n+1} + u_{ij}^{n+1}}{2}\right)\log\left(\frac{w_{ij}^n}{w_{i,j-1}^n}\right)\right]/h^2. \end{aligned} \tag{14.3}$$

First, we solve $u_t = \Delta u - \nabla \cdot (u \nabla(\log w))$ as the implicit time scheme for u , centered difference for the space derivatives by multigrid method [116, 122].

(Step 2)

Then the remaining equation in system (14.1) is

$$\frac{w_{ij}^{n+1} - w_{ij}^n}{\Delta t} = u_{ij}^{n+1}\left(w_{ij}^{n+1}\right)^{\lambda} \tag{14.4}$$

The above Eq. (14.4) is an approximation of the equation

$$\frac{\partial w}{\partial t} = u w^{\lambda} \tag{14.5}$$

by an implicit Euler's method with an initial condition w^*. We can solve Eq. (14.5) analytically by the method of separation of variables [123] and the solution is given as

$$w_{ij}^{n+1} = \left[u_{ij}^n(1-\alpha)\Delta t + (w_{ij}^n)^{1-\alpha}\right]^{\frac{1}{1-\alpha}} \tag{14.6}$$

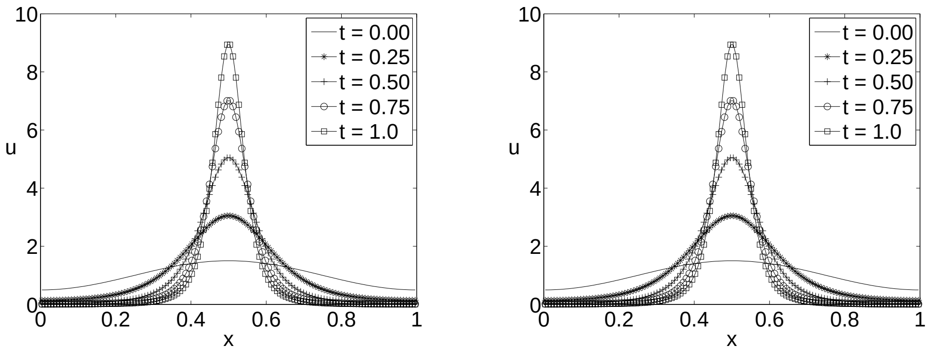
As the conclusion, our proposed numerical scheme to system (14.1) consists of *Steps 1)* and *2)*.

14.3. Numerical results

In this section, we perform several numerical experiments.

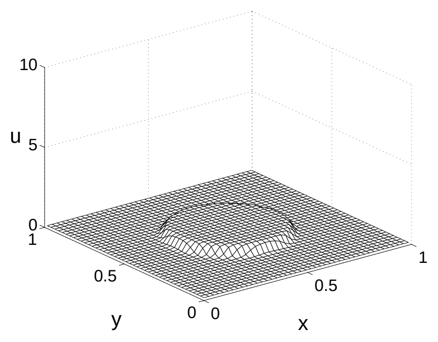
14.3.1. One-dimension. Initial condition :

$$\begin{cases} u(x,0) = 0.75 + 0.5\tanh((0.25 - \sqrt{(x-0.5)^2})), \\ w(x,0) = 0.0001. \end{cases} \tag{14.7}$$

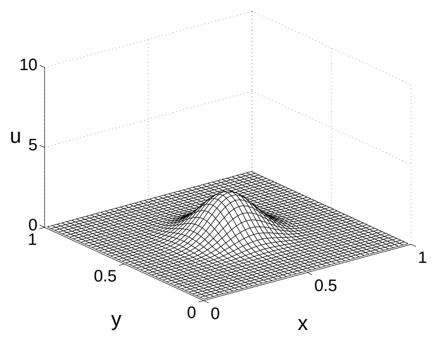


(a) (b)

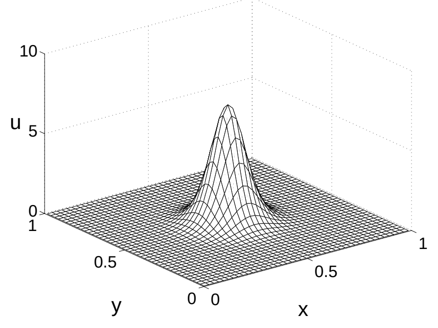
FIGURE 14.1. (a) Bacteria density $u(x,t)$ and (b) substrate concentration $w(x,t)$ with $\lambda = 0.5$, $N_x = 128$ and $L = 1.0$.



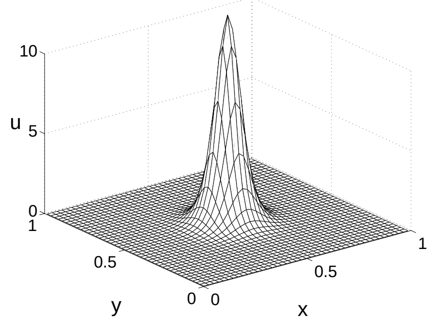
(a) u (t=0)



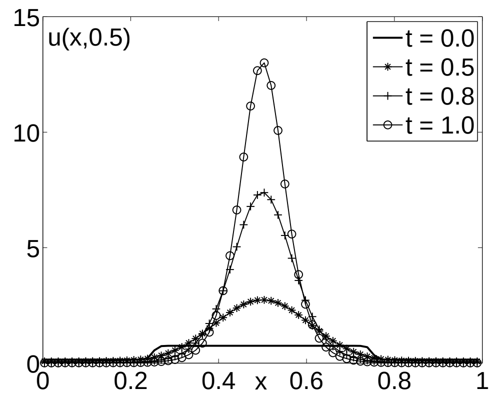
(b) u (t=50)



(c) u (t=80)



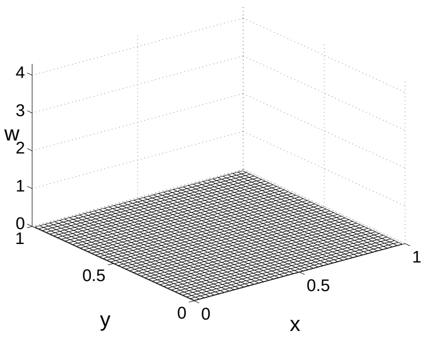
(d) u (t=100)



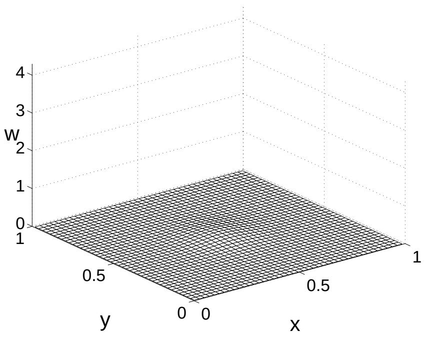
(e) cross section

FIGURE 14.2. Bacteria density $u(x, y, t)$ with $\lambda = 0.5$, $N_x = N_y = 128$ and $L = 1.0$.

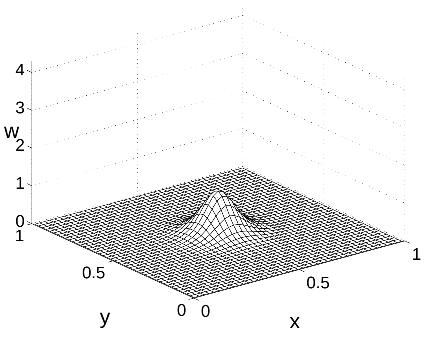
14.3.2. Two-dimension.



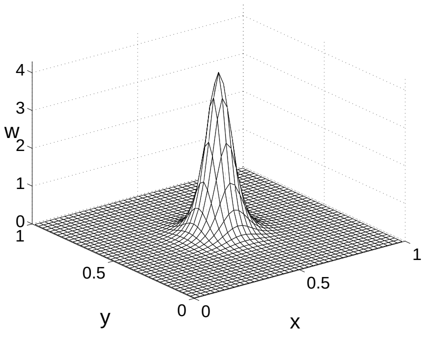
(a) w (t=0)



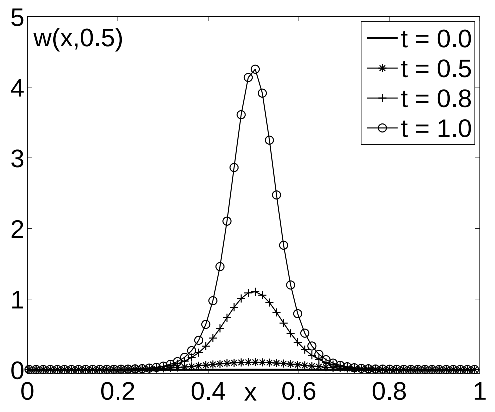
(b) w (t=50)



(c) w (t=80)



(d) w (t=100)



(e) cross section

FIGURE 14.3. Substrate concentration $w(x, y, t)$ with $\lambda = 0.5$, $N_x = N_y = 128$ and $L = 1.0$.

Chapter 15

Conclusion

(1) Phase-field model
First, we reviewed a derivation of the AC equation as a gradient flow and showed that a numerical scheme for the AC equation is unconditionally gradient stable by using eigenvalues of the Hessian matrix of the energy functional. We also showed that the decrease of the discrete total energy functional implies the pointwise boundedness of the numerical solution for the AC equation. We investigated a variety of phenomena associated with the AC equation. We have uncovered a traveling wave solution to the AC equation and found that its speed depends linearly on the interfacial energy parameter.
Secondly, we reviewed various numerical methods for solving the CH equation. We described the discrete scheme and its properties, and presented the multigrid method for the fully discrete system. Also, we provided a C program code for the CH equation. We hope that the code will play an useful role in the modeling and simulation for the phase-field models. And we present details of the efficient computational scheme of the phase-field model, CH equation, for the block copolymer.
Lastly, we have shown that our automatic switching algorithm achieves faster inpainting of binary images than the previous trial and error algorithm. Therefore, inpainting region is reconstructed more efficiently and faster than previous method. The developed automatic algorithm can be applied to calculating option pricing such as the Black-Scholes equations accurately and efficiently.

(2) Landau-Lifshitz model
We have proposed a Crank-Nicolson time-stepping procedure for LL equation which has a second-order convergence in time and space. We overcame the difficulties with CN scheme associated with LL equation by a cancelation. We used a nonlinear multigrid method for handling the nonlinearities of the discrete system at each time step. We validated our numerical algorithm by various numerical experiments. We tested the second-order convergence and an energy conservation of the proposed scheme. We also showed that the time step restriction for the stability is less restrictive than the accuracy. As future research, the full version of Landau-Liftshitz equation will be investigated.

(3) Computational finance
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.
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.
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.
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, an accurate and efficient numerical method for the Black-Scholes equations is derived in this Chapter. 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.

(4) Biomathematics
We investigate the efficient and accurate finite difference scheme to find numerical solutions of the ratio-dependent Michaelis-Menten type predator-prey model. To do this, we have considered constant stable equilibrium solutions and the conditions for Turing instability of the solutions. With Turing instability occurring conditions, we have presented the semi-implicit scheme to get numerical results of the prey and the predator solutions. Since we use the semi-implicit scheme, the total time iteration is much smaller and the scheme is stable. Also, we have proved the positivity and boundedness of the prey and predator solutions only depending on time scale Δt. To show the superiority of our proposed scheme, we compare an explicit schemes dependency

of time and space with our scheme. Moreover, we have shown numerical evidence for the non-constant stationary solutions not only with small amplitudes but also large enough amplitudes. Since our proposed scheme assures the stability, we expect that the proposed scheme is useful to demonstrating the biological system numerically.
And we proposed mathematical and numerical models for the cell growths of three-dimensional scaffolds. We included cell migrations in our model. For the numerical solution we utilize a multigrid method to achieve improved results. Like below figures we could approximate more accurate simulated results compared with experiment results.

Bibliography

[1] S.M. Allen and J.W. Cahn, Acta Metall. 27 (1979) 1085.
[2] A.S. Almgren et al., J. Comput. Phys. 142 (1998) 1.
[3] M. Beneš, V. Chalupecký, K. Mikula, Appl. Numer. Math. 51 (2004) 187.
[4] M.J. Berger, P. Colella, J. Comput. Phys. 82 (1989) 64.
[5] P. Colella, M. Dorr, D. Wake, J. Comput. Phys. 152 (1999) 550.
[6] C. Cowan, M.S. Thesis, Simon Fraser University, Canada, 2005.
[7] J.A. Dobrosotskaya, A.L. Bertozzi, IEEE Trans. Im. Proc. 17 (2008) 657.
[8] X. Feng, A. Prohl, Numer. Math. 94 (2003) 33.
[9] P.C. Fife, E. J. Diff. Eqns. 2000 (2000) 1.
[10] M. Gokieli, L. Marcinkowski, Nonlinear Anal. 63 (2005) 1143.
[11] Y. He, Y. Liu, T. Tang, Appl. Numer. Math. 57 (2007) 616.
[12] J.S. Kim, J. Korean Phys. Soc. 49 (2006) 1903.
[13] D.F. Martin, P. Colella, M. Anghel, F. Alexander, Computing in Science and Eng. 7 (2005) 24.
[14] A.A. Wheeler, W.J. Boettinger, G.B. McFadden, Phys. Rev. A 45 (1992) 7424.
[15] S.M. Wise, J.S. Kim, J.S. Lowengrub, J. Comput. Phys. 226 (2007) 414.
[16] D. Jacqmin, Calculation of two-phase Navier-Stokes flows using phase-field modeling, J. Comput. Phys. 155 (1999) 96-127.
[17] B. Nadiga, S. Zaleski, Investigations of a two-phase fluid model, Eur. J. Mech. B/Fluids 15 (1996) 885-896.
[18] D. Jasnow, J. Vinãls, Coarse-grained description of thermo-capillary flow, Phys. Fluids 8 (1996) 660-669.
[19] M. Verschueren, F.N. van de Vosse, H.E.H. Meijer, Diffuse-interface modeling of thermo-capillary flow instabilities in a Hele-Shaw cell, J. Fluid Mech. 434 (2001) 153-166.
[20] R. Chella, J. Vinãls, Mixing of a two-phase fluid by cavity flow, Phys. Rev. E 53 (1996) 3832-3840.
[21] D. Jacqmin, Contact-line dynamics of a diffuse fluid interface, J. Fluid Mech. 402 (2000) 57-88.
[22] D. Anderson, G.B. McFadden, A diffuse-interface description of internal waves in a near-critical fluid, Phys. Fluids 9 (1997) 1870-1879.
[23] L. de Sobrino, Note on capillary waves in the gradient theory of interfaces, Can. J. Phys. 63 (1985) 1132-1133.
[24] L. de Sobrino, J. Peternelj, On capillary waves in the gradient theory of interfaces, Can. J. Phys. 63 (1985) 131-134.
[25] M. Copetti, Numerical experiments of phase separation in ternary mixtures, Math. Comput. Simulation 52 (2000) 41-51.
[26] F. Dell'Isola, H. Gouin, G. Rotoli, Nucleation of spherical shell-like interfaces by second gradient theory: numerical simulations, Eur. J. Mech. B/Fluids 15 (1996) 545-568.
[27] M.E. Gurtin, D. Polignone, J. Vinãls, Two-phase binary fluids and immiscible fluids described by an order parameter, Math. Models Methods Appl. Sci. 6 (1996) 815-831.
[28] M.E. Gurtin, Generalized Ginzburg-Landau and Cahn-Hilliard equations based on a microforce balance, Phys. D 92 (1996) 178-192.
[29] C.M. Elliott, D.A. French, Numerical studies of the Cahn-Hilliard equation for phase separation, IMA J. Appl. Math. 38 (1987) 97-128.
[30] C.M. Elliott, D.A. French, A nonconforming finite-element method for the two-dimensional Cahn-Hilliard equation, SIAM J. Numer. Anal. 26 (1989) 884-903.

[31] C.M. Elliott, S. Larsson, Error estimates with smooth and nonsmooth data for a finite element method for the Cahn-Hilliard equation, Math. Comp. 58 (1992) 603-630.

[32] E.J. Dean, R. Glowinski, D.A. Trevas, An approximate factorization/least squares solution method for a mixed finite element approximation of the Cahn-Hilliard equation, Japan J. Indust. Appl. Math. 13 (1996) 495-517.

[33] D. Kay, R. Welford, A multigrid finite element solver for the Cahn-Hilliard equation, J. Comput. Phys. 212 (2006) 288-304.

[34] L. Banas, R. Nürnberg, Adaptive finite element methods for Cahn-Hilliard equations, J. Comput. Appl. Math. 218 (2008) 2-11.

[35] D. Furihata, A stable and conservative finite difference scheme for the Cahn-Hilliard equation, Numer. Math. 87 (2001) 675-699.

[36] S.M. Choo, S.K. Chung, Conservative nonlinear difference scheme for the Cahn-Hilliard equation, Comput. Math. Appl. 36 (1998) 31-39.

[37] S.M. Choo, S.K. Chung, K.I. Kim, Conservative nonlinear difference scheme for the Cahn-Hilliard equation: II, Comput. Math. Appl. 39 (2000) 229-243.

[38] S. Zhou, M.Y. Wang, Multimaterial structural topology optimization with a generalized Cahn-Hilliard model of multiphase transition, Struct. Multidisc. Optim. 33 (2007) 89-111.

[39] J. Kim, A numerical method for the Cahn-Hilliard equation with a variable mobility, Commun. Nonlinear Sci. Numer. Simul. 12 (2007) 1560-1571.

[40] Y. Feldman, A.Yu Gelfgat, On pressure-velocity coupled time-integration of incompressible Navier-Stokes equations using direct inversion of Stokes operator or accelerated multigrid technique, Comput. Struct. 87 (2009) 710-720.

[41] A. Frangi, P. Faure-Ragani, L. Ghezzi, Magneto-mechanical simulations by a coupled fast multipole method-finite element method and multigrid solvers, Comput. Struct. 83 (2005) 718-726.

[42] T. Hwang, I.D. Parsons, Parallel implementation and performance of multigrid algorithms for solving eigenvalue problems, Comput. Struct. 50 (1994) 325-336.

[43] D.J. Eyre, in: Computational and Mathematical Models of Microstructural Evolution, The Material Research Society, Warrendale, PA, 1998, pp. 39-46.

[44] D. Furihata, T. Onda, M. Mori, A finite difference scheme for the Cahn-Hilliard equation based on a Lyapunov functional, GAKUTO Int. Series, Math. Sci. Appl. 2 (1993) 347-358.

[45] M. Dehghan, Finite difference procedures for solving a problem arising in modeling and design of certain optoelectronic devices, Math. Comput. Simulation 71 (2006) 16-30.

[46] Cahn JW, On spinodal decomposition. Acta Metall 1961;9:795-801.

[47] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, *Image inpainting*, Proc. of SIGGRAPH 2000, New Orleans, USA 2000.

[48] A. Bertozzi, S. Esedoglu, and A. Gillette, *Inpainting of binary images using the Cahn-Hilliard equation*, IEEE Trans. Image Proc., **16** (2007), 285-291.

[49] A. Bertozzi, S. Esedoglu, and A. Gillette, *Analysis of a two-scale Cahn-Hilliard model for image inpainting*, Multiscale Modeling and Simulation, **6** (2007), 913-936.

[50] M. Burger, L. He, and C. Schöenlieb, *Cahn-Hilliard inpainting and a generalization for grayvalue images*, UCLA CAM report 08-41, 2008.

[51] J. W. Cahn and J. E. Hilliard, *Free energy of a nonuniform system. I. interfacial free energy*, J. Chem. Phys., **28** (1958), 258-267.

[52] D. J. Eyre, http://www.math.utah.edu/∼eyre/research/methods/stable.ps.

[53] D. J. Eyre, in *Computational and mathematical models of microstructural evolution*, The Material Research Society, Warrendale, PA, (1998), 39-46.

[54] J. L. Blue and M. R. Scheinfein, Using multipoles decreases computation time for magnetic self-energy, IEEE Trans. Magn. 27, 4778 (1991).

[55] I. Cimrák, A survey on the numerics and computations for the Landau-Lifshitz equation of micromagnetism, Arch. Comput. Methods Engrg. 15 (3) (2008) 277-309.

[56] I. Cimrák, Convergence result for the constraint preserving mid-point scheme for micromagnetism, Journal of computational and applied mathematics, 228 (2009) 238-246.

[57] J. Fidler and T. Schrefl, Micromagnetic modelling.The current state of the art, J. Phys. D: Appl. Phys. 33, R135 (2000).

[58] M. Jones and J. J. Miles, An accurate and efficient 3D micromagnetic simulation of metal evaporated tape, J. Magn. Magn. Mater 171, 190 (1997).

[59] M. Kruzik, A. Prohl, Recent developments in the modeling, analysis, and numerics of ferromagnetism, Siam Review, vol 48, 3 (2006) 439-483.

[60] L. Landau and E. Lifshitz, *On the theory of the dispersion of magnetic permeability in ferromagnetic bodies*, Physikalische Zeitchrift der Sowjetunion **8**, 153(1935).

[61] G. Sun and C.W. Trueman, Unconditionally stable Crank-Nicolson scheme for solving the two-dimensional Maxwell's equations, IEE Electron Lett 39 (2003), 595-597.

[62] X. P. Wang, C. J. Garcia-Cervera, E. Weinan, *A Gauss-Seidel projection method for micromagnetics simulations*, Journal of Computational Physics, Vol. 171, 357-372, 2001.

[63] J.C. Hull, *Options, Futures and Others*, Prentice Hall, 2003.

[64] 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.

[65] Y.K. Kwok, *Mathematical Models of Financial Derivatives*, Springer, 1998.

[66] C. Reisinger and G. Wittum, *On multigrid for anisotropic equations and variational inequalities*, Comput. Visual. Sci. **7** (2004), 189–197.

[67] H. Foester, K. Witsch, On efficient multigrid software for elliptic problems on rectangular domains, Math. Comput. Simul. 23 (1981) 293-298.

[68] R. Heynen, H. Kat, Brick by Brick, Risk Mag. 9 (1996) 28-31.

[69] S. Ikonen, J. Toivanen, Operator splitting methods for American option pricing, Appl. Math. Lett. 17 (2004) 809-814.

[70] Gh. Juncu, C. Popa, Preconditioning by Gram matrix approximation for diffusion-convection-reaction equations with discontinuous coefficients, Math. Comput. Simul. 60 (2002) 487-506.

[71] W. Liao, J. Zhu, An accurate and efficient numerical method for solving Black-Scholes equation in option pricing, Int. J. Math. Oper. Res. 1 (2009) 191-210.

[72] K. Maekawa., S. Lee, T. Morimoto, K. Kawai, Jump diffusion model with application to the Japanese stock market, Math. Comput. Simul. 78 (2008) 223-236.

[73] O. Marom, E. Momoniat, A comparison of numerical solutions of fractional diffusion models in finance, Nonlinear Anal. Real World Appl. 10 (2009) 3435-3442.

[74] http://www.mathworks.com/

[75] R. Merton, Theory of rational option pricing, Bell J. Econ. Manag. Sci. 4 (1973) 141-183.

[76] Y. Saad, M. Schultz, GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Stat. Comput. 7 (1986) 856-869.

[77] Y. Saad, H. van der Vorst, Iterative solution of linear systems in the 20th century, J. Comput. Appl. Math. 123 (2000) 1-33.

[78] H. Sun, N. Kang, J. Zhang, 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 (2003) 651-661.

[79] H. 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) 631-644.

[80] Y. Xiao, P. Zhang, S. Shu, Algebraic multigrid methods for elastic structures with highly discontinuous coefficients, Math. Comput. Simul. 76 (2007) 249-262.

[81] CHIN, R.C.Y., T.A. MANTEUFFEL, J. DE PILLIS (1984): ADI as a preconditioning for solving the convection-diffusion equation, *SIAM J. Sci. Stat. Comput.* 5, 281-299.

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

[83] S. Ikonen and J. Toivanen, *Operator splitting methods for American option pricing*, Applied Mathematics Letters, **17** (2004), 809–814.

[84] K.S. Lee, Y.E. Gwong, and J.H. Shin, *Deravatives modeling I: Using MATLAB*®, A-Jin, Seoul, 2008.

[85] Black F, Scholes M. The pricing of options and corporate liabilities. Journal of Political Economy 1973;81; 637-659.

[86] Duffy DJ. Finite difference methods in financial engineering. John Wiley & Sons: New York; 2006.

[87] Haug EG. The complete guide to option pricing formulas. MaGraw-Hill: New York; 1997.

[88] Kangro R, Nicolaides R. Far field boundary conditions for Black-Scholes equations. SIAM Journal of Numerical Analysis 2000;38; 1357-1368.

[89] Pironneau O, Hecht F. Mesh adaption for the Black & Scholes equations. East-West Journal of Numerical Mathematics 2000;8; 25-35.

[90] Pressacco F, Gaudenzi M, Zanette A, Ziani L. New insights on testing the efficiency of methods of pricing and hedging American options. European Journal of Operational Research 2008;185; 235-254.

[91] Persson J, von Sydow L. Pricing European multi-asset options using a space-time adaptive FD-method. Computing and Visualization in Science 2007;10; 173-183.

[92] Seydel R. Tools for Computational Finance. Springer-Verlag: Berlin; 2002.

[93] Topper J. Financial engineering with finite elements. John Wiley & Sons: New York; 2005.

[94] Tavella D, Randall C. Pricing financial instruments: the finite difference method. John Wiley & Sons: New York; 2000.

[95] Wilmott P, Dewynne J, Howison S. Option Pricing: Mathematical Models and Computation. Oxford Financial Press: Oxford; 1993.

[96] Windcliff R, Forsyth PA, Vetzal RA. Analysis of the stability of the linear boundary condition for the Black-Scholes equation. Journal of Computational Finance 2004;8; 65-92.

[97] Arditi, R., Ginzburg, L.R., 1989. Coupling in predator-prey dynamics: ratio-dependence. J. Theor. Biol. 139(3), 311-326.

[98] Calude, C.S., Păun, G., 2004. Bio-steps beyond Turing. BioSystems 77, 175-194.

[99] Cavani, M., Farkas, M., 1994a. Bifurcations in a predator-prey model with memory and diffusion. I: Andronov-Hopf bifurcation. Acta Math. Hungar. 63(3), 213-229.

[100] Cavani, M., Farkas, M., 1994b. Bifurcations in a predator-prey model with memory and diffusion II: Turing bifurcation. Acta Math. Hungar. 63(3), 375-393.

[101] Cavani, M., Lizana, M., Smith, H.L., 2000. Stable Periodic Orbits for a Predator-Prey Model with Delay. J. Math. Anal. Appl. 249(2), 324-339.

[102] Holling, C.S., 1959. Some characteristics of simple types of predation and parasitism. Canadian Entomologist 91: 385-398.

[103] Kuang, Y., Beretta, E., 1998. Global qualitative analysis of a ratio-dependent predator-prey system. J. Math. Biol. 36(4), 389-406.

[104] Lizana, M., Marín V, J.J., 2005. Pattern formation in a reaction diffusion ratio-dependent predator-prey model. Notas de Mathemática 239, 1 - 16.

[105] Medvinsky, A.B., Petrovskii, S.V., Tikhonova, I.A., Malchow, H., Li, B.L., 2002. Spatiotemporal complexity of plankton and fish dynamics. SIAM Rev. 44, 311-370.

[106] Murray, J.D., 2003. Mathematical Biology II: Spatial Models and Biomedical Applications. Springer, Berlin.

[107] Nakao, H., Mikhailov, A.S., 2010. Turing patterns in network-organized activator-inhibitor systems. Nature Physics 6, 544-550.

[108] Ruuth, S.J., 1995. Implicit-explicit methods for reaction-diffusion problems in pattern formation. J. Math. Biol. 34(2), 148-176.

[109] Shiferaw, Y., Karma, A., 2006. Turing instability mediated by voltage and calcium diffusion in paced cardiac cells. National Acad Sciences 103(15), 5670-5675.

[110] Smoller, J., 1991. Shock waves and reaction-diffusion equations. Springer-Verlag, Berlin.

[111] Stephens, P.A., Sutherland, W.J., Freckleton, R.P., 1999. What is the Allee effect. Oikos 87(1), 185-190.

[112] Tapaswi, P.K., Chattopadhyay, J., 1993. Turing structure during embryogenesis. BioSystems 29(1), 25-36.

[113] Wang, W., Liu, Q., Jin, Z., 2007. Spatiotemporal complexity of a ratio-dependent predator-prey system. Phys. Rev. E 75, 051913.

[114] Wines, J.A., Addicott, J.F., Case, T.J., Diamond, J., 1986. Overview: The importance of spatial and temporal scale in ecological investigations. Community Ecology. Harper and Row, New York.

[115] W. Hackbusch. Iterative Solution of Large Linear Systems of Equations, Springer, New York (1994).

[116] W.L.Briggs *A multigird tutorial,* SIAM, Philadelphia, PA, 1987.

[117] Keller, E., Segel, L. *Model for chemotaxis,* J. Theor. Biol., **30** (1971), 225–234.

[118] Yang,Y.,Chen,H., Liu,W. *On existence og global solutions and blow-up to a system of reaction diffusion equations modelling chemotaxis,* SIAM J.Math.Anal., **33(4)** (2001), 763–785.

[119] Kang,K.,Stevens,A.,Vela'zquez,J.J.L., *Qualitative behavior of a Keller-Segel model with Non-Diffusive memory,* J. Comm. Part. Diff. Eqs., **35** (2010), 245–274.

[120] Levine,H.,Sleeman,B. *A system of reaction diffusion equations arising in the theory of reinforced random walks,* SIAM J. Appl. Math, **57** (1997), 683–730.

[121] Othmer, H.G., Stevens, A. *Aggregation, blowup, and collapse: the ABC's of taxis in reinforced random walks,* SIAM J. Appl. Math, **57** (1997), 1044–1081.

[122] U. Trottenberg, C. Oosterlee, A. Schüller *MULTIGRID,* Academic press, 2001

[123] A. Stuart and A.R. Humphries, *Dynamical system and numerical analysis*, Cambridge University Press, Cambridge, 1998.

[124] K. Yamada, M. Nonomura, and T. Ohta, Kinetics of Morphological Transitions in Microphase-Separated Diblock Copolymers, Macromolecules, 2004, 37 (15), pp. 5762-5777.