Thesis for the Degree of

Doctor of Philosophy

Phase-field simulations of crystal growth with adaptive mesh refinement

by

Li Yibao

Department of Mathematics

Graduate School

Korea University

December, 2012

김 준 석 敎 授 指 導
碩 士 學 位 論 文

Phase-field simulations of crystal growth with adaptive mesh refinement

이 論文을 理學博士學位 論文으로 提出함.

2012년 12 月

高麗大學校 大學院
數學科
李 義 寶

李義寶의 理學博士學位 論文
審査를 完了함.

2012년 12 月 日

委員長 김준석 (印)

委員 안인경 (印)

委員 황운재 (印)

委員 박춘재 (印)

委員 권기운 (印)

Contents

Abstract

A great challenge in the simulation of crystal growth with various supercoolings is the large difference in time and length scales. The use of mesh adaptivity, which is based on the choice of a suitable time integration method, is a natural choice to overcome this problem. However adaptive technology also suffers the time step restriction and crystal growth simulation with various supercoolings is still very difficult. Therefore we need a scheme that allows the use of a sufficiently large time step without the technical limitations. In this dissertation, we will review our research on overcoming the stability restriction by introducing a fast, robust, and accurate operator splitting method. Then we extend this work by incorporating adaptive mesh refinement.

After giving a brief introduction to the model of solidification in physics, we will describe the crystal growth modeling. Later, we will introduce the fast, robust, and accurate operator splitting method for phase-field simulations of crystal growth. And then the description of adaptive mesh refinement method will be drawn. Finally we will demonstrate stability, robustness, and accuracy of the proposed method by a set of representative numerical experiments.

Acknowledgments

I am very grateful for my advisor, Prof. Junseok Kim, for his guidance, encouragement, and assistance to my professional development. My appreciation also goes to Prof. Woonjae Hwang, Prof. Inkyung Ahn, Prof. Chunjae Park, and Prof. Kiwoon Kwon for serving on my committee. I would also like to thank my colleagues and friends for their help and friendship in our research group. In particular, I thank Prof. Junseok Kim for providing a productive environment for research. Under his guidance, I have published ten SCI papers with the members of our research group, during the past four years. In addition, I would like to thank my parents and grandparents for their support and love. Finally, I want to thank my wife, Binhu Xia, for her love and understanding.

Chapter 1

Introduction

1.1. Motivation and objectives

Crystal growth is a classical example of phase transformations from the liquid phase to the solid phase via heat transfer. In the past, to understand and simulate crystal growth, several methods have been developed including boundary integral [33, 40, 47, 51], cellular automaton [32, 61, 63, 64], front-tracking [3, 18, 21, 55, 62], level-set [9, 17, 26, 56], Monte-Carlo [42, 49], and phase-field [8, 10, 11, 13, 19, 22, 23, 24, 27, 37, 38, 39, 43, 44, 45, 46, 48, 54, 57, 58, 60] methods. Among these various methods, the phase-field method is popular and widely used. Its advantage is that the explicit tracking of the interface is unnecessary by introducing an order parameter, i.e., a phase-field variable. In this chapter, we focus the phase-field method for crystal growth problems which avoids difficulties associated with tracking the interface and computes complex crystal shapes.

We consider the solidification of a pure substance from its supercooled melt in both two- and three-dimensional spaces. A great challenge in the simulation with various supercoolings is the large difference in time and length scales. In order to overcome

this, many numerical methods have been proposed such as explicit [20, 21, 23, 45, 58], mixed implicit-explicit [44, 57, 60], and adaptive methods [10, 11, 43, 46, 48]. In the case of explicit methods, which are widely used, the solutions become unstable for large time steps. For this reason, in [21, 58], the authors suggested $\Delta t < h^2/(4D)$ for stability of explicit methods. Here, Δt is the time step, h is the mesh size, and D is the thermal diffusivity. In [21], the time step is also restricted to $\Delta t \leq h/(10|V_{\max}|)$, where $|V_{\max}|$ is the magnitude of the maximum value of the interface velocity. And, in [58], the authors showed that $\Delta t = h^2/(5D_L)$ works well through numerical experiments. Here, $D_L = M_\phi \epsilon^2$, M_ϕ is the kinetic mobility, and ϵ is the interface energy anisotropy. Implicit methods allow relatively larger time steps, however they are more expensive per step than explicit ones. Another classical method [54] is a multiple time-step algorithm that uses a larger time step for the flow-field calculations while reserving a fine time step for the phase-field evolution. The use of mesh adaptivity, which is based on the choice of a suitable time integration method, is a natural choice to overcome this problem. However adaptive technology also suffers the time step restriction and the crystal growth simulation with various supercoolings is still very difficult. Therefore we need a scheme that allows the use of a sufficiently large time step without technical limitations.

This dissertation consists of published papers

1. Phase-field simulations of crystal growth with adaptive mesh refinement, Yibao Li and Junseok Kim, International Journal of Heat and Mass Transfer, 55 (2012) 7926–7932.
2. A fast, robust, and accurate operator splitting method for phase-field simulations of crystal growth, Yibao Li, Hyun Geun Lee, and Junseok Kim, Journal of Crystal Growth, 321 (2011) 176–182.
3. A robust and accurate phase-field simulation of snow crystal growth, Yibao Li, Dongsun Lee, Hyun Geun Lee, Darae Jeong, Chaeyoung Lee, Donggyu Yang, and Junseok Kim, Journal of the Korean Society for Industrial and Applied Mathematics, 16 (2012) 15–29.

1.2. Outline of thesis

In Chapter 2, we give a brief introduction to the model of solidification in physics. Here we review the modeling the solidification of a pure material.

In Chapter 3, we give a discussion on implementing the interfacial and anisotropic interfacial energy. And then we derive the phase–field modeling of crystal growth.

In Chapter 4, we present an operator splitting method for phase field method of crystal growth was introduced in our previous study [37, 38, 39]. We split the governing phase–field equation into three parts. The first equation is calculated by using an explicit Euler's method. The second one is a heat equation with source term and is solved by a fast solver such as a multigrid method. The third one is a nonlinear equation and is evaluated using a closed form solution.

In Chapter 5, we give a brief description of adaptive mesh refinement method including: hierarchical structured Cartesian grids, creation of the grid hierarchy, boundary interpolation, and algorithm for mesh plots.

In Chapter 6, adaptive mesh refinement multigrid algorithm is introduced.

In Chapter 7, various numerical methods are presented to demonstrate the accuracy and robustness of the proposed operator splitting method.

Finally, conclusions are drawn in Chapter 8.

Chapter 2

The model of solidification in physical

2.1. Modeling the solidification of a pure material

The model for the solidification of a pure liquid is formulated as a moving boundary problem [30]. It is known as a Stefan problem, which typically involve the evolution of smooth boundaries or interfaces between different phases of a pure substance.

The situation is that the liquid changes to solid due to the generated latent heat. Meanwhile this latent carries away from the interface due to the changing of liquid. The rate of solidification is limited by the diffusion of latent heat away from the solid-liquid interface. The heat conduction equation which is valid in bulk solid and liquid phases is presented:

$$\frac{\partial U}{\partial t} = D\Delta U. \tag{2.1}$$

The term $U = (T - T_m)/(L/C_p)$ denotes the dimensionless temperature and T, T_m, L, and C_p represent temperature, melting point of planer interface, latent heat of fusion, and specific heat at constant pressure, respectively. The term D is thermal diffusivity which is assumed to be different in the solid and liquid phases. There are also two

boundary conditions at the solid-liquid interface. The first condition is the Stefan condition whose motion is expressed as energy conservation at the interface under phase transformation:

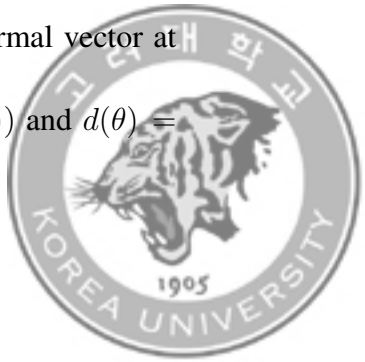
$$V_n = (D\mathbf{n} \cdot \nabla U)_{Solid} - (D\mathbf{n} \cdot \nabla U)_{Liquid}. \tag{2.2}$$

Here V_n is the velocity of the interface normal to the phase boundary and $\mathbf{n}$ is the unit normal vector to the interface. The subscripts $Solid$ and $Liquid$ stand for solid and liquid phase, respectively. Equation (2.2) can be described as that the interface velocity is proportional to the discontinuity in the heat flux across the interface.

The second boundary condition is Gibbs–Thomson condition which defines the equilibrium temperature of the interface of solid and liquid phases:

$$U_{interface} = -d(\theta)\kappa - \beta(\theta)V_n \tag{2.3}$$

Equation (2.3) describes that the interface temperature, $U_{interface}$ is shifted as a function of the local curvature κ and interface kinetics. $d(\theta) = \gamma(\theta)T_m C_p/L^2$ is the anisotropic capillary length, which is proportional to the surface tension $\gamma(\theta)$. $\beta(\theta)$ is the anisotropic kinetic coefficient. θ is the angle between the local normal vector at the interface. Traditionally, the expressions $\beta(\theta) = \beta_0(1 + \beta_k \cos(k\theta))$ and $d(\theta) =$



$\beta + \beta_{\theta\theta} = \beta_0 \left(1 - (k^2 - 1)\beta_k \cos(k\theta)\right)$ are used for a system with k-fold symmetry, where β_k is a measure of the anisotropy strength.

Chapter 3

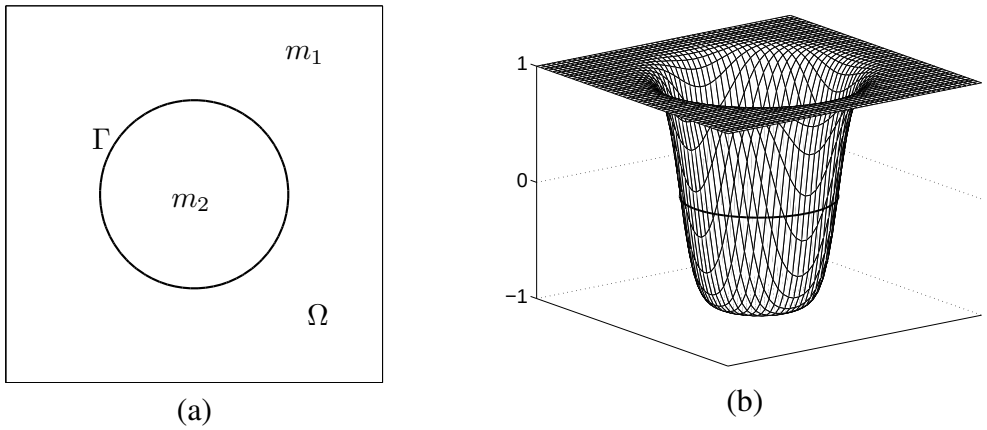
Crystal growth modeling

In the past, to understand and simulate crystal growth, several methods have been developed including boundary integral, cellular automaton, front-tracking, level-set, Monte-Carlo, and phase-field methods. Among these various methods, the phase-field method is popular and widely used. Its advantage is that the explicit tracking of the interface is unnecessary by introducing an order parameter, i.e., a phase-field variable.

3.1. Phase–field model

The phase-field model is a most popular technique for simulating dendritic growth and solving image analysis. It avoids front tracking by introducing an auxiliary order parameter, or phase-field $\phi(\mathbf{x}, t)$ that couples to the evolution of the thermal field. The phase-field interpolates between the two mixtures $\phi = (m_1 - m_2)/(m_1 + m_2)$, where m_1 and m_2 are the masses of two mixtures (see Fig. 3.1(a)). We note that the quantity $\phi(\mathbf{x}, t) \in [-1, 1]$. The variable ϕ known as the order parameter represents the local state of the entire system (see Fig. 3.1(b)). For example, $\phi = 1$ in the one phase

whose mass is m_1 and $\phi = -1$ in the other phase. The interface between two phases is defined by $\Gamma = \{\mathbf{x} \in \Omega | \phi(\mathbf{x}, t) = 0\}$.



(a) (b)

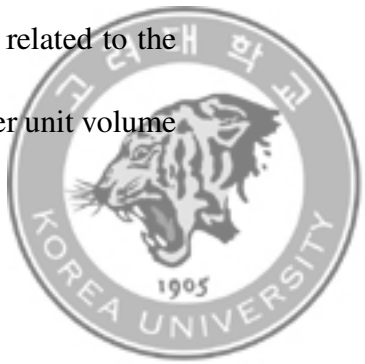
FIGURE 3.1. (a) Schematic illustration of two mixtures. (b) Schematic illustration of the definition of order parameter.

Because the interface is implicitly tracked, complicated topology changes are handled easily. Furthermore, the extension of the phase–field model to higher dimensions is straightforward. The phase–field varies smoothly from two phases within the diffuse interface, thus the phase is treated as diffuse rather than the sharp interface used in the usual sharp interface method. An illustration of sharp interface and diffuse interface is given in Fig. 3.2.

The Helmholtz free energy functional is defined as

$$\mathcal{E}(\phi) = \int_{\Omega} \left(F(\phi) + \frac{\varepsilon^2}{2} |\nabla \phi|^2 \right) d\mathbf{x}, \tag{3.1}$$

where $\Omega \subset \mathbf{R}^d$ ($d = 1, 2, 3$) and ε is the gradient energy coefficient related to the interfacial energy. $F(\phi) = 0.25(\phi^2 - 1)^2$ is the Helmholtz free energy per unit volume.



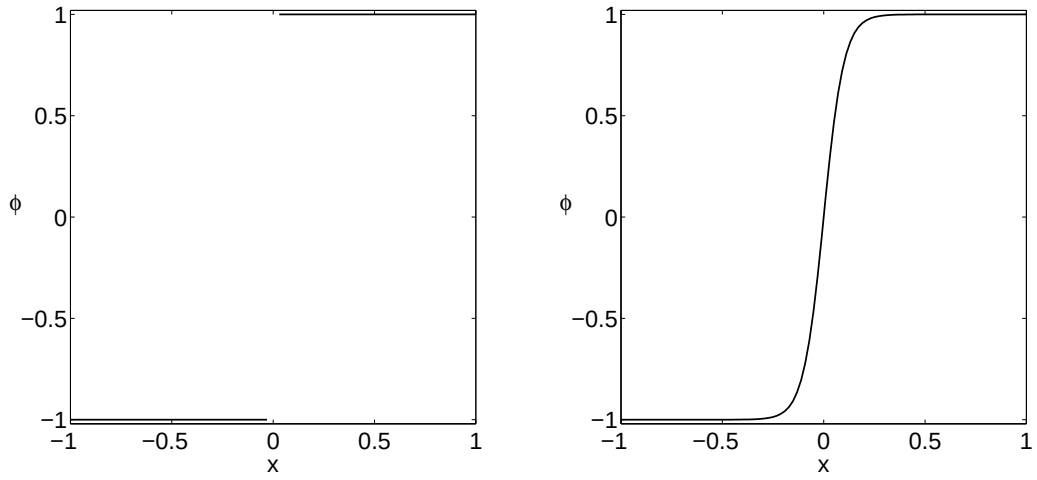


FIGURE 3.2. (a) Sharp interface and (b) Diffuse interface.

of homogeneous system of composition ϕ (see Fig. 3.3) and $\frac{\varepsilon^2}{2}|\nabla\phi|^2$ is a gradient energy. Various numerical methods are intensively studied with Neumann, periodic,

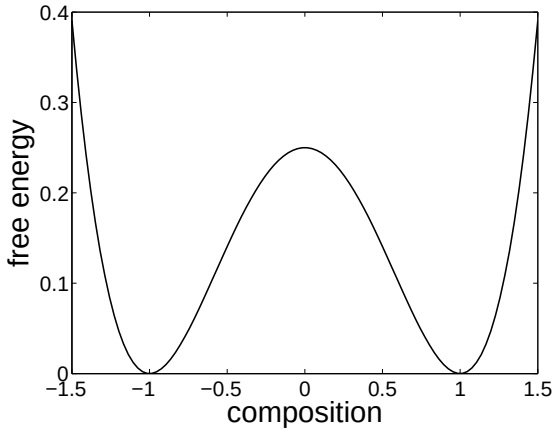


FIGURE 3.3. Helmholtz free energy density $F(\phi) = 0.25(\phi^2 - 1)^2$.

contact angle, or Dirichlet boundary conditions. Here we only describe this problem

with zero Neumann boundary condition,

$$\frac{\partial \phi}{\partial \mathbf{n}}(\mathbf{x}, t) = 0, \ \mathbf{x} \in \partial\Omega. \tag{3.2}$$

where $\frac{\partial}{\partial \mathbf{n}}$ denotes the normal derivative on $\partial\Omega$.

3.1.1. Allen–Cahn equation. The Allen–Cahn equation (AC) [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, image analysis, the motion by mean curvature flows, and crystal growth. The AC equation is the L^2-gradient flow of the total free energy $\mathcal{E}(\phi)$. Now, we review a derivation of the AC equation as a gradient flow [12, 16].

It is natural to seek a law of evolution in the form

$$\frac{\partial \phi}{\partial t} = -M \frac{\delta \mathcal{E}}{\delta \phi}. \tag{3.3}$$

The symbol "δ" here denotes the gradient on the manifold in $L^2(\Omega)$ space and the coefficient, M, is a constant mobility. Let the domain of definition for the functional $\mathcal{E}$ be $\mathcal{D} = \{\phi \in H^2(\Omega) | \frac{\partial \phi}{\partial \mathbf{n}} = 0 \text{ on } \partial\Omega\}$. Let $\phi,\ \psi \in \mathcal{D}$. Then, we have

$$\begin{aligned} \frac{d}{d\theta}\mathcal{E}(\phi + \theta\psi)\big|_{\theta=0} &= \lim_{\theta \to 0} \frac{1}{\theta}\big(\mathcal{E}(\phi + \theta\psi) - \mathcal{E}(\phi)\big) \\ &= \int_\Omega \big(F'(\phi) - \varepsilon^2 \Delta\phi\big)\psi d\mathbf{x} + \int_{\partial\Omega} \varepsilon^2 \frac{\partial \phi}{\partial \mathbf{n}} \psi \, ds \\ &= \int_\Omega \big(F'(\phi) - \varepsilon^2 \Delta\phi\big)\, \psi \, d\mathbf{x}. \end{aligned}$$

where we have used an integration by parts and the boundary condition (3.2). We identify

$$\frac{\delta \mathcal{E}}{\delta \phi} \equiv F'(\phi) - \varepsilon^2 \Delta\phi. \tag{3.4}$$

Then Eq. (3.3) becomes the AC equation [16].

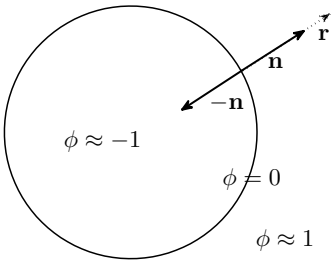
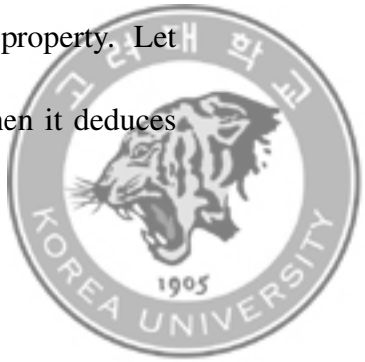


FIGURE 3.4. Illustration of a small section of a interface showing the order parameter, ϕ, and unit normal, $\mathbf{n}$.

We differentiate the energy $\mathcal{E}(\phi)$ to get

$$\begin{aligned}\frac{d}{dt}\mathcal{E}(\phi) &= \int_\Omega (F'(\phi)\phi_t + \varepsilon^2 \nabla\phi \cdot \nabla\phi_t) d\mathbf{x} \\ &= \int_\Omega (F'(\phi) - \varepsilon^2\Delta\phi)\phi_t d\mathbf{x} \\ &= -M\int_\Omega (\phi_t)^2 d\mathbf{x} \le 0, \end{aligned} \tag{3.5}$$

where we have used an integration by parts and the boundary condition (3.2). Therefore, the total energy is non-increasing in time; that is, the total energy is a Lyapunov functional for solutions of the AC equation. The AC equation and its various modified forms are widely applied to solving the problem of image analysis [4, 14, 34, 35, 36] due to its property of motion by mean curvature. We will review the property. Let us define $\mathbf{n} = \nabla\phi/|\nabla\phi|$ is the unit vector normal to the surfaces. Then it deduces



$\mathbf{n} \cdot \mathbf{n} = 1$ and $\mathbf{n} \cdot \mathbf{n}_r = 0$, where $\mathbf{n}_r$ is the rate of change of $\mathbf{n}$ in the direction of r coordinate (see Fig. 3.4).

Now the term $\Delta\phi$ can be rewritten as following:

$$\begin{aligned}
\Delta\phi &= \nabla \cdot \nabla\phi = \nabla \cdot (|\nabla\phi|\mathbf{n}) \\
&= \nabla \cdot ((\nabla\phi \cdot \mathbf{n})\mathbf{n}) = \nabla \cdot (-\phi_r \mathbf{n}) \\
&= -\nabla\phi_r \cdot \mathbf{n} - \phi_r \nabla \cdot \mathbf{n} = -(\nabla\phi)_r \cdot \mathbf{n} - \phi_r \nabla \cdot \mathbf{n} \\
&= (\phi_r \mathbf{n})_r \cdot \mathbf{n} - \phi_r \nabla \cdot \mathbf{n} \\
&= (\phi_{rr}\mathbf{n} + \phi_r \mathbf{n}_r) \cdot \mathbf{n} - \phi_r \nabla \cdot \mathbf{n} = \phi_{rr} + (\kappa_1 + \kappa_2)\phi_r.
\end{aligned}$$

Since the divergence of unit normal vector to a surface is equal to the negative of the mean curvature $(\kappa_1 + \kappa_2)$, we can have it for the kinetic equation.

$$\phi_t = -F'(\phi) + \varepsilon^2 \phi_{rr} + \varepsilon^2(\kappa_1 + \kappa_2)\phi_r \ , \tag{3.6}$$

where κ_1 and κ_2 are the principal curvatures of the surface. And for the planar interface at equilibrium, the following holds

$$-F'(\phi) + \varepsilon^2 \phi_{rr} \approx 0. \tag{3.7}$$

Therefore, Eq. (3.6) can be rewritten as

$$\phi_t = \varepsilon^2(\kappa_1 + \kappa_2)\phi_r. \tag{3.8}$$

At $\Gamma_t = \{(x, y, z)|\phi(x, y, z, t) = 0\}$, the velocity of a constant ϕ surface in the interface region is given by

$$0 = \left.\frac{d(\phi(r,t))}{dt}\right|_{\Gamma_t} = \phi_r r_t + \phi_t \Rightarrow r_t = -\phi_t/\phi_r = -\varepsilon^2(\kappa_1 + \kappa_2) \; .$$

Therefore all surfaces of constant at a point in the interface will move with the same velocity V, given by these above equations as

$$V = -\varepsilon^2(\kappa_1 + \kappa_2) = -\varepsilon^2 \left(\frac{1}{R_1} + \frac{1}{R_2}\right), \tag{3.9}$$

where R_1, R_2 are the principal radii of curvatures at the point of the surface [1]. Furthermore the AC type dynamics does not conserve the volume fractions, since the AC equation satisfies

$$\begin{aligned} \frac{d}{dt}\int_\Omega \phi d\mathbf{x} &= \int_\Omega \phi_t d\mathbf{x} = \int_\Omega M\left(-F'(\phi) + \varepsilon^2\Delta\phi\right) d\mathbf{x} \\ &= \int_\Omega -MF'(\phi)d\mathbf{x} + \int_{\partial\Omega} M\varepsilon^2 \mathbf{n}\cdot\nabla\phi ds \\ &= \int_\Omega -MF'(\phi)d\mathbf{x} \not\equiv 0. \end{aligned}$$

3.1.2. Phase-field model with anisotropic interfacial energy. To describe anisotropic interfacial energy, the gradient energy coefficient, ϵ to the depend on the angle of the normal to the order parameter ϕ

$$\epsilon(\theta) = \epsilon_0(1 + \epsilon_k \cos(k\theta)), \tag{3.10}$$

where ϵ_0 and ϵ_k are positive constants. The angle between normal vector and x-axis as ϕ that satisfies

$$\tan(\theta) = \phi_y/\phi_x. \tag{3.11}$$

Then we want to minimize the free energy functional

$$\begin{aligned} \mathcal{E}(\phi) &= \int_\Omega \left(F(\phi) + \frac{\epsilon(\theta)^2}{2} |\nabla\phi|^2 \right) d\mathbf{x}, \\ &= \int_\Omega E(\phi, \nabla\phi, \theta) d\mathbf{x}. \end{aligned} \tag{3.12}$$

Then

$$\begin{aligned} \delta E(\phi, \nabla\phi, \theta) &= \frac{\partial E}{\partial \phi}\delta\phi + \frac{\partial E}{\partial \nabla\phi}\delta\nabla\phi + \frac{\partial E}{\partial \theta}\delta\theta \\ &= \frac{dF}{d\phi}\delta\phi + \epsilon^2 \nabla\phi \cdot \delta\nabla\phi + |\nabla\phi|^2 \epsilon \frac{d\epsilon}{d\theta}\delta\theta. \end{aligned} \tag{3.13}$$

Thus

$$\begin{aligned} \delta\mathcal{E}(\phi) &= \int_\Omega \delta E(\phi, \nabla\phi, \theta) d\mathbf{x} \\ &= \int_\Omega \left(\frac{dF}{d\phi}\delta\phi + \epsilon^2 \nabla\phi \cdot \delta\nabla\phi + |\nabla\phi|^2 \epsilon \frac{d\epsilon}{d\theta}\delta\theta \right) d\mathbf{x} \\ &= \int_\Omega \frac{dF}{d\phi}\delta\phi d\mathbf{x} + \int_\Omega \epsilon^2 \nabla\phi \cdot \delta\nabla\phi d\mathbf{x} + \int_\Omega |\nabla\phi|^2 \epsilon \frac{d\epsilon}{d\theta}\delta\theta d\mathbf{x}. \end{aligned}$$

The second integral can be rewritten as

$$\begin{aligned}\int_\Omega \epsilon^2 \nabla\phi \cdot \delta\nabla\phi d\mathbf{x} &= -\int_\Omega \delta\phi \nabla\cdot(\epsilon^2\nabla\phi)d\mathbf{x} + \int_{\partial\Omega} \frac{\partial(\epsilon^2\nabla\phi)}{\partial n}\delta\phi ds \\ &= -\int_\Omega \delta\phi\nabla\cdot(\epsilon^2\nabla\phi)d\mathbf{x}, \end{aligned} \tag{3.14}$$

where we have used an integration by parts and the boundary condition (Eq. (3.2)).

The third integral can be rewritten as

$$\begin{aligned}&\int_\Omega |\nabla\phi|^2 \epsilon\frac{d\epsilon}{d\theta}\delta\theta d\mathbf{x} \\ =\ &\int_\Omega |\nabla\phi|^2 \epsilon\frac{d\epsilon}{d\theta}\delta\theta(\phi_x,\phi_y)d\mathbf{x} \\ =\ &\int_\Omega |\nabla\phi|^2 \epsilon\frac{d\epsilon}{d\theta}\frac{1}{1+\left(\frac{\phi_y}{\phi_x}\right)^2}\left[\frac{\delta\phi_y\phi_x-\phi_y\delta\phi_x}{(\phi_x)^2}\right]d\mathbf{x} \\ =\ &\int_\Omega |\nabla\phi|^2 \epsilon\frac{d\epsilon}{d\theta}\left[\frac{\delta\phi_y\phi_x-\phi_y\delta\phi_x}{(\phi_y)^2+(\phi_x)^2}\right]d\mathbf{x} \\ =\ &\int_\Omega \epsilon\frac{d\epsilon}{d\theta}[\delta\phi_y\phi_x-\phi_y\delta\phi_x]d\mathbf{x} \\ =\ &\int_\Omega \epsilon\frac{d\epsilon}{d\theta}\delta\phi_y\phi_x d\mathbf{x} - \int_\Omega \epsilon\frac{d\epsilon}{d\theta}\phi_y\delta\phi_x d\mathbf{x} \\ =\ &-\int_\Omega \delta\phi\left(\epsilon\frac{d\epsilon}{d\theta}\phi_x\right)_y d\mathbf{x} + \int_{\partial\Omega}\delta\phi\epsilon\frac{d\epsilon}{d\theta}\phi_x ds \\ &+\int_\Omega \delta\phi\left(\epsilon\frac{d\epsilon}{d\theta}\phi_y\right)_x d\mathbf{x} - \int_{\partial\Omega}\delta\phi\epsilon\frac{d\epsilon}{d\theta}\phi_y ds \\ =\ &-\int_\Omega \delta\phi\left(\epsilon\frac{d\epsilon}{d\theta}\phi_x\right)_y d\mathbf{x} + \int_\Omega \delta\phi\left(\epsilon\frac{d\epsilon}{d\theta}\phi_y\right)_x d\mathbf{x}. \end{aligned} \tag{3.15}$$

Here we have used the definition of the angle $\theta = \arctan(\phi_y/\phi_x)$, the integration by parts and the Neumann condition. Then substituting to Eq. (3.13), we get

$$\begin{aligned}\delta E(\phi, \nabla\phi, \theta) &= \int_\Omega \frac{dF}{d\phi}\delta\phi d\mathbf{x} - \int_\Omega \delta\phi \nabla\cdot(\epsilon^2\nabla\phi)d\mathbf{x} \\ &- \int_\Omega \delta\phi \left(\epsilon\frac{d\epsilon}{d\theta}\phi_x\right)_y d\mathbf{x} + \int_\Omega \delta\phi \left(\epsilon\frac{d\epsilon}{d\theta}\phi_y\right)_x d\mathbf{x} \\ &= \int_\Omega \left(\frac{dF}{d\phi} - \nabla\cdot(\epsilon^2\nabla\phi) - \left(\epsilon\frac{d\epsilon}{d\theta}\phi_x\right)_y + \left(\epsilon\frac{d\epsilon}{d\theta}\phi_y\right)_x\right)\delta\phi d\mathbf{x}. \end{aligned} \tag{3.16}$$

Finally, $\delta\mathcal{E}/\delta\phi$ is identified as

$$\frac{\delta\mathcal{E}}{\delta\phi} = \frac{dF}{d\phi} - \nabla\cdot\left(\epsilon^2\nabla\phi\right) - \left(\epsilon\frac{d\epsilon}{d\theta}\phi_x\right)_y + \left(\epsilon\frac{d\epsilon}{d\theta}\phi_y\right)_x. \tag{3.17}$$

3.2. Phase–field modeling for crystal growth

The phase field molding for crystal growth is raised from anisotropic interfacial energy with a source term

$$\mathcal{E}_c(\phi) = \int_\Omega \left(F(\phi) + \frac{\epsilon(\theta)^2}{2}|\nabla\phi|^2 + \lambda U g(\phi) \right) d\mathbf{x}, \tag{3.18}$$

Here $U(\mathbf{x}, t)$ is the temperature field and $g(\phi) = \phi^5/5 - 2\phi^3/3 + \phi$ is chosen as its minima make the double well potential fixed at $\phi = \pm 1$, i.e., $dg/d\phi = (1-\phi^2)^2$. λ controls the coupling between the order parameter and the thermal field. Taking the functional derivative of Eq. (3.18), we get

$$\frac{\delta\mathcal{E}_c}{\delta\phi} = \frac{dF}{d\phi} - \nabla\cdot(\epsilon^2\nabla\phi) + \lambda U\frac{dg}{d\phi} - \left(\epsilon\frac{d\epsilon}{d\theta}\phi_x\right)_y + \left(\epsilon\frac{d\epsilon}{d\theta}\phi_y\right)_x. \tag{3.19}$$

Then the equation of motion for ϕ becomes

$$\begin{aligned} \epsilon^2(\theta)\frac{\partial\phi}{\partial t} &= -\frac{\delta\mathcal{E}_c}{\delta\phi} \\ &= \nabla\cdot(\epsilon^2(\phi)\nabla\phi) + [\phi - \lambda U(1-\phi^2)](1-\phi^2) \\ &\quad - (\epsilon'(\theta)\epsilon(\theta)\phi_y)_x + (\epsilon'(\theta)\epsilon(\theta)\phi_x)_y. \end{aligned} \tag{3.20}$$

The equation for the thermal field is a diffusion equation with a source term that depends on changes in the order parameter ϕ, which accounts for the liberation of latent heat at the interface:

$$\frac{\partial U}{\partial t} = D\Delta U + \frac{1}{2}\frac{\partial\phi}{\partial t}. \tag{3.21}$$

Hence we summarize the governing equations of k-fold symmetric crystal growth as following:

$$\begin{aligned}\epsilon^2(\theta)\frac{\partial \phi}{\partial t} &= \nabla \cdot (\epsilon^2(\phi)\nabla\phi) + [\phi - \lambda U(1-\phi^2)](1-\phi^2) \\ &\quad - (\epsilon'(\theta)\epsilon(\theta)\phi_y)_x + (\epsilon'(\theta)\epsilon(\theta)\phi_x)_y \qquad (3.22)\\ \frac{\partial U}{\partial t} &= D\Delta U + \frac{1}{2}\frac{\partial \phi}{\partial t}. \qquad (3.23)\end{aligned}$$

3.2.1. Relation with physical problem. To be able to perform quantitative simulations with the phase-field model, the equations of motion have to reduce to the free boundary problem for the solidification of a pure substance, given in section 2.1. The interface condition for the dimensionless temperature is given in Eq. (2.3) as

$$U_{interface} = -d(\theta)\kappa - \beta(\theta)V_n$$

The above equation is a simple binary alloy of Gibbs–Tomson equation.

Then we can rewrite the above equation as

$$\beta(\theta)V_n = d(\theta)\kappa + U_{interface}. \tag{3.24}$$

The normal interface speed is given

$$V_n = V \cdot \mathbf{n} = V \cdot \left(\frac{\nabla\phi}{|\nabla\phi|}\right) = \frac{\partial\phi/\partial t}{|\nabla\phi|}. \tag{3.25}$$

Here V is the velocity of phase field. The expression for the curvature, κ, is given

$$\kappa = \nabla \cdot \left(\frac{\nabla\phi}{|\nabla\phi|}\right). \tag{3.26}$$

Since phase-field profile takes the following form across interface as

$$\phi(r) = \tanh\frac{r}{\sqrt{2}\epsilon}, \tag{3.27}$$

where a local coordinate r is from outside of the solid phase to inside normally and is zero at the interface. $\varepsilon > 0$ is a transition parameter that is taken to be very small. If the phase-field across the interface takes the form given by Eq. (3.27), then the following

equation satisfies

$$F(\phi) = \frac{(\phi^2-1)^2}{4} \approx \frac{\varepsilon^2}{2}|\nabla\phi|^2, \tag{3.28}$$

which means $|\nabla\phi| = (1-\phi^2)/(\sqrt{2}\varepsilon)$. Now, using Eq. (3.28), we rearrange Eq. (3.26) by

$$\begin{aligned}
\kappa &= \nabla\left(\frac{1}{|\nabla\phi|}\right)\cdot\nabla\phi + \frac{\Delta\phi}{|\nabla\phi|} \\
&= \nabla\left(\frac{\sqrt{2}\varepsilon}{1-\phi^2}\right)\cdot\nabla\phi + \frac{\Delta\phi}{|\nabla\phi|} \\
&= \frac{2\sqrt{2}\varepsilon\phi|\nabla\phi|^2}{(1-\phi^2)^2} + \frac{\Delta\phi}{|\nabla\phi|} \\
&= \frac{\sqrt{2}\phi}{\varepsilon} + \frac{\Delta\phi}{|\nabla\phi|} \\
&= \frac{1}{|\nabla\phi|}\left(\frac{\phi(\phi^2-1)}{\varepsilon^2} - \Delta\phi\right).
\end{aligned} \tag{3.29}$$

Substituting Eqs. (3.25) and (3.29) into Eq. (3.24), we get

$$\begin{aligned}
\beta(\theta)\frac{\partial\phi}{\partial t} &= d(\theta)\left(\frac{\phi(\phi^2-1)}{\varepsilon^2} - \Delta\phi\right) + U_{interface}|\nabla\phi| \\
&= d(\theta)\left(\frac{\phi(\phi^2-1)}{\varepsilon^2} - \Delta\phi\right) + U\frac{1-\phi^2}{\sqrt{2}\varepsilon} \\
&= d(\theta)\left(\frac{\phi(\phi^2-1)}{\varepsilon^2} - \Delta\phi\right) + U\frac{1-\phi^2}{\sqrt{2}\varepsilon}.
\end{aligned} \tag{3.30}$$

Here $U(1-\phi^2)/(\sqrt{2}\varepsilon)$ represents the themo-solutal driving force for ϕ by temperature. Note that we can take $U|\nabla\phi| = U_{interface}|\nabla\phi|$, since $|\nabla\phi|$ is zero in each phase. With the anisotropic interfacial energy, we can the mentioned phase–field equation (3.22).

One difference is that the last term, $1-\phi^2$ in Eq. (3.30), is replaced by $(1-\phi^2)^2$ in Eq. (3.22). These two equations are much similar as shown in Fig. 3.5. While the latter form corresponds to a greater concentration of the driving force and helps stabilize the front in the presence of a strong temperature gradient [23, 24]. It should be noted that the physics of the simulated system is given by the capillary length and the kinetic coefficient. In the phase-field simulations, the interface width ϵ can be artificially increased without changing the physics of the system. This makes the phase-field method so powerful.

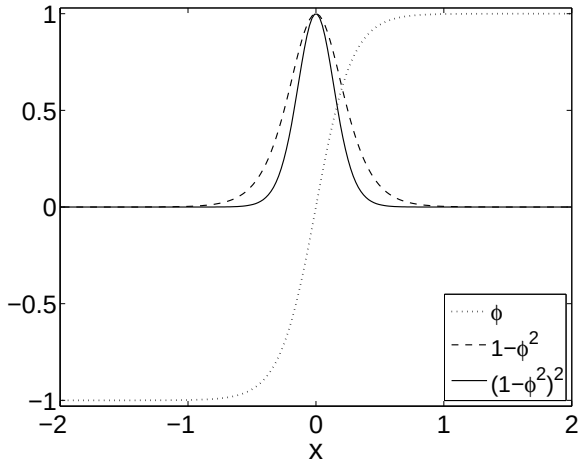
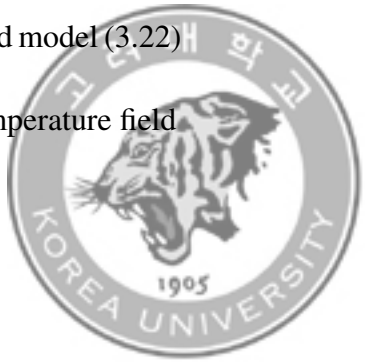


FIGURE 3.5. Plots of $1-\phi^2$ and $(1-\phi^2)^2$. Here $\phi = \tanh(x/(\sqrt{2}\varepsilon))$. Here we take $\varepsilon = 0.2$ to show a simple.

3.2.2. Non-dimensionalization. Asymptotic analysis the phase-field model (3.22) and (3.23) are expressed in dimensionless forms. The dimensionless temperature field



is given by

$$U = c_p(T - T_M)/L.$$

Here c_p is the specific heat at constant pressure, T_M is the melting temperature, and L is the latent heat of fusion. The diffusion factor is dimensioned as

$$D = \alpha\tau_0/\epsilon_0^2,$$

where α is the thermal diffusivity, τ_0 is the characteristic time, and ϵ_0 is the characteristic length. In the limit of thin interface width ϵ_0, The relation of parameters between phase–field model and the Gibbs–Tomson equation are given by

$$d_0 = a_1\frac{\epsilon_0}{\lambda}, \tag{3.31}$$

$$\beta = a_1\left(\frac{\gamma(\theta)}{\lambda\epsilon(\theta)} - a_2\frac{\epsilon(\theta)}{D}\right). \tag{3.32}$$

The constants $a_1 = I/J$ and $a_2 = (K + JF)/(2I)$, where

$$I = \int_{-\infty}^{\infty}(\partial_\eta\phi_0)^2 d\eta, \tag{3.33}$$

$$J = -\int_{-\infty}^{\infty}\partial_\eta\phi_0 g_\phi^0 d\eta, \tag{3.34}$$

$$K = \int_{-\infty}^{\infty}\partial_\eta\phi_0 g_\phi^0 d\eta\int_0^\eta h^0 d\zeta, \tag{3.35}$$

$$F = \int_0^\infty (h^0 + 1)d\eta. \tag{3.36}$$

Here both $g_\phi^0 = g_\psi(\phi^0(\eta))$ and $h^0(\eta) = \phi_0(\eta)$ are the functions of η. $\phi_0(\eta)$ is the solution of the following equation

$$(\partial_\eta \phi_0)^2 - f_\phi(\phi_0) = 0. \tag{3.37}$$

Then λ is given as $\lambda = a_1\epsilon_0/d_0$. It should be noted that in the numerical simulation, we generally fix measurable d_0, β_0 and D, and determine λ and τ_0 with known $a_1 = 0.8839$ and $a_2 = 0.6267$ [23, 24].

3.3. Four-fold crystal growth

If $k = 4$, we call the crystal as be four-fold crystal. And the phase–field modeling of crystal growth can be expressed as another form. Let us recall the definition of the gradient energy coefficient, ϵ,

$$\epsilon(\theta) = \epsilon_0(1 + \epsilon_4 \cos(4\theta)), \tag{3.38}$$

$$\theta = \arctan(\phi_y/\phi_x). \tag{3.39}$$

Then the following trigonometric functions can be represented as

$$\begin{aligned}
\sin(\theta) &= \frac{\phi_y}{|\nabla\phi|}, \\
\cos(\theta) &= \frac{\phi_x}{|\nabla\phi|}, \\
\sin(2\theta) &= 2\sin(\theta)\cos(\theta) = \frac{2\phi_x\phi_y}{|\nabla\phi|^2}, \\
\cos(2\theta) &= \cos^2(\theta) - \sin^2(\theta) = \frac{\phi_x^2 - \phi_y^2}{|\nabla\phi|^2}, \\
\sin(4\theta) &= 2\sin(2\theta)\cos(2\theta) = \frac{4(\phi_x^3\phi_y - \phi_x\phi_y^3)}{|\nabla\phi|^4}, \\
\cos(4\theta) &= \cos^2(2\theta) - \sin^2(2\theta) = \frac{4(\phi_x^4 + \phi_y^4) - 3(\phi_x^2 + \phi_y^2)^2}{|\nabla\phi|^4}.
\end{aligned}$$

Here $|\nabla\phi| = \sqrt{\phi_x^2 + \phi_y^2}$. Thus

$$\begin{aligned}
\epsilon(\theta) &= \epsilon_0(1 + \epsilon_4 \cos(4\theta)) \\
&= \epsilon_0 \left(1 + \epsilon_4 \frac{4(\phi_x^4 + \phi_y^4) - 3(\phi_x^2 + \phi_y^2)^2}{|\nabla\phi|^4}\right) \\
&= \epsilon_0(1 - 3\epsilon_4)\left(1 + \frac{4\epsilon_4}{1 - 3\epsilon_4}\frac{\phi_x^4 + \phi_y^4}{|\nabla\phi|^4}\right). \qquad (3.40)
\end{aligned}$$

Also we can get

$$\frac{d\epsilon}{d\theta} = -4\epsilon_0\epsilon_4 \sin(4\theta) = -\frac{16\epsilon_0\epsilon_4(\phi_x^3\phi_y - \phi_x\phi_y^3)}{|\nabla\phi|^4}.$$

In another way, by Eq. (3.40), we get

$$\begin{aligned}
|\nabla\phi|^2 \frac{\partial\epsilon}{\partial\phi_x} &= 4\epsilon_0\epsilon_4 |\nabla\phi|^2 \frac{\partial}{\partial\phi_x}\left(\frac{\phi_x^4 + \phi_y^4}{|\nabla\phi|^4}\right) \\
&= 4\epsilon_0\epsilon_4 |\nabla\phi|^2 \frac{\partial}{\partial\phi_x}\left(\frac{\phi_x^4 + \phi_y^4}{(\phi_x^2 + \phi_y^2)^2}\right) \\
&= 16\epsilon_0\epsilon_4 |\nabla\phi|^2 \left(\frac{\phi_x^3(\phi_x^2 + \phi_y^2)^2 - \phi_x(\phi_x^4 + \phi_y^4)(\phi_x^2 + \phi_y^2)}{(\phi_x^2 + \phi_y^2)^4}\right) \\
&= 16\epsilon_0\epsilon_4 \left(\frac{\phi_x^3\phi_y^2 - \phi_x\phi_y^4}{|\nabla\phi|^4}\right) \\
&= -\frac{d\epsilon}{d\theta}\phi_y.
\end{aligned}$$

Similarly, we can get

$$|\nabla\phi|^2 \frac{\partial\epsilon}{\partial\phi_y} = \frac{d\epsilon}{d\theta}\phi_x.$$

The basic equations of the phase-field model, i.e., Eq. (3.22) can be derived by

$$\begin{aligned}\epsilon^2(\phi)\frac{\partial\phi}{\partial t} &= \nabla\cdot(\epsilon^2(\phi)\nabla\phi)+[\phi-\lambda U(1-\phi^2)](1-\phi^2)\\ &\quad+\left(|\nabla\phi|^2\epsilon(\phi)\frac{\partial\epsilon(\phi)}{\partial\phi_x}\right)_x+\left(|\nabla\phi|^2\epsilon(\phi)\frac{\partial\epsilon(\phi)}{\partial\phi_y}\right)_y \qquad (3.41)\end{aligned}$$

Note that the solidification in three dimensions can be extended with the following form:

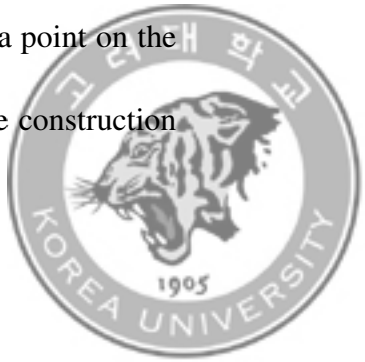
$$\begin{aligned}\epsilon^2(\phi)\frac{\partial\phi}{\partial t} &= \nabla\cdot(\epsilon^2(\phi)\nabla\phi)+[\phi-\lambda U(1-\phi^2)](1-\phi^2)\\ &\quad+\left(|\nabla\phi|^2\epsilon(\phi)\frac{\partial\epsilon(\phi)}{\partial\phi_x}\right)_x+\left(|\nabla\phi|^2\epsilon(\phi)\frac{\partial\epsilon(\phi)}{\partial\phi_y}\right)_y\\ &\quad+\left(|\nabla\phi|^2\epsilon(\phi)\frac{\partial\epsilon(\phi)}{\partial\phi_z}\right)_z. \qquad (3.42)\end{aligned}$$

The anisotropic function $\epsilon(\phi)$ in three-dimensional space is defined in three–dimensional space as:

$$\epsilon(\phi)=(1-3\epsilon_4)\left(1+\frac{4\epsilon_4}{1-3\epsilon_4}\frac{\phi_x^4+\phi_y^4+\phi_z^4}{|\nabla\phi|^4}\right).$$

3.4. The Wulff construction

The equilibrium crystal shape, which constructed by the Wulff's theorem [59], is determined by minimizing the total interfacial free energy. We describe the construction of the equilibrium shape geometrically [7]. Let $M=(\epsilon(\theta),\theta)$ be a point on the interfacial energy function in the polar coordinates (see Fig. 3.6). The construction



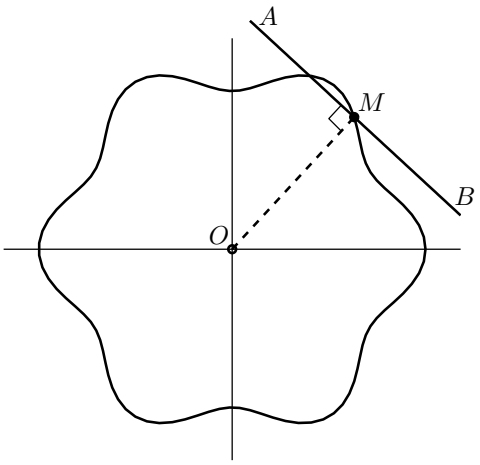


FIGURE 3.6. Interfacial free-energy density $\epsilon(\theta)$ in the polar coordinates.

starts from the origin O and draw the line segment $\overline{OM}$ to the point M. Draw the perpendicular line $\overleftrightarrow{AB}$ to the line segment $\overline{OM}$. Then the inner convex hull made from all such perpendiculars is an equilibrium crystal shape.

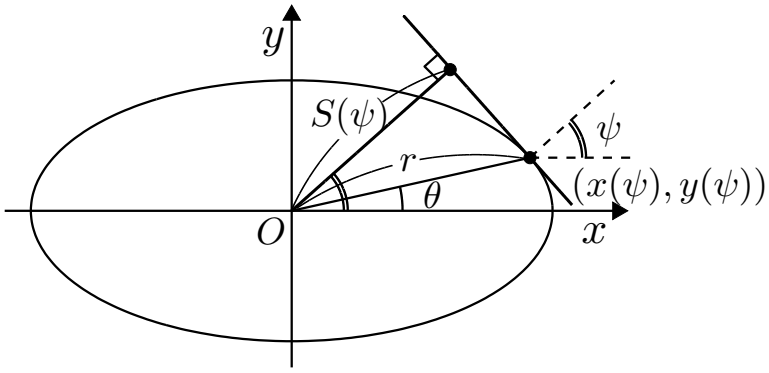


FIGURE 3.7. Parameter definitions.

Conversely, let us assume the equilibrium shape is known and (r, θ) be the polar coordinates of a point T of the crystal boundary S, that is, $T = (r, \theta)$. And let $T = (x(\psi), y(\psi))$ be the corresponding Cartesian coordinates, where ψ is a parameter and

is the angle between x-axis and the perpendicular line to the tangent line $\overleftrightarrow{AB}$ at the point T. Let M be the intersection point of the line $\overleftrightarrow{AB}$ and the perpendicular line containing the origin to $\overleftrightarrow{AB}$. Let the length of the line segment $\overline{OM}$ be $p(\psi)$. In Fig. 3.7, we can see these parameter definitions. Then $p(\psi)$ can be obtained from the right triangle $\triangle OTM$:

$$\begin{aligned} p(\psi) &= r\cos(\psi-\theta) = r\cos\psi\cos\theta + r\sin\psi\sin\theta \\ &= x(\psi)\cos\psi + y(\psi)\sin\psi. \end{aligned} \tag{3.43}$$

We can express $(x(\psi), y(\psi))$ in terms of $p(\psi)$. Taking a derivative to $p(\psi)$, we have

$$p_\psi(\psi) = x_\psi(\psi)\cos\psi - x(\psi)\sin\psi + y_\psi(\psi)\sin\psi + y(\psi)\cos\psi. \tag{3.44}$$

Here the normal vector $(\cos\psi, \sin\psi)$ and the tangent vector (x_ψ, y_ψ) are orthogonal, that is, $(\cos\psi, \sin\psi)\cdot(x_\psi, y_\psi) = 0$. Thus Eq. (3.44) can be simplify as

$$p_\psi = -x(\psi)\sin\psi + y(\psi)\cos\psi. \tag{3.45}$$

Now, by solving Eqs. (3.43) and (3.45) we have

$$x(\psi) = p(\psi)\cos\psi - p_\psi(\psi)\sin\psi,\ \ y(\psi) = p(\psi)\sin\psi + p_\psi(\psi)\cos\psi. \tag{3.46}$$

Let F and A be the total edge free energy and the area of crystal, respectively. They can be defined as

$$F = \int \epsilon(\psi)\sqrt{(x_\psi(\psi))^2 + (y_\psi(\psi))^2}d\psi, \tag{3.47}$$

$$A = \frac{1}{2}\int (x(\psi)y_\psi(\psi) - y(\psi)x_\psi(\psi))d\psi. \tag{3.48}$$

Using Eq. (3.46), we can rewrite Eqs. (3.47) and (3.48) in the form

$$F = \int \epsilon(\psi)(p(\psi) + p_{\psi\psi}(\psi))d\psi,$$

$$A = \frac{1}{2}\int p(\psi)(p(\psi) + p_{\psi\psi}(\psi))d\psi.$$

We want to minimize F with subject to a constant area constraint of A. Using the Lagrange multiplier λ, we seek to minimize

$$F + \lambda A = \int \left(\epsilon(\psi) + \frac{\lambda}{2}p(\psi)\right)(p(\psi) + p_{\psi\psi}(\psi))d\psi.$$

And then, the Euler–Lagrange equation is

$$\frac{\partial Q}{\partial p} - \frac{d}{d\psi}\left(\frac{\partial Q}{\partial p_\psi}\right) + \frac{d^2}{d\psi^2}\left(\frac{\partial Q}{\partial p_{\psi\psi}}\right) = 0, \tag{3.49}$$

where

$$Q = \left(\epsilon + \frac{\lambda}{2}p\right)(p + p_{\psi\psi}). \tag{3.50}$$

From these two Eqs. (3.49) and (3.50), we get

$$p + p_{\psi\psi} = -\frac{1}{\lambda}(\epsilon + \epsilon_{\psi\psi}). \tag{3.51}$$

A solution of differential equation (3.51) is

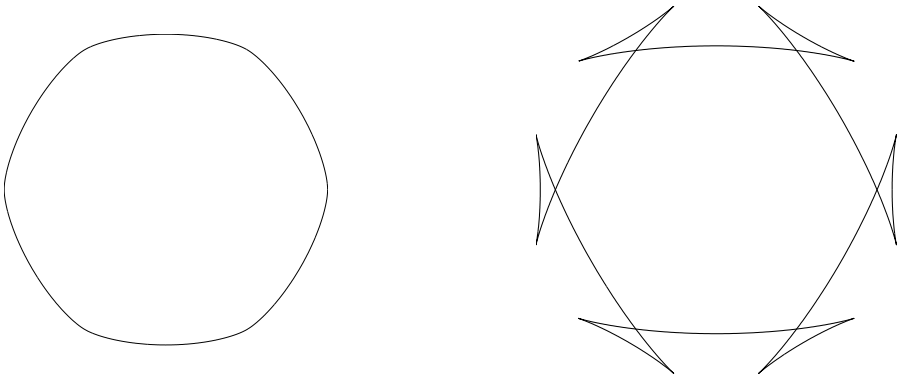
$$p(\psi) = -\frac{1}{\lambda}\epsilon(\psi).$$

This result implies that in a crystal at equilibrium, the distances of the faces from the center of the crystal are proportional to their surface free energies per unit area [7].

For large ϵ_k values, the crystal shape will be energy minimizing when certain orientations are missing. Missing orientations occur when the polar plot of $r = 1/\epsilon(\theta)$ changes convexity [15]. The curvature of a polar plot $r(\theta)$ is $\kappa = (r^2+2r_\theta^2-rr_{\theta\theta})/(r^2+r_\theta^2)^{\frac{3}{2}}$. For $r(\theta) = 1/\epsilon(\theta)$, the curvature is $\kappa = (\epsilon+\epsilon_{\theta\theta})/[1+(\epsilon_\theta/\epsilon)^2]^{\frac{3}{2}}$. So convexity changes whenever

$$\epsilon + \epsilon_{\theta\theta} = \epsilon_0(1-(k^2-1)\epsilon_k \cos k\theta) < 0.$$

If values of ϵ_k are larger than $1/(k^2-1)$, then missing orientations occur. In other words, some orientations do not appear on the equilibrium shape of a crystal. Figure 3.8 shows the 6-fold Wulff equilibrium shapes ($(x(\psi), y(\psi))$ for $0 \le \psi \le 2\pi$) with two different ϵ_6 values: (a) $\epsilon_6 = 1/50$ and (b) $\epsilon_6 = 1/10$ (which shows the missing orientation).



(a) $\epsilon_6 = 1/50$ (b) $\epsilon_6 = 1/10$

FIGURE 3.8. The 6-fold Wulff equilibrium shapes with two different ϵ_6 values.

Chapter 4

Numerical solutions

A great challenge in the simulation with various supercoolings is the large difference in time and length scales. In order to overcome this, many numerical methods have been proposed such as explicit [20, 21, 23, 45, 58], mixed implicit-explicit [44, 57, 60], and adaptive methods [10, 11, 43, 46, 48]. In the case of explicit methods, which are widely used, the solutions become unstable for large time steps. For this reason, in [21, 58], the authors suggested $\Delta t < h^2/(4D)$ for stability of explicit methods. Here, Δt is the time step, h is the mesh size, and D is the thermal diffusivity. In [21], the time step is also restricted to $\Delta t \leq h/(10|V_{\max}|)$, where $|V_{\max}|$ is the magnitude of the maximum value of the interface velocity. Also, in [58], the authors showed that $\Delta t = h^2/(5D_L)$ works well through numerical experiments, where $D_L = M_\phi \epsilon^2$, M_ϕ is the kinetic mobility. Implicit methods allow relatively larger time steps, however they are computationally more expensive per step than explicit ones. The use of mesh adaptivity, which is based on the choice of a suitable time integration method, is a natural choice to overcome this problem. However adaptive technology also suffers the time step restriction and crystal growth simulation with various supercoolings is still very

difficult. Therefore we need a scheme that allows the use of a sufficiently large time step without the technical limitations. In this chapter, we review our proposed computationally efficient, and robust operator splitting algorithms, which are introduced in [37, 38, 39], for solving the crystal growth phase-field simulation.

4.1. Time discretisation

In this section, we propose a robust hybrid numerical method for crystal growth simulation. For simplicity of exposition we shall discretize Eqs. (3.23) and (3.41) in two-dimensional space, i.e., $\Omega = (a,b) \times (c,d)$. Let N_x and N_y be positive even integers, $h = (b-a)/N_x = (d-c)/N_y$ be the uniform mesh size, and $\Omega_h = \{(x_i, y_j) : x_i = a + (i-0.5)h,\ y_j = c + (j-0.5)h,\ 1 \le i \le N_x,\ 1 \le j \le N_y\}$ be the set of cell-centers.

Let ϕ_{ij}^n be approximations of $\phi(x_i, y_j, n\Delta t)$, where $\Delta t = T/N_t$ is the time step, T is the final time, and N_t is the total number of time steps. The discrete differentiation operator is $\nabla_d \phi_{ij} = (\phi_{i+1,j} - \phi_{i-1,j}, \phi_{i,j+1} - \phi_{i,j-1})/(2h)$. We then define the discrete Laplacian by $\Delta_d \phi_{ij} = (\phi_{i+1,j} + \phi_{i-1,j} - 4\phi_{ij} + \phi_{i,j+1} + \phi_{i,j-1})/h^2$. We discretize Eqs. (3.23) and (3.41):

$$\begin{aligned}
\epsilon^2(\phi^n)\frac{\phi^{n+1}-\phi^n}{\Delta t} &= \epsilon^2(\phi^n)\Delta_d\phi^{n+1,2} + 2\epsilon(\phi^n)\nabla_d\epsilon(\phi^n)\cdot\nabla_d\phi^n \\
&\quad -F'(\phi^{n+1}) - 4\lambda U^n F(\phi^{n+1,1}) \\
&\quad + \left(|\nabla_d\phi|^2\epsilon(\phi)\frac{\partial\epsilon(\phi)}{\partial\phi_x}\right)_x^n + \left(|\nabla_d\phi|^2\epsilon(\phi)\frac{\partial\epsilon(\phi)}{\partial\phi_y}\right)_y^n, \quad (4.1) \\
\frac{U^{n+1}-U^n}{\Delta t} &= D\Delta_d U^{n+1} + \frac{\phi^{n+1}-\phi^n}{2\Delta t}, \quad (4.2)
\end{aligned}$$

where $F(\phi) = 0.25(\phi^2-1)^2$ and $F'(\phi) = \phi(\phi^2-1)$. Here $\phi^{n+1,m}$ for $m = 1, 2$ are defined in the operator splitting scheme. We propose the following operator splitting

scheme:

$$\epsilon^2(\phi^n)\frac{\phi^{n+1,1}-\phi^n}{\Delta t} = 2\epsilon(\phi^n)\nabla_d\epsilon(\phi^n)\cdot\nabla_d\phi^n + \left(|\nabla_d\phi|^2\epsilon(\phi)\frac{\partial\epsilon(\phi)}{\partial\phi_x}\right)_x^n + \left(|\nabla_d\phi|^2\epsilon(\phi)\frac{\partial\epsilon(\phi)}{\partial\phi_y}\right)_y^n, \tag{4.3}$$

$$\epsilon^2(\phi^n)\frac{\phi^{n+1,2}-\phi^{n+1,1}}{\Delta t} = -F'(\phi^{n+1,2}), \tag{4.4}$$

$$\epsilon^2(\phi^n)\frac{\phi^{n+1}-\phi^{n+1,2}}{\Delta t} = \epsilon^2(\phi^n)\Delta_d\phi^{n+1} - 4\lambda U^n F(\phi^{n+1,2}). \tag{4.5}$$

In Eq. (4.3), we can simplify the following terms

$$|\nabla_d\phi|^2\frac{\partial\epsilon(\phi)}{\partial\phi_x} = \frac{16\epsilon_4\phi_x(\phi_x^2\phi_y^2-\phi_y^4)}{|\nabla_d\phi|^4},\ |\nabla_d\phi|^2\frac{\partial\epsilon(\phi)}{\partial\phi_y} = \frac{16\epsilon_4\phi_y(\phi_x^2\phi_y^2-\phi_x^4)}{|\nabla_d\phi|^4}.$$

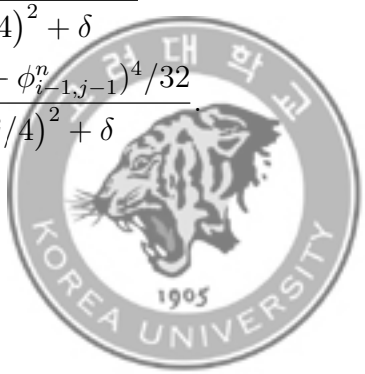
With nine local points, we describe the following term

$$\left(\frac{\phi_x\epsilon(\phi)(\phi_x^2\phi_y^2-\phi_y^4)}{|\nabla_d\phi|^4}\right)_{x,ij}^n = \frac{\left(\frac{\phi_x\epsilon(\phi)(\phi_x^2\phi_y^2-\phi_y^4)}{|\nabla_d\phi|^4}\right)_{i+\frac{1}{2},j}^n - \left(\frac{\phi_x\epsilon(\phi)(\phi_x^2\phi_y^2-\phi_y^4)}{|\nabla_d\phi|^4}\right)_{i-\frac{1}{2},j}^n}{h},$$

where

$$\begin{aligned}
&\left(\frac{\phi_x\epsilon(\phi)(\phi_x^2\phi_y^2-\phi_y^4)}{|\nabla_d\phi|^4}\right)_{i+\frac{1}{2},j}^n \\
&= \frac{(\epsilon(\phi_{i+1,j}^n)+\epsilon(\phi_{ij}^n))(\phi_{i+1,j}^n-\phi_{ij}^n)^3(\phi_{i+1,j+1}^n-\phi_{i+1,j-1}^n+\phi_{i,j+1}^n-\phi_{i,j-1}^n)^2/8}{h\left((\phi_{i+1,j}^n-\phi_{ij}^n)^2+(\phi_{i+1,j+1}^n-\phi_{i+1,j-1}^n+\phi_{i,j+1}^n-\phi_{i,j-1}^n)^2/4\right)^2+\delta} \\
&- \frac{(\epsilon(\phi_{ij}^n)+\epsilon(\phi_{i-1,j}^n))(\phi_{i+1,j}^n-\phi_{ij}^n)(\phi_{i+1,j+1}^n-\phi_{i+1,j-1}^n+\phi_{i,j+1}^n-\phi_{i,j-1}^n)^4/32}{h\left((\phi_{i+1,j}^n-\phi_{ij}^n)^2+(\phi_{i+1,j+1}^n-\phi_{i+1,j-1}^n+\phi_{i,j+1}^n-\phi_{i,j-1}^n)^2/4\right)^2+\delta},
\end{aligned}$$

$$\begin{aligned}
&\left(\frac{\phi_x\epsilon(\phi)(\phi_x^2\phi_y^2-\phi_y^4)}{|\nabla_d\phi|^4}\right)_{i-\frac{1}{2},j}^n \\
&= \frac{(\epsilon(\phi_{ij}^n)+\epsilon(\phi_{i-1,j}^n))(\phi_{ij}^n-\phi_{i-1,j}^n)^3(\phi_{i,j+1}^n-\phi_{i,j-1}^n+\phi_{i-1,j+1}^n-\phi_{i-1,j-1}^n)^2/8}{h\left((\phi_{ij}^n-\phi_{i-1,j}^n)^2+(\phi_{i,j+1}^n-\phi_{i,j-1}^n+\phi_{i-1,j+1}^n-\phi_{i-1,j-1}^n)^2/4\right)^2+\delta} \\
&- \frac{(\epsilon(\phi_{i-1,j}^n)+\epsilon(\phi_{i-2,j}^n))(\phi_{ij}^n-\phi_{i-1j}^n)(\phi_{i,j+1}^n-\phi_{i-1,j-1}^n+\phi_{i-1,j+1}^n-\phi_{i-1,j-1}^n)^4/32}{h\left((\phi_{ij}^n-\phi_{i-1,j}^n)^2+(\phi_{i,j+1}^n-\phi_{i,j-1}^n+\phi_{i-1,j+1}^n-\phi_{i-1,j-1}^n)^2/4\right)^2+\delta}.
\end{aligned}$$



The other terms can be described in a similar manner. Note that we added a small value $\delta = 1\text{e}{-10}$ in the denominator $|\nabla_d \phi|^4$ to avoid singularities.

Eq. (4.4) can be considered as an approximation of the equation

$$\phi_t = \frac{\phi - \phi^3}{\epsilon^2} \tag{4.6}$$

by an implicit Euler's method with the initial condition $\phi^{n+1,2}$. We can solve Eq. (4.6) analytically by the method of separation of variables [52]. Here we will describe it in detail. Assume there exits a continues function which satisfies

$$\frac{d\psi}{dt} = \frac{\psi - \psi^3}{\epsilon^2}, \tag{4.7}$$

$$\psi(0) = \phi^{n+1,1}, \tag{4.8}$$

$$\psi(\Delta t) = \phi^{n+1,2}. \tag{4.9}$$

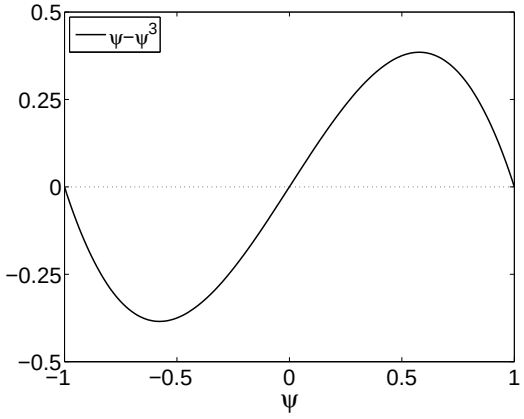


FIGURE 4.1. Plots of $\psi - \psi^3$.

Assume $\psi(0) \in (-1, 1)$ and refer to Fig. (4.1), we get

$$\frac{d\psi}{dt} \begin{cases} > 0, \text{ if } \psi(0) \in (0, 1), \\ < 0, \text{ if } \psi(0) \in (-1, 0), \\ = 0, \text{ otherwise.} \end{cases} \tag{4.10}$$

That is

$$\begin{cases} \phi^{n+1,2} > \phi^{n+1,1}, \text{ if } \phi^{n+1,1} > 0, \\ \phi^{n+1,2} < \phi^{n+1,1}, \text{ if } \phi^{n+1,1} < 0, \\ \phi^{n+1,2} = \phi^{n+1,1}, \text{ otherwise.} \end{cases} \tag{4.11}$$

Thus $\phi^{n+1,1}$ and $\phi^{n+1,2}$ are of the same sign for $\phi^{n+1,1} \in (-1, 1)$. It should be noted that if $\phi^{n+1,1} > 1$, Eq. (4.7) makes $\phi^{n+1,2}$ converge to 1. Thus $\phi^{n+1,1}$ and $\phi^{n+1,2}$ are always of the same sign for $\phi^{n+1,1} \in R$. Later we can consider $\phi^{n+1,1} \in (0, 1)$ and rewrite the Eq. (4.7) as

$$\begin{aligned} \frac{dt}{\epsilon^2} &= \frac{d\psi}{\psi(1-\psi)(1+\psi)} \\ \frac{dt}{\epsilon^2} &= \left(\frac{1}{\psi} + \frac{1}{2(1-\psi)} - \frac{1}{2(1+\psi)} \right) d\psi \\ \int \frac{dt}{\epsilon^2} &= \int \left(\frac{1}{\psi} + \frac{1}{2(1-\psi)} - \frac{1}{2(1+\psi)} \right) d\psi \\ \frac{t}{\epsilon^2} + c &= \ln(\psi) - \frac{\ln(1-\psi)}{2} - \frac{\ln(1+\psi)}{2} \\ \psi &= \frac{e^{\frac{t}{\epsilon^2}+c}}{\sqrt{1 + e^{\frac{2t}{\epsilon^2}+2c}}}. \end{aligned}$$

Here c is a constant. Applying the initial condition (Eqs. (4.8) and (4.9)), the following function is defined

$$\begin{aligned} \phi^{n+1,1} &= \psi(0) = \frac{e^c}{\sqrt{1+e^{2c}}}, \\ \phi^{n+1,2} &= \psi(\Delta t) = \frac{e^{\frac{\Delta t}{\epsilon^2}+c}}{\sqrt{1 + e^{\frac{2\Delta t}{\epsilon^2}+2c}}}. \end{aligned}$$

Then

$$\phi^{n+1,2} = \frac{\phi^{n+1,1}}{\sqrt{e^{-\frac{2\Delta t}{\epsilon^2(\phi^n)}} + (\phi^{n+1,1})^2\left(1 - e^{-\frac{2\Delta t}{\epsilon^2(\phi^n)}}\right)}}. \tag{4.12}$$

Note that if $\phi^{n+1,1} \in (-1, 0)$, we can get the same equation with the similar deviation.

Also we know

$$\lim_{\phi^{n+1,1}\to 0^+} \phi^{n+1,2} = \lim_{\phi^{n+1,1}\to 0^-} \phi^{n+1,2} = \phi^{n+1,2}\big|_{\phi^{n+1,1}=0} = 0.$$

Thus the solution is continuous and well defined for $\phi^{n+1,1} \in (-1, 1)$. With the similar method, the same solution can be extended for $\phi^{n+1,1} \in [1, +\infty)$ and $\phi^{n+1,1} \in (-\infty, -1]$. We will briefly describe them here. Finally, we summarize our proposed scheme as follows:

$$\epsilon^2(\phi^n)\frac{\phi^{n+1,1} - \phi^n}{\Delta t} = 2\epsilon(\phi^n)\epsilon(\phi^n)_x\phi^n_x + 2\epsilon(\phi^n)_y\epsilon(\phi^n)\phi^n_y \tag{4.13}$$
$$+ \left(\frac{16\epsilon(\phi^n)\epsilon_4\phi_x(\phi_x^2\phi_y^2 - \phi_y^4)}{|\nabla_d\phi|^4}\right)^n_x + \left(\frac{16\epsilon(\phi^n)\epsilon_4\phi_y(\phi_x^2\phi_y^2 - \phi_x^4)}{|\nabla_d\phi|^4}\right)^n_y,$$

$$\phi^{n+1,2} = \frac{\phi^{n+1,1}}{\sqrt{e^{-\frac{2\Delta t}{\epsilon^2(\phi^n)}} + (\phi^{n+1,1})^2\left(1 - e^{-\frac{2\Delta t}{\epsilon^2(\phi^n)}}\right)}}, \tag{4.14}$$

$$\epsilon^2(\phi^n)\frac{\phi^{n+1} - \phi^{n+1,2}}{\Delta t} = \epsilon^2(\phi^n)\Delta_d\phi^{n+1} - 4\lambda U^n F(\phi^{n+1,2}), \tag{4.15}$$

$$\frac{U^{n+1} - U^n}{\Delta t} = D\Delta_d U^{n+1} + \frac{\phi^{n+1} - \phi^n}{2\Delta t}. \tag{4.16}$$

Equations (4.15) and (4.16) can be solved by a multigrid method [6, 53] which is a fast solver.

4.1.1. Discreteization for k-ford model. We discretize Eqs. (3.22) and (3.23):

$$\begin{aligned}
\epsilon^2(\theta^n)\frac{\phi^{n+1}-\phi^n}{\Delta t} &= \epsilon^2(\theta^n)\Delta_d\phi^{n+1,2} + 2\epsilon(\theta^n)\nabla_d\epsilon(\theta^n)\cdot\nabla_d\phi^n \\
&\quad -F'(\phi^{n+1}) - 4\lambda U^n F(\phi^{n+1,1}) \\
&\quad -\left(\epsilon'(\theta)\cdot\epsilon(\theta)\phi_y\right)_x^n + \left(\epsilon'(\theta)\cdot\epsilon(\theta)\phi_x\right)_y^n, \\
\frac{U^{n+1}-U^n}{\Delta t} &= D\Delta_d U^{n+1} + \frac{\phi^{n+1}-\phi^n}{2\Delta t}.
\end{aligned}$$

Here $\epsilon(\theta) = \epsilon_0(1+\epsilon_k\cos(k\phi))$ and $\tan\theta = \phi_y/\phi_x$. For $m = 1, 2$, $\phi^{n+1,m}$ are defined in the operator splitting scheme. We propose the following operator splitting scheme:

$$\begin{aligned}
\epsilon^2(\theta^n)\frac{\phi^{n+1,1}-\phi^n}{\Delta t} &= 2\epsilon(\theta^n)\nabla_d\epsilon(\theta^n)\cdot\nabla_d\phi^n \\
&\quad -\left(\epsilon'(\theta)\cdot\epsilon(\theta)\phi_y\right)_x^n + \left(\epsilon'(\theta)\cdot\epsilon(\theta)\phi_x\right)_y^n, \\
\phi^{n+1,2} &= \frac{\phi^{n+1,1}}{\sqrt{e^{-\frac{2\Delta t}{\epsilon^2(\theta^n)}} + (\phi^{n+1,1})^2\left(1-e^{-\frac{2\Delta t}{\epsilon^2(\theta^n)}}\right)}}, \\
\epsilon^2(\theta^n)\frac{\phi^{n+1}-\phi^{n+1,2}}{\Delta t} &= \epsilon^2(\theta^n)\Delta_d\phi^{n+1} - 4\lambda U^n F(\phi^{n+1,2}), \\
\frac{U^{n+1}-U^n}{\Delta t} &= D\Delta_d U^{n+1} + \frac{\phi^{n+1}-\phi^n}{2\Delta t}.
\end{aligned}$$

It should be pointed that θ equals atan2(ϕ_y, ϕ_x) and the two-argument function, atan2 is a variation of the arctangent function whose range is $[-\pi, \pi]$. It can be expressed as follows:

$$\text{atan2}(\phi_y, \phi_x) = \begin{cases} \tan^{-1}\left(\frac{\phi_y}{\phi_x}\right), \text{ if } \phi_x > 0, \\ \tan^{-1}\left(\frac{\phi_y}{\phi_x}\right) + \pi, \text{ if } \phi_y \geq 0,\ \phi_x < 0, \\ \tan^{-1}\left(\frac{\phi_y}{\phi_x}\right) - \pi, \text{ if } \phi_y < 0,\ \phi_x < 0, \\ \pi/2, \text{ if } \phi_y > 0,\ \phi_x = 0, \\ -\pi/2, \text{ if } \phi_y < 0,\ \phi_x = 0, \\ \text{undefined}, \text{ if } \phi_y = 0,\ \phi_x = 0. \end{cases}$$

To avoid singularities, we added a small value $\delta = 1\text{e}{-10}$ in the denominator

$$\theta = \text{atan2}\left(\phi_y, \text{sign}(\phi_x)(|\phi_x| + \delta)\right).$$

The $\text{sign}(a)$ function is defined as

$$\text{sign}(a) = \begin{cases} 1 & \text{if } a \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

We describe the following term with nine local points

$$(\epsilon'(\theta) \cdot \epsilon(\theta)\phi_y)^n_{x,ij} = \frac{(\epsilon'(\theta) \cdot \epsilon(\theta)\phi_y)^n_{i+\frac{1}{2},j} - (\epsilon'(\theta) \cdot \epsilon(\theta)\phi_y)^n_{i-\frac{1}{2},j}}{h},$$

where

$$\begin{aligned} &(\epsilon'(\theta) \cdot \epsilon(\theta)\phi_y)^n_{i+\frac{1}{2},j} \\ &= \frac{(\epsilon'(\theta^n_{i+1,j}) + \epsilon'(\theta^n_{ij}))(\epsilon(\theta^n_{i+1,j}) + \epsilon(\theta^n_{ij}))(\phi^n_{i+1,j+1} - \phi^n_{i+1,j-1} + \phi^n_{i,j+1} - \phi^n_{i,j-1})}{8h}, \\ &(\epsilon'(\theta) \cdot \epsilon(\theta)\phi_y)^n_{i-\frac{1}{2},j} \\ &= \frac{(\epsilon'(\theta^n_{ij}) + \epsilon'(\theta^n_{i-1,j}))(\epsilon(\theta^n_{ij}) + \epsilon(\theta^n_{i-1,j}))(\phi^n_{i,j+1} - \phi^n_{i,j-1} + \phi^n_{i-1,j+1} - \phi^n_{i-1,j-1})}{8h}. \end{aligned}$$

The other solvers are similarly defined.

4.2. Calculation of the crystal tip position and velocity

The crystal tip position and velocity are the important parameters in the phase–field simulation. To calculate these parameters with a high degree of accuracy we use a method based on the quadratic polynomial approximation. For simplicity, we only describe the procedure along the y-axis since the crystal is symmetric. Let y_k be the maximum y position on the interface at each time, and the quadratic polynomial approximation be:

$$y = \alpha x^2 + \beta x + \gamma.$$

Given three points: (x_{k-1}, y_{k-1}), (x_k, y_k), and (x_{k+1}, y_{k+1}) on the interface, where one of the three y points is a maximum value along the interface points, we calculate the parameters α, β, and γ from:

$$\begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} = \begin{pmatrix} x_{k-1}^2 & x_{k-1} & 1 \\ x_k^2 & x_k & 1 \\ x_{k+1}^2 & x_{k+1} & 1 \end{pmatrix}^{-1} \begin{pmatrix} y_{k-1} \\ y_k \\ y_{k+1} \end{pmatrix}.$$

Then using α, β, and γ, we find the tip position y_* which satisfies the following conditions:

$$\left.\frac{dy}{dx}\right|_{x_*} = 0 \text{ and } y_* = \alpha x_*^2 + \beta x_* + \gamma.$$

Furthermore, the crystal tip velocity can be obtained from the difference of tip positions at each time.

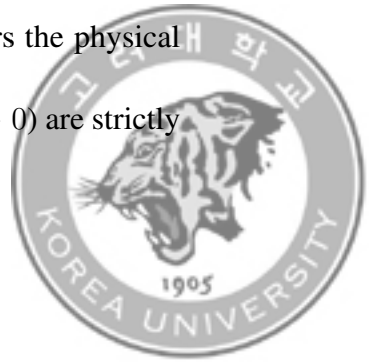
Chapter 5

Adaptive mesh refinement

5.1. Hierarchical structured Cartesian grids

The Cartesian grid is composed of the union of retangular grids. We consider a hierarchy of increasingly finer grids, $\Omega_{l+1}, \ldots, \Omega_{l+l^*}$, restricted to smaller and smaller subdomains, while the last hierarchy of global grids are $\Omega_0, \Omega_1, \ldots, \Omega_l$. That is, we consider a hierarchy of grids, $\Omega_0, \Omega_1, \ldots, \Omega_{l+0}, \Omega_{l+1}, \ldots, \Omega_{l+l^*}$. Here we denote Ω_{l+0} as level zero, Ω_{l+1} as level one, and so on. Construction of the multilevel mesh begins at the zero-level grid. Finer resolution grids are added at level one to cover those grid points on the zero grid where refinement is flagged. This process continues in the same fashion until the level l^* is reached. Moreover, the grid spacing h_k on level k is related to that of the level after $(k+1)$ as $h^k = \tau h^{k+1}$. τ is called as refinement ratio. The typical choices for this refinement radio are $\tau = 2$ or $\tau = 4$. Figure 5.1 shows a schematic illustration of the set of finer grids with four levels ($l^* = 3$) and refinement radio, $\tau = 2$.

It should be noted that in general only level zero completely covers the physical domain. That is the union of the grid patches at refined level $l + l^*$ ($l^* > 0$) are strictly



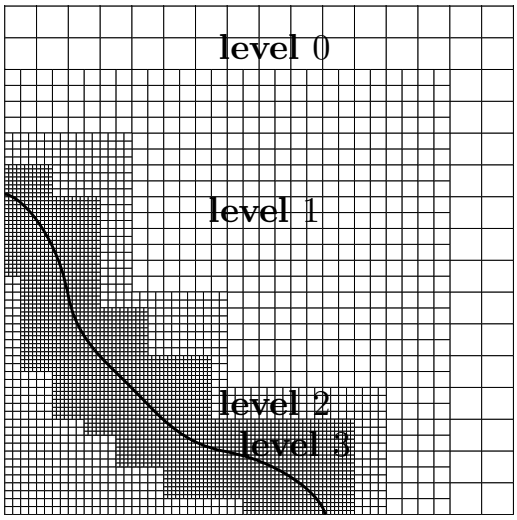


FIGURE 5.1. Hierarchical structured locally refinement with four levels.

contained in the uniform of the patches at coarse level $l + l^* - 1$. In this condition, we call the patch levels be properly nested. The nesting requirement is relaxed at the boundary condition for physical boundary condition and refined boundary condition. It is convenient to set the physical boundary condition at level zero Ω_{l+0} and apply a interpolation operation to the other levels in the proper nesting condition. The interpolation operation is applied with the information of its local domain. This algorithm will be described in Section 5.3. Figure 5.2(a) and (b) show the properly and improperly nested hierarchical structured locally refinement, respectively. As can be seen, in the proper nesting condition, the interpolations at level $l + l^*$ can be obtained by using the points at level $l + l^* - 1$. While improper refinement of cells will fail to satisfy that (see the second level in Fig. 5.2(b)).

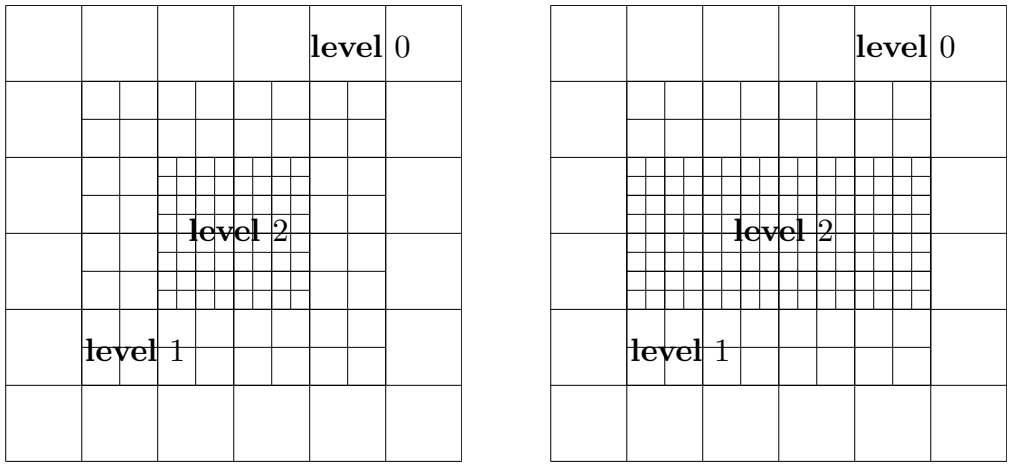


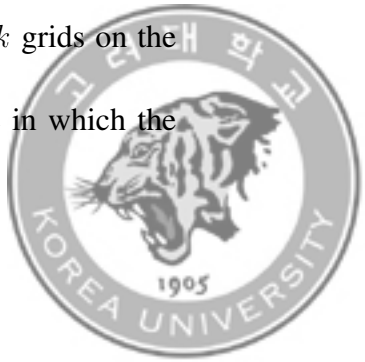
FIGURE 5.2. (a) A properly nested hierarchical structured locally refinement. (b) A improperly nested hierarchical structured locally refinement.

5.2. Creation of the grid hierarchy

To construct new refined grids, we may decide that previously refined cells were unnecessary and deallocate portions of refined domains. There are many possible criteria for deciding where refinement is necessary. In many physical problems, physical quantities like sharp density gradients or large charge distributions may provide indicators for refinement. In the current implementation, the grid is adapted dynamically based on the undivided gradient $|\nabla_u \phi|^k$ which is defined as following

$$|\nabla_u \phi|^k_{ij} = \sqrt{(\phi^k_{i+1,j} - \phi^k_{i-1,j})^2 + (\phi^k_{i,j+1} - \phi^k_{i,j-1})^2},$$

where ϕ^k_{ij} are cell center coordinates defined with respect to the level-k grids on the domain Ω_{l+k}. Then we can tag cells that contain the front, i.e., those in which the



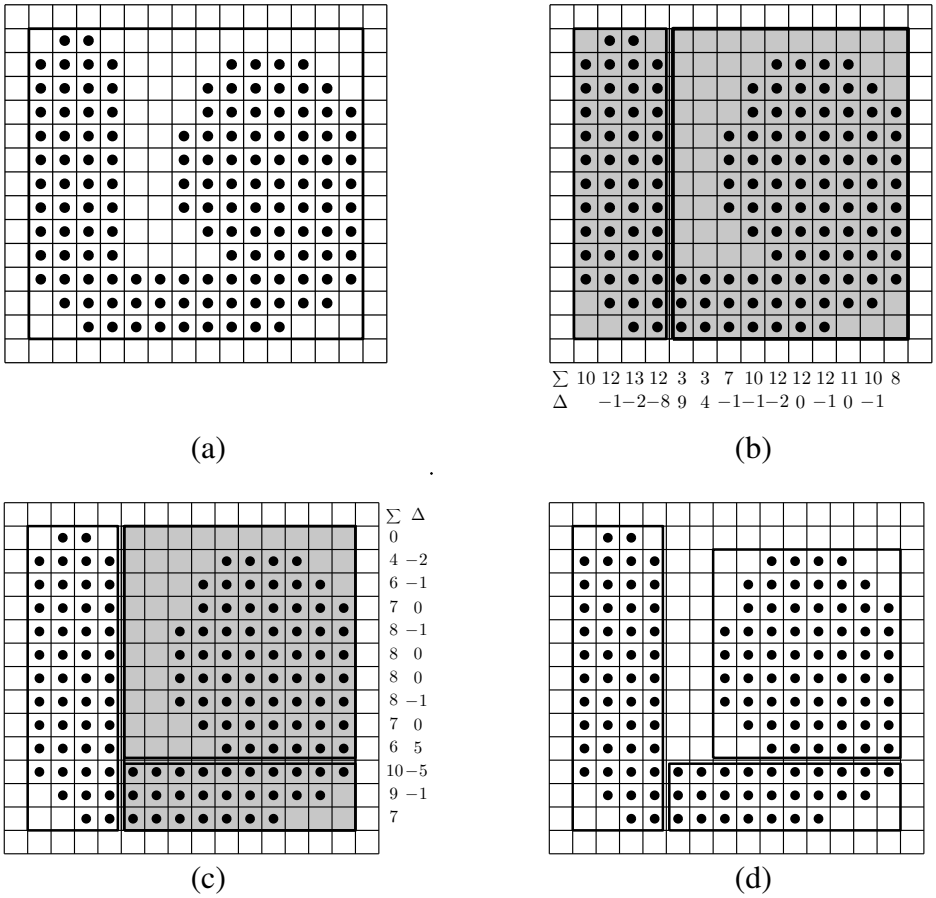
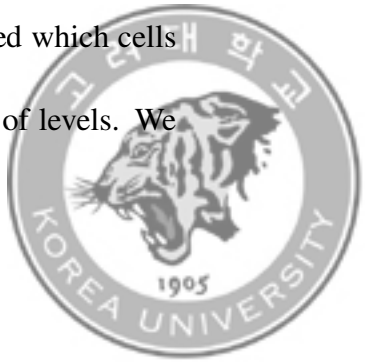


FIGURE 5.3. The three steps in regridding algorithm. (a) tag error cells and enclosed in a box, (b) split the box into two based on a histogram of the column or row sums of tagged cells, (c) fit new boxes to each split box and repeat if the ratio of tagged to untagged cells is too small, and (d) the most efficient rectangles.

undivided gradient of the film height is greater than a critical value tol, i.e., $|\nabla_u \phi|^k \geq tol$. All through our paper, we simplify set $tol = 0.01$. In this way, level-k grid cells are marked when their divided gradient $|\nabla \phi|^k \geq h^k tol = 2h^{k+1} tol$. That is, $\Omega_{l+k+1} \subseteq \Omega_{l+k}$. Thus this method is well defined. Once we have decided which cells are to be refined, we need to use this information to create a hierarchy of levels. We



use the algorithm of Berger and Rigoutsos [41], in which tagged points are clustered into efficient boxes. The efficiency of a grid is defined as the number of cells in the grid which were tagged for refinement divided by the total number of cells in the grid. The grid generator takes a list of tagged points and draws the smallest possible box around them. For efficiency, the boxes are not allowed to become too small, nor must they be too empty. Here we simplify drawn with the following steps:

1: Fit a box to enclose the tagged cells.

2: Recursively sub-divide the box. Split the box in the longest direction at a position based on the histogram formed from the sum of the number of tagged cells per row or column.

3: After splitting the box, fit new bounding boxes to each half and repeat the process. Continue until the size of every box is not smaller than given parameter and every box consists at least number of non-tagged cells.

4: Compute fill ratio, which is defined as equalling the number of tagged cells divides size of box. For example, fill ratio is $19/28$ in Fig. 5.3(c). If this grid does not meet an efficiency criterion (in the current implementation, fill ratio ≥ 0.75), the grid generator will look for the best way to subdivide the tagged points in order to create more efficient boxes.

5.3. Boundary interpolation

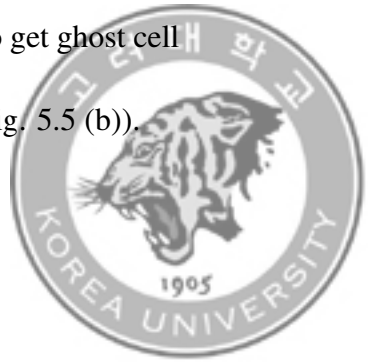
In order to employ the discretisation of Lapacian on a generic level grid, we should use the usual nearest neighbor stencils in the definitions of the discrete derivatives to fill the ghost-layer values by interpolation. It should be noted that in this section we consider $\tau = 2$ and the operation for $\tau = 4$ can be similar defined. There are two operations for boundary interpolation such as fine-coarse and coarse-fine boundary interpolations.

All possible fine-coarse boundary interpolations for two-dimensional locally refined grid are shown in Fig. 5.4. Ghost cells at the fine-coarse interface are indicated by square, whereas valid cell in coarse and fine interfaces are indicated by open circle and x-marks.

For the fine-coarse case, we use the average of near four fined cells as

$$\phi^c_{ij} = \frac{\phi^f_{2i-1,2j-1} + \phi^f_{2i-1,2j} + \phi^f_{2i,2j-1} + \phi^f_{2i,2j}}{4}.$$

The grid-to-ghost-layer exchange process is illustrated in Fig. 5.5 (a). The main idea is that using quadratic interpolation through coarse cells (open circles) to get intermediate values (solid circles), then use intermediate value with fine cells ($\times$'s) to get ghost cell values (open triangles) for computing coarse-fine fluxes (arrows) (see Fig. 5.5 (b)).



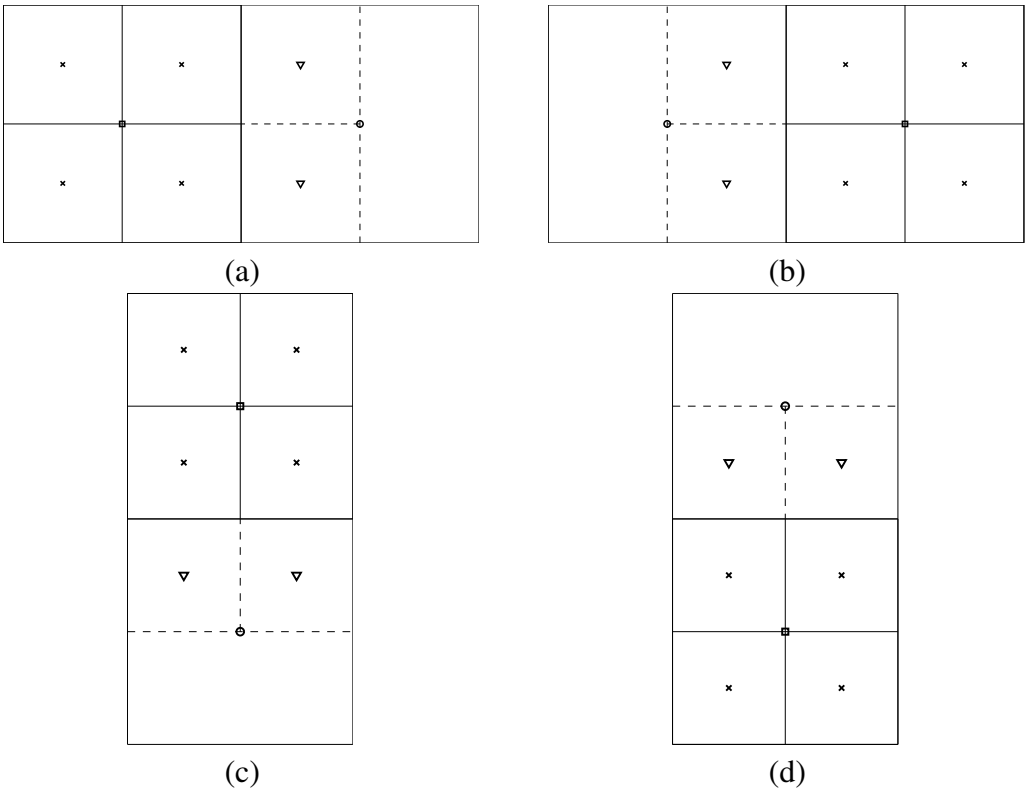
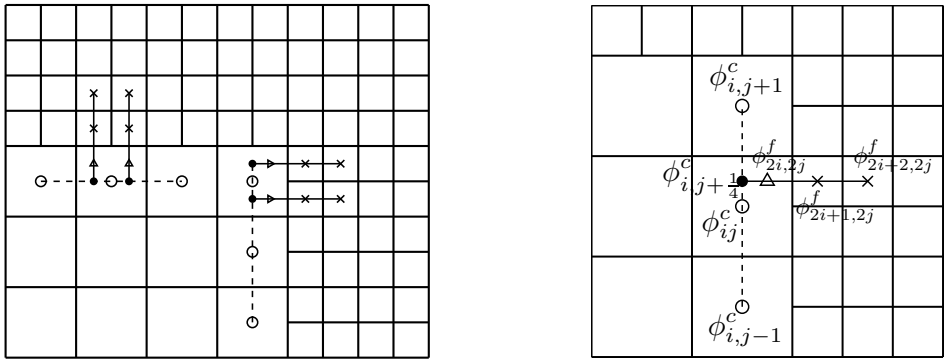


FIGURE 5.4. All possible fine-coarse and coarse-fine boundary interpolations for two-dimensional locally refined grid. Ghost cells at the fine-coarse and coarse-fine interface are indicated by square and triangle, respectively, whereas valid cell in coarse and fine interfaces are indicated by open circle and x-marks.

To obtain the value $\phi^c_{i,j+\frac{1}{4}}$ (solid circle), we can calculate it by using three points $\phi^c_{i,j-1}$, $\phi^c_{i,j}$, and $\phi^c_{i,j+1}$ with the quadratic polynomial approximation. Let the quadratic polynomial approximation be $\phi^c(t) = \alpha t^2 + \beta t + \gamma$. And assume $\phi^c(-h^c) = \phi^c_{i,j-1}$, $\phi^c(0) = \phi^c_{ij}$, and $\phi^c(h^c) = \phi^c_{i,j+1}$, then the parameters α, β, and γ can be calculated by the following equations.

$$\begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} = \begin{pmatrix} (h^c)^2 & -h^c & 1 \\ 0 & 0 & 1 \\ (h^c)^2 & h^c & 1 \end{pmatrix}^{-1} \begin{pmatrix} \phi^c_{i,j-1} \\ \phi^c_{ij} \\ \phi^c_{i,j+1} \end{pmatrix}.$$



(a) (b)

FIGURE 5.5. (a) The grid-to-ghost-layer exchange process. (b) Interpolation on the coarse fine boundary in detail: Use quadratic interpolation through coarse cells (open circles) to get intermediate values (solid circles), then use intermediate value with fine cells ($\times$'s) to get ghost cell values (open triangles) for computing coarse-fine fluxes (arrows).

Now using α, β, and γ, we get $\phi^c_{i,j+\frac{1}{4}} = \phi^c(-h^c/4)$. Secondly, to get the value $\phi^f_{2i,2j}$, we implement another quadratic interpolation with the values $\phi^c_{i,j+\frac{1}{4}}$, $\phi^f_{2i+2,2j}$, and $\phi^f_{2i+2,2j}$.

Let the quadratic polynomial approximation be $\phi^f(t) = \alpha t^2 + \beta t + \gamma$. And assume $\phi^f(-h^f) = \phi^c_{i,j+\frac{1}{4}}$, $\phi^f(\frac{h^f}{2}) = \phi^f_{2i+1,2j}$, and $\phi^f(\frac{3h^f}{2}) = \phi^f_{2i+2,2j}$, then the parameters α, β, and γ can be calculated by the following equations.

$$\begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} = \begin{pmatrix} (h^f)^2 & -h^f & 1 \\ (\frac{h^f}{2})^2 & \frac{h^f}{2} & 1 \\ (\frac{3h^f}{2})^2 & (\frac{3h^f}{2})^2 & 1 \end{pmatrix}^{-1} \begin{pmatrix} \phi^c_{i,j+\frac{1}{4}} \\ \phi^f_{2i+1,2j} \\ \phi^f_{2i+2,2j} \end{pmatrix}.$$

Then using α, β, and γ, we get $\phi^f_{2i,2j} = \phi^f(-h^f/2)$.

For the three dimensional case, the operations are defined in the same fashion.

5.4. Algorithm for mesh plots

In the adaptive system, the cells locate at the center in its level as shown in Fig. 5.6. To see the numerical results obtained by adaptive method, triangular mesh plot

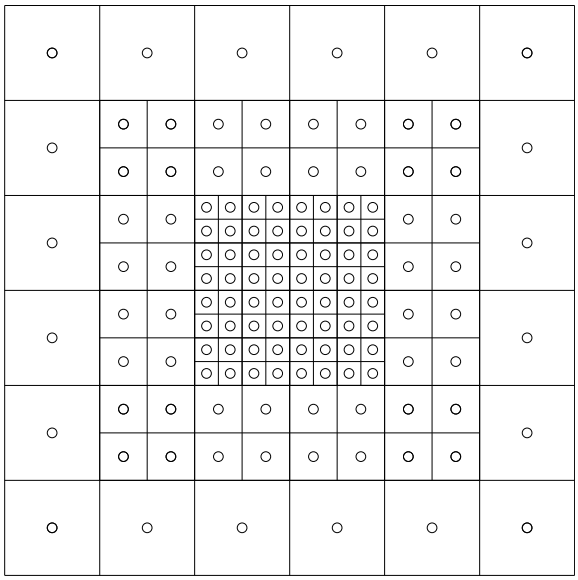


FIGURE 5.6. The cells locate at the center in its level.

or mesh plot is a general choice. Here we use mesh plot and drawn the algorithm as following.

1: Define the total level, l^* and set $k = 0$. For example, $l^* = 2$ in Fig. 5.6.

2: For every point (ϕ_{ij}^c) which locates in the coarse k-level, i.e., $k < l^*$, we use the same value by the near four fined cells as

$$\phi_{2i-1,2j-1}^f = \phi_{2i-1,2j}^f = \phi_{2i,2j-1}^f = \phi_{2i,2j}^f = \phi_{ij}^c.$$

Please refer to Fig. 5.7.

3: Set $k = k + 1$. Do Step 2, until $k = l^*$.

With these three steps, we can get the mesh plots of results as shown in Fig. 5.8. The codes of mesh plots in two- and three- dimensional spaces are given in the Appendix.

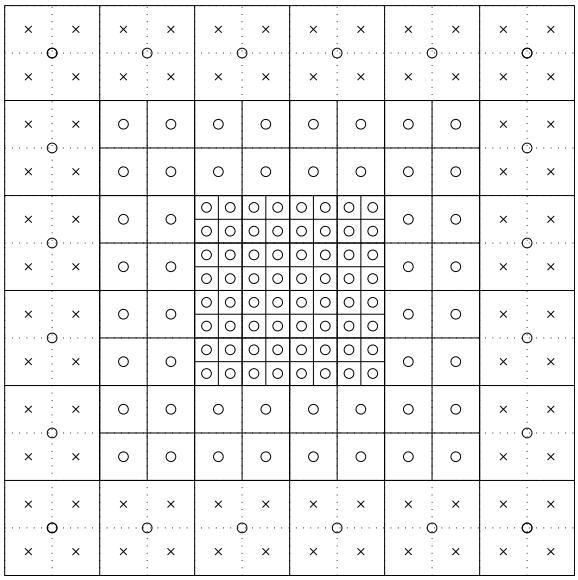


FIGURE 5.7. Set the coarse value, $\phi^f_{2i-1,2j-1}$ to the near four fined cells as $\phi^f_{2i-1,2j-1} = \phi^f_{2i-1,2j} = \phi^f_{2i,2j-1} = \phi^f_{2i,2j} = \phi^c_{ij}$. Here $\times$ respects the fine cells near to its coarse level.

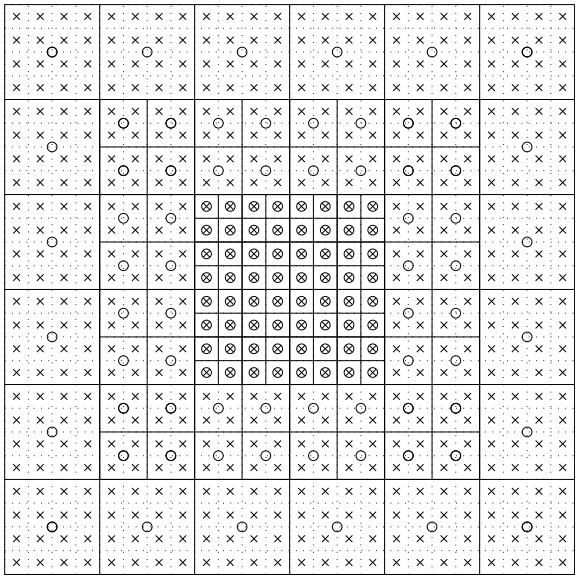


FIGURE 5.8. Final results ($\times$) for mesh plot. Circle respects the initial location.

Chapter 6

Adaptive mesh refinement multigrid algorithm

To solve the resulting system of discrete Eqs. (4.13)–(4.16) at the implicit time level, we use an adaptive mesh refinement [2, 50], whose schematic diagram is shown in Fig. 5.1. In the adaptive approach, the grid is adapted dynamically based on the undivided gradient. First, we tag cells that contain the front, i.e., those in which the undivided gradient of the phase-field is greater than a critical value. Then, the tagged cells are grouped into rectangular patches by using a clustering algorithm. These rectangular patches are refined to form the grids at the next level. The process is repeated until a specified maximum level is reached. In the rectangular patches at every level, we use the multigrid method to solve governing equation. This method is also called as adaptive mesh refinement multigrid algorithm. We describe the adaptive full approximation storage cycle to solve the discrete system on the hierarchy of increasingly finer grids. First, let us rewrite Eqs. (4.15) and (4.16) as

$$N(\phi^{n+1}, U^{n+1}) = (\varphi^n, \psi^n),$$

where $N(\phi^{n+1}, U^{n+1}) = \left(\frac{\phi^{n+1}}{\Delta t} - \Delta_d \phi^{n+1}, \frac{U^{n+1}}{\Delta t} - D\Delta_d U^{n+1} - \frac{\phi^{n+1}}{2\Delta t}\right)$ and the source term is $(\varphi^n,\ \psi^n) = \left(\frac{\phi^{n+1,2}}{\Delta t} - \frac{4\lambda U^n F(\phi^{n+1,2})}{\epsilon^2(\phi^n)},\ \frac{2U^n - \phi^n}{2\Delta t}\right)$.

Using the above notations on all levels $k = 0, 1, \ldots, l, l+1, \ldots, l+l^*$, an adaptive multigrid cycle is formally written as follows [53]:

Adaptive cycle

We calculate φ_k^n, ψ_k^n on all levels and set the previous time solution as the initial guess, i.e., $(\phi_k^0, U_k^0) = (\phi_k^n, U_k^n)$.

$$(\phi_k^{m+1}, U_k^{m+1}) = ADAPTIVEcycle(k, \phi_k^m, \phi_{k-1}^m, U_k^m, U_{k-1}^m, N_k, \varphi_k^n, \psi_k^n, \nu).$$

1) Presmoothing

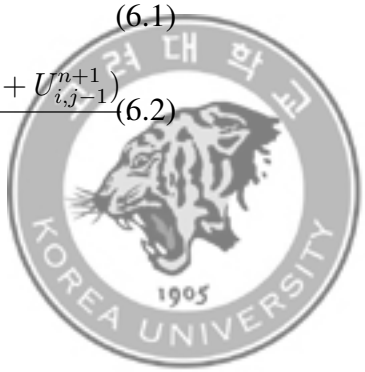
- Compute $(\bar{\phi}_k^m,\ \bar{U}_k^m)$ by applying ν smoothing steps to $(\phi_k^m,\ U_k^m)$ on Ω_k.

$$(\bar{\phi}_k^m,\ \bar{U}_k^m) = SMOOTH^{\nu}(\phi_k^m, U_k^m, N_k, \varphi_k^n, \psi_k^n),$$

where one $SMOOTH$ relaxation operator step consists of solving Eqs. (6.1) and (6.2) given below by a 2×2 matrix inversion for each i and j. Rewriting Eqs. (4.15) and (4.16), we get

$$\left(\frac{1}{\Delta t} + \frac{4}{h^2}\right)\phi_{ij}^{n+1} = \varphi_{ij}^n - \frac{\phi_{i+1,j}^{n+1} + \phi_{i-1,j}^{n+1} + \phi_{i,j+1}^{n+1} + \phi_{i,j-1}^{n+1}}{h^2}, \tag{6.1}$$

$$\left(\frac{1}{\Delta t} + \frac{4D}{h^2}\right)U_{ij}^{n+1} - \frac{\phi_{ij}^{n+1}}{2\Delta t} = \psi_{ij}^n - \frac{D(U_{i+1,j}^{n+1} + U_{i-1,j}^{n+1} + U_{i,j+1}^{n+1} + U_{i,j-1}^{n+1})}{h^2} \tag{6.2}$$



Next, we replace ϕ_{kl}^{n+1} and U_{kl}^{n+1} in Eqs. (6.1) and (6.2) with $\bar{\phi}_{kl}^{m}$ and $\bar{U}_{kl}^{m}$ if $k \leq i$ and $l \leq j$; otherwise we replace them with ϕ_{kl}^{m} and U_{kl}^{m}, i.e.,

$$\left(\frac{1}{\Delta t}+\frac{4}{h^2}\right)\bar{\phi}_{ij}^{m} = \varphi_{ij}^{n} - \frac{\phi_{i+1,j}^{m}+\bar{\phi}_{i-1,j}^{m}+\phi_{i,j+1}^{m}+\bar{\phi}_{i,j-1}^{m}}{h^2}, \tag{6.3}$$

$$\left(\frac{1}{\Delta t}+\frac{4D}{h^2}\right)\bar{U}_{ij}^{m} - \frac{\bar{\phi}_{ij}^{m}}{2\Delta t} = \psi_{ij}^{n} - \frac{D(U_{i+1,j}^{m}+\bar{U}_{i-1,j}^{m}+U_{i,j+1}^{m}+\bar{U}_{i,j-1}^{m})}{h^2}. \tag{6.4}$$

2) *Coarse-grid correction*

- Compute $(\bar{\phi}_{k-1}^{m}, \bar{U}_{k-1}^{m}) = \begin{cases} I_k^{k-1}(\bar{\phi}_k^m, \bar{U}_k^m) & \text{on } \Omega_{k-1}\cap\Omega_k \\ (\phi_{k-1}^m, U_{k-1}^m) & \text{on } \Omega_{k-1}-\Omega_k. \end{cases}$

- Compute the coarse grid source term

$$(\varphi_{k-1}^{n}, \psi_{k-1}^{n}) = \begin{cases} I_k^{k-1}\{(\varphi_k^n, \psi_k^n) - N_k(\bar{\phi}_k^m, \bar{U}_k^m)\} \\ +N_{k-1}I_k^{k-1}(\bar{\phi}_k^m, \bar{U}_k^m) \text{ on } \Omega_{k-1}\cap\Omega_k \\ (\varphi_{k-1}^n, \psi_{k-1}^n) \text{ on } \Omega_{k-1}-\Omega_k. \end{cases}$$

- Compute an approximate solution $(\hat{\phi}_{k-1}^{m},\ \hat{U}_{k-1}^{m})$ of the coarse grid equation on Ω_{k-1}, i.e.,

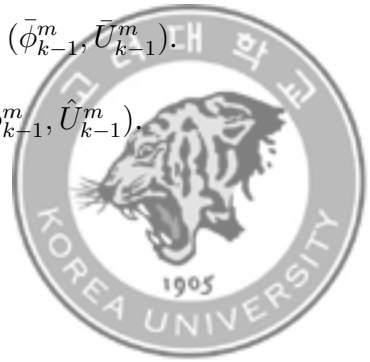
$$N_{k-1}(\phi_{k-1}^m, U_{k-1}^m) = (\varphi_{k-1}^n, \psi_{k-1}^n). \tag{6.5}$$

If $k = 1$, we explicitly invert a 2×2 matrix to obtain the solution. If $k > 1$, we solve Eq. (6.5) by using $(\bar{\phi}_{k-1}^{m},\ \bar{U}_{k-1}^{m})$ as an initial approximation to perform an adaptive multigrid k-grid cycle:

$$(\hat{\phi}_{k-1}^{m}, \hat{U}_{k-1}^{m}) = \text{ADAPTIVEcycle}(k-1, \bar{\phi}_{k-1}^{m}, \phi_{k-2}^{m}, \bar{U}_{k-1}^{m}, U_{k-2}^{m}, N_{k-1}, \varphi_{k-1}^{n}, \psi_{k-1}^{n}, \nu).$$

- Compute the correction at $\Omega_{k-1}\cap\Omega_k$. $(\hat{u}_{k-1}^{m}, \hat{v}_{k-1}^{m}) = (\hat{\phi}_{k-1}^{m}, \hat{U}_{k-1}^{m}) - (\bar{\phi}_{k-1}^{m}, \bar{U}_{k-1}^{m})$.

- Set the solution at the other points of $\Omega_{k-1}-\Omega_k$. $(\phi_{k-1}^{m+1}, U_{k-1}^{m+1}) = (\hat{\phi}_{k-1}^{m}, \hat{U}_{k-1}^{m})$.



- Interpolate the correction to Ω_k. $(\hat{u}_k^m, \hat{v}_k^m) = I_{k-1}^k(\hat{u}_{k-1}^m, \hat{v}_{k-1}^m)$.

- Compute the corrected approximation on Ω_k.

$$(\phi_k^{m,\text{ after }CGC}, U_k^{m,\text{ after }CGC}) = (\bar{\phi}_k^m + \hat{u}_k^m, \bar{U}_k^m + \hat{v}_k^m).$$

3) Postsmoothing

$$(\phi_k^{m+1}, U_k^{m+1}) = SMOOTH^{\nu}(\phi_k^{m,\text{ after }CGC}, U_k^{m,\text{ after }CGC}, N_k, \varphi_k^n, \psi_k^n).$$

This completes the description of an adaptive multigrid cycle. For additional details about the adaptive multigrid cycle, please refer to [53]. The following code (FORTRAN-style) retrieve this information, presmoothing and postsmoothing, coarse-grid correction.

```
ccccccccc presmoothing and postsmoothing ccccccccccc
 c   a_dt: \Dleta t
 c   dd: D
 c   phi(i,j,0): \phi
 c   phi(i,j,1): U
   dxinv = 1.0/(dx*dx)
   do j = CHF_LBOUND[region; 1], CHF_UBOUND[region; 1]
   do i = CHF_LBOUND[region; 0], CHF_UBOUND[region; 0]

        a = 1.0/a_dt + 4.0*dxinv
        b = 0.0
        c = -0.5/a_dt
        d = 1.0/a_dt + 4.0*dd*dxinv

        s1 = rhs(i,j,0) + (phi(i+1,j,0) + phi(i-1,j,0)
  &         + phi(i,j+1,0) + phi(i,j-1,0))*dxinv

        s2 = rhs(i,j,1) + dd*(phi(i+1,j,1) + phi(i-1,j,1)
  &         + phi(i,j+1,1) + phi(i,j-1,1))*dxinv

      phi(i,j,0) = (d*s1-b*s2)/(a*d-b*c)
```

```
      phi(i,j,1) = (-c*s1+a*s2)/(a*d-b*c)

       enddo
    enddo
```

```
ccccccccccccc coarse-frid correction ccccccccccccccccc
 c   a_dt: \Dleta t
 c   dd: D
 c   phi(i,j,0): \phi
 c   phi(i,j,1): U

   dxinv = 1.0/(dx*dx)
   do j = CHF_LBOUND[region; 1], CHF_UBOUND[region; 1]
   do i = CHF_LBOUND[region; 0], CHF_UBOUND[region; 0]

    lmu  = (phi(i+1,j,1) - 4.0*phi(i,j,1)
  &  + phi(i-1,j,1)+ phi(i,j+1,1) + phi(i,j-1,1))*dxinv

    lphi = (phi(i+1,j,0) - 4.0*phi(i,j,0)
  &  + phi(i-1,j,0)+ phi(i,j+1,0) + phi(i,j-1,0))*dxinv

    lofphi(i,j,0) = phi(i,j,0)/a_dt- lphi

    lofphi(i,j,1) = phi(i,j,1)/a_dt
  & -dd*lmu - 0.5*phi(i,j,0)/a_dt

      enddo
   enddo
```

Chapter 7

Numerical results

In this section we perform numerical experiments for two- and three-dimensional solidification to validate that our proposed scheme is accurate, efficient, and robust.

For two-dimensional tests, we take the initial state as:

$$\phi(x,y,0) = \tanh\left(\frac{R_0 - \sqrt{x^2+y^2}}{\sqrt{2}}\right) \text{ and } U(x,y,0) = \begin{cases} 0 & \text{if } \phi > 0 \\ \Delta & \text{else.} \end{cases}$$

For three-dimensional tests, these are similarly defined:

$$\phi(x,y,z,0) = \tanh\left(\frac{R_0 - \sqrt{x^2+y^2+z^2}}{\sqrt{2}}\right) \text{ and } U(x,y,z,0) = \begin{cases} 0 & \text{if } \phi > 0 \\ \Delta & \text{else.} \end{cases}$$

The zero level set ($\phi = 0$) represents a circle of radius R_0. From the dimensionless variable definition the value $U = 0$ corresponds to the melting temperature of the pure material, while $U = \Delta$ is the initial undercooling. The extension to three dimensions is straightforward. The capillary length, d_0, is defined as $d_0 = a_1/\lambda$ [8, 30, 46] with $a_1 = 0.8839$ [23, 24, 46] and $\lambda = 3.1913$ [46]. And other parameter is chosen as follows: $\epsilon_0 = 1$ and $\epsilon_4 = 0.05$.

7.1. Evolution for crystal growth in two- and three-dimensional spaces

In this experiment, we will show the evolution of crystal growth in two and three dimensions with adaptive mesh method. The computational domains are set as $\Omega = (-800, 800)^2$ with $l^* = 4$ levels in two-dimensional case and $\Omega = (-200, 200)^3$ with $l^* = 4$ levels in three-dimensional case. And other parameters are chosen as $\Delta t = 0.4$, $R_0 = 14d_0$, and $\Delta = -0.55$. The calculations are run up to time $T = 8000$ in two dimensional space. Figure 7.1(a) shows the temporal evolution of crystal interface with time step $\Delta t = 0.15$ at times $t = 800i$, for $i = 0, 1, \cdots, 10$ (from inside to outside). And the mesh plots of the corresponding order parameter at times $t = 400$, 2000, 3600, 5200, 6800, and 8000 are shown in Fig. 7.1(b-g). A sample two dimensional adaptive meshes with different views are shown in Fig. 7.2(a)-(c).

To show the evolution of the k-fold crystal growth in general, we simulate sequences of computational experiments of k-fold symmetric crystal growth for $k = 4, \ldots, 9$. A 1024×1024 mesh is used on the domain $\Omega = (-200, 200)^2$ and we take $R_0 = 15d_0$, $\Delta = -0.55$, and $\Delta t = 0.3$. Note that we use $\epsilon_k = 1/(k^2 - 1)$ to respond to the Wulff's algorithm. The evolutions for each k are shown in Fig. 7.3.

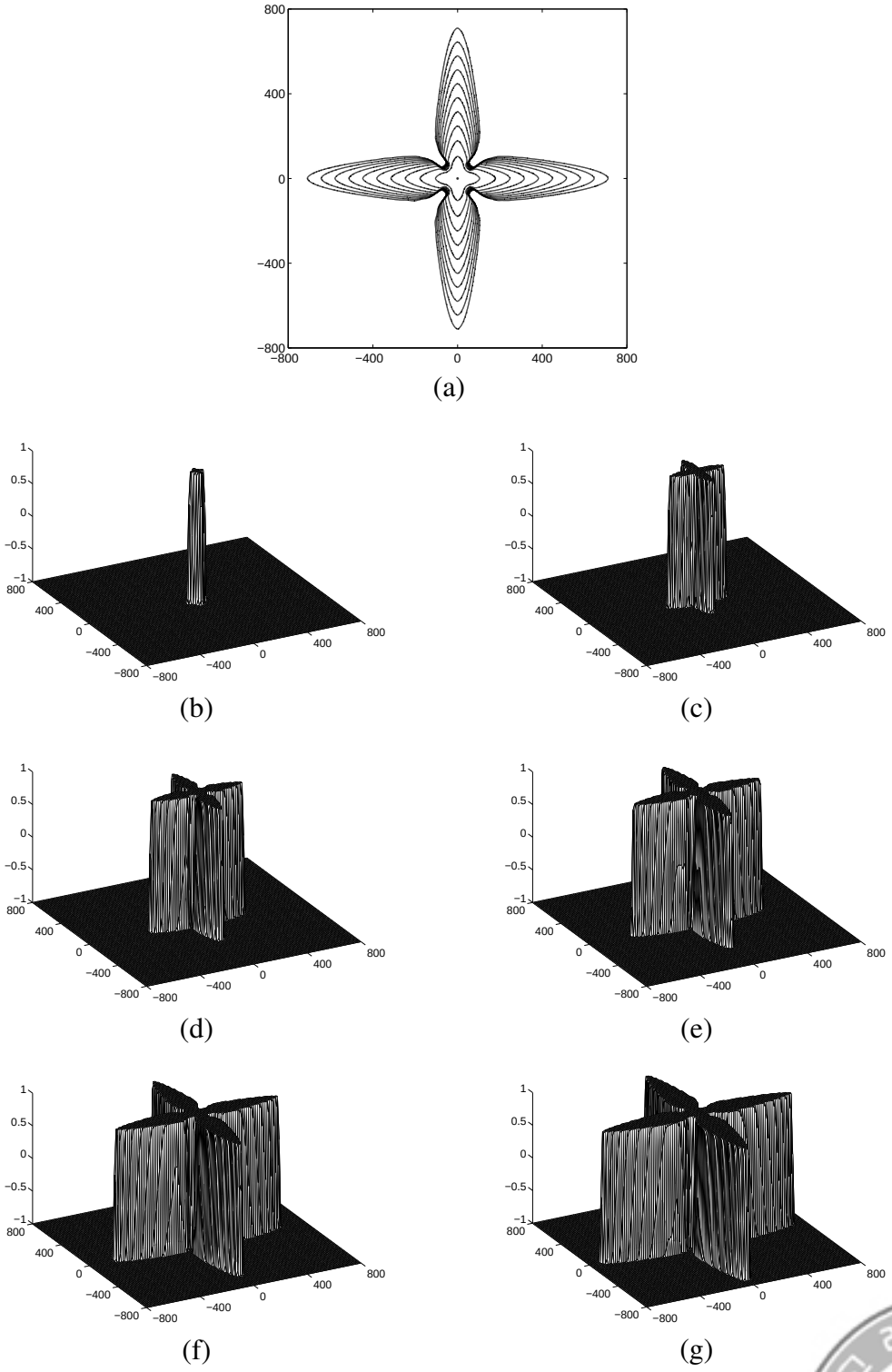
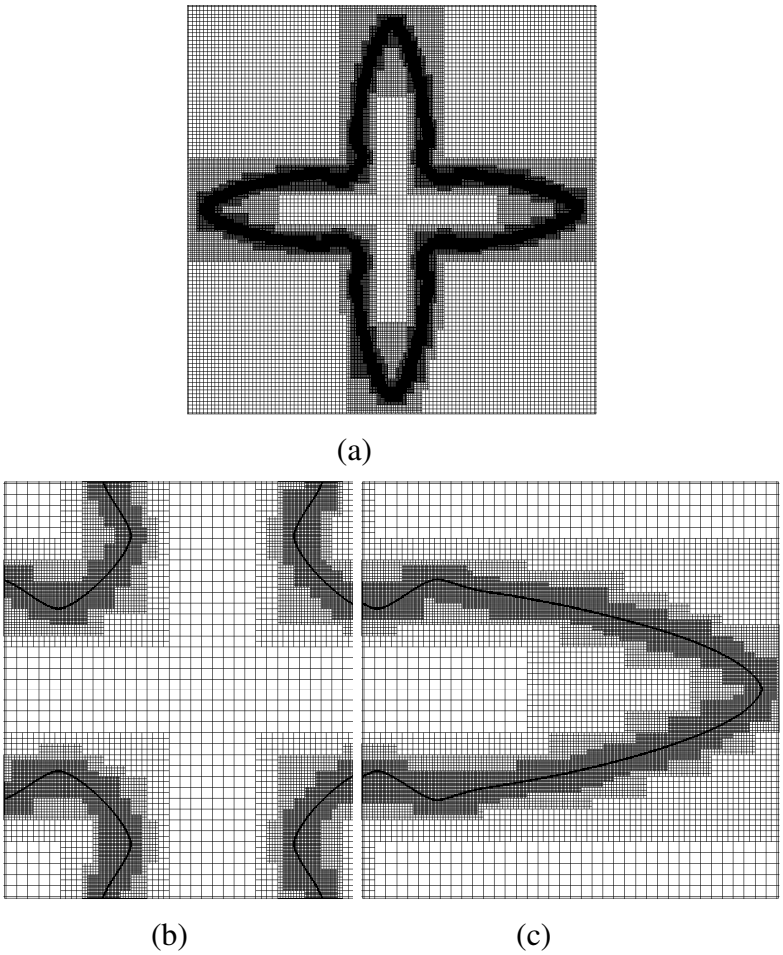


FIGURE 7.1. The evolution for crystal growth in two dimensions. (a) The temporal evolution by using the zero contour of the order parameter at times $t = 800i$, for $i = 0, 1, \cdots, 10$ (from inside to outside). (b-g) Mesh plots for order parameter at times $t = 400,\ 2000,\ 3600,\ 5200,\ 6800,$ and 8000.

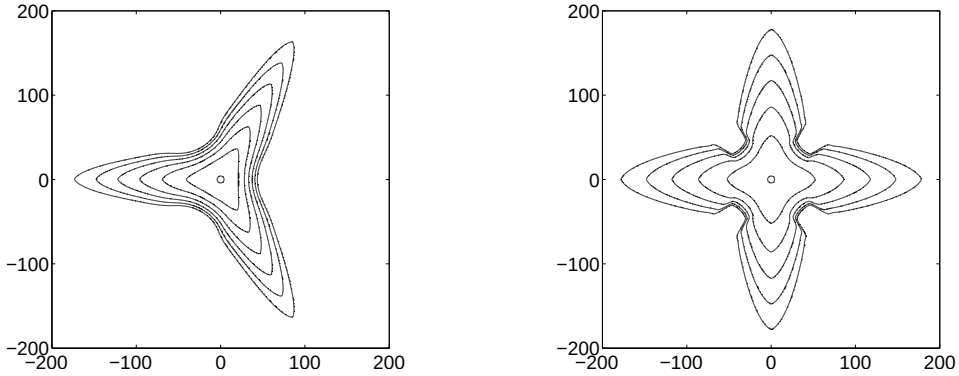


(a)

(b) (c)

FIGURE 7.2. A sample adaptive mesh in different views in two dimensions. (a) whole view. (b) and (c) closeup views.

Figure 7.4(a) shows three-dimensional structures at times $t = 0,\ 20,\ 40,\ 160,\ 240,$ and, 480 (from left to right). And in Fig 7.4(b), we show the bounding boxes at times $t = 0,\ 240,$ and 480 to show the structured local refinement.



(a) $k = 3$ (b) $k = 4$

(c) $k = 5$ (d) $k = 6$

(e) $k = 7$ (f) $k = 8$

FIGURE 7.3. The evolutions of k-fold crystal growth after time: (a) $T = 720$, (b) $T = 1200$, (c) $T = 1680$, (d) $T = 2160$, (e) $T = 2520$, and (f) $T = 2880$.

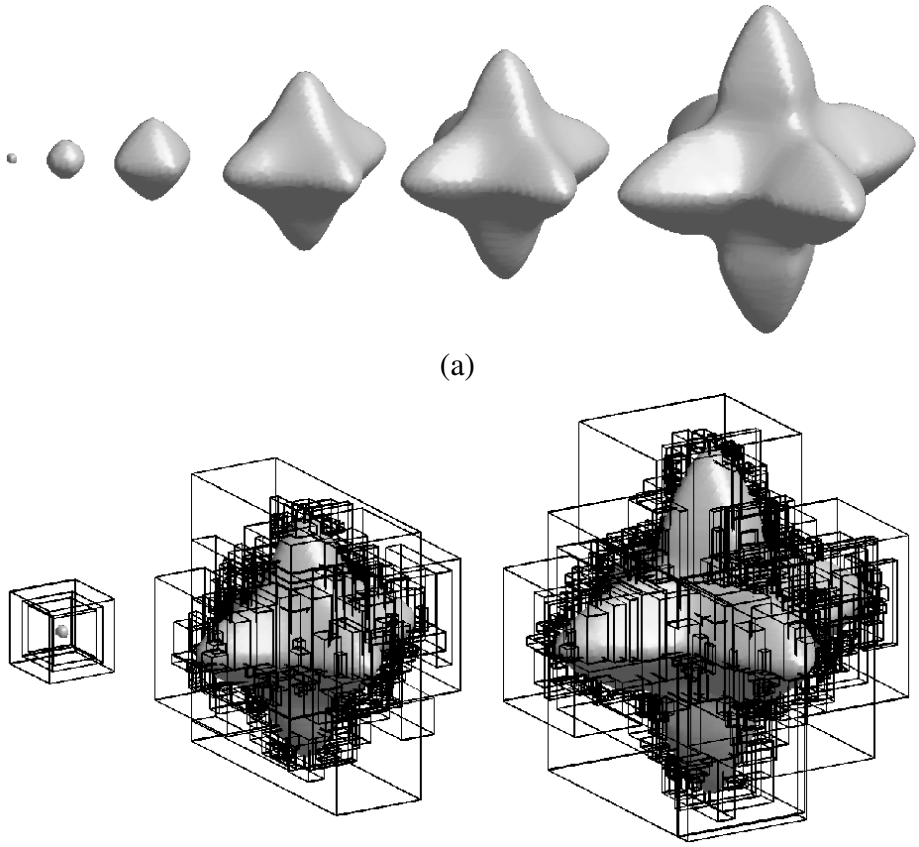
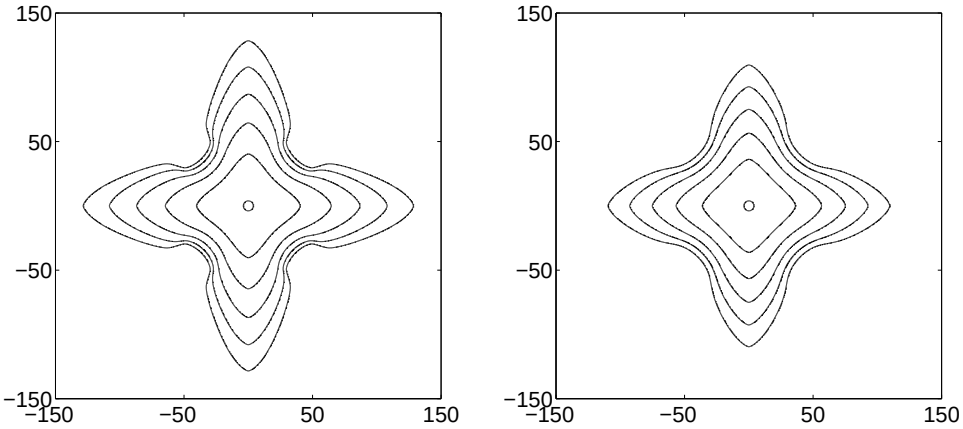


FIGURE 7.4. Snapshots of three-dimensional evolution of crystal growth. (a) crystal shape at times $t = 0,\ 20,\ 40,\ 80,\ 240,$ and, 480 (from left to right). (b) the bounding boxes at times $t = 0$, 240, and 480.

7.2. Stability of the operator splitting algorithm

As mentioned in Chapter 6, explicit schemes [19, 43, 44, 45] suffer from time step restrictions $\Delta t \leq O(h^2)$ for the stability. In order to show the stability of our proposed method we consider the evolution of an interface with arbitrarily large time steps. In these simulations a 2048×2048 mesh is used on the computational domain $\Omega =$

$(-200, 200)^2$. We choose $R_0 = 14d_0$, $\Delta = -0.55$, and $h = 0.1953$. The calculations are run up to time $T = 900$ with different time steps $\Delta t = 0.3$ and $\Delta t = 0.6$. Note that both time steps are larger than h. Figures 7.5 (a) and (b) show evolutions of the interface with different time steps $\Delta t = 0.3$ and $\Delta t = 0.6$, respectively. In general, large time steps may cause large truncation errors, however, as can be seen in Fig. 7.5 our proposed scheme works well with large time steps.



(a) $\Delta t = 0.3$ (b) $\Delta t = 0.6$

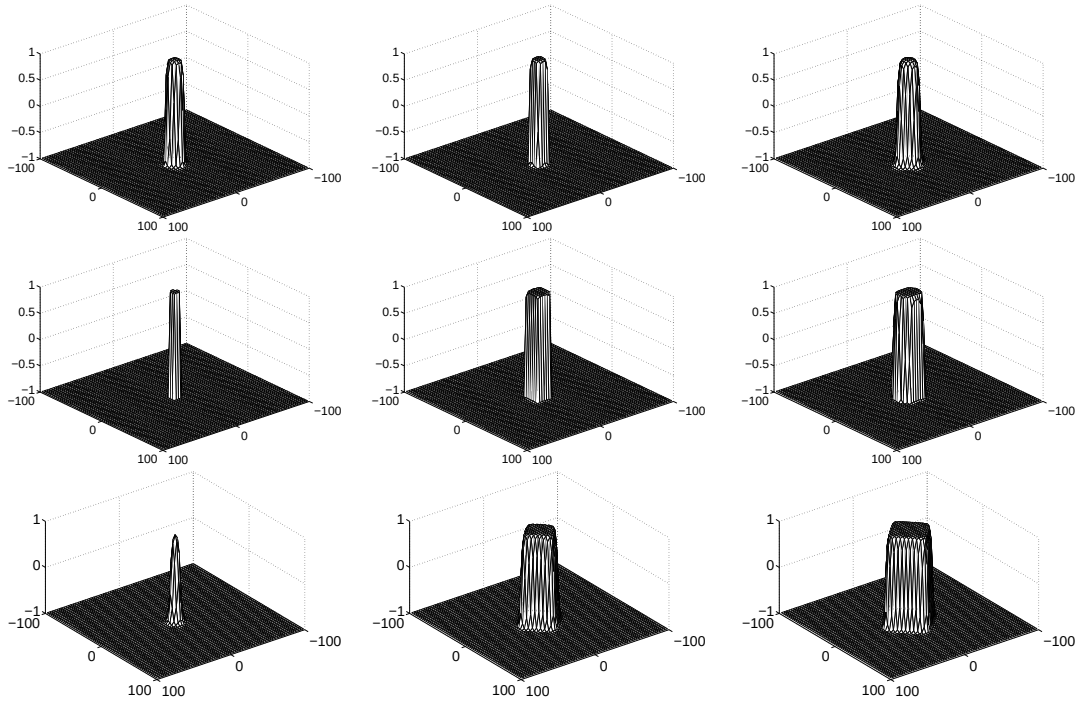
FIGURE 7.5. (a) and (b) show the sequence of interfaces with different time steps $\Delta t = 0.3$ and $\Delta t = 0.6$, respectively. The times are $t = 0$, 180, 360, 540, 720, and 900 (from inside to outside).

Next, we perform a number of simulations on a set of increasingly finer grids to show that our proposed method is restricted by the stability constraint $\Delta t \leq O(h)$. The computational domain is $\Omega = (-100, 100)^2$ and we take $R_0 = 14d_0$ and $\Delta = -0.55$. The numerical solutions are computed on the uniform grids $h = 400/2^n$ with corresponding time steps $\Delta t = 5.5h$ for $n = 8$, 9, and 10. Figure 7.6 shows the

crystal growth after time $T = 12.89$ with time step $\Delta t = 3h$. From the top to bottom, the results are obtained with uniform four-fold, uniform k-fold, and adaptive mesh refinement schemes. From these results it is clear that our scheme is stable for time steps $\Delta t \leq O(h)$. And we try to find the maximum Δt corresponding to different spatial grid sizes h so that stable solutions can be computed after 20 time step iterations. The results are shown in Table 7.1 and we obtain stable solutions for all three mesh sizes. Note that there is a linear relation between the time step and mesh sizes. Thus, for finer mesh sizes we may use larger time steps than previous conventional methods.

TABLE 7.1. Stability constraint of Δt for the proposed scheme. There are obtained by 4-fold and k-fold crystal growth in uniform domain.

Mesh size	$h = 400/256$	$h = 400/512$	$h = 400/1024$
Four fold crystal	$\Delta t \leq 20h$	$\Delta t \leq 15h$	$\Delta t \leq 12h$
k-fold crystal	$\Delta t \leq 12h$	$\Delta t \leq 15h$	$\Delta t \leq 12h$



(a) (b) (c)

FIGURE 7.6. The stability of crystal growth with different time steps: (a) $\Delta t = 2.24$ (256×256 mesh), (b) $\Delta t = 1.17$ (512×512 mesh), and (c) $\Delta t = 0.59$ (1024×1024 mesh). From the top to bottom, the results are obtained with uniform four-fold, uniform k-fold, and adaptive mesh refinement schemes.

7.3. Convergence test

To obtain an estimate of the convergence rate, we perform a number of simulations for crystal growth problem on a set of increasingly finer grids. The computational domain is $\Omega = (-100, 100)^2$ and we take $R_0 = 15d_0$, and $\Delta = -0.55$. We take $\epsilon_4 = 0.05$ and $\epsilon_6 = 0.02$ for four-fold and six-fold crystal growth, respectively. The corresponding time step is set as $\Delta t = 0.0625$. The numerical solutions are computed on the uniform grids $h = 200/2^n$ for $n = 8$, 9, 10, and 11. For the adaptive case, we

set the base mesh grids, 64×64 making the minimum grid spacing $h_{max} = 3.125$ and use $l^* = 2,\ 3,\ 4,$ and, 5 to match the uniform case. The calculations are run up to time $T = 30$. Since no analytical solutions are available, we use the Richardson method. We define the error to be the discrete of l_2-norm of the difference between that grid and the average of the next finer grid cells covering it:

$$e_{h/\frac{h}{2}ij} = c_{hij} - (c_{\frac{h}{2}2i-1,2j-1} + c_{\frac{h}{2}2i-1,2j} + c_{\frac{h}{2}2i,2j-1} + c_{\frac{h}{2}2i,2j})/4.$$

The rate of convergence is defined as:

$$\log_2(\parallel e_{h/\frac{h}{2}} \parallel_2 \ / \ \parallel e_{\frac{h}{2}/\frac{h}{4}} \parallel_2).$$

The obtained errors and rates of convergence using these definitions are given in Table 7.2. The second-order accuracy with respect to the space is observed as expected from the discretization.

TABLE 7.2. Error and convergence results with various mesh grids. Here, $\Delta t = 0.0625$. $N_x - 2N_x$ means $\|e^{\frac{200}{N_x}}\|_2$.

Grid	$256-512$	Rate	$512-1024$	Rate	$1024-2048$
Four-fold : l_2-error	$4.505e$-3		$1.264e$-3		$3.110e$-4
Four-fold: Rate		1.83		2.02	
Six-fold : l_2-error	$6.778e$-3		$1.822e$-3		$4.550e$-4
Six-fold : Rate		1.82		2.07	
AMR: l_2-error	$6.123e$-3		$1.883e$-3		$5.044e$-4
AMR: Rate		1.70		1.90	

To obtain the convergence rate for the temporal discretization, we fix the spatial grid as 512 and choose a set of decreasingly smaller time steps $\Delta t = 0.5, 0.25, 0.125,$

and 0.0625. We also run the computation up to time $T = 30$. Note that we similarly define the temporal discrete l_2-norm error as $e_{ij}^{\Delta t} := u_{ij}^{\Delta t} - u_{ij}^{\frac{\Delta t}{2}}$. The obtained errors and rates of convergence are given in Table 7.3. The first-order accuracy with respect to the time is observed, as expected from the discretization.

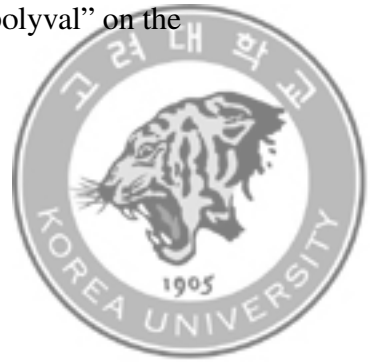
TABLE 7.3. Error and convergence results with various time steps. $0.5 - 0.25$ means $\|e^{\Delta t}\|_2$ with $\Delta t = 0.5$. Here, a 512×512 mesh grid is used.

Δt	$0.5 - 0.25$	Rate	$0.25 - 0.125$	Rate	$0.125 - 0.0625$
Four-fold: l_2-error	$4.389e$-2		$2.241e$-2		$1.117e$-2
Four-fold: Rate		0.969		1.004	
Six-fold: l_2-error	$4.532e$-2		$2.245e$-32		$1.127e$-2
Six-fold: Rate		1.013		0.994	
AMR: l_2-error	$4.823e$-2		$2.464e$-2		$1.251e$-2
AMR: Rate		0.968		0.978	

7.4. Effect of time step and mesh

In this experiment we consider the evolution of the interface with different time steps in order to investigate the effect of time step. A 1024×1024 mesh is used on the domain $\Omega = (-400, 400)^2$ and we take $h = 0.7813$, $R_0 = 14d_0$, and $\Delta = -0.55$. The simulations are run up to time $T = 1800$. Figures. 7.7 (a) and (b) show the position and velocity of the tip versus time respectively, both for different time steps $\Delta t = 0.6$, 0.3, 0.15, and 0.075. Figure 7.7 (c) shows evolutions of the interface with time step $\Delta t = 0.15$ at times $t = 0$, 225, 450, 675, 900, 1125, 1350, 1575, and 1800 (from inside to outside). For different time steps, the interfaces at time $T = 1800$ are shown in Figure 7.7 (d).

Next, we consider the evolution of the interface with different time steps in order to investigate the effect of time step. A 1024×1024 mesh is used on the domain $\Omega = (-200, 200)^2$ with $R_0 = 50d_0$, $\epsilon_6 = 0.02$, and $\Delta = -0.55$. Figure 7.8 shows the velocity of the tip versus at time $T = 1200$ with different time steps $\Delta t = 0.6$, 0.3, and 0.15. The four-fold and six-fold cases are presented in Figure 7.8(a) and (b), respectively. Here, we define the error between the fitting velocity $\widetilde{V}$ and V as $E_i = |\widetilde{V}_i - V_i|/V_i$. The linear fit $\widetilde{V}$ is done using the MATLAB function "polyfit" and the errors on the index i are calculated by the MATLAB function "polyval" on the



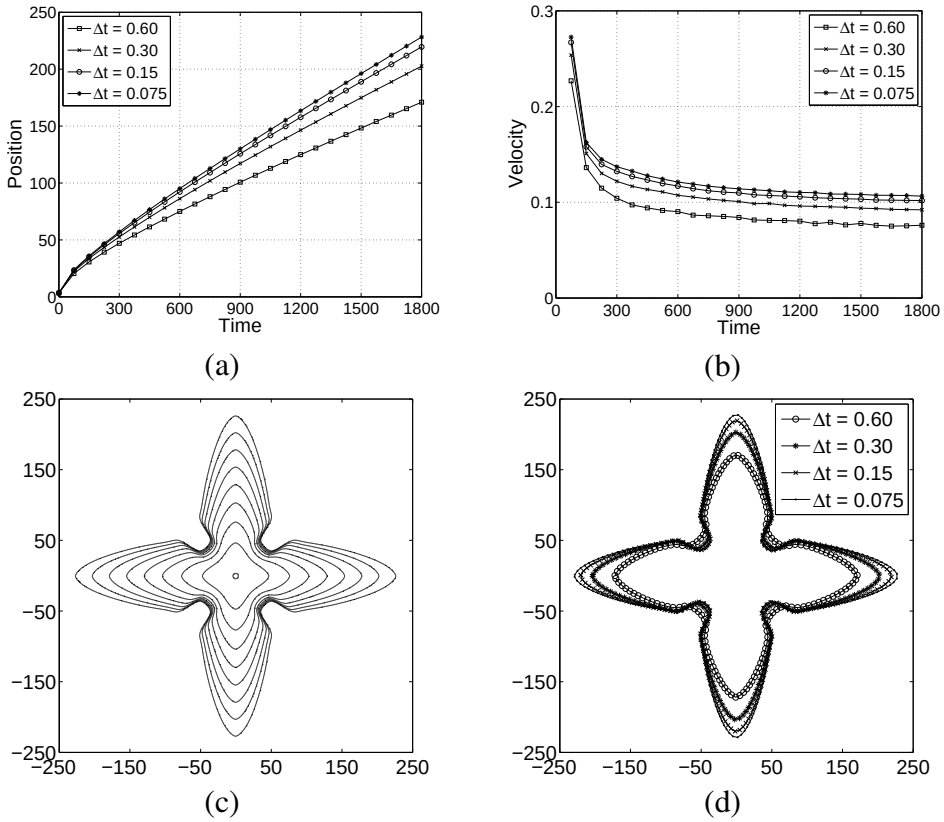
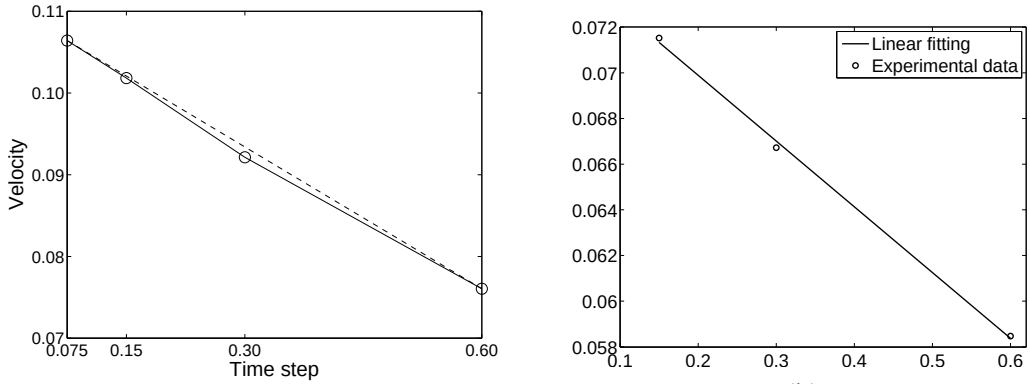


FIGURE 7.7. (a) and (b) show the position and velocity of the tip versus time respectively, for different time steps. (c) Evolutions of the interface with time step $\Delta t = 0.15$. (d) The interfaces at time $T = 1800$ for different time steps.

results of the linear fit. In this test, the l_2 error is 0.54%. Therefore the results suggest that the convergence rate of the tip velocity is linear with respect to the time step.

Total CPU times and average CPU times ($\overline{\text{CPU}}$) of the simulations for different time steps are listed in Table 7.4. The average CPU time is defined as the real computational time (excluding data printing times) divided by the total number of iterations.



(a) (b)

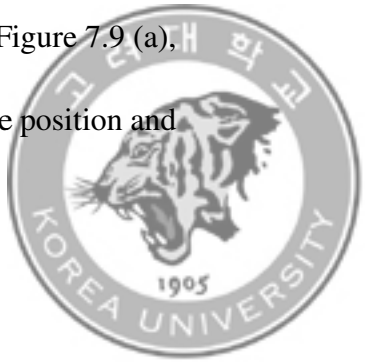
FIGURE 7.8. The numerical experimental and linear fitting velocities versus time step for (a) four-fold and (b) k-fold crystal.

the results are shown in Table 7.4, corresponding to data in Fig. 7.7. Table 7.4 suggests that our proposed scheme is robust for different time steps.

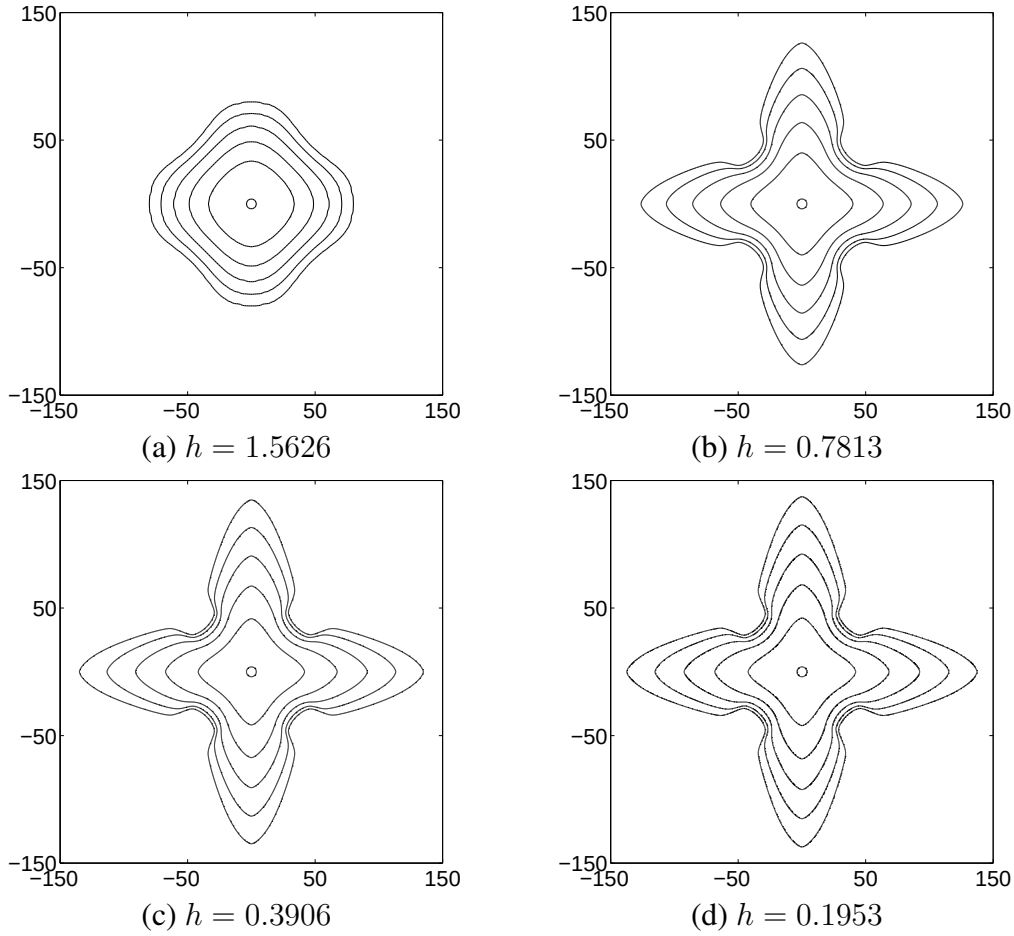
TABLE 7.4. Total CPU times and average CPU times ($\overline{\text{CPU}}$) for different time steps.

Case	$\Delta t = 0.6$	$\Delta t = 0.3$	$\Delta t = 0.15$	$\Delta t = 0.075$
CPU time (h)	5.07	9.06	16.77	32.59
$\overline{\text{CPU}}$ time (s)	5.87	5.19	4.80	4.84

In the next experiment we consider the evolution of the interface with different mesh sizes in order to find an effective mesh size for our proposed method. 256×256, 512×512, 1024×1024, and 2048×2048 meshes are used on the domain $\Omega = (-200, 200)^2$, i.e., we use four different $h = 1.5626$, 0.7813, 0.3906, and 0.1953. The parameters used are $R_0 = 14d_0$, $\Delta = -0.55$, $\Delta t = 0.15$, and $T = 900$. Figure 7.9 (a), (b), (c), and (d) show sequences of interface for different mesh sizes. The position and



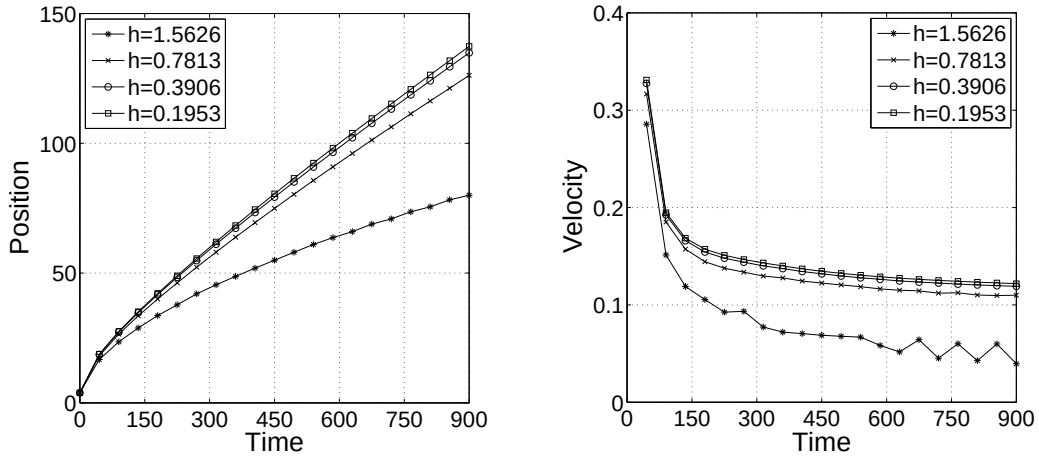
velocity of the tip versus time are shown in Fig. 7.10 (a) and (b), respectively. From the results shown in Fig. 7.10 we can observe that the spatial step size $h = 0.3906$ is enough to simulate accurately and robustly the evolution of crystal growth in our proposed method.



(a) $h = 1.5626$ (b) $h = 0.7813$

(c) $h = 0.3906$ (d) $h = 0.1953$

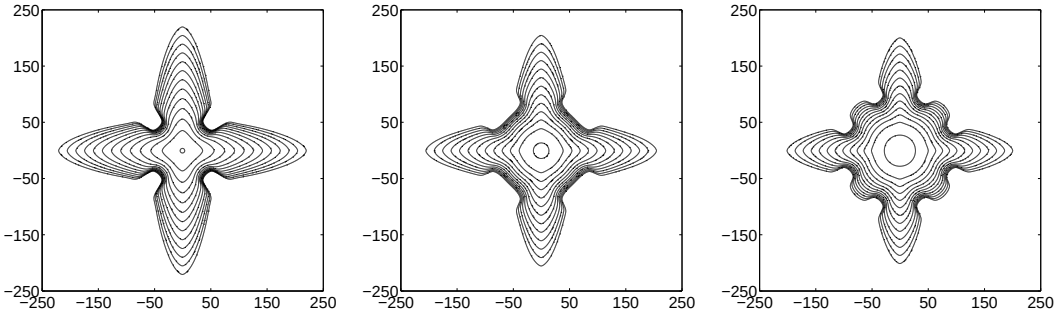
FIGURE 7.9. Sequences of interfaces with different spatial step sizes: (a) $h = 1.5626$, (b) $h = 0.7813$, (c) $h = 0.3906$, and (d) $h = 0.1953$.



(a) Position (b) Velocity

FIGURE 7.10. (a) and (b) show the position and velocity of the tip versus time, respectively.

7.5. Effect of radius



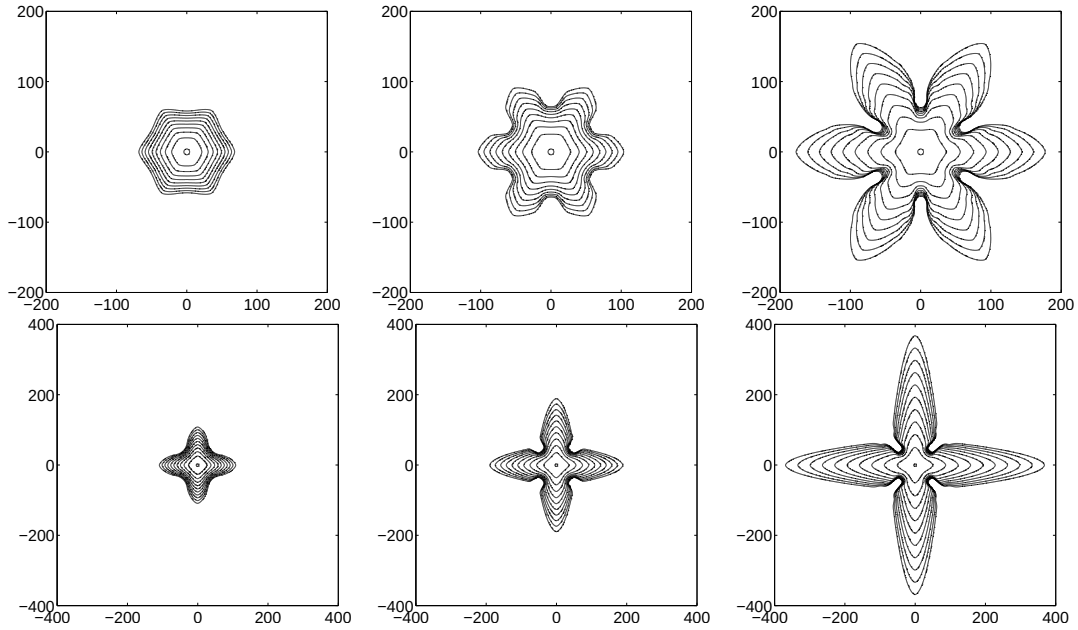
(a) $R_0 = 15d_0$ (b) $R_0 = 50d_0$ (c) $R_0 = 100d_0$

FIGURE 7.11. Sequences of interfaces with different initial radius of dendrite (a)$R_0 = 15d_0$, (b) $R_0 = 50d_0$, and (c) $R_0 = 100d_0$.

In this experiment we investigate the effects of radius of the initial solid seed. For each test a 1024×1024 mesh is used on the domain $\Omega = (-400, 400)^2$ and we choose $\Delta t = 0.15$ and $T = 1500$. Figure 7.11 shows sequences of interfaces with different

radii $R_0 = 15d_0$, $50d_0$, and $100d_0$ (from left to right). we can see that for an increase in the initial radius the dendrite growth faster. In this test we take $\Delta = -0.55$.

7.6. Effect of undercooling



(a) $\Delta = -0.45$ (b) $\Delta = -0.55$ (c) $\Delta = -0.65$

FIGURE 7.12. Sequences of interfaces with different undercooling. (a) $\Delta = -0.45$, (b) $\Delta = -0.55$, and (c) $\Delta = -0.65$. From top and bottom, these are the results for four-fold and six-fold cases.

Now we investigate the effects of undercooling of the initial solid seed. Sequences of interfaces with different undercooling $\Delta = -0.45$, $\Delta = -0.55$, and $\Delta = -0.65$ are presented in the Fig. 7.12. In two dimensional, a 1024×1024 mesh is used on the domain $\Omega = (-200, 200)^2$ and we choose $R_0 = 15d_0$, $\Delta t = 0.3$, and $T = 1500$. In this test we take $\epsilon_4 = 0.05$ and $\epsilon_6 = 0.02$ for four-fold and six-fold crystal, respectively.

From Fig. 7.12 we observe that the large initial undercooling causes the dendrite to grow faster.

7.7. Comparison between our proposed adaptive method, explicit adaptive method, and uniform mesh simulation

7.7.1. Comparison between our proposed adaptive method and explicit adaptive method. In general, an explicit scheme is fast, however, the overall CPU time for long time integration larger than an implicit scheme, which can use larger time steps. This is true for the adaptive method used here. In order to show our proposed method is more efficient than explicit adaptive methods, we consider CPU time comparison test in two dimensions. The computational domain is set as $\Omega = (-200, 200)^2$ with $l^* = 3$ levels. The minimum element size $h_{min} = 0.39$, $R_0 = 15d_0$, and $\Delta = -0.55$ are used. Since the explicit adaptive method surfers the time step size limitation, here we use a time step $\Delta t = 0.01$. For our proposed method, we use time step $\Delta t = 0.01$ and $\Delta t = 0.2$. The calculations are run up to time $T = 200$. We list the tip position of crystal and CPU time in Table 7.5. Form these results, we can observe that our proposed method with $\Delta t = 0.01$ needs more CPU time than the explicit method. However, with $\Delta t = 0.2$ our proposed method is about 5 times faster than the explicit method. Tip positions with different methods are similar.

7.7.2. Comparison with uniform mesh simulation. In this experiment, we will compare the results obtained on uniform meshes and on adaptively refined meshes for

TABLE 7.5. Comparison of CPU time and tip positions calculated by the explicit scheme and our proposed scheme.

Method	Time step	Tip position	CPU time (h)
Explicit method	0.01	45.94	0.10
Proposed method	0.01	45.91	0.60
Proposed method	0.20	44.48	0.02

two and three dimensions to show the efficiency and accuracy of our proposed method. For two dimensions, the computational domain is set as $\Omega = (-400, 400)^2$ with 1024×1024 mesh grids for uniform mesh calculation and $l^* = 3$ levels for the adaptive mesh method. And in three dimensions, we use $\Omega = (-100, 100)^3$, $256 \times 256 \times 256$, and $l^* = 3$. With time step $\Delta t = 0.3$, the calculations are run up to time $T = 1800$ and $T = 240$ in two and three dimensions, respectively. The comparisons with uniform meshes and adaptively refined meshes in two and three dimensions are drawn in the first row and in the second row of Fig. 7.13, respectively. As can be observed, the agreement between the results computed by uniform meshes and adaptive meshes is good. In the two-dimensional calculation, the taken CPU times are $16.15h$ and $0.82h$ for uniform and adaptive meshes, respectively. In the three-dimensional calculation, the taken CPU times are $29.15h$ and $2.71h$ for uniform and adaptive meshes, respectively.

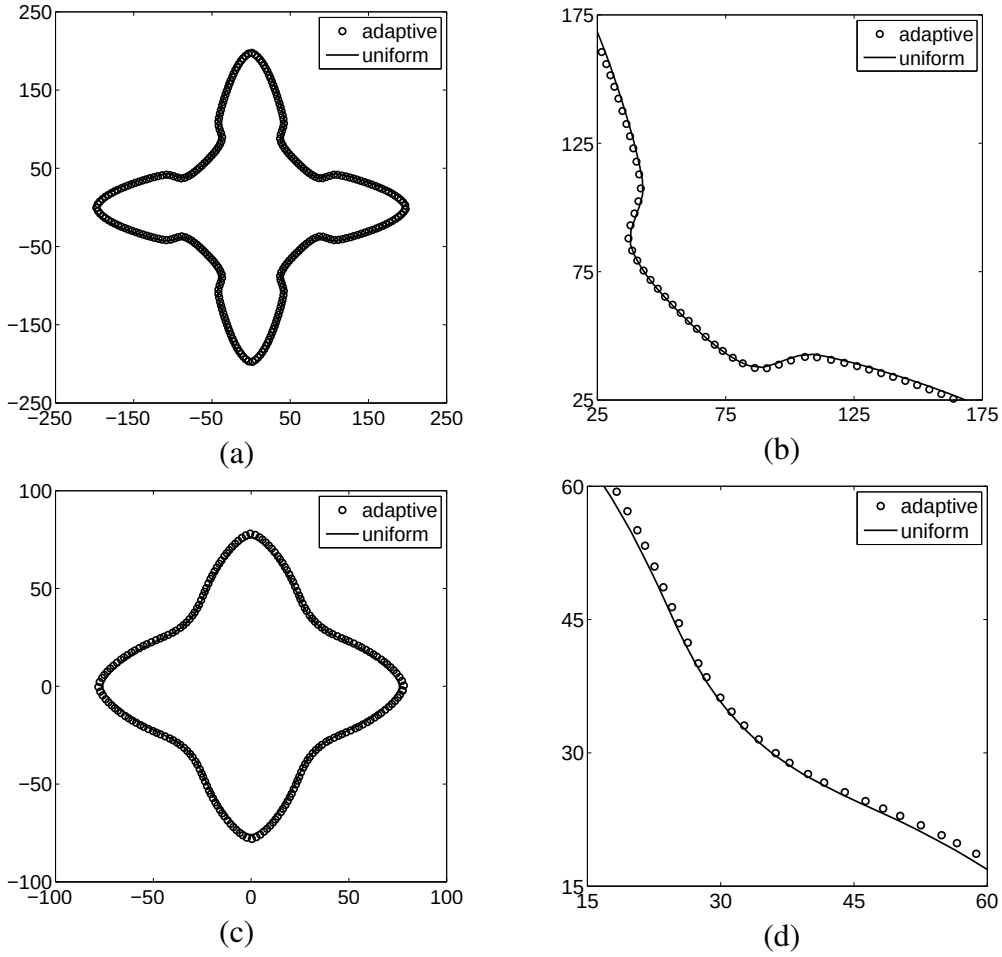


FIGURE 7.13. The comparison between uniform meshes and adaptive meshes in two and three dimensions. First and second rows are two and three dimensional cases, respectively. (a) crystal shape at $t = 1800$. (c) y-z plane of crystal shape at $t = 240$. (b) and (d) are closeup views of (a) and (b), respectively.

7.8. Dendritic growth at low undercooling

For low undercoolings, it requires much longer time to reach a steady-state tip velocity due to the lower growth rate. Furthermore an extremely large domain should be used to avoid the far field boundary effect. In this case, the adaptive mesh refinement

method is a better choice to overcome it. Here, we consider low undercoolings such as $\Delta = -0.25$ and $\Delta = -0.1$. The computational domain is $\Omega = (-6400, 6400)^2$ with the base mesh grids, 32×32. $l^* = 9$ is used and the minimum grid spacing h_{min} is 0.78. With time step $\Delta t = 0.4$, the numerical solutions for $\Delta = -0.25$ and $\Delta = -0.1$ are computed up to $T = 4000$ and $T = 40000$, respectively. Other parameters are $d_0 = 0.403$, $R_0 = 100d_0$, $D = 13$, and $\lambda = 20.744$. Figure 7.14 shows the evolution of tip velocity (V_{tip}) for $\Delta = -0.25$ and $\Delta = -0.1$ with our proposed method, the results in [29], and solvability theory [25, 31]. The numerical results show good agreement with previous results. Next, we consider steady-state tip velocities in three dimensions.

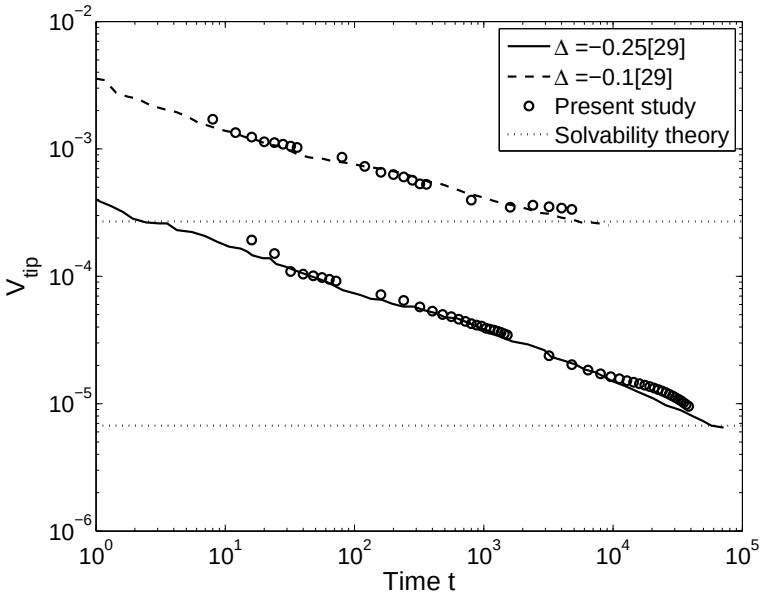


FIGURE 7.14. Evolution of tip velocity at $\Delta = -0.25$ and $\Delta = -0.1$. In order to compare the results in [29] and the results computed by solvability theory, we put them together.

There is the simple relationship which was obtained by Ivantsov [30]:

$$\Delta = -Pe \exp(Pe) \int_{Pe}^{\infty} \frac{\exp(-s)}{s} ds,$$

where $Pe = R_{tip}V_{tip}/(2D)$ is the Peclet number and R_{tip} is the tip radius. The stability constant $\sigma = 2Dd_0/(R_{tip}^2 V_{tip})$ was used in [25, 31]. Here we choose $\sigma = 0.02$. Thus for given Δ, D, and d_0, we can compute V_{tip}. In numerical experiment, we perform the simulation on the domain $\Omega = (-100, 100) \times (-100, 100) \times (0, 800)$ with the base mesh grids, $8 \times 8 \times 32$. Here $l^* = 5$ and the minimum grid spacing $h_{min} = 0.78$ are used. The other parameters are same as those used in two-dimensional space except for $R_0 = 50d_0$. Comparisons with theoretical solutions are drawn in Table 7.6. From these results, we can observe that dimensionless steady-state tip velocities obtained by our proposed scheme are in good agreement with the analytic solutions at low undercoolings.

TABLE 7.6. Comparison of dimensionless steady-state tip velocities calculated by our proposed method and the analytic solution.

Case	$\Delta = -0.25$	$\Delta = -0.1$
Analytic solution	0.00251	0.000139
Numerical solution	0.00252	0.000148

7.9. Accuracy of our proposed method

7.9.1. Comparison of the dimensionless steady-state tip velocities. To verify the accuracy of our proposed scheme we compare the dimensionless steady-state tip velocities obtained by our proposed scheme with phase-field simulations [23] and Green's function calculations. A 1024×1024 mesh is used on the domain $\Omega = (-200, 200)^2$. For the adaptive mesh refinement method, we take with $l^* = 3$, the minimum element size $h_{min} = 0.39$, We choose $R_0 = 6.924$, $W_0 = 1$, and $\lambda = D/a_2$. In Table 7.7, as can be seen the values obtained by our proposed scheme are in good agreement with results of previous phase-field and Green's theory over the whole range of d_0, Δ, and ϵ_4 investigated here. Note that despite the relatively large time step ($\Delta t = 5\Delta t^{\text{KR}} = 0.08$) used in our scheme, the results are almost identical.

TABLE 7.7. Comparison of dimensionless steady-state tip velocities calculated by our proposed scheme ($V_{\text{tip}} = Vd_0/D$), calculated by phase-field simulations ($V_{\text{tip}}^{\text{KR}}$), and calculated by the Green function method ($V_{\text{tip}}^{\text{GF}}$).

Δ	ϵ_4	D	d_0/W_0	$V_{\text{tip}}^{\text{LLK}}$	$V_{\text{tip}}^{\text{KR}}$	$V_{\text{tip}}^{\text{GF}}$	V_{tip}
-0.55	0.05	2	0.277	0.01710	0.01680	0.01700	0.01700
-0.55	0.05	3	0.185	0.01740	0.01750	0.01700	0.01720
-0.55	0.05	4	0.139	0.01720	0.01740	0.01700	0.01710
-0.50	0.05	3	0.185	0.01030	0.01005	0.00985	0.00997
-0.45	0.05	3	0.185	0.00599	0.00557	0.00545	0.00537

7.9.2. Tail morphology. Brener [5] derived a theory of the tail shape of a 3D needle crystal with the assumption that the cross section of a 3D needle crystal should grow as the time dependent 2D growth shapes away from the tip. In [23], Karma and Rappel compared the steady-state growth velocities from simulation and theory derived by Brener.

In this section we compare the velocities calculated by our scheme and those given in [23]. In 2D and 3D simulations, we choose $h = 0.3906$, $R_0 = 14d_0$, $\Delta t = 0.15$, and two different undercoolings $\Delta = -0.65$ and $\Delta = -0.70$. In the 2D test a 1024×1024 mesh is used on the domain $\Omega = (-200, 200)^2$ and the simulation time is $T = 750$. In the 3D test a $256 \times 256 \times 256$ mesh is used on the domain $\Omega = (-50, 50)^3$ and the simulation time is $T = 90$. Results of steady-state growth velocities obtained from 2D and 3D simulations are given in Table 7.8. Our results show good agreement with those of Karma and Rappel.

TABLE 7.8. Results of steady-state growth velocities.

Δ	ϵ_4	V_{2D}	V_{3D}	V_{2D}/V_{3D}	V_{2D}^{KR}/V_{3D}^{KR}	Slope
-0.70	0.0294	0.0353	0.0813	0.434	0.44	0.43
-0.65	0.0294	0.0243	0.0620	0.392	0.39	0.40

7.9.3. Comparison with the previous study. An isotropic finite-difference scheme for simulating 6-fold symmetric dendritic solidification is presented in [28]. The author showed that the stability criterion becomes $\Delta t \leq (3/8)h^2$. But, as we can see in Chapter 6, the time restriction of our proposed method is $\Delta t \sim O(h)$. In order to show the improvement of our proposed method, we use the same numerical parameters as in [28], e.g., $\lambda = 1.7680$, $\epsilon_0 = 1.1312$, $\epsilon_6 = 0.05$, $D = 2$, and $R_0 = 5$. Note that in [28], the author took the step size as $h = 0.4$ in the progressively increased mesh sizes as 500×500 for $0 \leq t \leq 150$, to 800×800 for $150 \leq t \leq 250$, and to 1200×1200 for $250 \leq t \leq 400$. Here we take a 1280×1280 mesh size. This simulation is run up to $T = 400$ with $\Delta t = 0.2$. Our proposed method took about only 5 hours of CPU time, which is drastically reduced faster than the CPU time (1000 hours) in [28].

Chapter 8

Conclusions

In this dissertation, we reviewed our research on overcoming the stability restriction by introducing a fast, robust, and accurate operator splitting method. Then we extended this work by incorporating adaptive mesh refinement.

After giving a brief introduction to the sharp-interface model and phase-field model, we described the crystal growth modeling. Later, we introduced the fast, robust, and accurate operator splitting method for phase-field simulations of crystal growth. And then the description of adaptive mesh refinement method was drawn. Finally we demonstrated stability, robustness, and accuracy of the proposed method by a set of representative numerical experiments.

Appendix

In this Appendix, we present a source code to make mesh plots for adaptive mesh refinement in two- and three- dimensions.

```
%%%%%%%%%%%% Mesh plots for Two Dimensional%%%%%%%%%%%%%
close all;clear;clc
basel=64;%%%% number of points in level 0
level=5+1;%%%%%%%%% l^*=5
mes=basel*2^(level-1);%%%%% refinement radio = 2
baselx=basel;
basely=baselx;
mesx=2^(level-1)*baselx;
mesy=2^(level-1)*basely;
xright=800.0;
yright=xright*basely/baselx;%whole domain(-400,400)*(-400,400)
h=yright/mes;
ns=11;
for ik=1:ns
    clear S
figure(ik)
hold on
ijj=1;
for ip=1:level
    hold
    clear op
str = sprintf('data/out%dplot%d.m',ip-1,ik);% data
op=load(str);
m_level=ip-1;
endd2=0;
ihj=0;
while endd2<size(op,1)

AA=op(endd2+1:endd2+4);
nx=AA(3)-AA(1)+1 ;
ny=AA(4)-AA(2)+1;
A=op(endd2+5:endd2+4+nx*ny);
```

```
[xx,yy]=meshgrid(linspace(AA(1)-1.0,AA(3),nx+1),...
    linspace(AA(2)-1.0,AA(4),ny+1));
m_level=3;
gx = xright*xx(1,:)/(baselx*2^(ip-1));
gy = yright*yy(:,1)/(basely*2^(ip-1));

for i=1:nx+1
 a = line([gx(i),gx(i)],[gy(1),gy(ny+1)],'linewidth',0.15);
set(a,'Color',[0 0 1]*ip/level);
end
for j=1:ny+1
 a = line([gx(1),gx(nx+1)],[gy(j),gy(j)],'linewidth',0.15);
set(a,'Color',[0 0 1 ]*ip/level);
end

for j=1:ny
   for i=1:nx
       for ii=1:mesx/(baselx*2^(ip-1))
           for jj=1:mesy/(basely*2^(ip-1))
     S((mesx/(baselx*2^(ip-1)))*(AA(1)+i-2)+ii,...
     (mesy/(basely*2^(ip-1)))*(AA(2)+j-2)+jj)=A((j-1)*nx+i);
           end
       end
   end
end
endd2=endd2+4+nx*ny;
ihj=ihj+1;
end
end
n=mesx;
x=linspace(0.5/n, xright*(n-0.5)/n,n);
n=mesy;
y=linspace(0.5/n, yright*(n-0.5)/n,n);
[xx,yy]=meshgrid(x,y);
hold on
[cc,hh]=contour(xx',yy',S,[0.5 0.5],'k','linewidth',1.5);
axis image
hold off
box on
 set(gca, 'fontsize',20)
 set(gca,'XTick', (0:200:xright)) ;
 set(gca,'YTick',(0:200:xright));
end
```

```
%%%%%%%%%%%% Mesh plots for three Dimensional%%%%%%%%%%%%%
close all;clear;clc
basel=32;%%%% number of points in level 0
level=4+1;%%%%%%%%% l^*=4
mes=basel*2^(level-1);%%%%%%%%%%%% refinement radio = 2
length=400;% whole domain (-200,200)*(-200,200)*(-200,200)
leftlength=-0.5*length;
rightlength=0.5*length;
h=length/mes;
ns=7;%%%%%%%%%%%%Number of print data
nnx=256;
for ik=1:ns
figure(ik)
hold on
for ip=1:level
    hold
str = sprintf('data/out%dplot%d.txt',ip-1,ik);%%data
op=load(str);
m_level=ip-1;
endd2=0;
ihj=0;
h=length/(basel*2^(ip-1));
xxx2=linspace(0.5*h-length*0.5,...
length*0.5-0.5*h,basel*2^(ip-1));
while endd2<size(op,1)
AA=op(endd2+1:endd2+6)
nx=AA(4)-AA(1)+1;
ny=AA(5)-AA(2)+1;
nz=AA(6)-AA(3)+1;
A=op(endd2+7:endd2+6+nx*ny*nz);
kkk=16;
if (ip>1)

if (xxx2(AA(1))>leftlength&&xxx2(AA(2))>leftlength...
      &&xxx2(AA(3))>leftlength...
      &&xxx2(AA(4))<rightlength&&xxx2(AA(5))...
      <rightlength&&xxx2(AA(6))<rightlength )

aaaa=[AA(1) AA(4)];
bbbb=[AA(2) AA(5)];
cccc=[AA(3) AA(6)];
for ii=1:size(aaaa,2)-1
    for jj=1:size(bbbb,2)-1
```

```
        for kk=1:size(cccc,2)-1
xl=xxx2(aaaa(ii));
yl=xxx2(bbbb(jj));
zl=xxx2(cccc(kk));
xr=xxx2(aaaa(ii+1));
yr=xxx2(bbbb(jj+1));
zr=xxx2(cccc(kk+1));
hold on
line([xl xr],[yl yl],[zl zl],'LineWidth',1)
line([xl xr],[yr yr],[zl zl],'LineWidth',1)
line([xl xr],[yl yl],[zr zr],'LineWidth',1)
line([xl xr],[yr yr],[zr zr],'LineWidth',1)

line([xl xl],[yl yr],[zl zl],'LineWidth',1)
line([xr xr],[yl yr],[zl zl],'LineWidth',1)
line([xl xl],[yl yr],[zr zr],'LineWidth',1)
line([xr xr],[yl yr],[zr zr],'LineWidth',1)

line([xl xl],[yl yl],[zl zr],'LineWidth',1)
line([xl xl],[yr yr],[zl zr],'LineWidth',1)
line([xr xr],[yl yl],[zl zr],'LineWidth',1)
line([xr xr],[yr yr],[zl zr],'LineWidth',1)

        end
    end
end
end
end


for k=1:nz
   for j=1:ny
      for i=1:nx
           for ii=1:mes/(basel*2^m_level)
            for jj=1:mes/(basel*2^m_level)
              for kk=1:mes/(basel*2^m_level)
          if((mes/(basel*2^m_level))...
   *(AA(1)+i-2)+ii<=nnx&&(mes/(basel*2^m_level))...
  *(AA(2)+j-2)+jj<=nnx&&(mes/(basel*2^m_level))...
   *(AA(3)+k-2)+kk<=nnx)
 S((mes/(basel*2^m_level))*(AA(1)+i-2)+ii,...
 (mes/(basel*2^m_level))*(AA(2)+j-2)+jj,...
 (mes/(basel*2^m_level))*(AA(3)+k-2)+kk)...
```

```
  =A(ny*nx*(k-1)+nx*(j-1)+i);
                  end
              end
            end
          end
      end
   end
end
endd2=endd2+6+nx*ny*nz;
ihj=ihj+1
end

end
clear A
clear AA

%%Since initial mesh grid is 512*512*512,
%%%it's difficlut to make mesh plots.
%%% We use 128*128*128 mesh grid to show fiugres

for i=1:nnx/2
    for j=1:nnx/2
        for k=1:nnx/2
S2(i,j,k)=(S(2*i,2*j,2*k)+S(2*i-1,2*j,2*k)...
+S(2*i,2*j-1,2*k)+S(2*i,2*j,2*k-1)+S(2*i-1,2*j-1,2*k)...
+S(2*i,2*j-1,2*k-1)+S(2*i-1,2*j,2*k-1)+S(2*i-1,2*j,2*k-1))/8;
        end
    end
end
    clear S
for i=1:nnx/2
    for j=1:nnx/2
        for k=1:nnx/2
    S2(i+nnx/2,j,k)=S2(nnx/2+1-i,j,k);
        end
   end
end
for i=1:nnx
    for j=1:nnx/2
        for k=1:nnx/2
    S2(i,nnx/2+j,k)=S2(i,nnx/2+1-j,k);
        end
   end
```

```
end
for i=1:nnx
    for j=1:nnx
        for k=1:nnx/2
    S2(i,j,nnx/2+k)=S2(i,j,nnx/2+1-k);
        end
   end
end


for i=1:nnx/2
    for j=1:nnx/2
        for k=1:nnx/2
S(i,j,k)=0.125*(S2(2*i,2*j,2*k)+S2(2*i-1,2*j,2*k)...
+S2(2*i,2*j-1,2*k)+S2(2*i,2*j,2*k-1)+S2(2*i-1,2*j-1,2*k)...
+S2(2*i,2*j-1,2*k-1)+S2(2*i-1,2*j,2*k-1)+S2(2*i-1,2*j,2*k-1));
        end
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=mes;
S(1,1,1)=-0.01;
S(end,end,end)=-0.01;
x=linspace(0.5*h-0.5*length, 0.5*length-0.5*h,nnx/2);
[xx,yy,zz]=meshgrid(x,x,x);
 p=patch(isosurface(xx,yy,zz,S,0.0));
 clear xx;
 clear yy;
 clear zz;
set(p,'FaceColor','yellow','EdgeColor','none');
daspect([1 1 1]); axis tight;
view(-31,18);
camlight;
lighting phong;
axis image
axis([-0.5*length 0.5*length -0.5*length ...
        0.5*length -0.5*length 0.5*length])
set(gca,'XTick', (-0.5*length:500:0.5*length)) ;
set(gca,'YTick',(-0.5*length:500:0.5*length)) ;
set(gca,'ZTick', (-0.5*length:500:0.5*length)) ;
set(gca, 'fontsize',18)
hold off
end
```

Bibliography

[1] S.M. Allen, J.W. Cahn, A microscopic theory for antiphase boundary motion and its application to antiphase domain coarsening, Acta Metall., 27 (1979) 1085–1095.

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

[3] N. Al-Rawahi, G. Tryggvason, Numerical simulation of dendritic solidification with convection: two-dimensional geometry, J. Comput. Phys. 180 (2002) 471–496.

[4] M. Beneš, V. Chalupecký, K. Mikula, Geometrical image segmentation by the Allen-Cahn equation, Appl. Numer. Math. 51 (2004) 187–205.

[5] E. Brener, Needle-crystal solution in three-dimensional dendritic growth, Phys. Rev. Lett. 71 (1993) 3653–3656.

[6] W.L. Briggs, A Multigrid Tutorial, SIAM, Philadelphia, 1987.

[7] W.K. Burton, N. Cabrera, and F.C. Frank, The growth of crystals and the equilibrium structure of their surfaces, Philosophical Transactions of the Royal Society A, 243 (1951), 299–358.

[8] G. Caginalp, Stefan and Hele-Shaw type models as asymptotic limits of the phase-field equations, Phys. Rev. A 39 (1989) 5887–5896.

[9] S. Chen, B. Merriman, S. Osher, P. Smereka, A simple level set method for solving Stefan problem, J. Comput. Phys. 135 (1997) 8–29.

[10] C.C. Chen, C.W. Lan, Efficient adaptive three-dimensional phase-field simulation of dendritic crystal growth from various supercoolings using rescaling, J. Cryst. Growth 311 (2009) 702–706.

[11] C.C. Chen, Y.L. Tsai, C.W. Lan, Adaptive phase field simulation of dendritic crystal growth in a forced flow: 2D vs. 3D morphologies, Int. J. Heat Mass Transfer 52 (2009) 1158–1166.

[12] C. Cowan, M.S. Thesis, Simon Fraser University, Canada, 2005.

[13] J.-M. Debierre, A. Karma, F. Celestini, R. Guérin, Phase-field approach for faceted solidification, Phys. Rev. E 68 (2003) 041604.

[14] J.A. Dobrosotskaya, A.L. Bertozzi, A Wavelet-Laplace variational technique for image deconvolution and inpainting, IEEE. Trans. Imag. Proc. 17 (2008) 657–663.

[15] F.C. Frank, Metal Surfaces, ASM, Cleveland, OH, 1963.

[16] P.C. Fife, Models for phase separation and their mathematics, E. J. Diff. Eqns., 2000 (48) (2000) 1–26.

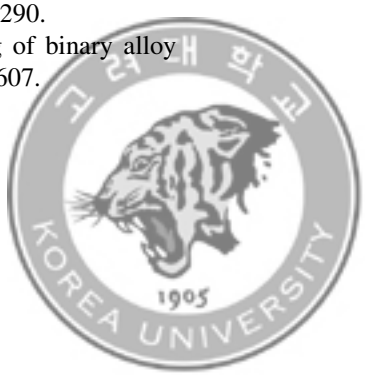
[17] F. Gibou, R. Fedkiw, R. Caflisch, S. Osher, A level set approach for the numerical simulation of dendritic growth, J. Sci. Comput. 19 (2002) 183–199.

[18] T. Ihle, Competition between kinetic and surface tension anisotropy in dendritic growth, Eur. Phys. J. B 16 (2000) 337–344.

[19] J.-H. Jeong, N. Goldenfeld, J.A. Dantzig, Phase field model for three-dimensional dendritic growth with fluid flow, Phys. Rev. E 64 (2001) 041602.

[20] A. Jacot, M. Rappaz, A pseudo-front tracking technique for the modelling of solidification microstructures in multi-component alloys, Acta Mater. 50 (2002) 1909–1926.

[21] D. Juric, G. Tryggvason, A front-tracking method for dendritic solidification, J. Comput. Phys. 123 (1996) 127–148.
[22] A. Karma, Y.H. Lee, M. Plapp, Three-dimensional dendrite-tip morphology at low undercooling, Phys. Rev. E 61 (2000) 3996–4006.
[23] A. Karma, W.-J. Rappel, Quantitative phase-field modeling of dendritic growth in two and three dimensions, Phys. Rev. E 57 (1998) 4323–4349.
[24] A. Karma, W.-J. Rappel, Phase-field method for computationally efficient modeling of solidification with arbitrary interface kinetics, Phys. Rev. E 53 (1996) 3017–3020.
[25] D. Kessler, J. Koplik, H. Levine, Pattern selection in fingered growth phenomena, Adv. Phys. 37 (1988) 255–339.
[26] Y.-T. Kim, N. Goldenfeld, J. Dantzig, Computation of dendritic microstructures using a level set method, Phys. Rev. E 62 (2000) 2471-2474.
[27] R. Kobayashi, Modeling and numerical simulations of dendritic crystal growth, Phys. D 63 (1993) 410-423.
[28] A. Kumar, Isotropic finite-differences, J. Comput. Phys. 201 (2004) 109–118.
[29] C.W. Lan, C.M. Hsu, C.C. Liu, Y.C. Chang, Adaptive phase field simulation of dendritic growth in a forced flow at various supercoolings, Phys. Rev. E 65 (2002) 061601.
[30] J.S. Langer, In: Directions in Condensed Matter, World Scientific, Singapore, 1986, pp. 164–186.
[31] J.S. Langer, Instabilities and pattern formation in crystal growth, Rev. Mod. Phys. 52 (1980) 1–28.
[32] D. Li, R. Li, P. Zhang, A cellular automaton technique for modelling of a binary dendritic growth with convection, Appl. Math. Modelling 31 (2007) 971–982.
[33] S. Li, J.S. Lowengrub, P.H. Leo, Nonlinear morphological control of growing crystals, Phys. D 208 (2005) 209–219.
[34] Y. Li, J. Kim, An unconditionally stable numerical method for bimodal image segmentation, Applied Mathematics and Computation, 219 (2012) 3083–3090.
[35] Y. Li, J. Kim, Multiphase image segmentation using a phase-field model, Computers and Mathematics with Applications, 62 (2011) 737–745.
[36] Y. Li, J. Kim, A fast and accurate numerical method for medical image segmentation, J. KSIAM, 14 (2010) 201-2-10.
[37] Y. Li, J. Kim, Phase-field simulations of crystal growth with adaptive mesh refinement, International Journal of Heat and Mass Transfer, 55 (2012) 7926–7932.
[38] Y. Li, H-G. Lee, J. Kim, A fast, robust, and accurate operator splitting method for phase-field simulations of crystal growth, Journal of Crystal Growth, 321 (2011) 176–182.
[39] Y. Li, D. Lee, H. G. Lee, D. Jeong, C. Lee, D. Yang, J. Kim, A robust and accurate phase-field simulation of snow crystal growth, J. KSIAM, 16 (2012) 15–29.
[40] D.I. Meiron, Boundary integral formulation of the two-dimensional symmetric model of dendritic growth, Phys. D 23 (1986) 329–339.
[41] M. J. Berger, I. Rigoutsos. An algorithm for point clustering and grid generation. IEEE Transactions Systems, Man and Cybernetics, 21 (1991) 1278–1286.
[42] M. Plapp, A. Karma, Multiscale finite-difference-diffusion-Monte-Carlo method for simulating dendritic solidification, J. Comput. Phys. 165 (2000) 592-619.
[43] N. Provatas, N. Goldenfeld, J. Dantzig, Efficient computation of dendritic microstructures using adaptive mesh refinement, Phys. Rev. Lett. 80 (1998) 3308-3311.
[44] N. Provatas, N. Goldenfeld, J. Dantzig, Adaptive mesh refinement computation of solidification microstructures using dynamic data structures, J. Comput. Phys. 148 (1999) 265-290.
[45] J.C. Ramirez, C. Beckermann, A. Karma, H.-J. Diepers, Phase-field modeling of binary alloy solidification with coupled heat and solute diffusion, Phys. Rev. E 69 (2004) 051607.



[46] J. Rosam, P.K. Jimack, A. Mullis, A fully implicit, fully adaptive time and space discretisation method for phase-field simulation of binary alloy solidification, J. Comput. Phys. 225 (2007) 1271-1287.
[47] J.A. Sethian, J. Strain, Crystal growth and dendlritic solidification, J. Comput. Phys. 98 (1992) 231-253.
[48] C.J. Shih, M.H. Lee, C.W. Lan, A simple approach toward quantitative phase field simulation for dilute-alloy solidification, J. Cryst. Growth 282 (2005) 515-524.
[49] T.P. Schulze, Simulation of dendritic growth into an undercooled melt using kinetic Monte Carlo techniques, Phys. Rev. E 78 (2008) 020601.
[50] M. Sussman, A.S. Almgren, J.B. Bell, P. Colella, L.H. Howell, M.L. Welcome, An adaptive level set approach for incompressible two-phase flows, J. Comput. Phys. 148 (1999) 81–124.
[51] J. Strain, A boundary integral approach to unstable solidification, J. Comput. Phys. 85 (1989) 342-389.
[52] A.M. Stuart, A.R. Humphries, Dynamical Systems and Numerical Analysis, Cambridge University Press, New York, 1998.
[53] U. Trottenberg, C. Oosterlee, A. Schüller, Multigrid, Academic Press, USA, 2001.
[54] X. Tong, C. Beckermann, A. Karma, Q. Li, Phase-field simulations of dendritic crystal growth in a forced flow, Phys. Rev. E 63 (2001) 061601.
[55] G. Tryggvason, B. Bunner, A. Esmaeeli, D. Juric, N. Al-Rawahi, W. Tauber, J. Han, S. Nas, Y.-J. Jan, A front-tracking method for the computations of multiphase flow, J. Comput. Phys. 169 (2001) 708-759.
[56] K. Wang, A. Chang, L.V. Kale, J.A. Dantzig, Parallelization of a level set method for simulating dendritic growth, J. Parallel Distrib. Comput. 66 (2006) 1379-1386.
[57] S.-L. Wang, R.F. Sekerka, Algorithms for phase field computation of the dendritic operating state at large supercoolings, J. Comput. Phys. 127 (1996) 110-117.
[58] J.A. Warren, W.J. Boettinger, Prediction of dendritic growth and microsegregation patterns in a binary alloy using the phase-field method, Acta Metall. Mater. 43 (1995) 689-703.
[59] G. Wulff, Zur frage der geschwindigkeit des wachsturms under auflosung der kristallflachen, Z Kristallogr, 34 (1901), 449–530.
[60] Y. Xu, J.M. McDonough, K.A. Tagavi, A numerical procedure for solving 2D phase-field model problems, J. Comput. Phys. 218 (2006) 770-793.
[61] H. Yin, S.D. Felicelli, A cellular automaton model for dendrite growth in magnesium alloy AZ91, Modelling Simul. Mater. Sci. Eng. 17 (2009) 075011.
[62] P. Zhao, J.C. Heinrich, D.R. Poirier, Fixed mesh front-tracking methodology for finite element simulations, Int. J. Numer. Meth. Engng. 61 (2004) 928-948.
[63] M.F. Zhu, C.P. Hong, A modified cellular automaton model for the simulation of dendritic growth in solidification of alloys, ISIJ Int. 41 (2001) 436-445.
[64] M.F. Zhu, S.Y. Lee, C.P. Hong, Modified cellular automaton model for the prediction of dendritic growth with melt convection, Phys. Rev. E 69 (2004) 061610.