Available online at www.sciencedirect.com
ScienceDirect
Journal of Computational Physics 227 (2008) 5984–6000
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

Efficient computation of dendritic growth with *r*-adaptive finite element methods

Heyu Wang [a,b], Ruo Li [c], Tao Tang [b,*]

[a] *Department of Mathematics, Zhejiang University, 310027 Hangzhou, China*
[b] *Department of Mathematics, Hong Kong Baptist University, Kowloon Tong, Kowloon, Hong Kong*
[c] *LMAM and School of Mathematical Sciences, Peking University, 100871 Beijing, China*

Received 31 July 2007; received in revised form 2 January 2008; accepted 1 February 2008
Available online 4 March 2008

Abstract

This paper deals with the application of a moving grid method to the solution of a phase-field model for dendritic growth in two- and three-dimensions. A mesh is found as the solution of an optimization problem that automatically includes the boundary conditions and is solved using a multi-grid approach. The governing equations are discretized in space by linear finite elements and a split time-level scheme is used to numerically integrate in time. One novel aspect of the method is the choice of a regularized monitor function. The moving grid method enables us to obtain accurate numerical solutions with much less degree of freedoms. It is demonstrated numerically that the tip velocity obtained by our method is in good agreement with the previously published results.

AMS: 65M20; 65N22; 80A22

Keywords: Moving mesh method; Phase-field model; Finite element method; Dendritic growth

1. Introduction

Dendritic growth has been a central problem in pattern formation and metallurgy [14,18,21]. Since the microscopic properties of such procedure are determined by the length scale of dendrites, understanding the mechanism of pattern selection during solidification procedure is the main concern of the theoretical [16,17,19,21] and experimental [10,12] studies.

With the theoretical development for mathematical models of solidification, numerical simulation has been demonstrated as one of the powerful tools for investigating dendritic growth. The Stefan model [11] describes the solidification of a pure material from its melt, which gives rise to dendrite patterns. However, this model is

* Corresponding author. Tel.: +852 2339 5148; fax: +852 2339 5811.
E-mail addresses: hywang@math.hkbu.edu.hk (H. Wang), rli@math.pku.edu.cn (R. Li), ttang@math.hkbu.edu.hk (T. Tang).

0021-9991/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2008.02.016

not computationally tractable, since it requires front tracking and lattice deformation to contain the interface at predefined locations on the grids [1,32]. To avoid the sharp interface tracking, the phase-field model was then considered to approximate the Stefan model by introducing a phase-field variable ψ which varies rapidly but smoothly from one value in the liquid phase to another value in the solid phase across a spatially diffuse interface region [5,20].

In [5], Caginalp et al. showed rigorously that the phase-field model converges to the sharp interface limit only when the interface width is much smaller than the capillary length d_0. Consequently, the grid spacing in numerical simulations should be much smaller than d_0 if we wish to obtain numerical solutions that converge to the sharp interface limit. This limitation makes it very difficult to simulate realistic large scale problems. Moreover, the interface kinetics β in the phase-field model should be big enough to ensure the convergence due to the Gibbs–Thomson boundary condition. However, most experiments performed at low undercooling were in the limit of zero interface kinetics [10,12,29], and it is physically important to simulate solidification micro structures in this limit. As a result, the application of the phase-field model to free boundary problems is severely restricted. On the other hand, Karma et al. [15] presented a quantitative phase-field model by introducing a different asymptotic analysis in powers of the ratio of the interface width to the diffusion length, which revealed that the phase-field approach can be extended to the case of arbitrarily small or even zero interface kinetics. Based on the new analysis, it is allowed to select certain parameters so that the phase-field model corresponds to the sharp interface limit when the interface width is of the order of the capillary length. Therefore, it seems possible to simulate the dendritic growth in a macro domain with physically meaningful dimensions. In spite of the emergent possibility, such numerical simulations are still extremely expensive in computational time.

It is noticed that the dimensionless thermal field u, which is the other variable in the phase-field model, varies mildly in the whole domain and that ψ keeps the two different constant values in either phase and transits from one value to another value rapidly in a thin diffuse interface region. As a result, it is attractive to utilize some adaptive mesh techniques to improve the numerical efficiency. The h-adaptive method and the r-adaptive method are two of the main classes in the mesh adaptive methods. In a pioneering work, Provatas et al. presented in [29,30] an h-adaptive finite element method for the quantitative phase-field model simulation in large 2D domains. Their adaptive solutions are in good agreement with the results of Karma et al. [14]. It is pointed out that the dimension of the computational domain affects the numerical results, and a larger computational domain is often preferred to obtain more reasonable approximations. This also implies the need of designing more efficient adaptive mesh methods.

The r-adaptive method for phase-field problems has been studied recently. In [2,26], a simple moving mesh strategy was proposed for simulating the phase-change problems, see also [33,36]. In [9], a moving mesh method combining with the spectral method was developed for solving a phase-field bending elasticity model. Inspired by the efficiency improvement in the moving mesh algorithm, we presented in this paper a moving mesh finite element algorithm for the quantitative phase-field equations provided by Karma et al. [15]. Since the phase-field equations given in [15] are very complicated, it is difficult to solve the governing equations in the logical domain. Consequently, we solved the governing equations in the physical domain, and then redistributed the meshes in the physical domain using a strategy proposed in [6] where a nonlinear multi-grid speedup for solving the mesh equations was developed. The main goal is to simulate the dendritic growth problem in three space dimensions.

For simulating the dendritic growth, designing a suitable monitor function turns out to be an important factor for enhancing the efficiency and quality of the moving mesh computations. For example, the initial solidification seed can be nonzero in an extremely small part of the solution domain, which is very difficult to be represented by a commonly adopted monitor function. Based on the dynamical interactions between the phase-field variable and the thermal field variable, we proposed in this paper a monitor function particularly useful for the dendritic growth. This monitor function was improved by introducing a gradually varying regularization field instead of using a constant regularization. It was shown that with the new monitor function a reasonable number of grid points can be clustered around the sharp layers in the phase-field, while the slow variation of the thermal field variable can still be approximated with good accuracy. With the numerical experiments, we validated our method in 2D case quantitatively, but with remarkable efficiency improvement. The qualitative 2D and 3D simulations for quite complex dendritic structures were carried out by using fairly

small number of degree of freedoms (DOFs). The moving mesh algorithm provided highly anisotropic grids with good quality in the simulations, demonstrating the robustness of the algorithm and the implementation.

The layout of this paper is as follows. In Section 2, we will briefly outline the mesh redistribution algorithm and its multi-level speedup. In Section 3, we will present the quantitative phase-field equations for the dendritic growth, together with a finite element scheme on a fixed grid. In Section 4, we will design suitable monitor functions useful for solving the phase-field equations. In Section 5, the numerical tests will be presented to demonstrate the numerical efficiency of our moving mesh method. The concluding remarks will be given in the last section.

2. Mesh redistribution and multi-level speedup

We briefly outline in this section the moving mesh strategy of [6]. In particular, we will concentrate on the multi-level speedup technique in solving the nonlinear algebraic system arising from the discretization of the mesh redistribution equations.

2.1. Mesh redistribution algorithm

Suppose the physical domain Ω and the computational domain Ω_c are two compact Riemannian manifolds of dimension s with metric tensors G_{ij} and $g_{\alpha\beta}$ in some local coordinates $\vec{x}$ and $\vec{\xi}$, respectively. In [23], a mesh redistribution was defined by a one-to-one mapping $\vec{\xi} = \vec{\xi}(\vec{x})$ from Ω to Ω_c. Such a mapping can be achieved by solving the Euler–Lagrange equations with Euclidean metric for the logical domain Ω_c (see, e.g. [3]). The Euler–Lagrange equations are given by

$$\frac{\partial}{\partial x^i}\left(G^{ij}\frac{\partial \xi^k}{\partial x^j}\right) = 0, \tag{2.1}$$

where $m = (G^{ij})^{-1}$ is the so-called monitor function.

To make the set of the candidate mappings from the physical domain to the logical domain as large as possible, an optimization problem

$$\begin{aligned} &\min \quad E(\vec{\xi}) \\ &\text{s.t.} \quad \vec{\xi}|_{\partial\Omega} = \vec{\xi}_b \in \mathbf{K} \end{aligned} \tag{2.2}$$

was introduced in [24], where

$$E(\vec{\xi}) = \sum_k \int_\Omega G^{ij}\frac{\partial \xi^k}{\partial x^i}\frac{\partial \xi^k}{\partial x^j}\mathrm{d}\vec{x} \tag{2.3}$$

is the mesh energy and $\mathbf{K}$ is the admissible set for the boundary mappings. Here we consider only the case that Ω and Ω_c are two polyhedra, with the same structure on their boundary. The admissible set $\mathbf{K}$ includes all the possible mappings from $\partial\Omega$ to $\partial\Omega_c$, which is to map the vertices, edges and faces of the polyhedron Ω to the corresponding ones of the polyhedron Ω_c, respectively. Fig. 2.1 illustrates the boundary mapping in the 2D case.

The method for solving (2.2) redistributes the interior and boundary grids simultaneously, which turns out to be dimension independent. We give the diagram of the mesh redistribution algorithm provided by [24] in Fig. 2.2, where $\vec{x}^{(n)}$, $\vec{\xi}^{(n)}$ and $u_h^{(n)}$ denote the local coordinates on Ω, the local coordinates on Ω_c and numerical solutions of PDEs under consideration, respectively, at $t = t^n$. The initial mesh $\vec{\xi}^{(0)}$ on Ω_c is generated by solving the following optimization problem:

$$\begin{aligned} &\min_{\vec{\xi}} \quad \sum_k \int_\Omega \sum_i \left(\frac{\partial \xi^k}{\partial x^i}\right)^2 \mathrm{d}\vec{x} \\ &\text{s.t.} \quad \vec{\xi}|_{\partial\Omega} = \vec{\xi}_b \in \mathbf{K}. \end{aligned} \tag{2.4}$$

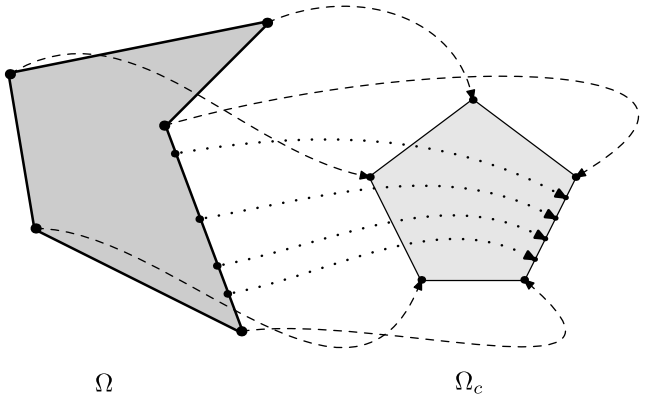

Fig. 2.1. The boundary mapping in the 2D case. The vertices and the edges in the physical domain are mapped to the corresponding ones of the logical domain, respectively.

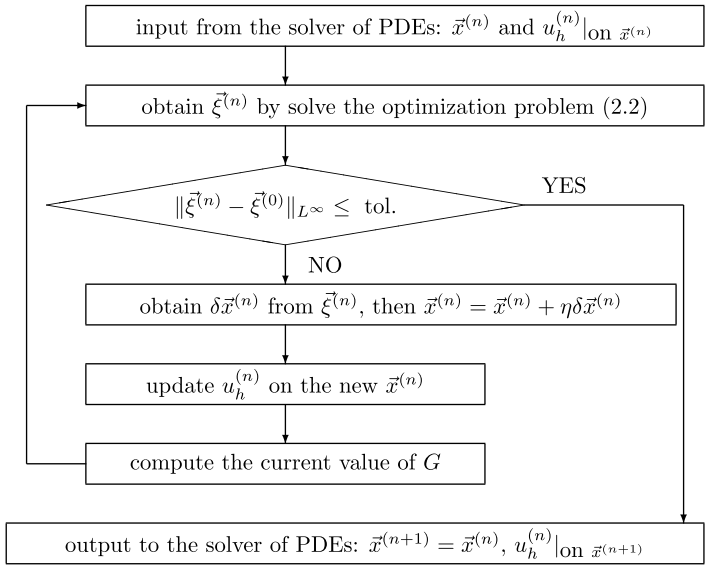

Fig. 2.2. Mesh redistribution algorithm.

In Fig. 2.2, $\eta \in [0, 1]$ is a parameter used to prevent mesh tangling. After $\vec{\xi}^{(n)}$ is obtained, the Jacobi matrix of the mapping $\vec{\xi}^{(n)}$ to $\vec{x}^{(n)}$ can be computed by using the corresponding nodes in Ω_c and Ω. Consequently, the displacement $\delta\vec{x}$ in Fig. (2.2) can be obtained by using $\delta\vec{\xi}$ and the Jacobi matrix. It is noted that the values of $\vec{\xi}^{(n)}$, $\vec{x}^{(n)}$ and $u_h^{(n)}$ are updated in each iteration step of the inner loop. We will introduce a multi-level speedup for solving the minimization problem (2.2) in the next section. The basic speedup idea was discussed in [6]. Some details of the mesh redistribution algorithm can be found in [24] for 2D problems and [7,8] for problems on a 3D sphere.

2.2. Multi-level speedup for the optimization problem

The numerical solution of the minimization problem (2.2) will lead to a large nonlinear algebraic optimization problem. Let us discretize (2.2) in the linear finite element space. The triangulation of the physical domain is $\mathcal{T}$, with $\mathcal{T}_i$ as its elements, and $\vec{X}_i$ as its nodes. The corresponding triangulation on the computational domain is $\mathcal{T}_c$, with $\mathcal{T}_{i,c}$ as its elements, and $\vec{\Xi}_i$ as its nodes. The linear finite element space on the mesh is denoted by $H_h^1(\Omega)$. If the basis function on the node $\vec{X}_i$ is denoted by $\vec{\phi}^i$, then $\vec{\xi}$ can be approximated by $\xi_i\vec{\phi}^i$

(here the standard summation convention is assumed). The coordinates of $\vec{X}_i$ are denoted as $(X_i^1 X_i^2 X_i^3)^{\mathrm{T}}$. Let the inner nodes be indexed from 1 to N_{inner} and the boundary nodes be indexed from $N_{\text{inner}}+1$ to N. The coordinates of the nodes $\vec{X}_i$ in the computational domain are denoted by $(\Xi_i^1 \Xi_i^2 \Xi_i^3)^{\mathrm{T}}$. Denote $X = (\vec{X}^1 \vec{X}^2 \vec{X}^3)^{\mathrm{T}}$, $\Xi = (\vec{\Xi}^1 \vec{\Xi}^2 \vec{\Xi}^3)^{\mathrm{T}}$, where $\vec{X}^k = (X_1^k \cdots X_N^k)^{\mathrm{T}}$, $\vec{\Xi}^k = (\Xi_1^k \cdots \Xi_N^k)^{\mathrm{T}}$, $k = 1, 2, 3$.

To approximate (2.2), a quadratic algebraic optimization problem for Ξ will be formed to determine the motion of the computational grids. Denote

$$H = \left(\int_\Omega G^{ij} \frac{\partial \phi^\alpha}{\partial x^i} \frac{\partial \phi^\beta}{\partial x^j} \mathrm{d}\vec{x} \right)_{1 \leqslant \alpha, \beta \leqslant N}. \tag{2.5}$$

Assume that the constraint leads to a linear system of the form

$$\sum_k A_k \vec{\Xi}^k = \vec{b},$$

or equivalently,

$$\sum_k A_{k,\text{inner}} \vec{\Xi}^k_{\text{inner}} + \sum_k A_{k,\text{bound}} \vec{\Xi}^k_{\text{bound}} = \vec{b}.$$

Remark 2.1. In general, the constraint can not be discretized into a linear system. It is assumed to be a linear one for simplicity but without loss of generality. Certain restrictions on the first-order derivatives of the boundary mapping should be applied to guarantee the existence of the mapping. For more details, see [24].

Notice that $A_{k,\text{inner}} = 0$ and the matrices $A_{k,\text{bound}}$ are the entries of the unit normal of the domain boundary. Thus the optimization problem (2.2) is discretized into the following quadratic optimization problem with linear constraints:

$$\min_{\Xi} \quad \sum_k \left\{ \vec{\Xi}^{k,\mathrm{T}} H \vec{\Xi}^k \right\} \tag{2.6}$$

$$\text{s.t.} \quad \sum_k A_k \vec{\Xi}^k = \vec{b}. \tag{2.7}$$

Since the solution of the resulting system is used just for updating the grid location, the accuracy of (2.6) and (2.7) is not of high priority. This allows us to use some efficient approximation method for obtaining the solutions of (2.6) and (2.7). Recently, a fast solution algorithm was developed using a multi-level speedup technique [6]. In Fig. 2.3 we give the diagram for solving $H\vec{\Xi}^k = \vec{f}$ with the constraints (2.7) using a multi-level iteration technique [4].

The diagram performed one V-cycle iteration of the n-level algorithm with v_1 pre- and v_2 post-smoothing operations. Suppose that a coarse set of points that forms a subset of the fine DOFs has been chosen with the strategy discussed below. Then, the fine-level DOFs can be represented as

$$C_0 = \{1, 2, \ldots, N\}, \quad C_{l-1} = C_l \cup F_l,$$

where C_l is the set of coarse-level points and F_l is the set of remaining fine-level points (so $C_l \cap F_l = \emptyset$). Due to the constraints on the boundary, the grid points in the fine mesh are chosen to be the points of the coarse mesh, with sorted priority based on the constraints applied on it. The points with top-priority are the ones relevant to the vertices of the domain polyhedron, followed by the points on the edges, then by the points on the faces. The interior points are at the lowest priority. To describe the relation between the coarse-grid and the fine-grid, we introduced "neighborhood" notation B_i^l including the points strongly connected with the point $\vec{\Xi}_i$ and

$$B_i^l \cap B_j^l = \emptyset \quad \text{if } i \neq j, \quad \bigcup_{i=0}^{N_l} B_i^l = C_{l-1},$$

where N_l is the number of the set C_l.

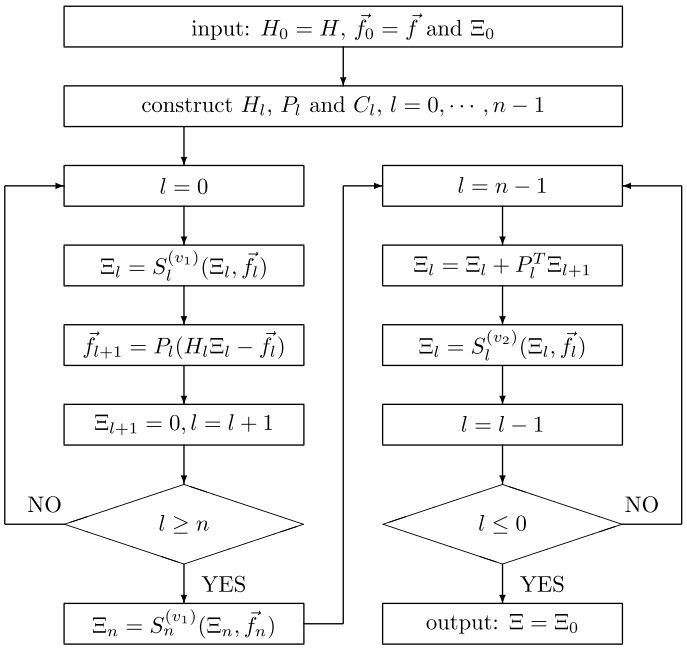

Fig. 2.3. A V-cycle of the n-level algorithm with v_1 pre- and v_2 post-smoothing iterations for solving the minimization problem (2.2).

The notation $S_l(\Xi_l, \vec{f}_l)$ is a smoothing operator which adopts the Gauss–Seidel iteration in this paper. Some special treatments for the boundary nodes have to be applied,

$$\vec{\Xi}_{\text{bound}} := \vec{\Xi}_{\text{bound}} - \delta\vec{\Xi}_{\text{bound}} \cdot \vec{n}_l, \tag{2.8}$$

where $\vec{n}_l$ is the unit normal of the face Γ_l and $\delta\vec{\Xi}_{\text{bound}}$ is the update introduced by the Gauss–Seidel iteration. Thus the boundary constraint is preserved in the smoothing.

A simple weighted restriction was chosen as the projection matrix $P_l(N_{l+1} \times N_l)$, i.e., the arithmetic mean average of the neighborhood of each coarse-grid point:

$$P_l(1 \leqslant i \leqslant N_{l+1}, 1 \leqslant j \leqslant N_l) := \begin{cases} \frac{1}{\sharp\{B^l_{(C_l)_i}\}} & \text{if } j \in B^l_{(C_l)_i}, \\ 0 & \text{otherwise}, \end{cases} \tag{2.9}$$

where $\sharp\{B\}$ represents the number of the set B and $(C_l)_i$ represents the ith number of the set C_l. Once the coarse points set C_l is fixed, the projection operator P_l can then be given using (2.9). It is natural to define a reasonable prolongation operator as the transpose of P_l and the coarse-level matrix as

$$H_{l+1} = P_l H_l P_l^{\mathrm{T}} (H_0 = H). \tag{2.10}$$

This way to construct the projection operator is very cheap in the implementation, which depends only on the connection of the grids and ignores the information from the values of the entries in the sparse matrices of the algebraic system. Thus the projection matrices are built only once in the whole simulation. The overall mesh redistribution algorithm is essentially dimension independent; thus the implementation in both the 2D and 3D cases can share most of the codes. In spite of the simplicity of this projection operator, the performance of the total algorithm is quite satisfactory according to our numerical experiments.

3. Phase-field model and its discretization

We will briefly describe the quantitative phase-field model provided by Karma et al. [15] and a finite element discretization to approximate the governing equations.

The quantitative phase-field model of the dendritic crystallization of a pure melt in two and three-dimensions takes the form

$$\partial_t u = D\Delta u + \frac{1}{2}\partial_t \psi, \tag{3.1}$$

$$\tau(\vec{n})\partial_t \psi = \vec{\nabla}\cdot[W^2(\vec{n})\vec{\nabla}\psi] + [\psi - \lambda u(1-\psi^2)](1-\psi^2) + \sum_{i=1}^{s}\partial_i\left(\left|\vec{\nabla}\psi\right|^2 W(\vec{n})\frac{\partial W(\vec{n})}{\partial(\partial_i\psi)}\right), \tag{3.2}$$

where the temperature T is rescaled to a dimensionless thermal field by $u = c_p(T - T_M)/L, c_p$ is the specific heat at constant pressure, L is the latent heat of fusion, and T_M is the melting temperature. The notation ∂_i denotes the partial differential operator $\partial/\partial x_i$. The phase-field variable ψ is defined as 1 in the solid phase and -1 in the liquid phase, with the interface implicitly given by its zero level set. The parameters D and λ are the thermal diffusivity and the constant control of the coupling between u and ψ, respectively. The time is rescaled by τ_0, a time characterizing atomic movement in the interface, and the length by W_0, a length characterizing the liquid–solid interface. Following [15], anisotropy is introduced in (3.2) by defining the width of the interface

$$W(\vec{n}) = W_0 a(\vec{n}) \tag{3.3}$$

and the characteristic time

$$\tau(\vec{n}) = \tau_0 a^2(\vec{n}), \tag{3.4}$$

where $a(\vec{n}) \in [0,1]$ is given by

$$a(\vec{n}) = (1-3\varepsilon_4)\left[1 + \frac{4\varepsilon_4}{1-3\varepsilon_4}\sum_{i=1}^{s}(\partial_i\psi)^4|\nabla\psi|^{-4}\right]. \tag{3.5}$$

In (3.5), the constant ε_4 parameterizes the deviation of $W(\vec{n})$ from W_0, the vector $\vec{n}$ is the unit normal to the isosurfaces of ψ and $s = 2, 3$ is the dimension of the domain. The asymptotic analysis in [15] was based on the sharp interface limit with the Gibbs–Thomson condition

$$u_{\text{int}} = -d(\vec{n})\kappa - \beta(\vec{n})V, \tag{3.6}$$

where u_{int} is the thermal field at the interface, $d(\vec{n})$ is the capillary length, κ is the local curvature, $\beta(\vec{n})$ is the interface kinetic coefficient and V is the normal velocity of the interface. In terms of $a(\vec{n})$,

$$d(\vec{n}) = d_0[a(\vec{n}) + \partial_\theta^2 a(\vec{n})],$$

where $d_0 = 0.8839/\lambda$. So the different choices of parameters W_0, τ_0 and λ can be used to simulate arbitrary β. Following [15,29], we set $W_0 = 1, \tau_0 = 1$ and $\lambda = 1.5957D$ for $\beta = 0$. The initial value of ψ takes the form

$$\psi_0(\vec{x}) = -\tan h\left(\frac{|\vec{x}| - r_0}{\sqrt{2}}\right), \tag{3.7}$$

where r_0 is the radius of the initial seed. The initial value of u decays exponentially from $u = 0$ at the interface to $-\Delta$ as $|\vec{x}| \to \infty$.

The finite element discretization described below is for three space dimension but the discretization in 2D is similar. The initial and boundary conditions are

$$\begin{aligned} &u(\vec{x},0) = u_0(\vec{x}), \quad \psi(\vec{x},0) = \psi_0(\vec{x}),\\ &\frac{\partial u}{\partial \vec{n}}(\vec{x},t)\Big|_{\partial\Omega} = 0, \quad \frac{\partial \psi}{\partial \vec{n}}(\vec{x},t)\Big|_{\partial\Omega} = 0. \end{aligned} \tag{3.8}$$

The weak formulation of the phase-field equations (3.2) takes the form

$$\begin{aligned} &(\partial_t u, v) = -D(\vec{\nabla}u, \vec{\nabla}v) + \frac{1}{2}(\partial_t\psi, v),\\ &\tau(\vec{n})(\partial_t\psi, \phi) = -(W^2(\vec{n})\vec{\nabla}\psi, \vec{\nabla}\phi) + ([\psi - \lambda u(1-\psi^2)](1-\psi^2), \phi) - (\vec{\omega}, \vec{\nabla}\phi), \end{aligned} \tag{3.9}$$

where ϕ, v are the test functions and

$$\vec{\omega} = |\vec{\nabla}\psi|^2 \left(\frac{\partial W(\vec{n})}{\partial(\partial_1\psi)} \frac{\partial W(\vec{n})}{\partial(\partial_2\psi)} \frac{\partial W(\vec{n})}{\partial(\partial_3\psi)} \right)^{\mathrm{T}} = \frac{16\epsilon W_0}{|\psi|^4} \begin{pmatrix} \partial_1\psi((\partial_2\psi)^2\mathcal{S}_{12} + (\partial_3\psi)^2\mathcal{S}_{13}) \\ \partial_2\psi((\partial_3\psi)^2\mathcal{S}_{23} + (\partial_1\psi)^2\mathcal{S}_{21}) \\ \partial_3\psi((\partial_1\psi)^2\mathcal{S}_{31} + (\partial_2\psi)^2\mathcal{S}_{32}) \end{pmatrix} \tag{3.10}$$

with

$$\mathcal{S}_{ij} = (\partial_i\psi)^2 - (\partial_j\psi)^2.$$

In our computations, linear finite elements were used in space, together with a semi-implicit alternating Crank–Nicolson scheme [27,36] in time:

$$\int \frac{\tau^{(n)}(\vec{n})}{\Delta t}\psi^{(n+1)}\phi \,\mathrm{d}\Omega + \int [W^{(n)}(\vec{n})]^2\vec{\nabla}\psi^{(n+1)} \cdot \vec{\nabla}\phi \,\mathrm{d}\Omega = \int \frac{\tau^{(n)}(\vec{n})}{\Delta t}\psi^{(n)}\phi \,\mathrm{d}\Omega - \int W^{(n)}(\vec{n})\vec{\omega}^{(n)} \cdot \vec{\nabla}\phi \,\mathrm{d}\Omega + \int (\psi^{(n)} - \lambda u^{(n+1/2)}[1 - (\psi^{(n)})^2])[1 - (\psi^{(n)})^2]\phi \,\mathrm{d}\Omega, \tag{3.11}$$

$$\int \frac{u^{(n+3/2)}}{\Delta t} v \,\mathrm{d}\Omega + D\int \vec{\nabla}u^{(n+3/2)} \cdot \vec{\nabla}v \,\mathrm{d}\Omega = \int \frac{u^{(n+1/2)}}{\Delta t} v \,\mathrm{d}\Omega + \frac{1}{2}\int \frac{\psi^{(n+1)} - \psi^{(n)}}{\Delta t} v \,\mathrm{d}\Omega, \tag{3.12}$$

where Δt is a constant time step.

Note that the coefficient matrices of both linear systems (3.11) and (3.12) are symmetric and positive definite. These matrices were inverted efficiently by using an algebraic multi-grid method proposed by Brandt et al. [4].

4. Monitor function for the phase-field equations

We now consider a suitable monitor function for the phase-field equations of the dendritic growth problem. The commonly adopted gradient-based monitor function is formulated as

$$m = \sqrt{|\nabla\psi|^2 + |\nabla u|^2 + \varepsilon_\alpha}, \quad \varepsilon_\alpha \text{ is a constant}, \tag{4.1}$$

see, e.g. [25,34]. But our numerical experiments indicated that the use of the monitor function (4.1) does not lead to satisfactory meshes. Fig. 4.1 shows the mesh of the initial value of the form (3.7) with the radius of $r_0 = 5$, which was produced using the monitor function (4.1). Since the scale of the solidification seed is quite

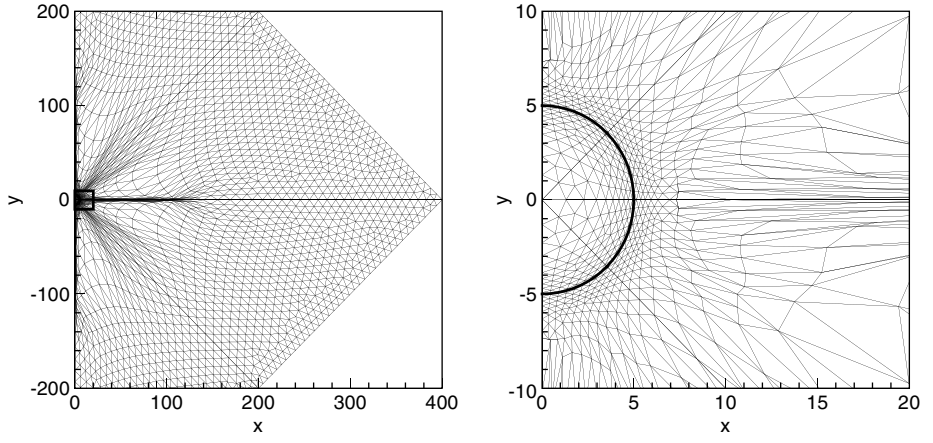

Fig. 4.1. The mesh generated for the initial value based on the gradient based monitor (4.1). The left figure is the mesh in the entire domain, and right figure is a zoomed view near the origin.

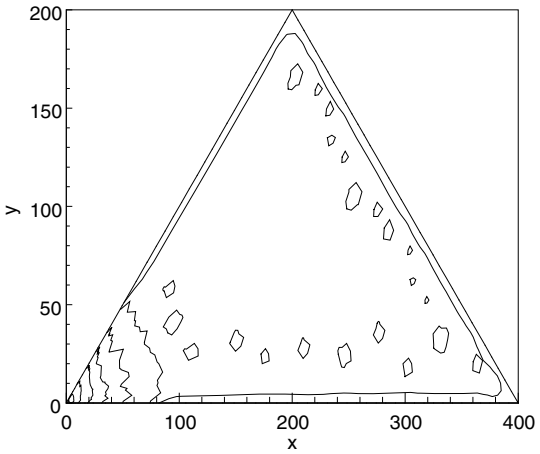

Fig. 4.2. The contours of the monitor function (4.1) which led to the mesh of Fig. 4.1. The contour values from left to right are ranged from 10^{-1} to 10^{-7}.

small compared to the scale of the domain size, a very small ε_α, say $\varepsilon_\alpha = 10^{-10}$, is adopted to cluster enough grid points around the phase interface. Making use of the symmetry, the actual computational domain is a triangular domain with vertices $(0,0)$, $(400,0)$ and $(200,200)$. The average spatial mesh size for the initial uniform mesh is about $\Delta x_{\text{init}} = 8$. Some mesh cells with poor quality are observed not far from the solidification seed. Those cells with poor quality will deteriorate the numerical solutions after several time steps.

Fig. 4.2 shows the contours of the monitor function (4.1) for the initial function. It is clearly observed that the quality of the contour is poor. Such unsatisfactory numerical behavior can sometimes be relieved by smoothing the monitor function. Our numerical experience shows that for this particular problem, the commonly used smoothing technique was not feasible anymore. We observed that the scaling of the large domain yields numerical difficulty, and the constant regularization parameter ε_α may be the reason for the quality degeneracy of the generated mesh. It is then expected that a gradually varying regularization field in the monitor function can be helpful to relieve this difficulty.

We noticed that the main dynamical contribution from the phase-field to the thermal field is concentrated along the solidification interface, where the transits of the phase-field variable from -1 to 1 are located. Thus the latent heat generated from the solidification has a similar spatial distribution as the magnitude of $|\nabla\psi|$ on the domain, which is then diffused to the whole domain by the heat kernel in the thermal field equation. The gradient field of the phase variable, after some moderate long-range diffusive processing, can then be the candidate of the gradually varying regularization in the monitor function. Based on such understanding of the interaction between the phase-field variable and the thermal field, we define a new variable $\tilde{\psi}$ which satisfies

$$\tilde{\psi} = (1 - \mu\Delta)^{-1}|\nabla\psi|^2. \tag{4.2}$$

We then reformulate the monitor function by replacing the constant parameter ε_α with a varying regularization field of the form

$$m = \sqrt{|\nabla\psi|^2 + |\nabla u|^2 + \tilde{\varepsilon}(\mu)}, \tag{4.3}$$

where

$$\tilde{\varepsilon}(\mu) = \tilde{\varepsilon}_\alpha|\tilde{\psi}| \tag{4.4}$$

and $\tilde{\varepsilon}_\alpha$ is again a small constant. It is remarked that the monitor function (4.3) can be regarded as a form of smoothing technique that has been used and analyzed by Huang and Russell [13]. Below we will discuss some practical issues of this monitor function, such as fast solution procedure and the choices of the parameters involved.

In the numerical experiments, the numerical results are found quite insensitive to the parameter $\tilde{\varepsilon}_\alpha$. Therefore, we always set $\tilde{\varepsilon}_\alpha = 0.01$. In the original monitor function (4.1), the role of the parameter $\tilde{\varepsilon}_\alpha$ is to keep the

monitor function away from zero. In the modified monitor function (4.3), the term $\tilde{\varepsilon}(\mu)$ should also do the same job, thus the strategy to choose the parameter μ is to satisfy this restriction. In general, the value of μ may be time dependent. However, in our numerical tests, the numerical results are also found insensitive to the choice of μ. Thus we just need to set μ sufficiently large. In practice, for a given computational domain, the value of μ is chosen to satisfy

$$\mu = \min\{\mu : \tilde{\varepsilon}(\mu) > \tilde{\varepsilon}_u \text{ for all grid points}\}, \tag{4.5}$$

where $\tilde{\varepsilon}_u$ is the machine epsilon.

In the implementation, it is not necessary to approximate

$$(1 - \mu\Delta)\tilde{\psi} = |\nabla\psi|^2 \tag{4.6}$$

accurately. Instead we used only 2–3 algebraic multi-grid iterations to obtain reasonable approximations for $\tilde{\psi}$. Thus the long-range diffusive effect can be achieved efficiently. Fig. 4.3 presents the grids for the initial value generated by the modified monitor function (4.3). As expected, the mesh quality is improved significantly by comparing with Fig. 4.1. Fig. 4.4 shows the mesh when the simulation is steady, with the contours of ψ and u.

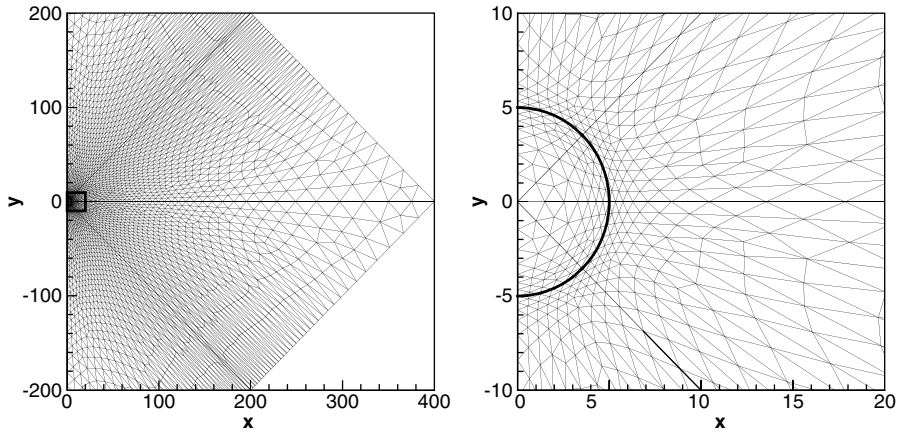

Fig. 4.3. The mesh for the initial value generated using the modified monitor function (4.3). The left figure is the global view of the mesh, and the right one is a zoomed part around the phase interface.

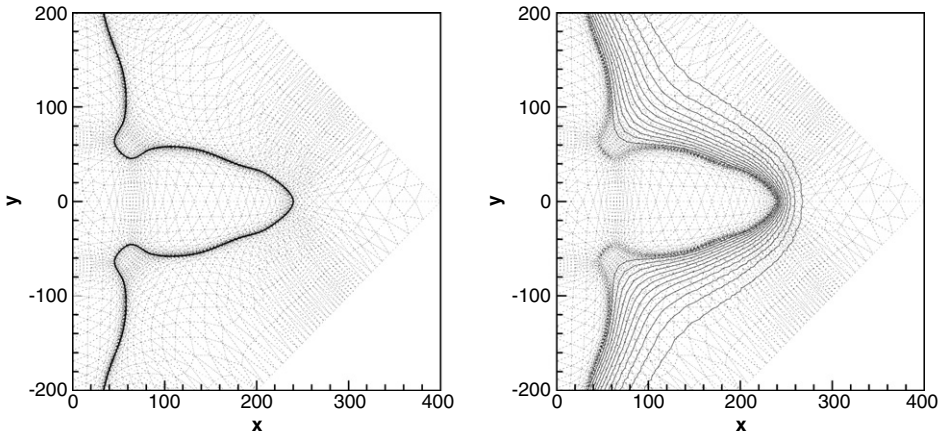

Fig. 4.4. The mesh generated using the monitor function (4.3) when the simulation is steady. The solid lines are the contours of ψ (left) and u (right).

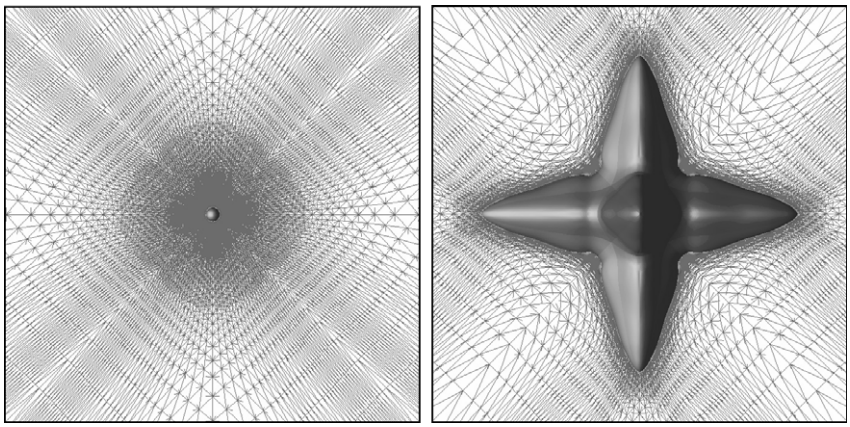

Fig. 4.5. The mesh generated by the modified monitor function (4.3) in 3D simulation. The left one is the mesh for the initial value and the right one is the mesh when the simulation is steady. Both meshes are sliced at $z = 0$. Here $\mu = 50, r_0 = 5$, the dimension of the displayed domain is $[-300, 300]^3, \Delta x_{\text{init}} \approx 9$ and the number of DOFs is 6545.

Numerical experiments also suggested that the modified monitor function (4.3) can also work well in 3D. In Fig. 4.5, a 3D mesh obtained by using (4.3) is plotted, where the parameters used are $\mu = 50, r_0 = 5$ and $\Delta x_{\text{init}} \approx 9$, and the computational domain is a tetrahedron with vertices $(0,0,0)$, $(300,0,0)$, $(150,150,0)$ and $(100,100,100)$.

5. Numerical results

In this section, we will present numerical results for both 2D and 3D simulations. All simulations were carried out on a Dell Precision 490 with core speed 3.0 GHz. All the simulations in this paper are based on the adaptive finite element library AFEPack [22]. The initial uniform meshes in 2D were generated by software Easymesh [28]. We made full use of the symmetries of the problems in our implementation.

5.1. Numerical validation in 2D

In [29], simulations on uniform meshes and *h*-adaptive meshes have been carried out in 2D, and the growing tip velocities of the dendrites were found to be in good agreement with the results calculated by using the Green function method [15]. In their works, it was noted that the dimension of the computational domain affects the numerical results, especially for the simulations at low undercooling. So we used the computational domain with the same scaling as that in [15,29] for both the high undercooling and the low undercooling cases.

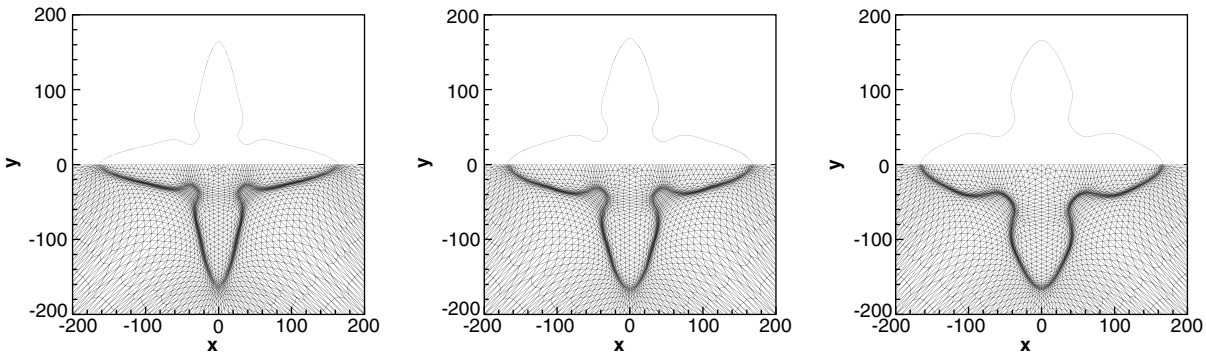

Fig. 5.1. The dendrites and meshes produced by 2D simulations with $\Delta = 0.65$, 0.55 and 0.45 from left to right.

To compare our results with the published ones, we set the phase-filed modeling parameter $\tilde{\varepsilon}_4 = 0.05$ and rescaled the tip velocity [15]:

$$\tilde{V}_{\text{tip}} = \frac{V_{\text{tip}} d_0}{D}. \tag{4.7}$$

We first present the numerical results at high undercooling. Fig. 5.2 shows the tip velocities at the undercooling parameter $\Delta = 0.65$, 0.55 and 0.45 which are compared with the corresponding solvability results provided by [15]. It is observed that the computational results (the marked curves) converge to the data given by the solvability theory provided the time t is sufficiently large. The other parameters for these simulations were $\Delta x_{\text{init}} = 8$ and $\mu = 500$ in the domain $[-400, 400]^2$. By using the symmetries, the computations were carried out on the triangular domain with vertices as $(0,0)$, $(400,0)$ and $(200,200)$. The number of the grid points in the simulations are 804 and the total computing time cost was less than 1 h with $\Delta t = 0.1$.

For the low undercooling cases, it had been noted in [29] that the simulations should be carried out in a large computational domain to catch up the correct tip velocity. Our simulations for low undercooling cases were carried out on an enlarged domain $[-1600, 1600]^2$ with $\Delta x_{\text{init}} = 8$ and $\mu = 50,000$. Other parameters were

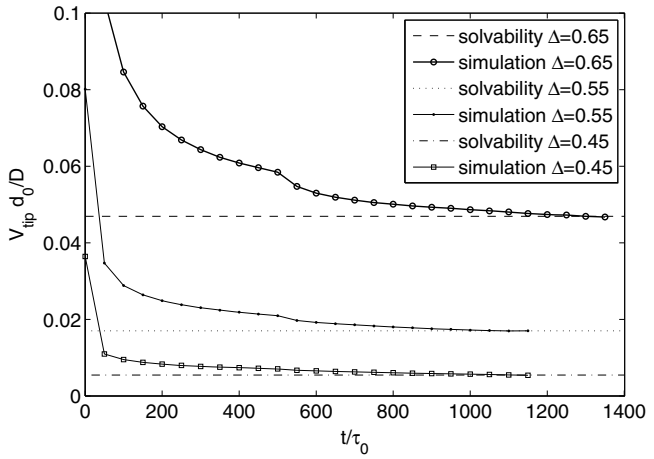

Fig. 5.2. Comparison of the tip velocity from numerical simulations and the Green function method in higher undercooling.

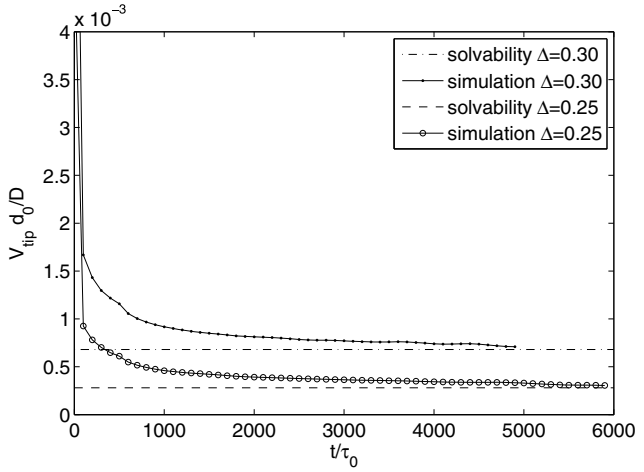

Fig. 5.3. Comparison of the tip velocity from numerical simulations and the Green function method in lower undercooling.

the same as those at high undercooling. The total number of grid points in these simulations was 11,904, and the CPU time for each simulation was about 40 h.

In Fig. 5.3, we plot the tip velocity from the numerical simulations in the rescaled time for two low undercooling parameters $\Delta = 0.30$ and 0.25. Again the numerical simulations can catch the correct tip velocity given by the solvability theory after the dendritic growth becomes steady for both undercooling parameters.

5.2. 2D complex dendritic structures

There were no observed side-branches in Fig. 5.1 while Fig. 5.4 presented a more physical result. The parameters for Fig. 5.4 are the same as those for Fig. 5.1, except that $\Delta = 0.60$ and $D = 1$. The computational domain was enlarged to {(0,0), (1600,0), (800,800)} to get more fully developed structures with $\Delta x_{\text{init}} = 16$ and $\mu = 10000$. The number of DOFs was 3022 and the CPU time was about 2 h. Though it has been pointed out [29] that the appearance of the side-branches is due to the numerical noises, the mesh generated in this example illustrated that our method has the potential to handle such complex interfaces as plotted. To introduce more numerical noises to produce the side-branch structures, we set $\tilde{\varepsilon}_\alpha = 1$ in the monitor function to make the mesh size big enough to deviate the numerical solution from a well resolved interface.

Fig. 5.5 is the result of a simulation on a refined mesh based on the result in Fig. 5.4 with even complex dendritic structure. We continued the simulation until the dendritic tip reached around the boundary of the

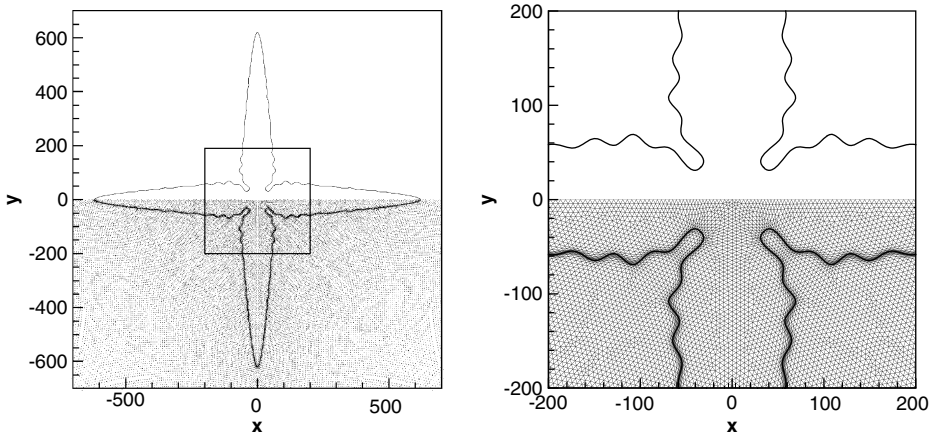

Fig. 5.4. A typical dendrite and mesh with large computational domain. The displayed domain is $[-1600, 1600]^2$. The right figure is a zoomed view of the left figure marked by a square.

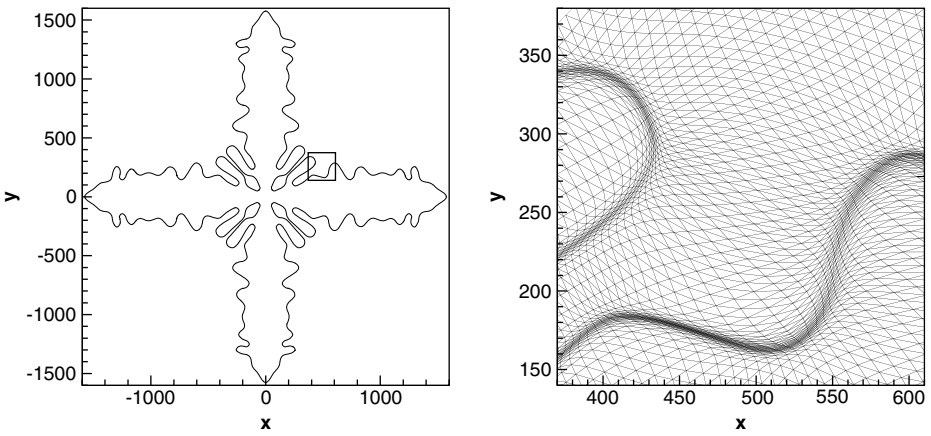

Fig. 5.5. A dendrite produced by the numerical simulation with complex side-branches. The right figure is a zoomed view of the left figure marked by a square.

domain that the side-branches were evolving to a very complex profile. The computational domain was same as that in Fig. 5.4, with $\Delta x_{\text{init}} = 8$.

5.3. 3D simulations

Utilizing the symmetries, the 3D computations were carried on a tetrahedron domain with vertices as $(0,0,0)$, $(L,0,0)$, $(L/2,L/2,0)$ and $(L/3,L/3,L/3)$. The background meshes used were the uniform meshes

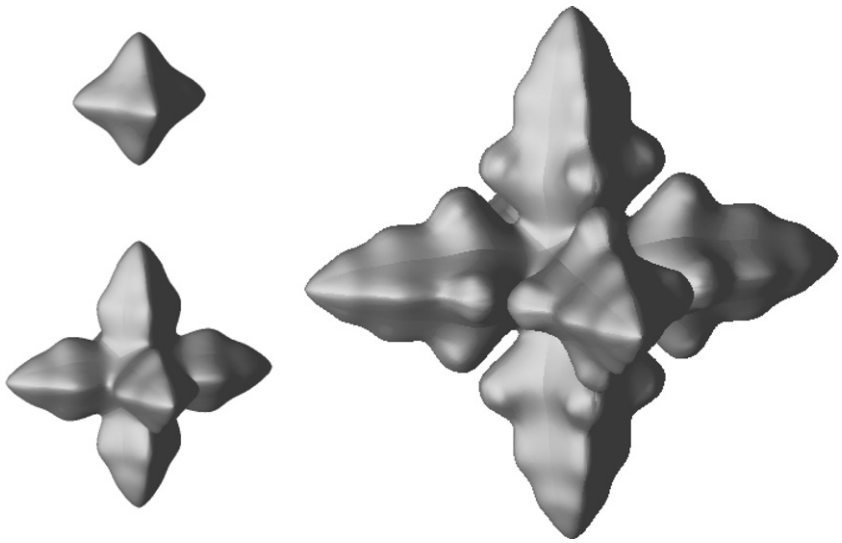

Fig. 5.6. A typical dendrites growth produced by 3D simulations with 47905 DOFs at three different times at low undercooling.

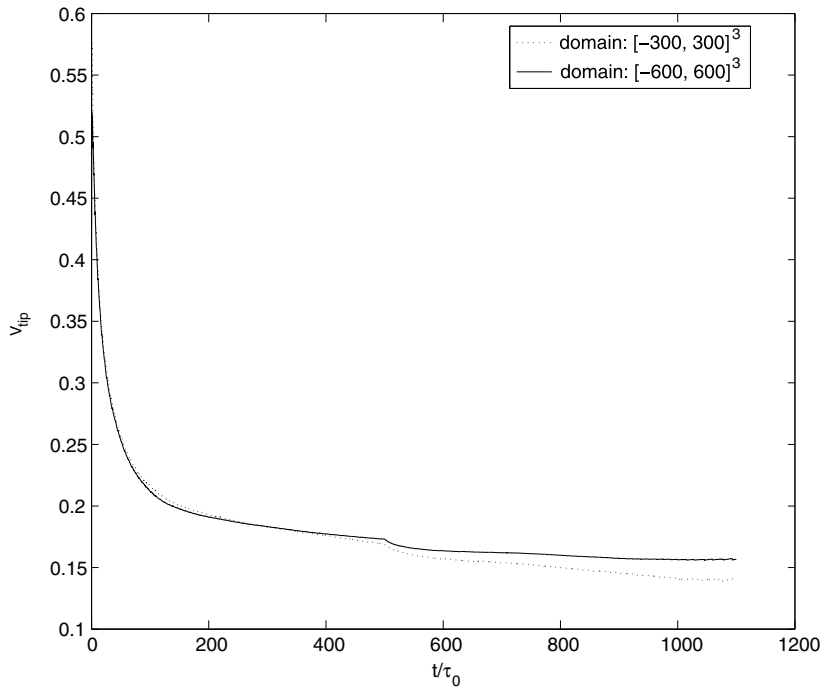

Fig. 5.7. The tip velocity of the dendritic growth at $\Delta = 0.45$ and $D = 2$ computed on two different solution domains without using scaling.

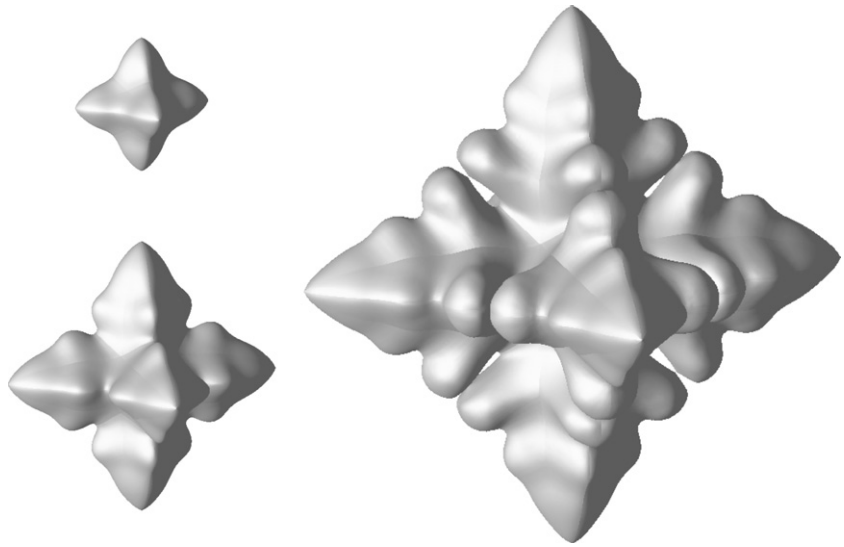

Fig. 5.8. Same as Fig. 5.6, except at high undercooling.

obtained from a sequence of uniform refinements from the original tetrahedron. The 3D numerical results were visualized with the software Open Data Explorer [35]. Our main concern in the following simulations was to illustrate the potential of our moving mesh method to approximate a quite complex 3D structure with relatively small number of DOFs.

Fig. 5.6 is a typical dendrite produced by our 3D simulation with parameters $\epsilon = 0.05, \Delta = 0.45, D = 2$ and $\mu = 1000$. The computational domain was the one with $L = 600$. The background mesh was generated by six uniform refinements from the single tetrahedron, and the number of nodes is 47,905. It can be seen that the side-branches are equally distributed along the dendrites as the physical realities. This simulation took us about 60 h of CPU time.

In Fig. 5.7, we plot the tip velocity of the dendritic growth at $D = 0.45$, with both small domain and large domain without rescaling. Though we have no solvability data for comparison, both tip velocity data con-

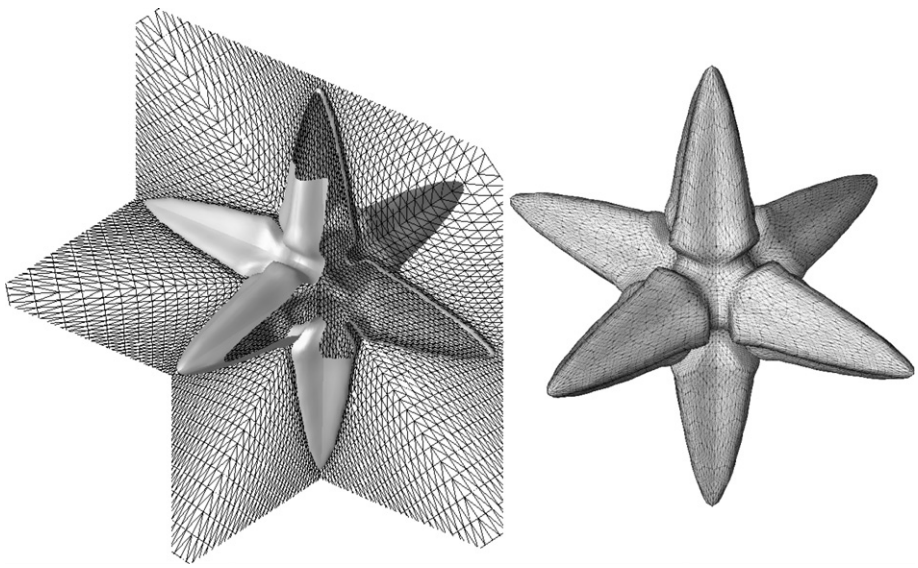

Fig. 5.9. The mesh structure of a typical 3D simulation. On the left, three mesh slices located on the coordinate planes are plotted with the dendritic interface. On the right, the meshes located on the phase interface are plotted. The data were obtained at the 1000th time step with the parameters $\Delta = 0.65, D = 2$ and $L = 300$.

Table 1
Comparison of the degree of freedoms with the previous works

Publications	Models	Methods	Adaptivity	DOFs/Nodes	Dimension
Karma and Rappel [15]	Phase-field	FDM	No adaptive	3×10^4 to 4×10^6	2D
Schmidt [32]	Front tracking	FEM	*h*-adaptive	$\approx 2 \times 10^5$	3D
Present work	Phase-field	FEM	*r*-adaptive	804–11,904	2D
				47,905	3D

verged to constant values with little difference. As expected, the result on the larger domain was smoother than the one obtained on the smaller domain, indicating that the computation seems to be reliable.

In Fig. 5.8, we plot the typical dendrites produced by numerical simulation at high undercooling $\Delta = 0.65$. All the other parameters and the background mesh were the same as those for the simulation at the low undercooling. This numerical simulation evolved to more complex dendritic structures, while our moving mesh method can still cluster enough grid points on the phase interface. To show the 3D mesh structure, we plot in Fig. 5.9 the mesh on some slices and on the interfaces. It can be seen that as desired that a large portion of the grid points are moved to the phase interfaces.

We close this section by comparing the degree of freedoms used in the present work and some previously published results. It is seen from Table 1 that the finest mesh used in [15] is about 4×10^6 while the present work uses only 1.2×10^4 for the 2D computations; and the finest mesh used in [32] (which uses the *h*-adaptive grid method) is about 2×10^5 while the present work uses only 4.8×10^4 for the 3D computations.

6. Concluding remarks

In this paper, we present an efficient *r*-adaptive mesh algorithm for the phase-field model of dendritic growth in both two- and three-dimensions. The mesh redistribution is realized by solving an elliptic boundary control problem together with a nonlinear multi-grid algorithm. The governing equations are discretized in space by linear finite elements and a split time-level scheme is used to numerically integrate in time. One novel aspect of the method is the choice of a regularized monitor function.

In 2D, the numerical experiments for both high and low undercooling cases suggested that the tip velocities obtained using the present method is in a good agreement with the published results. It is also found that in obtaining the same accuracy the present computational costs are smaller than those of the previous ones. By using a reasonably small number of DOFs, the proposed method can produce quite complex dendritic structures in two- and three-dimensions. This suggests that the proposed moving grid method has the potential to simulate more realistic physical problems with the presently available computing resources. Following this direction, we are now carrying out numerical simulations of the dendritic growth in multi-component alloys [31].

Acknowledgments

The research of Wang and Tang was supported by Hong Kong Baptist University and Hong Kong Research Grants Council. The research of Li was supported in part by the National Basic Research Program of China under the grant 2005CB321701, an award for National Excellent Doctoral Dissertation by the Ministry of Education of China, and by the Joint Applied Mathematics Research Institute between Peking University and Hong Kong Baptist University. The research of Tang was also supported by NSAF Grant #10476032 of National Science Foundation of China. The authors are very grateful to Prof. Hermann Brunner for polishing the English writing.

References

[1] Robert Almgren, Variational algorithms and pattern formation in dendritic solidification, J. Comput. Phys. 106 (1993).
[2] G. Beckett, J.A. Mackenzie, M.L. Robertson, An *r*-adaptive finite element method for the solution of the two-dimensional phase-field equations, Commun. Comput. Phys. 1 (2006) 805–826.

[3] J.U. Brackbill, J.S. Saltzman, Adaptive zoning for singular problems in two dimensions, J. Comput. Phys. 46 (1982) 342–368.
[4] A. Brandt, S.F. McCormick, J.W. Ruge, Algebraic multigrid (AMG) for sparse matrix equations, in: D.J. Evans (Ed.), Sparsity and its Applications, Cambridge University Press, Cambridge, 1984.
[5] G. Caginalp, X. Chen, On the evolution of phase boundaries, in: M.E. Gurtin, G.B. McFadden (Eds.), The IMA Volumes in Mathematics and its Applications, vol. 43, Springer-Verlag, New York, 1992, p. 1.
[6] Y. Di, R. Li, T. Tang, A general moving mesh framework in 3D and its application for simulating the mixture of multi-phase flows, Commun. Comput. Phys. 3 (2008) 582–602.
[7] Y. Di, R. Li, T. Tang, P. Zhang, Moving mesh methods for singular problems on a sphere using perturbed harmonic mappings, SIAM J. Sci. Comput. 28 (2006) 1490–1508.
[8] Y. Di, P.W. Zhang, Moving mesh kinetic simulation for sheared rodlike polymers with high potential intensities, Commun. Comput. Phys. 1 (2006) 859–873.
[9] W.M. Feng, P. Yu, S.Y. Hu, Z.K. Liu, Q. Du, L.Q. Chen, Spectral implementation of an adaptive moving mesh method for phase-field equations, J. Comput. Phys. 220 (2006) 498–510.
[10] M.E. Glicksman, Free dendritic growth, Mater. Sci. Eng. 65 (1984) 45–55.
[11] S.C. Gupta, The classical Stefan problem basic concepts, modelling and analysis, North-Holland Series in Applied Mathematics and Mechanics, vol. 45, JAI Press, Greenwich, 2003.
[12] S.C. Huang, M.E. Glicksman, Fundamentals of dendritic solidification – I and II, Acta Metall. 29 (1981) 701–734.
[13] W. Huang, R. Russell, Analysis of moving mesh partial differential equations with spatial smoothing, SIAM J. Numer. Anal. 34 (1997) 1106–1126.
[14] A. Karma, W.J. Rappel, Phase-field method for computationally efficient modeling of solidification with arbitrary interface kinetics, Phys. Rev. E 53 (4) (1996) R3017–R3020.
[15] A. Karma, W.J. Rappel, Quantitative phase-field modeling of dendritic growth in two and three dimensions, Phys. Rev. E 57 (4) (1998) 4323–4349.
[16] D.A. Kessler, J. Koplik, H. Levine, Pattern selection in fingered growth phenomena, Adv. Phys. 37 (3) (1988) 255–339.
[17] D.A. Kessler, J. Koplok, H. Levine, Geometrical models of interface evolution III. Theory of dendritic growth, Phys. Rev. A 31 (1985) 1712–1717.
[18] W. Kurz, D.J. Fisher, Fundamentals of Solidification, Trans Tech Publications Ltd., 1989.
[19] J.S. Langer, Existence of the needle crystals in local models of solidification, Phys. Rev. A 33 (1986) 436–441.
[20] J.S. Langer, Models of pattern formation in first-order phase transitions, in: G. Grinsteil, G. Mazenko (Eds.), Directions in Condensed Matter Physics: Memorial Volume in Honor of Shang-Keng Ma, Series on Directions in Condensed Matter Physics, vol. 1, World Scientific Press, Singapore, 1986.
[21] J.S. Langer, Lectures on the theory of pattern formation, in: J. Souletie, J. Vannimenus, R. Stora (Eds.), Chance and Matter, Les Houches, North-Holland, Amsterdam, 1987, pp. 629–711.
[22] R. Li, W.B. Liu, <http://circus.math.pku.edu.cn/AFEPack>.
[23] R. Li, T. Tang, P.W. Zhang, A moving mesh methods in multiple dimensions based on harmonic maps, J. Comput. Phys. 170 (2001) 562–588.
[24] R. Li, T. Tang, P.W. Zhang, A moving mesh finite element algorithm for singular problems in two and three space dimensions, J. Comput. Phys. 177 (2002) 365–393.
[25] K. Lipnikov, M. Shashkov, The error-minimization-based strategy for moving mesh methods, Commun. Comput. Phys. 1 (2006) 53–80.
[26] J.A. Mackenzie, M.L. Robertson, A moving mesh method for the solution of one-dimensional phase-field equations, J. Comput. Phys. 181 (2002) 526–544.
[27] M. Mo, Y.Q. Huang, An alternating Crank–Nicolson method for decoupling the Ginzburg-Landau equations, SIAM J. Numer. Anal. 35 (1998) 1740–1761.
[28] B. Niceno, <http://www-dinma.univ.trieste.it/nirftc/research/easymesh/>.
[29] N. Provatas, N. Goldenfeld, J. Dantzig, Efficient computation of dendritic microstructures using adaptive mesh refinement, Phys. Rev. Lett. 80 (1998) 3308–3311.
[30] N. Provatas, N. Goldenfeld, J. Dantzig, Adaptive mesh refinement computation of solidification microstructures using dynamic data structures, J. Comput. Phys. 148 (1999) 265–290.
[31] J. Rosam, P.K. Jimack, A.M. Mullis, A fully implicit fully adaptive time and space discretization method for phase-field simulation of binary alloy solidification, J. Comput. Phys. 225 (2007) 1271–1287.
[32] Alfred Schmidt, Computation of three dimensional dendrites with finite elements, J. Comput. Phys. 125 (1996) 293–312.
[33] Z.J. Tan, T. Tang, Z.R. Zhang, A simple moving mesh method for one- and two-dimensional phase-field equations, J. Comput. Appl. Math. 190 (2006) 252–269.
[34] H.Z. Tang, A moving mesh method for the Euler flow calculations using a directional monitor function, Commun. Comput. Phys. 1 (2006) 656–676.
[35] Inc. VIS, <http://www.opendx.org/>.
[36] H. Wang, R. Li, Mesh sensitivity for numerical solutions of phase-field equations using *r*-adaptive finite element methods, Commun. Comput. Phys. 3 (2008) 357–375.