Journal of Computational Physics **178,** 464–497 (2002)
doi:10.1006/jcph.2002.7037, available online at http://www.idealibrary.com on IDEAL®

An Adaptive Finite Volume Method for Incompressible Heat Flow Problems in Solidification

C. W. Lan, C. C. Liu, and C. M. Hsu

Department of Chemical Engineering, National Taiwan University, Taipei, Taiwan 106, Republic of China
E-mail: cwlan@ntu.edu.tw

Received January 11, 2001; revised July 6, 2001

An adaptive finite volume method is presented for solving incompressible heat flow problems with an unknown melt/solid interface, mainly in solidification applications, using primitive variables on a fixed collocated grid. A phase-field variable is introduced to treat the melt/solid interface, which is assumed to be diffusive, so that the complicated interfaces and phase change (using the enthalpy model) can be treated easily. The method is implemented through an object-oriented way based on adaptive mesh refinement and coarsening using dynamic data structures and derived data types of FORTRAN90. In addition to the refinement on the interfaces or boundaries, the mesh can be adapted to a solution based on numerical errors or gradients. Extensive tests are performed for cases with a fixed or free interface, and excellent agreement with the body-fitted or front tracking schemes is obtained. Furthermore, by gradual reduction of the interface thickness, the sharp-interface limit can be reached, which ensures the correctness of using a diffusive interface. The present approach is particularly suitable for problems having a complicated interface morphology as well as phase evolution, such as the phase-field simulation of dendritic growth. Two examples, without and with convection, are further given and good agreement with previous results are found.

Key Words: adaptive refinement; moving boundary; finite volume method, solidification; phase-field simulation.

1. INTRODUCTION

In many engineering problems, the interface between phases plays an important role in the product quality. Hence, its prediction (or design) is an important task in process optimization. However, the intricate coupling of interface morphology as well as its evolution with heat flow often makes the simulation very challenging. One important category of the problems is in the solidification processing, such as casting and crystal growth, where the

0021-9991/02 $35.00

interface deformation and its morphological instability due to convective heat and mass transfer have strong influences on the quality of the solidified materials. Moreover, the evolution of secondary morphological structures and the formation of a new phase are important to alloy design, and their prediction is also a great challenge in numerical simulation. Many numerical approaches [1–18] have been proposed to solve these problems. From the treatment of the interface, they can be grouped into the so-called front tracking method [e.g., 1–8] and the fixed-grid approach [8–18]. The front tracking method uses a distinguished equation for simulating the interface, while each phase is modeled separately. With the use of body-fitted coordinates, the front tracking method has been used extensively, especially in crystal growth modeling. However, the interface morphology that can be treated by this approach is usually relatively simple. As the morphology becomes complicated, particularly with the evolution of secondary structures or phase separation, the front tracking approach is in general less capable.

On the other hand, the fixed-grid approach using a phase-field variable to define the interface region, which is assumed to be diffusive, while using the same equations for the whole domain, becomes much more suitable. Although the use of diffusive interface has its physical meaning, it is more or less a numerical approximation, because in practical implementation the numerical thickness is still much thicker than the real one. This kind of approach includes the enthalpy model [8–14], phase-field simulation [e.g., Ref. 15], level sets [16–18], etc. Although one needs a delicate continuum physical model [9, 15, 19] for the coexisting two phases in the diffusive interface and in the computational cells, the complexity of interface morphology and its evolution can be treated implicitly and easily. Furthermore, for this approach a structured fixed Cartesian mesh is usually adopted, which is trivial in grid generation. Although the fixed-grid approach has proven to be powerful in many solidification simulations, there is a severe drawback, especially for the sharp interface, such as the phase boundary of a pure material. Because the interface is treated as a diffusive one, several cells need to be allocated in the region. Without a fine mesh, in addition to the loss of accuracy, the coarse grid often leads to numerical instability. Putting enough cells along the interface for a structured mesh is not realistic. Therefore, to implement the approach efficiently, using local mesh refinement or an adaptive mesh is highly required. Furthermore, as the interface moves away, not to increase the computational load, mesh coarsening is also inevitable. A new approach proposed recently using the so-called "cut-cell" [20–22] for front tracking in a fixed Cartesian mesh has also demonstrated some nice results in the simulation of highly deformed interfaces. However, this approach can be more difficult to be extended to three-dimensional (3D) problems. Beside that, similar to the fixed-grid approach, the adaptive mesh refinement (AMR) can be useful in the cut-cell implementation as well. Furthermore, a powerful approach using the "immersed boundary technique," [23] based on a fixed grid for a moving Lagrangian interface, has also been demonstrated for solidification [24]. Again, the diffusive interface concept is introduced, and therefore it is regarded as a mixed-type approach, even though the "front tracking" has been adopted there. Again, AMR should also be useful in such a technique.

Although the concept of using AMR on the fixed-grid method is simple and the implementation has been successful for finite element methods (FEMs) [e.g., 25, 26], it is rarely adopted for finite difference or finite volume methods (FVMs) in two-phase calculations. Nevertheless, some implementations for a single-phase domain have been reported [27–30], especially for solving hyperbolic conservation equations having a shock propagation [31].

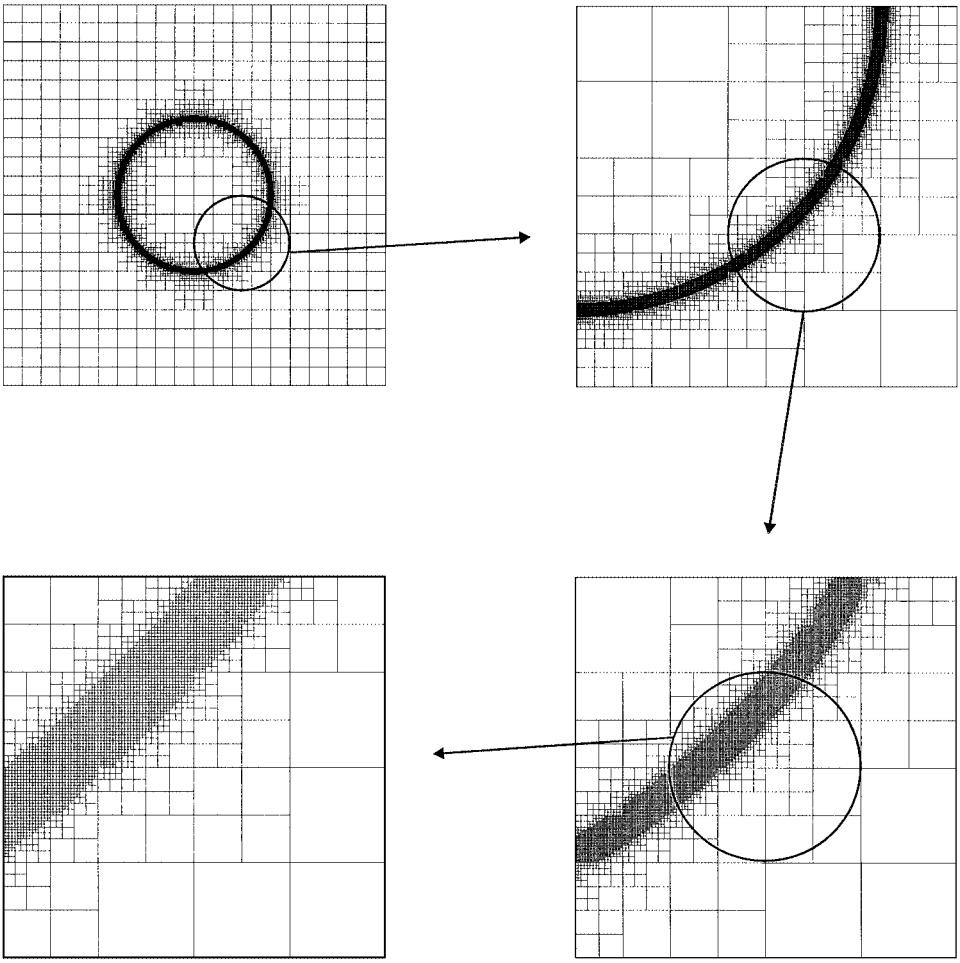

FIG. 1. Schematic of an adaptive multilevel mesh for a solid object inside a fluid domain; the mesh is refined along the interface.

As the adaptive scheme is concerned, two typical cell geometries are generally used for two-dimensional (2D) problems, i.e., the triangular and quadrilateral cells. The triangular cells are more popular in the FEM implementation. They are very flexible in cell division, but the implementation is more difficult due to the need of grid regularization for minimizing grid skewness and the complicated node numbering. The data structure and the node reordering could be more difficult as well. On the other hand, the mesh refinement for the quadrilateral cells on a structured mesh does not suffer the disadvantages. In addition, the refinement can be quite effective and have many varieties. As illustrated in Fig. 1, with the multilevel mesh refinement, one can achieve an excellent mesh quality for tracking the diffusive interface. Furthermore, the grid levels can span the domain over different length scales, which makes the simulation of multiscale problems feasible. Nevertheless, the dangling nodes are problems in the discretization by the finite difference scheme or FEMs. One needs different difference formula or shape functions for different node configurations. For an arbitrarily refinement, the implementation becomes extremely complicated. However, for the FVM implementation, the mesh is ideal if the vertex points can be avoided during

discretization [27–29]. Therefore, this type of refinement is adopted in this study. It should be pointed out that after refinement this "pseudo-structured" mesh is no longer a structured one because the number of faces for fluxes in a cell is no longer four, but arbitrary, for 2D problems. Since the root grid is still a structured mesh (e.g., a uniform Cartesian grid), the grid generation at the beginning is trivial. Although the FVM approximation for this kind of refined cells is straightforward, a delicate data structure is also required for an efficient implementation. Fortunately, from the computer programming point of view, this mesh fits perfectly into a quadtree data structure. Therefore, the use of pointers could reduce significantly the programming effort, and the use of C++ or FORTRAN90 becomes a better choice. Further, using the object-oriented concepts also makes the coding easier and enhances the reusability of the code. Another implementation using the patch [30] is a special case, but much less flexible.

As the fixed-grid approach is adopted, the use of a phase-field variable for accurately locating the interface, which is often referred as interface capturing, is crucial. As mentioned previously, several methods have been proposed in the literature, including the enthalpy–porosity method [8–14], level-set methods [16–18], and phase-field models [15, 25, 26]. Again, the "cut-cell" [22] and Lagrangian-cell algorithms (immersed boundary method) [23, 24] are regarded as front tracking here. The phase-field and level-set models require the solution of an additional equation for the phase-field variable, while the enthalpy model obtains the phase-field variable by simple interpolation from temperature fields. For all of the methods, the interface thickness is not zero due to the diffusive nature of the finite cell size, even for a sharp interface. Although the enthalpy method seems to require less effort for computation, it is notorious by its slow convergence and instability due to the large source term in the energy and momentum equations, especially when the grid number is not enough to describe the two-phase region. One also needs to use an exceptionally large viscosity for the solid phase, while the physical properties in the two-phase region are obtained by interpolation. Including the porosity model [11] may allow one to treat the no-slip boundary implicitly. More importantly, the difference between the liquidus and solidus temperatures restricts the cell size used. Therefore, for a sharp interface, an extremely fine mesh is required for a realistic simulation. On the other hand, the phase-field method using a smooth phase-field variable, which is obtained by solving the phase-field equation, often converges much faster. Therefore, it is believed that the combination of both concepts may help the convergence, but the solution of the phase-field equation is not necessary, at least for macroscopic solidification. Furthermore, through AMR, one can further gain both the efficiency and the accuracy. To our best knowledge, such an implementation based on the adaptive pseudo-structured mesh has not yet been reported before.

In addition to the interface capture, the numerical solution of incompressible heat flow using the pseudo-structured mesh, which is also a collocated grid, is necessary. Recently, the numerical procedures using primitive variables and the SIMPLE scheme for pressure/velocity coupling have been proposed for single-phase problems [27–29]. However, the solution of two-phase equations could be more difficult due to the large variation of the physical properties across the interface region, where a scheme for volume averaging is usually required. Furthermore, the large source term due to the heat of fusion and the viscous friction (the interactive force between two phases) [15, 19] of the interface region may degrade the convergence as well. Although a finite difference technique known as "immersed interface method" [23, 24] may not require the volume averaging, the use of discontinuous

coefficients is necessary. Hence, the computer programming with the mesh refinement is not so easy. Furthermore, the scheme becomes tedious as the order or accuracy is upgraded to two or higher. Indeed, as the sharp interface is treated as a diffusive interface, one needs to take care of the phase change and two-phase flow there, and they are considered in the source terms of the energy equation and the momentum equation, respectively. Nevertheless, the adaptive FVM scheme is believed to be attractive to the solution of multiphase flows having complicated interface morphology. In this study, we propose an iteration procedure starting from a thick interface. As the mesh refinement proceeds, the interface thickness is reduced gradually to approach the sharp interface limit. As will be discussed shortly, a variety of problems can be simulated easily through this approach. Again, such an implementation combined with AMR has not yet been found in the literature.

In the present report, an adaptive finite volume method (AMR/FVM) scheme using the fixed-grid approach for solidification problems is presented. A phase-field variable, based on a hyperbolic tangent function and thermal fields, is introduced first, so that the macroscopic phase change, where the interfacial energy is not important and the interactive force between solid and melt phases can be properly simulated. Through the adaptive mesh refinement and coarsening, both accuracy and efficiency of the model can be achieved. In addition to the phase change simulation, conjugated heat flow problems having complicated immersed boundaries can be easily simulated. When the capillary effect is important at the interface, the solidification temperature is no longer the equilibrium one from the phase diagram. This is typical for microscopic solidification, such as dendritic growth, where a proper phase-field variable satisfying thermodynamic constraints needs to be used in order to incorporate the contribution of the interfacial energy. In such case, the phase-field model [15, 25, 26] is a better choice for the phase-field variable. The present AMR/FVM scheme also perfectly fits into this category, especially for the dendritic growth. Sample calculations will be illustrated shortly. In the next section, a general physical model is described. The numerical scheme is discussed in detail in Section 3, where the concept of data structures and their computer implementation for AMR and error estimators are discussed. Section 4 is devoted to results and discussion, followed by conclusions in Section 5.

2. FORMULATION FOR PHYSICAL PROBLEMS

To model a general two-phase heat flow problem by the fixed-grid approach, the sharp interface (phase boundary) needs to be treated as a diffusive region first [19]. Inside the region, the physical properties are estimated by a mixing law through a phase-field variable ϕ. The phase-field variable is in general a continuous function ranging from 0 (liquid) to 1 (solid). In the context of the finite volume formulation, the mixing law may also be assumed to be valid inside the control volume (CV). For example, the velocity of the mixture can be estimated by $\boldsymbol{v} = (1-\phi)\boldsymbol{v}_l + \phi\boldsymbol{v}_s$, where $\boldsymbol{v}_l$ and $\boldsymbol{v}_s$ are the liquid and solid velocities, respectively. Other intrinsic properties can also be estimated in the same way [15, 19]. Furthermore, in this study, the solid is assumed to be stationary and rigid ($\boldsymbol{v}_s = 0$), so that further simplification can be made. The density of the liquid and solid is further assumed to be the same here, which makes the treatment of phase change much easier. The liquid is assumed to be incompressible and Newtonian, while the Bounissque's approximation is further adopted. By following the volume-averaging procedure [15, 19], the conservation equations for the mass, momentum, and energy can be derived as the following:

Equation of Continuity

$$\nabla \cdot (\rho \boldsymbol{v}) = 0, \tag{1}$$

where $\boldsymbol{v} = (1 - \phi)\boldsymbol{v}_l$ for $\boldsymbol{v}_s = 0$ and ρ is the average density.

Momentum Equation

$$\frac{\partial}{\partial t}(\rho \boldsymbol{v}) + \nabla \cdot (\rho \boldsymbol{v}\boldsymbol{v}) = \nabla \cdot \left(\mu_l \frac{\rho}{\rho_l} \nabla \boldsymbol{v} \right) - \nabla P + \rho \boldsymbol{B} + \frac{\rho}{\rho_l} \boldsymbol{F}, \tag{2}$$

where $\boldsymbol{B}$ is the body force, such as the gravitational acceleration $\boldsymbol{g}$, and $\boldsymbol{F}$ accounts the dissipative viscous force in the liquid due to interactions with the solid in the diffusive region [15, 19]; μ_l is the liquid viscosity and ρ_l the liquid density. Again, in the above equation the two-phase mixture is assumed to be Newtonian as well. For the interactive force $\boldsymbol{F}$, the approach proposed by Beckermann *et al*. [15] is adopted here, i.e.,

$$\boldsymbol{F} = -C \mu_l \phi \frac{\boldsymbol{v}_l}{\delta} |\nabla \phi|, \tag{3}$$

where C and δ are empirical constants. Beckermann *et al*. [15] used the analytical result of the Poiseuille flow between two plates to fit the model and got $C = 2.757$. Also, δ is related to the interface thickness. According to the definition in [15], the interface thickness can be taken to be about 6δ. For a thin interface (small δ), the constant C seems to be insensitive to the result if it is large enough. In addition, the phase-field variable ϕ needs to be specified. For a typical solid/liquid boundary, the form of hyperbolic tangent used in the phase-field simulation [15, 25, 26] can be adopted, i.e.,

$$\phi(d) = \frac{1}{2}\left[1 - \tanh\left(\frac{d}{2\delta}\right)\right], \tag{4}$$

where d is the normal distance from the interface. For the solidification of a pure material, if one does not want to introduce the phase-field equation, the normal distance d needs to be calculated efficiently. Of course, this is used for macroscopic solidification, where the capillary effect can be ignored. Because the isotherms are parallel to the interface near the solidification interface, one may take this into account for estimating the distance. For an equilibrium interface with the melting point T_m and with a normal unit vector $\boldsymbol{n} = \nabla T / |\nabla T|$, the distance to the interface can be estimated by $d = (T - T_m)/|\nabla T|$. Therefore, Eq. (4) can then be rewritten as

$$\phi(T) = \frac{1}{2}\left[1 - \tanh\left(\frac{T - T_m}{2\delta |\nabla T|}\right)\right]. \tag{5}$$

From this equation, $|\nabla \phi|$ can be derived, i.e., $|\nabla \phi| = |(\partial \phi / \partial T)\nabla T| = (\phi - \phi^2)/\delta$. Again, the above treatment is valid for an equilibrium interface with the solidification or melting temperature at T_m. This is quite true for macroscopic solidification, such as bulk crystal growth or casting. However, when the kinetic or capillary effect is important, such as the

dendritic growth, other approaches, such as the phase-field model, to define the phase-field variable are necessary. The final interactive force becomes

$$\boldsymbol{F} = -C\mu_l \left(\frac{\phi}{\delta}\right)^2 \boldsymbol{v}. \tag{6}$$

Equation (6) is the same as that derived by Beckermann *et al*. [15]. However, the phase-field variable defined by Eq. (5) allows us to find the phase-field variable directly from temperature rather than solving the phase-field equation. Indeed, for a macroscopic Stefan problem, where the interfacial energy is generally neglected, the present approach can be regarded as a modified enthalpy–porosity model. However, as compared with the traditional approach [8], the convergence characteristics are greatly improved.

Energy Equation

$$\frac{\partial}{\partial t}(\rho H) + \nabla \cdot (\rho \boldsymbol{v} H_l) = \nabla \cdot (k \nabla T), \tag{7}$$

where H is the enthalpy. In each phase i, $H_i = \int C_i \, dT + H_i^0$, $i = l$ or s, C_i is the specific heat, and H_i^0 is the reference enthalpy; the heat of fusion is assumed to be constant, i.e., $\Delta H = H_l - H_s$. In addition, k is the thermal conductivity of the mixture. Therefore, for a constant C_i and $H = \phi C_s T + (1-\phi)(C_l T + \Delta H)$, one can rewrite the energy equation as

$$\rho C_i \frac{\partial T}{\partial t} + \nabla \cdot (\rho \boldsymbol{v} H_l) = \nabla \cdot (k \nabla T) + \rho \Delta H \frac{\partial \phi}{\partial t}. \tag{8}$$

Clearly, the heat of fusion during solidification appears as a source term in the energy equation.

With the introduction of the Bounissque's approximation, the body force term $\rho \boldsymbol{g}$ can be rewritten as $\rho \beta_T \boldsymbol{g}(T - T_m)$; the static pressure is then absorbed into the pressure term; β_T is the thermal expansion coefficient. For the ease of discussion, the above equations are further transformed into dimensionless form after characteristic variables are chosen. In this study, we have chosen the system height (L) as the characteristic length and α_l / L as the characteristic velocity, where α_l is the thermal diffusion coefficient. Therefore, one can easily get the associated thermal Rayleigh number Ra and the Stefan number St as

$$Ra = \frac{\beta_T g \Delta T L^3}{\nu_l \alpha_l}; \quad St = \frac{C_l \Delta T}{\Delta H},$$

where ΔT is the temperature difference and ν_l the kinematic viscosity of the melt. Also, the Prandtl number $Pr = \nu_l / \alpha_l$.

The boundary conditions adopted in this study are quite simple. For the solid wall, the no-slip boundary condition, i.e., $\boldsymbol{v} = 0$, is used. For the dentritic growth in a forced convection in Section 4.3, the out-flow boundary condition is used for velocity, where the overall mass conservation is forced. The thermal condition is set to be either adiabatic or at a constant temperature. For time-dependent calculations, the zero velocity and the wall temperature are used as the initial condition.

3. NUMERICAL METHOD

To solve the previous equations, we have adopted a finite volume method using an arbitrary control volume (with an arbitrary number of faces) [27–29], even though we only consider the Cartesian mesh here. Due to the adaptive nature, we have purposely avoided the use of vertex (corner) points through a special interpolation scheme [29], which will be discussed in detail in the following sections. Because the adaptive mesh and its related data structures need to be constructed first before the numerical implementation, we shall discuss them first in Section 3.1. Also, their implementation requires a dynamic data structure based on a quadtree for refinement (the cell subdivision) through the pointers of FORTRAN90 and the derived types. We give an example of the programming there as well. Section 3.2 is devoted to the core FVM approximation. Since the primitive variables are used, the velocity/pressure coupling using collocated grids and the SIMPLE scheme are adopted and will be described briefly in Section 3.3. The numerical solution of the linearized equations during each nonlinear iterations is discussed in Section 3.4, followed by the error estimators and the solution procedure in Section 3.5.

3.1. Adaptive Mesh Refinement Algorithm

The solution of previous equations is carried out on a rectangular domain with an initial mesh (defined as the first level or the root level), as shown in Fig. 2a as an example. At this level, the finite volumes are very coarse having only three cells. As the calculation proceeds, mesh refinement or coarsening (not on the root level) needs to be carried out depending on the predefined criteria, which can be set by the geometry, error estimators, or variable

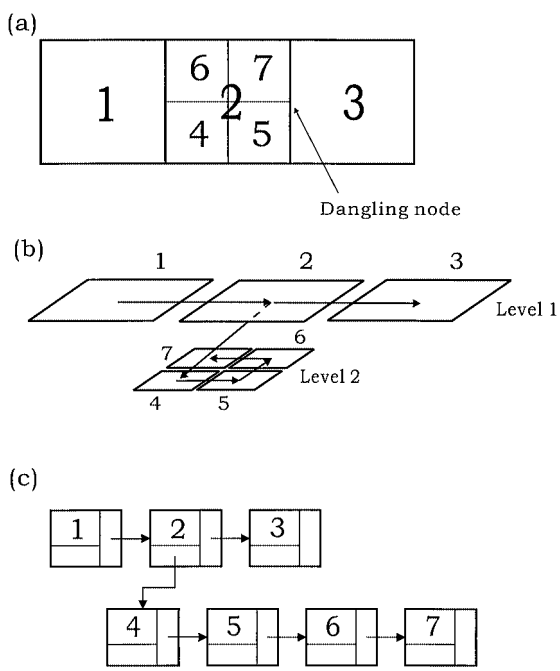

FIG. 2. Schematic of mesh refinement and data structure: (a) mesh, (b) topological view of refinement, (c) data structure.

gradients, etc. For example, if one finds the second cell needs to be refined, the second level is added and new four cells are born, namely the cells 4 to 7, as shown in Fig. 2b. Since these mesh are used for control volumes, the variables are stored at the geometric center of each cell. Therefore, the dangling node is not used to store the data there. The corresponding data structure is then constructed as shown in Fig. 2c, and it is a typical quadtree. Each cell carries its own information including the node identification (ID), level, coordinates, and related properties. Therefore, to implement AMR efficiently, a careful design of the data structure and search schemes (finding itself and neighbors) becomes crucial. In addition to refinement, which adds cells and data, coarsening that deletes cells and data is necessary. One may use FORTRAN77 to design such a data structure through pseudodynamic memory, which needs to declare a large array, as well as a linked list. However, removing cells and data during coarsening becomes troublesome. To avoid the waste of memory, one would need a special data collapsing operation. Therefore, the use of pointers and dynamic data structures as well as the recursive features, which are available in C++ and FORTRAN90, makes the design of the data structure much easier. As shown in Fig. 2c, the data structure turns out to be very simple and each cell has its own ID and requires only two pointers; one points to its next cell at the same level and one points to its first kid in the next level. In this study, we have adopted FORTRAN90 to develop the data structure and perform related numerical calculations. FORTRAN90 and C++ have very similar features. However, there are still some problems that need to be resolved. For example, there are no pointer arrays available in FORTRAN90. Therefore, further using derived data types for arrays is necessary. Nevertheless, the derived data types and the modular design in FORTRAN90 are very suitable for the FVM implementation. More importantly, the concept of object-oriented programming can be further adopted, which makes the coding and debugging much easier.

After the data structure (e.g., defined by a new type called CellTree) is developed, a searching scheme is also important, which includes the way for traveling in the quadtree and for finding cell neighbors. The depth first search (DFS) is further adopted. For each parent node, DFS goes through the tree starting from the root level until the youngest level (without any kid cells) is reached. A sample code (traverse_demo) is illustrated in Fig. 3 for reference. As shown in the main program, the cell_tree is a module to define all the data related to the cell, such as the temperature, velocities, etc. Also, C is a new derived type called CellTree, and so a node. The nodes start and current are pointers; the node start points to ($\Rightarrow$) the cell StartCell, then calls DFStraverse to perform DFS. As shown in the subroutine, the subroutine DFStraverse for DFS needs to be recursive (it can call itself). In DFStraverse, one needs to check if the next cell is null (empty) or not. If not, the search continues by calling DFStraverse (itself) until the youngest cell is reached. Once this search scheme is set up, the most laborious task is to find neighbor cells. There are several approaches that can be used. If one can avoid any floating point operations, the neighbor search can be very fast. After it is finished, the coding for FVM is as easy as that on a simple structured mesh.

3.2. Finite Volume Method

The general form of the previous conservation equations can be written as

$$\frac{\partial \varphi}{\partial t} + \boldsymbol{v} \cdot \nabla \varphi = \nabla \cdot \Gamma \nabla \varphi + S_{\varphi}, \tag{9}$$

```
program traverse_demo
use cell_tree                                   ! use module cell_tree
implicit none

type(CellTree):: C                              ! C is a CellTree type
type(node), poiner:: start, current             ! nodes start and current are pointers
      …
      start=>C%StartCell                        ! points to the StartCell of CellTree C
      call DFStraverse(start)                   ! call DFS from node start

stop
end program traverse_demo


!---------- DFS subroutine ----------------------------------------------------

recursive subroutine DFStraverse(current)       ! recursive subroutine
use cell_tree
implicit none

type(node), pointer:: current

do while( associated(current) )
     if( associated(current%nextlevel) ) then   ! check if the next level is null
          call DFStraverse(current%nextlevel)   ! if not, call DFStraverse
     else
          write(*,*) current%data
     end if
     current=>current%nextcell                  ! search for the next cell at the same level
end do
return

end subroutine DFStraverse
```

FIG. 3. A sample FORTRAN90 code for the DFS scheme.

where φ represents the velocities and temperature, respectively; Γ is the diffusivity; and S_φ is the source term. One can also represent it by an integral form, which is more suitable for the finite volume approximation, i.e.,

$$\frac{\partial}{\partial t}\int_V \varphi\, dV + \int_V \boldsymbol{v}\cdot\nabla\varphi\, dV = \int_V \nabla\cdot\Gamma\nabla\varphi\, dV + \int_V S_\varphi\, dV, \tag{10}$$

where V is the control volume (CV). One can apply Eq. (10) to each cell, such as the cell P in Fig. 4, to get the discretized form of the conservation equations. The various terms involved in the FVM approximation are discussed in detail as follows:

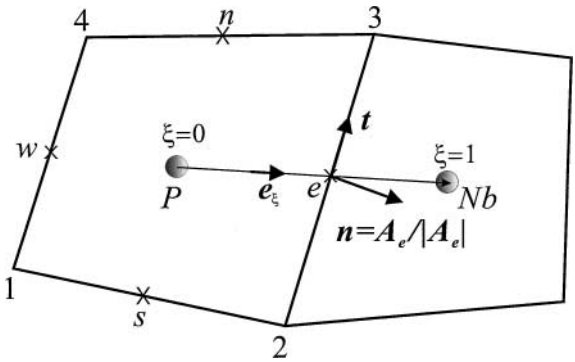

FIG. 4. Schematic of finite volume cells P and Nb and related notations.

Convective Term

Using Gauss's divergence theorem, the volume integration of the convective term can be transformed into a surface integral resulting in a simple convective flux balance of variable φ:

$$\oint \varphi \boldsymbol{v} \cdot \boldsymbol{n}\, dS \cong \sum_{c=1}^{4} \sum_{i=1}^{n(c)} \varphi_i m_i S_i, \tag{11}$$

where φ_i is the variable defined at cell face i, m_i the mass flux, and S_i the surface area of cell face i. In the above expression, the cell faces are grouped into four sides ($c = 1$, 2, 3, and 4), which corresponds to the orientation labeled by s, e, n, and w shown in Fig. 4. After refinement, the neighbor cells may be divided, and in each side the number of faces increases to $n(c)$. Although we have used a Cartesian grid for the study, in principle, the CVs are not restricted to the quadrilateral cells. Other types of cells can be used. As will be mentioned shortly, to maintain the second-order of accuracy, the variables used for the mass flux are evaluated at the cell faces. To stabilize the scheme, a special treatment, such as deferred correction [27, 28], is necessary.

Diffusive Term

Again, Gauss's divergence theorem is applied, i.e.,

$$\int_V \nabla \cdot \Gamma \nabla \varphi\, dV = \oint \Gamma \nabla \varphi \cdot \boldsymbol{n}\, dS \cong \sum_{c=1}^{4} \sum_{i=1}^{n(c)} \Gamma \nabla \varphi \cdot (\boldsymbol{n}S)_i. \tag{12}$$

However, in above equation, there are several ways to approximate $\nabla\varphi$. For example, a straightforward approximation can be derived easily from simple vector calculus. Taking face e in Fig. 4 as an example,

$$\nabla \varphi \cdot (\boldsymbol{n}S)_e = \nabla \varphi \cdot \boldsymbol{A}_e = \frac{\varphi_{Nb} - \varphi_P}{L_{PNb}} \frac{\boldsymbol{A}_e \cdot \boldsymbol{A}_e}{\boldsymbol{A}_e \cdot \boldsymbol{e}_\xi} + \frac{\varphi_3 - \varphi_2}{S_e} \frac{\boldsymbol{A}_e \cdot \boldsymbol{A}_e}{\boldsymbol{A}_e \cdot \boldsymbol{e}_\xi} \boldsymbol{t} \cdot \boldsymbol{e}_\xi, \tag{13}$$

where L_{PNb} is the distance between nodes P and Nb, $\boldsymbol{t}$ the tangential unit vector at cell face e, and $\boldsymbol{e}_\xi$ the unit vector in the ξ direction. Also, $\boldsymbol{A}_e$ is the surface vector, S_e the surface area,

φ_P the variable value at the cell center, φ_{Nb} the variable value at the neighbor cell, and φ_2 and φ_3 are the values at two corners (φ_2 and φ_3) of cell face e. Unfortunately, this formulation requires the variable at the corners to be evaluated, which is particularly troublesome when the neighbor cell Nb is refined. Therefore, another formulation using the cell values only is adopted [29], i.e.,

$$\nabla\varphi \cdot \boldsymbol{A}_e = \frac{\varphi_{Nb} - \varphi_P}{L_{PNb}} \frac{\boldsymbol{A}_e \cdot \boldsymbol{A}_e}{\boldsymbol{A}_e \cdot \boldsymbol{e}_\xi} + (\nabla\varphi)_e \cdot \left(\boldsymbol{A}_e - \boldsymbol{e}_\xi \frac{\boldsymbol{A}_e \cdot \boldsymbol{A}_e}{\boldsymbol{A}_e \cdot \boldsymbol{e}_\xi} \right). \tag{14}$$

Although the above approximation does not require the vertex values, the approximation of the gradient at the cell face (i.e., $(\nabla\varphi)_e$) is necessary. Two approaches are adopted here. The first one is to evaluate the values of $\nabla\varphi$ at the center of the related cells and then the value at the cell face is interpolated linearly from its adjacent cells. The second approach is to find the best fit by the least-square method from all neighbor cells [27]. Both approaches work well in this study, but the former approach takes less effort in computation.

Body Force

Finally, the body force is evaluated by simply assuming that the variable at the cell center is at its mean value

$$\int_V S_\varphi \, dV = (S_\varphi)_P \Delta V, \tag{15}$$

where ΔV is the cell volume.

The above approximations give the final discretization a second-order accuracy and have been used for all equations except the equation of continuity, which needs special attention as discussed in the next section.

3.3. Velocity/Pressure Coupling

Since the pressure variable does not appear explicitly in the continuity equation, the use of linearly interpolated velocities at cell faces for a collocated grid could lead to the velocity/pressure decoupling or the so-called checkerboard oscillation of pressure. In order to amend this, the idea of Rhie–Chow momentum interpolation scheme [32] is adopted. In other words, the velocity values required (at cell faces) for the continuity equation are interpolated from the momentum equations, rather than linearly from the adjacent nodal values.

The simplest way of finding a correct pressure field to satisfy the equation of continuity is through the SIMPLE scheme [33], where the pressure correction is used to find the velocity correction through the momentum equation. Detailed description of the scheme can be found elsewhere [27–29, 33, 34]. In brief, based on the SIMPLE scheme one can obtain the equation for velocity correction, by ignoring the off-diagonal terms, as

$$\boldsymbol{v}'_P = -\frac{\Delta V}{A_P} \nabla P'_P, \tag{16}$$

where $\boldsymbol{v}'_P$ and P'_P are the correction velocity and pressure, respectively, and A_P is a diagonal constant obtained from the linearized equation of the momentum equations. The new velocity and pressure can be updated by $\boldsymbol{v}_P + \boldsymbol{v}'_P$ and $P_P + P'_P$. However, as one wants

to substitute the velocities into the equation of continuity to get the equation for pressure correction, the values at cell faces are necessary. Unfortunately, the interpolation directly from adjacent nodes does not introduce the actual pressure gradient there. The so-called "2-δ" pressure difference causes a decoupling at even and odd nodes. Instead, one would require the pressure values right at the both sides of the cell face (the so-called 1-δ gradient). Therefore, the idea of Rhie and Chow [32] is to replace the original 2-δ pressure gradient by the 1-δ one. By doing so, we obtain the volumetric flow rate J_f^* at cell face f as

$$J_f^* = \boldsymbol{v}_f^* \cdot \boldsymbol{A}_f - \frac{(\Delta V_P + \Delta V_{Nb})}{(A_{P,P} + A_{P,Nb})} \left(\frac{P_{Nb} - P_P}{L_{PNb}} - (\nabla P)_f \cdot \boldsymbol{e}_\xi \right) \frac{\boldsymbol{A}_f \cdot \boldsymbol{A}_f}{\boldsymbol{A}_f \cdot \boldsymbol{e}_\xi}, \tag{17}$$

where $\boldsymbol{v}_f^*$ is the velocity obtained by linear interpolation from the nodes P and Nb, and A_P is again a diagonal coefficient. Putting the volumetric flow rate back into the equation of continuity, the equation for the pressure correction can be obtained:

$$\sum J_f^* = \sum_f \frac{(\Delta V_P + \Delta V_{Nb})}{(A_{P,P} + A_{P,Nb})} \frac{P'_{Nb} - P'_P}{L_{PNb}} \frac{\boldsymbol{A}_f \cdot \boldsymbol{A}_f}{\boldsymbol{A}_f \cdot \boldsymbol{e}_\xi}. \tag{18}$$

The above equation can then be reformulated into a set of linearized equation for each cell

$$A_P^P P'_P - \sum_{Nb} A_{Nb}^P P'_{Nb} = b_P^P, \tag{19}$$

where $A_{P \text{ or } Nb}^P$ are the related coefficients and the source term b_P^P is related to the residual volumetric flux $\sum J'_f$. After the pressure is corrected, the correction velocity is obtained by Eq. (16). The velocity values can then be corrected for the next iteration. Once the iteration converges, the equation of continuity is satisfied.

3.4. Solution of Linearized Equations

The implicit backward difference scheme is further adopted for the time derivative. By summing all the terms, one can obtain the linearized equation, similar to the pressure correction, for each cell

$$A_P^\varphi \varphi_P - \sum_{Nb} A_{Nb}^\varphi \varphi_{Nb} = b_P^\varphi, \tag{20}$$

where the coefficients are frozen during linear iterations. In order to retain second-order accuracy, while keeping the linear iteration easier to solve, the coefficients obtained by upwinding for the convection terms are used on the left-hand side of Eq. (20), while the central-differencing terms are moved to the right-hand side. On the right-hand side, the upwinding terms are also added to make equations exact as they converge. This is the so-called deferred correction [27, 28] and has been used widely in previous calculations using structured grid. The solution of the linear equation set takes most of computing effort in the simulation. An efficient way for solving it is crucial. However, since the matrix losses its structured pattern after mesh refinement, the use of traditional solvers, such as SIP (strong implicit procedure [35]), is not feasible. Therefore, two types of solvers are examined. One is the Gauss–Siedel (GS) method and one is the ILU(0) preconditioned conjugate gradient square (CGS) method. In general, CGS provides a much faster convergence. However, it

takes much more computer memory. A trade-off is to use diagonal scaling instead of ILU(0) for the preconditioner, but with less robustness. Furthermore, we found that the biggest difference is in the convergence of the continuity equation. Take the heated square problem [36] as an example; $\mathrm{Ra} = 10^5$ and $\mathrm{Pr} = 0.71$. The convergence of ILU(0)-preconditioned CGS takes less than 20 iterations, while the GS solver takes more than 250 iterations, for three orders of reduction in the residual. Since we have the multilevel grids and data structures being used, multigrid methods [e.g., 34, 36] can be adopted as well. However, in the present calculations, the single-grid solver is adequate. Indeed, for 3D applications, multigrid methods should be considered seriously.

3.5. Error Estimation and Solution Procedure

To implement AMR efficiently, the criteria for mesh refinement or coarsening are important. The geometric criteria can be easily implemented. As one gives the distance to the interfaces or the minimum cell size there, the level of grids can then be determined for refinement or coarsening. On the contrary, estimating numerical errors is much more troublesome. The most straightforward way is through the Richardson extrapolation if the solutions at different levels of grids are available. However, this approach is not quite robust, because one has to ensure the convergence at different levels of grids. Therefore, we need error estimators here that do not require the multigrid solutions. Moreover, the estimators need to be robust and take little computational effort. As will be discussed shortly, even with a robust error estimator, getting a cost-effective solution still requires some trial and error. Therefore, in practice a single error estimator is not adequate. Several error estimators are required for choice.

Typically, the calculated local errors are proportional to the gradients of variables. Therefore, we have first adopted the normalized gradients as one of the error estimators. In most cases, such an estimator is adequate. Since the variable gradient is a by-product of the calculation for each cell, it can be regarded as a good error estimator. The second approach, which uses the normalized truncated error, is also quite easy to compute. Take a simple cell, shown in Fig. 4, for example. For getting a third-order variation of the variable φ from nodes P to Nb, we can use the boundary conditions from the two contiguous cells as

$$\begin{aligned} \xi = 0: &\quad \varphi = \varphi_0; \quad (\nabla\varphi)_0^{\xi} = (\nabla\varphi)_0 \cdot \boldsymbol{\xi}, \\ \xi = 1: &\quad \varphi = \varphi_1; \quad (\nabla\varphi)_1^{\xi} = (\nabla\varphi)_1 \cdot \boldsymbol{\xi}. \end{aligned} \tag{21}$$

The variable variation along ξ is then represented by a cubic form:

$$\varphi(\xi) = C_0 + C_1\xi + C_2\xi^2 + C_3\xi^3, \tag{22}$$

where the constants C_i $(i = 0, 3)$ can be determined from the boundary conditions in Eq. (21). The new face values, $\tilde{\varphi}_f$ and $(\nabla\varphi)_f^{\xi}$, calculated by the above equation can then be used for the flux estimation. The new convective ($\tilde{F}_C$) and diffusive ($\tilde{F}_D$) fluxes can be written, respectively, as

$$\tilde{F}_C = m_f\tilde{\varphi}_f; \quad \tilde{F}_D = -(\nabla\varphi)_f^{\xi}(\boldsymbol{\xi} \cdot \boldsymbol{n}), \tag{23}$$

where m_f is the mass flux. The total difference between the new fluxes (based on the

third-order approximation) and the original ones (second-order) is obtained:

$$\tau = \sum_f \lfloor (\tilde{F}_C - F_C) + (\tilde{F}_D - F_D) \rfloor. \tag{24}$$

Because this value also depends on the cell volumes as well as face areas, a normalized value should be used. A typical way is to normalize the value by dividing it with $A_P^{\varphi}\varphi_P$. One can further use the new fluxes to calculate the variable (φ_p^{new}). More importantly, the error ($e = \varphi^{new} - \varphi$) also satisfies the following equation [28]:

$$A_P^{\varphi} e_P - \sum_{Nb} A_{Nb}^{\varphi} e_{Nb} = -\tau. \tag{25}$$

Since the coefficients in the above equations are already available after each iteration, the estimation of the error requires little effort to compute. In fact, for most of the situations (diagonally dominant cases), τ is proportional to e, but e is smoother than τ. Accordingly, the quality of refinement by both values is about the same in all cases here.

In summary, the whole solution procedure is illustrated in Fig. 5. The SIMPLE iterations also include the calculation of the phase-field variable. Once the iterations converge (or near),

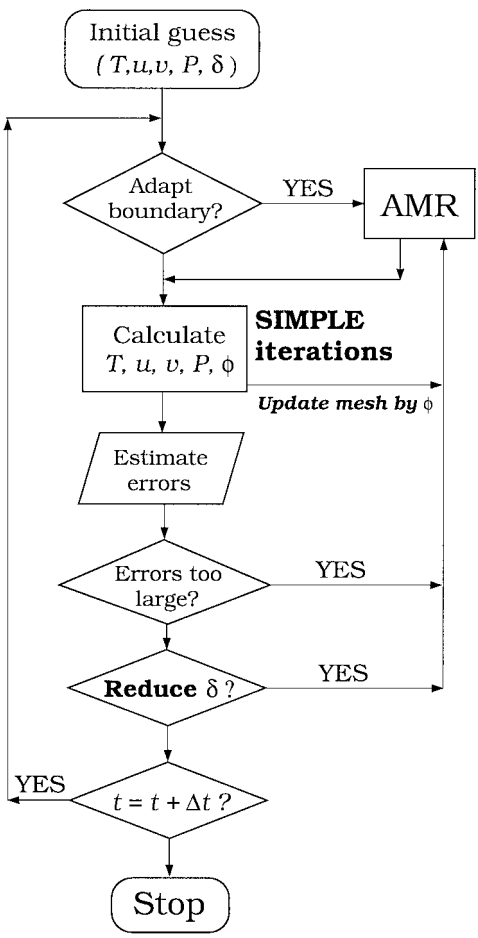

FIG. 5. Iteration scheme of the adaptive finite volume method; the adaptive mesh refinement (AMR) also updates variables by interpolation or averaging, geometry related quantities, and data structures.

the mesh refinement then adapts to the interface according to the phase-field variable. Because the interface thickness (6δ) should smear several cells for a reasonable solution, its value at the beginning needs to be large enough for convergence. Once the solution gets better and the interface position is located, its value is further reduced. Through such a procedure, as the solution converges, the sharp interface limit can be reached as well. Furthermore, for time-dependent calculations, the solution procedure is the same as the steady-state one at each time step. Therefore, the same program can be used for both cases. Nevertheless, the refined area cannot be too thin if one wants to use a bigger time step for calculation. For phase-field simulation, the time step used is small, so that the interface thickness can be very thin and this reduces the cell number significantly.

It should be pointed out that there is no a simple way to obtain a cost-effective solution (a better solution with less effort). Take the heated square [36] as an example again for $\mathrm{Pr} = 0.71$ and $\mathrm{Ra} = 10^5$. The isotherms are shown in Fig. 6a and the calculated errors for Nusselt number (ΔNu) based on two different refinement criteria are shown in Fig. 6b. In Fig. 6b, the solution based on uniform meshes is also shown (the straight line), and the mesh-independent solution is obtained from the Richardson extrapolation using the finest two levels of grids. As shown by the convergence slope for the uniform mesh, it shows a nice second-order of accuracy; the slope based on cell number is -1. The refinements by temperature and velocity errors show very different behaviors. The refinement based on velocity is more effective at the beginning, but it degrades as the cell number increases. The refinement based on temperature gives a more consistent result; the error decreases as the cell number increases. If we use three levels of grids for calculation, the final meshes for refinement based on temperature and velocity are shown in Fig. 6c, respectively, on the left (1042 cells) and the right (946 cells); the initial error counters are shown by the dashed lines. As the refinement continues, the error reduces and becomes more uniform. As a result, its contours show many local maximum values. Therefore, to avoid confusion, we only illustrate the initial errors here (before refinement is performed). As shown, the refinement based on temperature errors tends to allocate cells at the place with higher thermal gradients. The refinement based on velocity is slightly different, but still having more cells inside the velocity boundary layer near the wall. Interestingly, as shown in Fig. 6b for velocity, as the cell number is greater than 1000, the error with more refinements tends to approach to the result of the uniform mesh. As we further examine the meshes at two refinement criteria for velocity shown in Fig. 6d, indeed the further refined grid tends to be uniform (on the right having 4678 cells). Therefore, the result for velocity refinement is reasonable that the error with refinement is still less than that with a uniform mesh for the same number of cells. However, it is just hard to make a decision where to stop the refinement because the further refinement does not guarantee the decrease of the error (if ΔNu is the error indicator). Therefore, in general, it is difficult to define a universal rule for a cost-effective refinement. Some trial and error is still inevitable. Regardless all the troubles, for solidification applications, the refinement can be simply carried out on the diffusive interface region, and significant saving in CPU time can be expected. If the initial refinement period is ignored, in the present case, the CPU time is in the order of $(\text{No. of CVs})^{1.8}$ for both uniform and nonuniform grids after the grid is fixed, which is comparable to the previous study [34]. The overhead for the unstructured mesh is very little. For the case of 2,500 cells, it takes less than 2 minutes in a Pentium-III/800 MHz personal computer for the residuals of temperature and velocity to be reduced five orders of magnitude.

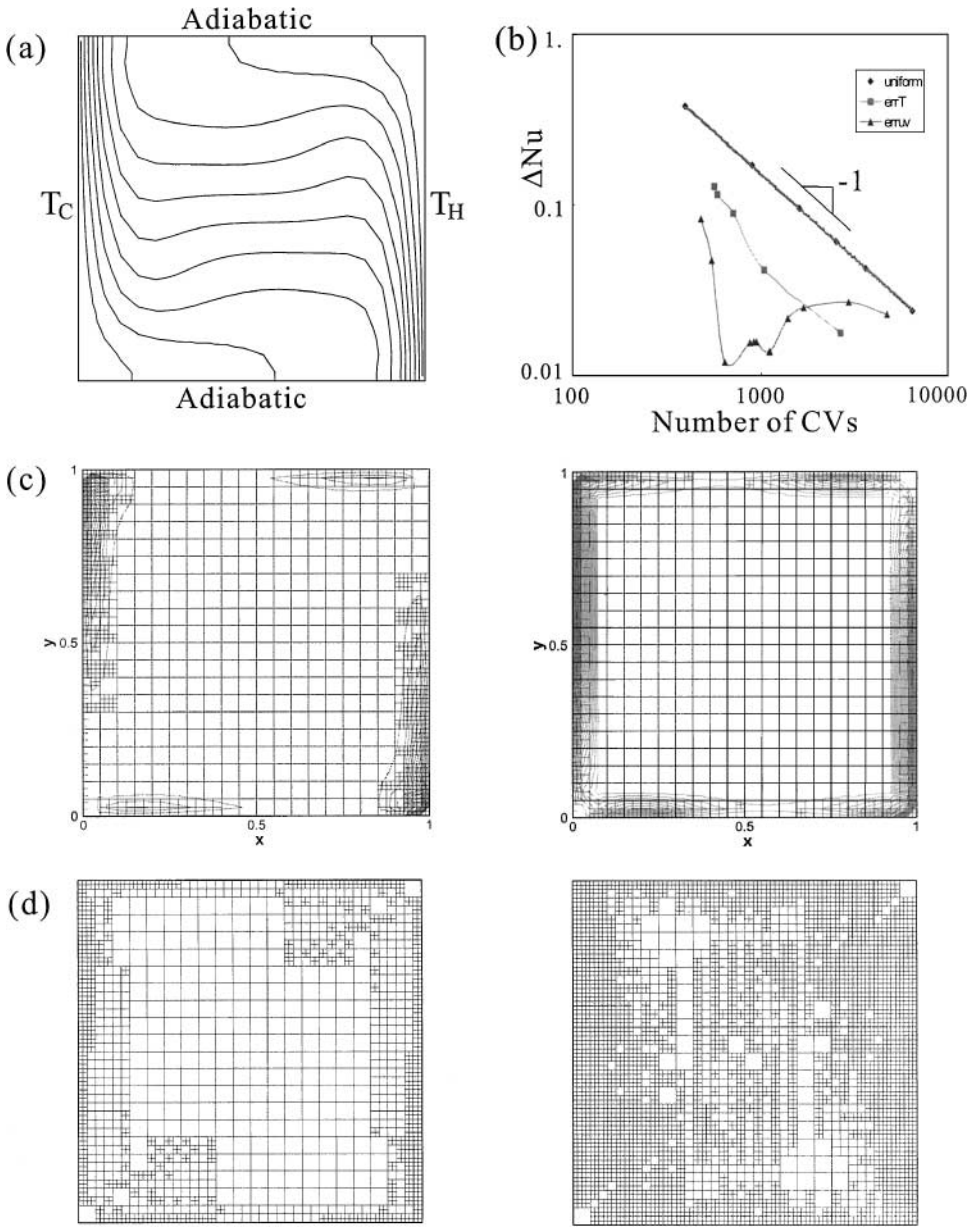

FIG. 6. Sample refinement results for heated square: (a) calculated isotherms based on a coarse mesh (spacing of isotherms = 0.09091); (b) convergence of the average Nusselt number for different refinement schemes; after the mesh is fixed, CPU time scales about (No. of CVs)$^{1.8}$; CPU time is about 2 min for 2500 CVs (Pentium-III/800 MHz). (c) meshes (1042 CVs on the left and 946 CVs on the right with three levels) and errors for temperature (left) and velocities (right); (d) the meshes for two refinement thresholds: 0.3 (left, 1678 CVs) and 0.1 (right, 4678 CVs) on the velocity.

4. RESULTS AND DISCUSSION

Before applying the AMR/FVM scheme to solidification, which contains a free or moving interface, we have tested first the method using problems having complex immersed bound-

aries in the domain in Section 4.1. As will be shown, as the thickness of the diffusive interface is small enough, the calculated solution based on the two-phase model agrees very well with the body-fitted one. In Section 4.2, the calculation is then performed for solidification problems using the modified enthalpy–porosity model, which is applied for macroscopic solidification that the interfacial energy contributes very little to the solidification temperature. The final examples are illustrated in Section 4.3 for the microscopic solidification using the dendritic growth as examples, where the interfacial energy (the local curvature of the interface) plays an important role. In such cases, the phase-field equation is considered so that the interface curvature and thickness can be determined implicitly. As will be illustrated, the simulation could be quite cost-effective as compared with that by a uniform mesh.

4.1. Applications on Problems with Fixed Complex Immersed Boundaries

The first example is illustrated again for the heated square, but with a highly conductive sphere inside (100 times higher in the thermal conductivity), as illustrated in Fig. 7a. The

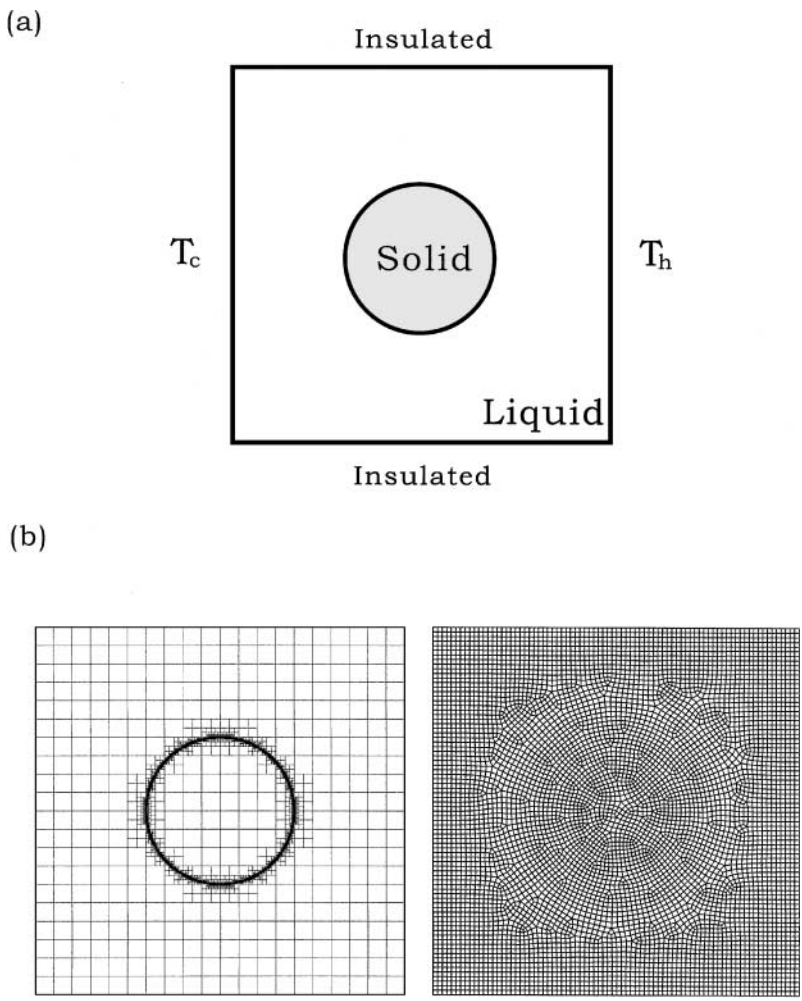

FIG. 7. (a) Schematic of a highly conductive sphere in a heated square; (b) initial AMR mesh having 5776 cells (6 levels) and $\delta = 5 \times 10^{-4}$ (left) and body-fitted mesh (right).

calculated result using Fluent, a commercial FVM code based on a body-fitted coordinate, is used for benchmarking. Figure 7b shows two typical meshes used for calculation; the left one is an initial mesh (six levels in the diffusive boundary) based on AMR/FVM, while the right one is an unstructured body-fitted coordinate generated by Fluent. The size of the finest grid here is 1/640. The comparison of isotherms is shown in Fig. 8a for $\delta = 5 \times 10^{-4}$; the interface thickness is defined by 6δ being 3×10^{-3} [15]. As shown, the agreement near the sphere is reasonably good. However, at the upper and lower boundaries, the errors are not trivial due to the coarse cells used there. To further refine the solution, the local refinement is performed based on velocity errors (only four levels of grids are used for the refinement, while six levels are used in the diffusive interface). As shown on the left-hand side of Fig. 8b,

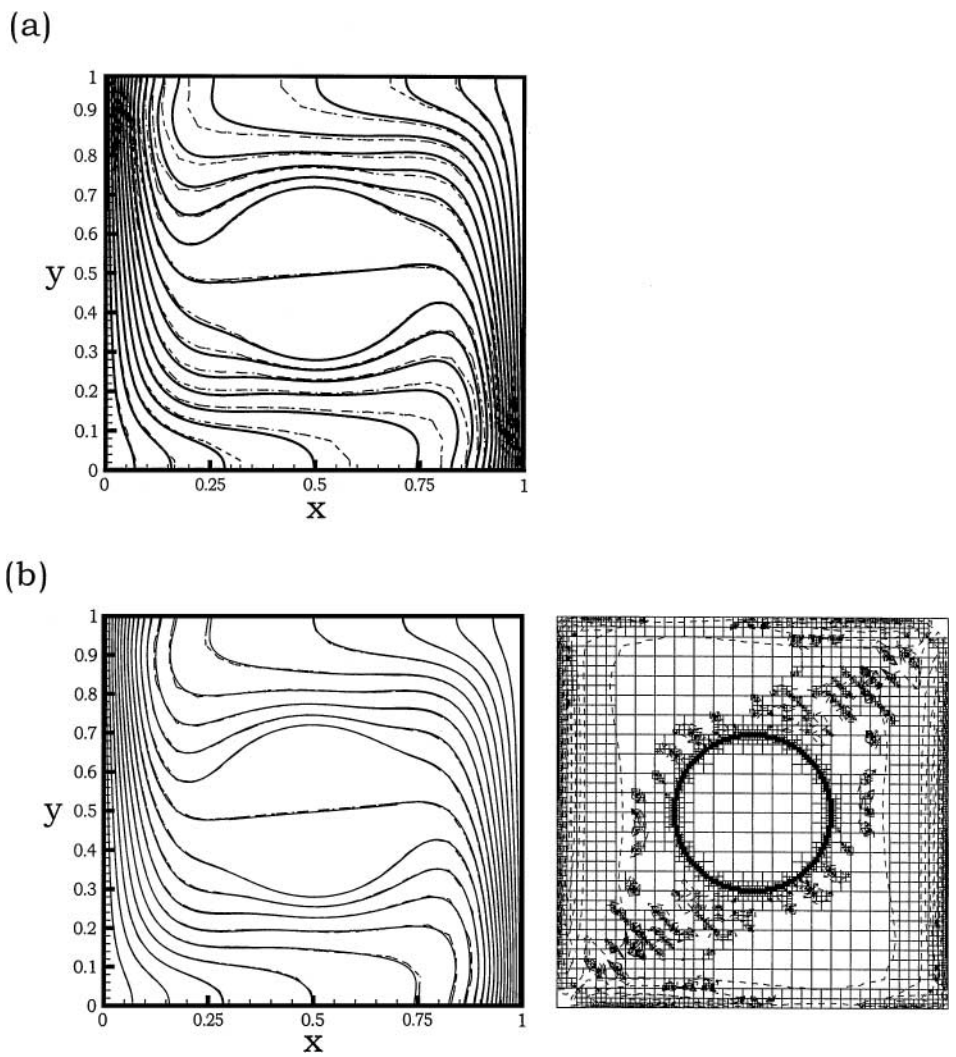

FIG. 8. (a) Comparison of calculated isotherms based on the meshes in Fig. 7b; (b) comparison of isotherms (left) using the final refinement mesh (right) based on error estimators (error contours (refinement criteria: $|e| = 0.4$ and 4 levels) before each refinement are shown by the dashed lines with the mesh) and interface thinning; the solid lines are obtained by fluent and dashed lines by the present code. Spacing of isotherms $= 0.05$ and $\delta = 5 \times 10^{-4}$.

excellent agreement is obtained for the whole domain. The final mesh (10,150 cells with six levels of grids) is shown on the right-hand side of Fig. 8b; all the error contours during refinements also shown by the dashed lines. As shown, more cells are allocated in the region with larger errors. At the initial stage (with an uniform mesh), the errors appear mainly near the boundaries and the maximum value is about ± 11.08. As the refinement continues, the magnitude of the errors decreases and the errors distribute more uniformly. At the final stage before the last mesh refinement is taken, the maximum error (near the upper right and lower left regions) is in the order of 0.4. Therefore, the error contours shown there are for $|e| = 0.4$. In this calculation, the coarsening stage is removed. Also, we do not restrict the level difference for adjacent cells. Therefore, in some places, one may fine the larger level difference as high as 3. It is believed that the discretization error increases as the level difference increases; one may find the interpolation by distance at the cell face becomes unrealistic. On the other hand, restricting the level difference between contiguous cells to 1 improves the accuracy and the cells are well distributed, but one may need more cells. Therefore, there is always a tradeoff there that needs to be determined by the users, and it is problem dependent. As will be illustrated later for the phase-field simulation, to maintain smooth isotherms, the level difference for adjacent cells is restricted to one. In such a case, the grid distribution is much smoother.

In this example, the interface thickness (6δ) plays an important role in the solution accuracy. We have examined several δ values, and found that it needs to be small enough for getting a reasonable result. In other words, without refinement, one needs to use a large number of cells for calculation. Figure 9a shows the distribution of the phase-field variable (left) and the x-component (u) of the velocity (right) near the interface at $x = 0.5$ for $\delta = 0.01$. Apparently, as the grid is refined, the solution converges but the velocity decays slowly across the solid; in reality, it should be zero in the solid due to the no-slip condition. In order to make the solution more realistic for the sharp interface, the thickness of the interface needs to be further reduced. Figure 9b shows the effect of the interface thickness on the velocity and temperature profiles near the interface. As shown, as the interface thickness is reduced, the solution becomes more realistic. This is the reason for using a very small δ value in the benchmark comparison in Fig. 8. If one uses a uniform mesh while using such a small δ value, the cell number increases linearly with the domain area, while in the AMR scheme it increases with the arclength of the immersed boundaries. This is the key to make AMR successful in the calculation.

Other criteria can also be used for refinement, but the results are similar. Again, one may further restrict the level difference between contiguous cells to be one or two, which usually makes the calculated isotherms smoother. Presumably, the accuracy may be further improved. For the phase-field simulation of dendritic growth, we have found that this criterion seems to be quite important.

Because of the use of the fixed-grid approach, the complexity of the immersed boundaries can be arbitrarily increased. For example, as shown in Fig. 10a, the number of spheres is increases to 18. As shown, the mesh based on the refinement near the interface and velocity error allows us to use a reasonable number of cells (27,472 cells with seven levels of grids) to get a result shown in Fig. 10b, where the velocity vectors and the isotherms (dashed lines) are presented. A further extension of the problem for simulating the heat flow in a porous media with a known solid structure is thus feasible. It should be pointed out that the cut-cell approach proposed by Ye *et al.* [22] could also be useful for calculating the viscous flow with complex immersed boundaries. Because the structured grid was used, they required

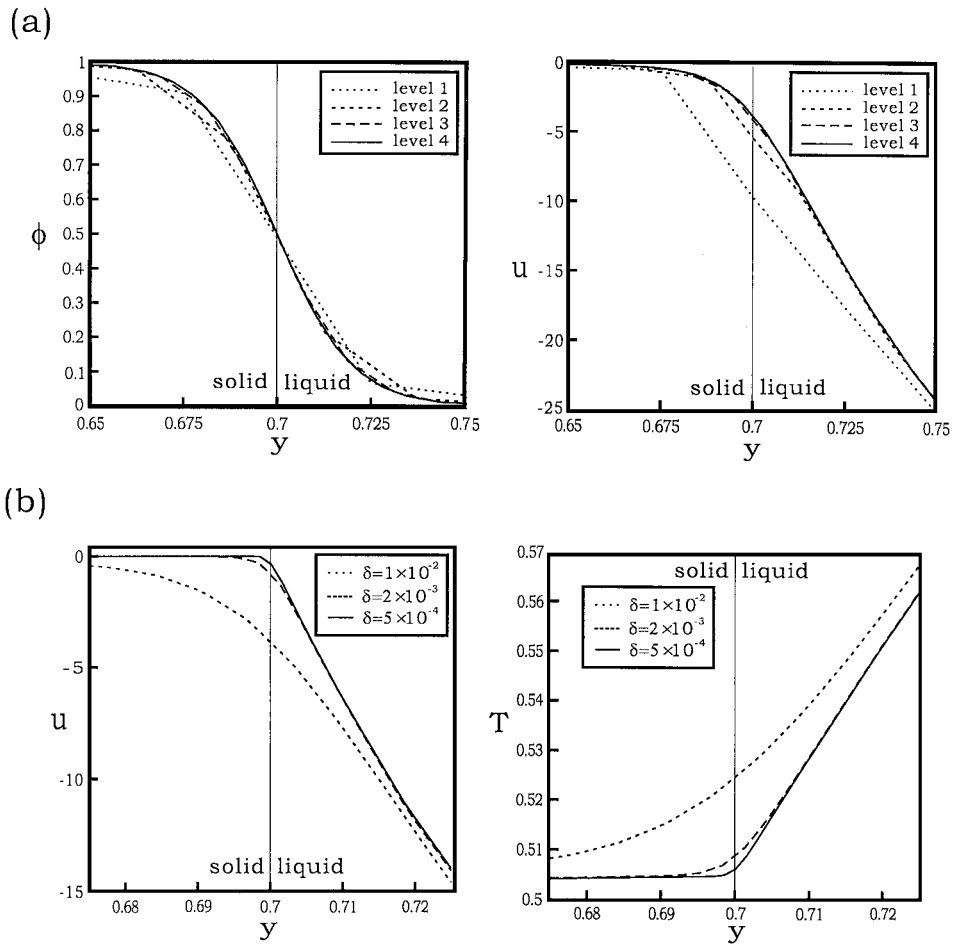

FIG. 9. (a) Effects of grid levels on the distribution of the phase-field variable (left) and the x-component velocity (right) across the interface at $\delta = 0.01$; (b) the effects of interface thickness (6δ) on the distribution of velocity (left) and temperature (right) across the interface.

a large number (256×256) of cells for calculation. Therefore, through AMR/FVM, it is clearly that the problems with complex boundaries can be easily treated. In addition to the refinement on the interfaces or boundaries, allocating cells based on numerical errors, or wherever it needs, is also helpful for further improving the accuracy.

4.2. Applications on Free or Moving Interface Problems

The second category of the simulation is devoted to macroscopic solidification problems. The first example is the heat flow and the interface shape in a horizontal Bridgman crystal growth, which is a popular process for growing compound semiconductor crystals. A typical configuration is illustrated in Fig. 11a; the crucible is neglected. The material is kept in a molten state on the left (with the temperature T_H higher than the melting point T_m), while on the right, the heat is removed for solidification (with a cooler temperature at T_C). Due to extremely low growth rate, the interface movement is neglected. For a stationary state with constant T_H $(= 1)$ and T_C $(= 0)$ the position and the morphology of the interface

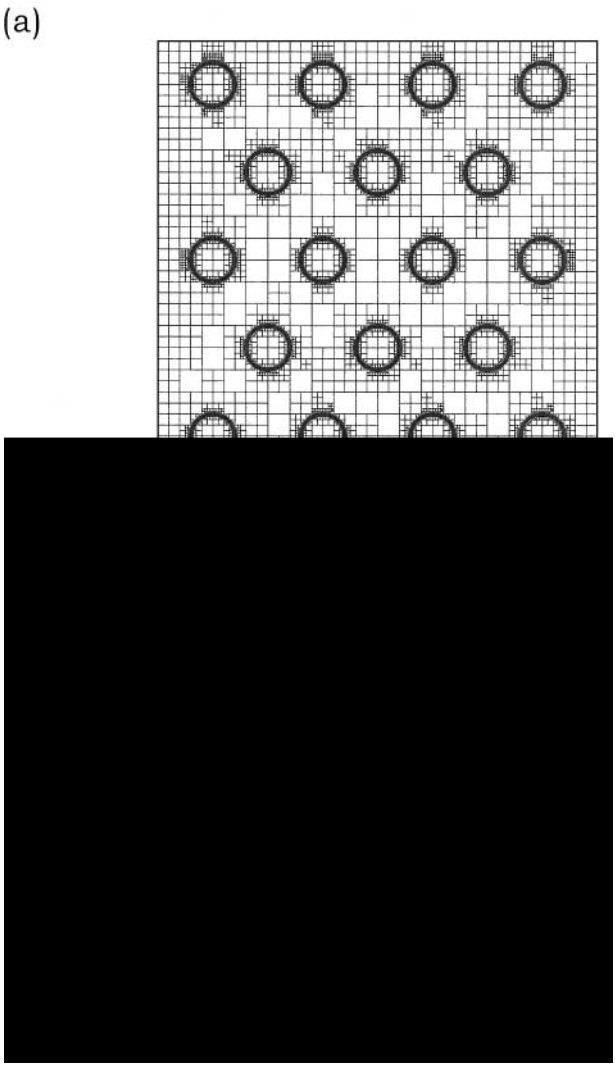

FIG. 10. (a) Final adaptive mesh (27,472 cells with 7 levels of grids; $\delta = 5 \times 10^{-4}$) for 18 spheres in a heated square; (b) the calculated velocity and thermal fields. Spacing of isotherm contours is 0.1.

are coupled with the heat transfer and melt flow. This problem has been solved extensively as a benchmark problem for various Rayleigh numbers ($\mathrm{Pr} = 0.71$) by Newton's method [7] and multigrid schemes [34]. Therefore, it is a good candidate for comparison. The first refined mesh (1922 cells with four levels and $\delta = 2 \times 10^{-3}$) based on the interface refinement is shown on the left-hand side of Fig. 11b, while the body-fitted coordinate is shown on the right (a coarse grid); $\mathrm{Ra} = 2 \times 10^4$. The structured grid for comparison is finer being 320×160 (in the fifth level). The comparison of the calculated isotherms is shown on the left figure of Fig. 11c, where the refinement is based on both the interface and velocity errors; the flow pattern is illustrated by the dashed lines. As shown, the agreement is excellent for both isotherms and the interface shape. Furthermore, the calculated Nusselt number ($\mathrm{Nu} = 1.980$) is in good agreement with the one by the structured grid (1.977); the error distribution is similar to Fig. 6b. Again, the error contours for velocity are shown

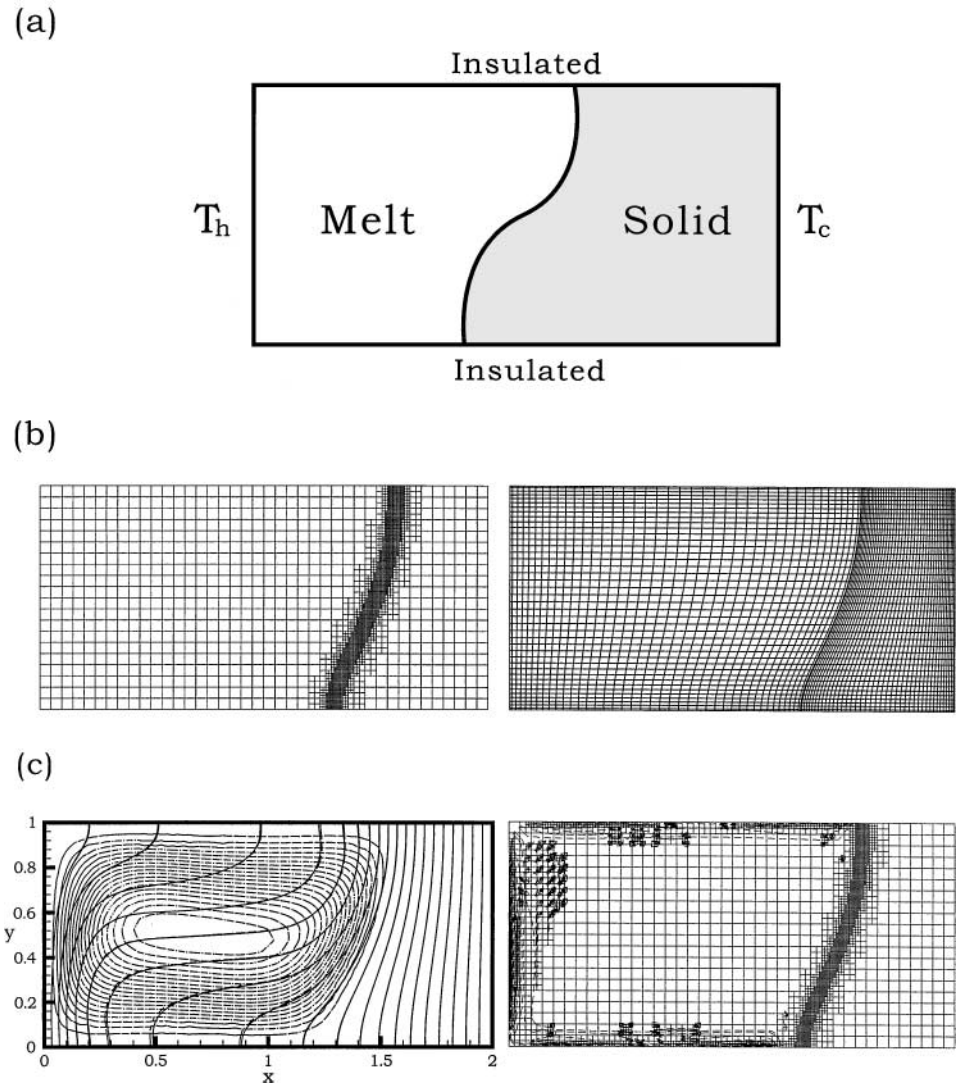

FIG. 11. (a) Schematic of the horizontal Bridgman configuration; (b) initial adaptive mesh (left) having 1922 cells with 4 levels and $\delta = 2 \times 10^{-3}$ and body-fitted mesh (right); (c) comparison of isotherms (flow is indicated by the dash-dotted lines) and the final mesh and errors (right); spacing for isotherms = 0.05.

on the right figure by the dashed lines; all the errors in each stages of refinement are added. Again, at the beginning, the largest errors (-1.56 and 0.63) are distributed near the boundary. After refinements, the maximum error in the order of 0.5 is distributed uniformly slightly away from the initial refined zone; one can find the small clusters of error contour at $|e| = 0.1$ there. The final mesh (4010 cells) on the right of Fig. 11c indicates that the refinement indeed follows the error estimators. The total number of cells on the final mesh (4010) is one order smaller than that by the structured mesh (320×160). As mentioned previously at the beginning of the calculation, a larger interface thickness (6δ) needs to be used. During mesh refinement, one can further reduce this value to reach the sharp interface limit. However, in some cases, if δ is reduced too fast, the new interface may be out of the

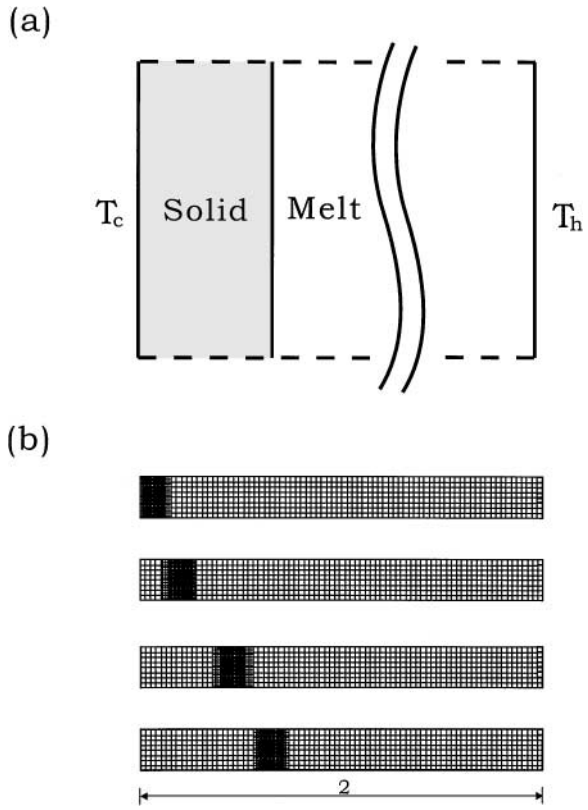

FIG. 12. (a) Schematic of one-dimensional directional solidification; (b) adaptive moving mesh (only half is down) for calculation; the infinite domain is simulated by a finite domain ($L = 4$); $\delta = 1 \times 10^{-2}$.

refined region. Then, the calculation can be greatly retarded and sometimes diverged. On the other hand, without mesh refinement, gradually reducing the interface thickness during iterations is useless.

In the previous case, there is no heat of fusion released due to the steady-state assumption. To further examine the effect of the heat of fusion, we also perform calculation for a one-dimensional solidification problem as shown in Fig. 12a, where its analytical solution on the interface position $d(t)$, the solid (T_s), and liquid (T_l) temperatures can be derived [38, 39] ($T_C = 0$, $T_H = 1$, and $T_m = 0.5$) as

$$\begin{aligned} d(t) &= 2\gamma\sqrt{t}; \\ T_s &= \frac{T_m}{erf(\gamma)} erf(x/\sqrt{4t}); \\ T_l &= 1 + \frac{T_m - 1}{erf(\gamma)} erf(x/\sqrt{4t}), \end{aligned} \tag{26}$$

where γ is a constant and can be calculated implicitly by

$$T_m = erf(\gamma)\left(1 + \frac{\sqrt{\pi}}{St}\gamma \exp(\gamma^2) erfc(\gamma)\right). \tag{27}$$

A sample moving mesh is shown in Fig. 12b (1826 cells). In this case, the solid phase does not exist at $t = 0^-$, but this does not cause any trouble in calculation. In order to take a bigger time step for integration, the refined zone cannot be too small. Otherwise, the interface can easily move out of the refined zone leading to the failure of refinement. If the interface

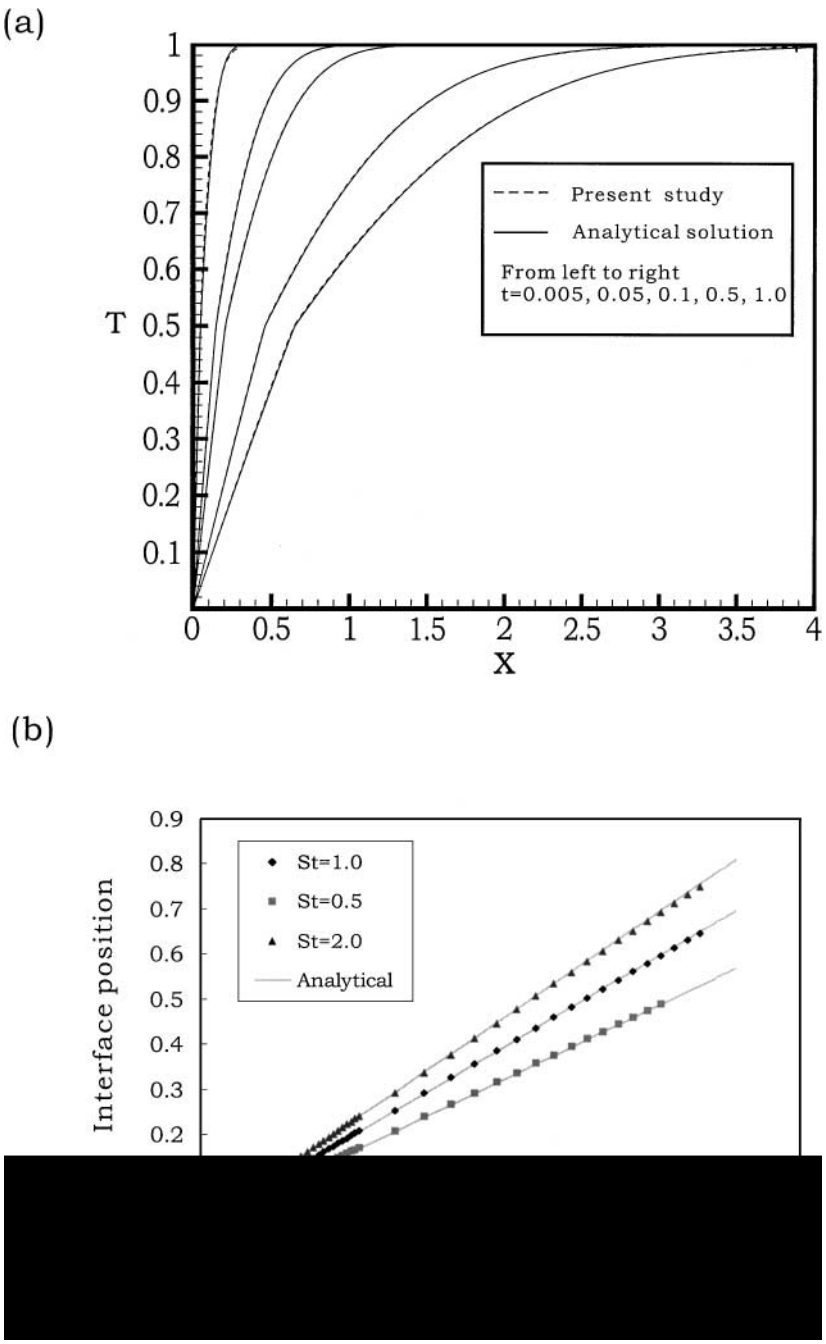

FIG. 13. (a) Comparison of calculated isotherms (dashed lines) at different times with the analytical results (solid lines); $St = 1$ and $\delta = 1 \times 10^{-2}$; (b) comparison of calculated interface position (symbols) as a function of square root of time at different Stefan numbers with the analytical results (solid lines); $\delta = 1 \times 10^{-2}$.

speed is low, a much thinner refined zone can be used. As will be illustrated shortly, this is the case for the dendritic growth using the phase-field model. The calculated isotherms at different times are shown in Fig. 13a. As shown, the agreement with the analytical solution is excellent. The smaller thermal gradient in the melt side of the interface at longer times is due to the release of the heat of fusion. The positions of the interface at different Stefan numbers (St) are also plotted against with the square root of time, and they happen to be linear (as shown by the first formula of Eq. (26)). Again, the agreement is also quite satisfactory.

Extension to the solidification in a crucible ($\mathrm{Pr} = 0.71$ and $\mathrm{Ra} = 10^5$), as shown in Fig. 14a, is also straightforward. Initially, the melt is kept at $T_H = 1$. Due to zero thermal gradients, the melt has no convection. When $t = 0^+$, we set the walls to be cold at $T_C = 0$, except the upper one, which is still assumed to be adiabatic. In addition to the solidification at the walls (with a new phase formation), melt convection also starts. The calculated result showing isotherms and flow pattern (dashed-line) at $t = 0.05$ is shown in Fig. 14b (on the left), and the mesh is shown on the right (about 8000 cells). The larger isotherm spacing in the melt is also due to the heat of fusion as well as the melt convection ($St = 1$ and $\delta = 5 \times 10^{-3}$). Again, in this case, we have purposely using a bigger refined zone in order to use a larger time step for integration. The refinement on the velocity error is also performed, as illustrated in Fig. 14c. Again, more cells are allocated near the interface, where the momentum boundary develops (about 500 cells are added). We have put the isotherms with and without this refinement on the left figure of Fig. 14c for comparison, but the difference is not significant. Because the upper boundary is adiabatic, the isotherms at the top are perpendicular to the top wall. At the end of solidification, the melt convection also becomes weaker due to the smaller space for convection. The thermal gradients become much smaller as well. The calculation can continue until the temperature of the whole domain approaches T_C. The disappearance of the melt and the interface do not cause any problems during calculation.

4.3. Phase-Field Simulation of Dendritic Growth

We have presented some examples for solidification using the enthalpy model, where the phase-field variable is estimated from the temperature as well as the hyperbolic tangent function (Eq. (5)). This approach is suitable to macroscopic solidification, where the capillary contribution due to the curvature of the interface can be ignored. However, for microscopic solidification, such as the dendritic growth, interface curvature and thus the interfacial energy play an important role. In such a situation, the local interface temperature is no longer the equilibrium melting point from the phase diagram. One has to incorporate the Gibbs–Thompson equation [15, 40] into account. Therefore, other approaches, such as the phase-field simulation or the level set method can also be used to obtain the phase-field variable. To this kind of simulation, the present numerical method can be adopted easily as well. In fact, we have found that the solution becomes much easier due to the more diffusive interface obtained by the phase-field equation. Therefore, the final examples are devoted to phase-field simulation of dendritic growth. The model developed by Karma and Rappel [40] is also adopted, which is thermodynamically consistent. For comparison purposes, we have purposely ignored the melt convection first. The effects of forced convection on the growth will be illustrated shortly. The governing equations for dimensionless temperature ($\theta \equiv C_s(T - T_\infty)/\Delta H$) and the phase-field variable [40] are written

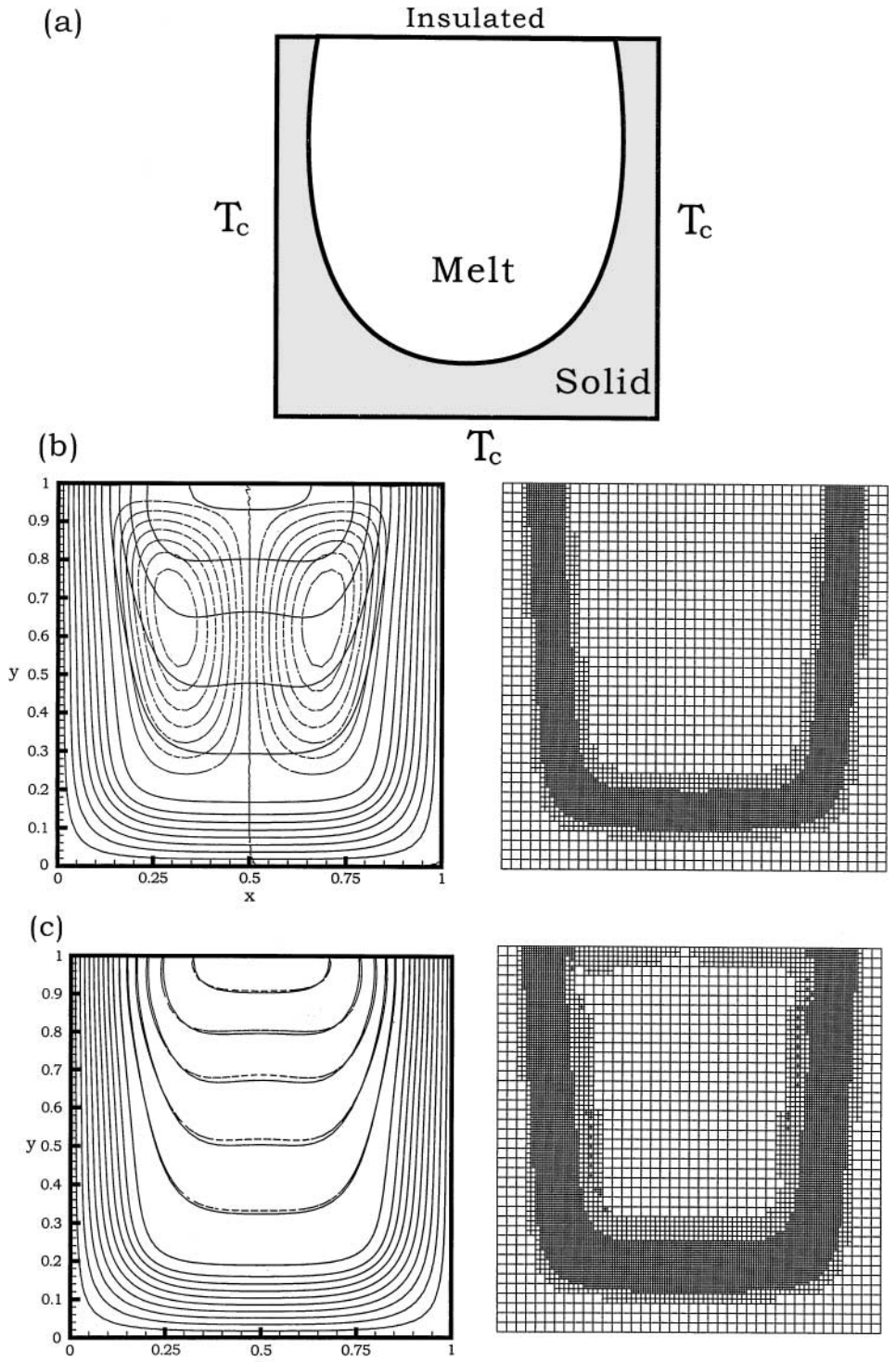

FIG. 14. (a) Schematic of solidification in a square crucible; (b) calculated isotherms and flow patterns (left) at $t = 0.05$ and the mesh (right); (c) comparison of the isotherms (left) after further mesh refinement (right) based on velocity errors; the isotherms based on the velocity refinement are represented by the dashed lines; $St = 1$ and $\delta = 5 \times 10^{-3}$.

as, respectively,

$$\frac{\partial \theta}{\partial t} = D\nabla^2\theta + \frac{1}{2}\frac{\partial \phi}{\partial t} \tag{28}$$

$$\tau_0 w^2 \frac{\partial \phi}{\partial t} = \nabla \cdot (w^2 \nabla \phi) + [\phi - \lambda\theta(1-\phi^2)](1-\phi^2) - \frac{\partial}{\partial x}\left(w\frac{\partial w}{\partial \beta}\frac{\partial \phi}{\partial y}\right) + \frac{\partial}{\partial y}\left(w\frac{\partial w}{\partial \beta}\frac{\partial \phi}{\partial x}\right), \tag{29}$$

where D is the dimensionless thermal diffusivity, $w = 1 + \varepsilon \cos 4(\beta - \beta_0)$, $\beta = \tan^{-1}[(\partial\varphi/\partial y)/(\partial\varphi/\partial x)]$, and τ_0 is a time constant; ε is a parameter for the interface energy anisotropy and β_0 the seed orientation. Since the four-fold symmetry is considered, if $\beta_0 = 0^\circ$, one can take advantage of symmetry, and only a quarter of the domain is needed for diffusive growth. However, when seed orientation is arbitrary, a full domain is required. To examine effect grid anisotropy, we have used a full domain for computation. In addition, for comparison purposes, the phase-field variable ϕ now ranges from 1 for solid and -1 for liquid, and $\phi = 0$ is at the interface. Due to the slow development of the dendrite, typically the time step is usually small ($\sim 0.01\ \tau_0$). Therefore, the refined zone can be very thin, which saves a tremendous amount of cells, especially for the simulation of low supercooling, where an extremely large domain is necessary. A sample of calculation is shown in Fig. 15a for the mesh (the first quarter), the isotherms (the second quarter), and the phase-field variable (the third quarter) for a fully developed dendrite (the fourth quarter) at the dimensionless supercooling (Δ) being 0.55; $\Delta \equiv C_l(T_m - T_\infty)/\Delta H$, where T_∞ is the far-field temperature. We have purposely magnified the mesh at the interface locally to show the mesh refinement. In this case, eight levels of grids are used and the cell number is 129036 for a full domain in Fig. 15a (only the mesh of a quarter domain is shown). In other words, the ratio of the largest to the smallest cell sizes is 2^8. Therefore, once the refinement can be carried out effectively, the mesh can cover different length scales for simulation. If an explicit integration scheme is used, the largest time step for integration is limited by the smallest cell size (the Courant–Friedrichs–Lewy (CFL) condition). Although a fully implicit scheme is used for the integration, due to the thin refined zone, to avoid numerical instability, the advancement of the new interface needs to be inside the refined zone. This is particularly true when side branching becomes important [41].

The comparison of our calculated dendrite tip speed for the supercooling of 0.55 with the solvability theory [40] is shown in Fig. 15b. Again, as shown the solvability limit can be reached as well and the agreement is very good. To examine the effect of growth orientation, we have used a full domain in this case. This also allows us to further examine the effect of grid anisotropy. Different seed orientations are considered. For $\beta_0 = 0^\circ$, the tip selection is in the x- or y-direction, and the agreement with the theory is very good. In this case, the calculation of the tip speed can be more accurate due to the search of zero contour of the phase-field variable at each time step. Although the calculated dendrite shapes with different orientations agree quite well with that of $\beta_0 = 0^\circ$, the calculation of the tip speed is more tedious and less accurate. Nevertheless, the error of the tip speed is still within 3%; the dendrite tip grows slightly faster for $\beta_0 = 30^\circ$ and 45°. Indeed, the grid anisotropy slightly affects the tip speed and its effect can be further reduced if the grid is further refined. Furthermore, the calculation is tested for several supercoolings, and the agreement with reported values is very good. Furthermore, the performance is also comparable to that

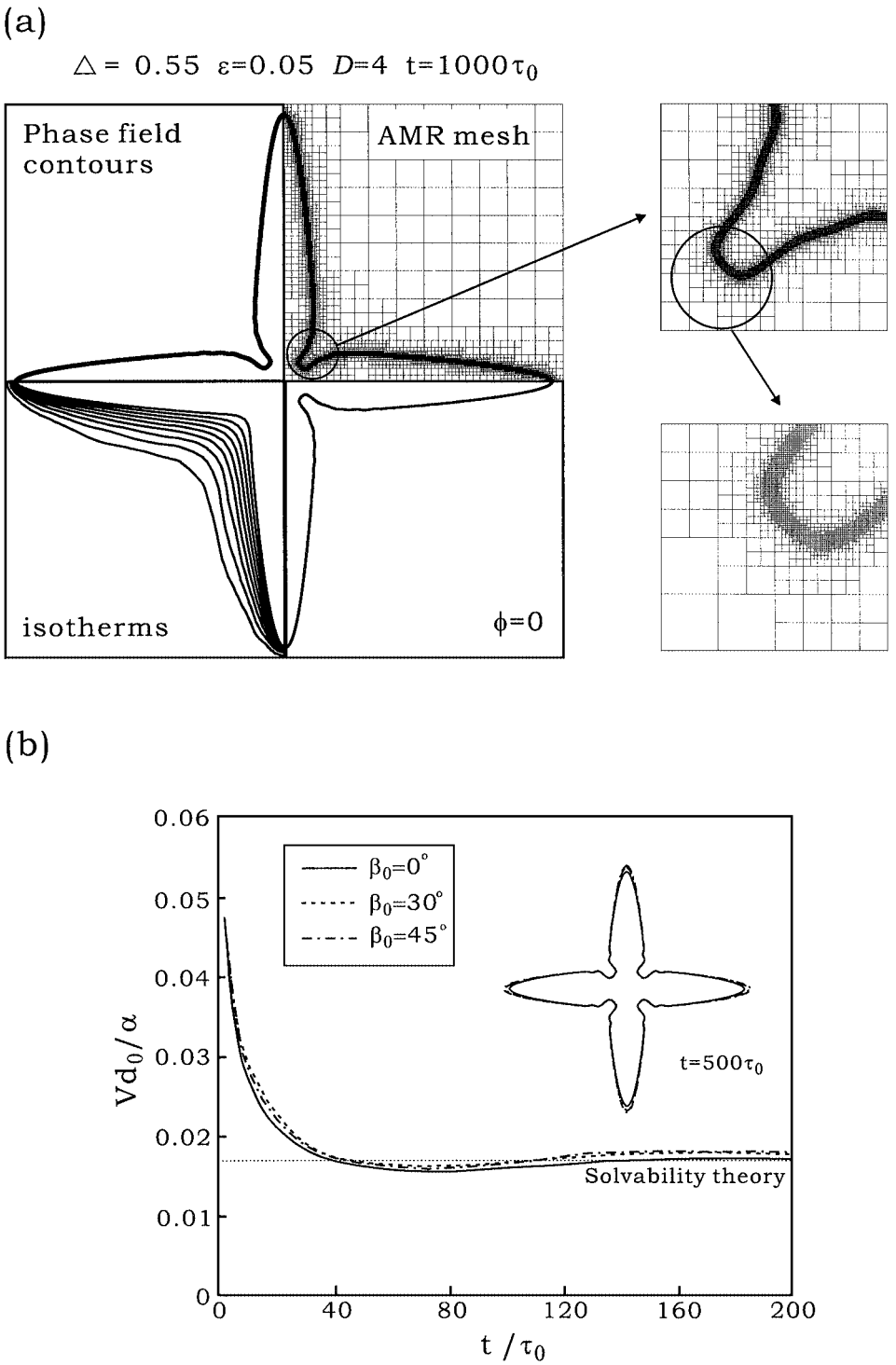

FIG. 15. (a) Calculated dendrite shape (the fourth quarter), mesh (the first quarter), isotherms (the second quarter), and the phase-fields (the third quarter); a local lookup of the mesh is further illustrated by arrows indicating the multilevel and multiscale nature of the refinement; the spacing = 0.055 for isotherms and 0.2 for phase-field variable; (b) evolution of tip speeds (for three different growth orientations); the solvability theory is shown by the dashed horizontal line. The dendrite shapes of different growth orientations are overlapped for comparison.

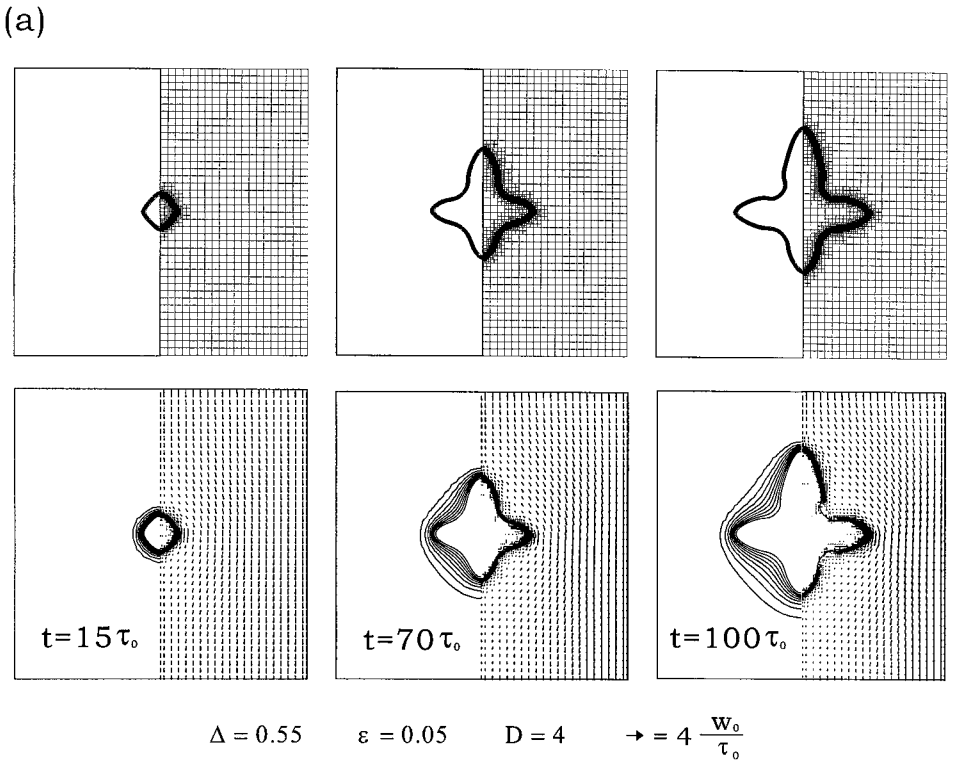

FIG. 16. (a) Calculated dendrite growth for phase-field variable (upper left), mesh (upper right), isotherms (lower lest), and flow patterns (lower right) at different times; the contour spacing $=0.055$ for isotherms; (b) comparison of tip speed evolution at three locations with the results by Beckermann *et al*. [15].

described by Provatas *et al*. [25] using an adaptive finite element method that the CPU time is proportional to the domain size. However, as compared with the their refinement scheme using triangular cells, the present approach seems to be much easier. Detailed comparison with the previous study and theory can be found elsewhere [41]. Furthermore, the present

approximation is fully conserved due to the nature of the FVM approximation, so that the extension to the solutal calculation, which requires a highly conserved scheme, may be easier. Furthermore, the converge speed of the calculation at each time step is also faster than that by enthalpy model with a very small δ. The interface thickness can be judged from the contours (10 contours) of the phase-field variable in Fig. 15a. In addition, according to Karma and Rappel [40], if the parameters are carefully chosen, such a thickness approaches the sharp-interface limit.

The final example is carried out for the dendritic growth under a forced convection for $\Delta = 0.55$. D = 4, and $\varepsilon = 0.05$. This example is the same as the one used by Beckermann *et al*. [15]; they used a uniform mesh for calculation, and they showed by simulation for the first time the effect of flow on the dendrite tip speed. The calculated dendrite shapes (phase fields) and meshes at different times are shown in the top of Fig. 16a; the corresponding isotherms and flow fields are shown in the lower figures. As shown, the tip at the upstream grows faster due to the thinner thermal boundary layer. The evolution of the tip velocities is further illustrated in Fig. 16b; the results of Beckermann *et al*. [15] are also put together for comparison. As shown, they are in good agreement. However, the cell numbers used here (2300, 5678, 7430 at $t = 15$, 70, $100\tau_0$, respectively) are one order of magnitude smaller than that used by Beckermann *et al*. (512×256), while our smallest cell size is also smaller than theirs. Due to the fully implicit scheme used here for integration, a much bigger time step being about $0.2\tau_0$ can be used ($0.01\tau_0$ for an explicit scheme on the phase-field equation), and this reduces overall CPU time dramatically. To save space, further discussion on the dendritic growth, especially at low supercooling ($\Delta = 0.25$ and 0.1) under a convective environment will be discussed elsewhere [41]. Furthermore, the grid at the root level has some effects on the result. Too coarse a grid will affect the overall growth environment.

5. CONCLUSIONS

An adaptive finite volume method using dynamic data structures, implemented in FORTRAN90, is developed for solidification problems. The method uses a fixed-grid and two-phase equations to treat free or moving interfaces as well as the associated heat transfer and fluid flow through a phase-field variable, which is defined by temperature fields and interface thickness. In addition to the refinement on the interfaces or boundaries, error estimators are also used for the refinement and the coarsening.

For the simulation of two-phase problems having a sharp interface, the accuracy of the fixed-grid approach using a structure-grid is usually limited by the cell size. However, through the adaptive mesh, the problem could be resolved while using much less cells for calculation. The strategy proposed here starts from a coarse grid having a thick interface. As the problem converges at coarse levels, mesh refinement or coarsening is carried out according to geometric constraints or error estimators, followed by the interface thinning. Through the extensive benchmarkings with the front tracking approach, we have found that the shape-interface problem can be well modeled. In addition, this approach is particularly useful for a domain having complex immersed interfaces (or with interface merging or splitting) that can be difficult to tackle by front tracking.

The present approach can be further extended to phase-field simulation. We have illustrated successfully the simulation of dendritic growth using the phase-field equation. For the

case without convection, the calculated dendrite tip speeds at various undercoolings agree very well with the solvability theory. With a forced convection, the calculated result also agrees well with previous calculation, but uses much less cells. Therefore, we believe that further extensions to other multiphase systems, such as bubbling and sintering, may be feasible. One may also use level sets or other definitions of the phase-field variable to describe the moving interfaces. Due to the simplicity of the refinement scheme and the related data structures, the extension to three-dimensional problems is straightforward. However, one needs to keep this in mind that the computational cost for 2D problems is proportional to the arclength of the interface (for a thin refined zone). However, for 3D problems, the cost is proportional to the overall interfacial area. Therefore, a dramatic increase of CPU time is expected. Furthermore, although the approach is also suitable for a highly diffusive interface, as the interface thickness increases the computational cost increases as well. A careful decision needs to be made for the refinement inside the interface region for a cost-effective simulation.

Finally, due to the nature of the multilevel in the implementation, multigrid methods [34, 36] fi

11. V. R. Voller and C. R. Swaminathan, General source-based method for solidification phase change, *Numer. Heat Trans. B* **19**, 175 (1991).
12. M. Yao and Henry de Groh III, Three-dimensional finite element method simulation of Bridgman crystal growth and comparison with experiments, *Numer. Heat Trans. A* **24**, 393 (1993).
13. W. Shyy and M. Chen, Steady-state natural convection with phase change, *Int. J. Heat Mass Trans.* **33**, 11, 2545 (1990).
14. M. Fabbri and V. R. Voller, Numerical solution of plane-front solidification with kinetic undercooling, *Numer. Heat Trans. B* **27**, 467 (1995).
15. C. Beckermann, H.-J. Diepers, I. Steinbach, A. Karma, and X. Tong, Modeling melt convection in phase-field simulations of solidification, *J. Comput. Phys.* **154**, 468 (1999).
16. J. A. Sethian and J. Strain, Crystal growth and dendritic solidification, *J. Comput. Phys.* **98**, 231 (1992).
17. S. Chen, B. Merriman, S. Osher, and P. Smereka, A simple level set method for solving Stefan problems, *J. Comput. Phys.* **135**, 8 (1997).
18. M. Sussman, A. S. Almgren, J. B. Bell, P. Colella, L. H. Howell, and M. L. Welcome, An adaptive level set approach for incompressible two-phase flows, *J. Comput. Phys.* **148**, 81 (1999).
19. D. A. Drew, Mathematical modeling of two-phase flow, *Ann. Rev. Fluid Mech.* **15**, 261 (1983).
20. H. S. Udaykumar, H. C. Kan, W. Shyy, and R. T.-S. Tay, Multiphase dynamics in arbitrary geometries on fixed Cartesian grids, *J. Comput. Phys.* **137**, 366 (1997).
21. H. S. Udaykumar, R. Mittal, and W. Shyy, Computation of solid–liquid phase fronts in the sharp interface limit on fixed grids, *J. Comput. Phys.* **153**, 535 (1999).
22. T. Ye, R. Mittal, H. S. Udaykumar, and W. Shyy, An accurate Cartesian grid method for viscous incompressible flows with complex immersed boundaries, *J. Comptut. Phys.* **156**, 209 (1999).
23. C. S. Peskin, Numerical analysis of blood flow in the heart, *J. Comput. Phys.* **25**, 220 (1977).
24. D. Juric and G. Tryggvason, A front-tracking method for dendritic solidification, *J. Comput. Phys.* **123**, 127 (1996).
25. N. Provatas, N. Goldenfeld, and J. Dantzig, Adaptive mesh refinement computation of solidification microstructures using dynamic data structures, *J. Comput. Phys.* **148**, 265 (1999).
26. R. Tonhardt and G. Amberg, Dendritic growth of randomly oriented nuclei in a shear flow, *J. Crystal Growth* **213**, 161 (2000).
27. I. Demirdzic and S. Muzaferija, Numerical method for coupled fluid flow, heat transfer and stress analysis using unstructured moving meshes with cells of arbitrary topology. *Comput. Meth. Appl. Mech. Eng.* **125**, 235 (1995).
28. S. Muzaferija and D. Gosman, Finite-volume CFD procedure and adaptive error control strategy for grids of arbitrary topology, *J. Comput. Phys.* **138**, 766 (1997).
29. S. R. Mathur and J. Y. Murthy, A pressure-based method for unstructured meshes, *Numer. Heat Trans. B* **31**, 195 (1996).
30. L. H. Howell and J. B. Bell, An adaptive mesh projection method for viscous incompressible flow, *SIAM J. Sci. Comput.* **18**, 996 (1997).
31. M. J. Berger and J. Oliger, Adaptive mesh refinement for hyperbolic partial differential equations, *J. Comput. Phys.* **53**, 484 (1984).
32. C. M. Rhie and W. L. Chow, Numerical study of the turbulent flow past an airfoil with trailing edge separation, *AIAA J.* **21**, 1527 (1983).
33. S. V. Patankar, *Numerical Heat Transfer and Fluid Flow* (Hemisphere, Washington, DC, 1980).
34. C. W. Lan and M. C. Liang, Multigrid method for incompressible heat flow problems with an unknown interface, *J. Comput. Phys.* **152**, 55 (1999).
35. H. L. Stone, Iterative solution of implicit approximations of multidimensional partial differential equations, *SIAM J. Numer. Anal.* **5**, 530 (1968).
36. Z. J. Wang, A fast nested multi-grid viscous flow solver for adaptive Cartesian/Quad grids, *AIAA-96-2091*, June 1996.
37. I. Demirdzic, Z. Lilek, and M. Peric, Fluid flow and heat transfer test problems for non-orthogonal grids: benchmark solutions, *Int. J. Numer. Meth. Fluids* **15**, 329 (1992).

38. V. R. Voller, Similarity solution for the solidification of a multicomponent alloy, *Int. J. Heat Mass Trans.* **40**, 2869 (1997).
39. N. Palle and J. A. Dantzig, Adaptive mesh refinement scheme for solidification problems, *Metall. Mater. Trans. A* **27A**, 707 (1996).
40. A. Karma and W. J. Rappel, Quantitative phase-field modeling of dendritic growth in two and three dimensions, *Phys. Rev.* **E57**, 4323 (1998).
41. C. W. Lan, C. M. Hsu, and C. C. Liu, Efficient adaptive phase filed simulation of dentritic growth in a forced flow at low supercooling, *J. Crystal Growth*, in press.