BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES
Vol. 55, No. 2, 2007

Parallel adaptive computation of some time-dependent materials-related microstructural problems

M. DO-QUANG*, W. VILLANUEVA, I. SINGER-LOGINOVA, and G. AMBERG

Linné Flow Centre, Department of Mechanics, Royal Institute of Technology, SE-100 44 Stockholm, Sweden

Abstract. Some materials-related microstructural problems calculated using the phase-field method are presented. It is well known that the phase field method requires mesh resolution of a diffuse interface. This makes the use of mesh adaptivity essential especially for fast evolving interfaces and other transient problems. Complex problems in 3D are also computationally challenging so that parallel computations are considered necessary. In this paper, a parallel adaptive finite element scheme is proposed. The scheme keeps the level of node and edge for 2D and level of node and face for 3D instead of the complete history of refinements to facilitate derefinement. The information is local and exchange of information is minimized and also less memory is used. The parallel adaptive algorithms that run on distributed memory machines are implemented in the numerical simulation of dendritic growth and capillary-driven flows.

Key words: parallel computing, adaptive finite element method, dendritic growth, wetting.

1. Introduction

Materials processing is one of the oldest of the applied sciences. From thousand years ago, people already knew about casting and metallurgy. But up to now, many problems in materials processing are still open from a physical point of view. For example, the effects of dendritic growth to the macroscopic properties such as the alteration of microstructure by the presence of melt flow during solidification, or the capillary-driven effects in the sintering of powders, etc. These problems involve multiphase flows with or without phase change. The mathematical descriptions of these multiphase-flow problems can be categorized into two types, sharp-interface models and diffuse-interface models. Diffuse-interface models can be traced back to Maxwell, Gibbs, Van der Waals [1] and Korteweg [2], and in a more modern setting by Cahn & Hilliard [3].

During the last two decades, significant progress has been made in the computation of dendritic solidification without convection. Simulations have been performed using techniques such as the phase-field method [4–7], level set method [8,9] and explicit interface tracking methods [10,11]. The extension of these methods to include the effect of melt flow during the solidification are relatively recent. Tönhardt and Amberg [12] and Beckermann et al. [13] consider the solid phase as rigid and stationary. They used the phase-field method to simulate the two-dimensional dendritic growth into an undercooled liquid and set the flow velocity in the solid phase to zero. Beckermann et al. introduced a mixture formulation and an auxiliary interfacial stress term into the momentum equation to ensure the correction of the shear stress at the interface and hold the solid in place. Al-Rawahi and Tryggvason [14] used the front tracking method to simulate the dendritic growth into an undercooled liquid. They used a fixed mesh for the temperature equation, in which the temperature boundary condition on the interface is applied explicitly and the heat source is found directly from the temperature gradient near the interface. They used another mesh for velocity and pressure, which exist in the fluid phase only. This leads to remeshing problems. The level set method, Osher et al. [15], is an alternative method to handle interface tracking. It was first used by Zhang et al. [16] to simulate the solidification of molten droplets on a cold substrate. Later, Kim et al. [9] and Gibou et al. [17] applied the level set method to simulate the dendritic growth. They used the level set method to keep track of the front and solved for the diffusion field using an implicit time discretization method.

Capillary-driven flows involve two immiscible liquids or a liquid and air separated by a deformable interface in which the dynamics of the interface is largely influenced by capillary forces. A contact of these fluids with a third phase, usually a solid surface, gives rise to a phenomenon called wetting. Apart from being a generic phenomenon in nature and technology, wetting is pertinent to numerous materials processes such as liquid phase sintering in powder metallurgy.

Liquid phase sintering is a technological process that combines a particulate solid and a softer powder that acts as a binder, melts at a lower temperature and enhances material movement during the sintering process. One of the oldest and most successful liquid phase systems is a cemented tungsten carbide with cobalt additive used for cutting and machining tools. Tungsten carbide is known to be hard and brittle while cobalt is relatively soft and

*e-mail: minh@mech.kth.se

ductile. The two metal powders which have typical fine grade sizes of about 1–10 microns in diameter are mixed and pressed, then heated until the cobalt binder melts. The liquid cobalt wets the solid grains and due to capillary forces, the microstructure undergoes rearrangement. Simultaneously, mass diffusion is present as well as grain growth that all contribute to pore elimination leading to densification. Some important factors influence the densification of the compact microstructure such as the amount of liquid present, particle size, solubility of the solid in liquid, contact angle, dihedral angle, etc. [18]. A long standing issue in powder metallurgy is to control and predict the shape of the finished, sintered part, from the pressed powder shape.

The wetting of a liquid on a solid can be classified into two types: total wetting, when the liquid spreads completely; and partial wetting, when the liquid at equilibrium rests on the solid with a contact angle θ_e [19]. Both are characterized by the spreading parameter $S = \sigma_{SM} - (\sigma_{SL} + \sigma_{LM})$, where the σ's are surface tensions at the solid/medium (medium is either air or another liquid), solid/liquid, liquid/medium interfaces, respectively. $S > 0$ corresponds to total wetting and $S < 0$ corresponds to partial wetting.

A difference between the phase-field method and the other methods for the simulation of dendritic solidification and capillary-driven flows is that the important physical mechanisms, such as curvature, anisotropy and kinetics effect, are implicitly incorporated in the phase-field equations. Also, by solving a diffuse interface on a fixed, or adaptively refined mesh, it avoids the need for applying temperature boundary conditions on the moving interface. It turns out that when we compute the heat fluxes from the temperature nodal values, it shall not have any problem with the discretization error that may otherwise affect the energy solution [20]. A very attractive feature of diffuse interface methods is that similar methodology can be used to simulate all the different free surface problems in materials problems, from phase change to capillarity.

The limitation of the phase-field method is the requirement of mesh resolution of the diffuse interface. For fast evolving interfaces, the use of mesh adaptivity maybe considered essential. Also, in many transient problems the regions of interest occur only in certain parts of the domain. With a given accuracy, the computational cost can be greatly reduced if we adapt the mesh, and available computational resources are more concentrated on regions where the solution changes rapidly. In many cases both refinement and derefinement are needed. While refinement tries to keep the solution accurate enough, dynamic derefinement makes the computation as efficient as possible by making sure that computational resources are not wasted or unnecessarily used. For the same reason of reducing the computational and storage requirements, the adaptive computation can be done in a parallel computing environment. In fact, some transient problems are way too complex to solve without the use of parallel and/or adaptive computation. However, the design and implementation of an efficient and reliable parallel adaptive algorithm remains difficult because there are many issues that must be resolved especially in the parallel implementation.

2. Parallel-adaptive implementation

A combination of parallel and adaptive computation introduces issues that must be resolved. For a parallel solver to be efficient, the total workload must be evenly distributed to each processor which is done by partitioning the mesh in such a way that each processor takes the same number of elements and communication between processors should be kept minimum. The communication between processors and exchange of data can also be made efficient by having a sound data management [21] as well as keeping less information to be shared with other processors. Since computational power of individual processors can be increased with increasing demand, focusing more on ways of improving communication between processors offers a great deal of efficiency. Without adaptivity, the mesh does not change and exchange of information is limited only on the data on the partition boundary. Mesh repartitioning is also unnecessary as well as mesh migration. The inclusion of adaptivity, however, which is done in parallel often requires the mesh to be repartitioned to keep a balanced workload. Mesh migration is unavoidable. Moreover, the standard use of adaptive refinement/derefinement also requires keeping the history of refinements, to facilitate derefinement and maintain the nestedness of the mesh [22]. But this greatly increases the communication cost because it is an added information that has to be shared between processors and also requires some memory to keep the information. In this paper, however, we propose a scheme that does not keep the history of refinement but keeps a local information about the node and edge level. This information can be used to track back to the previous level of mesh refinement, i.e., it is used to identify which nodes, edges or faces to be removed and generate another set of information to be used for the next level of derefinement.

There are several papers on parallel algorithms for the finite element method. Most of them tackles one or more of the following issues: mesh partitioning and repartitioning [23–28], load balancing and mesh migration [24,25,27–29], and data structures [21,27,30] . Several authors have also addressed both parallel and adaptive computation [27,31–34]. However, derefinement has not been considered in [31,33] which is necessary in some problems like the examples that will be shown in this paper. Jeong, et al. [32] studied fluid flow on 3D dendritic growth on an adaptive finite element grids implemented in parallel. In their work, hexahedral elements were used. Waltz [34] presented a parallel adaptive refinement algorithm for 3D tetrahedral unstructured grids. The algorithm has been parallelized for shared-memory platforms and overcame the indirect access memory problem by using domain decomposition. Moreover, the scheme was applied to unsteady flow and the mesh adaption was done serially. Castanos [27] have studied parallel adaptive unstructured computation. Rivara's longest-edge bisection algorithm is used for 2D and 3D mesh refinements and a refinement tree is utilized for derefinement.

A tool to solve partial differential equations with adaptive finite element method, called femLego, has been developed by Amberg et al. [35]. The partial differential equations, boundary conditions, initial conditions, and methods of solving each equation are all specified in a Maple worksheet. In this paper, we present an extended version of femLego that runs on distributed memory. To illustrate the applicability of the extended version of femLego, we show some examples from problems in heat and mass transfer, materials science and free boundaries. A flowchart of femLego is shown in Fig. 1. The mesh is partitioned using the ParMetis library [36]. A Fortran core code takes care of the matrix assembling which is done in parallel. A matrix solution is obtained using the Aztec library [37]. If adaptivity is switched on, the last computed results are used by an error criterion to indicate regions of high variation of variables, i.e., regions requiring finer mesh. A new mesh, adapted to the solution, will be generated for use at the next step. The new mesh is again partitioned by ParMetis and balanced using a smoothing function. And the process repeats until final time. To simplify implementation and coding for refinement/derefinement, STL (Standard Template Library) C++ [38] is used. Furthermore, MPI (Message Passing Interface) [39] is used in all interprocessor communication.

As mentioned earlier, the adaptivity is done in parallel and all elements have corresponding owners. Each processor contains a refinement/derefinement list. And at each mesh refinement step, individual elements are marked for refinement which will be included in the refinement list, or no change, based on an error indicator calculated from a given error criterion and the element size $h > h_{min}$, the minimum h allowed. At the next refinement step, elements containing hanging nodes are marked for refinement. Error indicators and element sizes are checked again for the new created elements and then the refinement list is updated. The refinement stops if and only if the refinement list of all processors is empty.

Level of node-edge scheme. Initially, we assign a level number to every node, edge of the original mesh. Newly created nodes and edges from every refinement process are also assigned a level number. The reason for keeping the level of nodes and edges is to facilitate derefinement as was mentioned in the previous section. It is cost effective since exchange of information between processors is minimized and less memory is required in keeping the information. The level of node gives the information which node will be removed first while the level of edge tells which edge will be removed first. It should be noted here that in three dimensions, we use the level of face instead of the level of edge but have the same purpose, that is, the faces to be removed and retained when merging simplices can be decided by their levels. The scheme is presented in two dimensions for simplicity but extension to a tetrahedral mesh in three dimensions is straightforward.

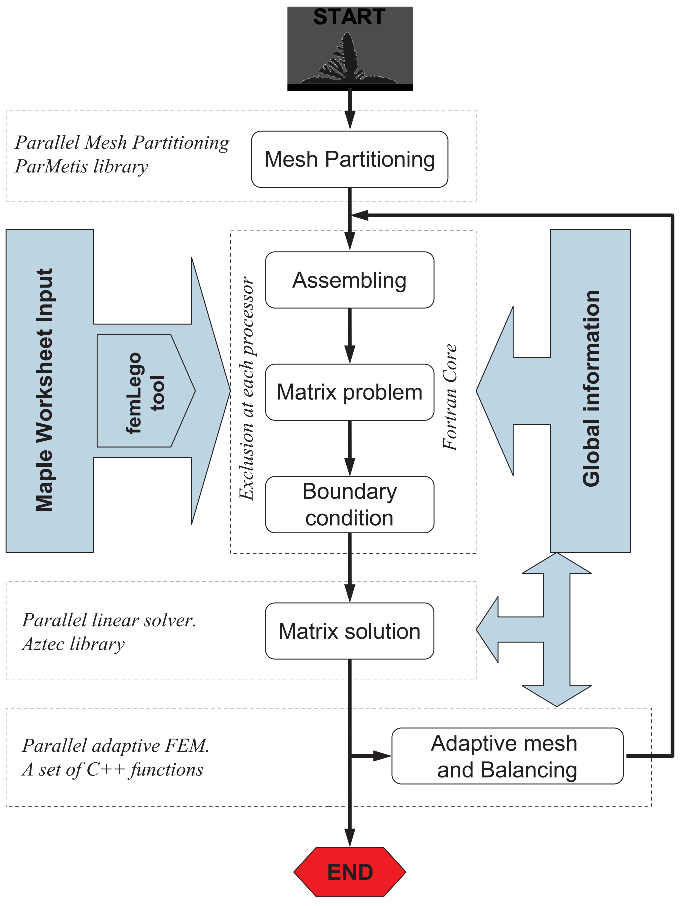

Fig. 1. Flow chart of femLego parallel adaptive version

Refinement. The following is set of rules for assigning the level of nodes and edges.

R-i All original nodes have level 0 and are never marked for removal.
R-ii All original edges have level 0.
R-iii Newly created nodes have level i+1 where i is the highest level of a node in the refinement element.
R-iv All new edges created by bisecting an edge have level j+1 where j is the level of the bisected edge. Otherwise, new created edges have level 0.

To elaborate further, we refer to Fig. 2a. For a simpler case we take a 2D initial mesh containing 4 elements. The figure also shows the node numbers with their levels and the level of the edges. Assuming for example that based on the error indicator, element 2 is marked for refinement. Then we refine element 2 creating node 7 with level 1. The original edge (edge n2-n3) that is bisected has level 0, thus the new edges (edge n2-n7 and n3-n7) has level 1 and the other edge (edge n4-n7) created has level 0. Note that node 7 is a hanging node so we also need to refine element 1 creating a new edge (edge n1-n7) with level 0. Similarly, say a new node 8 is created with level 2 by (R-iii). The edge n3-n4 with level 0 is bisected to create new edges with level 1. Note again that node 8 is a hanging node so we are required to refine element 3 creating node 9 which also in turn refining element 4 and finally refining element n3-n4-n9, see Fig. 2b. In the third level of refinement, let's say element n3-n7-n8 is marked for refinement. A new node is created, node 10 with level 3 because node 8 has level 2 and is the highest in the element. The edge n3-n7 with level 1 is bisected thus giving two new edges (edge n3-n10 and edge n7-n10) with level 2 while edge n8-n10 have level 0 by (R-iv). To satisfy conformity again, element n1-n3-n7 is refined and consequently element n3-n7-n11, see Fig. 2c.

In the parallel implementation our goal is to attain grid closure. As the refinement propagates, it may reach the partition boundary. If an element in the partition boundary owned by processor P1 is refined. The data class in P1 is updated and the new information is passed to the neighboring processors and their data is also updated to assure data consistency. The new information passed to each processor may be used for further refinement if for example the new node that was created in P1 is a hanging node in the receiving processor. The refinement process and exchange of information continues until a conforming grid is obtained. If the new mesh is unbalanced, then we need to repartition and mesh migration follows.

The coarsening is done in the same manner as in the refinement process. An individual element is marked for derefinement based on an error indicator. From the list of elements to be merged, we create a list of nodes to be removed. The coarsening stops if and only if the node list of all processors is empty.

Derefinement. The following are set of rules in the derefinement process.

D-i A node is marked for removal only if all the elements containing the node are marked for derefinement.
D-ii The node with the highest level will be removed first.
D-iii When a node is marked for removal, the edges connected to it with the lowest level will be removed first, then the remaining edges which come in pair will combine to form a new edge with level j-1 where j is their previous level.

Now with the refined mesh of Fig. 2c, we should be able to get back to the previous level of mesh when doing the derefinement. Let us again assume that based on the error indicator, the elements containing node 10 and 11 are marked for derefinement thus node 10 and 11 are marked for removal. Since node 10 has a higher level it will be removed first. When removing node 10, 4 elements are removed and replaced with 2 elements. With the information of the level of edges that will be removed, we can easily reconstruct back to the previous state of the mesh. By (D-iii), the edges n8-n10 and n10-n11 are removed and edges n3-n10 and n7-n10 are combined to form one edge n3-n10 with level 1. The next node to be removed is node 11. Using (D-iii) again, edge n7-n11 will be removed and edges n1-n11 and n3-n11 will combine to form edge n1-n3 with level 0, see Fig. 2b. For the second level of derefinement, elements containing nodes 8 and 9 are marked for merging. Node 8 will be removed first by (D-ii) and by (D-iii), edges n7-n8 and n8-n9 will be removed while edges n4-n8 and n3-n8 will form a new edge n3-n4 with level 0. Next, we remove node 9 and edges n3-n9 and n6-n9 while edges n4-n9 and n5-n9 will form edge n4-n5 with level 0. We assume still that all elements containing node 7 are marked for derefinement. Node 7 will be removed along with edges n1-n7 and n4-n7 and edges n2-n7 and n3-n7 will join to form edge n2-n3 with level 0.

3. Applications

3.1. Modeling solidification microstructure, dendrite growth.

Dendrites are the basic microstructural form of most crystalline materials. It may form from the vapor phase (e.g., snowflakes), from solution (e.g., polymer crystal), or by solidification from a melt (e.g., metals). The conditions under which the dendrite will grow are crucial for the final microstructure of the material that greatly influences its macroscopic properties.

There are several models that can be used to describe solidification problems. A classical way of describing solidification problems mathematically is the Stefan model. In this model the diffusion equations describe the transport of heat between phases, solid and liquid, and the boundary conditions are specified in moving phase interfaces. Finding the analytical solution for the Stefan problem is difficult, since the shape of the phases changes in time. That is why, numerical simulations are widely used since the last decades.

The phase-field, heat and/or diffusion equations are derived in a thermodynamically consistent way by considering the entropy change during solidification. The following equation is a modified heat equation in Stefan problem by using the semisharp method, Amberg [40],

$$\frac{\partial\theta}{\partial t} + (\mathbf{u}\cdot\nabla)\,\theta = \nabla^2\theta + \frac{\partial g_\delta(\phi)}{\partial t} \tag{1}$$

where ϕ is the phase field variable which is $+1$ in the solid and -1 in the liquid, $g_\delta(\phi)$ accounts for the change in internal energy on phase change and should increase from 0 to 1 as ϕ goes from -1 to $+1$. The advantage of this method is that the interface can be identified with the surface $\phi = 0$ precisely, in that it can be shown that the correct kinetics is satisfied there.

The phase field evolution equation is written as follows,

$$\tau\frac{\partial\phi}{\partial t} - \tau\,(\mathbf{u}\cdot\nabla)\,\phi = W^2\hat{\nabla}^2\phi - \frac{\partial f(\phi)}{\partial\phi} - \frac{\partial g_\delta(\phi)}{\partial\phi}h(\lambda\theta) \tag{2}$$

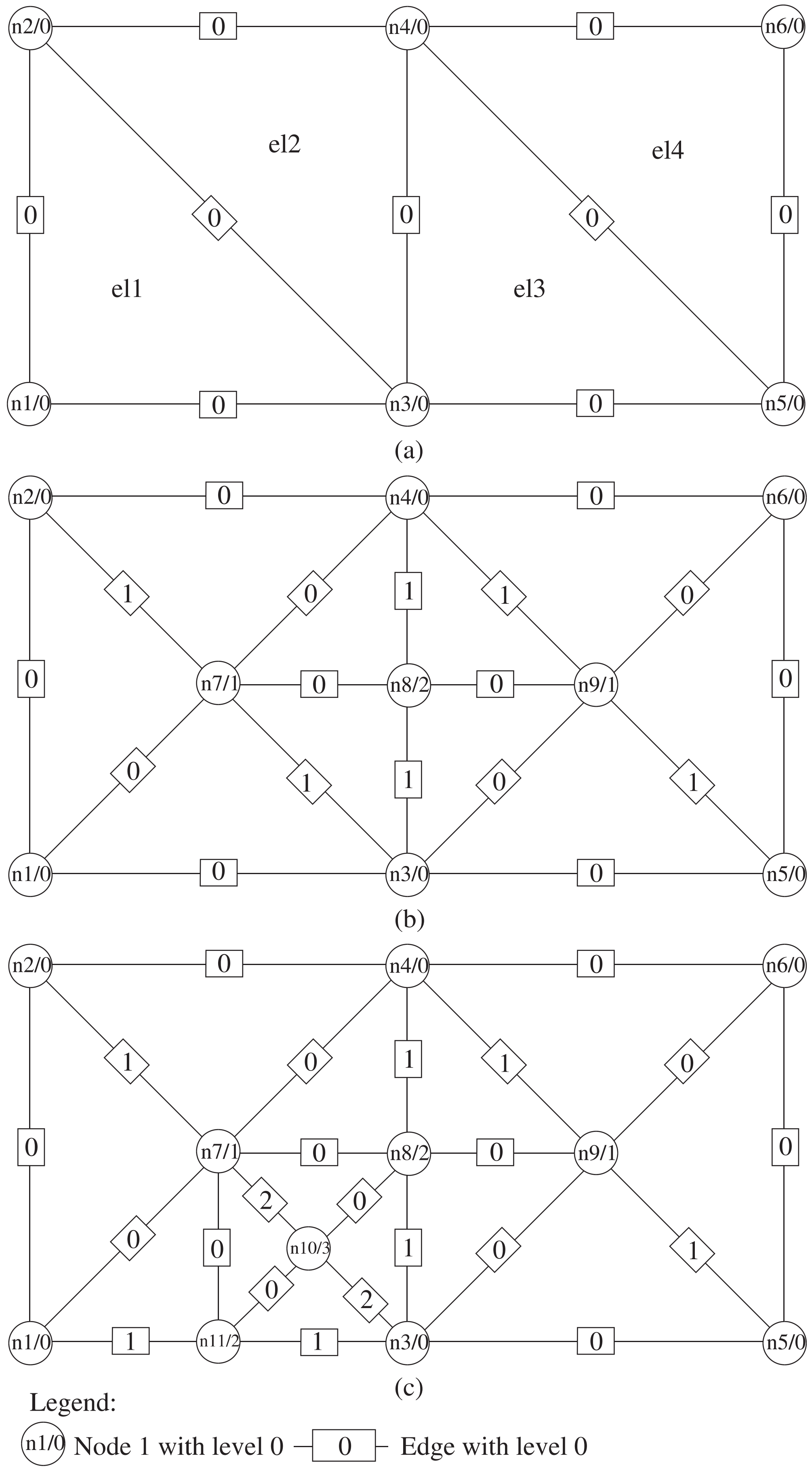

Fig. 2. Refinement/derefinement using level of node and edge

where W denotes the interface width parameter, τ links to the kinetic undercooling, and $f(\phi)$ accounts for the entropy densities,

$$f(\phi) = \begin{cases} (\phi-1)^2 & \text{for} \quad \phi > 0 \\ (\phi+1)^2 & \text{for} \quad \phi < 0 \end{cases} \tag{3}$$

and $h(\lambda\theta)$ is a function defined in [40] and given by,

$$h(\lambda\theta) = \frac{W^2}{2}\left[\left(\frac{d\phi^+}{dz}\right)^2 - \left(\frac{d\phi^-}{dz}\right)^2\right]. \tag{4}$$

The function $g_\delta(\phi)$ is a slightly smoothed step function

$$g_\delta(\phi) = \frac{1}{2}\left(1 + \phi\sqrt{\frac{1+\delta^2}{\phi^2+\delta^2}}\right) \tag{5}$$

with δ set to 0.05.

The anisotropy is included in Eq. (2) by writing the Laplacian $\hat{\nabla}^2\phi$ as a function of the local normal vector $\mathbf{n}$.

$$\begin{aligned} \hat{\nabla}^2\phi &= \nabla\cdot(\eta^2\nabla\phi) \\ &+ \frac{\partial}{\partial\phi_x}\left(|\nabla\phi|^2\eta\frac{\partial\eta}{\partial\phi_x}\right) + \frac{\partial}{\partial\phi_y}\left(|\nabla\phi|^2\eta\frac{\partial\eta}{\partial\phi_y}\right) \\ &+ \frac{\partial}{\partial\phi_z}\left(|\nabla\phi|^2\eta\frac{\partial\eta}{\partial\phi_z}\right) \end{aligned} \tag{6}$$

$$\eta(\phi_x,\phi_y,\phi_z) = (1-3\gamma)\left(1 + \frac{4\gamma}{1-3\gamma}\frac{\phi_x^4+\phi_y^4+\phi_z^4}{|\nabla\phi|^4}\right) \tag{7}$$

where γ is the strength of anisotropy.

Amberg also pointed out that the discontinuous condition on the gradient of ϕ can be written as a source for an integration over the surface of the interface

$$\int_{\phi=0}(\nabla\phi\cdot v)\,d\Gamma \sim \frac{d\phi^+}{dz} - \frac{d\phi^-}{dz} = -\frac{\tau V}{W^2} - \frac{1}{R} = \frac{\lambda\theta}{W} \tag{8}$$

where the symbol V denotes the normal speed of the interface, R is the local radius of curvature, λ is linked to capillary length.

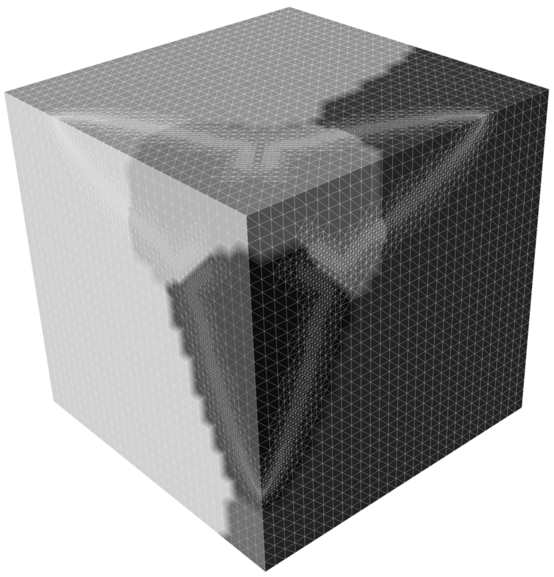
Fig. 3. Mesh partitioning at time $t = 0.093$

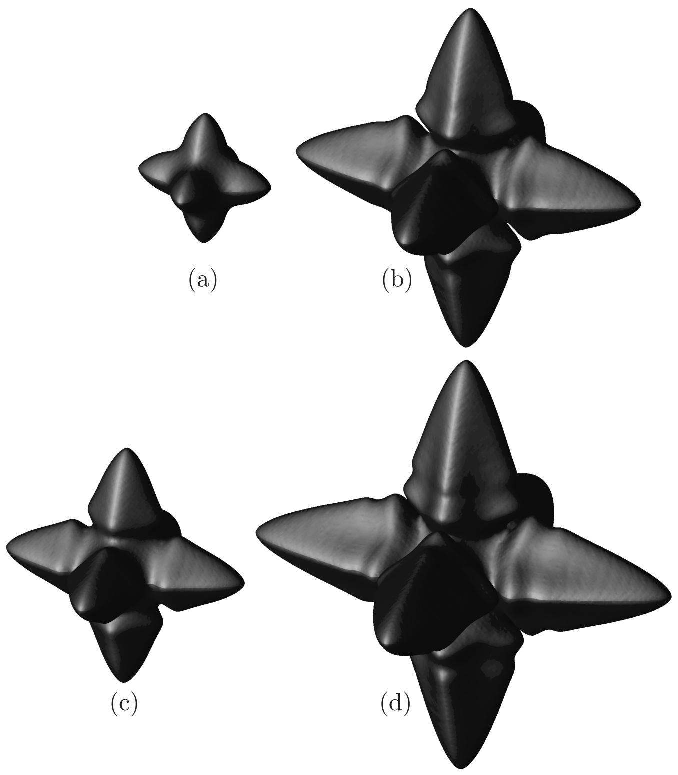

Fig. 4. The 3D simulated of an anisotropy dendrite growth at dimensionless time $t = 0.033$ (A); $t = 0.063$ (B); $t = 0.093$ (C); $t = 0.113$

As a result, the problem consists of solving several coupled time-dependent pde's which are applied to the whole domain, without distinction between the phases and consequently, tracking the solid/liquid interface. However, numerically, this results in a large length scale separation: the diffuse-interface must be much smaller than a typical size of dendritic microstructure and at the same time be highly resolved to get accurate results.

The large scale separation makes the problem appropriate for solving with finite element method on adaptive meshes. The meshes are typically highly non-uniform and change adaptively to follow the evolution of the interface and diffusion fields. A typical example of a mesh evolution for simulation of a 3D dendrite is shown in Fig. 3. Mesh is refined along the vicinity of the interface. Far from the interfaces are discretized with large elements. Color areas shown with every grid correspond to mesh partitions, which in this case, 4 processors were used.

The dendritic growth presented in Fig. 4 is obtained under large driving forces. The solid phase grows so fast that the mesh must be adaptively changed and repartitioned at every time step. The interface arclength as well as the number of nodes increases parabolically in time. This impedes choosing an optimal number of processors to be used in simulations. Figure 5 demonstrates a speedup function at different computational times. The speedup is taken as the ratio of computational time in a single processor and computational time in a number of processors. One observes that when the dendrite is in an early stage of development (number of nodes is small) performance of the code is poor – more processors are used than needed. However, linear speedup for 8 processors is obtained when the microstructure is well-developed and the number of nodes approaches 500000.

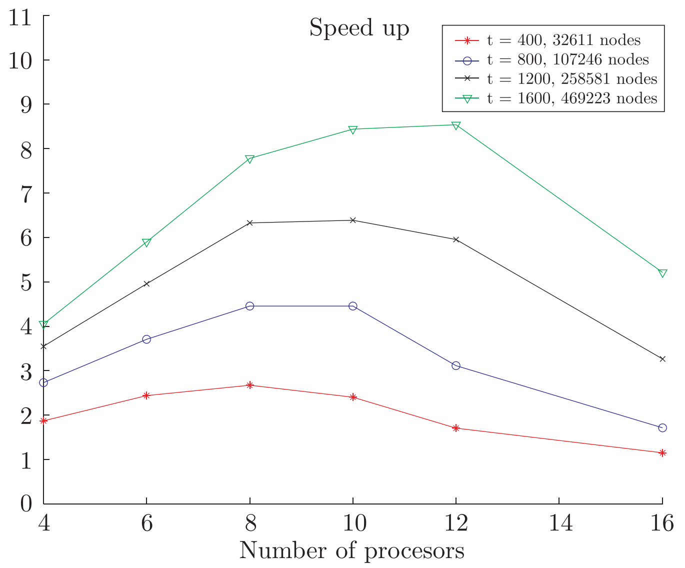

Fig. 5. A speedup function at different computational times

3.2. Capillary-driven flows. To model a basic wetting phenomenon, we consider the case of an isothermal, viscous, and incompressible binary fluid consisting of two components, A and B in a domain Ω. An order parameter, a phase-field C, analogous to the relative concentration of the two components can be introduced to characterize the two different phases. In each bulk phase, C assumes a distinct constant value that changes rapidly but smoothly in the interfacial region. For example, C assumes the value $C_A = -1$ in component A while it takes the value $C_B = 1$ in component B. The transition from C_A to C_B describes the interfacial region. With the introduction of a free energy density, the system can be modelled by a set of equations: the Cahn-Hilliard equation, modified to account for fluid motion, and the Navier-Stokes equations with surface tension forcing and forces due to gravity [41],

$$\begin{gathered}\frac{\partial C}{\partial t} + (\mathbf{u}\cdot\nabla)C = \frac{1}{\mathrm{Pe}}\nabla^2\mu = \frac{1}{\mathrm{Pe}}\nabla^2(\Psi'(C) - \mathrm{Cn}^2\nabla^2 C)\\ \mathrm{Re}\Big(\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u}\Big) = -\nabla p + \nabla^2\mathbf{u} - \frac{1}{\mathrm{Ca}\cdot\mathrm{Cn}}C\nabla\mu\\ \nabla\cdot\mathbf{u} = 0.\end{gathered} \tag{9}$$

where Ψ is a double-well potential and μ is the chemical potential. The dimensionless physical parameters are the Reynolds number Re, Capillary number Ca, and Peclet number Pe given by

$$\mathrm{Re} = \frac{U_c L_c}{\nu},\ \mathrm{Ca} = \frac{2\sqrt{2}\rho_0\nu U_c}{3\sigma},\ \mathrm{Pe} = \frac{2\sqrt{2}L_c U_c\xi}{3\kappa\sigma} \tag{10}$$

where ρ_0, ν, U_c, κ, σ are the mean density, kinematic viscosity, characteristic velocity, mobility, and surface tension, respectively. The Reynolds number is the ratio between the inertial and viscous forces. The Capillary number gives the ratio between the viscous and surface tension forces. The Peclet number is the ratio between the convective and diffusive mass transport. The Cahn number $\mathrm{Cn} = \xi/L_c$ is a dimensionless numerical parameter that provides a measure of the ratio between the mean-field thickness ξ and the characteristic length L_c. The mean-field thickness is directly proportional to the interface thickness [42].

Following Jacqmin [42], two boundary conditions are set for C. First, the no-flux condition $\mathbf{n}\cdot\nabla\phi = 0$. Second, the wetting condition, $\mathbf{n}\cdot\nabla C = -kg'(C)/\mathrm{Cn}$, where k is the wetting coefficient which will be discussed later and $g(C)$ is a local surface energy set to $0.75C - 0.25C^3$. Details of the nondimensionalization can be seen in [41] and application to the study of microdroplet deposition can be found in [43].

The Young's relation which is only valid when $S < 0$ can be defined as $\cos\theta_e = k = S/\sigma_{LM} + 1$ where θ_e is the equilibrium contact angle of the liquid/medium interface at a solid surface.

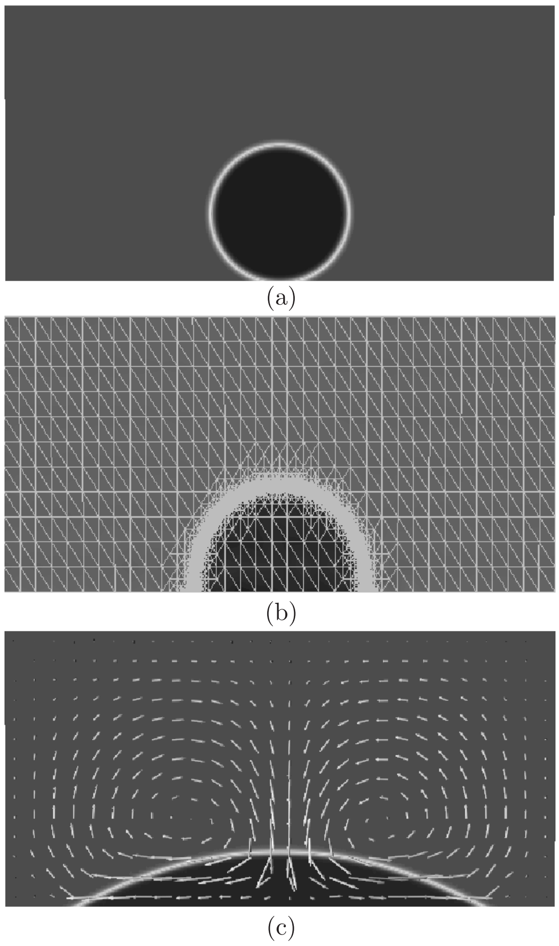

Fig. 6. Wetting of a liquid drop on a solid surface. Concentration field at dimensionless time $t = 0, 10, 200$ with $\mathrm{Ca} = 0.1$, $\mathrm{Re} = 1.0$, $\mathrm{Pe} = 1.5\cdot 10^4$, and $\theta_e \approx 25^\circ$. The mesh and velocity field are superimposed in (b) and (c), respectively

Figure 6 shows a basic wetting phenomena with partial wetting. The parameters are Ca = 1.0, Re = 1.0, Pe = 10^4, and $k = 0.9063$ which corresponds to $\theta_e \approx 25^o$. In Fig. 6b, the adaptive mesh is superimposed and shows fine resolution along the vicinity of the interface. The computation time is found to be 18 times faster using mesh adaptivity than having a uniform mesh [41]. Figure 6c shows the near equilibrium state of the wetting and the velocity field is superimposed exhibiting a symmetric profile with two vortices.

Wettability is the most significant phenomenon in liquid phase sintering [18,44,45]. Some factors affecting wettability include contact angle, particle size, particle shape, and particle arrangement. In Fig. 7, the compact microstructure consists of six solid particles (larger spheres) and thirteen softer drops that are evenly distributed. The drops spread over the solid grains. Phase deformation, coalescence, pore migration and pore elimination take place which are all important microstructural behaviors in liquid phase sintering.

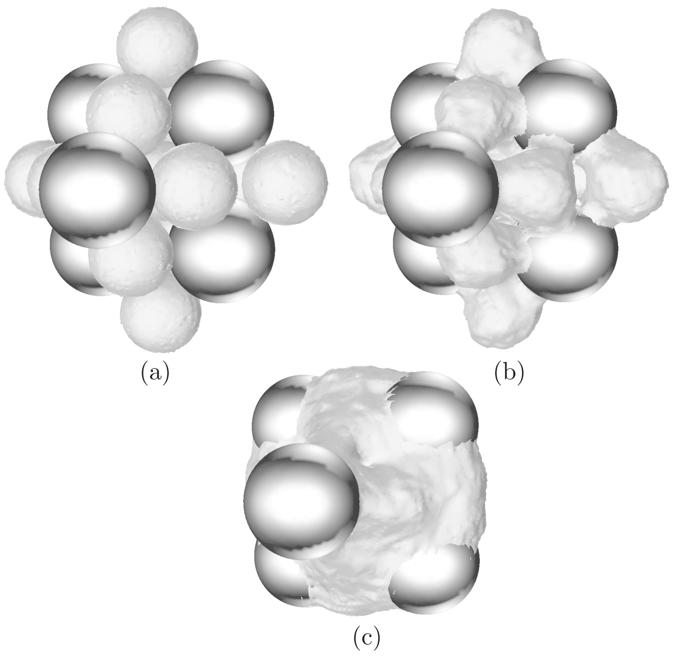

Fig. 7. Three-dimensional generic sintering with a fixed matrix of solid particles (larger spheres). Isosurface at dimensionless time $t = 0, 2, 10$ with Ca = 0.1, Re = 0.1, and Pe = 10^4

4. Conclusions

We have presented numerical simulations of some materials-related microstructural problems using the phase-field method. In such type of problems, implementation using parallel and/or adaptive computation is considered essential. Important issues in combining parallel and adaptive computation were identified; one of them is communication cost reduction. We proposed a scheme to minimize interprocessor communication by keeping the level of node and edge for 2D and the level of node and face for 3D instead of the whole history tree to facilitate derefinement. The local information on the node and edge level can be used to track back to the previous level of refinement. The scheme reduces the communication cost since exchange of information between processors is minimized. Finally, we have successfully implemented the scheme to simulate dendritic growth in 3D with a semisharp phase-field method and wetting dynamics in generic sintering in which important microstructural behaviours in liquid phase sintering have been captured.

References

[1] J.D. Van der Waals. "The thermodynamic theory of capillarity under the hypothesis of a continuous variation of density", *J. Statical Physics* 20, 197–244 (1979).

[2] D.J. Korteweg. "Sur la forme que prennent", *Arch. Neérl. Sci. Extactes Nat. Ser.* II 6, 1–24 (1901).

[3] J.W. Cahn and J.E. Hilliard. "Free energy of a nonuniform system. Interface free energy", *J. Chemical Physics* 58, 258 (1958).

[4] A.A. Wheeler, B.T. Murray, and R.J. Schaefer, "Computation of dendrites using a phase field model", *Physica D* 66(1–2), 243–262 (1993).

[5] G.B. McFadden, A.A. Wheeler, R.J. Braun, and S.R. Coriell, "Phase-field models for anisotropic interfaces", *Phys. Rev. E* 48(3), 2016–2024 (1993).

[6] B.T. Murray, A.A. Wheeler, and M.E. Glicksman, Simulations of experimentally observed dendritic growth behavior using a phase-field model, *J. Crystal Growth* 154 (3–4), 386–400 (1995).

[7] A. Karma and W.J. Rappel, "Quantitative phase-field modelling of dendritic growth in two and three dimensions", *Phys. Rev. E* 57, 4323–4349 (1998).

[8] T.Y. Hou, Z. Li, S. Osher, and P. Zhao, "A hybrid method for moving interface problems with application to the hele-shaw flow", *J. Comp. Phys.* 134(2), 236–247 (1997).

[9] I.-T. Kim, N. Provatas, N. Goldenfeld, and J. Dantzig, "Computation of dendritic microstructure using a level-set method", *Phys. Rev. E* 62(2), 2471–2474 (2000).

[10] D. Juric and G. Tryggvason, "A front-tracking method for dendritic solidification", *J. Comp. Phys.* 123, 127–148 (1996).

[11] P. Zhao and J.C. Heinrich, "Front-tracking ?nite element method for dendritic solidification", *J. Comp. Phys.* 173, 765–796 (2001).

[12] R. Tönhardt and G. Amberg, "Phase-field simulation of dendritic growh in a shear flow", *J. Crystal Growth* 194, 406 (1998).

[13] C. Beckermann, H.J. Diepers, I. Steinbach, A. Karma, and X. Tong, "Modeling melt convection in phase-field simulations of solidification", *J. Comp. Phys.* 154, 468 (1999).

[14] N. Al-Rawahi and G. Tryggvason, "Numerical simulation of dendritic solidification with convection: Two-dimensional geometry", *J. Comp. Phys.* 180(2), 471–496 (2002).

[15] S. Osher and J.A. Sethian, "Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations", *J. Comp. Phys.* 79, 12–49 (1998).

[16] H. Zhang, L.L. Zheng, V. Prasad, and T.Y. Hou, "A curvilinear level set formulation for highly deformable free surface problems with application to solidification", *Numer. Heat Tr. B* 43, 1–20 (1998).

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

[18] R. German, *Liquid Phase Sintering*, Plenum Press, N.Y., 1985.

[19] P.G. de Gennes, F. Brochard-Wyart, and D. Quéré, *Capillarity and Wetting Phenomena*, Springer-Verlag, N.Y., 2004.

[20] Y. Jaluria and K.E. Torrance, *Computational Heat Transfer*, Taylor & Francis, 2003.

[21] A. Laszloffy, J. Long, and A. Patra, "Simple data management, scheduling and solution strategies for managing the irregularities in parallel adaptive *hp* fnite element simulations", *Parallel Computing* 26, 1765–1788 (2000).

[22] M.C. Rivara, "Selective re‘finement/derefinement algorithms for sequences of nested triangulations", *Int. J. Numer. Meth. Eng.* 28, 2889–2906 (1989).

[23] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs", *SIAM J. Sci. Comput.* 20(1), 359–392 (1998).

[24] J.E. Flaherty, R.M. Loy, C. Özturan, M.S. Shepard, B.K. Szymanski, J.D. Teresco, and L.H. Ziantz, "Parallel structures and dynamic load balancing for adaptive finite element computation", *Appl. Numer. Math.* 26, 241–163 (1998).

[25] R. Diekmann, R. Preis, F. Schlimbach, and C.Walshaw, "Shape-optimized mesh partitioning and load balancing for parallel adaptive fem", *Parallel Computing* 26, 1555–1581 (2000).

[26] C. Walshaw and M. Cross, "Mesh partitioning: a multilevel balancing and refinement algorithm", *SIAM J. Sci. Comput.* 22, 63–80 (2000).

[27] J.G. Castanos, *Parallel Adaptive Unstructured Computation.* PhD thesis, Department of Computer Science, Brown University, Providence, 2000.

[28] N. Touheed, P. Selwood, P.K. Jimack, and M. Berzins, "A comparison of some dynamic load-balancing algorithms for a parallel adaptive flow solver", *Parallel Computing* 26, 1535–1554 (2000).

[29] L. Oliker, R. Biswas, and H.N. Gabow, "Parallel tetrahedral mesh adaptation with dynamic load balancing", *Parallel Computing* 26, 1583–1608 (2000).

[30] A. Bose and G.F. Carey, "A class of data structures and object-oriented implementation for finite element methods on distributed memory systems", *Comput. Methods Appl. Mech. Eng.* 171, 109–121 (1999).

[31] A. Stagg, J. Hallberg, and J. Schmidt, "A parallel, adaptive refinement scheme for tetrahedral and triangular grids", *Lecture Notes in Computer Science*, 512–518 (2000).

[32] J. Jeong, N. Goldenfeld, and A. Dantzig, "Phase field model for three dimensional dendritic growth with fluid flow", *Physical Review E* 64, 041602:1–14 (2001).

[33] R. Niekamp and E. Stein, "An object-oriented approach for parallel two and three-dimensional adaptive finite element computations", *Computers and Structures* 80317–328, (2002).

[34] J. Waltz, "Parallel adaptive refinement for unsteady flow calculations on 3d unstructured grids", *Int. J. Numer. Meth. Fluids* 46, 37–57 (2004).

[35] G. Amberg, R. Tönhardt, and C. Winkler, "Finite element simulations using symbolic computing", *Math. and Comp. in Sim.* 49, 257–274 (1999).

[36] G. Karypis, K. Schloegel, and V. Kumar, *ParMetis 3.1: Parallel Graph Partitioning and Sparse Matrix Ordering Library*, University of Minnesota, Minneapolis, 2003.

[37] `http://www.cs.sandia.gov/crf/aztec1.html`

[38] `http://www.sgi.com/tech/stl`

[39] Message Passing Interface Forum, *MPI: A Message-Passing Interface Standard*, 1994.

[40] G. Amberg, "Semisharp phase field method for quantitative phase change simulations", *Phys. Rev. Lett.* 91(26), 265505–265511 (2003).

[41] W. Villanueva and G. Amberg, "Some generic capillary-driven flows", *Int. J. Multiphase Flow* 32, 1072–1086 (2006).

[42] D. Jacqmin, "Contact-line dynamics of a diffuse fluid interface", *J. Fluid Mech.* 402, 57–88 (2000).

[43] W. Villanueva, J. Sjodahl, M. Stjernström, J. Roeraade, and G. Amberg, "Microdroplet deposition under a liquid medium", Langmuir 23, 1171–1177 (2007).

[44] F.V. Motta, R.M. Balestra, S. Ribeiro, and S.P. Taguchi, Wetting behaviour of sic ceramics, part I", *Materials Letters* 58, 2805–2809 (2004).

[45] S.P. Taguchi, F.V. Motta, R.M. Balestra, and S. Ribeiro, "Wetting behaviour of sic ceramics, part II", *Materials Letters* 58, 2810–2814 (2004).