Three-dimensional modeling of a thermal dendrite using the phase field method with automatic anisotropic and unstructured adaptive finite element meshing

Carole Sarkis, Luisa Silva, Charles-André Gandin, Mathis Plapp

Three-dimensional modeling of a thermal dendrite using the phase field method with automatic anisotropic and unstructured adaptive finite element meshing

Three-dimensional modeling of a thermal dendrite using the phase field method with automatic anisotropic and unstructured adaptive finite element meshing

C Sarkis[1a], L Silva[1b], Ch-A Gandin[1c] and M Plapp[2d]
[1] MINES ParisTech, CEMEF UMR CNRS 7635, CS10207, 06904 Sophia Antipolis, France
[2] Ecole Polytechnique, LPMC UMR CNRS 7643, 91128 Palaiseau, France

Email: [a]carole.sarkis@mines-paristech.fr, [b]luisa.silva@mines-paristech.fr, [c]charles-andre.gandin@mines-paristech.fr, [d]mathis.plapp@polytechnique.fr

Abstract. Dendritic growth is computed with automatic adaptation of an anisotropic and unstructured finite element mesh. The energy conservation equation is formulated for solid and liquid phases considering an interface balance that includes the Gibbs-Thomson effect. An equation for a diffuse interface is also developed by considering a phase field function with constant negative value in the liquid and constant positive value in the solid. Unknowns are the phase field function and a dimensionless temperature, as proposed by [1]. Linear finite element interpolation is used for both variables, and discretization stabilization techniques ensure convergence towards a correct non-oscillating solution. In order to perform quantitative computations of dendritic growth on a large domain, two additional numerical ingredients are necessary: automatic anisotropic unstructured adaptive meshing [2,[3] and parallel implementations [4], both made available with the numerical platform used (CimLib) based on C++ developments. Mesh adaptation is found to greatly reduce the number of degrees of freedom. Results of phase field simulations for dendritic solidification of a pure material in two and three dimensions are shown and compared with reference work [1]. Discussion on algorithm details and the CPU time will be outlined.

1. Introduction

The phase field approach is a method of choice for simulating interfacial pattern formation phenomena in solidification. The widely recognized appeal of this approach is to avoid the explicit tracking of macroscopically sharp phase boundaries. This makes it better suited than more conventional front tracking methods to simulate time-dependent free boundary problems in three dimensions (3D) or when complex geometries are involved. Tracking is avoided by introducing an order parameter, or phase field ϕ, which varies smoothly from one value in the liquid to another value in the solid across a spatially diffuse interface region related with a thickness W. This field naturally distinguishes the solid and liquid phases and converts the problem of simulating the advance of a sharp boundary to that of solving a stiff system of partial differential equations that govern the evolution of the phase and diffusion fields.

The main difficulty when solving numerically phase field models is due to the very rapid change of the phase field across the diffuse interface, which thickness has to be taken small enough to correctly capture the physics of the phase transformation. A high spatial resolution is therefore needed to describe the smooth transition. In order to reduce the computational time and the number of grid points, adaptive anisotropic and unstructured finite elements have been used. In the past, other authors have performed numerical simulations in 2D and 3D using the finite element method and the phase field model with an adaptive isotropic mesh based on an error estimator [5,[6]. Others used the phase field model with an adaptive structured isotropic mesh using finite difference method [7[**10**] and finite element method [9-[11]. And some of them used the level set model with an adaptive structured isotropic mesh and finite element method [12].

A first numerical goal of this paper is to present the influence of using an adaptive mesh [2, [3] on the computational cost adding the time adaptation and the element adaptation with parallel computations [4]. The second goal of this paper is to demonstrate that accurate quantitative solutions of the free-boundary problem given by the thin-interface limit are recovered by the finite element method. We present the results of simulations of dendritic growth in 2D and 3D. We take advantage of the crystal symmetries to reduce computation time.

2. Modeling

In solidification of a pure system, thermal dendritic growth is generally described by stating conservation of energy and using the Gibbs-Thomson relation to establish the normal velocity of propagation of the solid-liquid interface [13-15], providing a sharp interface formulation of our solidification problem. In a diffuse interface context, instead of solving the equations on each solid/liquid domain with the given interface conditions, we may obtain a set of equations valid in the whole domain by using a free energy approach.

2.1. Energy diffusion

We will use the symmetric model (α = const, c_p = const). Let us consider the dimensionless temperature, $q = c_p\left(\frac{T - T_M}{L}\right)$, with T the temperature, c_p the heat capacity, L the latent heat and T_M the melting temperature. We thus solve the energy diffusion equation:

$$\frac{\partial q}{\partial t} - a\,\mathrm{D}q = \frac{1}{2}\cdot\frac{\partial f}{\partial t} \tag{1}$$

with α the dimensionless thermal diffusion. The Gibbs-Thomson equation allows us to establish the relation at the solid/liquid interface $\theta = -\,d_0(\boldsymbol{n})\kappa - \beta(\boldsymbol{n})v_n$ with $d_0 = \Gamma c_p/L$ and $\beta = c_p/\mu_k L$, $\boldsymbol{n}$ being the normal vector, v_n the normal velocity at the interface ($v_n = \boldsymbol{v}.\ \boldsymbol{n}$), $\Gamma = \sigma^{s/l}T_M/\rho L$, the Gibbs-Thomson coefficient, $\sigma^{s/l}$ being the interfacial energy and ρ the density, μ_k the interface mobility. Thus, d_0 is the capillary length and β is the kinetic coefficient.

2.2. Phase field formulation

Let us consider the solidification of a pure material. We define ϕ as a function which describes the presence of the liquid and the solid phases in the computational domain, Ω. This function varies between −1 in the liquid and 1 in the solid. Let us suppose that the solution of our phase field problem is $\phi = -\tanh(\eta/\sqrt{2})$. In the expression, η is the signed distance to the solid/liquid interface. The variational derivative of the free energy functional provides the evolution equation for ϕ which, after manipulation and taking into account the solution form to use in our solver, is

$$\frac{1}{M_\phi}\frac{\partial\phi}{\partial t} - W^2(\boldsymbol{n})\Delta\phi - 2\,W(\boldsymbol{n})\,\nabla W(\boldsymbol{n}).\nabla\phi = [\phi - \lambda\theta(1-\phi^2)](1-\phi^2) + \sum_i \frac{\partial}{\partial i}\left(|\nabla\phi|^2\, W(\boldsymbol{n})\,\frac{\partial W(\boldsymbol{n})}{\partial\,\partial i\ \phi}\right) \tag{2}$$

In this expression $\frac{1}{M_\phi} = \tau(\boldsymbol{n}) = \tau_0\, a_s(\boldsymbol{n})^2$, M_ϕ being the molecular mobility, $W(\boldsymbol{n}) = W_0\, a_s(\boldsymbol{n})$ being the interface anisotropy and we write $\lambda = \frac{\alpha\tau_0}{W_0^2\,\mathrm{a}_2}$. $\partial_i\phi = \frac{\partial\phi}{\partial i}$ with $i = x, y$ in 2D and $i = x, y, z$ in 3D. We take $\mathrm{a}_1 = 0.8839$, $\mathrm{a}_2 = 0.6267$, $\tau_0 = 1$ and $W_0 = 1$.

In 2D, several authors have proposed the form [16]:

$$a_s(\boldsymbol{n}) = \left(1 + \varepsilon_4 \cos\left(4 \arctan \frac{\partial \phi_y}{\partial \phi_x}\right)\right) \tag{2}$$

The following derivative terms for the phase field formulation are thus computed

$$\frac{\partial}{\partial x}\left(|\nabla\phi|^2\, W(\boldsymbol{n})\frac{\partial W(\boldsymbol{n})}{\partial\, \partial_x\phi}\right) = -\frac{\partial}{\partial x}\big(W(\boldsymbol{n})\,(W(\boldsymbol{n}))'\,\partial_y\phi)\big) \text{ and } \frac{\partial}{\partial y}\left(|\nabla\phi|^2\, W(\boldsymbol{n})\frac{\partial W(\boldsymbol{n})}{\partial\, \partial_y\phi}\right) = \frac{\partial}{\partial y}\big(W(\boldsymbol{n})\,(W(\boldsymbol{n}))'\,\partial_x\phi)\big)$$

ε_4 indicates the strength for the anisotropic arms that we have. A more general form in 2D and 3D is given by [17]:

$$W(\boldsymbol{n}) = W_0(1 - 3\varepsilon_4)\left[1 + \frac{4\varepsilon_4}{1-3\varepsilon_4}\frac{(\partial_x\phi)^4 + (\partial_y\phi)^4 + (\partial_z\phi)^4}{|\nabla\phi|^4}\right] \tag{3}$$

2.3. *Numerical resolution*

In following section, we will discuss the numerical parameters and the mesh grid, computed from a metric, needed for the simulation, followed by the numerical resolution, using the phase field method, showing the propagation of the solid-liquid interface.

2.3.1. Phase field and energy solver. In what concerns the resolution of these equations, we have used the finite element method with a continuous approximation for both phase field and temperature functions in the mesh and stabilized resolution to take into account very small diffusion coefficients. In fact, numerical solution of convection-diffusion-reaction equations such as the ones we are treating, using a classical Galerkin formulation, normally exhibits global oscillations in convection-dominated problems, especially in the vicinity of sharp gradients. We use the SUPG (Streamline Upwind Petrov Galerkin) stabilization method to solve this problem by adding a perturbation term to the weighting functions with the aim to get an oscillation-free solution. The linear system of equations issuing from the discretization is solved implicitly using the conjugate bi gradient-least squares method (BCGSL). There is also preconditioning to the resolution using the Jacobi method with incomplete factorization LU per block of size 2.

Advantages of solving the phase field equation as a convection-diffusion problem concern mainly the diffusion term, $\Delta\phi$. In fact, this term, treated implicitly, will naturally smooth singularities in the interface shape. However, convergence towards the sharp interface solution is conditioned by the fact that there is a diffusion layer related with W_0, the thickness that is directly involved in such an equation.

2.3.2. Mesh adaptation. The mesh is initially (and throughout time) adapted using a topological mesher [2][3] that is incorporated to our library and that is based on a metric field, given at the nodes of the mesh. M is a unit metric field associated with any unstructured mesh. The metric is built using the affine transformation to a reference element which has to be equilateral of edge length equal to unity. It provides both the size and the stretching of elements. In our case, this field can be computed using the edge vectors of the mesh, $X_{ij} = X_j - X_i$, i and j being the extreme nodes of the edge. Starting from an existing mesh, the new nodal metrics field M_i we provide to the mesher is

$$M_i = \left(\frac{1}{q}\sum_{j\in G(i)} s_{ij}^2\, X_{ij} \otimes X_{ij}\right)^{-1} \tag{5}$$

q being the space dimension, Γ (i) being the set of nodes connected to node i. s the stretching factor applied to obtain the new edge size. The edge stretching factor, $s_{ij,}$ is obtained from the *a posteriori* estimated error, and is given by

$$s_{ij} = \left(L/e_{ij}\right)^{1/p} \tag{6}$$

where e represents the edge error and p a stretching exponent, $1 < p < q$, q being the dimension. Starting from a given element, we examine what information we can construct from the set of edges. Since more than only two edges can be encountered for a node, it is necessary then to find an approximation or an averaging process of the information. For this reason, we first state that the length size of the edges sharing a given node is exactly the interpolation of the continuous length distribution function defined in the space at the considered point. In this technique we computed the error along and in the direction of each edge.

$$L = \left(\mathrm{S}_{\mathrm{i}} \ \mathrm{S}_{j \in G(i)} \ e_{ij}^{\frac{p}{p+2}} / A \right)^{\frac{p+2}{p}} \tag{7}$$

The error is computed using the recovered gradient of the solution on which we wish to adapt (phase field, temperature, or both). We construct a solution vector that contains the two fields and we compute its gradient, $U = (\phi, \theta)$.

In fact, $e_{ij} = \max(|\nabla U \ . \ X_{ij}|; \ e_{min}|X_{ij}|^2)$ and $\nabla U = (X_i)^{-1}.U_i$, where $X_i = \dfrac{1}{|G(i)|} \mathrm{S}_{j \in G(i)} X_{ij} \otimes X_{ij}$ is the distribution function and $U_i = \mathrm{S}_{j \in G(i)} U_{ij} \ X_{ij}$. e_{min} is a chosen constant. A is the number of element given.

In what concerns the referred parameters, when we increase the total number of elements, the mesh is enriched around the interface and we decrease the error. If we want a small error, we should decrease the mesh size, but we also need to increase the number of nodes at the same time. If not, we will increase the error because we do not have enough elements to perform the simulation.

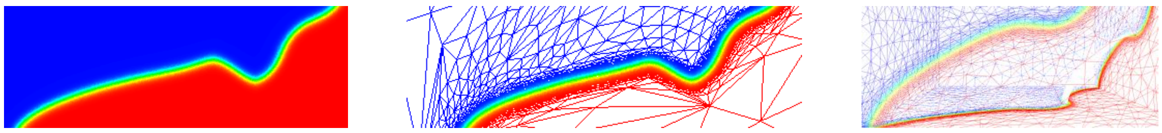


Figure 1. (a) Isotropic mesh, (b) Anisotropic adaptive mesh (2D), c) Anisotropic adaptive mesh (3D).

Figure 1 illustrates the difference between the mesh size inside the interface and outside the interface. We have smaller sizes at the interface and larger outside. The mesh is adapted according to θ and the function ϕ that indicates the interface. The red color represents the solid, the blue one represents the liquid and the green one is the interface.

For a matter of CPU time optimization, we may define a remesh frequency. For that, we compute the norm of the Gibbs-Thomson velocity v and we find the maximum velocity $v_{\max}$, over the entire domain as:

$$v = \frac{-\theta}{\beta} - d_0 K \quad \text{with } \beta = \frac{a_1 \tau}{\lambda \, W(\boldsymbol{n})}, \ d_0 = \frac{a_1 W(\boldsymbol{n})}{\lambda}, \text{ the curvature } K = \nabla \cdot \boldsymbol{n}$$

We compute the displacement $v_{\max}.\Delta t$ and we sum the displacements for each time step. When the sum of the displacements exceeds $3W_0/2$, remeshing is activated.

2.3.3. Time adaptation. Improvements on CPU time can also be obtained automatically by adapting the time step Δt as:

$$\Delta t = h_{min}/(10 \cdot v_{max}) \tag{8}$$

h_{min} is the minimal mesh size.

2.3.4. Adaptation of the prescribed number of elements. When a mesh adaptation step is performed, the "interface volume", defined as the volume occupied by the interface thickness of the phase field function, $V_{\text{interface}}$ (surface in 2D) may be obtained as follows:

$$V_{\text{Interface}} = \int_{-\infty}^{\infty} \delta_\phi \ \mathrm{dV} \text{ with } \delta_\phi = \begin{cases} 0 & \text{for } \phi < -E \\ \dfrac{1}{2E} + \dfrac{1}{2E}\cos\left(\dfrac{\pi\phi}{E}\right) & \text{for } -E < \phi < E \\ 0 & \text{for } \phi > E \end{cases} \tag{9}$$

with $E = 1$. Then we compute the number of elements needed at the interface as

$$N_1 = S_{\text{Interface}}/S_{\text{element}} = 2.S_{\text{interface}}/h_{min}^{\ 2} \quad \text{in 2D}$$

$$N_1 = V_{\text{Interface}}/V_{\text{element}} = 6.V_{\text{interface}}/h_{min}^{\ 3} \quad \text{in 3D}$$

Finally we compute the total number of element needed as $NE = N_1 + N_2$ with N_2 a constant, to add a certain number of elements outside the interface thickness. This adaptation is done because, during dendritic growth, the number of elements should largely increase during the computation because the surface of the interface increases, and so we need more elements to represent it.

3. Numerical Results

3.1. Convergence method.

Let us consider a rectangular domain [0;1000]x[0;300] on which we place an initial seed of size 5. To study the convergence of our method, we compute the tip velocity in the x direction. The anisotropy function for growth is $W(\boldsymbol{n}) = W_0(1 - 3\varepsilon_4)\left[1 + \dfrac{4\varepsilon_4}{1 - 3\varepsilon_4}\dfrac{(\partial_x\phi)^4 + (\partial_y\phi)^4}{|\nabla\phi|^4}\right]$ with $\varepsilon_4 = 0.05$. Other simulation parameters are: $\alpha = 1$, $\tau_0 = 1$, $\Delta t = h_{min}/(10 \cdot V_{max})$. In the following, we compare the solution for different values of the minimal mesh size h_{min}.

In figures 2 and 3, we observe that the tip velocity decreases with time and becomes constant until it decreases instantly to become very small when the tip reaches the border of the domain. On the left, we represent the sensitivity of the solution to the mesh's minimal size; on the right, the steady-state velocity computed value, compared with the value obtained using a green function calculation for an analytical test value. We see that with decreasing the minimal mesh size in the interface the velocity converges to the analytical solution.

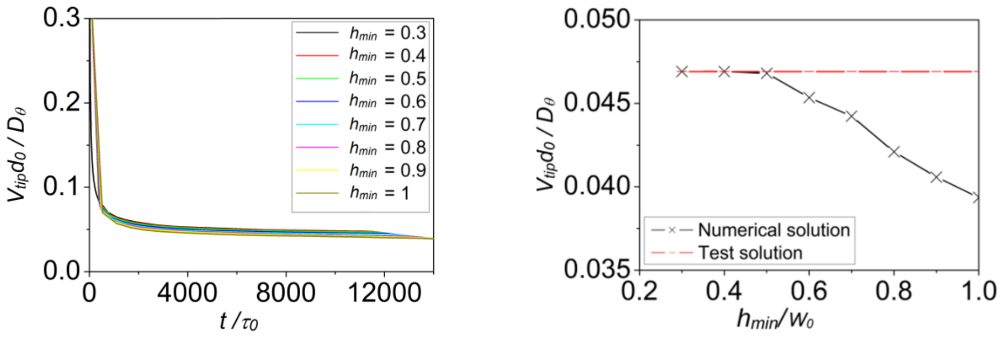
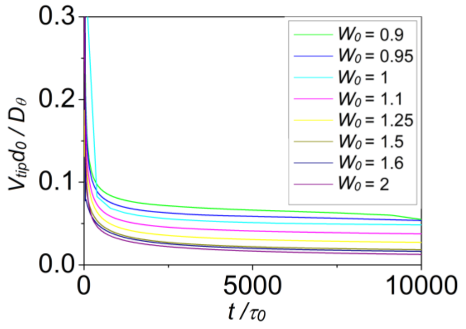


Figure 2. Dimensionless tip velocity as a function of grid spacing for $\Delta = 0.65$, $d_0/W_0 = 0.554$ and $N_2 = 25\,000$. The red line corresponds to the value obtained from the green function calculation [1].



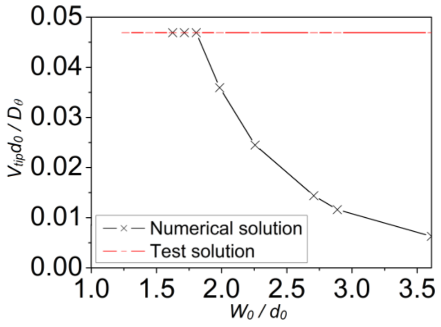


Figure 3. Dimensionless tip velocity as a function of interface thickness for $\theta_0 = -0.65$. The red line corresponds to the value obtained from the green function calculation.

3.2. Computational time

We will show the difference on the computational time with different conditions for a rectangular domain [1000;300] with $\theta_0 = -0.65$, $\varepsilon_4 = 0.05$, $\alpha = 1$ and $e_{min} = 10^{-7}$.

Table 1. Table for different time computation with different conditions

Case	V_{Tip}	***NE***	h_{min}	***Nb_Proc***	***Fr***	**Adapt Time**	**Adapt *NE***	T_{CPU}
1	0.0469	4 638 842	0.4	1	No	No	No	36 000h*
2	0.0469	4 638 842	0.4	4	No	No	No	2 400h*
3	0.0469	4 638 842	0.4	16	No	No	No	1 240h*
4	0.0485	100 000	0.4	16	1	No	No	504h
5	0.051	80 000	0.4	16	Yes	No	No	80h
6	0.05	80 000	0.4	16	Yes	Yes	No	13h
7	0.0469	46 000	0.4	16	Yes	Yes	Yes(N_2 = 25000)	10h

* Predicted time

We observe the influence on CPU time using the parallel computation from the first three cases, using 1, 4 or 16 processors. From case 3 and case 4, we show the decreased CPU time using the mesh adaptation. From case 4 to case 5, we show the influence of the mesh adaptation with the frequency computed above depending on the velocity. In case 6 and 7 we add the time adaptation and the adaption of the number of element. The graphs in figure 4 show the CPU time for the different cases.

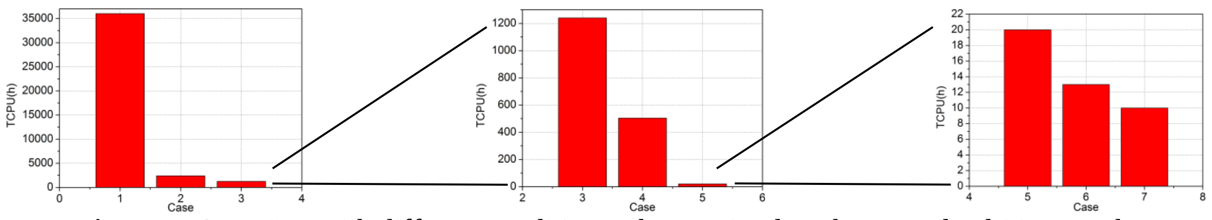


Figure 4. CPU time with different conditions taken to simulate the same dendritic growth.

In Table 2, we show that using our model we can obtain exactly the same tip velocity, for different parameters, computed using the green function supposed as a test solution.

Table 2. Table to compare the tip velocity

Δ	ε_4	α	d_0/W_0	V_{Tip}	V_{Tip}^{GF}	% error	Domain	*NE*	T_{CPU}
0.65	0.05	1	0.554	0.0469	0.0469	0	[1000;300]	46 000	10h
0.55	0.05	2	0.277	0.017	0.017	0	[1000;300]	44 000	6h 30min
0.55	0.05	4	0.139	0.017	0.017	0	[500;150]	34 000	2h 30 min
0.45	0.05	4	0.139	0.00545	0.00545	0	[1000;300]	50 000	5h 50 min
0.55	0.02	2	0.277	0.00685	0.00685	0	[1000;300]	30 000	7h 30 min

4. Conclusion

We have presented a quantitative phase field model that replicates previous published results [1] while taking advantage of an anisotropic adaptive mesh based on an error estimator, with variable number of elements and parallel computations with time adaptation. The computational cost is then shown to be decreased by a factor of 3600 (case 1 to case 7) using 16 CPU.

We have done 3D simulation for this model to show dendritic growth using the adaptive anisotropic and unstructured finite element mesh

Figure **5** and taking advantage of the symmetry. Next step is to simulate a thermal-solute dendritic growth with physical parameters using the kinetic anisotropy for Al-Cu.

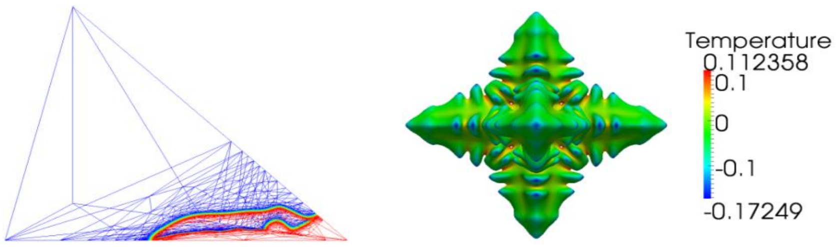


Figure 5. 3D thermal dendritic growth, $\varepsilon_4 = 0.05$, $\alpha = 1$, $\theta_0 = -0.65$, Domain = 1000.

References

[1] Karma A and Rappel W J 1998 Quantitative phase field modeling of dendritic growth in two and three dimensions *Physical review E* **57** 4323-4349
[2] Coupez T 2011 Metric construction by length distribution tensor and edge based error for anisotropic adaptive meshing *Journal of Computational Physics* 230 2391-2405
[3] Coupez T, Silva L and Hachem E (to appear) 2014 Implicit boundary and adaptive anisotropic meshing, *Tetrahedron IV, SEMA SIMAI Springer Series*.
[4] Digonnet H, Silva L and Coupez T 2007 Cimlib: A fully parallel application for numerical simulations based on components assembly *Air Conference Proceedings* **908** 269-274
[5] Narski J and Picasso M 2005 Adaptive finite elements with high aspect ratio for dendritic growth of a binary alloy including fluid flow induced by shrinkage *Elsevier Science* 1-9
[6] Narski J and Picasso M 2006 Adaptive 3D finite elements with high aspect ratio for dendritic growth of a binary alloy including fluid flow induced by shrinkage *FDMP* **1-1** 1-13
[7] Tonhard R and Amberg G 1998 Phase-field simulation of dendritic growth in shear flow *Journal of Crystal Growth* **194** 406-425
[8] Mullis A.M et al. 2008 An adaptive, fully implicit multigrid phase-field model for the quantitative simulation of non-isothermal binary alloy solidification *Acta Materiala* **56** 4559 4569

[9] Mullis A.M et al. 2011 An adaptive, multilevel scheme for the implicit solution of three dimensional phase-field equations *Wiley online library* **27** 106-120
[10] Lan C.W et al. 2011 Adaptive three-dimensional phase-field modeling of dendritic crystal growth with high anisotropy *Journal of Crystal Growth* **318** 51-54
[11] Provatas N et al. 1999 Adaptive Mesh Refinement Computation of Solidification Microstructures Using Dynamic Data Structures *Journal of computational physics* **148** 265-290
[12] Tan L and Zabaras N 2007 A levelset simulation of dendritic solidification of multi-component alloys *Journal of Computational Physics* **221** 9-40
[13] Heringer H, Gandin Ch-A, Lesoult G and Henein H 2006 Atomized droplet solidification as an equiaxed growth model *Acta Materialia* **54** 4427-4440
[14] Rathjen K A and Jiji L M 1971 Heat conduction with melting or freezing in a corner *Journal of Heat Transfer* **54** 101-108
[15] Provatas N, Goldenfeld N and Dantzig J1999 Adaptive mesh refinement computation of solidification microstructures using dynamic data structures *Journal of Computational Physics* **148** 265-290
[16] Plapp M and Karma A 2000 Multiscale Finite-Difference-Diffusion-Monte-Carlo Method for Simulating Dendritic Solidification *Journal of Computational Physics* **165,** 592–619
[17] Friedli J 2011 Interfacial energy anisotropy and growth morphologies in aluminum-zinc alloys *École polytechnique Fédérale de Lausanne* thesis