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

To cite this version:


HAL Id: hal-01353713

https://hal-mines-paristech.archives-ouvertes.fr/hal-01353713

Submitted on 12 Aug 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Three-dimensional modeling of a thermal dendrite using the phase field method with automatic anisotropic and unstructured adaptive finite element meshing

This content has been downloaded from IOPscience. Please scroll down to see the full text.
(http://iopscience.iop.org/1757-899X/117/1/012008)

View the table of contents for this issue, or go to the journal homepage for more

Download details:

IP Address: 194.167.55.2
This content was downloaded on 12/08/2016 at 14:17

Please note that terms and conditions apply.
Three-dimensional modeling of a thermal dendrite using the phase field method with automatic anisotropic and unstructured adaptive finite element meshing

C Sarkis\textsuperscript{1a}, L Silva\textsuperscript{1b}, Ch-A Gandin\textsuperscript{1c} and M Plapp\textsuperscript{2d}

\textsuperscript{1} MINES ParisTech, CEMEF UMR CNRS 7635, CS10207, 06904 Sophia Antipolis, France
\textsuperscript{2} Ecole Polytechnique, LPMC UMR CNRS 7643, 91128 Palaiseau, France

Email: \textsuperscript{a}carole.sarkis@mines-paristech.fr, \textsuperscript{b}luisa.silva@mines-paristech.fr, \textsuperscript{c}charles-andre.gandin@mines-paristech.fr, \textsuperscript{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 $\phi$, 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
interface anisotropy and we write the variational derivative of the free energy functional provides the evolution equation for

\[ \frac{\partial q}{\partial t} + a \nabla q = \frac{1}{2} \frac{\partial f}{\partial \phi} \]  

(1)

with \( \alpha \) the dimensionless thermal diffusion. The Gibbs-Thomson equation allows us to establish the relation at the solid/liquid interface \( \theta = -d_0(n) \lambda - \beta(n) \nu_n \) with \( d_0 = \Gamma c_p / \rho L \) and \( \beta = c_p / \mu L \), \( n \) being the normal vector, \( \nu_n \) the normal velocity at the interface \( (\nu_n = \nabla \cdot \mathbf{v}) \), \( \Gamma = \sigma^\ell T_m / \rho L \), the Gibbs-Thomson coefficient, \( \sigma^\ell \) being the interfacial energy and \( \rho \) the density, \( \mu \) the interface mobility. Thus, \( d_0 \) is the capillary length and \( \beta \) is the kinetic coefficient.

2.2. Phase field formulation

Let us consider the solidification of a pure material. We define \( \phi \) as a function which describes the presence of the liquid and the solid phases in the computational domain, \( \Omega \). 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 \( \phi = -\tanh \left( \eta / \sqrt{2} \right) \). In the expression, \( \eta \) is the signed distance to the solid/liquid interface. The variational derivative of the free energy functional provides the evolution equation for \( \phi \) which, after manipulation and taking into account the solution form to use in our solver, is

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

(2)

In this expression \( \frac{1}{M_\phi} = \tau(n) = \tau_0 a_i(n) \), \( M_\phi \) being the molecular mobility, \( W(n) = W_0 a_i(n) \) being the interface anisotropy and we write \( \lambda = \frac{\alpha_0}{W_0^2 a_i} \), \( \partial_i \phi = \frac{\partial \phi}{\partial t} \) with \( i = x, y \) in 2D and \( i = x, y, z \) in 3D. We take \( a_1 = 0.8839, a_2 = 0.6267, \tau_0 = 1 \) and \( W_0 = 1 \).
In 2D, several authors have proposed the form [16]:

$$a_s(n) = \left( 1 + \epsilon_s \cos \left( 4 \arctan \frac{\partial \phi_j}{\partial \phi_i} \right) \right)$$

(2)

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

$$\frac{\partial}{\partial x} \left( \nabla \phi \right)^2 W(n) \frac{\partial W(n)}{\partial \phi} = - \frac{\partial}{\partial x} \left( W(n) \left( \nabla \phi \right)^2 \frac{\partial W(n)}{\partial \phi} \right)$$

and

$$\frac{\partial}{\partial y} \left( \nabla \phi \right)^2 W(n) \frac{\partial W(n)}{\partial \phi} = \frac{\partial}{\partial y} \left( W(n) \left( \nabla \phi \right)^2 \frac{\partial W(n)}{\partial \phi} \right)$$

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

$$W(n) = W_0(1 - 3\epsilon_s) \left[ 1 + \frac{4\epsilon_s}{1 - 3\epsilon_s} \left( \frac{\partial \phi_i}{\partial \phi_j} \right)^2 + \left( \frac{\partial \phi_j}{\partial \phi_i} \right)^2 \right]$$

(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 from an existing mesh, the new nodal metrics field \(M_i\), we provide to the mesher is

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

(5)

\(q\) being the space dimension, \(\Gamma\) \((i)\) being the set of nodes connected to node \(i\), \(s_{ij}\) the stretching factor applied to obtain the new edge size. The edge stretching factor, \(s_{ij}\), is obtained from the \textit{a posteriori} estimated error, and is given by
\[ s_{ij} = \left( \frac{L}{e_{ij}} \right)^{1/p} \]  

(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.

\[ L = \left( \sum_{i \in G(i)} \sum_{j \in G(i)} \frac{e_{ij}^{p+2}}{A} \right)^{\frac{p+2}{p}} \]  

(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, \( e_{ij} = \max(|\nabla U \cdot X_{ij}|; e_{\text{min}}|X_{ij}|) \) and \( \nabla U = (X_i)^{\dagger} U_i \), where \( X_i = \frac{1}{|G(i)|} \sum_{j \in G(i)} X_{ij} \otimes X_{ij} \) is the distribution function and \( U_i = \sum_{j \in G(i)} U_j X_{ij} \). \( e_{\text{min}} \) 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).](image)

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 \( \theta \) and the function \( \phi \) 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 \( v_{\text{max}} \), over the entire domain as:

\[ v = -\frac{\theta}{\beta} - d_0 K \quad \text{with} \quad \beta = \frac{a_1 r}{\lambda W(n)} , \quad d_0 = \frac{a_2 W(n)}{\lambda} , \quad \text{the curvature} \; K = \nabla \cdot n \]

We compute the displacement \( v_{\text{max}} \Delta t \) 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 \( \Delta t \) as:

\[ \Delta t = \frac{h_{\text{min}}}{(10 \cdot v_{\text{max}})} \]  

(8)

\( h_{\text{min}} \) 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:

$$V_{\text{interface}} = \int_{-\infty}^{\infty} \delta_\phi \, dV$$

with

$$\delta_\phi = \begin{cases} 
0 & \text{for } \phi < -E \\
\frac{1}{2E} + \frac{1}{2E} \cos \left( \frac{\pi \phi}{E} \right) & \text{for } -E < \phi < E \\
0 & \text{for } \phi > E
\end{cases}$$

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

$$N_1 = \frac{S_{\text{interface}}}{S_{\text{element}}} = 2 \cdot \frac{S_{\text{interface}}}{h_{\text{min}}^2} \quad \text{in 2D}$$

$$N_1 = \frac{V_{\text{interface}}}{V_{\text{element}}} = 6 \cdot \frac{V_{\text{interface}}}{h_{\text{min}}^3} \quad \text{in 3D}$$

Finally we compute the total number of element needed as $N_E = N_1 + N_2$ 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


Let us consider a rectangular domain $[0;1000] \times [0;300]$ 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 $W(\mathbf{n}) = W_0 (1 - 3 \epsilon_4) \left[ 1 + \frac{4\epsilon_4}{1 - 3\epsilon_4} \left( \epsilon_\phi \phi \right)^4 + \left( \epsilon_\phi \phi \right)^4 \right]$ with $\epsilon_4 = 0.05$. Other simulation parameters are: $\alpha = 1$, $\tau_0 = 1$, $\Delta t = h_{\text{min}} / (10 \cdot V_{\text{max}})$. In the following, we compare the solution for different values of the minimal mesh size $h_{\text{min}}$.

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](image)

**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

<table>
<thead>
<tr>
<th>Case</th>
<th>$V_{Tip}$</th>
<th>$NE$</th>
<th>$h_{min}$</th>
<th>Nb_Proc</th>
<th>Fr</th>
<th>Adapt Time</th>
<th>Adapt $NE$</th>
<th>$T_{CPU}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0469</td>
<td>4638 842</td>
<td>0.4</td>
<td>1</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>36 000h*</td>
</tr>
<tr>
<td>2</td>
<td>0.0469</td>
<td>4638 842</td>
<td>0.4</td>
<td>4</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>2 400h*</td>
</tr>
<tr>
<td>3</td>
<td>0.0469</td>
<td>4638 842</td>
<td>0.4</td>
<td>16</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>1 240h*</td>
</tr>
<tr>
<td>4</td>
<td>0.0485</td>
<td>100 000</td>
<td>0.4</td>
<td>16</td>
<td>1</td>
<td>No</td>
<td>No</td>
<td>504h</td>
</tr>
<tr>
<td>5</td>
<td>0.051</td>
<td>80 000</td>
<td>0.4</td>
<td>16</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>80h</td>
</tr>
<tr>
<td>6</td>
<td>0.05</td>
<td>80 000</td>
<td>0.4</td>
<td>16</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>13h</td>
</tr>
<tr>
<td>7</td>
<td>0.0469</td>
<td>46 000</td>
<td>0.4</td>
<td>16</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes($N_2 = 250000$)</td>
<td>10h</td>
</tr>
</tbody>
</table>

* 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

<table>
<thead>
<tr>
<th>$\Delta$</th>
<th>$\varepsilon_4$</th>
<th>$\alpha$</th>
<th>$d_0/W_0$</th>
<th>$V_{Tip}$</th>
<th>$V_{Tip}^{GF}$</th>
<th>% error</th>
<th>Domain</th>
<th>$NE$</th>
<th>$T_{CPU}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.65</td>
<td>0.05</td>
<td>1</td>
<td>0.554</td>
<td>0.0469</td>
<td>0.0469</td>
<td>0</td>
<td>[1000;300]</td>
<td>46000</td>
<td>10h</td>
</tr>
<tr>
<td>0.55</td>
<td>0.05</td>
<td>2</td>
<td>0.277</td>
<td>0.017</td>
<td>0.017</td>
<td>0</td>
<td>[1000;300]</td>
<td>44000</td>
<td>6h 30min</td>
</tr>
<tr>
<td>0.55</td>
<td>0.05</td>
<td>4</td>
<td>0.139</td>
<td>0.017</td>
<td>0.017</td>
<td>0</td>
<td>[500;150]</td>
<td>34000</td>
<td>2h 30 min</td>
</tr>
<tr>
<td>0.45</td>
<td>0.05</td>
<td>4</td>
<td>0.139</td>
<td>0.00545</td>
<td>0.00545</td>
<td>0</td>
<td>[1000;300]</td>
<td>50000</td>
<td>5h 50 min</td>
</tr>
<tr>
<td>0.55</td>
<td>0.02</td>
<td>2</td>
<td>0.277</td>
<td>0.00685</td>
<td>0.00685</td>
<td>0</td>
<td>[1000;300]</td>
<td>30000</td>
<td>7h 30 min</td>
</tr>
</tbody>
</table>

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.](image)

References