DigitalSnow - ANR-11-BS02-009
Deliverable 4

Discrete-Continuous approach for deformable partitions

Élie Bretin
Institut Camille Jordan - INSA Lyon
elie.bretin@insa-lyon.fr

Roland Denis
LAMA - Université Savoie Mont Blanc
roland.denis@univ-smb.fr

Frédéric Flin
CEN - Météo-France & CNRS
frederic.flin@meteo.fr

Jacques-Olivier Lachaud
LAMA - Université Savoie Mont Blanc
jacques-olivier.lachaud@univ-smb.fr

Édouard Oudet
LJK - Université de Grenoble
edouard.oudet@imag.fr

Tristan Roussillon
LIRIS - INSA Lyon
tristan.roussillon@insa-lyon.fr

November 6, 2015

Contents

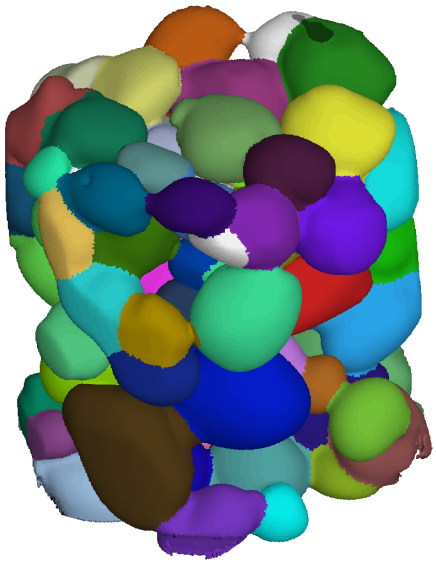


Figure 1: Visualization of the crystal orientation of a snow sample. Each color corresponds to a particular crystal orientation. These data were acquired at the ESRF ID19 beamline under the SNOW-WHITE project (ANR-06-BLAN-0396).

1 Introduction

As explained in the first DigitalSnow deliverable about the modeling of dry snow metamorphism, dry snow accumulated on the ground is a complex porous medium constituted of air, water vapor and ice. The evolution of the interfaces between these phases, due to water condensation and sublimation (thermodynamical effect), and due to the structure rearrangement (mechanical effect), is known as the snow metamorphism.

In fact, as illustrated in figure 1, the ice phase is constituted of multiple ice crystals with many different orientations. Taking into account this polycrystalline microstructure is important to properly model the anisotropic behavior of the snow metamorphism, especially under temperature gradient conditions.

Therefore, it is important to model dry snow as multiple ice grains with their own dynamics, and interacting with each others. As main physical phenomena occur at the interfaces between the grains and the air (condensation and sublimation) and since the geometrical properties of those surfaces impact directly the model (e.g. the curvature modifies the saturated vapor pressure, by Kelvin's law), an adapted surface representation is needed.

A common choice is to use an interface capturing method, like the level-set

methods or the phase-field method. However, those methods basically need one field for each ice grain. Since we have to deal with hundreds of grains on a three-dimensional space discretized with up to 1000^3 points, it implies a very large memory consumption and CPU usage. Several algorithms have been developed so far, for level-set (see e.g. [8]) and phase-field methods (see e.g. [5], [10] and [9]), in order to deal with those issues.

In this deliverable, we will focus on the phase-field description of a simplified multi-grain growth model based on the Allen-Cahn equation, associated with some of those algorithms and a computationally efficient implementation.

This framework is meant to be integrated as a package in the DGtal library[1].

2 Grain growth velocity

In a first step, let's recall the Hertz-Knudsen equation relating the growth velocity to the supersaturation of the vapor:

$$\rho_{\text{ice}} v_n = \alpha \sqrt{\frac{kT}{2\pi m}} \left(\rho_{\text{v}} - \rho_{\text{vs}}(T, K)\right) \quad \text{on } \Gamma, \tag{1}$$

where ρ_{ice} is the density of the ice, v_n the growth velocity normal to the ice/air interface Γ, oriented toward the air, α the condensation coefficient, k the Boltzmann's constant, T the temperature at the interface Γ, m the mass of a water molecule, ρ_{v} the vapor density, and $\rho_{\text{vs}}(T, K)$ is the saturation density that depends on the temperature and the interface mean-curvature K.

This saturation density can be approximated by:

$$\rho_{\text{vs}}(T, K) \approx \rho_{\text{vs}}(T)(1 + d_0 K), \tag{2}$$

where $\rho_{\text{vs}}(T)$ is the reference saturation density when the mean-curvature is null, and where d_0 is the capillary length.

In the following, we will denote as C the main factor in (1):

$$C := \alpha \sqrt{\frac{kT}{2\pi m}}. \tag{3}$$

We will now consider that the supersaturation term $(\rho_{\text{v}} - \rho_{\text{vs}}(T, K))$ depends only on curvature effect, e.g. that $\rho_{\text{v}} \equiv \rho_{\text{vs}}(T)$, and we get the following normal velocity expression:

$$v_n = -C \frac{\rho_{\text{vs}}(T)}{\rho_{\text{ice}}} d_0 K. \tag{4}$$

[1] http://dgtal.org/

Since this mean-curvature dynamics is a very simplified model of the crystal growth (see equations (15) to (21) of the first DigitalSnow deliverable), it does not guarantee that the ice volume stays constant.

3 The phase-field point of view

In the phase-field method, the interface Γ is the level-set of value $\frac{1}{2}$ of a phase function u. More precisely, if the interface is smooth enough, the phase-field function is defined from the distance function d to the interface Γ, with an interface sharpness parameter ε:

$$u(x,t) := q\left(\frac{d(x,t)}{\varepsilon}\right) \tag{5}$$

with the profile function q:

$$q(s) := \frac{1}{2}\left(1 - \tanh\left(\frac{s}{2}\right)\right). \tag{6}$$

This last function is the solution of a minimization problem associated to a double-well potential W:

$$W(s) := \frac{1}{2}s^2\left(1-s\right)^2. \tag{7}$$

For more details about this construction, please refer, for example, to [1].

3.1 Derivation of the model

From this framework, we get the following useful relations:

$$q^{'}(s) = -\sqrt{2W\left(q(s)\right)}, \tag{8}$$

$$q^{''}(s) = W^{'}\left(q(s)\right). \tag{9}$$

Other relations arise from the derivation of u:

$$\frac{\partial u}{\partial t}(x,t) = \frac{1}{\varepsilon}q^{'}\left(\frac{d}{\varepsilon}\right)\frac{\partial d}{\partial t}(x,t), \tag{10}$$

$$\nabla u(x,t) = \frac{1}{\varepsilon}q^{'}\left(\frac{d}{\varepsilon}\right)\nabla d(x,t), \tag{11}$$

$$\Delta u(x,t) = \frac{1}{\varepsilon^2}q^{''}\left(\frac{d}{\varepsilon}\right) + \frac{1}{\varepsilon}q^{'}\left(\frac{d}{\varepsilon}\right)\Delta d(x,t). \tag{12}$$

Since the interface normal speed is directly linked to the distance function by:

$$\frac{\partial d}{\partial t}(x,t) = -v_n \tag{13}$$

and noting that the mean curvature of a surface is equal to the laplacian of the distance function:

$$\Delta d(x) = K \quad \text{on } \Gamma, \tag{14}$$

we obtain the following phase-field equation related to the interface normal speed (4):

$$\frac{\partial u}{\partial t}(x,t) = d_0 C \frac{\rho_{\text{vs}}(T)}{\rho_{\text{ice}}} \left[\Delta u(x,t) - \frac{1}{\varepsilon^2} W^{'}(u) \right] \quad \text{on } \Omega. \tag{15}$$

In the following, we will work at constant temperature. We can therefore omit the $d_0 C \frac{\rho_{\text{vs}}(T)}{\rho_{\text{ice}}}$ coefficient that acts as a time-scale factor. The canonical form of this equation is thus:

$$\frac{\partial u}{\partial t}(x,t) = \Delta u(x,t) - \frac{1}{\varepsilon^2} W^{'}(u) \quad \text{on } \Omega \tag{16}$$

kwown as the Allen-Cahn equation.

3.2 Extension to multi-grain case

If we consider N ice grains, we basically duplicate the previous equation (15) for each grain:

$$\frac{\partial u_i}{\partial t}(x,t) = \Delta u_i(x,t) - \frac{1}{\varepsilon^2} W^{'}(u_i(x,t)) \quad \text{on } \Omega, \text{ for } i = 1, \ldots, N. \tag{17}$$

and the border Γ_i of the i-th grain is defined as follows from his phase function u_i:

$$\Gamma_i(t) := \left\{ x \ ; \ u_i(x,t) = \frac{1}{2} \right\}. \tag{18}$$

3.2.1 Non-overlapping condition

However, this set of equations doesn't guarantee that the grains do not overlap. A common solution is to add a Lagrange multiplier $\lambda(x,t)$ associated to a non-overlapping condition $\sum_{i=1}^{N} u_i(x,t) = 1$, $\forall x \in \Omega$. We obtain:

$$\frac{\partial u_i}{\partial t}(x,t) = \Delta u_i(x,t) - \frac{1}{\varepsilon^2} W^{'}(u_i(x,t)) + \lambda(x,t) \quad \forall (x,t) \in \Omega \times (0,\infty), \tag{19}$$

$$\sum_{i=1}^{N} u_i(x,t) = 1 \quad \forall (x,t) \in \Omega \times [0,\infty). \tag{20}$$

The non-overlapping constraint is equivalent to:

$$\sum_{i=1}^{N} u_i(x,0) = 1 \quad \text{and} \quad \sum_{i=1}^{N} \frac{\partial u_i}{\partial t}(x,t) = 0 \quad \forall (x,t) \in \Omega \times (0,\infty). \tag{21}$$

Using this into the Allen-Cahn equation, we get

$$\sum_{i=1}^{N} \Delta u_i(x,t) - \frac{1}{\varepsilon^2} \sum_{i=1}^{N} W'(u_i(x,t)) + N\lambda(x,t) = 0. \tag{22}$$

Since the laplacian operator is linear and the sum of u_i is equal to 1, the Lagrange multiplier expresses as:

$$\lambda(x,t) = \frac{1}{N\varepsilon^2} \sum_{i=1}^{N} W'(u_i(x,t)) \tag{23}$$

and we finally obtain the grain growth equations:

$$\frac{\partial u_i}{\partial t}(x,t) = \Delta u_i(x,t) - \frac{1}{\varepsilon^2} W'(u_i(x,t)) + \frac{1}{N\varepsilon^2} \sum_{i=1}^{N} W'(u_i(x,t)) \tag{24}$$

with initial condition

$$\sum_{i=1}^{N} u_i(x,0) = 1 \quad \forall x \in \Omega. \tag{25}$$

3.2.2 Volume conservation

As previously noted, this mean-curvature movement does not conserve the grain's volume. For that purpose, we also add a Lagrange multiplier $\mu_i(t)$ for each grain, that acts as a forcing term in the Allen-Cahn equation:

$$\frac{\partial u_i}{\partial t}(x,t) = \Delta u_i(x,t) - \frac{1}{\varepsilon^2} W'(u_i(x,t)) + \mu_i(t) \frac{1}{\varepsilon} \sqrt{2W\left(u_i(x,t)\right)}, \tag{26}$$

$$\frac{\partial}{\partial t} \int_\Omega u_i(x,t)\,\mathrm{d}x = 0 \quad \forall i = 1, \ldots, N \tag{27}$$

where the volume of the i-th grain is approximated by the integral of the i-th phase-field. This approximation is justified by the proposal 12 of [1]:

$$|\Omega_i| = \int_\Omega u_i \,\mathrm{d}x + \mathcal{O}(\varepsilon^2) \tag{28}$$

where $\Omega_i = \{x \in \Omega \ ; \ u_i(x) \leq \frac{1}{2}\}$.

The Lagrange multiplier is here weighted by $\sqrt{2W(u_i(x,t))}$ to keep a good behavior of the phase-field function (see chapter 3 of [1] for more details).

As in the previous section, we put the constraint into the evolution equation:

$$\int_\Omega \Delta u_i(x,t)\,\mathrm{d}x - \frac{1}{\varepsilon^2} \int_\Omega W'\left(u_i(x,t)\right)\mathrm{d}x + \mu_i(t) \frac{1}{\varepsilon} \int_\Omega \sqrt{2W\left(u_i(x,t)\right)}\,\mathrm{d}x = 0. \tag{29}$$

Under common assumptions, i.e. if Ω is periodic or if u_i observes a Neumann boundary condition (that is consistent with volume conservation), the laplacian term is null due to the Stokes theorem. Thus, the Lagrange multiplier expresses as:

$$\mu_i(t) = \frac{1}{\varepsilon} \frac{\int_\Omega W'\left(u_i(x,t)\right) \mathrm{d}x}{\int_\Omega \sqrt{2W\left(u_i(x,t)\right)}\, \mathrm{d}x} \tag{30}$$

and we finally obtain the following evolution equation:

$$\frac{\partial u_i}{\partial t}(x,t) = \Delta u_i(x,t) - \frac{1}{\varepsilon^2}\left[W'(u_i(x,t)) + \varepsilon \frac{\int_\Omega W'\left(u_i(x,t)\right) \mathrm{d}x}{\int_\Omega \sqrt{2W\left(u_i(x,t)\right)}\, \mathrm{d}x} \sqrt{2W\left(u_i(x,t)\right)} \right]. \tag{31}$$

3.2.3 Mixing the two constraints

As we want to satisfy both constraints, we write:

$$\frac{\partial u_i}{\partial t}(x,t) = \Delta u_i(x,t) - \frac{1}{\varepsilon^2} W'(u_i(x,t)) + \mu_i(t) \frac{1}{\varepsilon} \sqrt{2W\left(u_i(x,t)\right)} + \lambda(x,t), \tag{32}$$

$$\sum_{i=1}^{N} u_i(x,t) = 1 \quad \forall x \in \Omega, \tag{33}$$

$$\frac{\partial}{\partial t} \int_\Omega u_i(x,t)\, \mathrm{d}x = 0 \quad \forall i = 1, \ldots, N. \tag{34}$$

Using the second constraint gives us:

$$\mu_i(t) = \frac{\frac{1}{\varepsilon^2} \int_\Omega W'\left(u_i(x,t)\right) \mathrm{d}x - \int_\Omega \lambda(x,t)\, \mathrm{d}x}{\frac{1}{\varepsilon} \int_\Omega \sqrt{2W\left(u_i(x,t)\right)}\, \mathrm{d}x}. \tag{35}$$

We remark that λ is spatially defined up to a constant, and thus we can set:

$$\int_\Omega \lambda(x,t)\, \mathrm{d}x \equiv 0. \tag{36}$$

Now using the first constraint, we get:

$$\lambda(x,t) = \frac{1}{\varepsilon^2 N} \sum_{i=1}^{N} W'\left(u_i(x,t)\right) - \frac{1}{\varepsilon} \sum_{i=1}^{N} \mu_i(t) \sqrt{2W\left(u_i(x,t)\right)}, \tag{37}$$

$$= \frac{1}{\varepsilon^2 N} \sum_{i=1}^{N} \left[W'\left(u_i(x,t)\right) - \frac{\int_\Omega W'\left(u_i(x,t)\right) \mathrm{d}x}{\int_\Omega \sqrt{2W\left(u_i(x,t)\right)}\, \mathrm{d}x} \sqrt{2W\left(u_i(x,t)\right)} \right] \tag{38}$$

that enable us to directly numerically solve equation (32).

We now have the set of equations needed to simulate a simplified dry snow metamorphism. Let's see how to efficiently implement it in the next section.

4 Numerical implementation

4.1 Reference scheme

We suppose that the domain Ω is equal to the cartesian product of periodic fields of $\mathbb{R}$:

$$\Omega = \bigtimes_{k=1}^{n} \mathbb{R}/L_k\mathbb{Z}, \tag{39}$$

We propose to use a Lie splitting of the Allen-Cahn equation in order to exactly solve the diffusion part by Fourier transform and we use an explicit Euler time-integration scheme for the reaction part (including the two Lagrange multipliers).

Thus, given a time step δt, we obtain the following algorithm 1 where FFT and IFFT stand for forward and backward Fast Fourier Transform algorithm, and where k denotes the coordinates in the frequency domain.

Algorithm 1: Reference scheme to solve Allen-Cahn equation

Data: $u^0_{i\in\{1,\dots,N\}}$, δt and ε

foreach $n \geq 0$ **do**

  // Solve the diffusion part

  for $i \leftarrow 1$ **to** N **do**

    $\hat{u}_i^n \leftarrow \mathrm{FFT}(u_i^n)$;

    $\hat{u}_i^{n+1/2}[k] \leftarrow \exp\left(-4\pi^2|k|^2\delta t\right)\hat{u}_i^n[k]$;

    $u_i^{n+1/2} \leftarrow \mathrm{IFFT}\left(\hat{u}_i^{n+1/2}\right)$;

  end

  // Pre-calculate recurrent terms

  for $i \leftarrow 1$ **to** N **do**

    $\mathrm{Int}_{W',i} \leftarrow \displaystyle\int_\Omega W'\left(u_i^{n+1/2}(x)\right)\mathrm{d}x$;

    $\mathrm{Int}_{\sqrt{2W},i} \leftarrow \displaystyle\int_\Omega \sqrt{2W\left(u_i^{n+1/2}(x)\right)}\,\mathrm{d}x$;

  end

  // Calculate the Lagrangian multiplier associated to the non-overlapping constraint

  $\lambda(x) \leftarrow \dfrac{1}{\varepsilon^2 N}\displaystyle\sum_{i=1}^{N}\left[W'\left(u_i^{n+1/2}(x)\right) - \frac{\mathrm{Int}_{W',i}}{\mathrm{Int}_{\sqrt{2W},i}}\sqrt{2W\left(u_i^{n+1/2}(x)\right)}\right]$;

  // Calculate the Lagrangian multiplier associated to the volume conservation

  for $i \leftarrow 1$ **to** N **do**

    $\mu_i \leftarrow \dfrac{1}{\varepsilon}\dfrac{\mathrm{Int}_{W',i}}{\mathrm{Int}_{\sqrt{2W},i}}$;

  end

  // Solve the reaction part

  for $i \leftarrow 1$ **to** N **do**

    $u_i^{n+1} \leftarrow u_i^{n+1/2} + \delta t\left[-\frac{1}{\varepsilon^2}W'\left(u_i^{n+1/2}\right) + \frac{1}{\varepsilon}\mu_i\sqrt{2W\left(u_i^{n+1/2}\right)} + \lambda\right]$;

  end

end

This numerical scheme is stable under the condition:

$$\delta t \leq \frac{\varepsilon^2}{2}. \tag{40}$$

The periodicity of the domain is necessary to use the Fast Fourier Transformation in order to exactly solve the diffusion part. Nevertheless, there exists some methods to use other boundaries conditions while using the FFT.

4.2 Remarks on the computational complexity

A typical set of initial data, like those obtained by tomography of the dry snow (see e.g. chapter 7 of [2]), involves hundreds of ice grains on a three-dimensional spatial domain discretized with 1000^3 points.

In that case, storing one phase field u_i^n in double precision needs 8 GiB of memory. Thus, simulating the metamorphism of more than a few ice grains becomes nearly impossible on an average computation server.

As pointed out by many previous works (see e.g. [5], [10] and [9]), the Allen-Cahn equation has a "less effect" far away of the grain interface Γ. More precisely, the phase-field u tends to 1 (interior) or 0 (exterior) far from the interface and a majority of terms in the equation (32) are weighted by $W(u_i)$ and $W^{'}(u_i)$ where W and $W^{'}$ are null at 0 and 1. The diffusion term tends also to 0 far from the interface.

Therefore, the time derivative of u_i tends to 0 in the same conditions. It appears reasonable to save memory and computational time in those areas.

4.3 Zero approximation

Let $\tau \geq 0$ be a positive tolerance, and introduce the approximation function ξ_τ as:

$$\xi_\tau(x) = \begin{cases} 0 & \text{if } x < \tau, \\ x & otherwise. \end{cases} \tag{41}$$

The following algorithm 2 adapts the algorithm 1 to take advantage of this approximation. The modified parts are highlighted in red.

Algorithm 2: Scheme for Allen-Cahn equation with zero approximation

Data: $u^0_{i\in\{1,\dots,N\}}$, δt and ε

// Initial zero-approximation

for $i \leftarrow 1$ **to** N **do**

  $u_i^0 \leftarrow \xi_\tau(u_i^0)$;

end

foreach $n \geq 0$ **do**

  // Solve the diffusion part

  for $i = 1$ **to** N **do**

    $\hat{u}_i^n = \mathrm{FFT}(u_i^n)$;

    $\hat{u}_i^{n+1/2}[k] = \exp\left(-4\pi^2|k|^2\delta t\right)\hat{u}_i^n[k]$;

    $u_i^{n+1/2} = \xi_\tau\left(\mathrm{IFFT}\left(\hat{u}_i^{n+1/2}\right)\right)$;

  end

  // Pre-calculate recurrent terms

  for $i \leftarrow 1$ **to** N **do**

    $\mathrm{Int}_{W',i} \leftarrow \int_\Omega W'\left(u_i^{n+1/2}(x)\right)\mathrm{d}x$;

    $\mathrm{Int}_{\sqrt{2W},i} \leftarrow \int_\Omega \sqrt{2W\left(u_i^{n+1/2}(x)\right)}\,\mathrm{d}x$;

  end

  // Calculate the Lagrangian multiplier associated to the non-overlapping constraint

  foreach x **do**

    $\lambda(x) \leftarrow 0$;

    for $i \leftarrow 1$ **to** N **do**

      if $u_i^{n+1/2}(x) > 0$ **then**

        $\lambda(x) \leftarrow \lambda(x) + \frac{1}{\varepsilon^2 N}\sum_{i=1}^{N}\left[W'\left(u_i^{n+1/2}(x)\right) - \frac{\mathrm{Int}_{W',i}}{\mathrm{Int}_{\sqrt{2W},i}}\sqrt{2W\left(u_i^{n+1/2}(x)\right)}\right]$;

      end

    end

  end

  // Calculate the Lagrangian multiplier associated to the volume conservation

  for $i \leftarrow 1$ **to** N **do**

    $\mu_i \leftarrow \frac{1}{\varepsilon}\frac{\mathrm{Int}_{W',i}}{\mathrm{Int}_{\sqrt{2W},i}}$;

  end

  // Solve the reaction part

  for $i \leftarrow 1$ **to** N **do**

    $u_i^{n+1} \leftarrow \xi_\tau\left(u_i^{n+1/2} + \delta t\left[-\frac{1}{\varepsilon^2}W'\left(u_i^{n+1/2}\right) + \frac{1}{\varepsilon}\mu_i\sqrt{2W\left(u_i^{n+1/2}\right)} + \lambda\right]\right)$;

  end

end

To illustrate the effects of this approximation, we use the algorithm 1 to simulate a problem of tiling space with a fixed number of parts of equal volume, known as the Kelvin's problem (see section 6.1 page 27 and [7] in the three dimensional space), on a 128^2 discretized space with ε equal to two space steps. The result, after reaching equilibrium, is shown on figure 2.

The first sub-figure, 2a, shows all the parts with artificial colors to see their borders. The sub-figure 2b illustrates, in gray level, the values of one phase-field u_i, where we barely see its value increasing around some points outside of its frontier. This effect is amplified when only the positive part of u_i is shown, with a positive cutoff at 0.05, on figure 2d. The negative part of u_i is illustred on figure 2c.

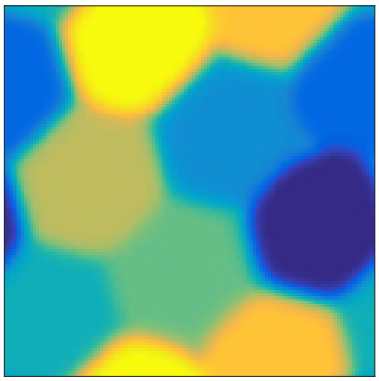
In case of a memory usage depending on a zero-approximation ξ_τ with tolerance $\tau = 10^{-4}$, the figure 2e shows which points would be stored. If we analyze the amount of significant values to be stored, there will be an average of ≈ 3.2 of those values per space point, with a standard deviation of ≈ 2.35.

These figures are good examples of the problems that arise when using a zero-approximation with the Allen-Cahn equation (32):

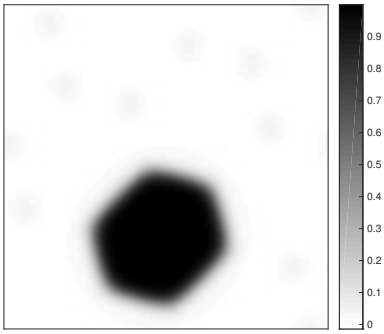
1. positive values appear far from the grain, especially at triple points. That leads to an inefficient memory usage when using zero-approximation.

2. u_i exhibits negatives values, around the contacts between two grains. That leads to significant volume loss when the negative part is lost during zero-approximation.

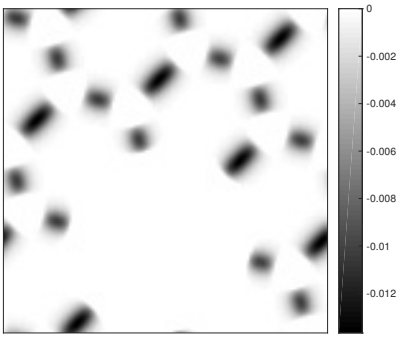
This is due to the fact that the non-overlapping Lagrange multiplier λ has the same effect for each phase-field, even far away from a grain.



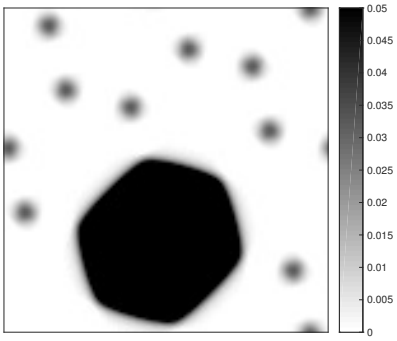
(a) Each phase-field with different color.



(b) One phase-field u_i.



(c) Negative part of one phase-field $\min(u_i, 0)$.



(d) Positive part of one phase-field with cutoff, $\min(\max(u_i, 0), 0.05)$.

(e) Non null values u_i with zero approximation at $\tau = 10^{-4}$.

Figure 2: Kelvin's problem in two-dimension (see 6.1, page 27) with phase-field behavior analysis.

4.4 New non-overlapping Lagrange multiplier

4.4.1 First formulation

To fix this, we propose to weight λ by the current phase-field value:

$$\begin{aligned}\frac{\partial u_i}{\partial t}(x,t) &= \Delta u_i(x,t) \\ &- \frac{1}{\varepsilon^2} W^{'}(u_i(x,t)) + \mu_i(t)\frac{1}{\varepsilon}\sqrt{2W\left(u_i(x,t)\right)} + u_i(x,t)\lambda(x,t),\end{aligned} \tag{42}$$

$$\sum_{i=1}^{N} u_i(x,t) = 1 \quad \forall x \in \Omega, \tag{43}$$

$$\frac{\partial}{\partial t}\int_\Omega u_i(x,t)\,\mathrm{d}x = 0 \quad \forall i = 1,\dots,N. \tag{44}$$

Unlike the previous Lagrange multiplier formulation, we will not be able to obtain a direct solution. Injecting the non-overlapping condition into the Allen-Cahn equation, we get:

$$\lambda(x,t) = \frac{1}{\sum_{j=1}^{N} u_j(x,t)}\left[\sum_{j=1}^{N} W^{'}\left(u_j(x,t)\right) - \sum_{j=1}^{N}\mu_j\sqrt{2W\left(u_j(x,t)\right)}\right]. \tag{45}$$

The volume conservation constraint gives us:

$$\int_\Omega W^{'}\left(u_i(x,t)\right)\mathrm{d}x - \mu_i\int_\Omega\sqrt{2W\left(u_i(x,t)\right)}\,\mathrm{d}x - \int_\Omega\lambda(x,t)u_i(x,t)\,\mathrm{d}x \quad \forall i = 1,\dots,N \tag{46}$$

and thus

$$\begin{aligned}&\mu_i\int_\Omega\sqrt{2W\left(u_i(x,t)\right)}\,\mathrm{d}x - \sum_{j=1}^{N}\mu_j\int_\Omega\frac{u_i(x,t)}{\sum_{k=1}^{N}u_k(x,t)}\sqrt{2W\left(u_j(x,t)\right)}\,\mathrm{d}x \\ &= \int_\Omega W^{'}\left(u_i(x,t)\right)\mathrm{d}x - \int_\Omega\frac{u_i(x,t)}{\sum_{k=1}^{N}u_k(x,t)}\sum_{k=1}^{N}W^{'}\left(u_k(x,t)\right)\mathrm{d}x \quad \forall i = 1,\dots,N.\end{aligned} \tag{47}$$

Solving this linear system gives us an expression for μ_i, and thus for λ.

4.4.2 Second formulation with given volume

Another formulation is possible in the case where we need to enforce that each phase-field has a given volume V_i. However, with this method, the Lagrange multipliers are applied in an ad-hoc step after having integrated the Allen-Cahn over one time step, more precisely, the method can be described as follows:

1. define $\tilde{u_i}^{n+1}$ as the result of the integration of

$$\frac{\partial u_i}{\partial t}(x,t) = \Delta u_i(x,t) - \frac{1}{\varepsilon^2} W^{'}(u_i(x,t)) \tag{48}$$

over one time-step δt with initial condition u_i^n.

2. find $\lambda(x)$ and $\mu_{i=1,\dots,N}$ so that

$$u_i^{n+1}(x) := \tilde{u_i}^{n+1}(x) + \mu_i \sqrt{2W\left(\tilde{u_i}^{n+1}(x)\right)} + \lambda(x)\tilde{u_i}^{n+1}(x) \tag{49}$$

verifies the following constraints:

$$\int_\Omega u_i^{n+1}(x)\,\mathrm{d}x = V_i \quad \forall i = 1,\dots,N, \tag{50}$$

$$\sum_{i=1}^{N} u_i^{n+1}(x) \equiv 1. \tag{51}$$

As previously, the second constraint implies that

$$\lambda(x) = -\frac{\sum_{k=1}^{N} \tilde{u_k}^{n+1} - 1 + \sum_{k=1}^{N} \mu_k \sqrt{2W\left(\tilde{u_i}^{n+1}(x)\right)}}{\sum_{k=1}^{N} \tilde{u_i}^{n+1}(x)} \tag{52}$$

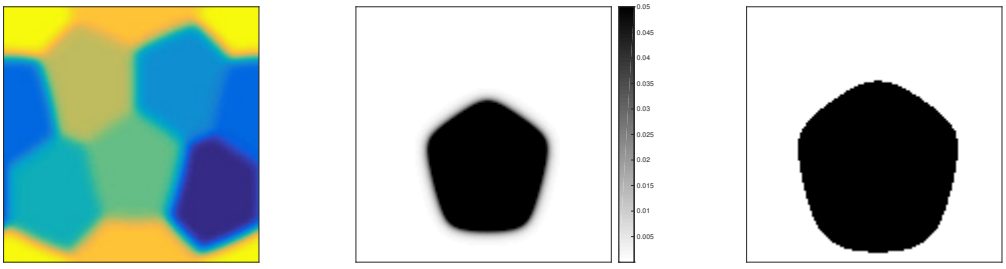
and, together with the first constraint, we obtain the following linear system:

$$\begin{aligned}\mu_i \int_\Omega \sqrt{2W\left(\tilde{u_i}^{n+1}(x)\right)}\,\mathrm{d}x - \sum_{j=1}^{N} \mu_j \int_\Omega \frac{\tilde{u_i}^{n+1}(x)}{\sum_{k=1}^{N} \tilde{u_k}^{n+1}(x)} \sqrt{2W\left(\tilde{u_j}^{n+1}(x)\right)}\,\mathrm{d}x \\ = V_i - \int_\Omega \tilde{u_i}^{n+1}(x)\,\mathrm{d}x + \int_\Omega \frac{\tilde{u_i}^{n+1}(x)}{\sum_{k=1}^{N} \tilde{u_k}^{n+1}(x)} \left(\sum_{k=1}^{N} \tilde{u_k}^{n+1} - 1\right).\end{aligned} \tag{53}$$

4.4.3 Results

Let us take a look at the behavior of the first formulation on the same problem as in the previous section. The figure 3 shows the result without applying zero-approximation. All the phase-field u_i have values in the interval $[0,1]$, and the non-overlapping constraint and volume conservation are respected with relative error of the working precision order.

The significant values are also located around the grain zone, and that leads to an average of ≈ 2.6 significant values per space point (for $\tau = 10^{-4}$) with



(a) Each phase-field with different color.

(b) Positive part of one phase-field with cutoff, $\min(\max(u_i, 0), 0.05)$.

(c) Non null values u_i with zero approximation at $\tau = 10^{-4}$.

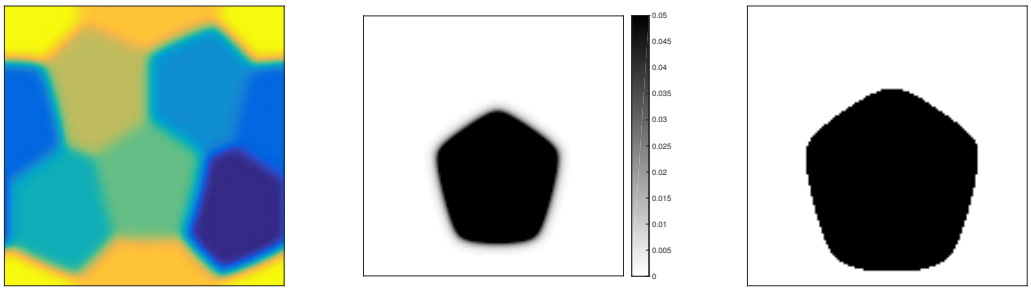
Figure 3: Kelvin's problem in two-dimension (see 6.1, page 27) with the first modified Lagrange multipliers formulation (42).

a standard deviation of ≈ 0.79, i.e. those values are more evenly spread on the domain.

When applying the algorithm 2 that applied zero-approximation during the simulation, with tolerance $\tau = 10^{-4}$, the constraints are fulfilled only to a relative order of 10^{-1}. With $\tau = 10^{-8}$, the maximum relative error is greatly reduced to 10^{-5} but at the price of memory usage. However, there is no visible difference, as illustrated by the figure 4.

This mitigated results can be explained by the fact that these Lagrange multipliers aim to maintain a zero variation of $\sum_{i=1}^{N} u_i(x)$ and $\int_\Omega u_i(x)\,\mathrm{d}x$ but doesn't take into account that the zero-approximation modified those values.

Conversely, the second formulation 4.4.2 guarantees, by construction, an exact fulfillment of the constraints even while applying the zero-approximation.



(a) Each phase-field with different color.

(b) Positive part of one phase-field with cutoff, $\min(\max(u_i, 0), 0.05)$.

(c) Non null values u_i with zero approximation at $\tau = 10^{-4}$.

Figure 4: Kelvin's problem in two-dimension (see 6.1, page 27) with the first modified Lagrange multipliers formulation (42) and using zero-approximation algorithm 2 with $\tau = 10^{-4}$.

4.5 Efficient storage

Now that we have reduced the amount of significant values, we must use an adapted storage. We have chosen to use a specific storage for each space point, that is basically a list of (index,value) couples.

4.5.1 Basic storage operations

A value storage for one space point, that takes into account the zero-approximation ξ_τ, would have basic read and write operations described in algorithms 3 and 4.

Algorithm 3: Read operation in a zero-approximated storage

Data: The storage for a point $x \in \Omega$.
Input: The grain index i.
Output: The phase-field value $u_i(x)$.

```
if u_i(x) is already stored then
    return the stored value u_i(x) ;
else
    return the default value 0 ;
end
```

Algorithm 4: Write operation in a zero-approximated storage

Data: The storage for a point $x \in \Omega$ and the zero-approximation tolerance τ.
Input: The grain index i and the new value $u_i(x)$.

```
if u_i(x) is already stored then
    if the new value is significant, i.e. ξ_τ(u_i(x)) > 0 then
        Update the stored value with the given u_i(x) ;
    else
        Remove the associated value from the storage ;
    end
else
    if the new value is significant, i.e. ξ_τ(u_i(x)) > 0 then
        Add the given u_i(x) to the storage ;
    else
        Do nothing ;
    end
end
```

4.5.2 LabelledMap class

Using a simple list coupling the grain index and the associated value would lead to a heavy use of dynamically allocated memory, thus implying a performance penalty due to the non-alignment of the data.

Therefore, we propose to use a structure of a fixed size that can be extended dynamically if needed. Such a structure already exists in DGtal and is named LabelledMap[2].

It depends on three main parameters, fixed at compile time:

- L is maximum number of data that can be stored (e.g. the number of grains in the domain),
- N is the number of data that the structure can store without needing to allocate extra-space,
- M is the number of data stored in each allocated extra-space.

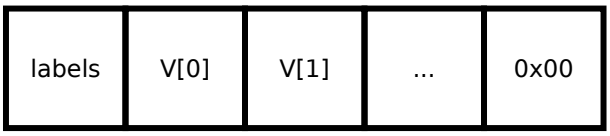
[2] see http://www.dgtal.org/doc/nightly/classDGtal_1_1LabelledMap.html.

As described on figure 5, the structure is composed of:

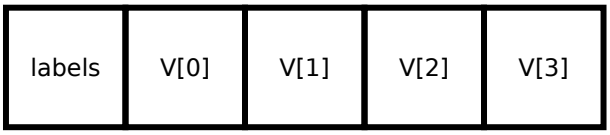
- a bit field of size greater than L bits (named labels in the schematic),
- an array of N values,
- a pointer that is either used for an extra value or, if more than $N + 1$ values need to be stored, as a pointer to the allocated extra-space.

Each allocated extra-space is composed of:

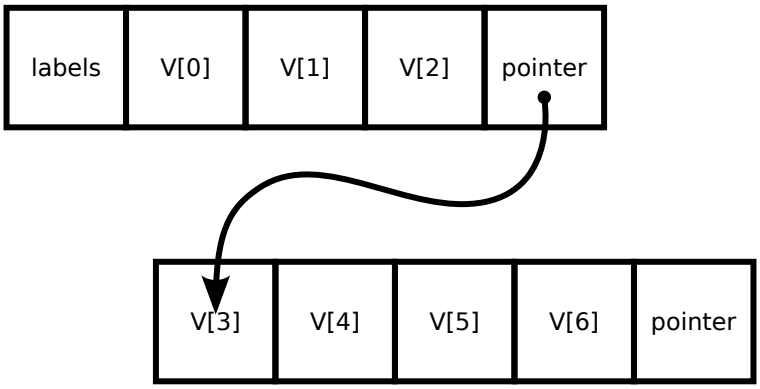
- an array of M values,
- a pointer that can point to an another allocated extra-space.



(a) When there is less than 4 data.



(b) When there is exactly 4 data.



(c) When there is more than 4 data.

Figure 5: Schematic of the structure of the LabelledMap class with parameters $N = 3$ and $M = 4$.

For example, on a 64 bits architecture, storing double precision values in a LabelledMap with parameters $L = 64$, $N = 3$ and $M = 3$ will use 40 bytes for the fixed size structure with extra-storage of size 32 bytes.

The bit field has the i-th bit set if the i-th grain has a stored value in the LabelledMap. The values are thus sorted by grain index. Therefore, the complexity of common operations are:

- constant for checking if the i-th grain has a stored value;
- linear with the number of stored values with index lower than i, when reading or updating the i-th value;
- linear with the total number of stored values when adding a new value.

In the case of storing values for each point of a discretized space, the fixed size of this structure allows us to allocate it by array and thus having a performance bonus due to memory alignment. However, the parameter N needs to be chosen accordingly to balance memory usage and heavy use of dynamic allocation. In addition, as this parameter is fixed at compile-time, it isn't easy to adapt it in real-time depending on the data properties.

4.6 Bounding boxes

In addition to this structure, we propose to use an axis aligned bounding box for each grain, that is based on the significant values (i.e. not approximated to zero) of the grain. All of the following features are not already implemented but those bounding box could be used:

1. to localize the Fast-Fourier Transformation on the bounding box, with an additional buffer zone around the bounding box in order to take into account the diffusion;
2. to localize the computation of finite-difference operators, taking into account the scheme stencil;
3. to accelerate reading of one specific phase-field, e.g. when integrating it.

This bounding box is composed, for each dimension of the space, of a list of values containing the number of stored values in each orthogonal hyper-plan along the corresponding axis. Thus, the bounding box is updated when one of these counters reaches zero or when it becomes not null.

The memory usage is of the order of the number of discretization points along each axis, and the computational penalty is also low (see the performance analysis below).

4.7 Other optimizations

The main focus when optimizing the code was about memory usage. For example, in the algorithms 1 and 2, caching the values of $W'\left(u_i^{n+1/2}(x)\right)$ and $\sqrt{2W\left(u_i^{n+1/2}(x)\right)}$ would use too much memory and it is not obvious that reading from the memory would take less time than recalculating those values.

5 Performance analysis

In order to illustrate the benefits of those strategies, we compare the performances of common operations with a reference implementation, that is using a vector of images. An image is a mapping between the discretization points and the values of a corresponding phase-field. All phase-field values are stored in double precision format.

The compared storage structures are:

- the reference structure, a STL vector that associates each grain to an instance of ImageContainerBySTLVector[3] (a common image model in DGtal library), denoted as a horizontal red line in the following figures;
- an image of LabelledMap with parameters $M = 5$ and $N = 1, \ldots, 4$, with no bounding boxes and no zero-approximation, denoted as "$\emptyset$" in the following figures;
- an image of LabelledMap with parameters $M = 5$ and $N = 1, \ldots, 4$, without bounding boxes and with zero-approximation at tolerance $\tau = 10^{-10}$, denoted as "$\tau = 10^{-10}$" in the following figures;
- an image of LabelledMap with parameters $M = 5$ and $N = 1, \ldots, 4$, without bounding boxes and with zero-approximation at tolerance $\tau = 10^{-4}$, denoted as "$\tau = 10^{-4}$" in the following figures;
- an image of LabelledMap with parameters $M = 5$ and $N = 1, \ldots, 4$, with bounding boxes but with no zero-approximation, denoted as "BB" in the following figures;
- an image of LabelledMap with parameters $M = 5$ and $N = 1, \ldots, 4$, with bounding boxes and zero-approximation at tolerance $\tau = 10^{-10}$, denoted as "BB & $\tau = 10^{-10}$" in the following figures;

[3] see http://dgtal.org/doc/nightly/classDGtal_1_1ImageContainerBySTLVector.html

- an image of LabelledMap with parameters $M = 5$ and $N = 1, \ldots, 4$, with bounding boxes and zero-approximation at tolerance $\tau = 10^{-4}$, denoted as "BB & $\tau = 10^{-4}$" in the following figures.

The benchmark is made on a two-dimensional periodic space $[0, 1]^2$, discretized with 1024^2 points and is composed of the following steps:

1. initialization with 64 phase-fields corresponding to 64 circular grains of radius $\frac{\sqrt{2}}{16}$, evenly spaced on a 8×8 grid, and with interface sharpness parameter $\varepsilon = \frac{2}{1024}$ (see first graph in figure 6). It tests write performance of the structures.

2. summing all phase-field values, firstly by summing the values at each space point and then summing the result over the discretized space (see second graph in figure 6). It tests read performance for point-wise operations.

3. summing all phase-field values, firstly by summing the values of each image separately (each one corresponding to a phase-field) and then summing the resulting values (see third graph in figure 6). It tests read performance when an operation needs to be done on each image separately.

4. summing the sinus of all phase-field values, firstly for each space point and then summing the result over the discretized space (see first graph in figure 7). It tests the performance when using a more complex operation, taking advantage of the fact $\sin(0) = 0$;

5. summing the sinus of all phase-field values, firstly for each image separately and then summing the resulting values (see second graph in figure 7).

6. summing the shifted sinus (i.e. $\sin(x + \pi)$) of all phase-field values, firstly for each space point and then summing the result over the discretized space (see first graph in figure 8). It tests the performance when using an even more complex operation (the implementation of sin function is commonly optimized for near-zero values), taking advantage of the fact that $\sin(\pi) = 0$;

7. summing the shifted sinus (i.e. $\sin(x + \pi)$) of all phase-field values, firstly for each image separately and then summing the resulting values (see second graph in figure 8).

The additional graphs in figure 9 show the total computational time for all of these benchmarks and the memory usage of each structure.

Using a zero-approximation with tolerance $\tau = 10^{-10}$ leads to an average of ≈ 3.20 non-approximated values per space point, and ≈ 2.11 for $\tau = 10^{-4}$.

As expected, the reference structure has the best performance when writing all values since it depends only on memory bandwidth. His read performance is better when summing by images because of the cache-friendly memory alignment in that case. However, this benefit over the point-wise summing becomes insignificant when the applied operation is more complex.

The LabbeledMap based structures have poor write performance but the speed increases when storing less values due to zero-approximation. Using bounding-box implies a slight performance penalty in that case.

For reading operations, summing firstly by points, the use of zero-approximation implies better performance that the reference structure, up to a factor 190 when computing the shifted sinus. However, without bounding boxes, the performances are worst than the reference structure for image-wise operations.

The localization of the computation allowed by the use of bounding boxes and zero-approximation allows to partially compensate this performance penalty. In fact, this strategy offers the best overall performances, except for writing operations.

In this benchmark, the choice of N slightly impacts the computational time but infers more significantly the memory usage, depending of the zero-approximation tolerance.

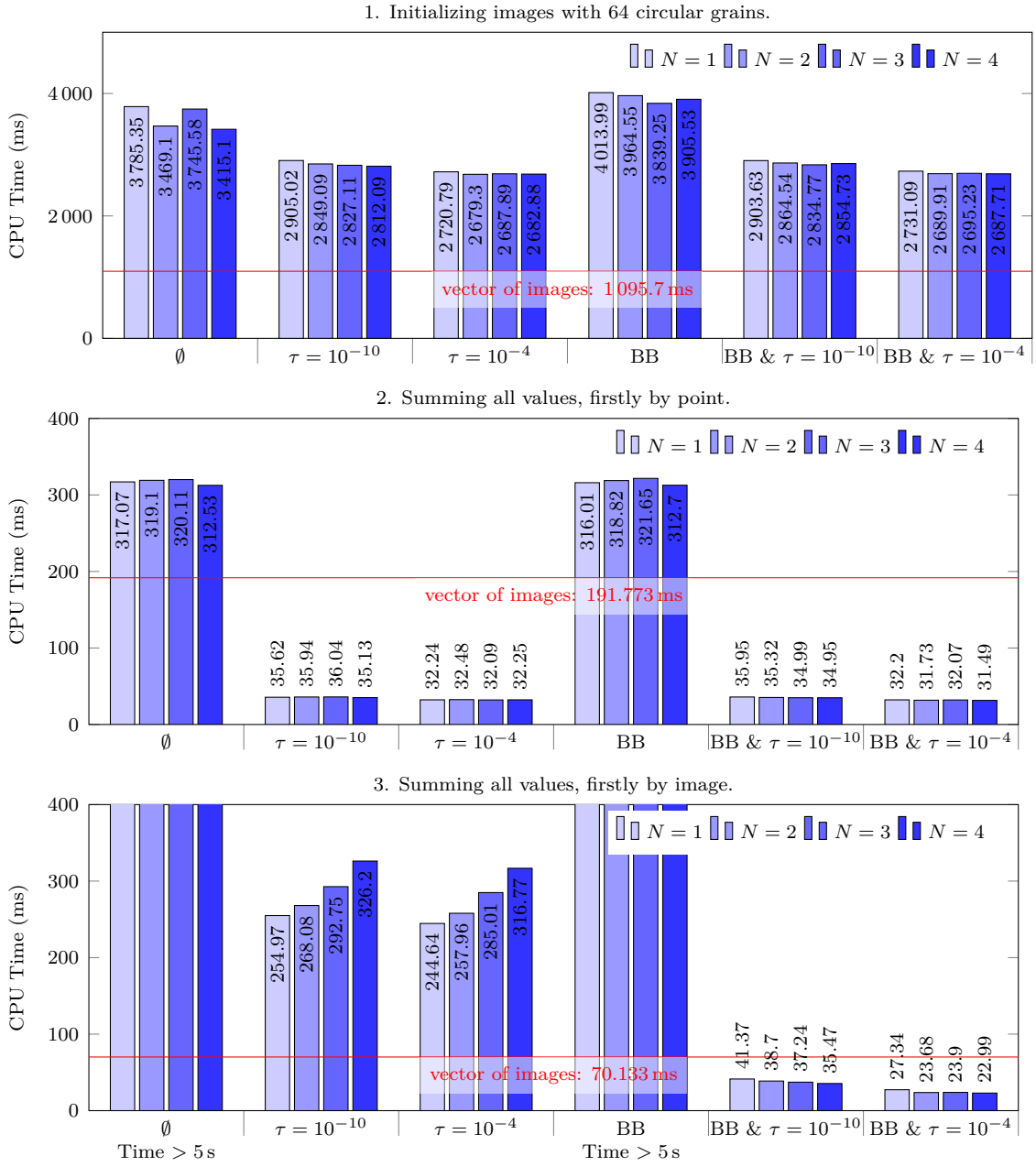


Figure 6: Performance comparison between using a list of images and an array of LabelledMap with different N values, different zero-approximation tolerances and with or without bounding box.

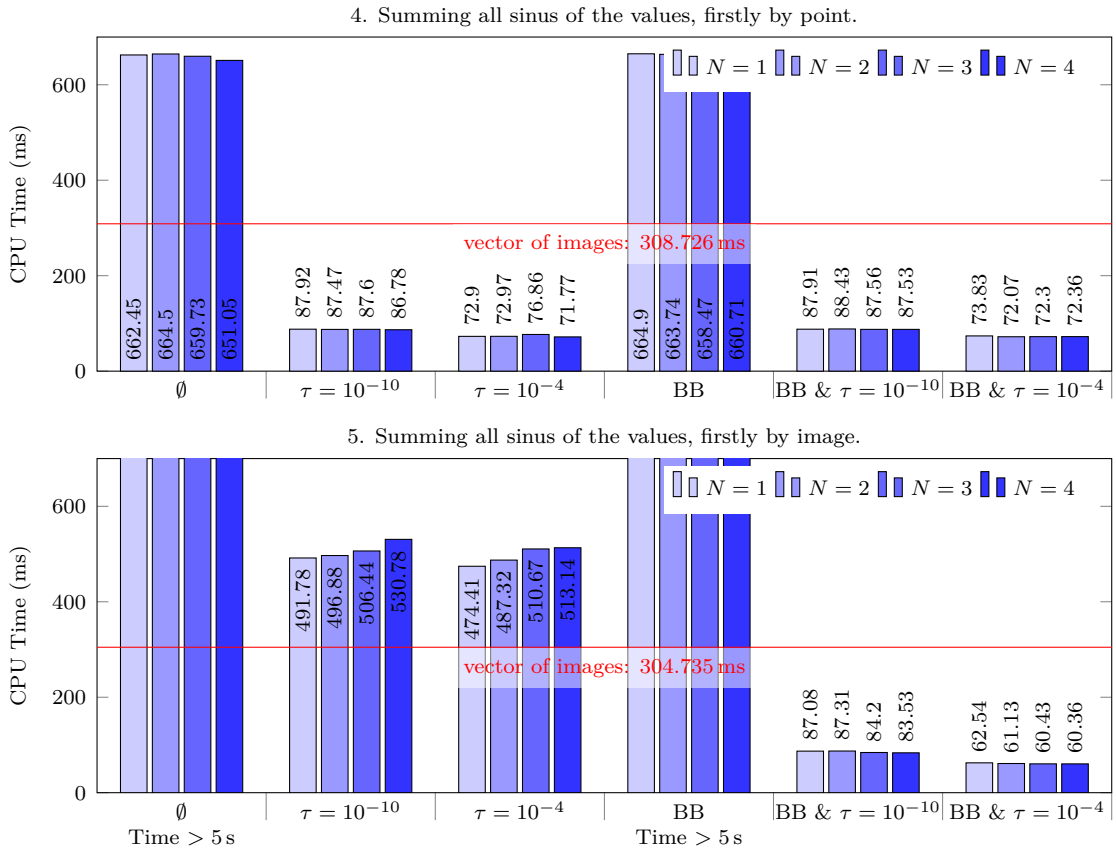


Figure 7: Performance comparison between using a list of images and an array of LabelledMap with different N values, different zero-approximation tolerances and with or without bounding box.

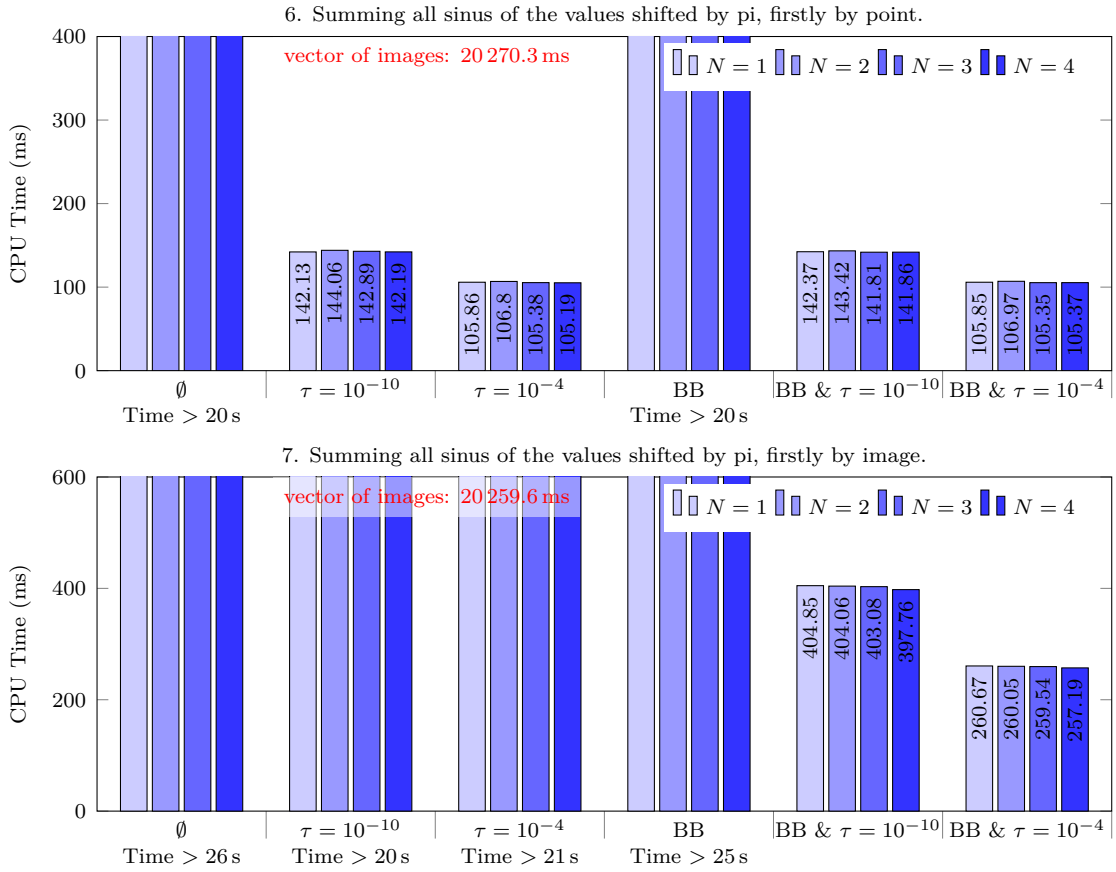


Figure 8: Performance comparison between using a list of images and an array of LabelledMap with different N values, different zero-approximation tolerances and with or without bounding box.

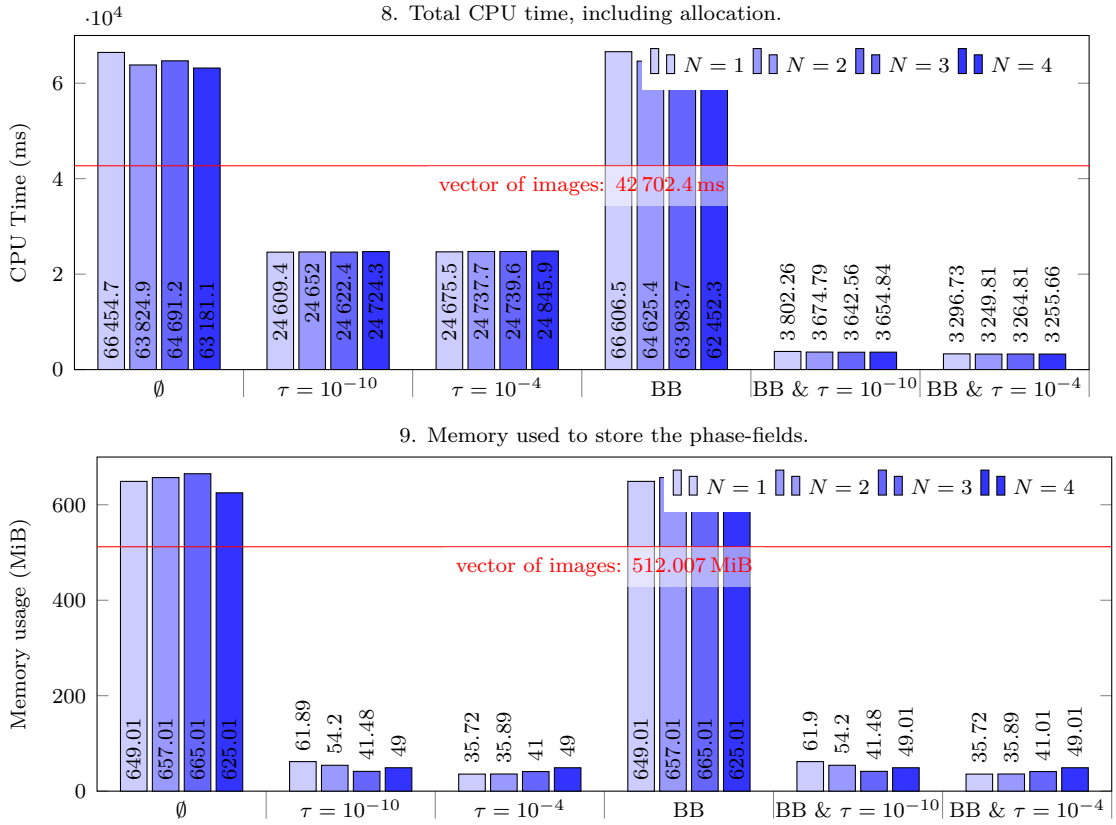


Figure 9: Performance comparison between using a list of images and an array of LabelledMap with different N values, different zero-approximation tolerances and with or without bounding box.

6 Some applications

We now illustrate some current applications of the implemented C++ code.

6.1 The Kelvin's problem

The Kelvin's problem is about filling the space with cells of equal volume such that the total surface area is minimal.

6.1.1 The 2D case: the honeycomb

In the two-dimensional case, this problem has already been solved in 1999 by Thomas C. Hales (see [6]) and the honeycomb is proved to be the optimal filling. The surface minimization problem behind this can be formulated in a volumetric manner using the Allen-Cahn equation (17) with non-overlapping condition and volume conservation. An another volumetric reformulation using phase-fields is described in [7].

To illustrate this, we use a random initial density of 128 phases on a periodic domain $[0, 1]^2$ (see figure 10), discretized with 512^2 points and we applied our framework, with the volume conservation formulation given in 4.4.2 (adapted to fix an equal volume for each phase), until an equilibrium is reached. We use an interface sharpness order of $\varepsilon = \frac{3}{512}$.

As this domain is square, we do not expect to obtain the honeycomb structure but it is a good example to analyze the performance evolution during the simulation.

The figures 10, 11 and 12 show the result at different steps. Each color is associated to the phase-field with the maximum value at the given point.

As we can see, the evolution is quite fast at the beginning and goes very slowly at the end. The equilibrium is still not reached at the 1000th step.

The figures 13 show how the computational time of each step and the memory usage, evolve during the simulation. At the beginning, as the phase-fields have significant values everywhere, the memory consumption is high and thus, the computational time too. In addition, updating values in the LabelledMap take more time due to the linear complexity.

During the simulation, as the cells become bigger, the mean number of significant values per point decreases, so as the memory usage and computational time.

This means that an adapted strategy must be used in order to avoid high initial memory consumption while keeping fast evolution of the simulation.

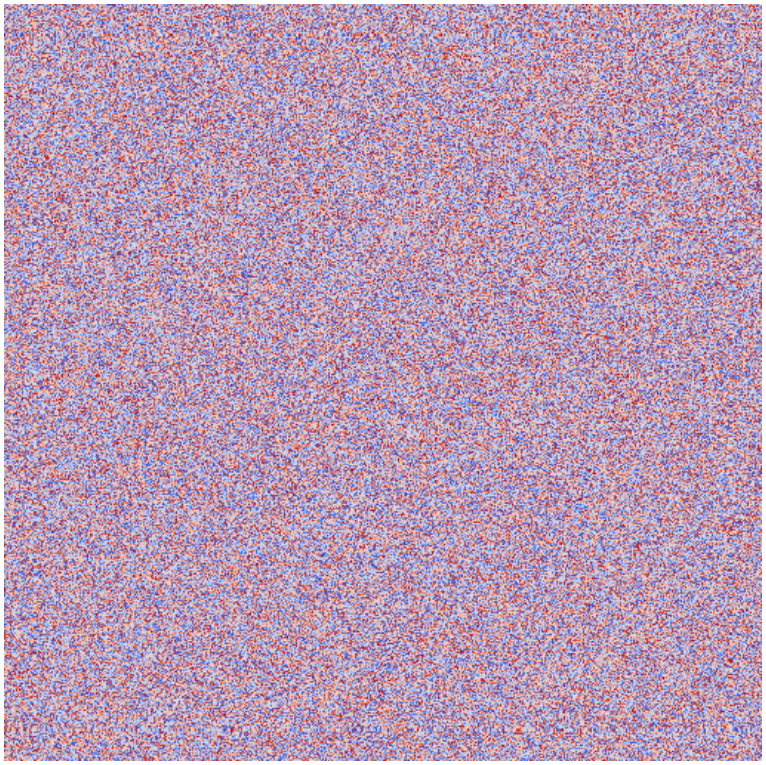
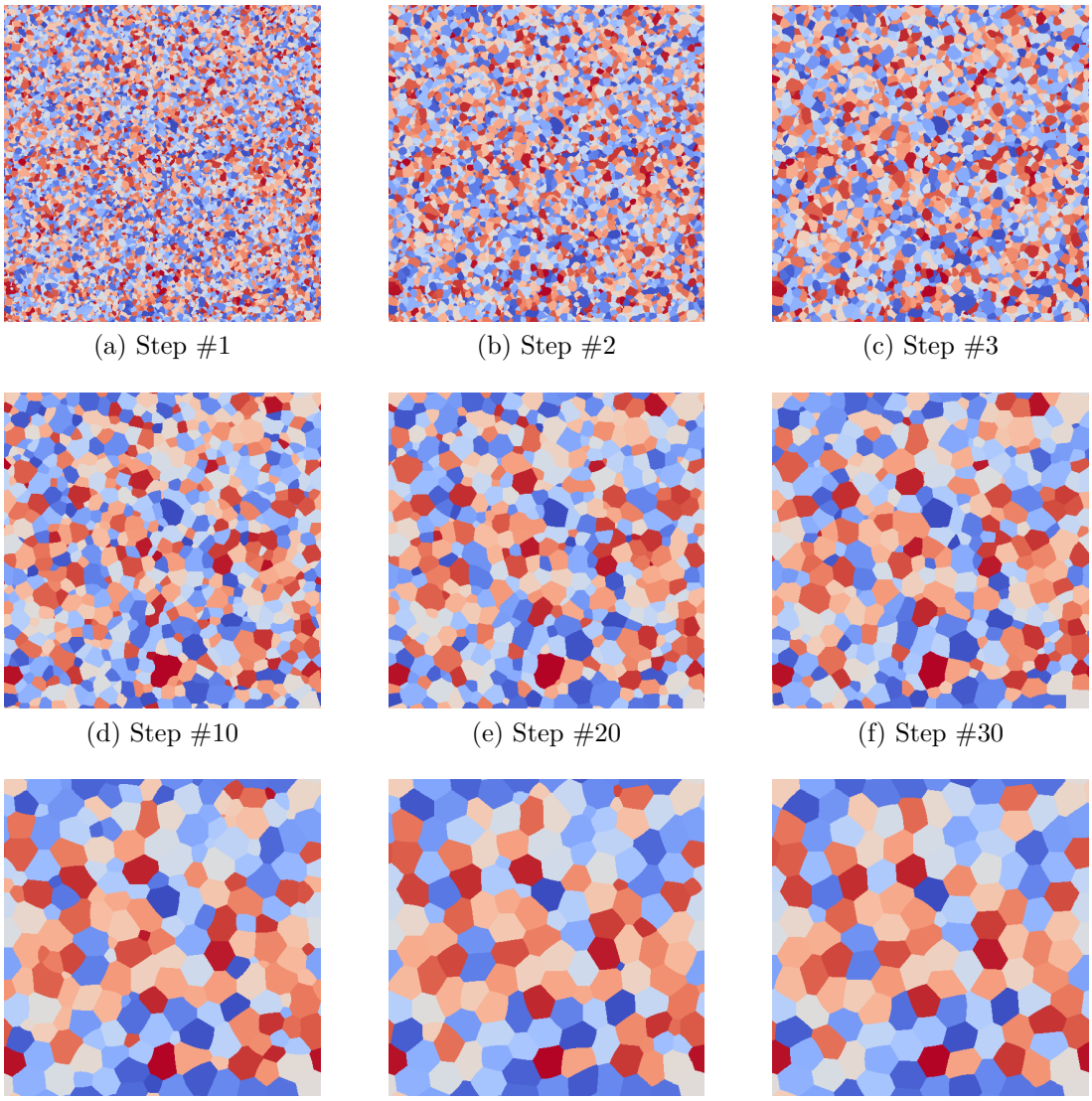


Figure 10: Surface minimization problem from a random initial density of 128 phases in 2D, at initial step.



(a) Step #1 (b) Step #2 (c) Step #3

(d) Step #10 (e) Step #20 (f) Step #30

(g) Step #100 (h) Step #200 (i) Step #300

Figure 11: Surface minimization problem from a random initial density of 128 phases in 2D.

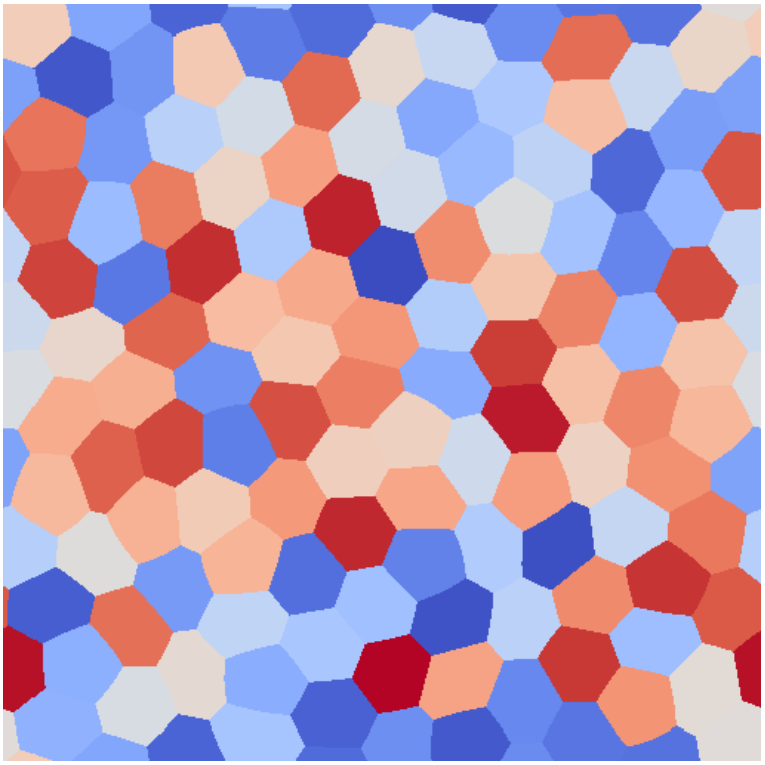


Figure 12: Surface minimization problem from a random initial density of 128 phases in 2D, at step #1000.

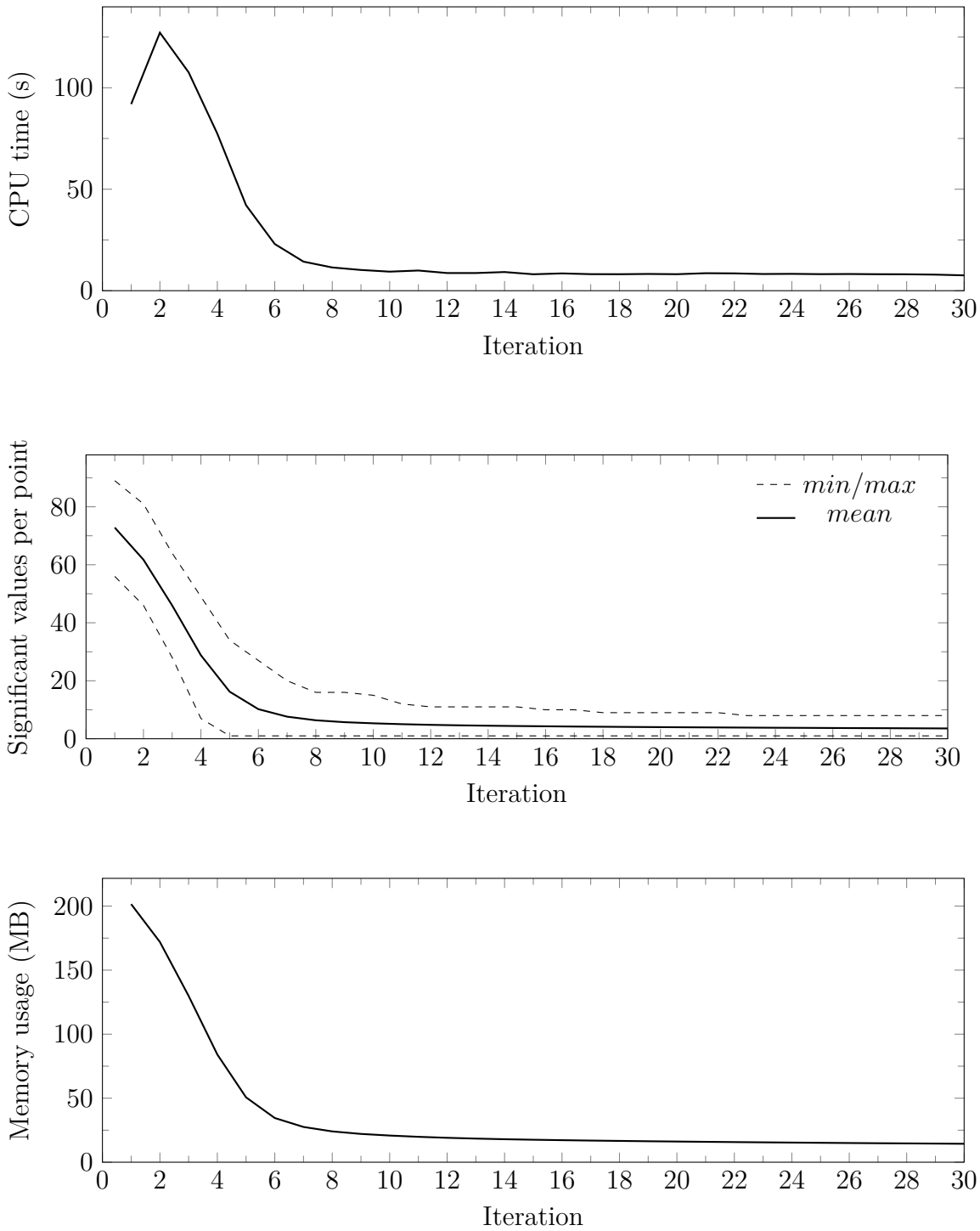


Figure 13: Performance evolution during the surface minimization problem with 128 phases.

6.1.2 The 3D case: the Weaire and Phelan structure

In the three-dimensional case, there is no proven minimal structure but the one from Weaire and Phelan (see [11]) is currently the best known. It is composed of eight cells that compose a filling of the periodic space $[0, 1]^3$.

As in [7], we try to find this structure using our volumetric formulation. To overcome the convergence issues that arise from previous case, we start for a random initial density on a low-resolution grid (of 64^3 points) to avoid an excessive memory usage and a high interface sharpness ε to have a fast convergence rate. When the system seems to be at equilibrium, we alternate decrease of ε and increase of the resolution up to 512^3.

The figures 14 show the result at different steps, with a volumetric rendering (some cells are partially transparent). The surface of some cells is visible in the figure 15a, and a slice of the data in 15b. In this last figure, we can clearly see the expected curvature of some faces of the cells.

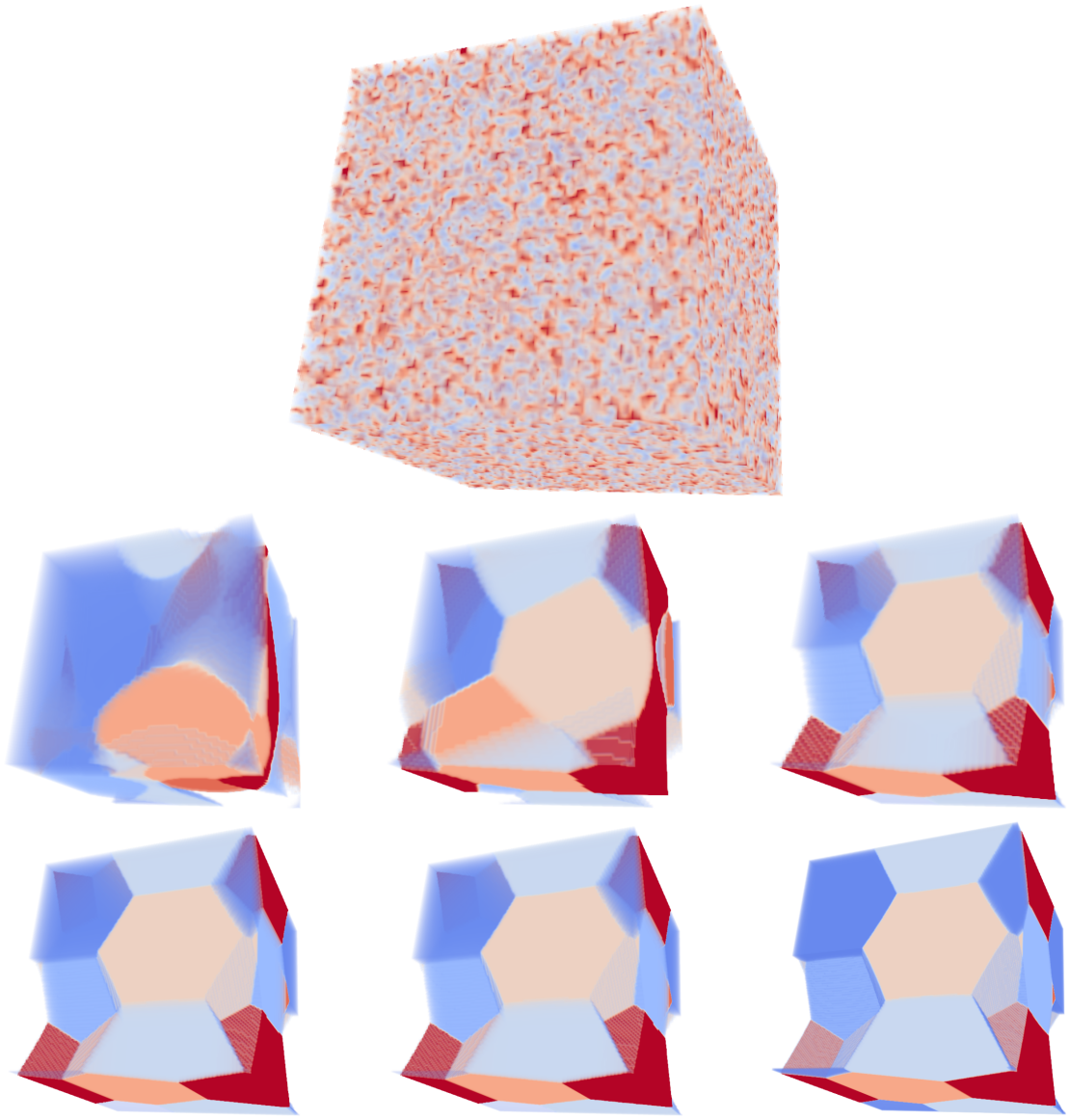
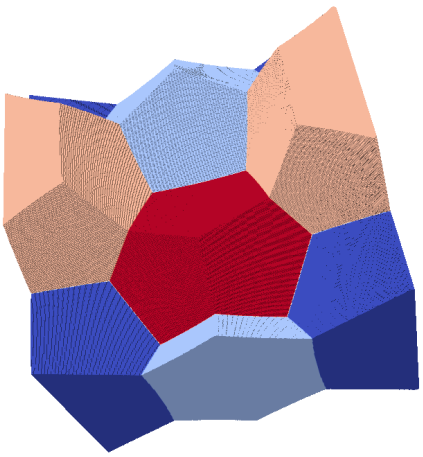
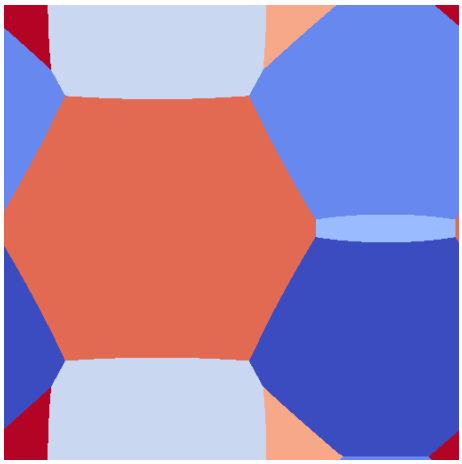


Figure 14: Surface minimization problem from an random initial density of 8 phases in 3D.



(a) Surface of some cells.



(b) Slice of the data.

Figure 15: Result of the surface minimization problem with 8 phases in 3D.

6.2 Estimation of the condensation coefficient

Another current application is about estimating the condensation coefficient α. Even though it has a direct influence on the interface speed (see equation (1)), this coefficient is know with a very poor precision, usually as $10^{-3} \leq \alpha \leq 10^{-1}$.

We propose here to compare simulated and physically observed results in order to estimate α. The reference observations come from an isothermal metamorphism experience applied on a snow sample at $-7\,°\mathrm{C}$ and from which the ice domain has been extracted by X-ray microtomography (see [3]) at the initial time and after 28 hours (see figure 16).

The comparison with multiple α values is easy because of the fact that our simplified isothermal evolution equation (15) (with C defined in (3)) is equivalent to the Allen-Cahn equation (16) up to a time-scale factor that is linearly dependent to the condensation coefficient α.

Thus, simulating the Allen-Cahn equation and taking snapshots at specific times is equivalent to use different α values for a simulation of 28 hours of isothermal metamorphism.

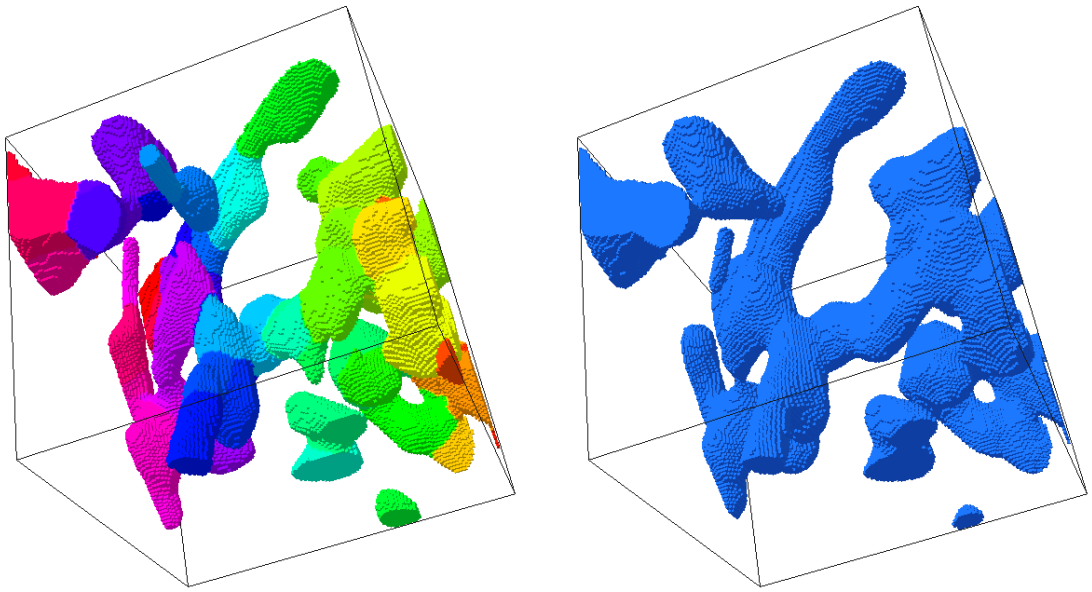
From that, we propose to find α that minimize the symmetric difference between the observed ice domain and the simulated one. The figure 17 shows the result when the ice domain is simulated as one phase.

On the other side, the figure 18 comes from a simulation where a curvature-based segmentation algorithm (see [4]) has been applied to the initial data to split the ice domain into separate grains. The metamorphism is then simulated using our multi-phase framework and the symmetric difference is taken from the full ice domain.

The initial segmented observation is illustrated in figure 16a and the final observation in figure 16b. They are composed of 131^3 sampling points and the symmetric difference, after simulation at full resolution, is taken on a central domain of 100^3 points in order to offset the fact that the snow sample is not periodic.

Without segmentation, the condensation coefficient that minimize the symmetric difference is approximately $\alpha \approx 0.026$, and $\alpha \approx 0.034$ when the grains are initially segmented. The difference in those estimations and the fact that the segmented-based simulation has a higher symmetric difference volume may be explained by an over-segmentation of the curve-based algorithm.

All these values are likely but the use of a more complete model is necessary to have a better confidence is those results.



(a) Initial observation with curvature-based segmentation (61 ice grains) using [4].

(b) Observation after 28 hours, without grain segmentation.

Figure 16: Initial and final observation of a snow sample during an isothermal metamorphism experiment at $-7\,^{\circ}\mathrm{C}$, by X-ray microtomography (see [3]).

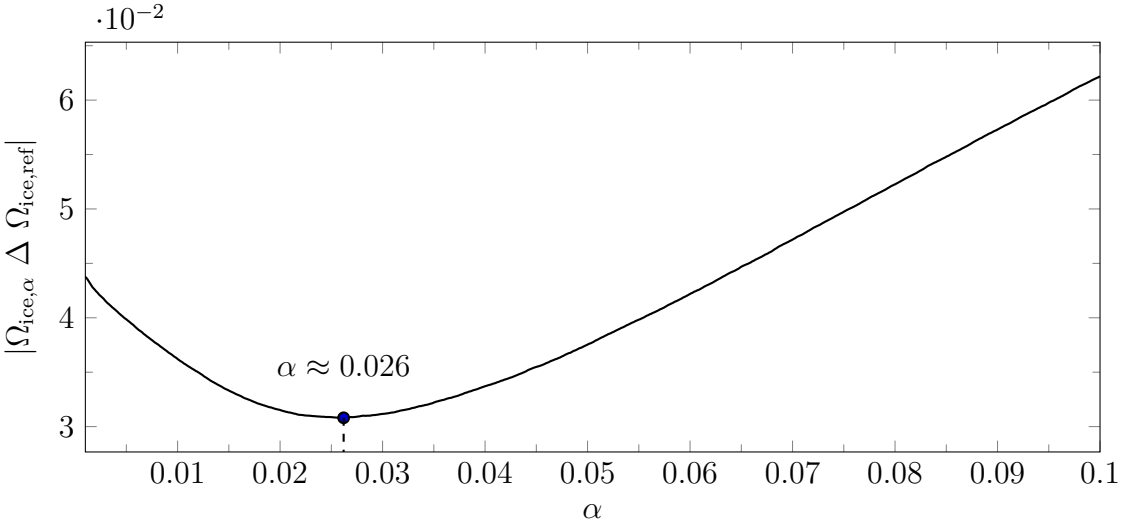


Figure 17: Volume of the symmetric difference between the ice phase of the observed state after 28 hours of isothermal metamorphism, and the simulated result from the non-segmented initial observation and for different condensation's coefficient values.

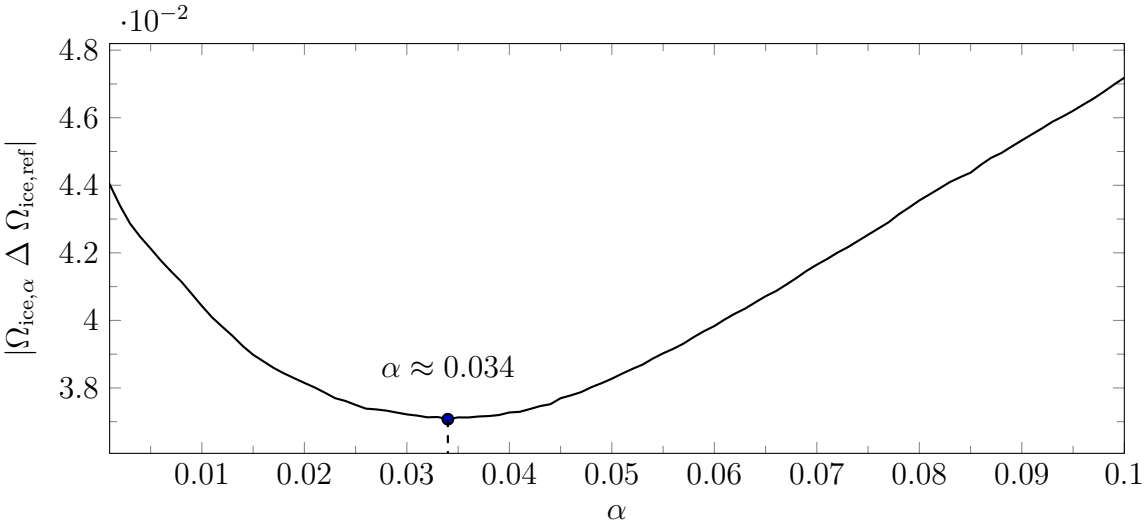


Figure 18: Volume of the symmetric difference between the ice phase of the observed state after 28 hours of isothermal metamorphism, and the simulated result from the curvature-based grain segmentation of the initial observation and for different condensation's coefficient values.

7 Conclusion

In this report, we have proposed a new reformulation of the Allen-Cahn equation in the multiphasic case, with non-overlapping and volume conservation properties, adapted to the use of an efficient computer memory structure based on the significant values of the phase-fields. This combination allows us to run multi-grain snow metamorphism simulations with a low memory usage and an optimized computational time.

We will now focus on adding more complex terms to the model, like anisotropy, variable surface tensions, vapor diffusion in the gas domain and temperature diffusion in the whole domain.

References

[1] E. Bretin. *Mouvements par courbure moyenne et méthode de champs de phase.* PhD thesis, Institut National Polytechnique de Grenoble-INPG, 2009.

[2] N. Calonne. *Physique des métamorphoses de la neige sèche : de la microstructure aux propriétés macroscopiques.* PhD thesis, Université de Grenoble, 2014.

[3] N. Calonne, F. Flin, B. Lesaffre, A. Dufour, J. Roulle, P. Puglièse, A. Philip, F. Lahoucine, C. Geindreau, J.-M. Panel, et al. Celldym: A room temperature operating cryogenic cell for the dynamic monitoring of snow metamorphism by time-lapse x-ray microtomography. *Geophysical Research Letters*, 42:3911–3918, 2015.

[4] F. Flin, B. Lesaffre, A. Dufour, L. Gillibert, A. Hasan, S. Rolland du Roscoat, S. Cabanes, and P. Pugliese. On the computations of specific surface area and specific grain contact area from snow 3d images. *Physics and Chemistry of Ice*, pages 321–328, 2011.

[5] J. Gruber, N. Ma, Y. Wang, A. D. Rollett, and G. S. Rohrer. Sparse data structure and algorithm for the phase field method. *Modelling and Simulation in Materials Science and Engineering*, 14(7):1189, 2006.

[6] T. C. Hales. The honeycomb conjecture. *Discrete & Computational Geometry*, 25(1):1–22, 2001.

[7] É. Oudet. Approximation of partitions of least perimeter by γ-convergence: around kelvin's conjecture. *Experimental Mathematics*, 20(3):260–270, 2011.

[8] R. I. Saye and J. A. Sethian. The voronoi implicit interface method for computing multiphase physics. *Proceedings of the National Academy of Sciences*, 108(49):19498–19503, 2011.

[9] L. Vanherpe, N. Moelans, B. Blanpain, and S. Vandewalle. Bounding box algorithm for three-dimensional phase-field simulations of microstructural evolution in polycrystalline materials. *Physical Review E*, 76(5):056702, 2007.

[10] S. Vedantam and B. S. V. Patnaik. Efficient numerical algorithm for multiphase field simulations. *Physical Review E*, 73(1):016703, 2006.

[11] D. Weaire and R. Phelan. A counter-example to kelvin's conjecture on minimal surfaces. *Philosophical Magazine Letters*, 69(2):107–110, 1994.