Computational Materials Science 160 (2019) 315–333

Contents lists available at ScienceDirect

Computational Materials Science

journal homepage: www.elsevier.com/locate/commatsci

Evolution of two-dimensional grain boundary networks implemented in GPU

Alejandro H.J. Sazo[a], Pablo Ibarra S.[a], Ariel Sanhueza R.[a], Francisco J.A. Casas[a], Claudio E. Torres[a,b,*], Maria Emelianenko[c], Dmitry Golovaty[d]

[a] Departamento de Informática, Universidad Técnica Federico Santa María, Chile
[b] Centro Científico Tecnológico de Valparaíso, Chile
[c] George Mason University, VA, United States
[d] University of Akron, OH, United States

ARTICLE INFO

Keywords:
Polycrystalline materials
Grain growth
Numerical simulation
Spectral methods

ABSTRACT

We consider a variational model for two-dimensional grain growth that allows for finite mobilities of boundaries and boundary junctions. We use a collocation method to describe the grain boundaries and posit evolution equations that ensure energy dissipation. The resulting algorithm is amenable to parallelization and a GPU implementation described in this work leads to significant reduction in computational time. We benchmark our code against known numerical and analytical solutions in some special cases. We also conduct an extensive analysis of statistics that develop for various parameter regimes.

1. Introduction

Polycrystalline materials consist of grains separated by boundaries. The corresponding configuration of grains defines the macroscopic physical characteristics of polycrystals, such as structural strength, electric conductivity, durability, among other. The material properties of a sample can therefore be modified by manipulating its grains structure. This is typically done via a combination of heating and deformation. For example, during annealing, polycrystalline materials undergo a process known as coarsening, when some grains grow at the expense of others [1–4]. The goal of modeling considered in the present paper is to predict evolution of the grain boundary structure that develops as the result of coarsening.

There is a number of numerical approaches available in the literature to study grain growth by means of computational simulations. Among them are Monte Carlo Potts models [5–12], molecular dynamics [13], and front-tracking methods. The latter can be further sub-divided into the phase field models [14–20], diffusion generated motion [21–23], grain boundary discretization with curvature-driven motion [24–28] and vertex-driven motion [29–33]. For example, within the phase field approach, Lobkovsky et al. [14] study two-dimensional dynamics of crystalline grains using gradient flow, including rotation of grains. In [15], the "Voronoi Implicit Interface Method" is analyzed for tracking multiple interacting and evolving phases. The authors report that they can handle triple and quadruple junctions in both two and three dimensions. Vondrous et al. [16] couple a phase-field method with a finite element approach to simulate recrystallization. They develop a multidimensional decomposition to efficiently parallelize the computation and study the isotropic and the Read-Shockley grain boundary energy functions.

In [18], the authors use the phase field crystal model for grain growth, the model minimize the surface energy until a single grain prevails. A multiphase-field theory is discussed in [19] where mathematical and physical consistency is ensured by construction of an appropriate free energy functional. In [20], the authors propose a 2D theoretical framework that couples a Cahn-Hillard model with a phase field crystal model to explore the effects of coarse-grained lattice symmetry and distortions on a phase transition process. In the case of diffusion generated motion, Elsey et al. [21] propose an efficient algorithm for simulating curvature flow for large networks of curves in two and three dimensions. This is achieved using a level set method and allowing a single signed distance function to represent a large subset of spatially separated grains. In [22], they use a distance function to generate diffusion based motion and produce an efficient numerical algorithm. The interfaces are represented implicitly, allow automatic topological transitions and arbitrarily large time-steps. A mean

* Corresponding author.
E-mail addresses: alejandro.sazo@usm.cl (A.H.J. Sazo), pablo.ibarras@alumnos.usm.cl (P. Ibarra S.), ariel.sanhuezar@usm.cl (A. Sanhueza R.), francisco.casas.13@sansano.usm.cl (F.J.A. Casas), ctorres@inf.utfsm.cl (C.E. Torres), memelian@gmu.edu (M. Emelianenko), dmitry@uakron.edu (D. Golovaty).

https://doi.org/10.1016/j.commatsci.2019.01.022
Received 22 October 2018; Received in revised form 12 January 2019; Accepted 12 January 2019

curvature motion and surface diffusion is considered in [23,24] The effect of triple-junction drag was considered by the number of authors [9,34–38,10], usually within front-tracking models. These models principally describe two-dimensional systems of grains, but some three-dimensional results are also available [6,12,29,39,40]. PDE-based mesoscale models focused on the role of the critical events during evolution have also recently been investigated, e.g. in [41,42]. A recent review of unresolved issues in grain growth is presented in [43].

In this paper, we focus on a variational model for two-dimensional grain growth that allows for finite mobilities of boundaries and boundary junctions. We take advantage of Graphics Processing Unit (GPU) computations with CUDA [44] for the management of large numerical simulations [19,45], which calls for new algorithms. The principal reason to explore this computer architecture is the need to obtain appropriate statistics that requires simulations of very large systems of grains [33,46–48].

The algorithm proposed preserves the continuous description of grain boundaries by taking advantage of a collocation method [49]. Using this approach allows us to obtain an integral description of the grain boundaries configuration when considering finite mobilities for grain boundaries and grain boundary junctions.

The paper is organized as follows. We set up the problem in Section 2, then introduce the boundary representation in Section 3. The evolution equations for the triple junctions and the normal component of the velocity for the interior points are outlined in Section 4, followed by the computation of the tangential component of the velocity of the interior points in Section 5. The numerical algorithms developed and the numerical experiments performed are presented in Sections 6 and 7, respectively. Finally, the conclusions and future work are given in Section 8.

2. Problem setup

We begin by setting up the problem, mostly following the formalism developed in [50]. Suppose that the network of grains occupies a square domain $[0, 1]^2 \subset \mathbb{R}^2$, subject to periodic boundary conditions. The grain structure consists of a set of disjoint grains covering the entire domain $[0, 1]^2$. We denote the set of grains by

$$\mathbf{\Sigma} = \mathbf{\Sigma}(t) = \{\Sigma^{(1)}, \Sigma^{(2)}, ..., \Sigma^{(N)}\}, \quad N = N(t).$$

Each grain boundary is defined as,

$$\mathbf{\Gamma} = \mathbf{\Gamma}(t) = \{\Gamma^{(1)}, \Gamma^{(2)}, ..., \Gamma^{(K)}\}, \quad K = K(t),$$

and each boundary is parameterized as follows:

$$\Gamma^{(k)}(t) = \{\boldsymbol{\xi}^{(k)}(s, t), \quad 0 \leqslant s \leqslant 1\}, \quad k \in K.$$

These boundaries meet at triple junctions. The triple junctions are the start or the end point of exactly three boundaries. Let $k_1, k_2, k_3 \in K$ be the indices of three different boundaries. For $s_{k_i} \in \{0, 1\}$, the triple junctions can be represented as,

$$\boldsymbol{\xi}^{(k_1)}(s_{k_1}, t) = \boldsymbol{\xi}^{(k_2)}(s_{k_2}, t) = \boldsymbol{\xi}^{(k_3)}(s_{k_3}, t).$$

The total energy is given by the following expression:

$$E(t) = \sum_{k=1}^{K} \int_0^1 \gamma_k \left(\Delta\alpha_k\right) \left\| \mathbf{l}^{(k)}\left(s, t\right) \right\| ds, \tag{1}$$

where $\mathbf{l}^{(k)}(s, t) = \frac{\partial}{\partial s}(\boldsymbol{\xi}^{(k)}(s, t))$, γ_k is the grain boundary energy of boundary is the misorientation between the grains that meet at boundary $\Gamma^{(k)}$ and $||\cdot||$ is the l^2-norm. We will omit, except when needed, the spatial and temporal dependence. This energy is equal to the required amount of work to create an infinitesimal grain boundary [50]. The evolution equations of the system are obtained by finding the derivative with respect to time of the total energy and making it negative, so the system is dissipative. Computing the derivative of (1) with respect to t we get:

$$\frac{d}{dt}E(t) = \sum_{k=1}^{K} \int_0^1 \gamma_k \frac{\partial}{\partial t}(||\mathbf{l}^{(k)}||) ds = \sum_{k=1}^{K} \int_0^1 \gamma_k \mathbf{T}^{(k)} . \frac{\partial}{\partial s}(\mathbf{v}^{(k)}) ds, \tag{2}$$

where $\mathbf{T}^{(k)} = \frac{\mathbf{l}^{(k)}}{||\mathbf{l}^{(k)}||}$. Note that (2) is valid as long as $K'(t) = 0$ within $[t_1, t_2]$, i.e. there are no topological changes. Integrating by parts (2), we finally get,

$$\frac{d}{dt}E(t) = -\sum_{k=1}^{K} \int_0^1 \gamma_k \frac{\partial}{\partial s}(\mathbf{T}^{(k)}) \cdot \mathbf{v}^{(k)} ds + \sum_{m=1}^{M} \mathbf{v}_m \cdot \sum_{l=1}^{3} \gamma_{m,l} \mathbf{T}^{(m,l)}. \tag{3}$$

3. Interpolation of the boundaries

We propose a representation of the grain boundaries based on a Lagrange interpolation on s as follows:

$$\boldsymbol{\xi}^{(k)}\left(s, t\right) = \sum_{i=1}^{n} \mathbf{x}_i^{(k)}(t) \phi_i(s), \tag{4}$$

where $\phi_i(s) = \frac{l_i(s)}{l_i(s_i)}$, $l_i(s) = \prod_{\substack{k=1 \\ k \neq i}}^{n} \left(s - s_k\right)$ and the s_k will be defined in Section 6.1. The points $\mathbf{x}_1^{(k)}$ and $\mathbf{x}_n^{(k)}$ are the triple junctions. The rest of the collocation points $\mathbf{x}_p^{(k)}(t)$, with $p \in \{2, \cdots, n-1\}$, are called interior points and initially placed equispaced along a straight line connecting $\mathbf{x}_1^{(k)}$ and $\mathbf{x}_n^{(k)}$. The different initial configuration of interior points is also possible but not discussed in this paper. Notice that if we parameterize the grain boundary with only two points, this approach becomes a vertex-based model [32].

On the other hand, curvature based evolution requires that the velocity of the grain boundary moves in the normal direction and proportional to its curvature. This means that:

$$\mathbf{v}(s, t) = \kappa(s, t)\mathbf{N}(s, t). \tag{5}$$

In our case, we need to ensure this is actually happening. This behavior can be built from the definition of the curvature, which is the derivative of the unitary tangent vector with respect to the arc length ($\mathcal{L}$):

$$\begin{aligned} \frac{\partial}{\partial \mathcal{L}}(\mathbf{T}(s, t)) &= \kappa\left(s, t\right)\mathbf{N}\left(s, t\right) \\ \frac{\partial}{\partial s}(\mathbf{T}(s, t))\frac{ds}{d\mathcal{L}} &= \kappa\left(s, t\right)\mathbf{N}\left(s, t\right) \\ \frac{\partial}{\partial s}(\mathbf{T}(s, t))\frac{1}{||\mathbf{l}(s,t)||} &= \kappa\left(s, t\right)\mathbf{N}\left(s, t\right), \end{aligned} \tag{6}$$

where $\frac{ds}{d\mathcal{L}}$ was obtained from the definition of arc length $\mathcal{L}(s) = \int_0^s ||\mathbf{l}(\hat{s}, t)|| d\hat{s}$. This means that if we want curvature-based grain growth, we can compute $\kappa(s, t)\mathbf{N}(s, t)$ by means of $\frac{1}{||\mathbf{l}(s,t)||}\frac{\partial}{\partial s}(\mathbf{T}(s, t))$, which is the derivative of a unitary vector scaled by $||\mathbf{l}(s, t)||^{-1}$.

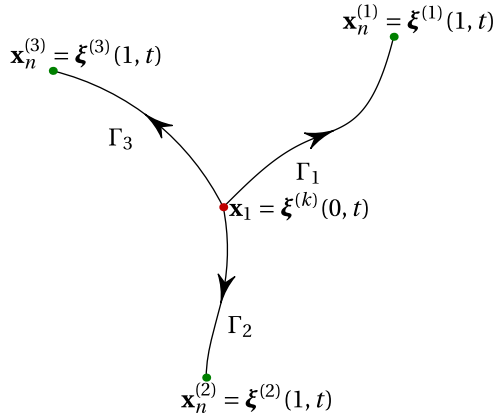
4. Derivation of triple junction evolution and the normal component of the velocity of the interior points

For the sake of simplicity of notation, we will omit the scaling γ_k from $\gamma_k \mathbf{T}^{(k)}$, but it will be re-introduced at the end.

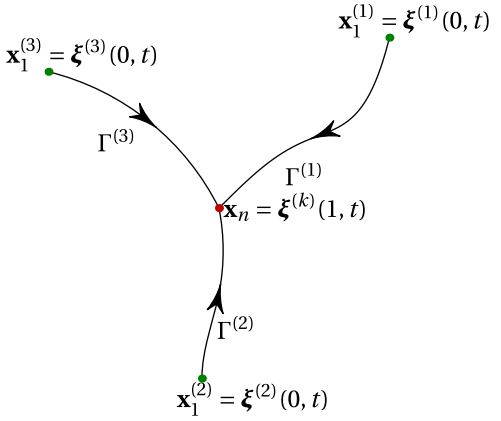
Let us consider Eq. (3) and plug the parameterization (4) in it:

$$\frac{d}{dt}E(t) = -\sum_{k=1}^{K} \int_0^1 \frac{\partial}{\partial s}(\mathbf{T}^{(k)}) \cdot \left[\sum_{i=1}^{n} \dot{\mathbf{x}}_i^{(k)}(t) \phi_i(s)\right] ds + \sum_{m=1}^{M} \mathbf{v}_m \sum_{l=1}^{3} \mathbf{T}^{(m,l)}.$$

Reordering the terms we obtain:



(a) Set of boundaries sharing a triple junction at the start of their parameterizations.



(b) Set of boundaries sharing a triple junction at the end of their parameterizations.

Fig. 1. Two different sets of boundaries sharing a triple junction. Arrows indicates direction of parameterization.

$$\frac{d}{dt}E(t) = \underbrace{-\sum_{k=1}^{K}\left[\sum_{i=1}^{n}\dot{\mathbf{x}}_i^{(k)}(t)\cdot\int_0^1\frac{\partial}{\partial s}(\mathbf{T}^{(k)})\phi_i(s)\,ds\right]}_{(a)} + \underbrace{\sum_{m=1}^{M}\mathbf{v}_m\sum_{l=1}^{3}\mathbf{T}^{(m,l)}}_{(b)}.$$

As indicated previously, the triple junctions of a boundary are the collocation points $\boldsymbol{\xi}^{(k)}(0,t)$ and $\boldsymbol{\xi}^{(k)}(1,t)$, and are associated to the Lagrange polynomials $\phi_1(s)$ and $\phi_n(s)$, respectively. This allows us to split the equation in two parts, (a) and (b). If we extract the terms related to triple junctions in (a) and add them to (b) we obtain:

$$\frac{d}{dt}E(t) = -\sum_{k=1}^{K}\left[\sum_{i=2}^{n-1}\dot{\mathbf{x}}_i^{(k)}\cdot\int_0^1\frac{\partial}{\partial s}(\mathbf{T}^{(k)})\phi_i(s)\,ds\right] + \sum_{m=1}^{M}\mathbf{v}_m\cdot\sum_{l=1}^{3}\mathbf{T}^{(m,l)} - \underbrace{\sum_{k=1}^{K}\left[\dot{\mathbf{x}}_1^{(k)}\cdot\int_0^1\frac{\partial}{\partial s}(\mathbf{T}^{(k)})\phi_1(s)\,ds + \dot{\mathbf{x}}_n^{(k)}\cdot\int_0^1\frac{\partial}{\partial s}(\mathbf{T}^{(k)})\phi_n(s)\,ds\right]}_{(c)}. \tag{7}$$

The terms we extracted are now explicitly indicated in (c), in Eq. (7). For simplicity, consider Fig. 1a, where the set of boundaries and $\Gamma^{(3)}$ share a triple junction $\boldsymbol{\xi}^{(k)}(0,t) = \mathbf{x}_1 = \mathbf{x}_1^{(1)} = \mathbf{x}_1^{(2)} = \mathbf{x}_1^{(3)}$, that is, their parameterization starts at that triple junction. Expression (c) in Eq. (7) for this example becomes:

$$\sum_{k=1}^{3}\dot{\mathbf{x}}_1^{(1)}\cdot\int_0^1\frac{\partial}{\partial s}(\mathbf{T}^{(k)})\phi_1(s)\,ds + \sum_{k=1}^{K-3}\dot{\mathbf{x}}_1^{(k)}\cdot\int_0^1\frac{\partial}{\partial s}(\mathbf{T}^{(k)})\phi_1(s)\,ds + \sum_{k=1}^{K}\dot{\mathbf{x}}_n^{(k)}\cdot\int_0^1\frac{\partial}{\partial s}(\mathbf{T}^{(k)})\phi_n(s)\,ds.$$

The velocity of the shared triple junction remains the same, no matter which boundary is considered for its computation. Therefore we can factorize by $\dot{\mathbf{x}}_1^{(1)}$, resulting in:

$$\dot{\mathbf{x}}_1^{(1)}\cdot\sum_{k=1}^{3}\int_0^1\frac{\partial}{\partial s}(\mathbf{T}^{(k)})\phi_1(s)\,ds + \sum_{k=1}^{K-3}\dot{\mathbf{x}}_1^{(k)}\cdot\int_0^1\frac{\partial}{\partial s}(\mathbf{T}^{(k)})\phi_1(s)\,ds + \sum_{k=1}^{K}\dot{\mathbf{x}}_n^{(k)}\cdot\int_0^1\frac{\partial}{\partial s}(\mathbf{T}^{(k)})\phi_n(s)\,ds.$$

Let us call the triple junction $\mathbf{x}_1^{(1)}$ as $\mathbf{x}_m$ and its velocity $\dot{\mathbf{x}}_m$, where the sub-index m indicates that we are re-indexing triple junctions. We can rewrite (c) as:

$$\sum_{m=1}^{M}\dot{\mathbf{x}}_m\cdot\sum_{l=1}^{3}\int_0^1\frac{\partial}{\partial s}(\mathbf{T}^{(m,l)})\phi_{1,m}(s)\,ds$$

Replacing it in (7) we obtain:

$$\frac{d}{dt}E(t) = -\sum_{k=1}^{K}\left[\sum_{i=2}^{n-1}\dot{\mathbf{x}}_i^{(k)}(t)\cdot\int_0^1\frac{\partial}{\partial s}(\mathbf{T}^{(k)})\phi_i(s)\,ds\right] - \sum_{m=1}^{M}\left[\dot{\mathbf{x}}_m\cdot\sum_{l=1}^{3}\int_0^1\frac{\partial}{\partial s}(\mathbf{T}^{(m,l)})\phi_{1,m}(s)\,ds\right] + \sum_{m=1}^{M}\mathbf{v}_m\cdot\sum_{l=1}^{3}\mathbf{T}^{(m,l)}.$$

Notice that $\mathbf{v}_m = \dot{\mathbf{x}}_m$. Thus, Eq. (3) becomes:

$$\frac{d}{dt}E(t) = -\sum_{k=1}^{K}\left[\sum_{i=2}^{n-1}\dot{\mathbf{x}}_i^{(k)}(t)\cdot\int_0^1\frac{\partial}{\partial s}(\mathbf{T}^{(k)})\phi_i(s)\,ds\right] + \sum_{m=1}^{M}\dot{\mathbf{x}}_m\left[\sum_{l=1}^{3}\mathbf{T}^{(m,l)} - \int_0^1\frac{\partial}{\partial s}(\mathbf{T}^{(m,l)})\phi_{1,m}(s)\,ds\right].$$

From here on, we can obtain the expressions for the *average normal* component of the velocity for interior points and the velocity for the triple junctions that ensures the system effectively decreases its energy. Including the grain boundary energy $\gamma_{m,l}$, the velocity for a triple junction $\mathbf{x}_m$ is,

$$\dot{\mathbf{x}}_m(t) = \lambda\sum_{l=1}^{3}\gamma_{m,l}\left(-\mathbf{T}^{(m,l)} + \int_0^1\frac{\partial}{\partial s}(\mathbf{T}^{(m,l)})\phi_{1,m}(s)\,ds\right), \tag{8}$$

where λ is the mobility coefficient for the triple junctions. Similarly, the *average normal* component of the velocity for an interior point $\mathbf{x}_i^{(k)}$ is:

$$\beta_i^{(k)}(t)\widehat{\mathbf{N}}_i^{(k)}(t) = \frac{\mu\gamma_k}{\|\mathbf{l}(s_i,t)\|}\int_0^1\frac{\partial}{\partial s}(\mathbf{T}^{(k)})\phi_i(s)\,ds, \quad i \neq \left\{1, n\right\}, \tag{9}$$

where μ is the mobility coefficient for the interior points, γ_k is the grain boundary energy, $\widehat{\mathbf{N}}_i^{(k)}(t)$ is the unitary *average normal* vector obtained from the integration of the normal components $\frac{\partial}{\partial s}(\mathbf{T}^{(k)})$ together with the basis function $\phi_i(s)$. Notice that for Eq. (9), we used the identity from (6). However, here it becomes an approximation since we are integrating first $\frac{\partial}{\partial s}(\mathbf{T}^{(k)})$ and then scaling it by $\|\mathbf{l}(s_i,t)\|^{-1}$. We also assumed that for computing (8) the parameterization started at that triple junction, where in general this may not be case, see Fig. 1. However, we can easily obtain that by making a change of variable $\hat{s} = 1 - s$ and the assumption holds.

Finally, we have been able to build a the evolution equations for triple junctions and the average normal component of velocity of the interior points with finite mobilities, which are energy decreasing by construction.

5. Tangential component of the velocity for interior points

In Eq. (9), the average normal component of the velocity of the interior points was defined. This was obtained to ensure the evolution of the boundary is energy decreasing. Specifically, the evolution is chosen such that it is parallel to the $\int_0^1 \frac{\partial}{\partial s}(\mathbf{T}^{(k)})\phi_i(s)ds$ vector. Thus, we are free to add a tangential component $\widehat{\mathbf{T}}_i$ to the velocity as long as it is orthogonal to the $\int_0^1 \frac{\partial}{\partial s}(\mathbf{T}^{(k)})\phi_i(s)ds$ vector. This means that the total velocity of an interior point will be defined as:

$$\dot{\mathbf{x}}_i(t) = \beta_i(t)\widehat{\mathbf{N}}_i(t) + \alpha_i(t)\widehat{\mathbf{T}}_i(t). \tag{10}$$

We have omitted the grain boundary dependence (k) and will omit the time dependence for the sake of clear notation. We have introduced the *hat* notation (^) to indicate the average normal and tangential components. Notice that the average normal $\widehat{\mathbf{N}}_i$ is not necessarily equal to the normal $\mathbf{N}_i = \mathbf{N}(s_i)$, unless the grain boundary is a straight line, but it is required that $\widehat{\mathbf{N}}_i \cdot \widehat{\mathbf{T}}_i(t) = 0$. The main reason to add the tangential component to the velocity is numerical, in order to avoid the coalescence of collocation points and making the computation unstable. Fortunately, this problem has already been extensively studied [51–53]. We will follow the same procedure here. Specifically, this means satisfying the following equations at the interior points:

$$\frac{d}{dt}\left(\frac{\|\mathbf{l}_i(t)\|}{\mathcal{L}(t)}\right) = 0, \quad i \in 2,\ 3,\ \ldots, n-2$$

where $\mathbf{l}_i(t) = \mathbf{l}(s_i, t)$, $\|\mathbf{l}_i(t)\|$ denotes the *local* arc length at the collocation point i and $\mathcal{L}(t)$ is the arc length of the grain boundary under analysis. Computing the derivative with respect to time, we obtain the following equation:

$$\begin{aligned}
&\frac{\frac{d}{dt}(\|\mathbf{l}_i(t)\|)\mathcal{L}(t) - \frac{d}{dt}(\mathcal{L}(t))\left\|\mathbf{l}_i(t)\right\|}{\mathcal{L}^2(t)} = 0,\\
&\frac{d}{dt}(\|\mathbf{l}_i(t)\|)\mathcal{L}(t) - \frac{d}{dt}(\mathcal{L}(t))\|\mathbf{l}_i(t)\| = 0,\\
&\frac{d}{dt}(\|\mathbf{l}_i(t)\|)\mathcal{L}(t) = \frac{d}{dt}(\mathcal{L}(t))\|\mathbf{l}_i(t)\|,
\end{aligned} \tag{11}$$

where $\frac{d}{dt}(\|\mathbf{l}_i(t)\|) = \mathbf{T}_i(t)\cdot\frac{d}{dt}(\mathbf{l}_i(t))$ and $\frac{d}{dt}(\mathcal{L}(t)) = \int_0^1 \mathbf{T}(s,t)\cdot\frac{\partial}{\partial t}(\mathbf{l}(s,t))ds$. Eq. (11) needs to be applied to each interior point, but before we do that we need to build each of its elements. Finally, we will obtain a linear system of equations for the α_is. For simplicity, we will list here all the components we will use:

$$\begin{aligned}
&\boldsymbol{\xi}\left(s,t\right) = \sum_{i=1}^{n} \mathbf{x}_i(t)\phi_i(s),\\
&\frac{\partial}{\partial s}\boldsymbol{\xi}\left(s,t\right) = \mathbf{l}\left(s,t\right) = \sum_{i=1}^{n} \mathbf{x}_i(t)\phi_i'(s) = \sum_{i=1}^{n} \mathbf{y}_i(t)\phi_i(s),
\end{aligned}$$

where the computation of the $\mathbf{y}_i(t)$ vector is discussed next in Section 6.2. Here, we compute the derivative with respect to time of $\mathbf{l}(s,t)$ and we obtain,

$$\begin{aligned}
&\frac{\partial}{\partial t}\mathbf{l}\left(s,t\right) = \sum_{i=1}^{n} \dot{\mathbf{y}}_i(t)\phi_i(s),\\
&\frac{d}{dt}(\mathbf{l}_i(t)) = \dot{\mathbf{y}}_i(t).
\end{aligned}$$

Thus, plugging in each component to (11) we obtain,

$$\begin{aligned}
&\frac{d}{dt}(\|\mathbf{l}_i(t)\|)\mathcal{L}(t) = \frac{d}{dt}(\mathcal{L}(t))\|\mathbf{l}_i(t)\|\\
&\mathbf{T}_i(t)\cdot\frac{d}{dt}(\mathbf{l}_i(t))\mathcal{L}(t) = \int_0^1 \mathbf{T}(s,t)\cdot\frac{\partial}{\partial t}(\mathbf{l}(s,t))ds\,\|\mathbf{l}_i(t)\|\\
&\mathbf{T}_i(t)\cdot\dot{\mathbf{y}}_i(t)\mathcal{L}(t) = \int_0^1 \mathbf{T}\left(s,t\right)\cdot\left(\sum_{k=1}^{n} \dot{\mathbf{y}}_k(t)\phi_k(s)\right)ds\left\|\mathbf{l}_i(t)\right\|\\
&\mathbf{T}_i(t)\cdot\dot{\mathbf{y}}_i(t)\mathcal{L}(t) = \left(\sum_{k=1}^{n} \dot{\mathbf{y}}_k(t)\cdot\int_0^1 \mathbf{T}\left(s,t\right)\phi_k(s)ds\right)\left\|\mathbf{l}_i(t)\right\|,
\end{aligned}$$

moving $\|\mathbf{l}_i(t)\|$ to the left-hand-side, we obtain,

$$\mathbf{T}_i(t)\cdot\dot{\mathbf{y}}_i(t)\frac{\mathcal{L}(t)}{\|\mathbf{l}_i(t)\|} = \sum_{k=1}^{n} \dot{\mathbf{y}}_k(t)\cdot\underbrace{\int_0^1 \mathbf{T}\left(s,t\right)\phi_k(s)ds}_{\mathbf{w}_k(t)}, \quad i \in 2,\ \ldots, n-1. \tag{12}$$

Now, it is time to use the orthogonal decomposition (10) to obtain the $\alpha_i(t)$, which is achieved by the following linear transformation with the differentiation matrix D (See Section 6.2 for more details),

$$\begin{pmatrix} \dot{\mathbf{y}}_1(t) \\ \dot{\mathbf{y}}_2(t) \\ \vdots \\ \dot{\mathbf{y}}_i(t) \\ \vdots \\ \dot{\mathbf{y}}_{n-1}(t) \\ \dot{\mathbf{y}}_n(t) \end{pmatrix} = D \begin{pmatrix} \dot{\mathbf{x}}_1(t) \\ \beta_2(t)\widehat{\mathbf{N}}_2(t) + \alpha_2(t)\widehat{\mathbf{T}}_2(t) \\ \vdots \\ \beta_i(t)\widehat{\mathbf{N}}_i(t) + \alpha_i(t)\widehat{\mathbf{T}}_i(t) \\ \vdots \\ \beta_{n-1}(t)\widehat{\mathbf{N}}_{n-1}(t) + \alpha_{n-1}(t)\widehat{\mathbf{T}}_{n-1}(t) \\ \dot{\mathbf{x}}_n(t) \end{pmatrix} \tag{13}$$

where the vectors $\dot{\mathbf{y}}_i(t)$, $\widehat{\mathbf{N}}_i(t)$, $\widehat{\mathbf{T}}_i(t)$, $\dot{\mathbf{x}}_1(t)$ and $\dot{\mathbf{x}}_n(t)$ are considered row-vectors, the $\dot{\mathbf{x}}_1(t)$ and $\dot{\mathbf{x}}_n(t)$ are the triple junction velocities of the boundary. So,

$$\dot{\mathbf{y}}_i(t) = D_{i,1}\dot{\mathbf{x}}_1(t) + \sum_{j=2}^{n-1} D_{i,j}(\beta_j(t)\widehat{\mathbf{N}}_j(t) + \alpha_j(t)\widehat{\mathbf{T}}_j(t)) + D_{i,n}\dot{\mathbf{x}}_n(t).$$

Moreover, if we define $\mathbf{m}_i(t) = D_{i,1}\dot{\mathbf{x}}_1(t) + \sum_{j=2}^{n-1} D_{i,j}\beta_j(t)\widehat{\mathbf{N}}_j(t) + D_{i,n}\dot{\mathbf{x}}_n(t)$, this allows us to define $\dot{\mathbf{y}}_i(t) = \mathbf{m}_i(t) + \sum_{j=2}^{n-1} D_{i,j}\alpha_j(t)\widehat{\mathbf{T}}_j(t)$. Now we first compute the unknown part and the known part of the left-hand-side of (13), and we omit for now the coefficient $\frac{\mathcal{L}(t)}{\|\mathbf{l}_i(t)\|}$,

$$\begin{aligned}
\mathbf{T}_i(t)\cdot\dot{\mathbf{y}}_i(t) &= \mathbf{T}_i(t)\cdot\left(\mathbf{m}_i(t) + \sum_{j=2}^{n-1} D_{i,j}\alpha_j(t)\widehat{\mathbf{T}}_j(t)\right)\\
&= \mathbf{T}_i(t)\cdot\mathbf{m}_i(t) + \mathbf{T}_i(t)\cdot\left(\sum_{j=2}^{n-1} D_{i,j}\alpha_j(t)\widehat{\mathbf{T}}_j(t)\right)
\end{aligned}$$

Similarly for the right-hand-side of (12), we obtain the following for the k-th element of the sum,

$$\begin{aligned}
\mathbf{w}_k(t)\cdot\dot{\mathbf{y}}_k(t) &= \mathbf{w}_k(t)\cdot\left(\mathbf{m}_k(t) + \sum_{j=2}^{n-1} D_{k,j}\alpha_j(t)\widehat{\mathbf{T}}_j(t)\right)\\
&= \mathbf{w}_k(t)\cdot\mathbf{m}_k(t) + \mathbf{w}_k(t)\cdot\left(\sum_{j=2}^{n-1} D_{k,j}\alpha_j(t)\widehat{\mathbf{T}}_j(t)\right)
\end{aligned}$$

Thus, collecting the unknown terms on the left-hand-side and the known terms on the right-hand-side of (12), we finally obtain the linear system of equations that we need to solve in order to obtain the $\alpha_i(t)$,

$$\begin{aligned}
&\frac{\mathcal{L}(t)}{\|\mathbf{l}_i(t)\|}\mathbf{T}_i(t)\cdot\left(\mathbf{m}_i(t) + \sum_{j=2}^{n-1} D_{i,j}\cdot\widehat{\mathbf{T}}_j(t)\alpha_j(t)\right)\\
&= \sum_{k=1}^{n}\left(\mathbf{m}_k(t) + \sum_{j=2}^{n-1} D_{k,j}\,\widehat{\mathbf{T}}_j(t)\alpha_j(t)\right)\cdot\mathbf{w}_k(t).
\end{aligned}$$

Collecting the coefficient that multiplies $\alpha_i(t)$, and rearranging the right-hand-side we obtain,

$$\begin{aligned}
\sum_{j=2}^{n-1}\left(D_{i,j}\frac{\mathcal{L}(t)}{\|\mathbf{l}_i(t)\|}\mathbf{T}_i(t) - \sum_{k=1}^{n} D_{k,j}\mathbf{w}_k(t)\right)\cdot\widehat{\mathbf{T}}_j(t)\alpha_j(t) &= -\frac{\mathcal{L}(t)}{\|\mathbf{l}_i(t)\|}\mathbf{m}_i(t)\cdot\mathbf{T}_i(t)\\
&\quad + \sum_{k=1}^{n}\mathbf{m}_k(t)\cdot\mathbf{w}_k(t).
\end{aligned}$$

Therefore, we have shown that to obtain the $\alpha_i(t)$ coefficient we need to solve a linear system of equation of the size of the number of

interior points for each grain boundary. Every time step for the interior points needs to be explicitly computed, as do quantities that depend on the velocity of the interior points such as the extension time of the grain boundary. Fortunately, this can be handle in parallel in the GPU.

6. Algorithms for numerical implementation of the model

Before we introduce the main algorithm for the evolution of the grain structure, we need to analyze and solve a series of numerical challenges. We first propose a convenient way to parameterize the grain boundaries and triple junctions. Afterwards we study and propose a novel way to compute: the derivative of a unitary vector, the collision time of grain boundaries and the velocity for interior points and triple junctions. We finally present the main algorithm of the model proposed.

6.1. Numerical boundary parameterization

We will use the Chebyshev points s_c of the second kind for the parameterization variable s, see [49,54]. This allows us to include the end points as collocations points. The grain boundary $\Gamma^{(k)}$ is parameterized by $\boldsymbol{\xi}^{(k)}(s,\,t)$, $0 \leqslant s \leqslant 1$, with n interpolation points. The grain boundary is initially generated by placing $n-2$ equispaced interior points $\mathbf{x}_p^{(k)}(s_p,\,t)$, $p = 2,\,...,n-1$ along the straight line defined by $\boldsymbol{\xi}^{(k)}(0,\,t)$ and $\boldsymbol{\xi}^{(k)}(1,\,t)$. Energy minimization makes the interior points and triple junctions move, so $\boldsymbol{\xi}^{(k)}(s,\,t)$ may no longer be a straight line. It could remain as a straight line if only 2 collocations points are chosen, and in this case the model will become a vertex model [32].

6.2. Numerical evaluation of the unit interpolator and its constrained derivative

In order to obtain the velocities for the interior points and triple junctions, it is necessary to obtain the following expressions: $\mathbf{l}^{(k)}(s,\,t) = \frac{\partial}{\partial s}(\boldsymbol{\xi}^{(k)}(s,\,t))$, $\frac{\partial}{\partial s}(\mathbf{l}^{(k)})$ and $\frac{\partial}{\partial s}(\mathbf{T}^{(k)}) = \frac{\partial}{\partial s}\left(\frac{\mathbf{l}^{(k)}}{\|\mathbf{l}^{(k)}\|}\right)$. We can obtain $\mathbf{l}^{(k)}$ algebraically from the parametric definition of the grain boundary in (4) as:

$$\mathbf{l}^{(k)}\left(s,\,t\right) = \frac{\partial}{\partial s}(\boldsymbol{\xi}^{(k)}(s,\,t)) = \sum_{i=1}^{n} \mathbf{x}_i^{(k)}(t)\,\frac{d}{ds}(\phi_i(s)).$$

This is just taking the derivative of the parameterization of the grain boundary with respect to s. Unfortunately this approach is too cumbersome and it does not take advantage of interpolation at the Chebyshev point, see Appendix A for an extended analysis.

Two algorithms that take advantage of the interpolation at the Chebyshev points are proposed. They approximate $\frac{\partial}{\partial s}\left(\frac{\mathbf{l}^{(k)}}{\|\mathbf{l}^{(k)}\|}\right)$. The naive approach takes the derivative directly at the Chebyshev points of the unitary vector $\frac{\mathbf{l}^{(k)}(s_c,t)}{\|\mathbf{l}^{(k)}(s_c,t)\|}$, see Algorithm 1. Unfortunately, this approach only ensures that the unitary vector is unitary at the collocation points.

Algorithm 1. Naive spectral derivation for unit vector

1: Compute $\mathbf{l}^{(k)}(s_c,\,t) = 2D_{n-1}\boldsymbol{\xi}^{(k)}(s_c,\,t)$

2: Generate $\frac{\mathbf{l}^{(k)}(s_c,t)}{\|\mathbf{l}^{(k)}(s_c,t)\|}$ by dividing each vector in its norm.

3: Compute $\frac{\partial}{\partial s}\left(\frac{\mathbf{l}^{(k)}(s_c,t)}{\|\mathbf{l}^{(k)}(s_c,t)\|}\right) = 2D_{n-1}\frac{\mathbf{l}^{(k)}(s_c,t)}{\|\mathbf{l}^{(k)}(s_c,t)\|}$.

Algorithm 1 adds a significant error in the computation of the derivative of its interpolation in the domain. Thus, to correct this behavior, we propose Algorithm 2.

Algorithm 2. Spectral derivation of interpolation of unit vector

1: Compute $\mathbf{l}^{(k)}(s_c,\,t) = 2D_{n-1}\boldsymbol{\xi}^{(k)}(s_c,\,t)$

2: Compute $\frac{\partial}{\partial s}(\mathbf{l}^{(k)}(s_c,\,t)) = 4D_{n-1}^2\boldsymbol{\xi}^{(k)}\left(s_c,\,t\right)$

3: Compute each component of $\frac{\partial}{\partial s}\left(\frac{\mathbf{l}^{(k)}(s,t)}{\|\mathbf{l}^{(k)}(s,t)\|}\right) = \left\langle \frac{\partial}{\partial s}(\hat{\mathbf{l}}_x^{(k)}\left(s,\,t\right)),\, \frac{\partial}{\partial s}(\hat{\mathbf{l}}_y^{(k)}\left(s,\,t\right)) \right\rangle$ as:

$$\frac{\partial}{\partial s}(\hat{\mathbf{l}}_x^{(k)}\left(s,\,t\right)) = \mathbf{l}_y^{(k)}\left(s,\,t\right)\cdot\frac{\mathbf{l}_y^{(k)}\left(s,t\right)\cdot\frac{\partial}{\partial s}(\mathbf{l}_x^{(k)}\left(s,t\right)) - \mathbf{l}_x^{(k)}\left(s,t\right)\cdot\frac{\partial}{\partial s}(\mathbf{l}_y^{(k)}\left(s,t\right))}{(\mathbf{l}_x^{(k)^2}\left(s,t\right)+\mathbf{l}_y^{(k)^2}\left(s,t\right))^{3/2}}$$

$$\frac{\partial}{\partial s}(\hat{\mathbf{l}}_y^{(k)}\left(s,\,t\right)) = -\mathbf{l}_x^{(k)}\left(s,\,t\right)\cdot\frac{\mathbf{l}_y^{(k)}\left(s,t\right)\cdot\frac{\partial}{\partial s}(\mathbf{l}_x^{(k)}\left(s,t\right)) - \mathbf{l}_x^{(k)}\left(s,t\right)\cdot\frac{\partial}{\partial s}(\mathbf{l}_y^{(k)}\left(s,t\right))}{(\mathbf{l}_x^{(k)^2}\left(s,t\right)+\mathbf{l}_y^{(k)^2}\left(s,t\right))^{3/2}}$$

This new algorithm proposed allows us to get the derivative of the unit vector at Chebyshev points where it is assured that *the interpolation is unitary on the whole domain*. See Section 7.1 for the a numerical comparison of Algorithms 1 and 2.

6.3. Topological changes and collision times

Topological changes in 2D grain growth have been extensively documented and analyzed, see [50,32] to name a few. Thus, we will not discuss them here. However, what will be discussed here is when they need to be applied.

Following the definition of extinction time t_{ext} of [32], which is the time when a boundary $\Gamma^{(k)}$ shrinks until it has arc length equal to 0. We compute the arc length of the k-grain boundary at time t as follows,

$$\mathcal{L}_k(t) = \int_0^1 \left\|\frac{\partial}{\partial s}(\boldsymbol{\xi}^{(k)})\right\| ds = \int_0^1 \left\|\mathbf{l}^{(k)}\right\| ds = \int_0^1 \sqrt{\mathbf{l}_x^{(k)^2} + \mathbf{l}_y^{(k)^2}}\, ds. \tag{14}$$

Thus, the arc length at time $t + \Delta t$ can be approximated as an infinitesimal amount of length being added-to/removed-from $\mathcal{L}_k(t)$,

$$\mathcal{L}_k\left(t+\Delta t\right) \approx \mathcal{L}_k(t) + \Delta t\frac{d\mathcal{L}_k}{dt}(t). \tag{15}$$

To determine if a grain boundary is shrinking or not we need to compute $\frac{d\mathcal{L}_k}{dt}(t)$. If $\frac{d\mathcal{L}_k}{dt}(t) < 0$, it shrinks; otherwise, it does not. Then, taking the derivative of Eq. (14) with respect to time t we obtain:

$$\frac{d\mathcal{L}_k}{dt}(t) = \int_0^1 \frac{\partial}{\partial t}(\|\mathbf{l}^{(k)}\|)ds = \int_0^1 \mathbf{T}^{(k)}.\frac{\partial}{\partial t}\left(\frac{\partial}{\partial s}(\boldsymbol{\xi}^{(k)})\right)ds.$$

Interchanging the spatial with the temporal derivative,

$$= \int_0^1 \mathbf{T}^{(k)}.\frac{\partial}{\partial s}\left(\frac{\partial}{\partial t}(\boldsymbol{\xi}^{(k)})\right)ds = \int_0^1 \mathbf{T}^{(k)}.\frac{\partial}{\partial s}(\mathbf{v}^{(k)})ds, \tag{16}$$

So, if we consider that the grain boundary collapses within $[t,\,t+\Delta t]$, i.e. $\mathcal{L}_k(t + t_{\text{ext}}) = 0$, we can estimate the extinction time as follows:

$$0 \approx \mathcal{L}_k(t) + t_{\text{ext}}\frac{d\mathcal{L}_k}{dt}.$$

Thus, t_{ext} turns out to be:

$$t_{\text{ext}} \approx -\mathcal{L}_k(t)/\frac{d\mathcal{L}_k}{dt}. \tag{17}$$

Therefore, the algorithm to check when a boundary is shrinking and collapsing is the one described in Algorithm 3.

Algorithm 3. Criteria to determine if grain boundary will collapse within time $[t, t + \Delta t]$

1: Compute $t_{\text{ext}} = -\frac{\mathcal{L}_k(t)}{\frac{d\mathcal{L}_k}{dt}}$
2: **if** $t_{\text{ext}} < \Delta t$ **then**
3: Grain boundary will collapse within $[t, t + \Delta t]$
4: **else**
5: Grain boundary will NOT collapse within $[t, t + \Delta t]$
6: **end if**

A critical step for the management of topological transitions in a GPU is the parallel management of them. They become critical due to the fact that we expect to update the data structure in parallel and this could cause race conditions. A race condition occurs when two GPU-threads try to modify the same component of the data structure. For instance, if two different threads try to apply a flipping next to the same triple junction, the data structure could become corrupted. This is avoided by using a polling system. We first decide for each triple junction which neighboring grain boundary has the smallest positive extinction time. This implies that each triple junction will select a grain boundary. Then, we look at each grain boundary's two triple junctions and check if they were selected or not. If a grain boundary receives two votes, it is a candidate for flipping and blocks its neighboring grain boundaries preventing them from flipping. Notice that these neighboring grain boundaries could have received at most 1 vote. The process repeats but without considering boundaries that have been selected as candidate grain boundaries for flipping, or grain boundaries that have been blocked. After a few iterations, the algorithm converges to a fixed point. This fixed point contains the list of candidates grain boundaries selected for safe flipping. Now, the flipping could be performed in parallel and no risk of damaging the data structure is taken. Notice that this is not an issue when one implements topological transitions in a sequential fashion, since there is no risk of damaging the data structure when only one process is updating the data structure at a time. See [55] for more details.

6.4. Numerical integration

Eqs. (8), (9), (16) and $\mathbf{w}_k(t)$ from Eq. (12) must be computed every time step during the numerical simulation. In general, to compute the velocities for interior points and triple junctions given by Eqs. (8), (9) and (12) we require the computation of the following integrals:

$$\int_0^1 \frac{\partial}{\partial s}(\mathbf{T}^{(k)}(s,\,t))\phi_i(s)\,ds, \tag{18}$$

and

$$\int_0^1 \mathbf{T}^{(k)}\Big(s,\,t\Big)\phi_i(s)\,ds \tag{19}$$

where k is the index of the k-th grain boundary, and i is the index of the i-th collocation point in the grain boundary k. Notice that i could be either an interior point or a triple junction.

All integrals can be computed using the Gaussian quadrature [54,49] with Q points. The quadrature weights w_q, as well as the quadrature nodes x_q, are pre-computed at the beginning of the numerical simulation. Thus, (18) and (19) are numerically approximated as:

$$\int_0^1 \frac{\partial}{\partial s}(\mathbf{T}^{(k)}(s,\,t))\phi_i\,ds \approx \sum_{q=0}^{Q} w_q \cdot \frac{\partial}{\partial s}(\mathbf{T}^{(k)}(s,\,t))\cdot\phi_i(s)\,\Big|_{s=x_q},$$

$$\int_0^1 \mathbf{T}^{(k)}\Big(s,\,t\Big)\phi_i\,ds \approx \sum_{q=0}^{Q} w_q \cdot \mathbf{T}^{(k)}(s,\,t)\cdot\phi_i(s)|_{s=x_q},$$

where $\frac{\partial}{\partial s}(\mathbf{T}^{(k)}(s,\,t))$ is computed using Algorithm 2 and $\mathbf{T}^{(k)}(s,\,t)$ using Algorithm 6.

6.5. Main algorithm

The main algorithm is shown in Algorithm 4.

Algorithm 4. Main Algorithm

1: $CurrIter = 0$
2: $G(0) \leftarrow$ Initial grain structure.
3: $(\lambda,\,\mu) \leftarrow$ Set values.
4: **while** End criteria not satisfied **do**
5: $\boldsymbol{\xi}^{(k)}(s)$, $\mathbf{I}^{(k)}(s)$, $\frac{\partial}{\partial s}(\mathbf{I}^{(k)}(s))$, $\frac{\partial}{\partial s}(\mathbf{T}^{(k)}(s))\leftarrow$ Compute these terms using Algorithm 2 at the collocation points and at the quadrature points.
6: Compute average normal component of velocity for interior points.
7: Compute triple junction velocities.
8: Compute tangential component of velocity of interior points.
9: Compute total velocity for interior points.
10: Compute extinction time of each grain boundary with the total velocities.
11: Flip-list $\leftarrow$ Select grain boundaries to be flipped using the Polling System [55].
12: $\widetilde{G} \leftarrow$ Flip grain boundaries in "Flip-list" and/or remove grains if a 3-sided grain needs to flip any of its grain boundaries.
13: $G(t + \Delta t) \leftarrow$ Evolve grain structure $\widetilde{G}$ to next time step with a predictor-corrector scheme [55].
14: **if** $CurrIter \mod r == 0$ **then**
15: Reparameterize boundaries.
16: **end if**
17: $CurrIter \leftarrow CurrIter + 1$
18: **end while**

Algorithm 4 initially required the following parameters for the numerical simulation: the number of interior points n_{int}, Δt_0, λ, and μ. We also apply a reparameterization every r. It is also necessary to precompute D_{n-1}, D_{n-1}^2 and nodes and weights of the Gaussian quadrature. We define $\kappa = \lambda/\mu$ for the study of the effects of the relationship between λ y μ. For the cases when we expect to have a large value of λ we instead consider a small value for μ in order to still capture the effect of the grain boundary energy.

7. Numerical experiments

We performed two types of experiments based on the development of the model. The first set of experiments are related to the analysis of computations that involves the unitary vector $\mathbf{T}^{(k)} = \frac{\mathbf{I}^{(k)}}{\|\mathbf{I}^{(k)}\|}$ and a validation test for a circular grain. The second set of experiments are related to numerical simulations of grain growth varying parameters such as the $\kappa = \lambda/\mu$ ratio and grain boundary energy parameter ε.

7.1. Analysis of unit vector interpolation algorithms

The performance of the algorithms for interpolating unitary parametric curves is analyzed for a test boundary:

$$\boldsymbol{\xi}(s) = \langle (s,\, \exp(s)\cos(s)\sin(5s)\rangle, \quad s \in [0,\,1]$$

Results of interpolation of the unit vector $\frac{\mathbf{I}^{(k)}}{\|\mathbf{I}^{(k)}\|}$ are shown in the first two rows of Fig. 3. The naive Algorithm 5 needs many interpolation points to achieve the behavior of a unitary vector, while Algorithm 6 with a few points achieves great accuracy.

Fig. 3 shows that with $n = 10$ interpolation points, Algorithm 6 proposed makes a much better approximation than the naive algorithm. Even with $n = 50$ interpolation points, the naive algorithm is incapable of achieving a good approximation for an unit vector.

The convergence analysis confirms the observations previously mentioned, see Fig. 2. In the case of $\frac{\mathbf{I}^{(k)}}{\|\mathbf{I}^{(k)}\|}$ the approximation shows a precision accurate to ten digits with $n = 20$ interpolation points for

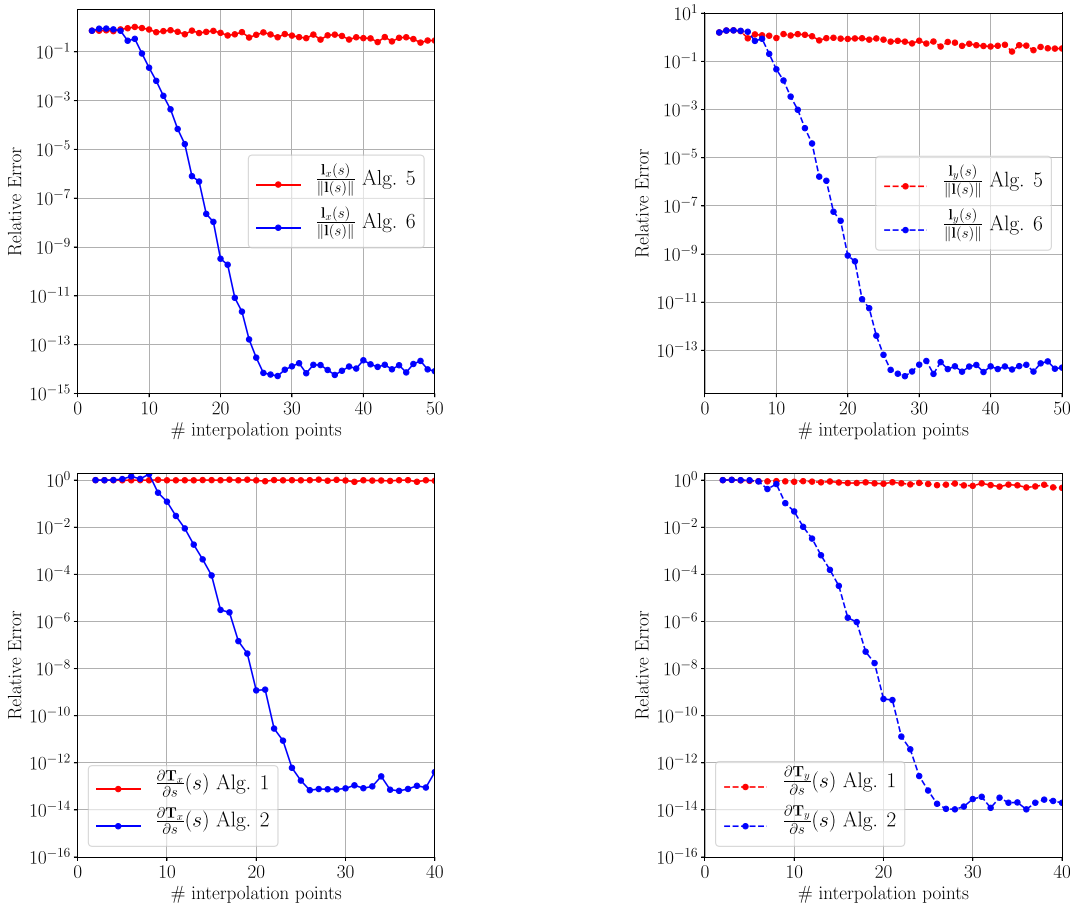


Fig. 2. Convergence analysis for $\mathbf{l}^{(k)}/\|\mathbf{l}^{(k)}\|$ (first row) and $\frac{\partial}{\partial s}(\mathbf{l}^{(k)}/\|\mathbf{l}^{(k)}\|)$ (last row).

Algorithm 6. On the other hand, the naive algorithm cannot decrease the error below of 10^{-1}.

7.2. Analysis of the algorithms computing derivatives of unit vectors

Like we discussed in the previous section, we obtain the same results for the approximation of the derivative of the unit vector. Fig. 2 shows the convergence analysis study of the approximation of $\frac{\mathbf{l}^{(k)}}{\|\mathbf{l}^{(k)}\|}$ and $\frac{\partial}{\partial s}(\mathbf{T}^{(k)})$. The first row of Fig. 2 shows that Algorithm 6 (blue curve) does a much better job than Algorithm 5 (red curve). In the second row of Fig. 2 we observe a similar behavior but for Algorithm 2 (blue curve) and Algorithm 1. It shows that Algorithm 2 handles with great accuracy the derivative of the unit vector with a precision of 10 digits of accuracy with $n = 22$ interpolation points in both components. The error for the naive Algorithm 1 cannot decrease significantly even using $n = 50$ points.

Fig. 3 shows a visual comparison of Algorithms 1, 2, 5, and 6. We observe that even with $n = 10$ interpolation points (middle columns), Algorithms 6 and 2 make again a much better approximation than the naive Algorithms 5 and 1. With $n = 50$ interpolation points the naive algorithm is incapable of achieving an stable approximation.

7.3. Evolving circle - code validation

In this section we discuss the classical test used to validate curvature-based grain growth—the evolution of a circular boundary. In this case the evolution of the boundary is described by a simple initial value problem for the radius $R(t)$ of the circle

$$\frac{d}{dt}R(t) = -\frac{\widetilde{\mu}}{R(t)},$$
$$R(0) = R_0.$$

Here $\widetilde{\mu}$ denotes the grain boundary mobility coefficient. The analytical solution of this problem is readily available and is given by $R(t) = \sqrt{R_0^2 - 2\widetilde{\mu}\, t}$. To mimic evolution of the circle within the current implementation of the code, we constructed the initial grain boundary network as shown in Fig. 4a. The grain boundary network in Fig. 4a satisfy the periodic boundary conditions on $[0, 1]^2$. At the center of the domain, we placed a circular grain with 10 sides, where each side has 2 inner points, respectively. To ensure that the circle shrinks by curvature, we defined the grain boundary energy so that the grain boundaries colored in red have finite grain boundary energy that is bounded away from zero. At the same time, we assume that the energy of the remaining boundaries is infinitesimally small. In order to achieve this effect, we assumed that the red grain has orientation α_0 while all other grains have the same orientation $\alpha_1 \neq \alpha_0$. Thus misorientation is only present for the grain boundaries colored in red. In practice, this means that we define the grain boundary energy for the light blue boundaries as 10^{-6} and for the red boundaries as $1 + 10^{-6}$. The triple junction mobility λ is set equal to 1 (green dots) and the grain boundary mobility μ equal to 10^{-4} (gray dots), so that $\kappa = \lambda/\mu = 10^4$, see Fig. 4b.

Fig. 4c compares the analytical solution $R(t) = \sqrt{R_0^2 - 2\widetilde{\mu}\, t}$ (shown in blue) with the numerically computed radius (shown in red). The comparison was made after the initial transition from the initial polygonal 10-sided shape already took place. We also computed the $\widetilde{\mu}$ from

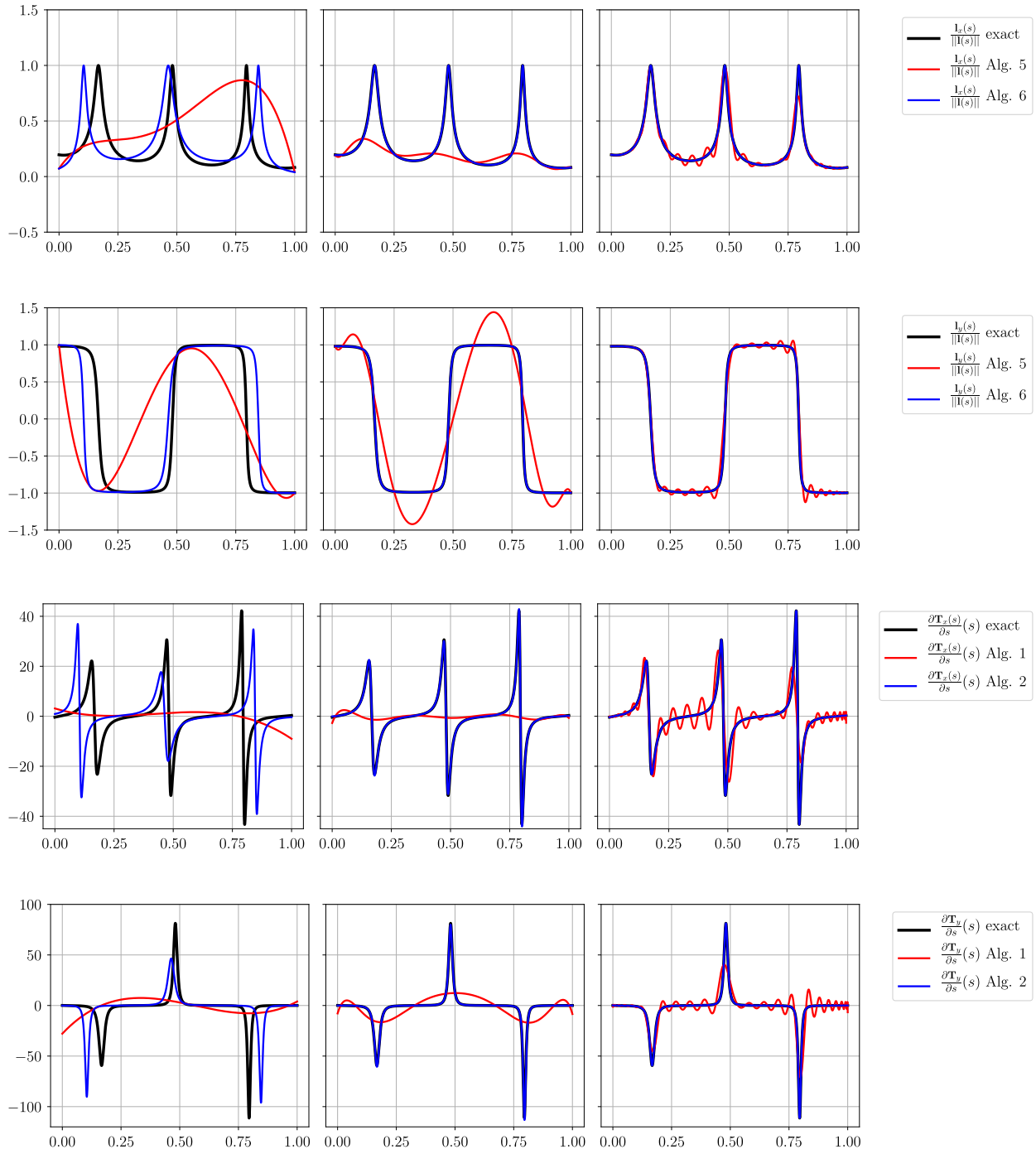
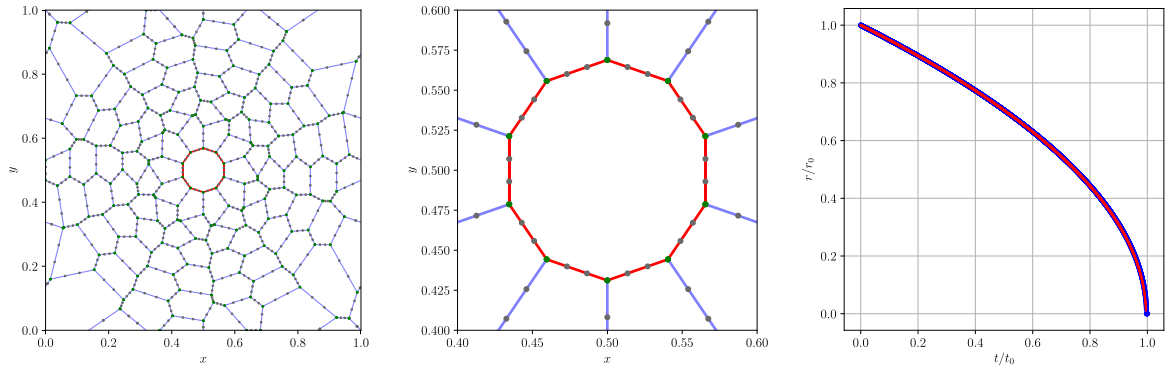


Fig. 3. Behavior for $\mathbf{l}^{(k)}/\|\mathbf{l}^{(k)}\|$ (first two rows) and $\frac{\partial}{\partial s}(\mathbf{T}^{(k)})$ (last two rows) using, from left to right, $n = 5$, 10 and 50 interpolation points, respectively.

the data since in the code we used the finite mobilities $\lambda = 1$ and $\mu = 10^{-4}$. The curves obtained are on top of each other. We conclude that the analytical solution matches the results of the numerical simulation.

7.4. Numerical simulations of the grain boundary network

As discussed before, we aim at large numerical simulation due to the need for significant statistics [46,47,33,48]. To achieve this, we have chosen the use of a GPU hardware architecture. This brings a great computational power but requires to adapt the numerical algorithms. The GPU is used through the CUDA library. The main difference between a GPU implementation and a CPU implementation is the parallel management of operations. In a CPU, without considering the use of multi-threading, only one operation is computed at a time, where in a GPU several operations can be computed in parallel. This shows a clear advantage of doing to whole computation in less time. However, a critical care must be take into account to avoid corrupting the data structure. The issue comes, for instance, when naive implementations allow different thread to modify the same data at the same time, and race-condition may appear. This unfortunately, in our case, may corrupt the grain boundary network. Particularly when topological transitions occurs. On the other hand, we can take full advantage of the GPU when a large and independent computations are performed. For



(a) Initial condition of experimental setup for circular boundary (b) Zoom of initial condition (c) Comparison of numerical solution and algebraic solution

Fig. 4. Circular boundary numerical experiment.

instance, when we compute the velocities of the interior points and triple junctions, or when we compute the tangential components of the inner points. All of these tasks can be computed massively in a GPU, thus the GPU framework suits well for the numerical implementation of this proposed model, see [56].

We performed several numerical simulations of grain growth with initially 100, 000 grains with a Multi-Step Predictor-Corrector Euler method [55]. The Predictor-Corrector algorithm was used due to numerical instabilities right before and after a flipping where the grain boundary is expected to be a straight line. Its used avoided the introduction of any threshold in the code to manage these cases.

We used $n = 2$ interior points per boundary, that is, four collocation points in total. We used the grain boundary energy function $\gamma(\Delta\alpha) = 1 + \frac{\varepsilon}{2}(1 - \cos^3(4\Delta\alpha))$. We computed experiments with the combination of the following parameters: $\kappa \in \{1, 10, 000\}$ and $\varepsilon \in \{0, 0.02, 0.2\}$. More experiments were performed but we only show the limiting cases for κ for sake of clarity. The κ parameter changes the kinetics of the whole system [57]; thus, it is worth studying it and understanding it. This can be seen as having finite mobilities [9]. In the plots shown below we use a continuous line with square (■) for the case when $\kappa = 10, 000$ and transparent continuous lines with diamond markers (♦) for $\kappa = 1$. Fig. 5 shows the grain structures for the two κ's used. Like in Fig. 2, in [28,27], grain boundaries may allow non-convex shapes.

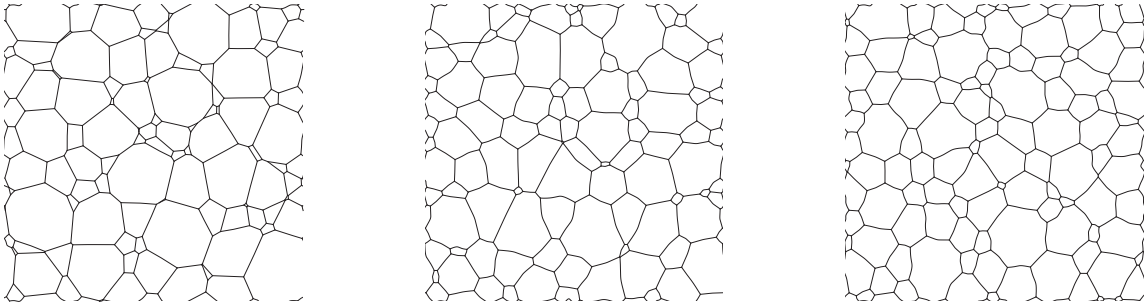
Fig. 6a shows the relative area distributions in logarithmic an linear scales (compare to Fig. 6a from [58] for 2D data). In this case, the square markers were omitted for $\kappa = 10, 000$ for clarity. We clearly observe that the numerical experiment with $\kappa = 10, 000$ has fewer grains with small areas compared to $\kappa = 1$. The different values for the grain boundary energy does not seem to affect the distribution of relatives areas since for both κ's used, the histograms for all the ε look very similar.

Fig. 6b shows the dihedral angle distributions. Markers for $\kappa = 10, 000$ were omitted as well. We see again that the wider distributions belong to the numerical experiment with $\kappa = 1$, whilst the numerical experiment with $\kappa = 10, 000$ tends to accumulate about $2\pi/3$. An interesting feature observed is the case for $\kappa = 10, 000$ and $\varepsilon = 0.2$, which shows a slightly wider distribution compared to the other $\kappa = 10, 000$ numerical experiments.

Fig. 6c shows the average area over time. We observe again a clustering based on κ. $\kappa = 10, 000$ shows a linear tendency over time after an initial transient stage.

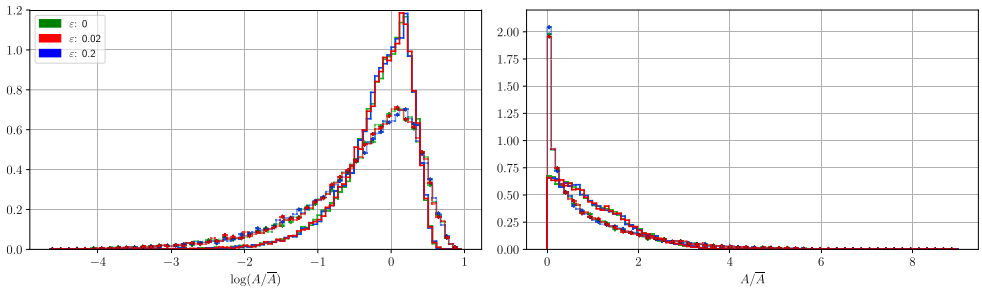
Fig. 6d shows the distribution of the number of sides of grains. The numerical experiments for $\kappa = 10, 000$ have fewer 3-sided grains and the mode is 6-sided grains, where for $\kappa = 1$ the mode of the distribution decreases to 5-sided grains and the number of 3-sided grains increases.

Fig. 6e shows the average number of sides of neighbors per grain class. As we increase the values of κ, the average number of sides of neighbors decreases and we do not see a clear dependence on the grain boundary energy.

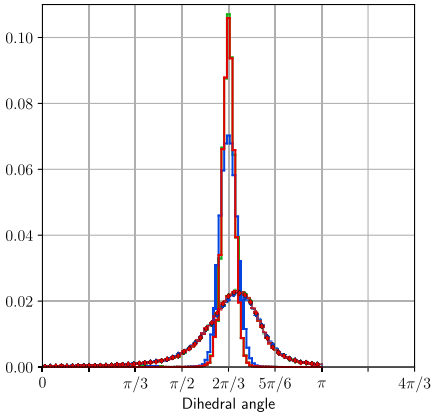


(a) $\kappa = 1$ (b) $\kappa = 100$ (c) $\kappa = 10{,}000$

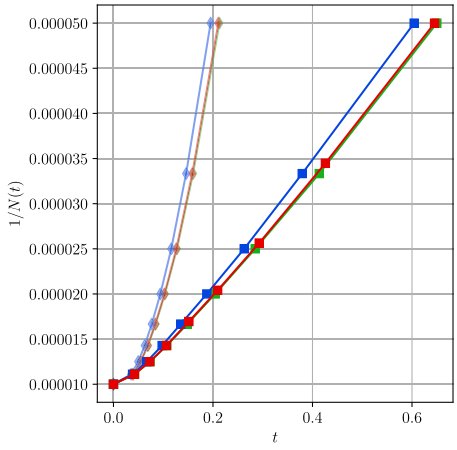
Fig. 5. Grain structures under different parameters from 1000 initial grains.



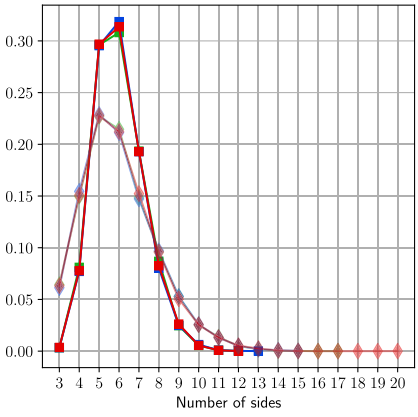
(a) Distribution of relative areas in log scale and linear scale.



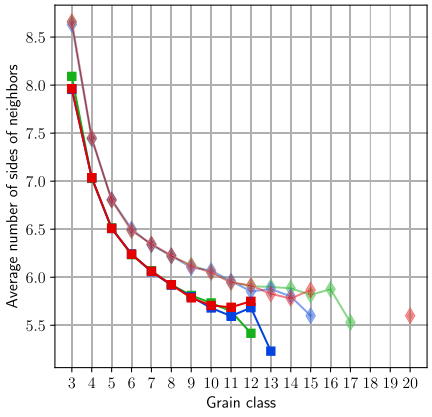
(b) Distribution of dihedral angle.



(c) Average area over time.



(d) Histogram of grains per number of sides.



(e) Average number of sides of neighbors.

Fig. 6. Statistics for $\kappa = 1$ (diamonds ◆ over solid line) and $\kappa = 10,000$ (squares ■ over lighter solid line).

Figs. 7 and 8 show the mean and median of the rate of change of area per class scaled by μ^{-1}, respectively. This scaling is applied to observe the concordance with the Von Neumann-Mullins $n - 6$ relation [59], i.e. $\frac{dA}{dt}(\Sigma) \sim \mu(\text{ns}(\Sigma) - 6)$, where $\text{ns}(\Sigma)$ is the number of sides of grain Σ. The plots on the left resembles a vertex code for $\kappa = 1$ and the plot on the right resembles and curvature-based code for $\kappa = 10,000$; thus, the linear relation is observed. We plot them independently since their range of scales differs considerably. This results is similar to the one shown in Fig. 14 of [12].

Fig. 9 shows the Grain Boundary Character Distributions (GBCD), see [60], for $\varepsilon = 0.02$ and $\varepsilon = 0.2$, respectively. In these cases, the κ parameters do not seem to have an effect, so we only show the GBCD for $\kappa = 10,000$. It is clear that the GBCD shows the expected behaviour in relation to the grain boundary energy function used, i.e. arc lengths are accumulated for misorientations near 0.

Fig. 10 shows the Misorientation Distribution Function (MDF) for $\varepsilon = 0.02$ (first row) and $\varepsilon = 0.2$ (second row). Similarly to the case of the GBCD, the MDF shows a more prominent behavior as we increase ε and κ does not seem to have and effect. Thus, only the output for $\kappa = 10,000$ is shown.

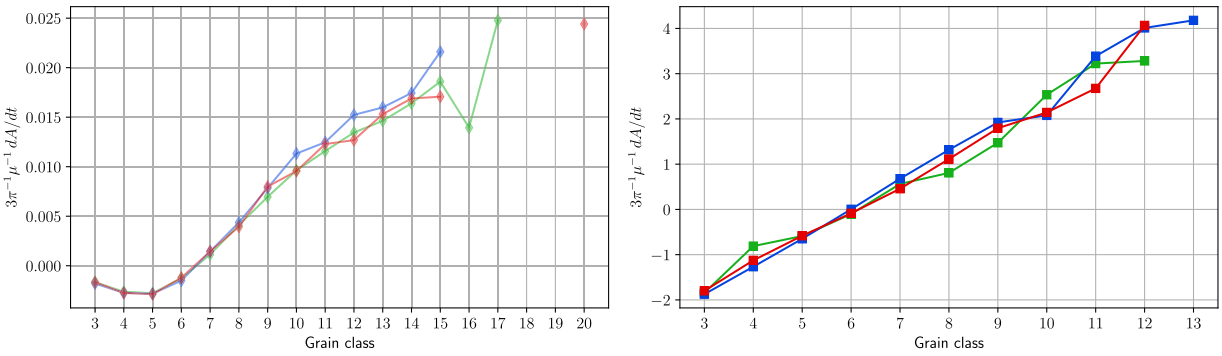


Fig. 7. Rate of change of area per grain class. (Left) Mean of dA/dt for $\kappa = 1$. (Right) Mean dA/dt for $\kappa = 10,000$. This was computed by removing less than 10% of extreme values per class.

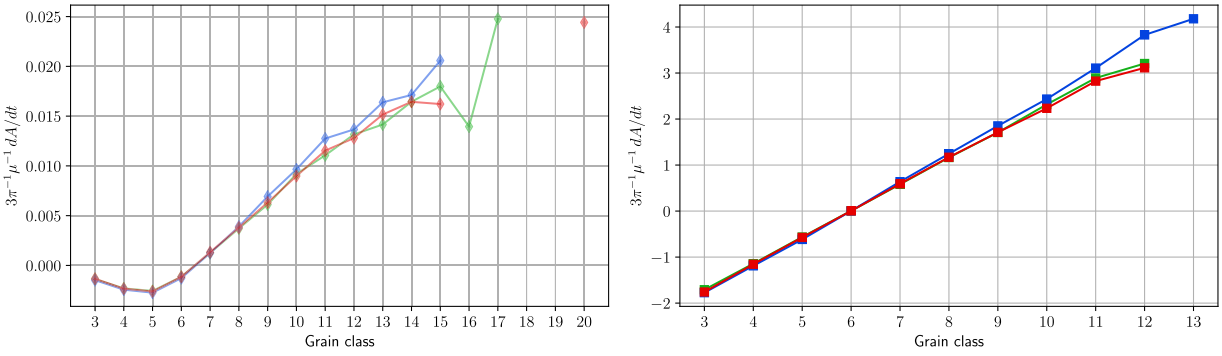


Fig. 8. Rate of change of area per grain class. (Left) Median of dA/dt for $\kappa = 1$. (Right) Median dA/dt for $\kappa = 10,000$.

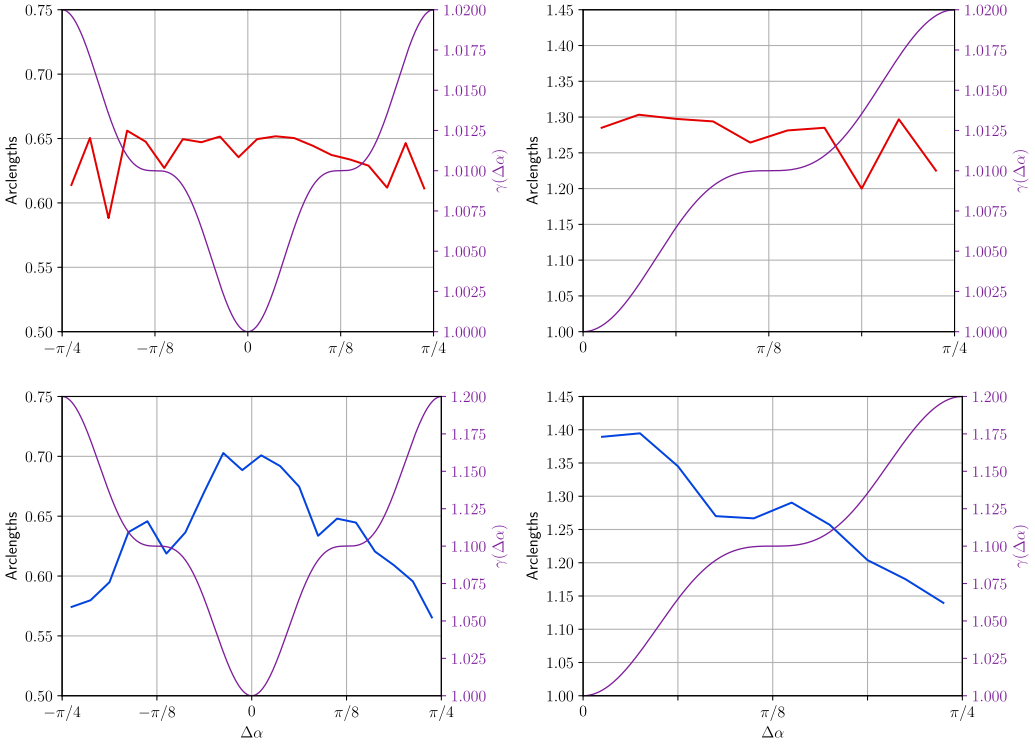


Fig. 9. The Grain Boundary Character Distributions (GBCD) for $\varepsilon = 0.02$ (first row) and for $\varepsilon = 0.2$ (second row). The right column accumulates the misorientations for symmetry on $[0, \pi/4]$.

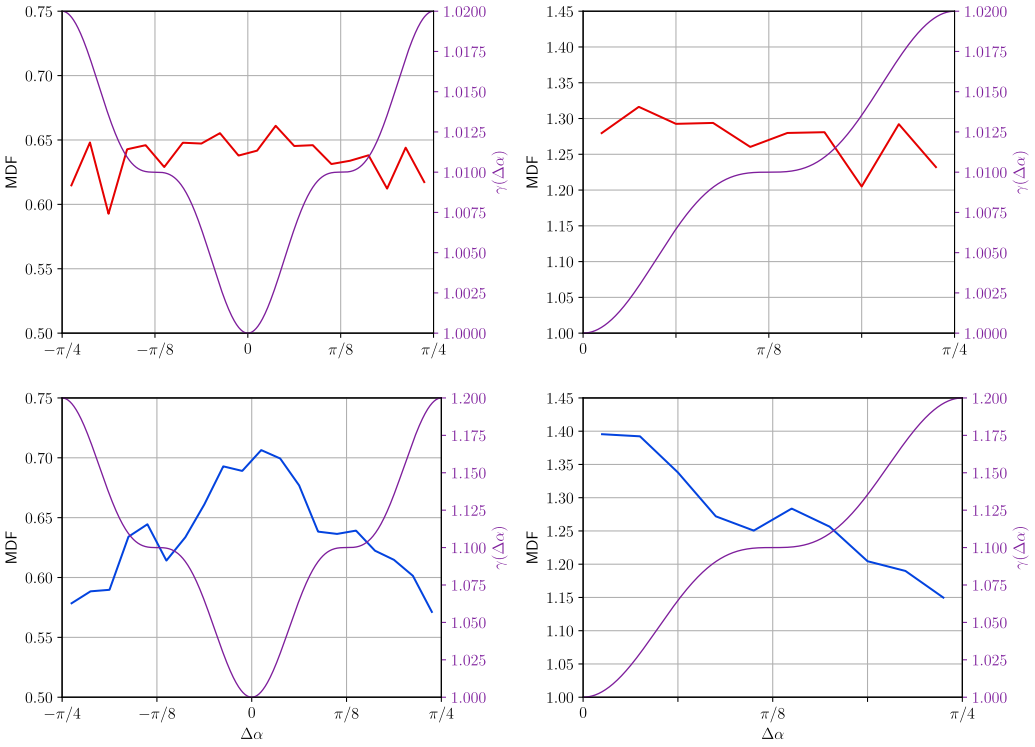


Fig. 10. The Misorientation Distribution Function (MDF) for $\varepsilon = 0.02$ (first row) and for $\varepsilon = 0.2$ (second row). The right column accumulates the misorientations for symmetry on $[0, \pi/4]$.

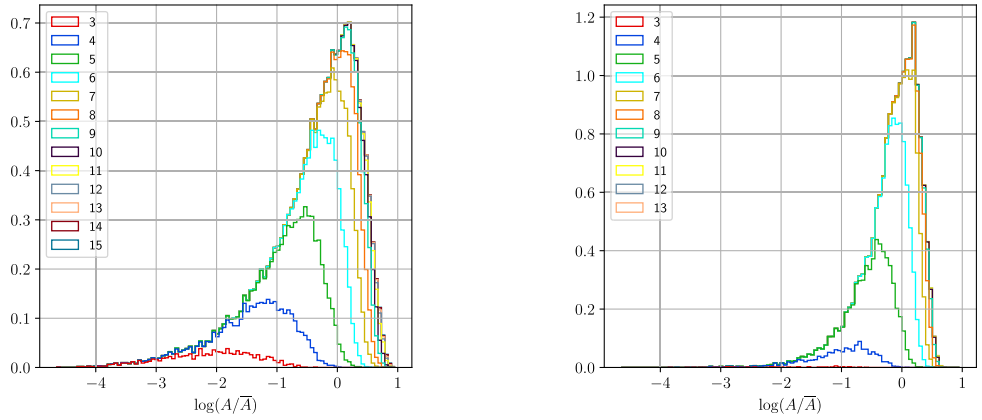


Fig. 11. Relative grain area distributions showing each class independently. The figure on the left corresponds to the numerical experiment performed with $\kappa = 1$ and the figure on the right corresponds to $\kappa = 10,000$. In both cases we used $\varepsilon = 0.2$ for the grain boundary energy. For the cases with $\varepsilon = 0$ and $\varepsilon = 0.02$ the figures look alike so they were not included.

A visual analysis of the relationship between Fig. 6a and d may lead to the conclusion that grains with small areas will be 3 and 4-sided grains for $\kappa = 1$. To answer this question, we constructed Fig. 11. We show each class with a different color, for instance, from the x-axis to the red curve we have the proportion of 3-sided grains for each of the covered bins. From the red curve to the blue curve, we observe the proportion of 4-sided grains, and so on. This is the same for all the six experiments. Thus, for $\kappa = 1$, we can conclude that the long tail to the left belongs to 3-sided grains and many 4-sided grains. On the other side, we see that few 4-sided grains exist for $\kappa = 10,000$ and very few 3-sided grains. On the contrary, more 5-sided grains and 6-sided grains can be observed.

Fig. 12 shows the relative area distribution over time. The colorbar shows the percentage of grain removed for each color, which means that the blue curves were obtained when 20% of the grains were removed. They clearly show the stationary evolution for all experiments after 50% of the grains were removed. Fig. 13 shows the evolution of the relative area over time as a heatmap in a logarithmic scale, where

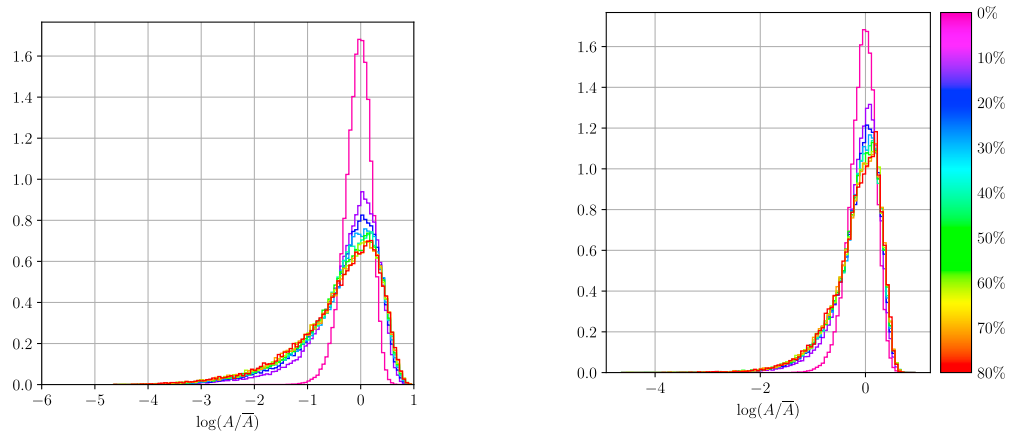


Fig. 12. Evolution over time of relative grain area distribution. The figure on the left corresponds to the numerical experiment performed with $\kappa = 1$ and the figure on the right corresponds to $\kappa = 10,000$. In both cases we used $\varepsilon = 0.2$ for the grain boundary energy. For the cases with $\varepsilon = 0$ and $\varepsilon = 0.02$ the figures look alike so they were not included. The colorbar represents the percentage of grains removed.

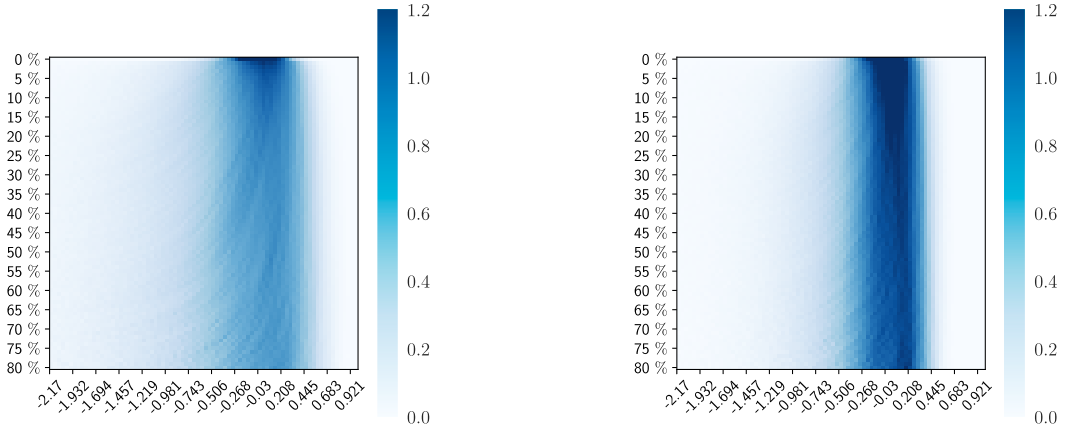


Fig. 13. Evolution over time of relative grain area distribution. The figure on the left corresponds to the numerical experiment performed with $\kappa = 1$ and the figure on the right corresponds to $\kappa = 10,000$. In both cases we used $\varepsilon = 0.2$ for the grain boundary energy. For the cases with $\varepsilon = 0$ and $\varepsilon = 0.02$ the figures look alike so they were not included.

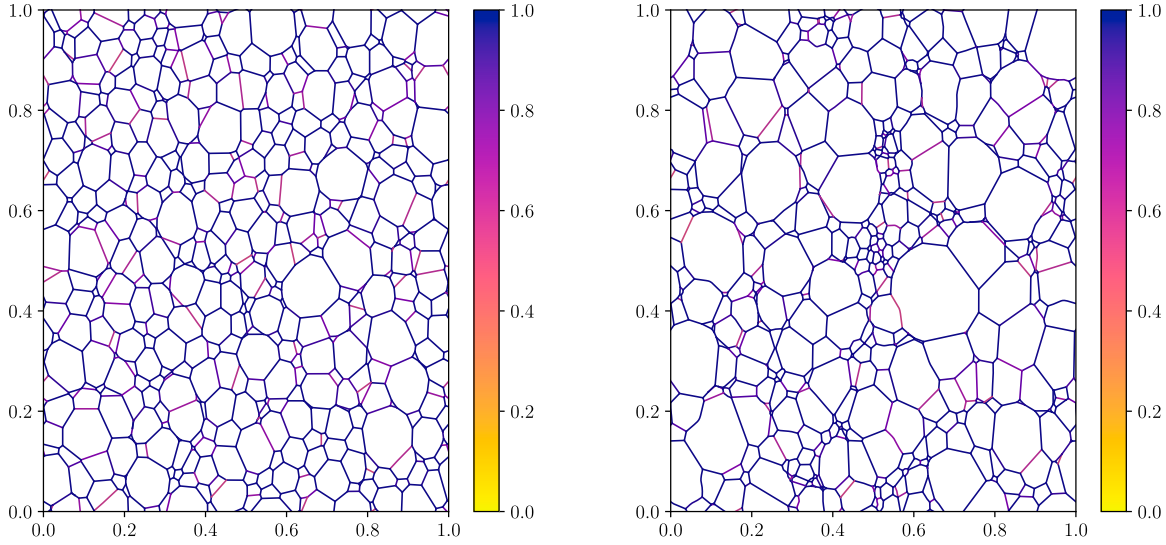
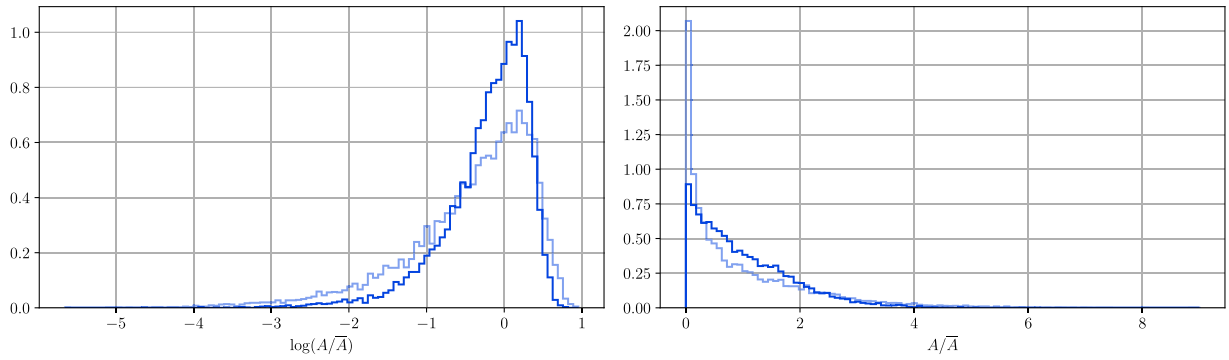
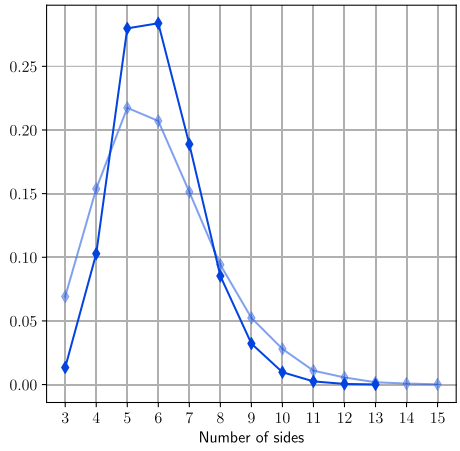


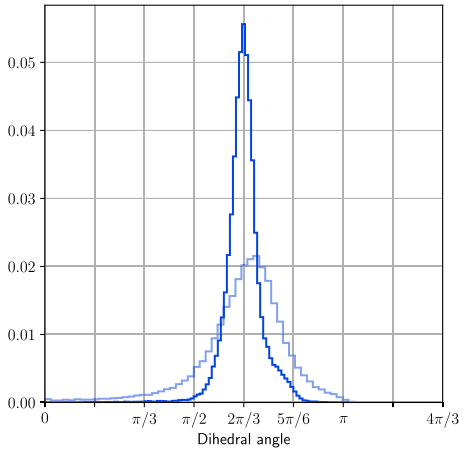
Fig. 14. Grain boundary network for $\kappa = 1$ (left) and $\kappa = 10,000$ (right) for Read-Shockley with $\delta = 0.5$. The colorbar show the corresponding value of the grain boundary energy for each boundary.



(a) Distribution of relative areas in log scale and linear scale.



(b) Histogram of grains per number of sides.



(c) Dihedral angle.

Fig. 15. Statistics for the scaled Read-Shockley grain boundary energy with $\delta = 0.5$. Light-blue lines correspond to $\kappa = 1$ and blue lines correspond to $\kappa = 1000$.

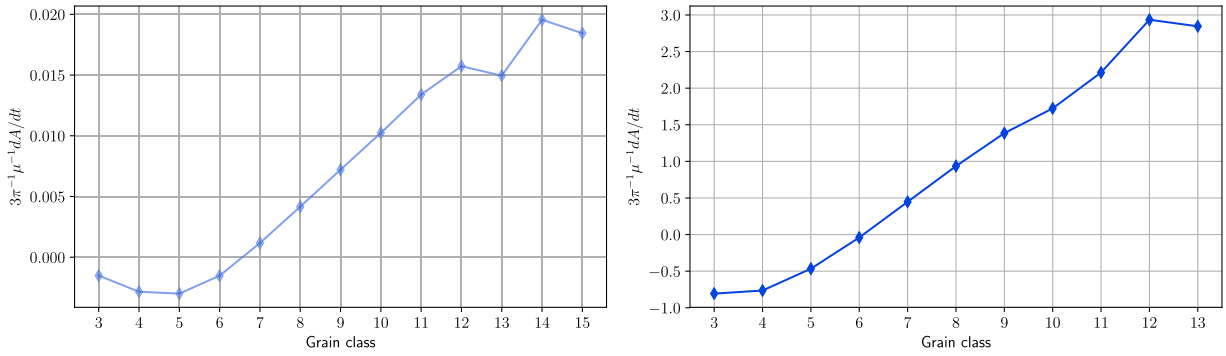


Fig. 16. Rate of change of area per grain class. (Left) Mean of dA/dt for $\kappa = 1$. (Right) Mean dA/dt for $\kappa = 1000$. This was computed by removing less than 10% of extreme values per class. This uses the Read-Shockley grain boundary energy.

darker bins show a higher population. This was constructed every time 1% of the population was removed. The evolution over time is shown on the y-axis, and for the bins we used the range $[-2.17, 1]$ for visualization purposes. Again, after 50% of the grains have been removed, we observe the stationary evolution.

7.5. Numerical simulation with Read-Shockley type grain boundary energy function

Another type of the grain boundary energy function is the Read-Shockley energy (see Eq. (34) in [42] and references therein). Here we

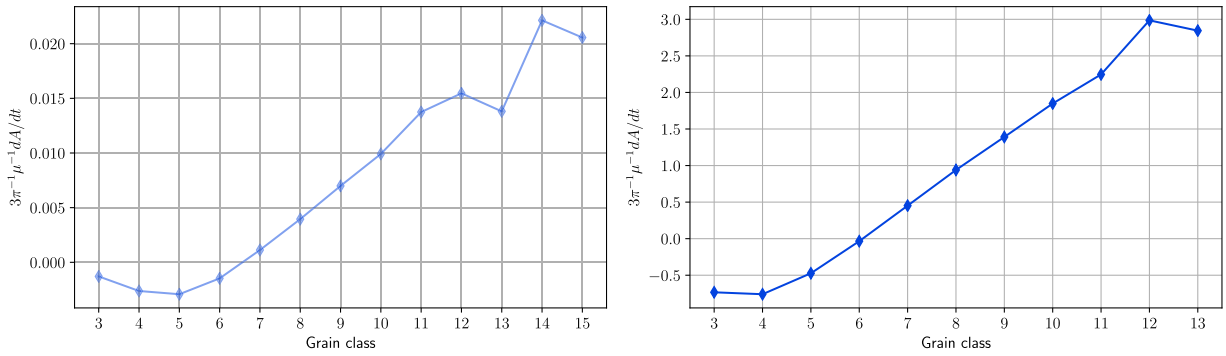


Fig. 17. Rate of change of area per grain class. (Left) Median of dA/dt for $\kappa = 1$. (Right) Median dA/dt for $\kappa = 1000$. This uses the Read-Shockley grain boundary energy.

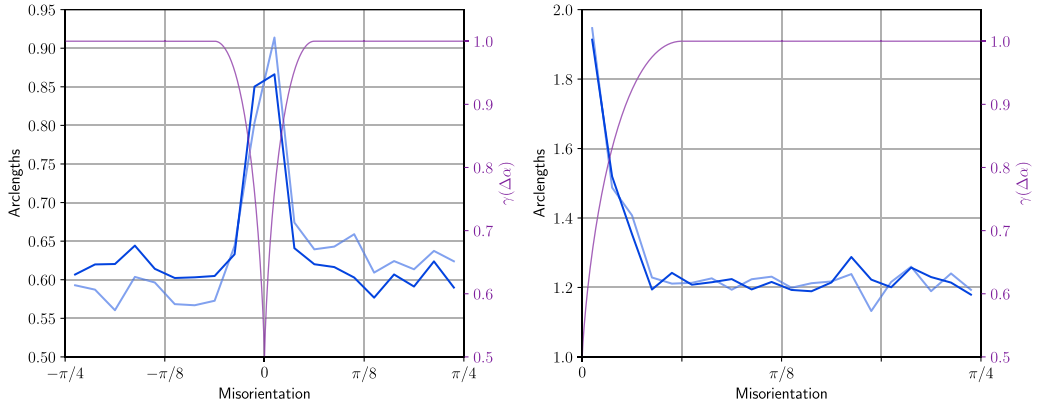


Fig. 18. The Grain Boundary Character Distributions (GBCD) for the Read-Shockley grain boundary energy. The right column accumulates the misorientations for symmetry on $[0, \pi/4]$. Light-blue lines correspond to $\kappa = 1$ and blue lines correspond to $\kappa = 1000$.

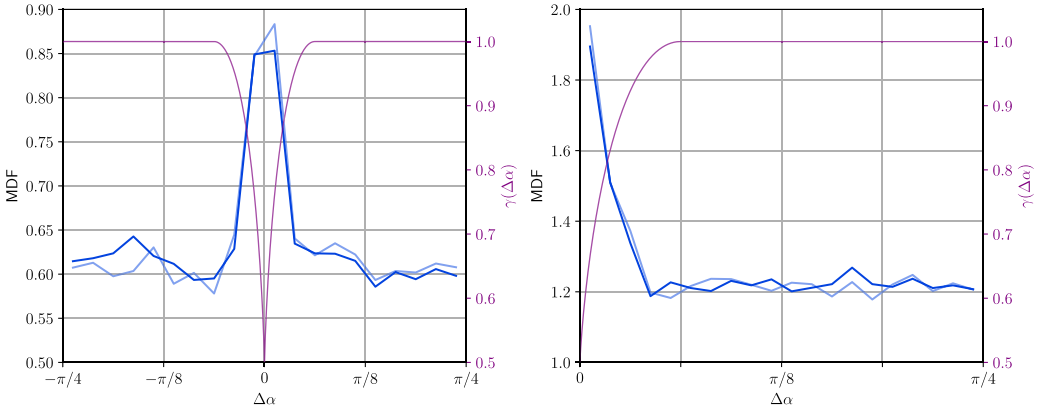


Fig. 19. The Misorientation Distribution Function (MDF) for the Read-Shockley grain boundary energy. The right column accumulates the misorientations for symmetry on $[0, \pi/4]$. Light-blue lines correspond to $\kappa = 1$ and blue lines correspond to $\kappa = 1000$.

use a scaled Read-Shockley in order to avoid large discrepancies of grain boundary energies that may lead to singular behavior of the network. We have the following expression for Read-Shockley energy

$$\gamma\left(\Delta\alpha\right) = \begin{cases} \left(1-\delta\right)\frac{|\Delta\alpha|}{\theta}\left(1-\log\left(\frac{|\Delta\alpha|}{\theta}\right)\right)+\delta, & \text{if } |\Delta\alpha| \leqslant \theta, \\ 1, & \text{otherwise,} \end{cases}$$

where $\theta = \frac{\pi}{16}$ and $\delta = 0.5$. This function has its minimum δ at $\Delta\alpha = 0$. In Fig. 14 we show two grain boundary networks obtained for the Read-Shockley grain boundary energy function when $\kappa = 1$ and $\kappa = 10,000$, respectively. The colorbar shows the grain boundary energy value for each boundary.

We present the same numerical experiments for the Read-Shockley as those described in Section 7.4, light-blue lines correspond to $\kappa = 1$ and blue lines correspond to $\kappa = 1000$. We omit the description of the

plots Figs. 15–19 as they are the same as in Section 7.4.

8. Conclusions

In this paper we have derived and implemented in CUDA a new model for grain growth in 2D, which reproduces a vertex-based model and curvature-based model statistics depending on the parameters used. The main reason to implement it in CUDA is due to the integral formulation of the evolution equations.

We proposed a novel way to compute the derivative of a unitary vector from discrete data points. This is done through the use of a spectral method with Chebyshev points in a coupled fashion. We showed that taking the derivative without considering the constraint of being a unitary vector produces poor results. The proposed coupled algorithms successfully solve this numerical challenge. This was needed to build a numerical implementation of the coupled model.

A numerical simulation was run when 80% of initial grains were removed and we validated that the stationary relative area distribution is obtained after 50% of the grains are removed. We were able to reproduce similar statistics one can get from a vertex-based model when $\varkappa = 1$, and approach the statistics of a curvature-based model as we increase the value of $\varkappa$ until 10,000. We confirmed that vertex-based models have a long tail due to the presence of 3-sided and 4-sided grains, compared to curvature-based models. We were also able to successfully obtain the GBCD and reproduce well-known results in relation to the inverse correlation with the grain boundary energy $\gamma(\Delta\alpha)$. Similar statistics were also obtained for the scaled Read-Shockley grain boundary energy function.

Future work may include adding a stored energy term to study the effect of recrystallization within the framework considered in this work.

Data availability

The raw/processed data required to reproduce these findings cannot be shared at this time due to technical or time limitations.

CRediT authorship contribution statement

Alejandro H.J. Sazo: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing - original draft, Writing - review & editing, Visualization. **Pablo Ibarra S.:** Software, Validation, Formal analysis, Investigation, Data curation, Writing - review & editing, Visualization. **Ariel Sanhueza R.:** Software, Validation, Formal analysis, Investigation, Data curation, Writing - review & editing, Visualization. **Francisco J.A. Casas:** Software, Validation, Formal analysis, Investigation, Writing - review & editing, Visualization. **Claudio E. Torres:** Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Writing - original draft, Writing - review & editing, Supervision, Project administration, Funding acquisition. **Maria Emelianenko:** Conceptualization, Formal analysis, Investigation, Resources, Writing - review & editing, Supervision. **Dmitry Golovaty:** Conceptualization, Formal analysis, Investigation, Resources, Writing - review & editing, Supervision.

Acknowledgements

This work has been partially funded by - CCTVal, CONICYT PIA/Basal FB0821, and FONDECYT 11160744. ME acknowledges support provided by the US National Science Foundation CAREER grant DMS-1056821.

Appendix A. Computation of the constrained derivative

We briefly discuss in Section 6.2 Algorithms 1 and 2, for the computation of the constrained derivative. As mentioned before, to obtain $\mathbf{l}^{(k)}$ we can explicitly compute the derivative with respect to s of the parametric definition of the grain boundary, see (4), as:

$$\mathbf{l}^{(k)}\left(s,\, t\right) = \frac{\partial}{\partial s}(\boldsymbol{\xi}^{(k)}(s,\, t)) = \sum_{i=1}^{n} \mathbf{x}_i^{(k)}(t)\, \frac{d}{ds}(\phi_i(s)).$$

Notice that the Lagrange polynomial $\phi_i(s)$ is decoupled from the boundary data $\mathbf{x}_i^{(k)}(t)$. The derivative of $\phi_i(s)$ is obtained as follows,

$$\frac{d}{ds}(\phi_i(s)) = \frac{1}{l_i(s_i)}\frac{d}{ds}(l_i(s)), \tag{A.1}$$

which implies that the derivative of $l_i(s)$ must be obtained:

$$\frac{d}{ds}(l_i(s)) = \frac{d}{ds}\left(\prod_{\substack{k=1\\k\neq i}}^{n}\left(s - s_k\right)\right) = \sum_{\substack{k=1\\k\neq i}}^{n}\prod_{\substack{j=1\\j\neq i\\j\neq k}}^{n}\left(s - s_j\right).$$

Replacing in (A.1) we obtain:

$$\frac{d}{ds}(\phi_i(s)) = \frac{1}{l_i(s_i)}\sum_{\substack{k=1\\k\neq i}}^{n}\prod_{\substack{j=1\\j\neq i\\j\neq k}}^{n}\left(s - s_j\right). \tag{A.2}$$

This shows that evaluating (A.2) requires $O(n^2)$ floating point operations and thus, the overall computational cost of computing $\mathbf{l}^{(k)}$ requires $O(n^3)$ floating point operations for each s.

We also need to compute $\frac{\partial}{\partial s}(\mathbf{l}^{(k)})$, taking the derivative with respect to s of (4) twice we obtain:

$$\frac{\partial}{\partial s}(\mathbf{l}^{(k)}(s,\, t)) = \sum_{i=0}^{n} \mathbf{x}_i^{(k)}(t)\, \frac{d^2}{ds^2}(\phi_i(s)).$$

Now the second derivative of $\phi_i(s)$ is needed, which can be computed by taking the derivative with respect to s of (A.2),

$$\frac{d^2}{ds^2}(l_i(s)) = \sum_{\substack{k=1 \\ k\neq i}}^{n} \sum_{\substack{j=1 \\ j\neq i \\ j\neq k}}^{n} \frac{l_i(s)}{(s-s_k)(s-s_j)}.$$

Thus,

$$\frac{d^2}{ds^2}(\phi_i(s)) = \phi_i(s) \sum_{\substack{k=1 \\ k\neq i}}^{n} \sum_{\substack{j=1 \\ j\neq i \\ j\neq k}}^{n} \frac{1}{(s-s_k)(s-s_j)}$$

A new sum appeared while taking the derivative. The floating point operations needed to evaluate $\frac{\partial}{\partial s}(\mathbf{I}^{(k)})$ are $O(n^3)$. This analysis shows that it is desirable to avoid the evaluation of these polynomials. An alternative approach is to take advantage of having the data at Chebyshev points. This means we can use Spectral Methods [49] to approximate the derivative numerically. The derivative of $\boldsymbol{\xi}^{(k)}(s,\, t)$ at the Chebyshev points is obtained by computing the product between the differentiation matrix D_{n-1} and the vector $\mathbf{x}^{(k)}(t)$, where the data points are evaluated at the Chebyshev points. Notice that this matrix is multiplied by 2 since the domain is mapped from $[-1,\, 1]$ to $[0,\, 1]$. The second derivative is computed by multiplying the same data by D_{n-1}^2, where both differentiation matrices can be pre-computed. This change of approach produces an algorithm $O(n^2)$, since it is just a matrix-vector multiplication and the data is needed at the collocation points. The only $O(n^3)$ component is the pre-computation of D_{n-1}^2, but since it is done once we disregard its cost. Therefore $\mathbf{I}^{(k)}$ and $\frac{\partial}{\partial s}(\mathbf{I}^{(k)})$ are now computed as:

$$\mathbf{I}^{(k)}(s_c,\, t) = D_{n-1}\boldsymbol{\xi}^{(k)}(s_c,\, t)$$
$$\frac{\partial}{\partial s}(\mathbf{I}^{(k)}(s_c,\, t)) = D_{n-1}^2\boldsymbol{\xi}^{(k)}(s_c,\, t)$$

We still need to compute the constrained derivative $\frac{\partial}{\partial s}(\mathbf{T}^{(k)})$. This means we need to compute the derivative of the unit vector $\frac{\mathbf{I}^{(k)}}{\|\mathbf{I}^{(k)}\|}$.

Two algorithms are proposed to compute $\frac{\partial}{\partial s}\left(\frac{\mathbf{I}^{(k)}}{\|\mathbf{I}^{(k)}\|}\right)$. The naive approach takes the derivative directly at the Chebyshev points of the unitary vector $\frac{\mathbf{I}^{(k)}(s_c,t)}{\|\mathbf{I}^{(k)}(s_c,t)\|}$, see Algorithm 1. Unfortunately, this approach only ensures that the unitary vector is unitary at the collocation points.

Algorithm 1. Naive spectral derivation for unit vector (uncoupled)

1: Compute $\mathbf{I}^{(k)}(s_c,\, t) = 2D_{n-1}\boldsymbol{\xi}^{(k)}(s_c,\, t)$

2: Generate $\frac{\mathbf{I}^{(k)}(s_c,t)}{\|\mathbf{I}^{(k)}(s_c,t)\|}$ by dividing each vector in its norm.

3: Compute $\frac{\partial}{\partial s}\left(\frac{\mathbf{I}^{(k)}(s_c,t)}{\|\mathbf{I}^{(k)}(s_c,t)\|}\right) = 2D_{n-1}\frac{\mathbf{I}^{(k)}(s_c,t)}{\|\mathbf{I}^{(k)}(s_c,t)\|}$.

This adds a large error in the computation of the derivative of its interpolation. Thus, to correct this behavior, we propose Algorithm 2.

Algorithm 2. Spectral derivation with coupled interpolation of unit vector

1: Compute $\mathbf{I}^{(k)}(s_c,\, t) = 2D_{n-1}\boldsymbol{\xi}^{(k)}(s_c,\, t)$

2: Compute $\frac{\partial}{\partial s}(\mathbf{I}^{(k)}(s_c,\, t)) = 4D_{n-1}^2\boldsymbol{\xi}^{(k)}\left(s_c,\, t\right)$

3: Compute each component of $\frac{\partial}{\partial s}\left(\frac{\mathbf{I}^{(k)}(s,t)}{\|\mathbf{I}^{(k)}(s,t)\|}\right)$ as:

$$\frac{\partial}{\partial s}(\hat{\mathbf{I}}_x^{(k)}\left(s,\, t\right)) = \mathbf{I}_y^{(k)}\left(s,\, t\right)\cdot\frac{\mathbf{I}_y^{(k)}\left(s,t\right)\cdot\frac{\partial}{\partial s}(\mathbf{I}_x^{(k)}\left(s,t\right)) - \mathbf{I}_x^{(k)}\left(s,t\right)\cdot\frac{\partial}{\partial s}(\mathbf{I}_y^{(k)}\left(s,t\right))}{(\mathbf{I}_x^{(k)^2}\left(s,t\right) + \mathbf{I}_y^{(k)^2}\left(s,t\right))^{3/2}}$$

$$\frac{\partial}{\partial s}(\hat{\mathbf{I}}_y^{(k)}\left(s,\, t\right)) = -\mathbf{I}_x^{(k)}\left(s,\, t\right)\cdot\frac{\mathbf{I}_y^{(k)}\left(s,t\right)\cdot\frac{\partial}{\partial s}(\mathbf{I}_x^{(k)}\left(s,t\right)) - \mathbf{I}_x^{(k)}\left(s,t\right)\cdot\frac{\partial}{\partial s}(\mathbf{I}_y^{(k)}\left(s,t\right))}{(\mathbf{I}_x^{(k)^2}\left(s,t\right) + \mathbf{I}_y^{(k)^2}\left(s,t\right))^{3/2}}$$

This new algorithm proposed allows us to get the derivative of the unit vector at Chebyshev points where it is assured that *the interpolation is unitary on the whole domain*. See Section 7.1 for a numerical comparison of Algorithms 1 and 2.

Another important algorithm needed is the interpolation of the unitary vector valued function $\mathbf{T}^{(k)} = \frac{\mathbf{I}^{(k)}}{\|\mathbf{I}^{(k)}\|}$. Again, we present 2 cases, the naive version and the coupled version. The naive version is presented in Algorithm 5 and the coupled version is presented in Algorithm 6.

Algorithm 5. Naive (uncoupled) interpolation of $\mathbf{T}^{(k)}$

1: Compute $\mathbf{l}^{(k)}(s_c, t) = 2D_{n-1}\boldsymbol{\xi}^{(k)}(s_c, t)$
2: $\widehat{\mathbf{T}_x^{(k)}}\left(s, t\right) \leftarrow$ Interpolate $\mathbf{l}_x^{(k)}(s, t)/\|\mathbf{l}^{(k)}(s, t)\|$
3: $\widehat{\mathbf{T}_y^{(k)}}\left(s, t\right) \leftarrow$ Interpolate $\mathbf{l}_y^{(k)}(s, t)/\|\mathbf{l}^{(k)}(s, t)\|$
4: Compute $\mathbf{T}^{(k)}\left(s, t\right) = \langle \widehat{\mathbf{T}_x^{(k)}}\left(s, t\right), \widehat{\mathbf{T}_y^{(k)}}\left(s, t\right)\rangle$

Algorithm 6. Spectral (coupled) interpolation of $\mathbf{T}^{(k)}$

1: Compute $\mathbf{l}^{(k)}(s_c, t) = 2D_{n-1}\boldsymbol{\xi}^{(k)}(s_c, t)$
2: $\widehat{\mathbf{l}^{(k)}}\left(s, t\right) \leftarrow$Interpolate $\mathbf{l}^{(k)}(s, t)$
3: Compute $\mathbf{T}^{(k)}\left(s, t\right) = \dfrac{\widehat{\mathbf{l}^{(k)}}\left(s,t\right)}{\left\|\widehat{\mathbf{l}^{(k)}}\left(s,t\right)\right\|}$

Therefore, we have all the components we need to implement the coupled model.

Appendix B. Supplementary material

Supplementary data associated with this article can be found, in the online version, at https://doi.org/10.1016/j.commatsci.2019.01.022.

References

[1] K. Barmak, E. Eggeling, D. Kinderlehrer, R. Sharp, S. Ta'asan, A.D. Rollett, K.R. Coffey, Grain growth and the puzzle of its stagnation in thin films: the curious tale of a tail and an ear, Prog. Mater Sci. 58 (2013) 987–1055.
[2] J.E. Darnbrough, P.E.J. Flewitt, Growth of abnormal planar faceted grains in nanocrystalline nickel containing impurity sulphur, Acta Mater. 79 (2014) 421–433.
[3] J.S. Shin, S.H. Ko, K.T. Kim, Development and characterization of low-silicon cast aluminum alloys for thermal dissipation, J. Alloy. Compd. 644 (2015) 673–686.
[4] J.E. Darnbrough, F. Christien, P.E.J. Flewitt, Kinetics and dynamics of planar abnormal grain growth in nanocrystalline nickel, Acta Mater. 141 (2017) 67–74.
[5] D. Zöllner, A Potts model for junction limited grain growth, Comput. Mater. Sci. 50 (2011) 2712–2719.
[6] D. Zöllner, Grain microstructural evolution in 2D and 3D polycrystals under triple junction energy and mobility control, Comput. Mater. Sci. 118 (2016) 325–337.
[7] K. Ito, Two-dimensional simulation of the effect of the migration of triple junctions on crystallographic texture evolution through grain coarsening, Comput. Mater. Sci. 62 (2012) 117–125.
[8] P. Streitenberger, D. Zöllner, Evolution equations and size distributions in nanocrystalline grain growth, Acta Mater. 59 (2011) 4235–4243.
[9] P. Streitenberger, D. Zöllner, Triple junction controlled grain growth in two-dimensional polycrystals and thin films: self-similar growth laws and grain size distributions, Acta Mater. 78 (2014) 114–124.
[10] D. Zöllner, A phenomenological approach to investigate nanocrystalline grain growth, Comput. Mater. Sci. 92 (2014) 114–119.
[11] D. Zöllner, P. Streitenberger, Studying the influence of triple junction energy and mobility on annealing processes, IOP Conf. Ser.: Mater. Sci. Eng. 89 (2015) 012061.
[12] D. Zöllner, Treating grain growth in thin films in three dimensions: a simulation study, Comput. Mater. Sci. 125 (2016) 51–60.
[13] A.J. Haslam, S.R. Phillpot, D. Wolf, Mechanisms of grain growth in nanocrystalline fcc metals by molecular-dynamics simulation, Mater. Sci. Eng. 318 (2001) 293–312.
[14] A.E. Lobkovsky, J.A. Warren, Sharp interface limit of a phase-field model of crystal grains, Phys. Rev. E 63 (2001) 051605.
[15] R.I. Saye, J.A. Sethian, Analysis and applications of the Voronoi Implicit Interface Method, J. Comput. Phys. 231 (2012) 6051–6085.
[16] A. Vondrous, Grain Growth Behavior and Efficient Large Scale Simulations of Recrystallization with the Phase-field Method, KIT Scientific Publishing, 2013.
[17] R. Backofen, K. Barmak, K.E. Elder, A. Voigt, Capturing the complex physics behind universal grain size distributions in thin metallic films, Acta Mater. 64 (2014) 72–77.
[18] J.M. Tarp, J. Mathiesen, Rotation-limited growth of three-dimensional body-centered-cubic crystals, Phys. Rev. E, Stat., Nonlinear, Soft Matter Phys. 92 (2015) 12409.
[19] G.I. Tóth, T. Pusztai, L. Gránásy, Consistent multiphase-field theory for interface driven multidomain dynamics, Phys. Rev. B - Condens. Matter Mater. Phys. 92 (2015) 1–19.
[20] A.R. Balakrishna, W.C. Carter, Combining phase-field crystal methods with a Cahn-Hilliard model for binary alloys, Phys. Rev. E 97 (2018) 043304.
[21] M. Elsey, S. Esedoglu, P. Smereka, Diffusion generated motion for grain growth in two and three dimensions, J. Comput. Phys. (2009) 1–24.
[22] M. Elsey, S. Esedoglu, P. Smereka, Large-scale simulations and parameter study for a simple recrystallization model, Phil. Mag. 91 (2011) 1607–1642.
[23] V. Derkach, J. McCuan, A. Novick-Cohen, A. Vilenkin, Geometric interfacial motion: coupling surface diffusion and mean curvature motion, in: Y. Maekawa, S. Jimbo (Eds.), Mathematics for Nonlinear Phenomena — Analysis and Computation, Springer Proceedings in Mathematics & Statistics, vol. 215, Springer International Publishing, Cham, 2017, pp. 23–46.
[24] H.J. Frost, C.V. Thompson, D.T. Walton, Simulation of thin film grain structures—i. Grain growth stagnation, Acta Metall. Mater. 38 (1990) 1455–1462.
[25] S. Ta'asan, P. Yu, I. Livshits, D. Kinderlehrer, J. Lee, Multiscale modeling and simulation of grain boundary evolution, in: Proc. 44th AIAA/ASME/ASCE/AHS Structures, Structural Dynamics, and Materials Conference 7-10 April 2003, Norfolk, Virginia, 2003.
[26] D. Kinderlehrer, I. Livshits, S. Ta'asan, A variational approach to modeling and simulation of grain growth, SIAM J. Scientif. Comput. 28 (2006) 1694–1715.
[27] S. Esedoglu, Grain size distribution under simultaneous grain boundary migration and grain rotation in two dimensions, Comput. Mater. Sci. 121 (2016) 209–216.
[28] E. Miyoshi, T. Takaki, Y. Shibuta, M. Ohno, Bridging molecular dynamics and phase-field methods for grain growth prediction, Comput. Mater. Sci. 152 (2018) 118–124.
[29] D. Weygand, Y. Brechet, J. Lepinoux, A vertex simulation of grain growth in 2D and 3D, Adv. Eng. Mater. 3 (2001) 67–71.
[30] R. Henseler, B. Niethammer, F. Otto, A reduced model for simulating grain growth, in: P. Colli, C. Verdi, A. Visintin (Eds.), Free Boundary Problems, Birkhäuser Basel, Basel, 2004, pp. 177–187.
[31] L.A.B. Mora, 2D vertex modeling for the simulation of grain growth and related phenomena, Math. Comput. Simul. 49 (2010) 1–23.
[32] C.E. Torres, M. Emelianenko, D. Golovaty, D. Kinderlehrer, S. Ta'asan, Numerical analysis of the vertex models for simulating grain boundary networks, SIAM J. Appl. Math. 75 (2015) 762–786.
[33] Y. Mellbin, H. Hallberg, M. Ristinmaa, An extended vertex and crystal plasticity framework for efficient multiscale modeling of polycrystalline materials, Int. J. Solids Struct. 125 (2017) 150–160.
[34] G. Gottstein, a. H King, L. Shvindlerman, The effect of triple-junction drag on grain growth, Acta Mater. 48 (2000) 397–403.
[35] G. Gottstein, L. Shvindlerman, Triple junction drag and grain growth in 2D polycrystals, Acta Mater. 50 (2002) 703–713.
[36] V.Y. Novikov, On the influence of triple junctions on grain growth kinetics and microstructure evolution in 2D polycrystals, Scr. Mater. 52 (2005) 857–861.
[37] E.A. Holm, S.M. Foiles, How grain growth stops: a mechanism for grain-growth stagnation in pure materials, Science 328 (2010) 1138–1141.
[38] K. Barmak, E. Eggeling, R. Sharp, S. Roberts, T. Shyu, T. Sun, B. Yao, S. Ta'asan, D. Kinderlehrer, A.D. Rollett, K. Coffey, Grain growth and the puzzle of its stagnation in thin films a detailed comparison of experiments and simulations, Mater. Sci. Forum 715–716 (2012) 473–479.

[39] P. Thamburaja, M. Jamshidian, A multiscale Taylor model-based constitutive theory describing grain growth in polycrystalline cubic metals, J. Mech. Phys. Solids 63 (2014) 1–28.
[40] A.H.J. Sazo, C.E. Torres, An implicit-transition model for numerical simulation of 3D grain growth, 2017 36th International Conference of the Chilean Computer Science Society (SCCC), IEEE, 2017, pp. 1–6.
[41] K. Barmak, M. Emelianenko, D. Golovaty, D. Kinderlehrer, S. Ta'asan, A new perspective on texture evolution, Int. J. Numer. Anal. Model. 5 (2008) 93–108.
[42] I. Yegorov, C.E. Torres, M. Emelianenko, A Boltzmann-type kinetic model for misorientation distribution functions in two-dimensional fiber-texture polycrystalline grain growth, Acta Mater. 109 (2016) 230–247.
[43] P.R. Rios, D. Zöllner, Critical assessment 30: grain growth – unresolved issues, Mater. Sci. Technol. 34 (2018) 629–638.
[44] J. Nickolls, I. Buck, M. Garland, K. Skadron, Scalable parallel programming with CUDA, Queue 6 (2008) 40–53.
[45] Y. Mellbin, H. Hallberg, M. Ristinmaa, Accelerating crystal plasticity simulations using GPU multiprocessors, Int. J. Numer. Meth. Eng. 100 (2014) 111–135.
[46] K. Piękoś, J. Tarasiuk, K. Wierzbanowski, B. Bacroix, Generalized vertex model of recrystallization–application to polycrystalline copper, Comput. Mater. Sci. 42 (2008) 584–594.
[47] M. Bernacki, R. Logé, T. Coupez, Level set framework for the finite-element modelling of recrystallization and grain growth in polycrystalline materials, Scr. Mater. 64 (2011) 525–528.
[48] B. Korbuly, T. Pusztai, H. Henry, M. Plapp, M. Apel, L. Gránásy, Grain coarsening in two-dimensional phase-field models with an orientation field, Phys. Rev. E 95 (2017) 1–12.
[49] L.N. Trefethen, Spectral Methods in MATLAB, Software, Environments, and Tools, Society for Industrial and Applied Mathematics, SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104, 2000.
[50] D. Kinderlehrer, I. Livshits, S. Ta'Asan, A variational approach to modeling and simulation of grain growth, SIAM J. Scientif. Comput. 28 (2006) 1694–1715.
[51] K. Mikula, D. Sevcovic, Evolution of plane curves driven by a nonlinear function of curvature and anisotropy, SIAM J. Appl. Math. 61 (2001) 1473–1501.
[52] J.W. Barrett, H. Garcke, R. Nürnberg, A parametric finite element method for fourth order geometric evolution equations, J. Comput. Phys. 222 (2007) 441–467.
[53] J.W. Barrett, H. Garcke, R. Nurnberg, Numerical approximation of gradient flows for closed curves in Rd, IMA J. Numer. Anal. 30 (2010) 4–60.
[54] T. Sauer, Numerical Analysis, second ed., Addison-Wesley Publishing Company, USA, 2011.
[55] A.H.J. Sazo Gómez, Analysis of 2D and 3D Grain Growth Models in Polycrystalline Materials, Master thesis Universidad Técnica Federico Santa María, 2018.
[56] Alejandro H.J. Sazo, S. Pablo Ibarra, R. Ariel Sanhueza, Francisco J.A. Casas, Claudio E. Torres, Maria Emelianenko, Dmitry Golovaty, Evolution of two-dimensional grain boundary networks implemented in GPU, 2018. https://github.com/tclaudioe/coupled-model-grain-growth-GPU (accessed: 2018-12-27).
[57] P.R. Rios, M.E. Glicksman, Grain boundary, triple junction and quadruple point mobility controlled normal grain growth, Phil. Mag. 95 (2015) 2092–2127.
[58] J.K. Mason, E.A. Lazar, R.D. MacPherson, D.J. Srolovitz, Geometric and topological properties of the canonical grain-growth microstructure, Phys. Rev. E 92 (2015) 063308.
[59] W.W. Mullins, Two-dimensional motion of idealized grain boundaries, J. Appl. Phys. 27 (1956) 900–904.
[60] P. Bardsley, K. Barmak, E. Eggeling, Y. Epshteyn, D. Kinderlehrer, S. Ta'asan, Towards a gradient flow for microstructure, Atti della Accademia Nazionale dei Lincei, Classe di Scienze Fisiche, Matematiche e Naturali, Rendiconti Lincei Matematica e Applicazioni 28 (2017) 777–805.