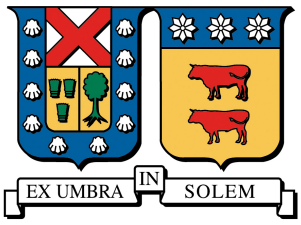
Universidad Técnica Federico Santa María
Departamento de Informática
Valparaíso, Chile



Analysis of 2D and 3D Models for Grain Growth in Polycrystalline Materials

Alejandro Herminio José Sazo Gómez

Tesis para optar al Grado de
Magíster en Ciencias de la Ingeniería Informática

Profesor Guía: Claudio Torres López, Ph.D.
Profesor Correferente Interno: Luis Salinas Carrasco, Ph.D.
Profesora Correferente Externa: Maria Emelianenko, Ph.D.
Profesor Correferente Externo: Dmitry Golovaty, Ph.D.
Presidente Comisión: Marcelo Mendoza, Ph.D.

26 de octubre de 2018

Acknowledgements

This is the result of a work started back in 2014 when I was undergrad research assistant. It has been a challenging study that would not have been possible to do without the support of many people.

I would like to express my sincere gratitude to my advisor, Prof. Claudio Torres, for the time, patience, motivation and knowledge.

My gratitude to the collaborators on the grain growth research, Maria Emelianenko and Dmitry Golovaty, for their questions, comments and commitment.

I would like to thank the robust Grain Growth Team lead by Prof. Torres: Pablo Ibarra and Ariel Sanhueza. We turned Prof. Torres office to a laboratory and we had good times discussing, plotting and sharing.

Thanks to Andrea Figueroa, my friend since we were first-year students. She always had time to listen my ideas and worries.

A special mention to my friends at the Artificial Intelligence Laboratory, LabIA, who were very kind letting me work in their facilities.

All my gratitude to Alondra, my girlfriend, whose love and patients was fundamental.

Last, but not least, I want to say a heartfelt thank you to my mother, Lucila. She's the pillar of the family.

This thesis project was supported by FONDECYT 11160744, CCTVal CONICYT PIA/Basal FB0821 and with the support of the *Dirección de Postgrado y Pogramas*, UTFSM through the PIIC program.

Abstract

The microstructure of polycrystalline materials is composed of grains separated by their interfaces called grain boundaries, meeting at triple junctions. The orientation and shape of these grains define material properties such as resistance, electric conductivity, among others. The improvement of such properties is achieved by modifying the underlying structure via primary recrystallization and grain growth. In this work we analyze in depth and solve two-dimensional and three-dimensional models of grain growth and nucleation and obtain relevant statistics. We deal with computational challenges related to algorithms stability, scalability and parallel programming in GPU with the objective to simulate hundreds of thousands of grains, for example, by improving the extinction time estimation and flipping detection. The Prediction-Correction algorithm that ensures boundaries stability when they are close to topological transitions, and the parallel management of these transitions in two-dimensional models. Finally, we extract statistics from a three-dimensional model using image analysis techniques.

Resumen

La estructura interna de los materiales policristalinos está compuesta por granos, separados por sus interfaces llamadas fronteras, las cuales inciden en uniones triples. La orientación y la forma de estos granos definen propiedades de los materiales tales como resistencia, conductividad eléctrica entre otras. Para mejorar estas propiedades se debe modificar la estructura subyacente de granos vía recristalizacion y crecimiento de granos. En este trabajo se analizan en profundidad y se resuelven modelos de nucleación y de crecimiento de granos en dos y tres dimensiones y se obtienen estadísticas relevantes. Además, se abordan retos computacionales relacionados a la estabilidad y escalabilidad de los algoritmos y programación paralela en GPU con el objetivo de simular cientos de miles de granos, por ejemplo la mejora de la estimación del tiempo de extinción y de detección de flippings. El algoritmo Predictor-Corrector que asegura fronteras estables al momento de *flippings* y el manejo en paralelo de transiciones topológicas en modelos de dos dimensiones. Finalmente se extraen estadísticas de otro modelo tridimensional utilizando técnicas de análisis de imágenes.

Contents

Acknowledgements **i**

Abstract **ii**

Resumen **iii**

Contents **iv**

List of Figures **vii**

1 Introduction **1**
1.1 Objectives 1
1.1.1 Specific Objectives 2
1.2 Structure 2

2 Two-dimensional Grain Growth **3**
2.1 Topological Transitions 4
2.2 Topology of Grain Structures 6
2.3 Curvature and Vertex Models 7

3 Closed Boundary Motion **9**
3.1 Curvature based motion 10
3.2 Motion for a Closed Boundary with Discrete Points 10
3.2.1 Circular Boundary 11
3.2.2 Non-regular Boundary 14

4 The Coupled Model **19**
4.1 Derivation of Evolution Equations 19
4.1.1 Triple Junction Evolution and Normal Component of Interior Points Velocities 19
4.1.2 Tangential Component of Interior Points Velocities 21
4.2 Numerical Implementation 24
4.2.1 Estimation of Extinction Time 24
4.2.2 Multistep Euler 25

4.2.3 Multistep Second Order Runge-Kutta . . . 26
4.2.4 Extinction Time Estimation with Multistep Methods . . . 27
4.2.5 Predictor-Corrector Algorithm . . . 27
4.3 Numerical Experiments . . . 32
4.3.1 Energy Minimization . . . 32
4.3.2 Statistics . . . 32

5 A Continuous Stored Energy Vertex Model with Nucleation 35
5.1 Numerical Algorithm . . . 37
5.2 The Stored Energy Effect Over the Dynamics of a Triple Junction . . 39
5.3 The Nucleation Process: Adding a three side Grain and Letting it Grow 41
5.3.1 A Symmetric Nucleation Analysis . . . 43
5.3.2 Analysis of the Nucleated Grain . . . 45
5.3.3 Alternative Nucleation with an Specific Orientation . . . 45
5.4 Stored Energy Vertex Model with Nucleation Algorithm . . . 46
5.5 Numerical Experiments . . . 46
5.5.1 Energy Minimization . . . 48
5.5.2 Statistics from Model with Basic Nucleation . . . 48
5.5.3 Statistics for Model with Alternative Nucleation . . . 50

6 Parallel Management of Topological Transitions 67
6.1 Sequential Management . . . 67
6.2 Parallel Polling System . . . 68

7 Implicit-Transition Model for Three-dimensional Grain Growth 72
7.1 Numerical Experiments . . . 73
7.1.1 Energy Minimization . . . 74
7.1.2 Topological Transitions . . . 74

8 Generation of Two-dimensional statistics from a Three-dimensional Grain Growth Model 78
8.1 Generation of statistics . . . 78
8.2 Numerical Experiments . . . 79

9 Conclusions 81
9.1 Future Work . . . 83

Bibliography 85

Appendix A Tools and Technical Specifications 88
A.1 Operating Systems . . . 88
A.2 Programming Languages . . . 88
A.3 Simulation Software . . . 88
A.4 Numerical Libraries . . . 89

A.5 Art and Text Libraries 89
A.6 Hardware 89

List of Figures

2.1 Simulated grain structures under periodic boundary conditions. . . . 4

3.1 Circular boundary evolution 15
3.2 Circular boundary area and rate of change 16
3.3 Circular boundary energy and rate of change 16
3.4 Non-regular boundary evolution 17
3.5 Non-regular boundary area and rate of change 18
3.6 Non-regular boundary energy and rate of change 18

4.1 Example of delayed flipping. 25
4.2 Flipping detection mechanism that prevents delay in topological transitions. 27
4.3 Sketch of the Predictor-Corrector Algorithm. 30
4.4 Total energy of the grain network with the Coupled Model. 32
4.5 Distributions for Coupled Model numerical simulations. 34

5.1 Vertex perturbation in Stored Energy model 38
5.2 Configuration of three grains with different values of stored energy and a vertex with one degree of freedom 39
5.3 Stability regions for the relation of b and $\Delta\mathcal{E}$ in the three grains experiment. 41
5.4 Steady states for different configurations of stored energies. 42
5.5 Convergence of steady states for a fixed value of b and different values of $b\Delta\mathcal{E}$ 42
5.6 Sketch of energies associated to a 3-side grain before and after it is nucleated. Red shows what it is going to be removed and green what it is going to be added. 43
5.7 $\Delta E^{\triangle}$ induced by nucleation in a symmetric case. 44
5.8 Sketch of energies for computing $\Delta^2 E^{\triangle}$ 45
5.9 System energy over time for different models of stored energy. 49
5.10 Stages of grain network evolution nucleating with $\alpha = 0$. 53
5.11 Grain relative area distributions at different stages of grain network evolution and nucleation with $\alpha = 0$. 54

5.12 Dihedral angle distributions at different stages of grain network evolution and nucleation with $\alpha = 0$. 55
5.13 Grain Boundary Character distributions at different stages of grain network evolution and nucleation with $\alpha = 0$. 56
5.13 Grain Boundary Character distributions at different stages of grain network evolution and nucleation with $\alpha = 0$. (Cont.) 57
5.14 Average number of sides distributions at different stages of grain network evolution and nucleation with $\alpha = 0$. 58
5.15 Stored energy distributions at different stages of grain network evolution and nucleation with $\alpha = 0$. 59
5.16 Stages of grain network evolution nucleating with $\alpha = \alpha_{\text{nucl}}$. 60
5.17 Grain relative area distributions at different stages of grain network evolution and nucleation with $\alpha = \alpha_{\text{nucl}}$. 61
5.18 Dihedral angle distributions at different stages of grain network evolution and nucleation with $\alpha = \alpha_{\text{nucl}}$. 62
5.19 Grain Boundary Character distributions at different stages of grain network evolution and nucleation with $\alpha = \alpha_{\text{nucl}}$. 63
5.19 Grain Boundary Character distributions at different stages of grain network evolution and nucleation with $\alpha = \alpha_{\text{nucl}}$. (Cont.) 64
5.20 Average number of sides distributions at different stages of grain network evolution and nucleation with $\alpha = \alpha_{\text{nucl}}$. 65
5.21 Stored energy distributions at different stages of grain network evolution and nucleation with $\alpha = \alpha_{\text{nucl}}$. 66

6.1 Polling system scheme. 70

7.1 Total energy of three-dimensional Models: Implicit-transition Model and evolution with normal vectors. 75
7.2 Flipping scheme on a single three-dimensional grain. 76
7.3 Surface transition in a three-dimensional grain 76
7.4 Three-dimensional grain removal 77

8.1 Discrete and continuous representations of the two-dimensional slide obtained from the three-dimensional simulation. 79
8.2 Distribution of relative areas for two-dimensional cuts with surface tensions equal to 1. 80
8.3 Distribution of relative areas for two-dimensional cuts with surface tensions obeying Read-Shockley function. 80

9.1 Overview of the addressed topics in this Thesis. 81

List of Algorithms

4.1 Multistep Euler for Coupled Model 26
4.2 Multistep Second Order Runge-Kutta for Coupled Model 26
4.3 Predictor-Corrector 29
5.1 Stored Energy Vertex Model 47
6.1 Sequential Management of Topological Transitions 68
6.2 Polling Routine for Inhibit Boundaries 71
6.3 Parallel Polling System for Managing Topological Transitions 71
7.1 Implicit-transition Model for Grain Growth 74

Chapter 1

Introduction

The microstructure of polycrystalline materials consists of small crystallites, called grains which are separated by their interfaces, called grain boundaries, and they meet at triple junctions. The orientation and shape of these grains define material's properties across wide scales such as thermal and electric conductivity, resistance, fracture toughness, corrosion resistance, among others [11, 12, 6, 32].

The improvement of material properties is achieved by inducing the modification of the microstructure primary through recrystallization and corresponding grain growth [11, 12, 6, 32, 20, 22, 21, 19]. The study of mathematical models and their computational implementation is very important since it allows to understand the underlying natural process being modeled. It also allows to address how accurately the model behaves and how stable it is under parameter variations.

All of this allows the extraction of robust statistics that can be compared to the experimental data. Computational scalability and performance of the implementation is important since running larger systems in short times is required to obtain reliable statistics. Therefore this study has a strong emphasis on efficient implementation of the models, specifically taking advantage of the graphic processing units (GPUs) [18, 17], allowing us to run simulations for systems that contain hundreds of thousands of grains.

1.1 Objectives

The main objective of this thesis is to contrast different grain growth mathematical models in two and three dimensions with the experimental data obtained from real materials through statistics related to geometry and energy of the grain structure. These statistics are average area, number of sides of each grain or grain class, dihedral angle at triple junctions, stored energy, grain boundary energy and misorientation.

1.1.1 Specific Objectives

- Analysis of curvature motion for boundaries and and stability study towards the improvement of the coupled model developed in [27] as is a work-in-progress publication [28]. This is addressed in Chapters 3, 4 and 6.
- Develop two-dimensional as well as three-dimensional models of grain growth and extract relevant statistics. This is addressed in Chapters 4, 5 and 7.
- Use a three-dimensional model of grain growth, generate two-dimensional slices and extract relevant statistics from them to analyze the relation with statistics from two-dimensional models. This is addressed in Chapters 8.

1.2 Structure

The present work is structured as follows. Chapter 2 presents a general overview of the two-dimensional grain growth, notation, definitions, the topological transitions idea and topological characteristics related to the periodic boundary conditions used for numerical simulation, all key ideas needed to understand the background of the algorithms that will be presented in the following chapters. Chapter 3 presents an extensive analysis of curvature driven motion in closed boundaries with the objective of understand the behavior of such motion and apply it to other models. Chapter 4 is an overview of the Coupled Model developed during the bachelor thesis (see [27]) with improvements of the stability of the interior points that defines grain boundaries by introducing a novel tangential term to the velocity as well as capturing the curvature driven motion by the introduction of a correction coefficient derived from the analysis of closed boundaries. We present numerical experiments related to this accomplishment. Chapter 5 presents the Continuous Stored Energy Vertex Model which is an extension of a Vertex Model for grain growth and includes a new term that allows nucleation, that is, the introduction of new grains that can grow despite having three sides. An extensive analysis of the conditions that enables growth is presented along with numerical experiments comparing nucleation and grain growth processes. Chapter 6 develops a parallel algorithm for handling topological transitions since the continuous formulations for Vertex Model and Coupled Model consider this stage as sequential and thus is a non optimized part of both algorithms. This is required since the implemented code for this Thesis was done for a GPU. Chapter 7 presents a three-dimensional model for grain growth based on the idea of the Vertex model. The model is a first approach to obtain an algorithm free of explicit handling of topological transitions since the topological transitions increase their complexity in higher dimensions. Chapter 8 briefly introduces a three-dimensional model from the state-of-the-art and a procedure to extract two-dimensional slices of the simulated grain structure and then obtain statistics using image analysis software techniques. Finally, Chapter 9 summarizes the conclusions of each chapter and presents future work.

Chapter 2

Two-dimensional Grain Growth

In this section, grain growth is studied in thin films of polycristalline materials as a simplification of dynamics of real three-dimensional grain structures. These structures are composed by grains and boundaries. Each boundary is shared by two grains and the point where three boundaries (and also three grains) meet is called a vertex or triple junction. General topology also admits more than three neighbor vertices, however we will only take into account in the models the existence of triple junctions, also calling them vertices. Formally, the mathematical description of a grain structure is defined over the unit squared domain $[0,1]^2 \subset \mathbb{R}^2$ with periodic boundary conditions, thus the domain is a torus. The grain structure is composed by a set of N disjoint regions over the whole domain. We denote this set as,

$$\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{G}}(t) = \left\{\mathcal{G}^{(1)}, \mathcal{G}^{(2)}, \ldots, \mathcal{G}^{(N)}\right\}, \quad N = N(t). \tag{2.1}$$

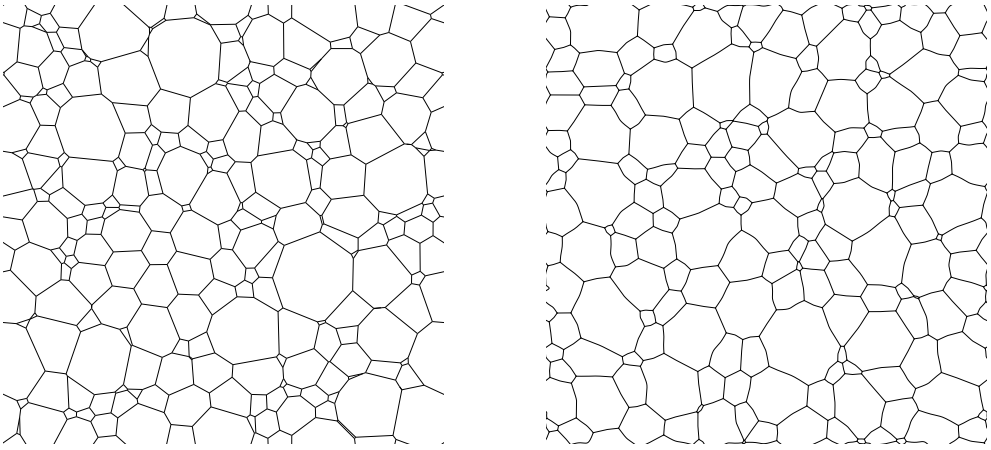
Note that the number of grains varies over time, which is indicated in the temporal dependence notation. With each grain $\mathcal{G}^{(l)},\ l = 1, \ldots, N$ we associate an orientation $\alpha_l \in [0, 2\pi)$. The grains are limited by their boundaries. The boundary set is defined as,

$$\boldsymbol{\Gamma} = \boldsymbol{\Gamma}(t) = \{\Gamma^{(1)}, \Gamma^{(2)}, \ldots, \Gamma^{(K)}\}, \quad K = K(t). \tag{2.2}$$

Again, the number of boundaries in the system also depends on time. The grain misorientation parameter $\Delta\alpha^{(k)}$ is defined as $\Delta\alpha^{(k)} = \alpha_{l_2} - \alpha_{l_1},\ k = 1, \ldots, K$ where l_1 and l_2 are the indices of the two grains adjacent to the k-th boundary. The grain boundary energy $\gamma^{(k)}$ is assumed to depend only on the grain misorientation parameter, that is $\gamma^{(k)} = \gamma(\Delta\alpha^{(k)}),\ k = 1, \ldots, K$ and some even periodic function $\gamma : \mathbb{R} \to \mathbb{R}$. In this work we will use the grain energy function defined in [12] which is,

$$\gamma^{(k)}(\Delta\alpha) = 1 + \frac{\varepsilon}{2}\left(1 - \cos^3(4\Delta\alpha)\right). \tag{2.3}$$

If we set $\varepsilon = 0$ all the energies will be equal to 1, which we call *isotropic setting*. Other values of ε yields the *anisotropic setting*, giving different values of energy for



(a) Vertex-based structure (b) Curvature-based structure

Figure 2.1: Simulated grain structures under periodic boundary conditions.

each boundary. Each boundary is parametrized as a curve in the plane defined as:

$$\Gamma^{(k)}(t) = \left\{ \boldsymbol{\xi}^{(k)}(s,t), \quad s \in [0,1] \right\}, \quad k = 1, \ldots, K. \tag{2.4}$$

As stated previously, boundaries meet at triple junctions, and these correspond to the start or the end point of exactly three boundaries. We denote the set of triple junctions as,

$$\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{X}}(t) = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M\}, \quad M = M(t). \tag{2.5}$$

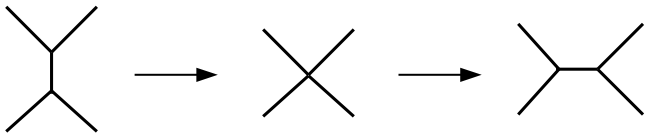
Let $\Gamma^{(k_1)}, \Gamma^{(k_2)}, \Gamma^{(k_3)}$ three different boundaries with a common triple junction $\mathbf{x}_m$. For the curve parameters $s_{k_i} = \{0, 1\}$ the triple junctions are defined in function of the boundaries as

$$\mathbf{x}_m(t) = \boldsymbol{\xi}^{(k_1)}(s_{k_1}, t) = \boldsymbol{\xi}^{(k_2)}(s_{k_2}, t) = \boldsymbol{\xi}^{(k_3)}(s_{k_3}, t), \quad m = 1, \ldots, M.$$

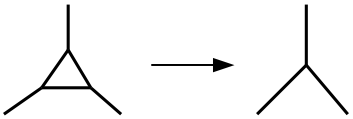
Figure 2.1 shows a simulated grain structures under the given definition. As discussed before, the grain system evolves over time, vertices and boundaries moves, some grains shrink and other grow. When a boundary decreases its length to zero, or when a grain area becomes zero, a component of the system disappears. These changes are denominated topological transitions and are described in the next section, see Section 2.1.

2.1 Topological Transitions

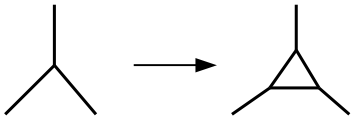
Topological transitions are the result of coarsening during grain growth and are well documented by Ferro et al. [9]. Ferro et al. describes two-dimensional and three-dimensional elimination of grains, considering for instance the removal of ensembles



(a) Neighbor flipping.



(b) Grain removal.



(c) Nucleation in a vertex.

of five-sided grains. The simplest and most common handled topological transitions [11, 12, 32, 34] in grain growth models are neighbor switching (also known as flipping) and the removal of three-sided grains, as shown in Figures 2.2a and 2.2b. The addition of a new grain into the system, also called nucleation, it is a type of topological transition that happens in primary recrystallization before grain growth. Along with this transition, other types of transition related to nucleation have been documented, but they will not be included here, see [22].

Neighbor flipping (Figure 2.2a) consist in a grain boundary switching neighboring boundaries. This occurs when a boundary collapses, the vertices approach each other, decreasing the boundary length to zero. This configuration, called a quadruple junction since it involves four boundaries, is unstable under the regimes studied and breaks in two new vertices and the new boundary grows in a new direction.

Grain removal (Figure 2.2b) consists in the removal of a grain which decreased its area to zero. The grain then is replaced by a group of boundaries or a single vertex. In [9] are shown different configurations of grains and how they are arranged after grain removals. We focus only in the removal of three sided grains. A three sided grain is removed when one or more of its boundaries switch neighbors. This will generate a degenerated grain with two sides. Instead of that, a new vertex is created and the disjointed boundaries adjacent to the former grain are now connected at a vertex.

It is possible that a grain being removed takes part of a configuration where one of its neighbors also has three sides. In this case the grain removal generates another two-sided grain that is not valid in the network and for network consistency we can

either stop the simulation or repair the structure by replacing the two three-sided grains by a single boundary.

Nucleation (Figure 2.2c) consists of the addition of a grain in the grain system. The energetic conditions and the site in the grain structure to add the new grain (on a boundary, vertex or inside another grain) are part of the primary recrystallization phenomenon (see [22, 21, 20]). Chapter 5 addresses a Vertex Model that is able to produce this transition.

2.2 Topology of Grain Structures

As stated before, the grain structure is projected on a torus where each vertex is connected to three other vertices, and thus belongs to three boundaries. The structure has the following Euler characteristic considering the number of vertices M, the number of boundaries K and the number of grains N [25]:

$$M - K + N = 0. \tag{2.6}$$

If we count the number of vertices in a grain structure we count three boundaries per vertex, but as each boundary is shared by two vertices, the boundaries are counted twice. Using these facts and considering (2.6) we obtain the following relations:

$$3\,M = 2\,K \qquad\qquad N = \frac{K}{3} = \frac{M}{2}.$$

Consider an initial condition of a grain structure with N_0 number of grains, M_0 vertices and K_0 boundaries where $N_0 = M_0/2 = K_0/3$. The topological transitions during grain structure evolution will change the number of these components. The only topological transition considered that alter the number of components in a grain structure is grain removal. When a grain is removed, three boundaries and two vertices are removed, leaving a single *survivor* vertex. Now assume that at a given time t we started with N_t grains and then we removed n_t grains, leaving at the next time step N_{t+1} grains, $N_{t+1} < N_t$. Topological transitions described induces that we removed exactly $3\,n_t$ boundaries and $2\,n_t$ vertices and the following definitions yields between vertices, boundaries and grains between time steps t and $t+1$:

$$\begin{aligned} dN_t &= N_{t+1} - N_t = -n_t \\ dK_t &= K_{t+1} - K_t = -3n_t \\ dM_t &= M_{t+1} - M_t = -2n_t. \end{aligned}$$

We can build the following relation in terms of the variation of grains as

$$3\,dM_t = 2\,dK_t. \tag{2.7}$$

If we add the variations over all the time steps, from time $t = 0$ to $t = T$ we obtain

$$\begin{aligned}\sum_{t=0}^{T} 3\, dM_t &= 3(M_1 - M_0 + M_2 - M_1 + \cdots + M_{T+1} - M_T) \\ &= 3(M_{T+1} - M_0).\end{aligned}$$

$$\begin{aligned}\sum_{t=0}^{T} 2\, dK_t &= 2(K_1 - K_0 + K_2 - K_1 + \cdots + K_{T+1} - K_T) \\ &= 2(K_{T+1} - K_0).\end{aligned}$$

This total variations for vertices and boundaries are equal according to (2.7). Under the fact that the initial condition is true, i.e., $3M_0 = 2K_0$, for a time $t = T$ the number of vertices and boundaries are related as,

$$N_{T+1} = \frac{M_{T+1}}{2}, \qquad N_{T+1} = \frac{K_{T+1}}{3}, \qquad 3\, M_{T+1} = 2\, K_{T+1}.$$

This implies that grain structure evolution holds the same relation between grains, boundaries and vertices along the simulation.

2.3 Curvature and Vertex Models

The total energy of the grain structure depends on the individual energies along each grain boundary. The total energy thus takes the form,

$$E(t) = \sum_{k=1}^{K} \int_0^1 \gamma^{(k)} \left\| \mathbf{l}^{(k)}(s,t) \right\| \, ds, \tag{2.8}$$

where $\mathbf{l}^{(k)}(s,t) = \dfrac{\partial \boldsymbol{\xi}}{\partial s}^{(k)}(s,t)$ is a tangent vector to $\boldsymbol{\xi}^{(k)}$ and $\|\cdot\|$ is the l^2-norm. Grain network evolution equations can be found from this equation by taking the derivative of the energy with respect to time. Computing the derivative of (2.8) we obtain,

$$\begin{aligned}\frac{dE}{dt}(t) &= \sum_{k=1}^{K} \int_0^1 \gamma^{(k)} \frac{\mathbf{l}^{(k)}(s,t)}{\|\mathbf{l}^{(k)}(s,t)\|} \cdot \frac{\partial \mathbf{l}}{\partial t}^{(k)}(s,t)\, ds \\ &= \sum_{k=1}^{K} \int_0^1 \gamma^{(k)}\, \mathbf{T}^{(k)}(s,t) \cdot \frac{\partial \mathbf{v}}{\partial s}^{(k)}(s,t)\, ds,\end{aligned} \tag{2.9}$$

where $\mathbf{T}^{(k)}(s,t)$ is a unit tangent vector of the boundary and $\gamma^{(k)}\mathbf{T}^{(k)}$ it is called capillar force. Note that (2.9) is valid as long as $K'(t) = 0$ within $[t_1, t_2]$, i.e., there

are not topological transitions. Integrating by parts (2.9) we obtain,

$$\frac{dE}{dt}(t) = -\sum_{k=1}^{K} \int_0^1 \gamma^{(k)} \frac{\partial \mathbf{T}^{(k)}}{\partial s} \cdot \mathbf{v}^{(k)}\, ds + \sum_{m=1}^{M} \mathbf{v}_m \cdot \sum_{l=1}^{3} \gamma^{(m,l)} \mathbf{T}^{(m,l)}, \tag{2.10}$$

In order to enforce the decreasing energy of the system we must ensure that (2.10) be negative.

The triple junction velocity is set to decrease energy as,

$$\mathbf{v}_m = -\lambda \gamma^{(m,l)} \mathbf{T}^{(m,l)},$$

where λ is called triple junction mobility. Boundary velocity is set analogously as,

$$\mathbf{v}^{(k)} = \mu \gamma^{(k)} \frac{\partial \mathbf{T}^{(k)}}{\partial s},$$

where μ is called boundary mobility.

If the mobility of the triple junction is much lower than the mobility of the boundaries, the boundaries will evolve towards a straight line, which implies $\frac{\partial \mathbf{T}^{(k)}}{\partial s} = 0$ for each boundary. Under this assumption the velocity of the boundaries becomes zero. This model is called *Vertex model*.

On the other hand if we consider the mobility of grain boundaries higher than the triple junctions mobility we obtain a curvature-driven motion, i.e., there is no triple junction influence. This model is called *Curvature model*.

A geometrical consequence under this model is the *Herring condition* where the angle between boundaries, namely dihedral angle, is always $2\pi/3$ under isotropic conditions [10]. This comes from forcing the geometrical condition over tangent vectors:

$$\sum_{l=1}^{3} \mathbf{T}^{(m,l)} = 0. \tag{2.11}$$

Moreover the rate of growth for a grain becomes independent of its area, being proportional to the number of sides of the grain or grain class $\mathrm{ns}(\mathcal{G})$. This linear relation establishes that a grain with $\mathrm{ns}(\mathcal{G}) < 6$ will shrink and with $\mathrm{ns}(\mathcal{G}) > 6$ will grow, in fact the case of 6-sided grain the rate of growth is zero. This relation is known as *Von Neumann-Mullins relation* [15] and can be expressed as

$$\frac{dA}{dt}(\mathcal{G}) = \frac{\nu}{3}(\mathrm{ns}(\mathcal{G}) - 6) \tag{2.12}$$

where ν is a term proportional to grain boundary energies $\gamma^{(k)}$ and mobility μ.

Chapter 3

Closed Boundary Motion

An interesting case of study for understanding mathematical and numerical aspects of grain growth simulation is to understand the evolution of a closed boundary which tries to minimize its energy. Let $\boldsymbol{\xi}(s,t) = \langle x(s,t), y(s,t)\rangle$, $s \in [0, 2\pi]$ be a closed curve in $\mathbb{R}^2$ such that $\boldsymbol{\xi}(0,t) = \boldsymbol{\xi}(2\pi,t)$, where s is the parametrization variable and t is the time variable. This will be the base definition of a closed boundary. Let $\mathbf{l}(s,t) = \dfrac{\partial \boldsymbol{\xi}}{\partial s}(s,t)$ a tangent vector to the boundary $\boldsymbol{\xi}$ and $\mathbf{T}(s,t) = \dfrac{\mathbf{l}(s,t)}{\|\mathbf{l}(s,t)\|}$ the unit tangent vector in the same direction. Also let $\mathbf{N}(s,t) = \dfrac{\partial \mathbf{T}}{\partial s}(s,t)$ a unit normal vector to $\boldsymbol{\xi}$. From the total energy equation in (2.8), the energy of this boundary becomes a single integral term:

$$E(t) = \int_0^{2\pi} \|\mathbf{l}(s,t)\| \, ds, \tag{3.1}$$

where for simplicity we assumed an isotropic regime, i.e., $\gamma = 1$. Taking the derivative of (3.1) with respect to the time yields:

$$\frac{dE}{dt}(t) = \int_0^{2\pi} \frac{\mathbf{l}(s,t)}{\|\mathbf{l}(s,t)\|} \cdot \frac{\partial \mathbf{l}}{\partial t}(s,t)\, ds. \tag{3.2}$$

Let $\mathbf{v}(s,t) = \dfrac{\partial \boldsymbol{\xi}}{\partial t}(s,t)$ the velocity of the boundary. Equation (3.2) becomes:

$$\frac{dE}{dt}(t) = \int_0^{2\pi} \mathbf{T}(s,t) \cdot \frac{\partial \mathbf{v}}{\partial s}(s,t)\, ds. \tag{3.3}$$

Integrating by parts (3.3) we obtain:

$$\begin{aligned} \frac{dE}{dt}(t) &= \mathbf{T}(s,t) \cdot \mathbf{v}(s,t)\Big|_0^{2\pi} - \int_0^{2\pi} \frac{\partial \mathbf{T}}{\partial s}(s,t) \cdot \mathbf{v}(s,t)\, ds \\ &= -\int_0^{2\pi} \frac{\partial \mathbf{T}}{\partial s}(s,t) \cdot \mathbf{v}(s,t)\, ds. \end{aligned} \tag{3.4}$$

Thus, we will use (3.4) to understand the motion of a closed boundary.

3.1 Curvature based motion

Curvature-based grain growth models assumes that the velocity of the boundaries is in the direction of its normal vector and proportional to its curvature $\kappa(s,t)$ [12]. Thus the velocity of the boundary can be defined as [31]:

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

This term can be obtained also from the derivative of the vector $\mathbf{T}$ with respect to the arc length $\mathcal{L}$:

$$\begin{aligned} \frac{\partial \mathbf{T}}{\partial \mathcal{L}}(s,t) &= \kappa(s,t)\mathbf{N}(s,t) \\ \frac{\partial \mathbf{T}}{\partial s}(s,t)\frac{ds}{d\mathcal{L}} &= \kappa(s,t)\mathbf{N}(s,t). \end{aligned} \tag{3.5}$$

Let $\mathcal{L}(s)$ the arc length of the boundary up to s, given by:

$$\mathcal{L}(s) = \int_0^s \|\mathbf{l}(s,t)\|\, ds, \tag{3.6}$$

then we can obtain $\dfrac{ds}{d\mathcal{L}}$ from (3.6) by derivating with respect to s as:

$$\begin{aligned} \frac{d\mathcal{L}}{ds} &= \|\mathbf{l}(s,t)\| \\ \frac{ds}{d\mathcal{L}} &= \frac{1}{\|\mathbf{l}(s,t)\|}. \end{aligned}$$

Replacing in (3.5) we obtain:

$$\frac{1}{\|\mathbf{l}(s,t)\|}\frac{\partial \mathbf{T}}{\partial s}(s,t) = \kappa(s,t)\mathbf{N}(s,t). \tag{3.7}$$

Therefore to obtain curvature based motion we need to compute the grain boundary velocity as:

$$\mathbf{v}(s,t) = \frac{1}{\|\mathbf{l}(s,t)\|}\frac{\partial \mathbf{T}}{\partial s}(s,t). \tag{3.8}$$

3.2 Motion for a Closed Boundary with Discrete Points

Let's consider the boundary $\boldsymbol{\xi}(s,t)$ and a continuous representation via the collocation points $\boldsymbol{\mathcal{X}} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ and a interpolating function ϕ as:

$$\boldsymbol{\xi}(s,t) = \sum_{i=1}^{n} \mathbf{x}_i(t)\, \phi_i(s)\,, \quad s \in [0, 2\pi]. \tag{3.9}$$

Due to our construction of the boundary, this interpolating function must preserve the periodicity. For example an interpolating function to be considered here is the periodic sinc [33]. The velocity of the boundary $\mathbf{v}$ is expressed in terms of the parametrization as:

$$\mathbf{v}(s,t) = \sum_{i=1}^{n} \dot{\mathbf{x}}_i(t)\, \phi_i(s)\,. \tag{3.10}$$

We can replace the velocity term in (3.4) to obtain the derivative of the energy in terms of the boundary.

$$\begin{aligned} \frac{dE}{dt}(t) &= -\int_0^{2\pi} \frac{\partial \mathbf{T}}{\partial s}(s,t) \cdot \left(\sum_{i=1}^{n} \dot{\mathbf{x}}_i(t)\, \phi_i(s) \right) ds \\ &= -\sum_{i=1}^{n} \dot{\mathbf{x}}_i(t) \cdot \int_0^{2\pi} \frac{\partial \mathbf{T}}{\partial s}(s,t)\phi_i(s)\ ds. \end{aligned} \tag{3.11}$$

The velocity at the boundary points $\mathbf{x}_i$ can be defined from (3.11) so that the energy of the grain system decreases in time:

$$\dot{\mathbf{x}}_i(t) = \int_0^{2\pi} \frac{\partial \mathbf{T}}{\partial s}(s,t)\phi_i(s)\ ds, \tag{3.12}$$

In order to capture the curvature motion given this approximation of closed boundary we introduce the term $1/\left\|\mathbf{l}(s,t)\right\|$ at a given point $\mathbf{x}_i$ from (3.7). We also introduce a constant ζ which depends on the specific parametrization and interpolation.

$$\dot{\mathbf{x}}_i(t) = \frac{\zeta}{\left\|\mathbf{l}(s_i,t)\right\|} \int_0^{2\pi} \frac{\partial \mathbf{T}}{\partial s}(s,t)\phi_i(s)\ ds, \tag{3.13}$$

This evolution equation differs with the curvature based evolution equation found before in (3.8). Instead of having normal vector $\dfrac{\partial \mathbf{T}}{\partial s}(s,t)$ we have an approximation given by an integral term, that is, an *integral* normal vector at a given point along the boundary. It can be seen as a smoothed normal term. In order to understand this evolution equation, we will study a circular boundary in the next subsection, in which simple formulas can be obtained.

3.2.1 Circular Boundary

This simple case is very illustrative, since we can obtain an explicit formula for $\dfrac{dE}{dt}$ and also the area rate of change $\dfrac{dA}{dt}$. First we describe analytic results for the boundary evolution and then we apply the coupled model idea to interpolate the boundary and obtain the new evolution equations.

The equation that describes this boundary is:

$$\boldsymbol{\xi}(s,t) = R(t)\langle\cos(s),\sin(s)\rangle, \quad R(t) > 0 \ \ \forall\, t \geq 0, \tag{3.14}$$

where we have decoupled the spatial parametrization in polar coordinates with the radial time-dependence. Notice that $\|\boldsymbol{\xi}(s,t)\| = R(t)$, that is, for a fixed time $t = \tau$, the boundary preserves a constant radius and thus constant curvature along the boundary. The tangent vector $\mathbf{l}(s,t)$ is nothing but:

$$\mathbf{l}(s,t) = R(t)\langle-\sin(s),\cos(s)\rangle,$$

and again its norm $\|\mathbf{l}(s,t)\| = R(t)$ just like $\boldsymbol{\xi}$. This implies that the unit tangent vector is:

$$\mathbf{T}(s,t) = \langle-\sin(s),\cos(s)\rangle,$$

and the normal vector is:

$$\frac{\partial \mathbf{T}}{\partial s}(s,t) = \langle-\cos(s),-\sin(s)\rangle,$$

which is a vector always pointing to the circle center. If we took these results and plug them in (3.4) yields:

$$\frac{dE}{dt}(t) = \int_0^{2\pi} \langle\cos(s),\sin(s)\rangle \cdot \mathbf{v}(s,t)\, ds.$$

Considering that the boundary moves proportional to the curvature and in a normal direction, we replace the velocity term with the curvature result in (3.7):

$$\begin{aligned}
\frac{dE}{dt}(t) &= \int_0^{2\pi} \langle\cos(s),\sin(s)\rangle \cdot \frac{1}{\|\mathbf{l}(s,t)\|}\langle-\cos(s),-\sin(s)\rangle\, ds \\
&= \int_0^{2\pi} -\frac{1}{R(t)}\, ds \\
&= -2\pi\kappa(t)
\end{aligned} \tag{3.15}$$

The rate of change of the area can also be obtained from the definition of area given the boundary parametrization in (3.14):

$$\begin{aligned}
A(t) &= \frac{1}{2}\int_0^{2\pi} \|\boldsymbol{\xi}(s,t)\|^2\, ds \\
\frac{dA}{dt}(t) &= \int_0^{2\pi} \boldsymbol{\xi}(s,t)\cdot\mathbf{v}(s,t)\, ds \\
&= \int_0^{2\pi} R(t)\langle\cos(s),\sin(s)\rangle \cdot \frac{1}{R(t)}\langle-\cos(s),-\sin(s)\rangle\, ds \\
&= -2\pi
\end{aligned} \tag{3.16}$$

This means that the rate of change for the area of a circle is constant. The evolution of the radius $R(t)$ over time can be explicitly found by comparing the velocity of the curvature in (3.7) with the circle velocity expression:

$$\begin{aligned} \frac{dR}{dt}(t) &= -\frac{1}{R(t)} \\ R(0) &= R_0, \end{aligned} \tag{3.17}$$

where R_0 is the initial radius of the circle at time $t = 0$. The solution therefore is,

$$R(t) = \sqrt{R_0^2 - 2t}. \tag{3.18}$$

The solution is valid as long as $R_0^2 > 2t$. In the limit, the circle should become a single point, and the related tangent and normal vectors becomes indeterminate. Finally, the complete description of a circular boundary is given by:

$$\boldsymbol{\xi}(s,t) = \sqrt{R_0^2 - 2t}\,\langle \cos(s), \sin(s)\rangle. \tag{3.19}$$

The coupled model based formula induces a different result from the velocity found in (3.17). Assuming equispaced points in $[0, 2\pi]$, we replace $\mathbf{l}$ and $\dfrac{\partial \mathbf{T}}{\partial s}$ for the circle in (3.13) to obtain:

$$\dot{\mathbf{x}}_i(t) = \frac{1}{R(t)} \int_0^{2\pi} \langle -\cos(s), -\sin(s)\rangle\, \phi_i(s)\, ds, \tag{3.20}$$

where ϕ the periodic sinc interpolation function. Notice that the periodic sinc interpolation is defined only for even number of discrete points [33]. If we replace the definition of the periodic sinc and considering $h = 2\pi/n$ we obtain:

$$\dot{\mathbf{x}}_i(t) = -\frac{1}{R(t)} \int_0^{2\pi} \left\langle \cos(s) \frac{\sin(\pi(s-s_i)/h)}{\frac{2\pi}{h}\tan((s-s_i)/2)}, \sin(s) \frac{\sin(\pi(s-s_i)/h)}{\frac{2\pi}{h}\tan((s-s_i)/2)} \right\rangle ds \tag{3.21}$$

$$= -\frac{1}{R(t)} \frac{2\pi}{n} \left\langle \cos\left(\frac{2\pi i}{n}\right), \sin\left(\frac{2\pi i}{n}\right) \right\rangle. \tag{3.22}$$

Under the sinc interpolant, the constant ζ from (3.13) becomes $2\pi/n$. Figure 3.1 shows the evolution of the boundary with 14 points using the velocity equation from (3.22). The discrete data is extracted from a circle and then is interpolated using the sinc periodic interpolator. For this example we set $R_0 = 2$, which implies that the limit time of the simulation is $t_{\lim} = 2$. The theoretical description of the circle matches qualitatively with the interpolated curve, and for simplicity of presentation the theoretical circle is not shown.

Figures 3.2 and 3.3 shows relevant statistics of the experiment. Boundary area is in good agreement with the theoretical area. Area rate of change shows that

whilst we should expect exactly $\frac{dA}{dt} = -2\pi$, we get some oscillations around this value, increasing in magnitude as simulation reaches t_{lim}. Actually the simulation is stopped at some time before reach t_{lim}, because of the mentioned acceleration of the circular boundary. The grain boundary energy under isotropic regime becomes just the perimeter of the circle $P(t) = 2\pi R(t)$. The measured energy of the simulated circle is a monotonic decreasing function in good agreement with the theoretical perimeter obtained from the radii in (3.18), as well as the rate of change of the energy.

3.2.2 Non-regular Boundary

Most grains and their boundaries are not restricted to hold circular shapes. The following example shows a boundary described by a polar rose with an initial condition:

$$\boldsymbol{\xi}(s, 0) = 3 + \cos(3s)$$

using 14 interpolation points. The number of points is chosen merely to resemble the interpolated function. The simulation is performed using the curvature equations from (3.22).

Figure 3.4 shows the evolution of the polar rose. The equations used ensures energy minimization, which induces that the velocity of the discrete points lying in the concave sections points outside the the rose, trying to achieve some iterations later a fully convex form. In the convex form now the points try to approximate a circle shape, which leads to the known evolution presented in Section 3.2.1. Figures 3.5 and 3.6 shows relevant statistics of the experiment. The polar rose area shows qualitatively linear behavior, but a closer look to the derivative shows that there is some perturbation during the minimization, which is soften towards the end of the simulation. The grain boundary energy shows a monotonic decreasing behavior, and its derivative shows a transient where the derivative becomes less negative due to the polar rose correcting its concave parts. After the rose becomes a convex figure, the derivative becomes more and more negative, until numerical collapse at the end of the evolution.

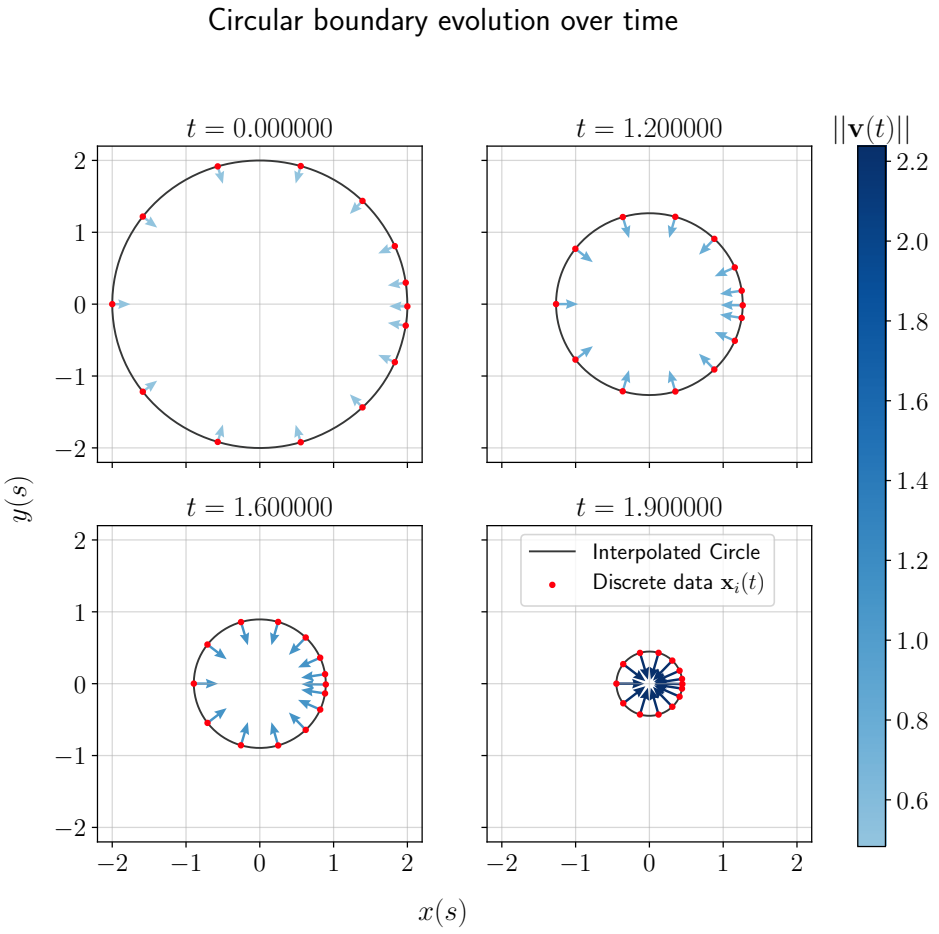
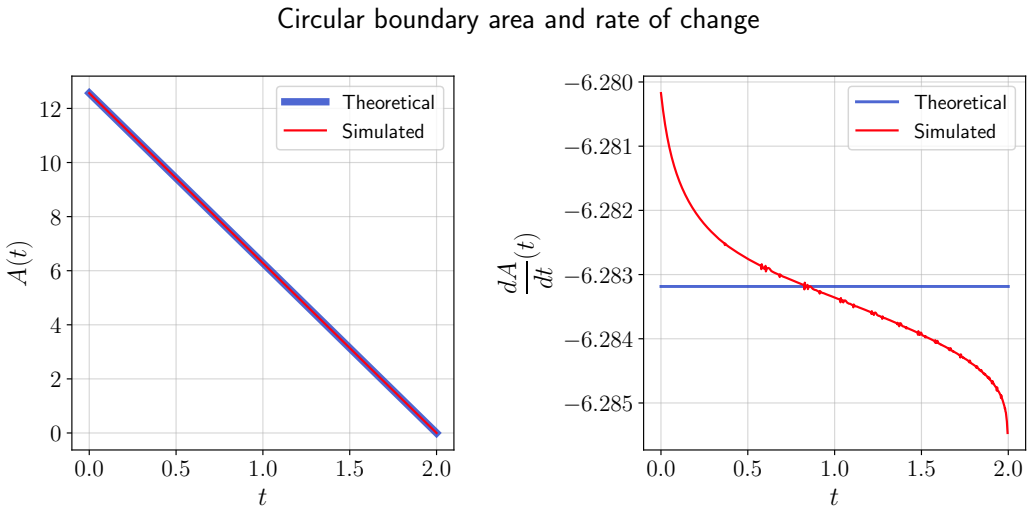
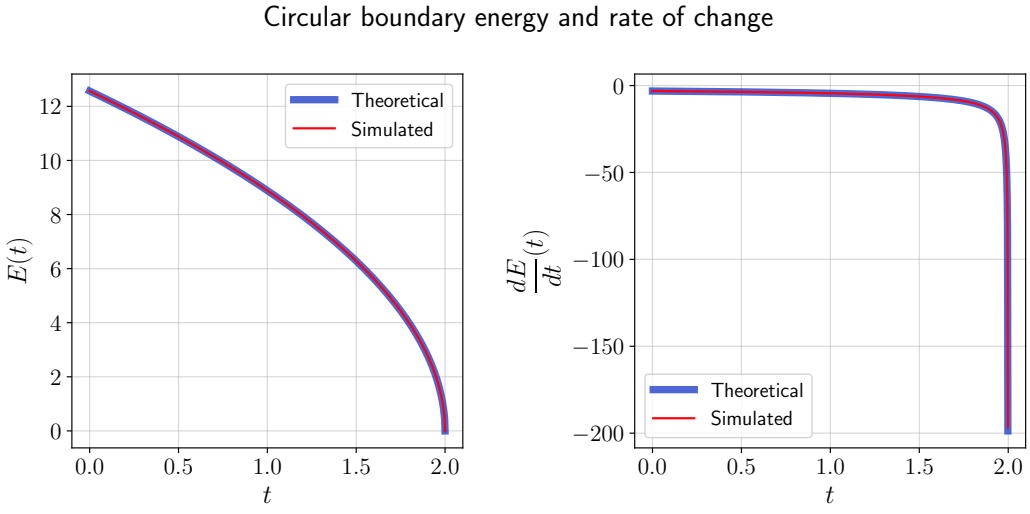


Figure 3.1: Evolution of circular boundary. Velocity vectors are marked in blue gradient, white for the smallest vector and blue for the largest. (Top left) The initial condition at $t = 0$ of the circular boundary with $R_0 = 2$. (Top right, bottom left) After some time the circle start to shrink and the velocity of the boundary increases. (Bottom right) Near $t = 2$ the circle becomes smaller and the velocity vectors increased their magnitude.



(a) (b)

Figure 3.2: (a) Circle area is in good agreement with the theoretical description. (b) Rate of change is not constant in the numerical simulation, but it is near the theoretical value.



(a) (b)

Figure 3.3: (a) Grain energy function is in good agreement with the evolution of the perimeter of the circle. (b) Rate of change of the energy, in agreement with the precipitation of the boundary motion as the radio goes to zero.

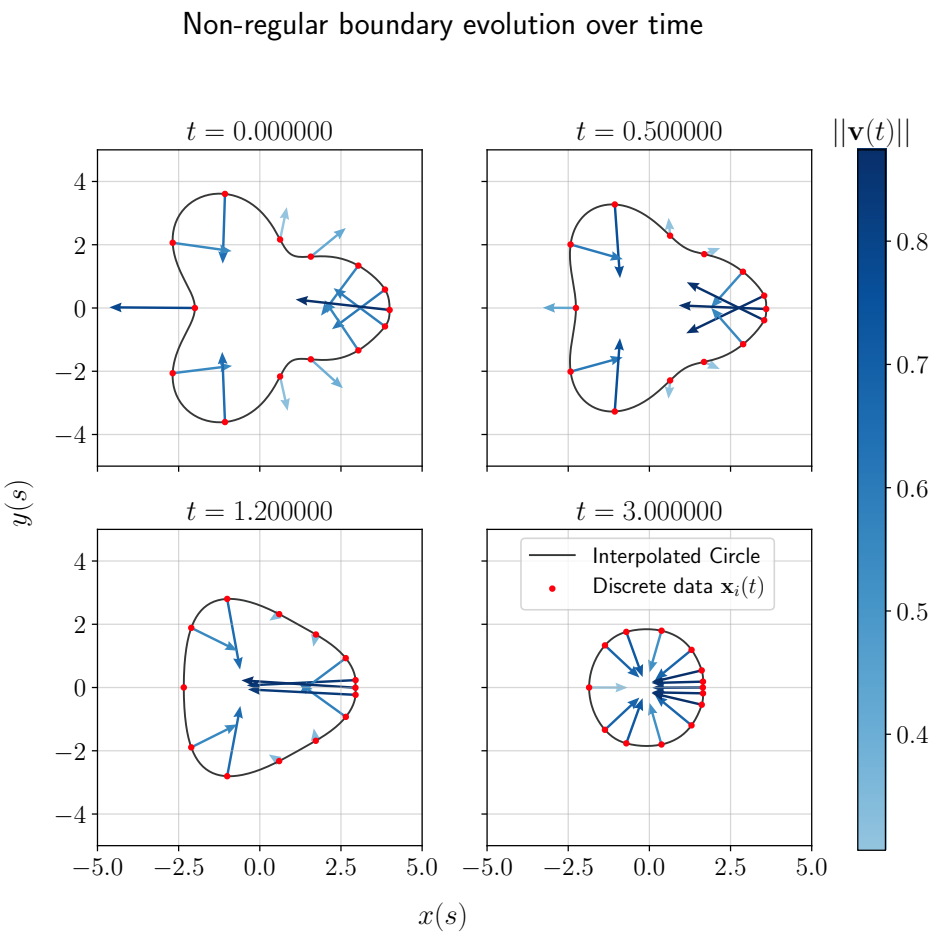
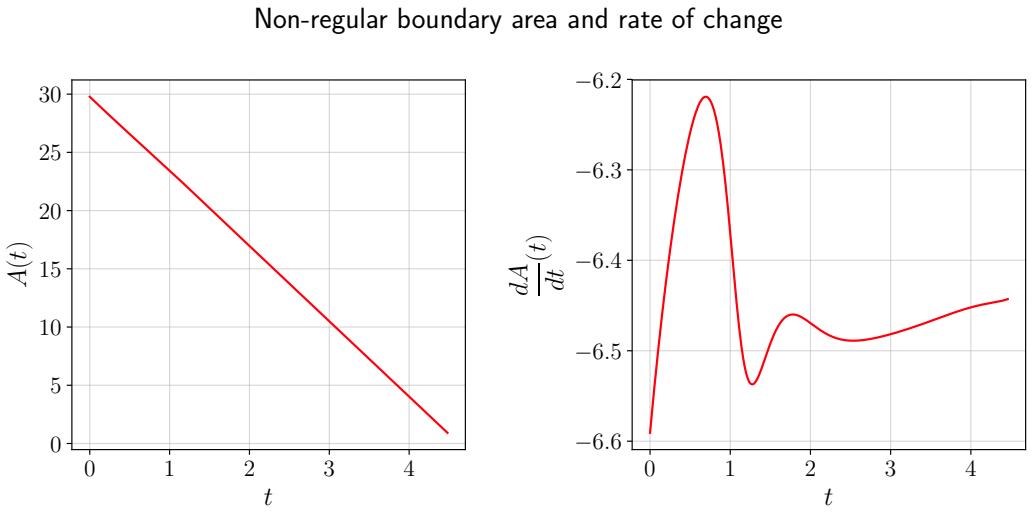
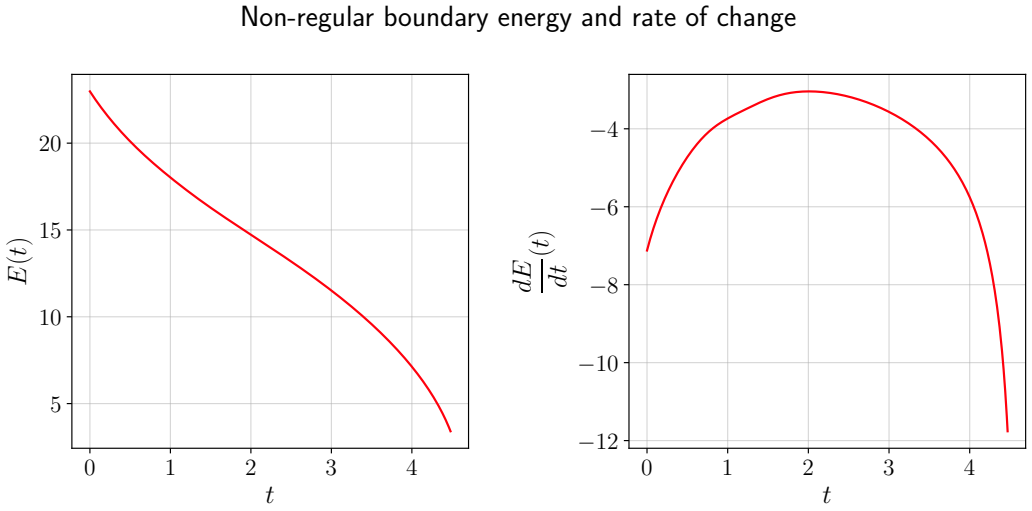


Figure 3.4: Evolution of non-regular boundary. Velocity vectors are marked in blue gradient, white for the smallest vector and blue for the largest. (Top left) The initial condition at $t = 0$ for the polar rose. (Top right) After some time the boundary is driven to correct the curvature while trying to minimize energy. (Bottom left) The boundary is completely convex and all the velocity vectors points to the interior. (Bottom right) The polar rose now has a circle-like shape and the velocity vectors tends to point to the center.



(a) (b)

Figure 3.5: (a) Polar rose area shows qualitatively linear behavior, as expected. (b) Rate of change is not constant in the numerical simulation but tries to stabilize.



(a) (b)

Figure 3.6: (a) Grain energy function shows a monotonic decreasing behavior. (b) Rate of change of the energy collapses at the end of the simulation, meaning that the polar rose is becoming a single point.

Chapter 4

The Coupled Model

NUMERICAL simulations studied by vertex-driven motion and curvature-driven motion as exposed in Chapter 2 can be also studied in a coupled way with the purpose of capturing the dynamics of both models and being able to reproduce each one of them individually. This novel model is called Coupled Model and has been developed in a first version in [27] and improved during the study of the present Thesis. The boundary parametrization proposed here consist in a Lagrange interpolation of n points as follows:

$$\boldsymbol{\xi}^{(k)}(s,t) = \sum_{i=1}^{n} \mathbf{x}_i^{(k)}(t)\,\phi_i(s)\,, \quad s \in [0,1], \tag{4.1}$$

where $\phi_i(s)$ are the Lagrange interpolation functions and the points $\mathbf{x}_1{}^{(k)}$ and $\mathbf{x}_n{}^{(k)}$ are triple junctions. The remaining $n-2$ collocation points are called interior points. For numerical stability each point $\mathbf{x}_i{}^{(k)}$ is parametrized on Chebyshev nodes of the second kind [33] using a linear map $[-1,1] \to [0,1]$ to match the definition of s.

4.1 Derivation of Evolution Equations

4.1.1 Triple Junction Evolution and Normal Component of Interior Points Velocities

We start with the derivative of the total energy of the grain structure in (2.10) and we replace the proposed parametrization as:

$$\frac{dE}{dt}(t) = \int_0^1 -\sum_{k=1}^{K}\left[\sum_{i=1}^{n} \dot{\mathbf{x}}_i^{(k)}(t)\cdot\int_0^1 \gamma^{(k)}\frac{\partial \mathbf{T}}{\partial s}^{(k)}\phi_i(s)\,ds\right] + \sum_{m=1}^{M}\mathbf{v}_m(t)\cdot\sum_{l=1}^{3}\gamma^{(m,l)}\mathbf{T}^{(m,l)}. \tag{4.2}$$

In order to find the motion equations of the system we separate the terms related to triple junctions from the summation over grain boundaries in (4.2) as:

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

Then we add them up to the former triple junction term. Notice that we must choose which contribution (from $\mathbf{x}_1^{(k)}$ or $\mathbf{x}_n^{(k)}$) will be added, since for some arbitrary pair of boundaries $\Gamma^{(k_1)},\Gamma^{(k_2)}$, the initial and end points coincide, i.e., $\mathbf{x}_1^{(k_1)}=\mathbf{x}_n^{(k_2)}$ and we would be adding repeated terms. Re-indexing this contribution in terms of the triple junctions we obtain:

$$\frac{dE}{dt}(t) = -\sum_{k=1}^{K}\left[\sum_{i=2}^{n-1}\dot{\mathbf{x}}_i^{(k)}(t)\cdot\int_0^1\gamma^{(k)}\frac{\partial\mathbf{T}^{(k)}}{\partial s}\phi_i(s)\,ds\right]+\sum_{m=1}^{M}\mathbf{v}_m(t)\cdot\sum_{l=1}^{3}\gamma^{(m,l)}\mathbf{T}^{(m,l)}$$
$$-\sum_{m=1}^{M}\dot{\mathbf{x}}_m(t)\cdot\sum_{l=1}^{3}\int_0^1\gamma^{(m,l)}\frac{\partial\mathbf{T}^{(m,l)}}{\partial s}\phi_{1,m}(s)\,ds, \quad (4.3)$$

where $\mathbf{v}_m=\dot{\mathbf{x}}_m$ The triple junctions velocity is defined as:

$$\dot{\mathbf{x}}_m(t) = -\lambda\sum_{l=1}^{3}\gamma^{(m,l)}\left[\mathbf{T}^{(m,l)}-\int_0^1\frac{\partial\mathbf{T}^{(m,l)}}{\partial s}\phi_{1,m}(s)\,ds\right], \quad (4.4)$$

where λ is the mobility parameter associated to triple junctions.

In the case of the interior points we are interested in obtaining a curvature driven motion behavior as well as to maintain stability for the points. The velocity of interior points $\dot{\mathbf{x}}_i^{(k)}(t)$ is defined to have a normal component, obtained from the curvature study in Chapter 3, and a tangential component that ensures a correct distribution of the interior points along the boundary as:

$$\dot{\mathbf{x}}_i^{(k)}(t) = \alpha_i(t)\,\widehat{\mathbf{T}}_i(t)+\beta_i(t)\,\widehat{\mathbf{N}}_i(t), \quad (4.5)$$

where $\widehat{\mathbf{T}}_i$ and $\widehat{\mathbf{N}}_i$ are the tangential and normal components of the velocities and $\alpha_i,\beta_i\in\mathbb{R}$ are coefficients to be determined, $\widehat{\mathbf{T}}_i\cdot\widehat{\mathbf{N}}_i=0$, and $\|\widehat{\mathbf{T}}_i\|=\|\widehat{\mathbf{N}}_i\|=1$. In the case of the normal component $\beta_i(t)\,\widehat{\mathbf{N}}_i(t)$ it can be obtained directly from (4.3) as:

$$\beta_i(t)\,\widehat{\mathbf{N}}_i(t)=\dot{\mathbf{x}}_{i,\widehat{\mathbf{N}}}^{(k)}(t)=\frac{\mu\gamma^{(k)}}{\|\mathbf{l}(s_i,t)\|}\int_0^1\frac{\partial\mathbf{T}^{(k)}}{\partial s}\phi_i(s)\,ds \quad (4.6)$$

where μ is the mobility for interior points inherited from boundary motion.

4.1.2 Tangential Component of Interior Points Velocities

Defining only a normal component for interior points evolution does not ensure the stability of the boundary since these points might coalesce. In order to minimize energy we can add a tangential component of velocity as long as it is orthogonal to (4.6). Specifically, we can force the interior points to be always equispaced with respect to the total boundary arc length. This has been studied in [14, 3, 4] where the following restriction has to be met:

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

where $\|\mathbf{l}_i(t)\|$ denotes the local arc length at collocation point s_i and $\mathcal{L}(t)$ is the arc length of the boundary under analysis. Without loss of generality we omit the indexation of boundaries from this point and the result applies for all boundaries in the system. Continuing the analysis we obtain the following equation:

$$\begin{aligned}
\frac{\frac{d}{dt}\left(\|\mathbf{l}_i(t)\|\right)\mathcal{L}(t) - \frac{d}{dt}\left(\mathcal{L}(t)\right)\|\mathbf{l}_i(t)\|}{\mathcal{L}^2(t)} &= 0, \\
\frac{d}{dt}\left(\|\mathbf{l}_i(t)\|\right)\mathcal{L}(t) - \frac{d}{dt}\left(\mathcal{L}(t)\right)\|\mathbf{l}_i(t)\| &= 0, \\
\frac{d}{dt}\left(\|\mathbf{l}_i(t)\|\right)\mathcal{L}(t) &= \frac{d}{dt}\left(\mathcal{L}(t)\right)\|\mathbf{l}_i(t)\|, \qquad (4.7)
\end{aligned}$$

where $\frac{d}{dt}\left(\|\mathbf{l}_i(t)\|\right) = \mathbf{T}_i(t) \cdot \frac{d}{dt}\left(\mathbf{l}_i(t)\right)$ and $\frac{d}{dt}\left(\mathcal{L}(t)\right) = \int_0^1 \mathbf{T}(s,t) \cdot \frac{d}{dt}\left(\mathbf{l}(s,t)\right) ds$. Equation (4.7) needs to be applied to each interior point, but before we do that, we need to build each element, obtaining a linear system of equations.

Recalling the definition of the boundary in (4.1) we define the following components:

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

Computation of such components is presented in [27]. Now, it is direct to compute the derivative with respect to time of $\mathbf{l}(s,t)$ and it is also direct to obtain,

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

where $\mathbf{y}_i(t)$ are the components of $\mathbf{l}(s,t)$ obtained via the interpolation of the spectral derivative of $\boldsymbol{\xi}(s,t)$. Thus, plugging in each component to (4.7) we obtain,

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

Moving $\|\mathbf{l}_i(t)\|$ to the right hand side we obtain,

$$\mathbf{T}_i(t)\cdot\dot{\mathbf{y}}_i(t)\frac{\mathcal{L}(t)}{\|\mathbf{l}_i(t)\|} = \sum_{k=1}^{n}\dot{\mathbf{y}}_k(t)\cdot\underbrace{\int_0^1 \mathbf{T}(s,t)\,\phi_k(s)\,ds}_{\mathbf{w}_k(t)},\quad i\in 2,\ldots,n-1. \tag{4.8}$$

Now, it is time to use our orthogonal decomposition (4.5) to obtain the $\alpha_i(t)$, which is achieved by the following linear transformation with the differentiation matrix D that allows numerical differentiation at Chebyshev nodes.

$$\begin{pmatrix}\dot{\mathbf{y}}_1(t)\\ \dot{\mathbf{y}}_2(t)\\ \vdots\\ \dot{\mathbf{y}}_i(t)\\ \vdots\\ \dot{\mathbf{y}}_{n-1}(t)\\ \dot{\mathbf{y}}_n(t)\end{pmatrix} = D\begin{pmatrix}\dot{\mathbf{x}}_1(t)\\ \beta_2(t)\,\widehat{\mathbf{N}}_2(t)+\alpha_2(t)\,\widehat{\mathbf{T}}_2(t)\\ \vdots\\ \beta_i(t)\,\widehat{\mathbf{N}}_i(t)+\alpha_i(t)\,\widehat{\mathbf{T}}_i(t)\\ \vdots\\ \beta_{n-1}(t)\,\widehat{\mathbf{N}}_{n-1}(t)+\alpha_{n-1}(t)\,\widehat{\mathbf{T}}_{n-1}(t)\\ \dot{\mathbf{x}}_n(t)\end{pmatrix} \tag{4.9}$$

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. Recall that $\dot{\mathbf{x}}_1(t)$ and $\dot{\mathbf{x}}_n(t)$ are the triple junctions velocities of the boundary. Thus each component of the system can be written as

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

We now first compute the unknown part and the known part of the left hand side of

(4.9), we omit for now the coefficient $\dfrac{\mathcal{L}(t)}{\|\mathbf{l}_i(t)\|}$:

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

Similarly for the right-hand-side of (4.8) we obtain the following for the k-th element of the sum,

$$\begin{aligned}
\dot{\mathbf{y}}_k(t)\cdot\mathbf{w}_k(t) &= D_{k,1}\,\mathbf{w}_k(t)\cdot\dot{\mathbf{x}}_1(t)+\sum_{j=2}^{n-1}D_{k,j}\,\left(\beta_j(t)\,\mathbf{w}_k(t)\cdot\widehat{\mathbf{N}}_j(t)+\alpha_j(t)\,\mathbf{w}_k(t)\cdot\widehat{\mathbf{T}}_j(t)\right)\\
&\quad+D_{k,n}\,\mathbf{w}_k(t)\cdot\dot{\mathbf{x}}_n(t)\\
&= \sum_{j=2}^{n-1}D_{k,j}\,\alpha_j(t)\,\mathbf{w}_k(t)\cdot\widehat{\mathbf{T}}_j(t)\\
&\quad+\left(D_{k,1}\,\mathbf{w}_k(t)\cdot\dot{\mathbf{x}}_1(t)+\sum_{j=2}^{n-1}D_{k,j}\,\beta_j(t)\,\mathbf{w}_k(t)\cdot\widehat{\mathbf{N}}_j(t)+D_{k,n}\,\mathbf{w}_k(t)\cdot\dot{\mathbf{x}}_n(t)\right).
\end{aligned}$$

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

$$\begin{aligned}
&\frac{\mathcal{L}(t)}{\|\mathbf{l}_i(t)\|}\sum_{j=2}^{n-1}D_{i,j}\,\alpha_j(t)\,\mathbf{T}_i(t)\cdot\widehat{\mathbf{T}}_j(t)-\sum_{k=1}^{n}\sum_{j=2}^{n-1}D_{k,j}\,\alpha_j(t)\,\mathbf{w}_k(t)\cdot\widehat{\mathbf{T}}_j(t)=\\
&-\frac{\mathcal{L}(t)}{\|\mathbf{l}_i(t)\|}\left(D_{i,1}\,\mathbf{T}_i(t)\cdot\dot{\mathbf{x}}_1(t)+\sum_{j=2}^{n-1}D_{i,j}\,\beta_j(t)\,\mathbf{T}_i(t)\cdot\widehat{\mathbf{N}}_j(t)+D_{i,n}\,\mathbf{T}_i(t)\cdot\dot{\mathbf{x}}_n(t)\right)\\
&+\sum_{k=1}^{n}\left(D_{k,1}\,\mathbf{w}_k(t)\cdot\dot{\mathbf{x}}_1(t)+\sum_{j=2}^{n-1}D_{k,j}\,\beta_j(t)\,\mathbf{w}_k(t)\cdot\widehat{\mathbf{N}}_j(t)+D_{k,n}\,\mathbf{w}_k(t)\cdot\dot{\mathbf{x}}_n(t)\right).
\end{aligned}$$

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

$$\sum_{j=2}^{n-1}\left(D_{i,j}\frac{\mathcal{L}(t)}{\|\mathbf{l}_i(t)\|}\,\mathbf{T}_i(t)-\sum_{k=1}^{n}D_{k,j}\mathbf{w}_k(t)\right)\cdot\widehat{\mathbf{T}}_j(t)\,\alpha_j(t)=$$
$$-\frac{\mathcal{L}(t)}{\|\mathbf{l}_i(t)\|}\left(D_{i,1}\,\dot{\mathbf{x}}_1(t)+\sum_{j=2}^{n-1}D_{i,j}\,\beta_j(t)\,\widehat{\mathbf{N}}_j(t)+D_{i,n}\,\dot{\mathbf{x}}_n(t)\right)\cdot\mathbf{T}_i(t)$$
$$+\sum_{k=1}^{n}\left(D_{k,1}\,\dot{\mathbf{x}}_1(t)+\sum_{j=2}^{n-1}D_{k,j}\,\beta_j(t)\,\widehat{\mathbf{N}}_j(t)+D_{k,n}\,\dot{\mathbf{x}}_n(t)\right)\cdot\mathbf{w}_k(t).$$

Thus, we have shown how to obtain the coefficient $\alpha_i(t)$. They need to be computed explicitly every time the interior points will move or when we need to compute quantities that depend on their velocity, such as the extinction time of each grain boundary.

4.2 Numerical Implementation

4.2.1 Estimation of Extinction Time

The extinction time t_{ext} is defined as the time when a boundary collapses and its arc length becomes zero. Consider the arc length of a boundary as a function of time $\mathcal{L}(t)$. The arc length at time $t+\Delta t$ can be estimated as

$$\mathcal{L}(t+\Delta t)=\mathcal{L}(t)+\Delta t\,\mathcal{L}'(t)+\mathcal{O}((\Delta t)^2).$$

To determine if a boundary is shrinking or not we need to compute $\mathcal{L}'(t)$. If $\mathcal{L}'(t)$ is negative it shrinks. If we consider that the boundary collapses when $\mathcal{L}(t+t_{\text{ext}})=0$ we can estimate the extinction time as follows:

$$0\approx\mathcal{L}(t)+t_{\text{ext}}\,\mathcal{L}'(t)$$
$$t_{\text{ext}}\approx-\frac{\mathcal{L}(t)}{\mathcal{L}'(t)}.$$

An important result of the study of extinction time is that topological transitions are delayed because the extinction time estimation has an error proportional to the simulation time step Δt. Figure 4.1 shows extinction times over time for a boundary that has to flip but nevertheless when the extinction time is close to Δt the estimation fails and the boundary cannot flip. Even if we decrease Δt the flipping is delayed. This analysis suggests that the continuous motion of triple junctions and boundaries has to be more accurate than estimation of the extinction time. The first idea to obtain a better approximation of extinction time is to perform r steps of size $\Delta\tau$ such

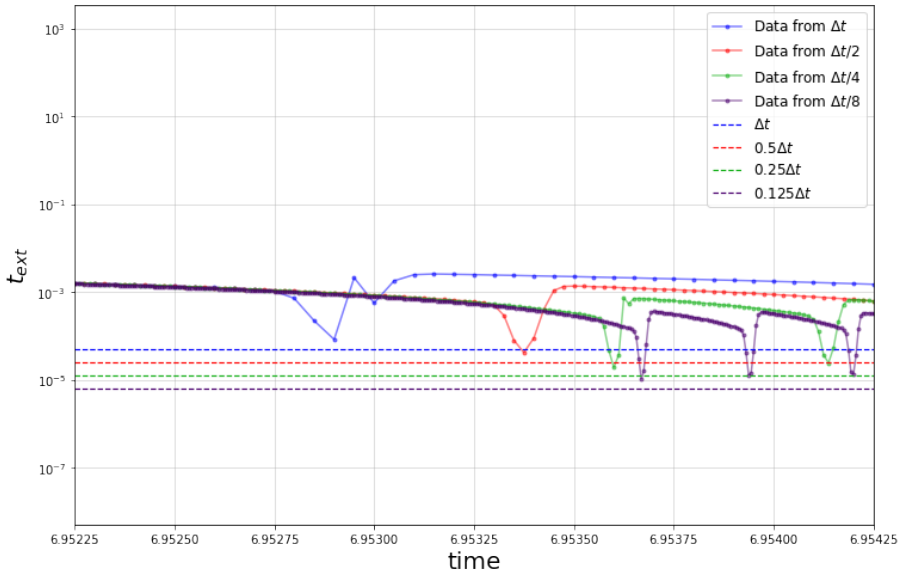


Figure 4.1: Example of a delayed flipping. Estimation of t_{ext} is proportional to to the simulation Δt. As we decrease Δt (dashed lines) we improve t_{ext} but when t_{ext} approaches Δt the estimation may fail, thus flippings are delayed or never detected.

that $r\Delta\tau = \Delta t$. During the execution of this steps, the basic assumption is that no topological transition occurs and thus the system just evolves. Using a smaller time step helps to ensure a better approximation. On the other hand, straightforward higher order methods can be used. For example second order Runge-Kutta ensures that the grain system evolves with precision $\mathcal{O}(\Delta t^2)$, and therefore the extinction time has the same precision. Runge-Kutta can also be improved by the introduction of many steps of smaller size. Both methods must detect the upcoming topological transitions and after the evolution is complete the transitions are handled.

4.2.2 Multistep Euler

This method assumes that Euler method can be improved if steps of smaller size are performed to evolve the grain structure while handling topological transitions. A number r of smaller steps of size $\Delta\tau$ is chosen, where $\Delta\tau = \Delta t/r$. The execution of this r steps is called the *multistep stage* and the system evolves assuming that there are not topological transitions. Algorithm 4.1 shows this method.

Using a smaller step-size helps to ensure a better approximation of the extinction time proportional to $\mathcal{O}(\Delta\tau)$.

No extra memory is required for this method, but the cost is purely computational since the method is performing r steps per main step and thus is almost r times slower. Notice that when $r = 1$, we recover the original Forward Euler method.

Algorithm 4.1 Multistep Euler for Coupled Model

procedure ME
  $\Delta\tau \leftarrow \dfrac{\Delta t}{r}$ Time step of multistep phase.
  for $k : 1, \ldots, r$ **do**
    $\mathbf{V}_t \leftarrow$ Compute velocities
    $\boldsymbol{\mathcal{X}}_{t+\Delta\tau} \leftarrow \boldsymbol{\mathcal{X}}_t + \Delta\tau \mathbf{V}_t$
    $t \leftarrow t + \Delta\tau$
  end for
end procedure

4.2.3 Multistep Second Order Runge-Kutta

If the goal is to obtain a good approximation of the extinction time, a higher order method can be implemented straightforward. For example extinction times with precision $\mathcal{O}(\Delta t^2)$ can be estimated using second order Runge-Kutta (RK2). We can also improve this method by means of introducing the multistep idea to perform several RK2 steps within $[t, t + \Delta t]$ as shown in Algorithm 4.2.

Algorithm 4.2 Multistep Second Order Runge-Kutta for Coupled Model

procedure MRK2
  $\Delta\tau \leftarrow \dfrac{\Delta t}{r}$ Time step of multistep phase.
  for $k : 1, \ldots, r$ **do**
    $\overline{\boldsymbol{\mathcal{X}}}_t \leftarrow$ Backup positions $\boldsymbol{\mathcal{X}}_t$ of triple junctions and interior points
    $\mathbf{V}_t \leftarrow$ Compute velocities
    $\boldsymbol{\mathcal{X}}_{t+\Delta\tau/2} \leftarrow \boldsymbol{\mathcal{X}}_t + \dfrac{\Delta\tau}{2}\mathbf{V}_t$. Evolve structure for first RK estimation
    $\mathbf{V}_{t+\Delta\tau/2} \leftarrow$ Compute velocities
    $\boldsymbol{\mathcal{X}}_{t+\Delta\tau} \leftarrow \overline{\boldsymbol{\mathcal{X}}}_t + \Delta\tau \mathbf{V}_{t+\Delta\tau/2}$. Evolve structure for second RK estimation
    $t \leftarrow t + \Delta\tau$
  end for
end procedure

The cost of this method lies in the memory needed to store the extra data for performing the two estimations at time $\Delta\tau/2$ and $\Delta\tau$ and the number of steps r. Notice that when $r = 1$ we recover the original RK2. In this implementation it is only necessary to backup the positions of the vertices and interior points to be used in the last step of the method and not the whole data structure i.e., arc lengths, curvatures, grain areas, etc.

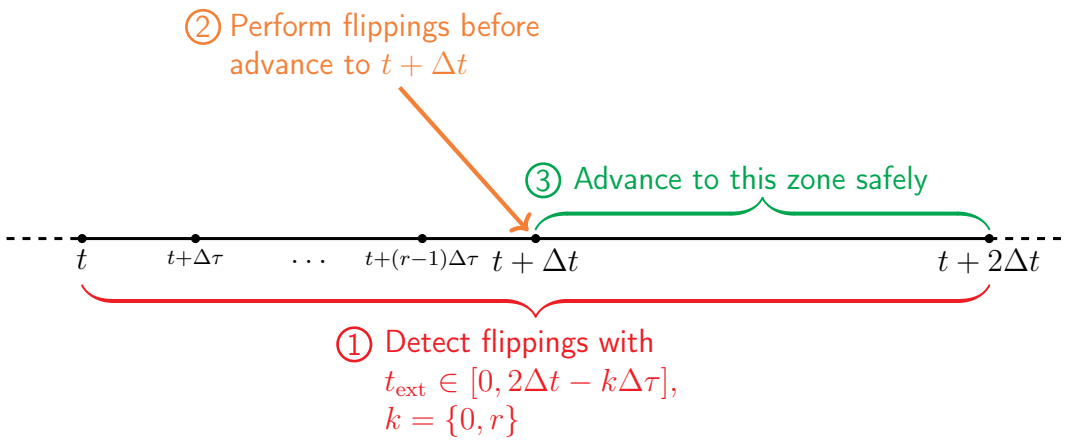


Figure 4.2: Flipping detection mechanism that prevents delay in topological transitions.

4.2.4 Extinction Time Estimation with Multistep Methods

Given the proposed multistep methods to evolve the grain network, a final step is required in the analysis. The question here is: What is the correct interval that we should study to detect upcoming flippings? When we start a time simulation at t, we need to detect all topological transitions so that, once they are resolved, we can start the next step $t + \Delta t$ safely.

Starting from a time t, we should first look ahead for all upcoming flippings between $[0, 2\Delta t]$, since those events have to be resolved before advance to $t + \Delta t$, otherwise we might be skipping their resolution and delaying them unnecessarily.

The next step, which is an inner step $t + \Delta\tau$ has to detect flippings in the smaller interval $[0, 2\Delta t - \Delta\tau]$ since we just advanced τ. Notice that we are just detecting and not performing the flippings.

We continue this method during the r inner steps. Before reaching time $t + \Delta t$ we collect the detected flippings and perform only those which will not introduce race conditions, i.e., flippings that will not modify the same neighborhood of vertices and boundaries. This is addressed in Chapter 6. After the topological transitions are solved, we are ready to start with a safe structure at time $t + \Delta t$. Figure 4.2 shows an sketch of the detection algorithm used with the multistep algorithms.

4.2.5 Predictor-Corrector Algorithm

The tangential component $\widehat{\mathbf{T}}_i$ of the total velocity of interior points, explained in Section 4.1.2, helps to avoid the interior points coalescing and its computation depends partially on the normal component $\beta_i(t)\,\widehat{\mathbf{N}}_i$. When the boundaries are shrinking and close to collapse, the curvature of the boundary grows a lot and thus the vector $\beta_i(t)\,\widehat{\mathbf{N}}_i$ is large, requiring a very small Δt to perform a stable evolution. The

computation of $\widehat{\mathbf{T}}_i$ depends on $\beta_i(t)\,\widehat{\mathbf{N}}_i$ and thus also grows. The magnitude of these vectors increases and leads to unstable boundaries with irregular shapes and delayed flippings since the unstable boundaries actually grew and may present negative extinction times, (i.e., the boundary grows) or large positive extinction times (the boundary will collapse in a time greater than Δt). Thus we must be able to detect when this behavior will happen prior to the boundary collapse without assuming anything about the boundary shape and just looking to the total velocities.

Considering the sketch in Figure 4.3a, which shows a boundary with two interior points $\mathbf{x}_2(t), \mathbf{x}_3(t)$. Each interior point has their velocities $\dot{\mathbf{x}}_2(t), \dot{\mathbf{x}}_3(t)$, respectively. We evolve the triple junctions given their current velocities and now they move to the locations $\mathbf{x}_1(t+\Delta t)$ and $\mathbf{x}_4(t+\Delta t)$. With this information we want to predict where the interior points will be at time $t+\Delta t$.

The prediction stage of the algorithm considers the movement of the interior points using the current velocities, this position $\widetilde{\mathbf{x}}_i$ is called the *prediction*:

$$\widetilde{\mathbf{x}}_i = \mathbf{x}_i(t) + \Delta t\,\dot{\mathbf{x}}_i(t). \tag{4.10}$$

Figure 4.3b shows the predictions made for the interior points. Interior point $\mathbf{x}_2(t)$ will be far from the steady state whilst interior point $\mathbf{x}_3(t)$ will be located before the displacement limit.

The correction stage of the algorithm now has to check the following cases given the predictions of the prior stage:

- The prediction is beyond the steady state. Since moving to that position generates instability in the boundary and given the fact that we know the steady state line, we set the interior point to evolve at the intersection of the displacement and the steady state. For example, interior point $\mathbf{x}_2$ shall move back to the position in red over the line given by its velocity vector.

- The prediction is between the steady state and the current location of the interior point position, so we let the point evolve to that position. For example, interior point $\mathbf{x}_3$ can safely evolve to the prediction located at $\widetilde{\mathbf{x}}_3$.

- The prediction is before the steady state and before the interior point position. This case indicates that for some reason the point is moving away from the steady state and we force the velocity to be zero in order to stop any threat to the stability of the computation.

The cases can be algebraically analyzed since we are interested in finding for each interior point the intersections of two straight lines, one line formed by the two vertices at $t+\Delta t$ parametrized by ω, and the other line formed by the interior point at time t and its respective velocity vector at time t. Each of this $n-2$ lines can be parametrized by $\omega_i,\ i = 2, \ldots, n-1$. The intersection can be written as

$$\mathbf{x}_i + \omega_i\,\dot{\mathbf{x}}_i = (1-\omega)\,\mathbf{x}_1 + \omega\,\mathbf{x}_n \tag{4.11}$$

Algorithm 4.3 Predictor-Corrector

```
1: procedure PRED-CORR(boundary Γ, TOL)
2:     (x_1, x_n) ← First and Last vertex of Γ
3:     for i = 2, ..., n − 1 do
4:         (x_i, ẋ_i) ← i-th interior point and its total velocity
5:         M ← GetSystemMatrix(x_1, x_n, ẋ_i)   ▷ Notice it is always a 2 × 2 matrix.
6:         if |det(M)| > TOL then
7:             b ← x_1 − x_i
8:             ω_i ← Solve Mω_i = b
9:             ω_i ← First component of ω_i.
10:            ω_i ← ω_i/Δt
11:            if ω_i < 1 then
12:                if ω_i > 0 then
13:                    ẋ_i ← ω ẋ_i
14:                else
15:                    ẋ_i ← 0
16:                end if
17:            end if
18:        end if
19:    end for
20: end procedure
```

We write (4.11) as the following system of linear equations:

$$\underbrace{\left(\dot{\mathbf{x}}_i \mid \mathbf{x}_1 - \mathbf{x}_n\right)}_{M} \boldsymbol{\omega}_i = \underbrace{\left(\mathbf{x}_1 - \mathbf{x}_i\right)}_{\mathbf{b}},$$

where $\boldsymbol{\omega}_i = (\omega_i, \omega)^T$ is the intercept vector for each pair of lines. Since we need to move along the line given by the velocity of the interior point, we are interested in the first element of $\boldsymbol{\omega}_i$ which is ω_i. Notice that ω_i is a displacement, and thus it is divided by Δt. The velocity $\widetilde{\dot{\mathbf{x}}}_i(t)$ is what we call the *correction*.

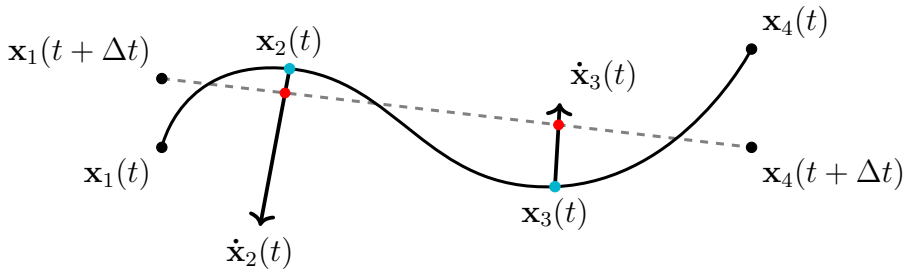
Figure 4.3c shows the corrections made to the interior points velocities. The velocity $\dot{\mathbf{x}}_2(t)$ had to be scaled by $\omega_2/\Delta t$ such that the displacement of $\mathbf{x}_2(t)$ is on the steady state limit, that is

$$\widetilde{\dot{\mathbf{x}}}_2(t) = \frac{\omega_2}{\Delta t}\,\dot{\mathbf{x}}_2(t).$$

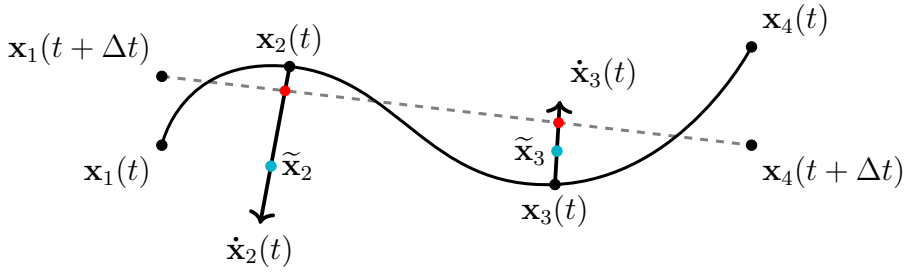
The velocity $\dot{\mathbf{x}}_3(t)$ remains the same since the interior point prediction does not cross the steady state.

Figure 4.3d shows the applied corrections to the interior points movement. Notice that $\mathbf{x}_2$ was moved to the displacement limit whilst $\mathbf{x}_3$ moved without correction.

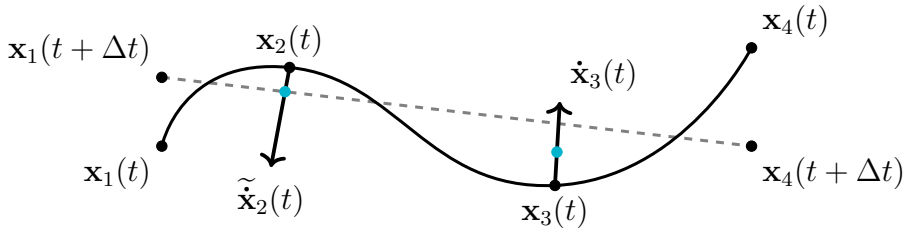
Algorithm 4.3 summarizes the predictor-corrector algorithm that has to be applied to each boundary. We need to solve the system safely, thus we first check if



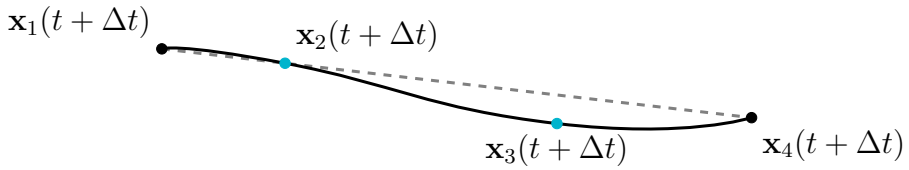
(a) Sketch of a boundary with two interior points marked in blue. The *candidate* steady state is marked in red on the dashed gray line. The location of the interior points on the next time-step shall not cross the *candidate* steady state in red.



(b) The prediction stage. Interior points predict their location at next time-step, marked in blue, given the current total velocities.



(c) The interior point velocities are corrected such that their location at next time-step, marked in blue, is at most the *candidate* steady state locations from Figure 4.3b in red. The correction here corresponds to modify the velocity $\dot{\mathbf{x}}_2(t)$ to $\widetilde{\dot{\mathbf{x}}}_2(t)$ as indicated in Algorithm 4.3. The velocity $\dot{\mathbf{x}}_3(t)$ remains the same.



(d) The correction stage. Interior points are moved with corrections, if apply. At this moment the boundary points are safe at $t+\Delta t$.

Figure 4.3: Sketch of the Predictor-Corrector Algorithm.

the system can be solved by checking the determinant of the system matrix M is different from zero given certain tolerance TOL. The cases when M is singular can be analyzed by looking at its columns. If the total velocity of an interior point is $\mathbf{0}$, the first column of M will be null. If the triple junctions are numerically in the same location we will have the second column of M to be null. We set the tolerance TOL to 10^{-12}. Finally, M will be a singular matrix if the velocity is parallel to the steady state line. In all of these cases there is no need for a correction.

We also need to check a valid value for ω_i. If $\omega_i > 1$ we know that the interior point will not pass over the steady state and we do not apply any correction. If $0 < \omega_i < 1$ we need to apply a correction proportional to ω_i. Here we are interested in update the velocity of the interior point and thus we divide ω_i by Δt. If $\omega_i < 0$ the interior point is moving away from the steady state and we set the velocity of the boundary to zero to avoid that point to escape from the steady state.

4.3 Numerical Experiments

4.3.1 Energy Minimization

The Coupled Model defines several parameters such as the number of interior points n, triple junctions mobility λ, interior points mobility μ, the number of steps for the multistep algorithms r and the step size Δt. With the introduction of the total velocity of interior points as a combination of a normal component and a tangential component, it is important to maintain the energy decreasing behavior of the simulation. Figure 4.4 shows the total energy of a simulation of initial 1,000 grains with parameters $\mu = 10^{-2}, \lambda = 1$ and two interior points. following the energy definition in (2.8). The behavior despite adding a new component to the velocities remains monotonically decreasing.

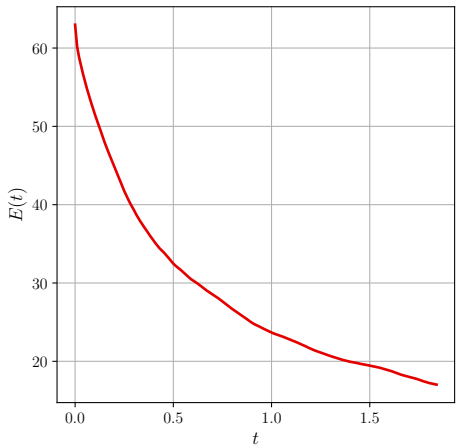


Figure 4.4: Total energy of the grain network with the Coupled Model.

4.3.2 Statistics

All experiments were run with initial 100,000 grains, using Euler Multistep with $r = 4$ and $n = 2$ interior points. Figure 4.5 shows the statistics for following simulations:

- Experiment 1 (red): $\mu = 10^{-3}, \lambda = 1, \varepsilon = 0, \Delta t = 10^{-5}$.
- Experiment 2 (green): $\mu = 10^{-1}, \lambda = 10^{-1}, \varepsilon = 2 \cdot 10^{-2}, \Delta t = 10^{-5}$.
- Experiment 3 (blue): $\mu = 10^{-2}, \lambda = 1, \varepsilon = 2 \cdot 10^{-1}, \Delta t = 10^{-4}$
- Experiment 4 (yellow): $\mu = 10^{-1}, \lambda = 1, \varepsilon = 0, \Delta t = 10^{-5}$

Figure 4.5a shows the relative area distribution. We clearly observe that the experiment with $\mu = 10^{-3}$ in red has few small grains, in contrast to the experiment with equal mobilities in green which shows a long tail in the log-scale plot related to small grains. Experiments in blue and yellow, which varies μ, effectively shows that reducing this parameter implies a reduction in the number of small grains.

Figure 4.5b shows the dihedral angle distribution. Again we see the most dispersed distribution belongs to the experiment with $\mu = \lambda = 10^{-1}$ in green, whilst the experiment in red, related to $\mu = 10^{-3}$ tends to be accumulated around the stationary state angle $2\pi/3$.

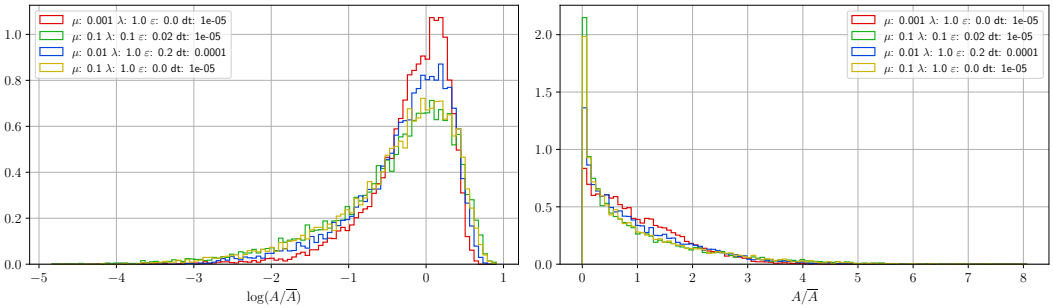
Figure 4.5c shows the distribution of grains per number of sides. The relation is clear and in concordance with the presented distributions. The red experiment has the lowest number of three sided grains and as we increase μ until the green curve we see how we increase that number. The mode of the distribution increases from five to six as we reduce μ.

Figure 4.5d shows the average number of neighbor distribution per grain class. As we increase the values of μ the average number of sides of neighbors increases.

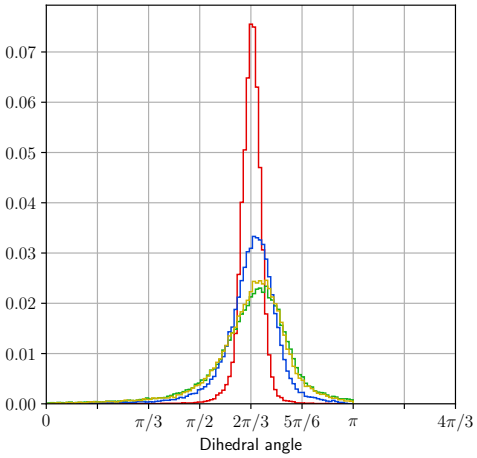
Figure 4.5e shows the area rate of change per class scaled by μ^{-1}. This scaling is applied to observe the concordance with the Von Neumann-Mullins $n-6$ relation from (2.12) which we recall here:

$$\frac{dA}{dt}(\mathcal{G}) \sim \mu(\mathrm{ns}(\mathcal{G}) - 6),$$

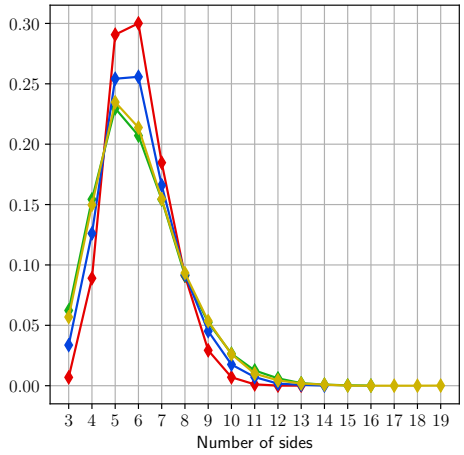
where $\mathrm{ns}(\mathcal{G})$ is the number of sides or class of grain $\mathcal{G}$. Experiments with high values of μ, in green and yellow, shows slow rates that increases non-linearly as we look for large grain classes. The experiment with $\mu = 10^{-2}$ and $\lambda = 1$, tries to rectify its tendency to linearity, although it does not shows a linear behavior for grains of low class. The red experiment with the smallest μ has the most acceptable linear behavior in concordance with the $n-6$ relation.



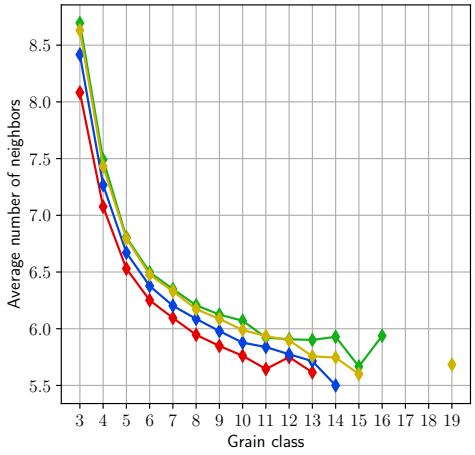
(a) Relative areas in log scale (left) and linear scale (right).



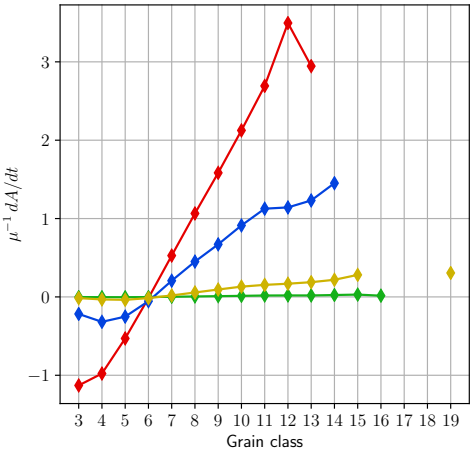
(b) Dihedral angle.



(c) Number of sides



(d) Average number of sides of neighbors.



(e) Area rate of change per class. The rate is scaled by μ^{-1}.

Figure 4.5: Distributions for Coupled Model numerical simulations.

Chapter 5

A Continuous Stored Energy Vertex Model with Nucleation

Extensions of the classic Vertex Model [32] considers the introduction of an intragranular stored energy $\mathcal{E}$ which plays a key role in primary recrystallization, that is, the generation of new grains before the grain growth regime [21, 22]. This process will significantly modify the material and produce drastic changes of many physical properties like electrical resistance, internal stresses, micro-hardness, among others, see [21, 22]. Vertex Model implementations defining the structure as a graph, as presented in Chapter 2, are common but models that include nucleation are rare since the data structure does not allow dislocation inside grains [19].

In this chapter, we first introduce the evolution equations of vertices considering the stored energy term and then we analyze the nucleation process.

Following the notation from [22], the local energy of a vertex i considering the stored energy term is defined as:

$$E_i = \sum_{j \in \mathcal{N}_i} \left(\gamma^{(i,j)} \mathcal{L}_{i,j} + \mathcal{E}_{i,j} A_{i,j} \right),$$

where $\mathcal{N}_i$ is the set of neighbor vertices to the vertex i, $\gamma^{(i,j)}$ and $\mathcal{L}_{i,j}$ are the grain boundary energy and the arc length of the boundary formed by vertices i and j, $\mathcal{E}_{i,j}$ and $A_{i,j}$ is the stored energy and area of a grain adjacent to the boundary i, j using the right-hand rule. We split the local energy between the grain boundary contribution and the grain area contribution, so:

$$E_i = \sum_{j=1}^{3} \gamma^{(k_{i,j})} \mathcal{L}_{k_{i,j}} + \sum_{j=1}^{3} \mathcal{E}_{g_{i,j}} A_{g_{i,j}},$$

where $k_{i,\cdot}$ and $g_{i,\cdot}$ are indices for the three boundaries and grains relative to vertex i. Now, we could define the total energy of the system as:

$$E = \sum_{k=1}^{K} \gamma^{(k)} \mathcal{L}_k + \sum_{g=1}^{N} \mathcal{E}_g A_g. \tag{5.1}$$

where K and N are total number of boundaries and grains respectively. So, now we could decompose the contribution from each vertex to the total energy as follows,

$$\widehat{E}_i = \sum_{j=1}^{3} \gamma^{(k_{i,j})} \frac{\mathcal{L}_{k_{i,j}}}{2} + \sum_{j=1}^{3} \mathcal{E}_{g_{i,j}} \frac{A_{g_{i,j}}}{\mathrm{ns}(g_{i,j})}, \tag{5.2}$$

where $\mathrm{ns}(g)$ is the number of sides (or class) of grain g. This allows us to build the total energy as the sum of the contribution of each vertex such that we ensure we only consider once each grain boundary contribution and also consider once each grain contribution, thus,

$$\sum_{i=1}^{M} \widehat{E}_i = \underbrace{\sum_{i=1}^{M} \sum_{j=1}^{3} \gamma^{(k_{i,j})} \frac{\mathcal{L}_{k_{i,j}}}{2}}_{(a)} + \underbrace{\sum_{i=1}^{M} \sum_{j=1}^{3} \mathcal{E}_{g_{i,j}} \frac{A_{g_{i,j}}}{\mathrm{ns}(g_{i,j})}}_{(b)}$$

In order to compute (a) we need to understand which elements are being counted. Consider an specific vertex i and their related boundaries. The expansion of (a) for vertex i looks like:

$$\gamma^{(k_{i,1})} \frac{\mathcal{L}_{k_{i,1}}}{2} + \gamma^{(k_{i,2})} \frac{\mathcal{L}_{k_{i,2}}}{2} + \gamma^{(k_{i,3})} \frac{\mathcal{L}_{k_{i,3}}}{2}.$$

The same term for some neighbor vertex looks like:

$$\gamma^{(k_{1,i})} \frac{\mathcal{L}_{k_{1,i}}}{2} + \gamma^{(k_{1,\cdot})} \frac{\mathcal{L}_{k_{1,\cdot}}}{2} + \gamma^{(k_{1,\cdot})} \frac{\mathcal{L}_{k_{1,\cdot}}}{2}.$$

It is clear that for each vertex we are counting the grain boundary energy and arc length twice, each time per vertex in a boundary. This justifies the introduction of the term $1/2$ in (a). The analysis for (b) holds a similar argument. The inner sum of here runs over the grains related to vertex i. We can build a similar example starting from a vertex i and counting the three stored energy terms related as:

$$\mathcal{E}_{g_{i,1}} \frac{A_{g_{i,1}}}{\mathrm{ns}(g_{i,1})} + \mathcal{E}_{g_{i,2}} \frac{A_{g_{i,2}}}{\mathrm{ns}(g_{i,2})} + \mathcal{E}_{g_{i,3}} \frac{A_{g_{i,3}}}{\mathrm{ns}(g_{i,3})}.$$

Consider now that grain $g_{i,1}$ has $\mathrm{ns}(g_{i,1})$. We already know that vertex i forms part of that grain. The other $\mathrm{ns}(g_{i,1}) - 1$ vertices of $g_{i,1}$ have also in their stored energy terms the term $\mathcal{E}_{g_{i,1}} \dfrac{A_{g_{i,1}}}{\mathrm{ns}(g_{i,1})}$ and thus we are adding the same term $\mathrm{ns}(g_{i,1})$ times. This justifies the introduction of the term $1/\mathrm{ns}(g)$. Finally we can map the sum in order to count boundaries and grains. Each term related to a boundary is counted twice and each term related to a grain is counted $\mathrm{ns}(g)$ times and we can recover an

expression for the total energy of the system as:

$$\begin{aligned}\sum_{i=1}^{M}\widehat{E}_i &= \sum_{k=1}^{K} 2\cdot\gamma^{(k)}\frac{\mathcal{L}_k}{2} + \sum_{g=1}^{N}\mathrm{ns}(g)\cdot\mathcal{E}_g\frac{A_g}{\mathrm{ns}(g)}\\ &= \sum_{k=1}^{K}\gamma^{(k)}\mathcal{L}_k + \sum_{g=1}^{N}\mathcal{E}_g A_g\\ &= E.\end{aligned}$$

If we explicit the dependency of the boundary arc lengths and grain areas with respect to the vertices positions $\{\mathbf{x}_i = \langle x_i, y_i\rangle\}$ we can build the evolution equations for this model such that it decreases its energy by a gradient descent method, that is to compute $\dot{\mathbf{x}}_i(t) = \left\langle -\frac{\partial E}{\partial x_i}, -\frac{\partial E}{\partial y_i}\right\rangle$.

5.1 Numerical Algorithm

Computing the *velocity* of each vertex is actually the computation of the gradient of the energy E from equation (5.1). Here we propose a matrix-free approach to approximate the gradient. The main advantage of this approach is that it only needs the implementation of computation of the total energy of the system, this greatly simplifies the bookkeeping we would need to handle this task.

A convenient way to approximate each partial derivative is with a matrix-free approach. Consider the vector $\mathbf{X}\in\mathbb{R}^{2M}$ of stacked components of $\mathbf{x}_i$, this is:

$$\mathbf{X} = (x_1, x_2, \cdots, x_M, y_1, y_2, \cdots, y_M)^T,$$

where M is the total number of triple junctions and ^{T} is the transpose operator. Each m-th component of the gradient for this vector, say $\dfrac{\partial E(\mathbf{X})}{\partial \mathbf{X}_m}$ is approximated by:

$$\frac{\partial E(\mathbf{X})}{\partial \mathbf{X}_m} \approx \frac{E(\mathbf{X}+\varepsilon\,\mathbf{e}_m) - E(\mathbf{X})}{\varepsilon}, \tag{5.3}$$

where $\mathbf{e}_m$ is the m-th canonical vector in $\mathbb{R}^{2M}$. The numerical evaluation of this approximation implies that the energy of the system must be recomputed twice for each vertex, one to compute $E(\mathbf{X})$ and the second one is $E(\mathbf{X}+\varepsilon\,\mathbf{e}_m)$, where a small perturbation is added to the m-th component. According to (5.1), this means to recompute each individual vertex energy each time. Fortunately, this can be handled efficiently by pointing out which terms will be changed after the perturbation.

A quadratic cost would be required for a naive implementation. A more detailed analysis shows that the perturbation only affects the x or y component of a vertex $\mathbf{x}_i$, which implies that only the three related grain areas and the three grain boundary

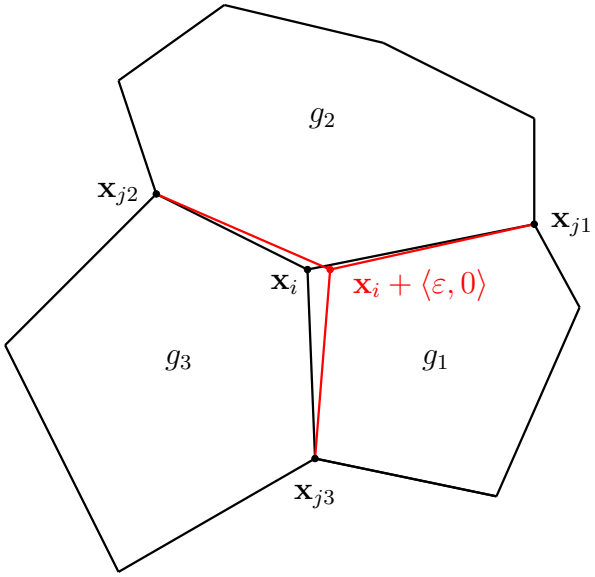


Figure 5.1: Sketch of modified component related to vertex $\mathbf{x}_i$, i.e., neighboring arc lengths and areas get modified.

arc lengths connected to vertex $\mathbf{x}_i$ will change, as shown in Figure 5.1. So, instead of recomputing the energy for each one of the $2M$ components of the gradient, we can compute directly the difference of energies given by the perturbation. Expanding the difference of energies from (5.3) and taking into account that the perturbation effect is local, we obtain:

$$\begin{aligned} E(\mathbf{X} + \varepsilon\, \mathbf{e}_m) - E(\mathbf{X}) &= \sum_{k=1}^{K} \gamma^{(k)} \Delta \mathcal{L}_k + \sum_{g=1}^{N} \mathcal{E}_g \Delta A_g \\ &= \sum_{j=1}^{3} \gamma^{(k_{i,j})} \Delta \mathcal{L}_{k_{i,j}} + \sum_{j=1}^{3} \mathcal{E}_{g_{i,j}} \Delta A_{g_{i,j}}, \end{aligned}$$

where $i = m \bmod M$. Notice that the three boundaries and grain areas are modified by adding a perturbation to the vertex i. Now, dividing by ε we obtain the right-hand-side of (5.3),

$$\frac{E(\mathbf{X} + \varepsilon\, \mathbf{e}_k) - E(\mathbf{X})}{\varepsilon} = \sum_{j=1}^{3} \gamma^{(k_{i,j})} \frac{\Delta \mathcal{L}_{k_{i,j}}}{\varepsilon} + \sum_{j=1}^{3} \mathcal{E}_{g_{i,j}} \frac{\Delta A_{g_{i,j}}}{\varepsilon}$$

Moreover, each estimation of the gradient can be computed efficiently in parallel since only local information of areas and arc lengths is needed.

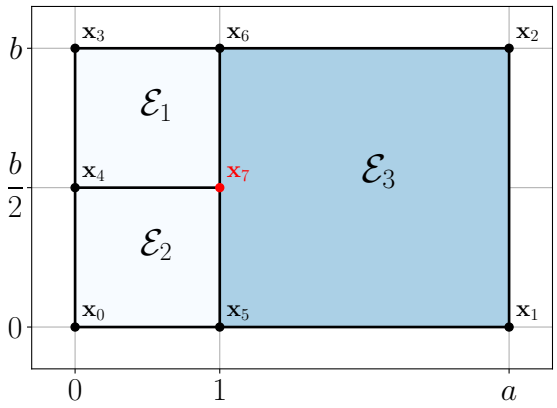


Figure 5.2: Configuration of three grains with different values of stored energy and a vertex with one degree of freedom (red dot). Initially the dot is placed at $(1, \frac{b}{2})$.

5.2 The Stored Energy Effect Over the Dynamics of a Triple Junction

This section studies how the stored energy affects the configuration and stability of the grain network. We extend the stability study in [5] where they only analyze stable configurations.

Considering a bounded domain where three grains coexists, where two of them have the same stored energy, i.e., $\mathcal{E}_1 = \mathcal{E}_2$, and the third has a different stored energy $\mathcal{E}_3$ as shown in Figure 5.2. To measure the effect of the stored energy we let the central vertex located at $(x, b/2)$ to freely move, due to symmetry this only happens along the x-axis. Let A_1, A_2, A_3 the areas of the grains with stored energy $\mathcal{E}_1$, $\mathcal{E}_2$ and $\mathcal{E}_3$, respectively. Grain areas can be explicitly computed as:

$$A_1 = A_2 = \frac{b\,(1+x)}{4}, \qquad A_3 = a\,b - \frac{b\,(1+x)}{2}. \tag{5.4}$$

We now compute each local stored energy based on 5.2. Grain class for A_1 and A_2 is four and for A_3 is five. Notice that in this domain, boundaries on the edges are not

counted twice per grain, thus they are not divided by 2. We also assume $\mathcal{E}_3 > \mathcal{E}_1$.

$$\widehat{E}_0 = \widehat{E}_3 = 1 + \frac{b}{2} + \mathcal{E}_1 \frac{b\,(1+x)}{16}$$
$$\widehat{E}_1 = \widehat{E}_2 = a - 1 + b + \mathcal{E}_3 \frac{ab - \frac{b(1+x)}{2}}{5}$$
$$\widehat{E}_4 = b + \frac{x}{2} + \mathcal{E}_1 \frac{A_1}{2}$$
$$\widehat{E}_5 = \widehat{E}_6 = a + \frac{\sqrt{(x-1)^2 + (b/2)^2}}{2} + \mathcal{E}_1 \frac{b(1+x)}{16} + \mathcal{E}_3 \frac{ab - \frac{b(1+x)}{2}}{5}$$
$$\widehat{E}_6 = \frac{x}{2} + \sqrt{(x-1)^2 + (b/2)^2} + \mathcal{E}_1 \frac{b(1+x)}{8} + \mathcal{E}_3 \frac{ab - \frac{b(1+x)}{2}}{5}$$

If we add up all the individual energy terms and replace the definitions of areas in (5.4), the total energy becomes:

$$E = \sqrt{b^2 + 4\,(-1+x)^2} + x + a\,(4 + b\,\mathcal{E}_3) + \frac{b\,(8 + \mathcal{E}_1\,(x+1) - \mathcal{E}_3\,(1+x))}{2}. \quad (5.5)$$

If we compute the derivative of (5.5) with respect to x we obtain

$$\frac{dE}{dx} = 1 + \frac{4(x-1)}{\sqrt{b^2 + 4(x-1)^2}} - \frac{b\Delta\mathcal{E}}{2} \quad (5.6)$$

where $\Delta\mathcal{E} = \mathcal{E}_3 - \mathcal{E}_1$. Thus, the motion of the central vertex is influenced by the difference of stored energies and the height of the domain b. The domain width a does not affect the steady state. The steady state can be found making (5.6) equal to 0, which yields the following roots:

$$x = \frac{-24 + 2\,b\,\Delta\mathcal{E}(b\,\Delta\mathcal{E} - 4) \pm \sqrt{-b^2\,(b\Delta\mathcal{E} - 6)\,(b\,\Delta\mathcal{E} - 2)^2\,(b\,\Delta\mathcal{E} + 2)}}{2\,(b\,\Delta\mathcal{E} - 6)(b\,\Delta\mathcal{E} + 2)}. \quad (5.7)$$

When $\Delta\mathcal{E} = 0$, the domain height b conditions the position of the steady state which is at:

$$x_{\text{steady}} = x_+ = 1 - \frac{b}{2\sqrt{3}}.$$

By analyzing the obtained roots only x_+ has physical sense. The critical point, assuming $\Delta\mathcal{E} > 0$, is just before the squared root in (5.7) becomes complex. This is at $b\Delta\mathcal{E} = 6$ and solutions with $b\Delta\mathcal{E} < 6$ will be real and will be a valid steady states, as shown in Figure 5.3. From the numerical point of view several experiments were performed by estimating $\frac{dE}{dx}$ with the matrix-free approach. As this approach will not capture complex solutions, from $b\Delta\mathcal{E} \geq 6$ the solutions become unstable and unbounded, as shown in Figure 5.5.

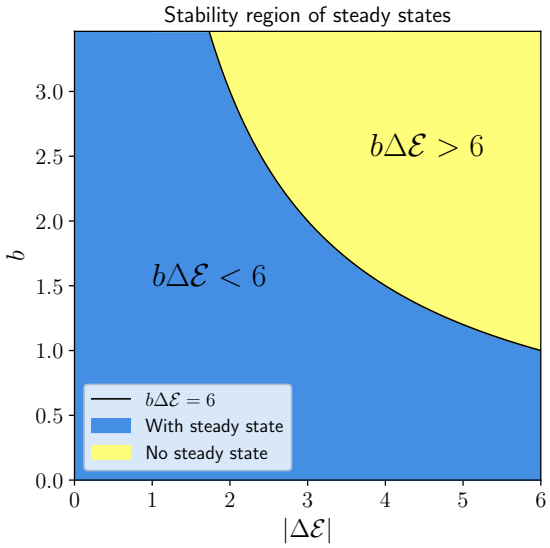


Figure 5.3: Stability regions for the relation of b and $\Delta\mathcal{E}$ in the three grains experiment.

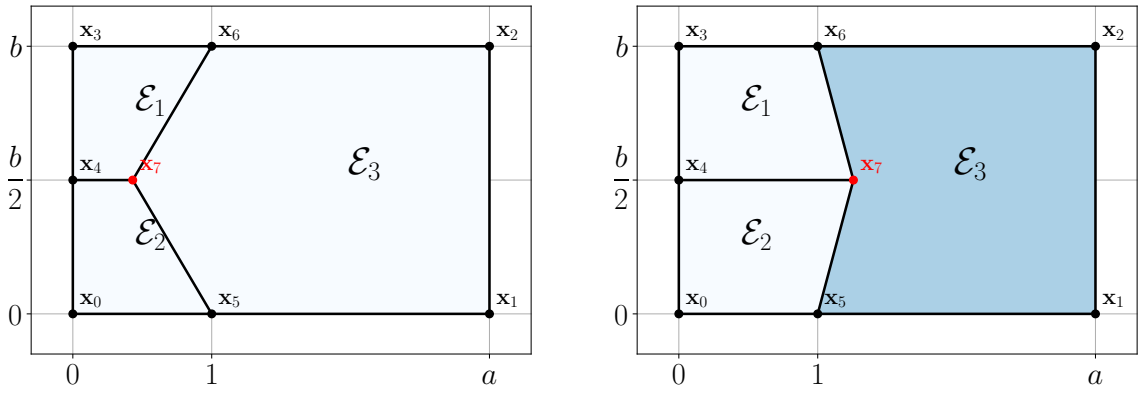
The concept of stability here refers to the interpretation of the vertex location in the domain. If the vertex stays inside the three grains we say that it is an steady state. If the vertex penetrates a grain without stopping, the configuration is said to be unstable.

Figures 5.4a and 5.4b shows two valid steady states. Figure 5.4a shows the steady state when there is no stored energy, therefore the steady state is reached at dihedral angle $2\pi/3$. In presence of a large stored energy, the steady state is situated away from the isotropic steady state as shown in Figure 5.4b. Again, when $b\Delta\mathcal{E} \geq 6$ the solution becomes unbounded.

This estimation can be used as an upper bound for numerical simulations of this model, ensuring that no grains will suffer from instabilities. The interpretation is that the parameter b of this study can be compared to boundaries arc lengths.

5.3 The Nucleation Process: Adding a three side Grain and Letting it Grow

The nucleation process can be classified as a new type of topological change in the system, considering the already well-known topological changes (flipping and grain removal) described in Chapter 2. As usual, this new type of topological change should be energy decreasing, otherwise, it would induce the removal of the nucleated grain right away. To analyze this topological change, we propose the following sketch, see Figure 5.6. In this sketch we observe what we call the current configuration of energy



(a) $\mathcal{E}_1 = \mathcal{E}_3$.

(b) $\mathcal{E}_1 < \mathcal{E}_3$

Figure 5.4: Steady states for different configurations of stored energies. Notice the magnitude of the perturbed steady state in (b).

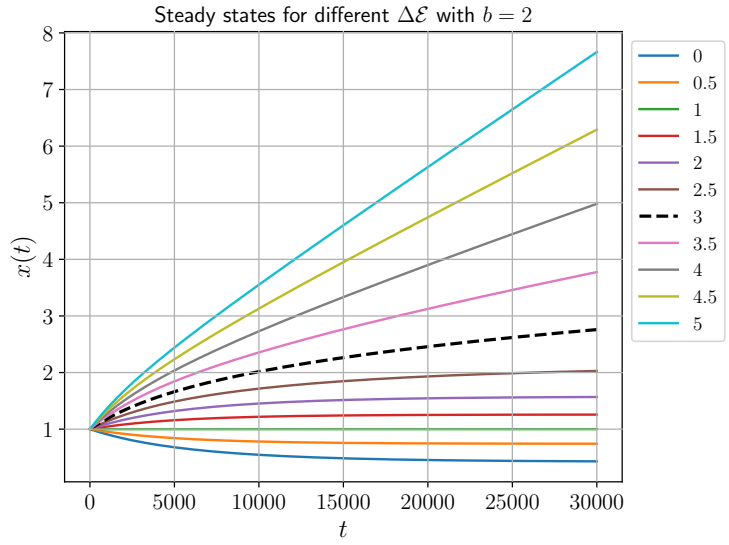


Figure 5.5: Convergence of steady states for a fixed value of b and different values of $b\Delta\mathcal{E}$. Values of $2\,\Delta\mathcal{E} \geq 6$ become unbounded.

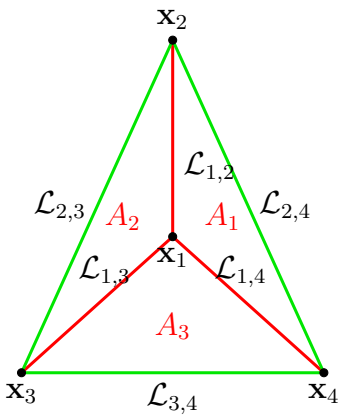


Figure 5.6: Sketch of energies associated to a 3-side grain before and after it is nucleated. Red shows what it is going to be removed and green what it is going to be added.

$E_0^{\triangle}$, in red, and a candidate configuration of energy $E_1^{\triangle}$. The main idea is that for a candidate vertex $\mathbf{x}_1$, where we could perform a nucleation, we can explicitly compute the difference of energy before and after the nucleation, i.e. $\Delta E^{\triangle} = E_1^{\triangle} - E_0^{\triangle}$. So, as long as the difference is negative we can conclude that nucleation will be successful. Thus, the $\Delta E^{\triangle}$ is as follows,

$$\begin{aligned} E_0^{\triangle} &= \gamma^{(1,2)}\,\mathcal{L}_{1,2} + \gamma^{(1,3)}\,\mathcal{L}_{1,3} + \gamma^{(1,4)}\,\mathcal{L}_{1,4} + \mathcal{E}_1\,A_1 + \mathcal{E}_2\,A_2 + \mathcal{E}_3\,A_3, \\ E_1^{\triangle} &= \gamma^{(2,4)}\,\mathcal{L}_{2,4} + \gamma^{(2,3)}\,\mathcal{L}_{2,3} + \gamma^{(3,4)}\,\mathcal{L}_{3,4}, \\ \Delta E^{\triangle} &= E_1^{\triangle} - E_0^{\triangle}, \end{aligned} \tag{5.8}$$

where $\mathbf{x}_i$ are the coordinates of the vertex i for $i \in \{1,2,3,4\}$, $\mathcal{L}_{i,j}$ is the arc length from vertex $\mathbf{x}_i$ to vertex $\mathbf{x}_j$, $\gamma^{(i,j)}$ is the grain boundary energy from vertex $\mathbf{x}_i$ to vertex $\mathbf{x}_j$, A_1 is the area of grain with vertices $\{\mathbf{x}_1, \mathbf{x}_4, \mathbf{x}_2\}$, A_2 is the area of grain with vertices $\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\}$, A_3 is the area of grain with vertices $\{\mathbf{x}_1, \mathbf{x}_3, \mathbf{x}_4\}$, and $\mathcal{E}_g$ is the store energy of the grain associated to A_g for $g \in \{1,2,3\}$.

5.3.1 A Symmetric Nucleation Analysis

To gain insight in the general conditions for allowing nucleation, we will consider a particular case. Specifically, we will consider isotropic grain boundary energy γ, i.e., $\gamma = 1$, $\mathcal{L}_{1,i} = r$ for $i \in \{2,3,4\}$ $\mathcal{L}_{2,4} = L$, $\mathcal{L}_{2,3} = L$, $\mathcal{L}_{3,4} = L$, and dihedral angle of $2\pi/3$. This leads us to the following simplification of equation (5.8),

$$\Delta E^{\triangle} = 3\,L - 3\,r - \underbrace{(\mathcal{E}_1 + \mathcal{E}_2 + \mathcal{E}_3)}_{\bar{\mathcal{E}}}\,A.$$

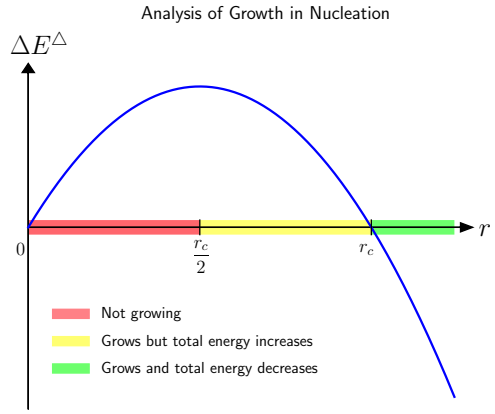
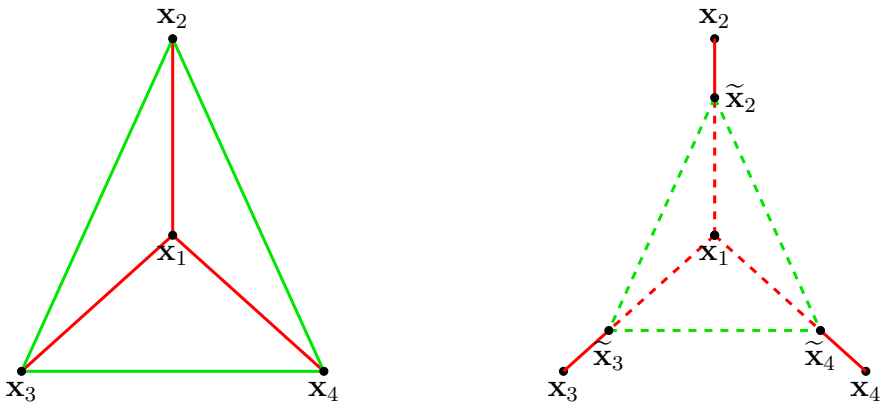


Figure 5.7: Plot of $\Delta E^{\triangle}$ symmetric with $\bar{\mathcal{E}} = \mathcal{E}_1 + \mathcal{E}_2 + \mathcal{E}_3 = 18$. The ability to growth depends of the size of the nucleated grain.

Moreover, due to symmetry, we have that $L = \sqrt{3}\, r$ and $A = \frac{\sqrt{3}}{4}\, r^2$. Thus, we obtain,

$$\begin{aligned} \Delta E^{\triangle} &= 3\sqrt{3}\, r - 3\, r - \bar{\mathcal{E}}\, \frac{\sqrt{3}}{4}\, r^2 \\ &= 3\, r \left(\sqrt{3} - 1 - \frac{\sqrt{3}}{12}\, \bar{\mathcal{E}}\, r \right). \end{aligned}$$

From which we found out that there are two critical points with $\bar{\mathcal{E}} \neq 0$, these are $r_0 = 0$ and $r_c = \dfrac{4\,(3-\sqrt{3})}{\bar{\mathcal{E}}} \approx 5.071796\,(\mathcal{E}_1+\mathcal{E}_2+\mathcal{E}_3)^{-1}$. The first critical point $r_0 = 0$ indicates that making a nucleation of area 0 will add 0 energy, which is correct but we are not interested in that case. On the other hand, the second critical point is the one we are interested, $r_c \approx 5.071796\,\bar{\mathcal{E}}^{-1}$, since from that point on we ensure we are not increasing the energy of the system and will let the nucleated grain to grow. In Figure 5.7 we observe a qualitative behavior of (5.8) for the symmetric case, this plot shows that we require r from Figure 5.6 to be at least $r_c = 4\,(3-\sqrt{3})\,\bar{\mathcal{E}}^{-1}$. This implies to nucleate a large grain, otherwise we will be adding energy to the system. An important consideration needs to be made, if the nucleated grain has radius r in the range $[0, r_c/2]$, the nucleated grain will not grow. On the other hand, if the radius of the nucleated grain is in the range $]r_c/2, r_c]$, the total energy of the system will increase and then will decrease with the growth of the introduced grain. When the nucleated grain satisfies $r > r_c$, the nucleated grain will grow and also will remove energy from the system. Figure 5.7 show these ranges in red, yellow and green, respectively.



(a) $\Delta E_1^{\triangle}$ (b) $\Delta E_0^{\triangle}$

Figure 5.8: Sketch of energies for computing $\Delta^2 E^{\triangle}$, see Figure 5.6

5.3.2 Analysis of the Nucleated Grain

For a successful energy decreasing nucleation, as discussed in the previous section, we have a constraint on the size of the grain to be nucleated. Otherwise the nucleated grain will not grow. Therefore, in the yellow zone there is a potential for a grain to grow but at a cost of a temporary increase of energy. The main interest in this type of configuration is that it imposes a lower requirement for the size of the grain to make the nucleation successful.

A preliminary analysis indicates that the *slope* of the $\Delta E^{\triangle}$ is negative in that range and an energy decreasing model will favor the enlargement of the grain since it will decrease the energy. More rigorously, we need to compute $\Delta(\Delta E^{\triangle}) =: \Delta^2 E^{\triangle}$. Figure 5.8 shows and sketch of two possible nucleation at vertex $\mathbf{x}_1$. In this case the nucleation on the left is the one explained previously in Figure 5.6, which will be denoted as $\Delta E_1^{\triangle}$. And the one on the right is a nucleation for a smaller grain for the same vertex $\mathbf{x}_1$, which will be denoted as $\Delta E_0^{\triangle}$. These two *virtual* nucleations allow us to obtain $\Delta^2 E^{\triangle} = \Delta E_1^{\triangle} - \Delta E_0^{\triangle}$. Thus, if $\Delta^2 E^{\triangle} < 0$, we obtain that the nucleated grain may grow since it will decrease the energy of the system. This value will be denoted as the *nucleation threshold*.

Notice that under anisotropic grain boundary energy the computation of $\Delta^2 E^{\triangle}$ will include each grain boundary energy terms, which have to be computed with respect to the orientation of the nucleated grain. By default we set the orientation of the nucleated grain as $\alpha = 0$.

5.3.3 Alternative Nucleation with an Specific Orientation

Consider the $\Delta E^{\triangle}$ function from (5.8). This definition considers the grain boundary energy function which depends on grains misorientation $\Delta\alpha$ between the nucleated

grain and surrounding grains. We can choose the orientation of the nucleated grain α_{nucl} to minimize the energy $E_1^{\triangle}$ of the nucleated grain instead of setting it to zero, that is to find:

$$\min_{\alpha \in [0,2\pi)} \quad E_1^{\triangle}. \tag{5.9}$$

It is preferable a fast algorithm to approximate (5.9) rather than an expensive solver that finds the exact minimum since each vertex will execute the same procedure, even if the search is performed in parallel. A simple procedure is to obtain an approximation to the minimum. We define the orientation of the nucleated grain as α_{nucl} to be an approximation to (5.9). The algorithm consists in generate a sample of values in the range $[0, 2\pi]$ and evaluate $E_1^{\triangle}$ and find the value that yields the minimum energy.

5.4 Stored Energy Vertex Model with Nucleation Algorithm

The algorithm of this model considers first the computation of boundaries properties such as arc lengths and energies and the computation of grain areas. We choose to perform continuous nucleation, that is, nucleating at regular intervals of the simulation. In our case we choose to nucleate at each time step a single grain as seen in [24]. Therefore we choose a suitable site for nucleation as studied in previous sections considering the site with the lowest nucleation factor $\Delta^2 E^{\triangle}$. If there is a site for nucleation the grain is added and neighboring topology is updated along with their related modified quantities, that is, recompute arc lengths, energies and areas. The next stage considers the computation of the vertices velocities using the matrix-free approach described in Section 5.1. Then, extinction times for each boundary are computed and we label boundaries that will flip, that is when $t_{\text{ext}} \in [0, \Delta t]$. Here we must decide which flippings are safe to perform and therefore we avoid neighbor transitions. Algorithm 6.3 from Chapter 6 solves this issue. This model may generate configurations of two grains of three sides, we detect them and remove them if any of the boundaries related will flip. We also remove in this stage any three sided grain that will have a boundary marked as flip. After this fixes we apply the remaining flippings, we update the vertices positions and the system evolves to the time $t + \Delta t$. Algorithm 5.1 summarizes the described algorithm.

5.5 Numerical Experiments

Numerical experiments were performed using 100,000 initial grains. Grain boundary energy anisotropy is added by using (2.3) with parameters $\varepsilon = \{0.02, 0.2\}$. The initial distribution of stored energy is chosen to propitiate more grains with high

Algorithm 5.1 Stored Energy Vertex Model

```
1: procedure SEVM(n_iters, boundaries Γ, vertices 𝒳, grains 𝒢)
2:     (t, n_iters) ← (0, 0)
3:     for iter = 0, ..., n_iters do
4:         Γ ← ComputeArclengthsAndEnergies(Γ)
5:         𝒢 ← ComputeAreas(𝒢)
6:         𝒳 ← ComputeNucleationFactor(𝒳)
7:         i ← GetVertexWithMinNucleationFactor(𝒳)
8:         if i ≥ 0 then
9:             (Γ, 𝒳, 𝒢) ← Nucleate(i, Γ, 𝒳, 𝒢)
10:            Γ ← ComputeArclengthsAndEnergies(Γ)
11:            𝒢 ← ComputeAreas(𝒢)
12:            𝒳 ← ComputeNucleationFactor(𝒳)
13:        else
14:            No nucleation site has been found
15:        end if
16:        V ← ComputeMatrixFreeVelocities(𝒳)
17:        Γ ← PollingSystem(Γ)                          ▷ See Alg. 6.3
18:        (Γ, 𝒳, 𝒢) ← RemoveThreeSidedGrains(Γ, 𝒳, 𝒢)
19:        Γ ← ApplyFlips(Γ)
20:        𝒳 ← UpdateVerticesPositions(𝒳, V)
21:        iter ← iter + 1
22:        t ← t + Δt
23:    end for
24: end procedure
```

values of stored energies and thus more places for nucleation. We use a triangular distribution for this purpose with the following parameters:

$$\mathcal{E} \sim \text{Triangular}(3, 6, 6) \tag{5.10}$$

where $\mathcal{E} = 6$ is chosen due to previous study in Section 5.2 (see Figure 5.5 and numerical experiments related). Figure 5.15a shows the triangular distribution. The computational domain is $\Omega = [0, 15]^2$ with periodic boundary conditions.

Notice, since the bound for maximum difference of stored energy is an absolute value (it is 6), we needed to increase the computational domain so we were able to obtain $\Delta^2 E^{\triangle} < 0$, otherwise this is very unlikely. If this constraint is not satisfied, we have have triple junctions with dihedral angles greater that π and those triple junctions could cross grain boundaries and create a corrupt grain network. This may be handled with different types of topological transition, see [22].

Stages of the numerical simulation with the two explained nucleation variations can be observed in Figures 5.10 and 5.16. Both experiments were run using $\varepsilon = 0.02$

5.5.1 Energy Minimization

Figure 5.9a shows energy curves over time for various initial conditions of grain networks following the formula (5.1) with $\varepsilon = 0.2$ to make more noticeable the effect of the grain boundary energy. The stored energy algorithms used were the following: No nucleation, which is a Vertex model with stored energy, nucleation choosing sites with orientation $\alpha = 0$ and nucleation choosing sites with orientation $\alpha = \alpha_{\text{nucl}}$.

The Stored Energy models shows some interesting behaviors. At the first time steps the three models behave very similar, minimizing the overall energy. Near $t = 0.6$ the nucleating models actually induces some small energy increments with respect to the no nucleation model but they are very small with respect to the magnitude of the total energy but still appreciable since both associated families of curves are above the family of curves corresponding to the no nucleation model. This behavior can be seen in the highlighted zone in Figure 5.9a. A zoom of this behavior is shown in Figure 5.9b where both nucleating models are more energetic until $t \approx 1$.

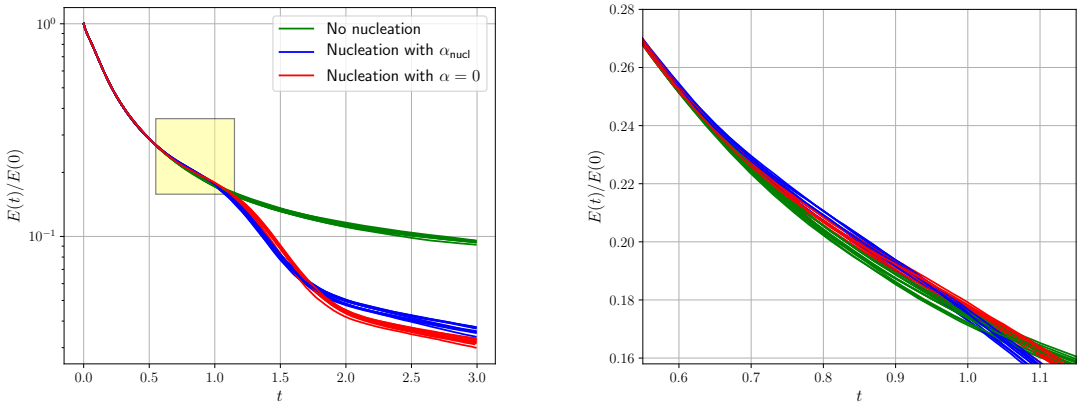
At some point near $t = 1$ nucleating models starts to minimize the energy faster than the no nucleating model. The nucleation model using $\alpha = 0$ does not minimize the energy as fast the nucleation model using α_{nucl}, which is caused by greedy choice of the best possible α for nucleation. This greedy choice does not guarantee obtaining the minimum configuration of energy at later stages of the numerical simulation. In fact near $t = 1.6$ the model with nucleation using $\alpha = 0$ reaches faster a minimum energy than using α_{nucl}, which is explained by the fact that after the recrystallization process - where there are not possible sites for nucleating - and the new stage of grain growth one model has an isotropic configuration since all orientations are equal and thus the misorientation is zero yielding $\gamma^{(k)} = 1$, and the other model reaches an anisotropic configuration which naturally has more energy since grains are misaligned.

5.5.2 Statistics from Model with Basic Nucleation

Figure 5.10 shows the stages observed in the numerical simulation. Figure 5.10a is a zoom of the initial condition with 100,000 grains generated via Voronoi tessellation. Figure 5.10b shows first nucleations in the grain network. Those grains are initialized with $\mathcal{E} = 0$ and $\alpha = 0$. After some time the nucleated grains will start to increase its number of sides as seen in Figure 5.10c. Finally, Figure 5.10d shows a fully recrystallized grain network. Under the mentioned parameters this network will evolve ruled by isotropic vertex model naturally since $\Delta\alpha = 0$ for every boundary (and thus $\gamma^{(k)} = 1$) and the energy of the system becomes:

$$E = \sum_{k=1}^{K} \mathcal{L}_k. \tag{5.11}$$

Statistics extracted for each of the stages are shown in Figures 5.11, 5.12, 5.13 ,5.14 and 5.15. We begin describing relative area distribution from Figure 5.11 in



(a) System energy (normalized) over time for different models of stored energy.

(b) Zoom of the first nucleation stage where energy is added to the system over non-nucleating model simulations. Yellow square in Figure on the left side.

Figure 5.9: System energy over time for different models of stored energy.

both linear and logarithmic scale. From now on we choose three representative experiments per stage based on the first bin of the linear scale distribution. First experiment is related to the largest first bin (in blue), second experiment is related to the median of the first bin (in red) and third experiment is related to the shortest first bin (in green).

Figure 5.11a shows the initial condition distribution. Figure 5.11b shows the distribution during the first nucleations, which is resembles the Vertex model tail in the log scale distribution related to a great quantity of small grains. Figure 5.11c and 5.11d shows a slightly reduce in the number of small grains but in overall the shape of the distribution remains the same.

Figure 5.12a shows the distribution of dihedral angle of the initial condition. Figure 5.12b shows the distribution during the stage of first nucleations. The distribution is regular as expected since before this stage only grain growth has occurred and there is also a shift to bigger angles from the steady state dihedral angle of $2\pi/3$. Figure 5.12c shows the distribution when the nucleation stage has generated a lot of new grains. This creates a bimodal distribution of small angles and therefore the angles greater than π has also increased. This can be related to the existence of large grains as seen in Figure 5.10c. Finally, Figure 5.12d shows the dihedral distribution angle after full recrystallization which is very similar as the obtained in the early grain growth stage prior to nucleations.

Grain boundary character distribution (GBCD) is shown in Figure 5.13. Dark blue continuous line in background shows the grain boundary energy function from

(2.3) which is

$$\gamma(\Delta\alpha) = 1 + \frac{\varepsilon}{2}\left(1 - \cos^3(4\Delta\alpha)\right).$$

We present the GBCD in two plots. One is the measured misorientation in $[-\pi/4, \pi/4)$ at the left of Figure 5.13 and at the right a symmetric version of the GBCD which accumulates the misorientations by symmetry in $[0, \pi/4)$.

The GBCD of initial condition is shown in Figure 5.13a. Notice that the misorientations are distributed uniformly over $[-\pi/4, \pi/4)$. Figure 5.13b shows a perturbation in the initial distribution towards $\Delta\alpha = 0$ and the tendency to form minimum at $\pm\pi/4$. Figure 5.13c shows the GBCD at nucleation stage. Since the nucleated grains have orientation zero, the middle bin grows accumulating all the data, the entire grain network is evolving towards an isotropic state. In Figure 5.13d the final distribution is shown after full recrystallization. Notice that all orientations have become zero and thus an isotropic regime is reached, the GBCD becomes a single bin at misorientation zero.

Distribution of average number of sides is shown in Figure 5.14. Figure 5.14b shows the initial condition distribution, where the mode is at six, which means that the most frequent grains are six sided. After grain growth and during first nucleations the mode changes to five and we observe many small grains, mostly three sided grains, as seen in Figure 5.14c. During nucleation regime the mode in fact is three because nucleated grains are from this class. Also the frequency of four sided increases. This may be explained since they were formerly nucleated three sided grains that may have increased their number of sides if a nucleation happened in one of their vertex. We observe that huge grains exists but they are not determined to stay forever. Finally in Figure 5.14d after fully recrystrallization the distribution stabilizes and three sided grains are again of low frequency, the mode also returns to be five.

Stored energy distribution is shown in Figure 5.15. Figure 5.15a shows the initial condition distribution. In dark blue the triangular distribution from (5.10) is shown. After the first grain growth regime and first nucleations the distribution shows a clear tendency to remove grains with high values of stored energy and maintain those with lower values. Nucleated grains appear in the plot with stored energy equal to zero, as shown in Figure 5.15b. During the nucleation the grains with less stored energy still survives but will be removed no matter what happens, as shown in Figure 5.15c. Figure 5.15d shows the final stage of recrystallization where there are no grains with store energy.

5.5.3 Statistics for Model with Alternative Nucleation

Similar to Section 5.5.2, we present the evolution stages of the grain network in Figure 5.16. Figure 5.16a shows the same initial condition as Figure 5.10a. Figure 5.16b shows first nucleations in the grain network. Those grains are initialized with $\mathcal{E} = 0$ and $\alpha = \alpha_{\text{nucl}}$ which was computed from (5.9). As discussed in Section 5.3.3, we generate α_{nucl} from a sampling over $[0, 2\pi)$ using 500 discretization points.

After some time the nucleated grains will start to increase its number of sides as seen in Figure 5.16c. Finally, Figure 5.16d shows a fully recrystallized grain network. We do not expect in the final recrystallization an isotropic grain network since in general $\Delta\alpha \neq 0$, that is, we do not enforce isotropy. However, we do use stored energy zero for nucleated grains, therefore this stage represents an anisotropic vertex model, which has the following energy equation:

$$E = \sum_{k=1}^{K} \gamma^{(k)} \mathcal{L}_k.$$

Area distribution is shown in Figure 5.17 in both linear and logarithmic scale. Figure 5.17a shows the initial condition distribution. Figure 5.17b shows the distribution during the first nucleations, which again resembles the Vertex model area distribution. Figures 5.17c and 5.17d shows that the distributions over time are similar.

Figure 5.18 shows the dihedral angle distribution. The distributions shown at different stages of the growth are similar to those from Figure 5.12, showing the same bimodal behavior related to the thin and large grains being removed during the nucleation stage in Figure 5.18c with the particularity that the bimodal seems to be more pronounced in earlier steps as seen in Figure 5.18b. The distribution again stabilizes when the structure is fully recrystallizated in Figure 5.18d.

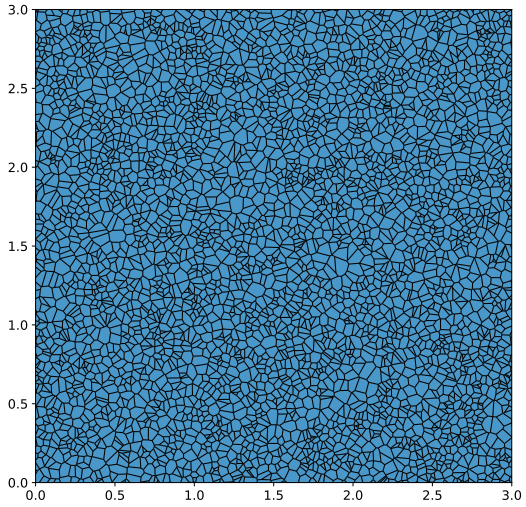
Grain boundary character distribution (GBCD) is shown in Figure 5.19. Dark blue line shows the grain boundary energy function from (2.3).

Again, we present the GBCD in two plots. One is the measured misorientation in $[-\pi/4, \pi/4)$ at the left of Figure 5.13 and at the right a symmetric version of the GBCD which accumulates the misorientations by symmetry in $[0, \pi/4)$.

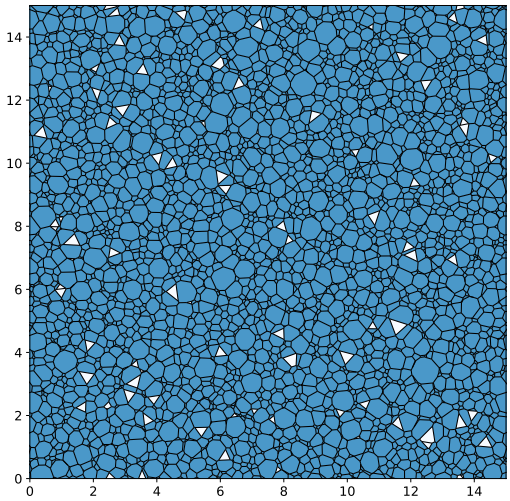
The initial uniform GBCD is shown in Figure 5.19a. Figure 5.19b shows the GBCD just after grain growth and first nucleations. There are more steep slopes, related to the inflection angles 0 and $\pm\pi/8$ and also due to the misorientation choice for nucleated grains which is no longer zero but an approximation to a local energy minimizer which helps to preserve the distribution. Figure 5.19c shows the GBCD at the nucleation stage, which clearly has minimum at misorientation angles $\pm\pi/4$ and maximum at 0. Notice the local maximum reached near $\pm\pi/8$ and where the energy function has its inflection points. Figure 5.19d shows the GBCD after full recrystallization. Since the recrystallized structure holds different values of orientations, this regime is anisotropic and thus the GBCD is not uniform and preserves some behavior of early stages.

Distribution of average number of sides is shown in Figure 5.20. The behavior is very similar comparing to the nucleation model with $\alpha = 0$ in Figure 5.14 where the four plots reflects the growth stage and first nucleations in Figure 5.20b, nucleation stage in Figure 5.20c and recrystallization stage in Figure 5.20d where the mode of the average number of sides is five. In general the distributions are more regular than the other nucleation model and preserve the same characteristics.

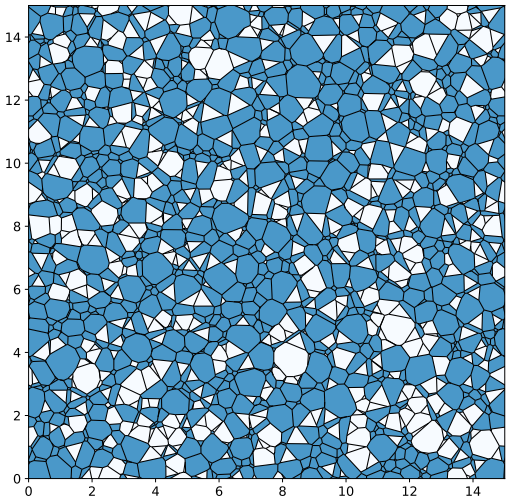
Stored energy distribution is shown in Figure 5.21. Figure 5.21a shows the initial condition distribution. In dark blue the triangular distribution from (5.10) is shown. Similar to the nucleation model with $\alpha = 0$, we observe in Figure 5.21b that after initial growth and first nucleations, the distribution tends to preserve grains with lower stored energy values as well as showing a bin at zero stored energy related to nucleated grains. Figure 5.21c shows that during the nucleation stage the tendency of the energy minimization via removing grains with high stored energy values is maintained and finally in Figure 5.21d the only residual bin of the distribution is at zero which marks the existence of the fully recrystallization structure where the stored energy does not influences the system anymore.



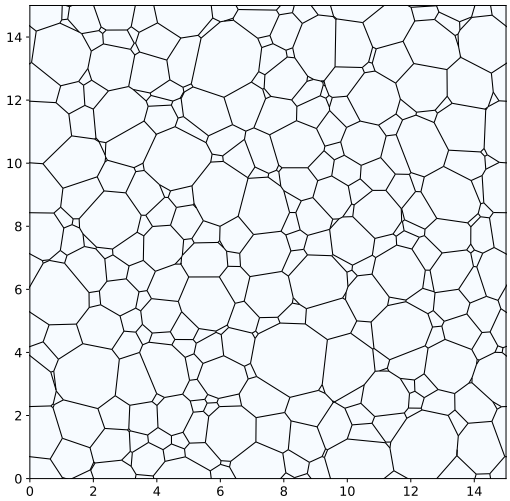
(a) Zoom of Initial distribution generated via Voronoi tessellation.



(b) First nucleations.

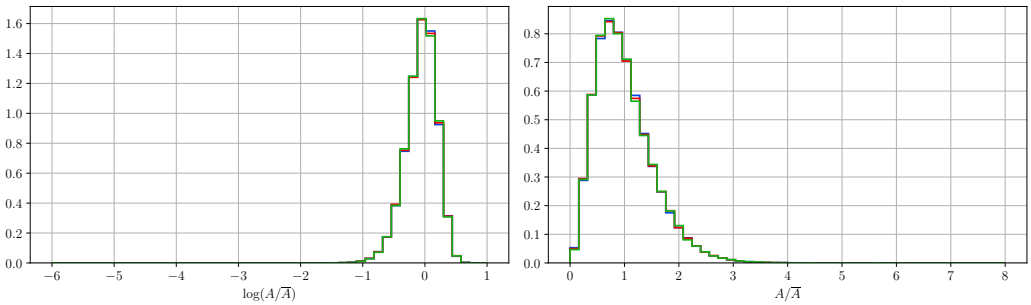


(c) Nucleation stage.

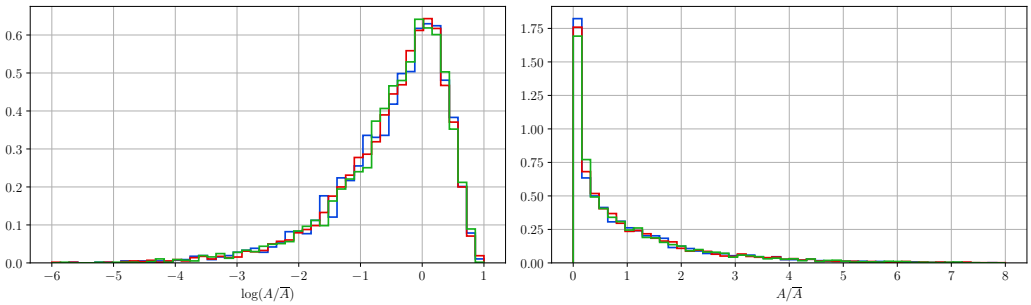


(d) Recrystallization.

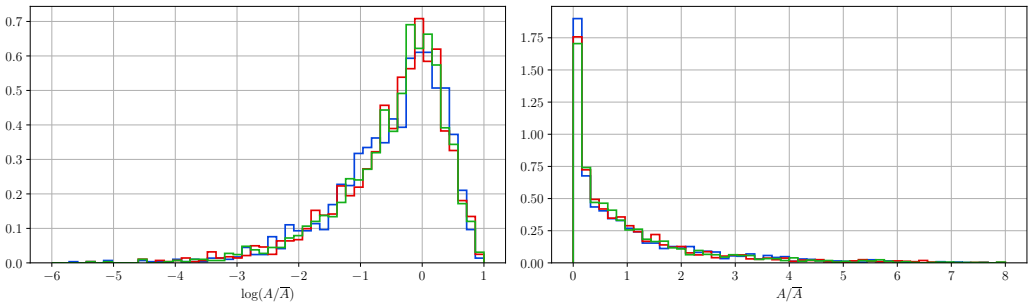
Figure 5.10: Stages of grain network evolution nucleating with $\alpha = 0$.



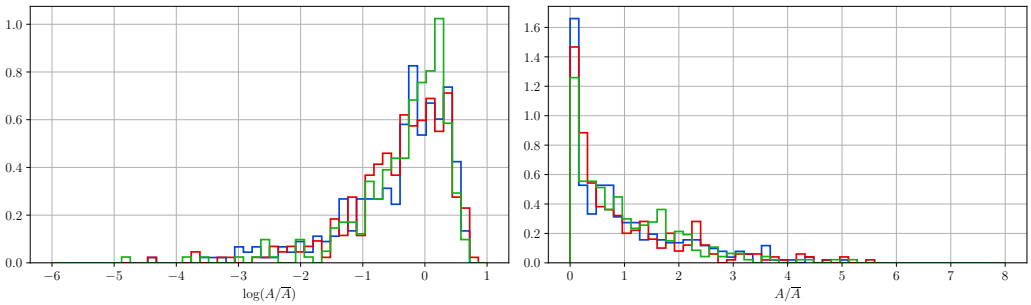
(a) Initial condition.



(b) First nucleations.

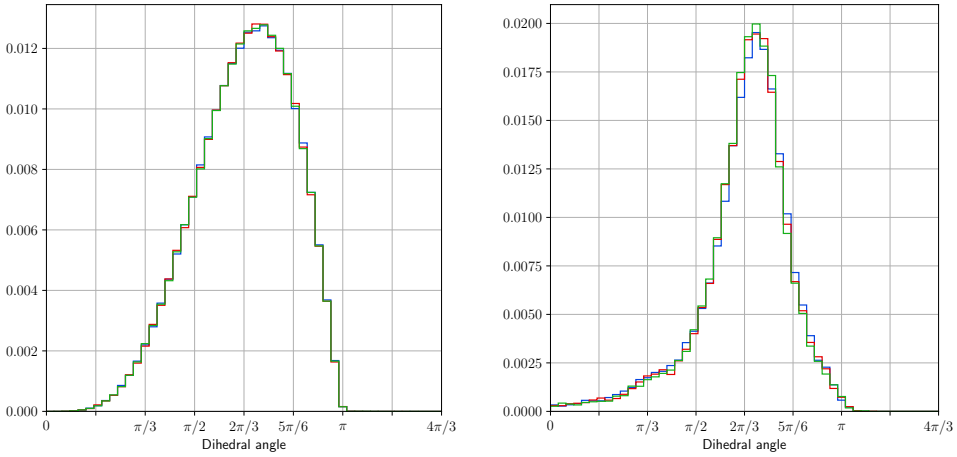


(c) Nucleation stage.



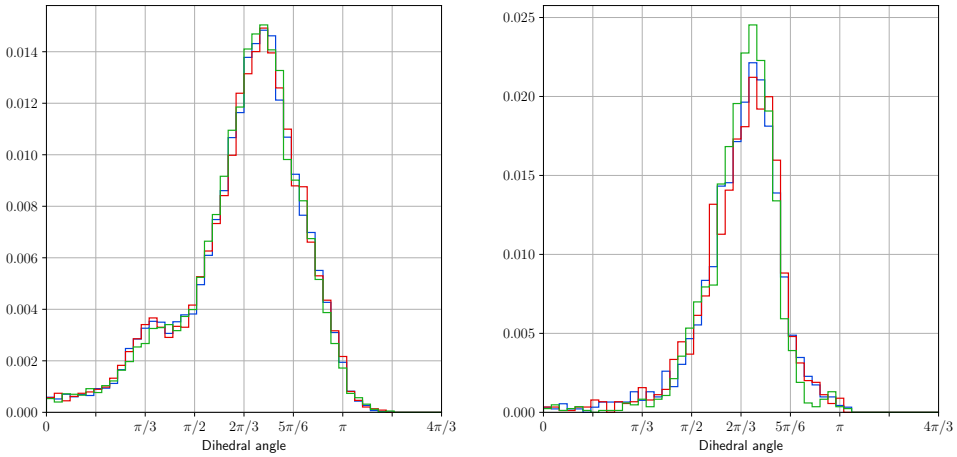
(d) Recrystallization.

Figure 5.11: Grain relative area distributions at different stages of grain network evolution and nucleation with $\alpha = 0$.



(a) Initial condition.

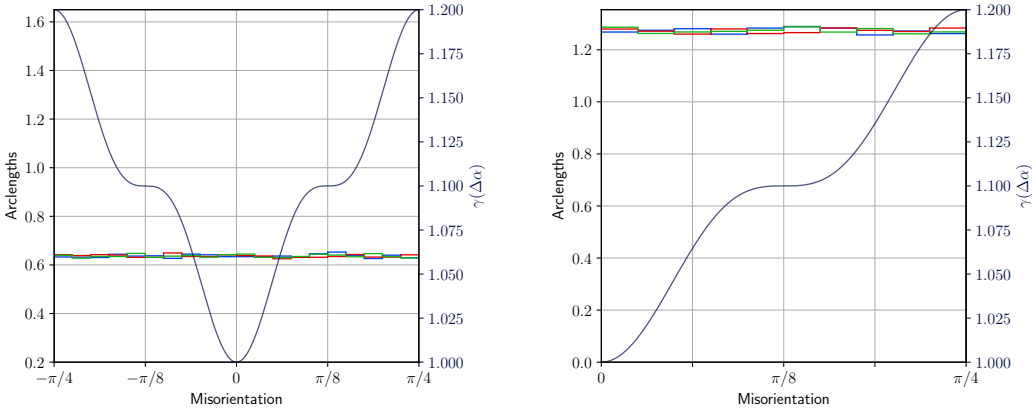
(b) First nucleations.



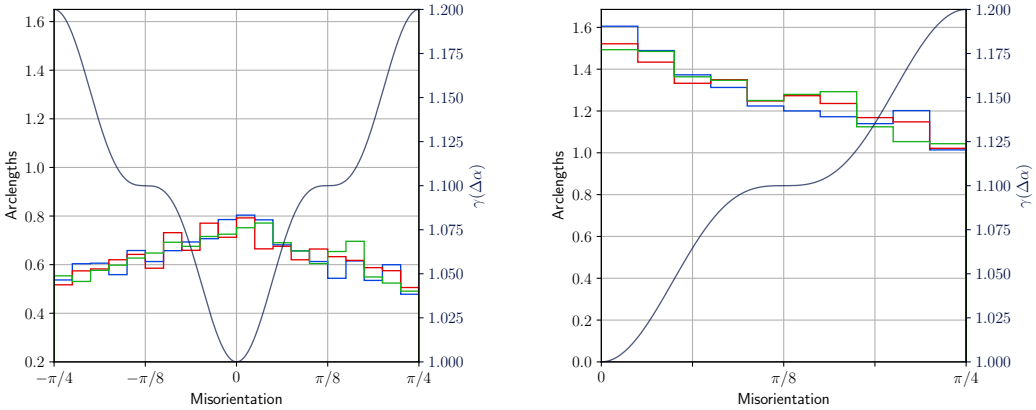
(c) Nucleation stage.

(d) Recrystallization.

Figure 5.12: Dihedral angle distributions at different stages of grain network evolution and nucleation with $\alpha = 0$.

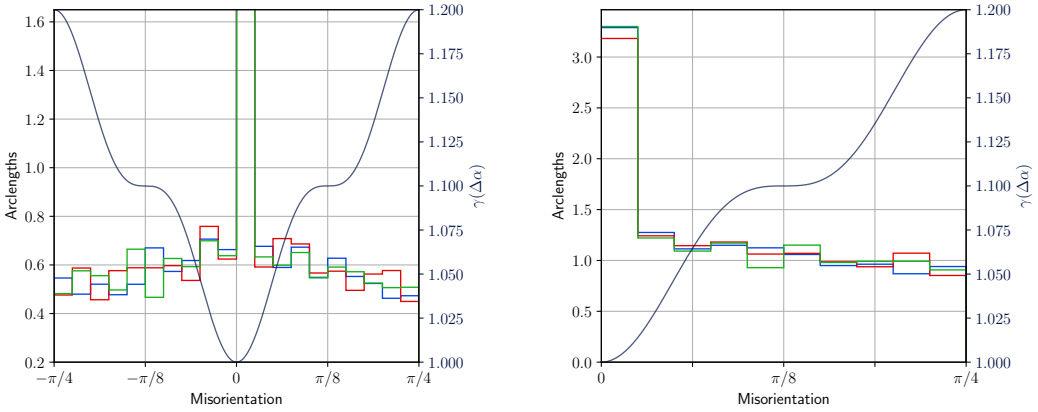


(a) Initial condition.

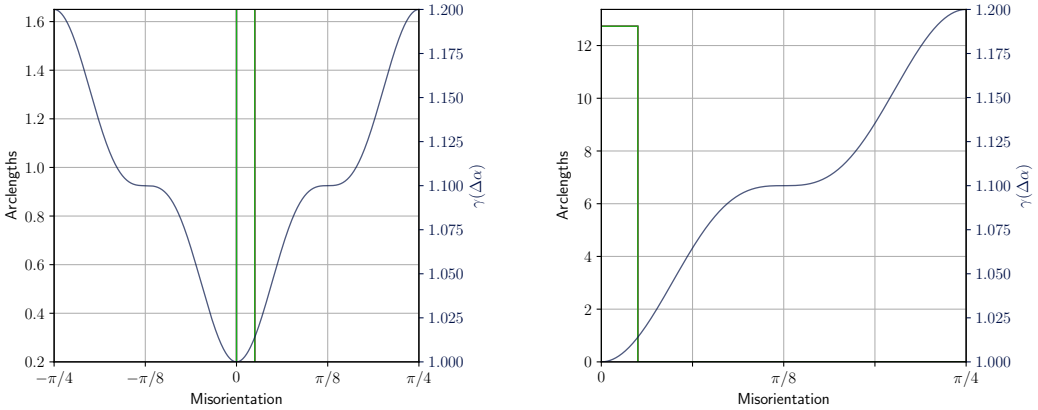


(b) First nucleations.

Figure 5.13: Grain Boundary Character distributions at different stages of grain network evolution and nucleation with $\alpha = 0$.

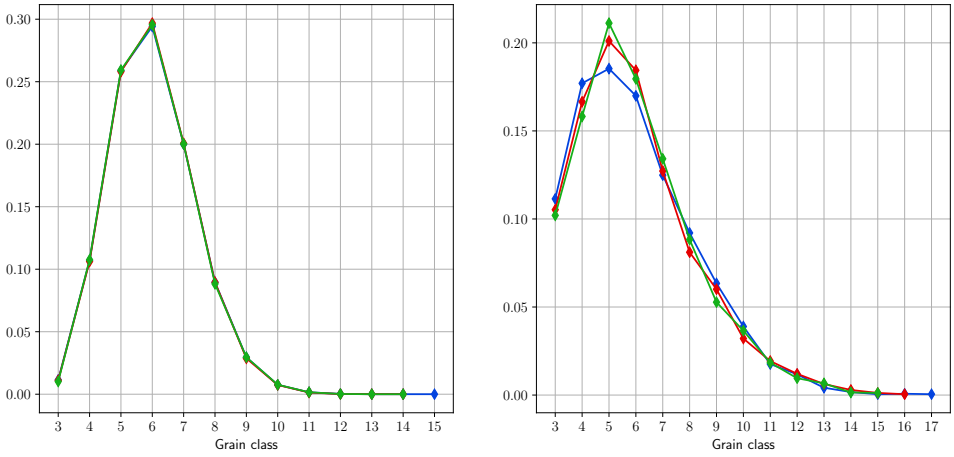


(c) Nucleation stage.



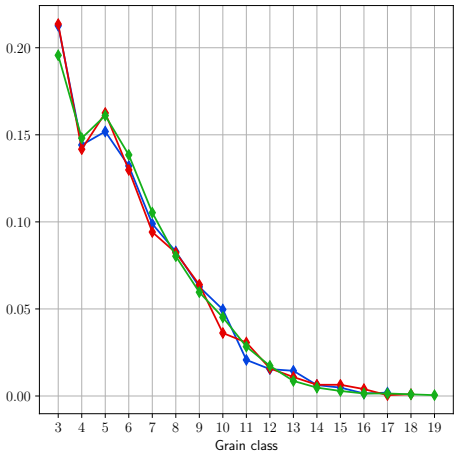
(d) Recrystallization.

Figure 5.13: Grain Boundary Character distributions at different stages of grain network evolution and nucleation with $\alpha = 0$. (Cont.)

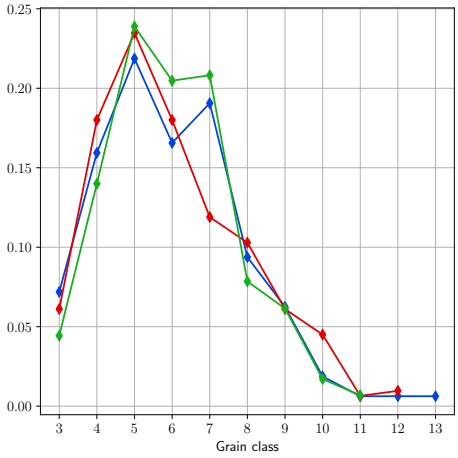


(a) Initial condition.

(b) First nucleations.

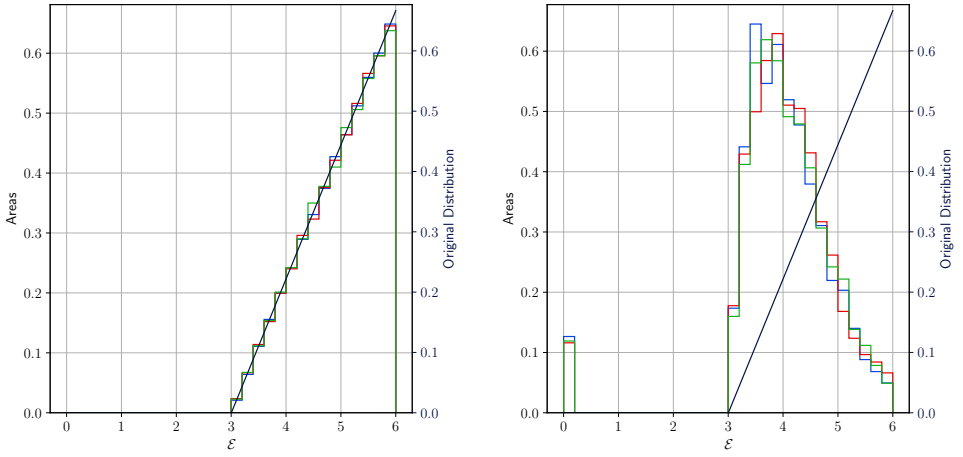


(c) Nucleation stage.



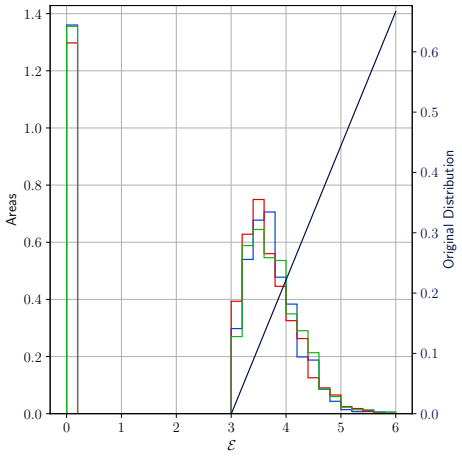
(d) Recrystallization.

Figure 5.14: Average number of sides distributions at different stages of grain network evolution and nucleation with $\alpha = 0$.

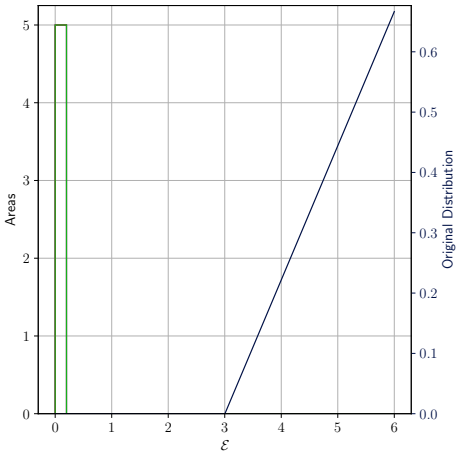


(a) Initial condition.

(b) First nucleations.

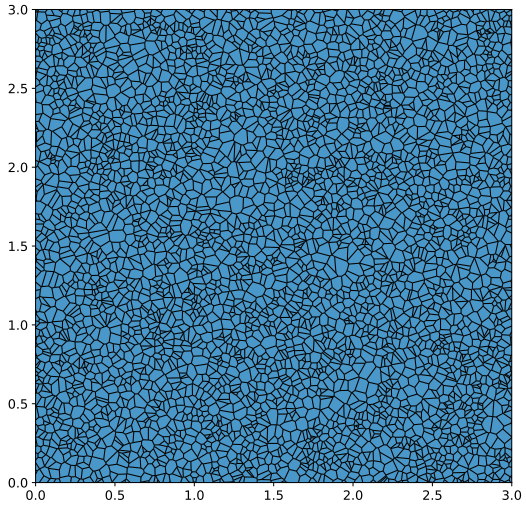


(c) Nucleation stage.

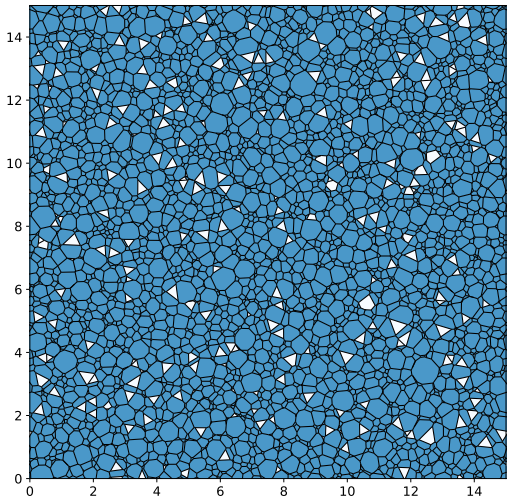


(d) Recrystallization.

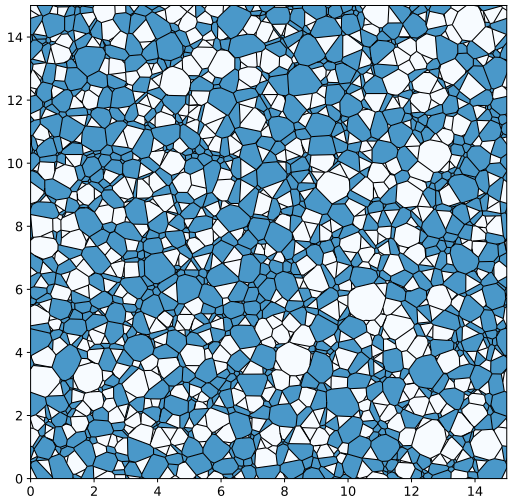
Figure 5.15: Stored energy distributions at different stages of grain network evolution and nucleation with $\alpha = 0$.



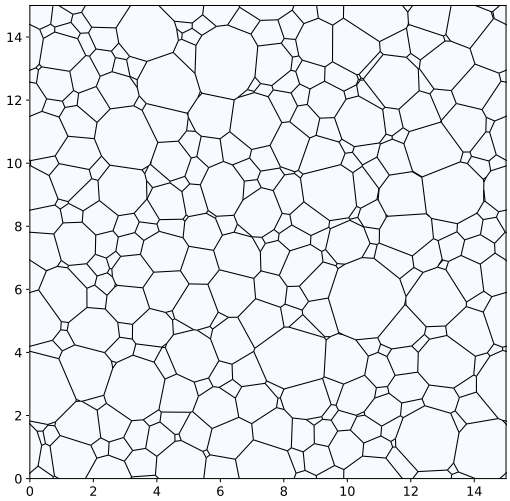
(a) Zoom of Initial distribution generated via Voronoi tessellation.



(b) First nucleations.

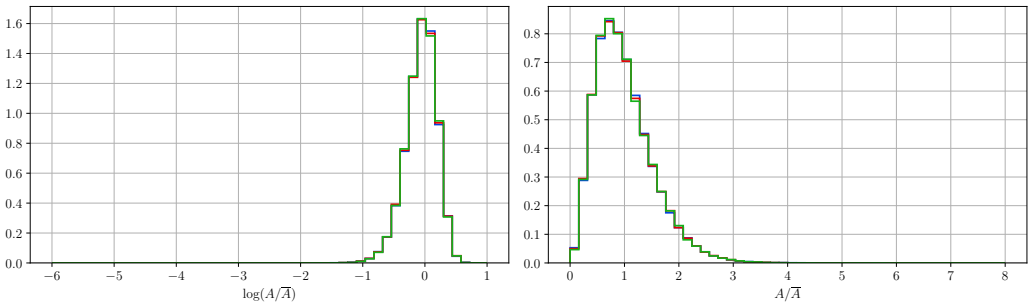


(c) Nucleation stage.

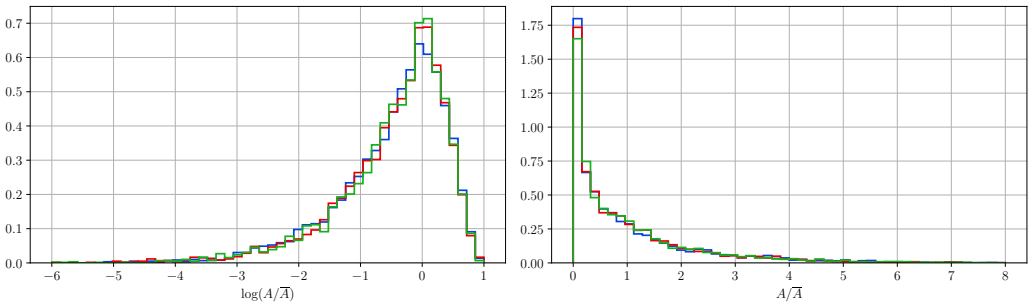


(d) Recrystallization.

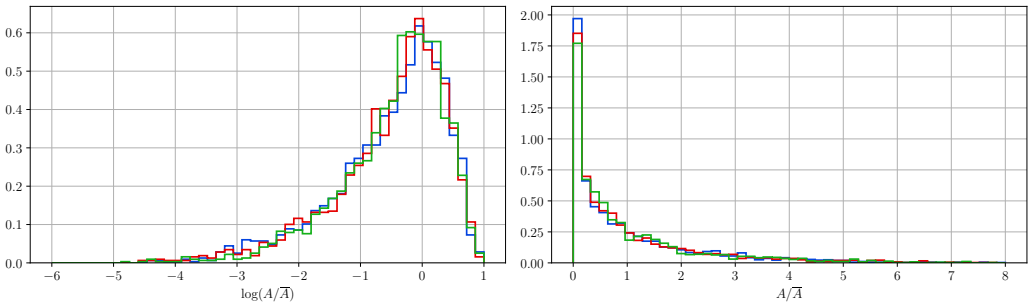
Figure 5.16: Stages of grain network evolution nucleating with $\alpha = \alpha_{\text{nucl}}$.



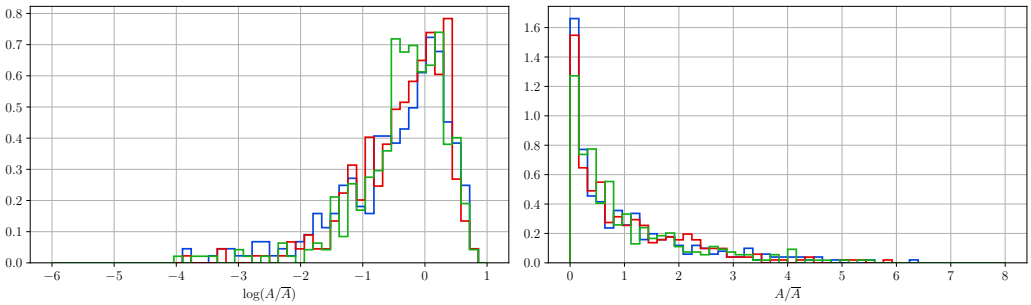
(a) Initial condition.



(b) First nucleations.

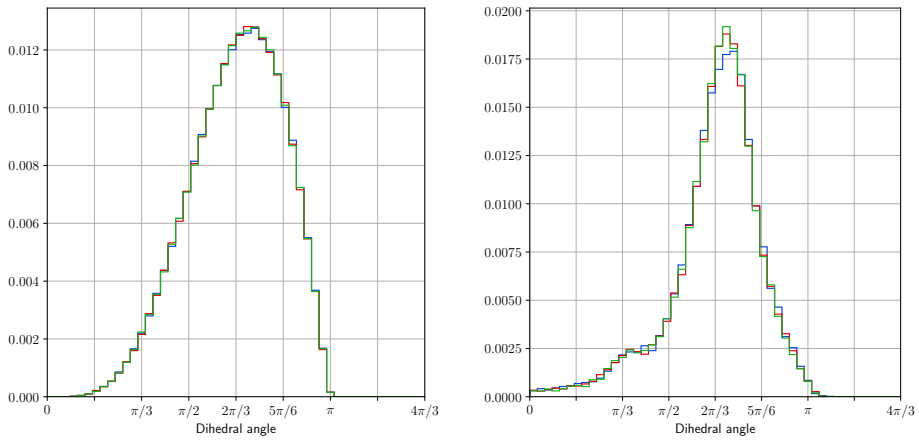


(c) Nucleation stage.



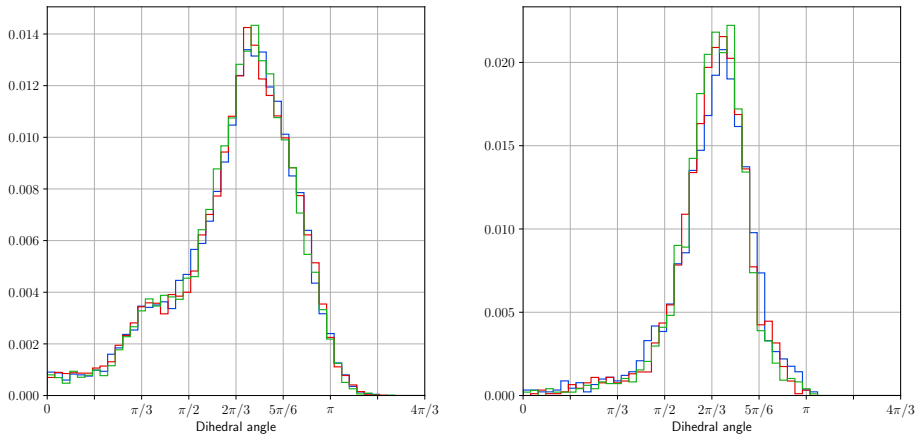
(d) Recrystallization.

Figure 5.17: Grain relative area distributions at different stages of grain network evolution and nucleation with $\alpha = \alpha_{\text{nucl}}$.



(a) Initial condition.

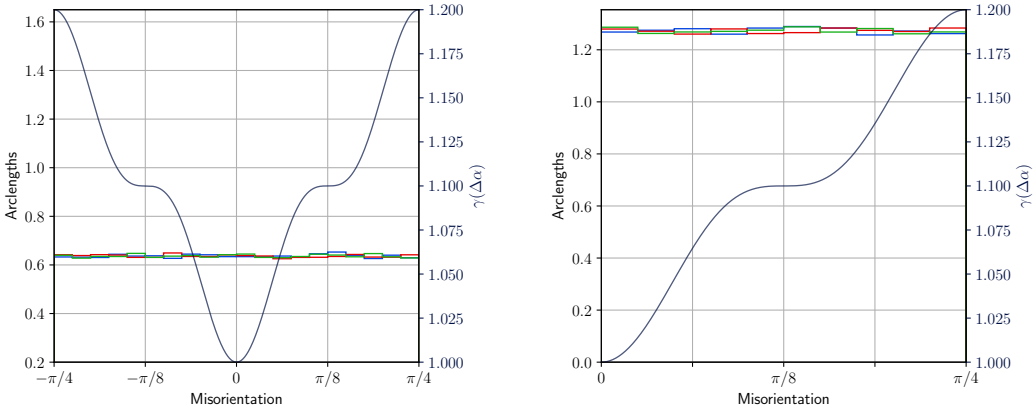
(b) First nucleations.



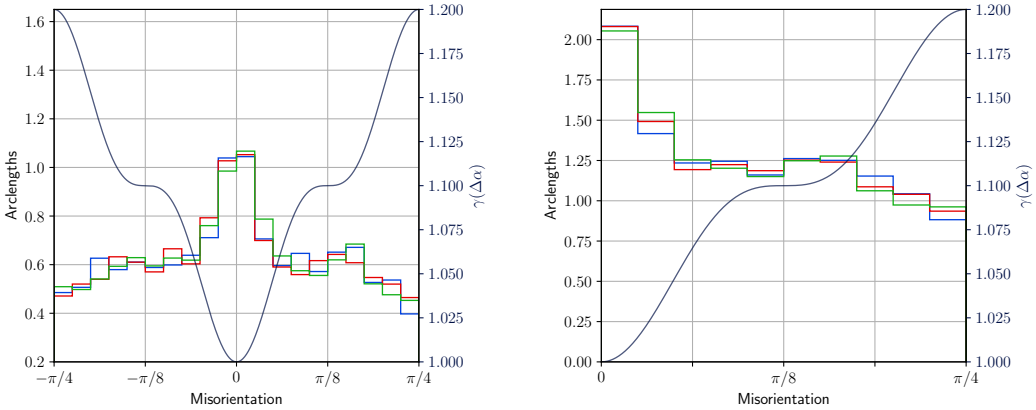
(c) Nucleation stage.

(d) Recrystallization.

Figure 5.18: Dihedral angle distributions at different stages of grain network evolution and nucleation with $\alpha = \alpha_{\text{nucl}}$.

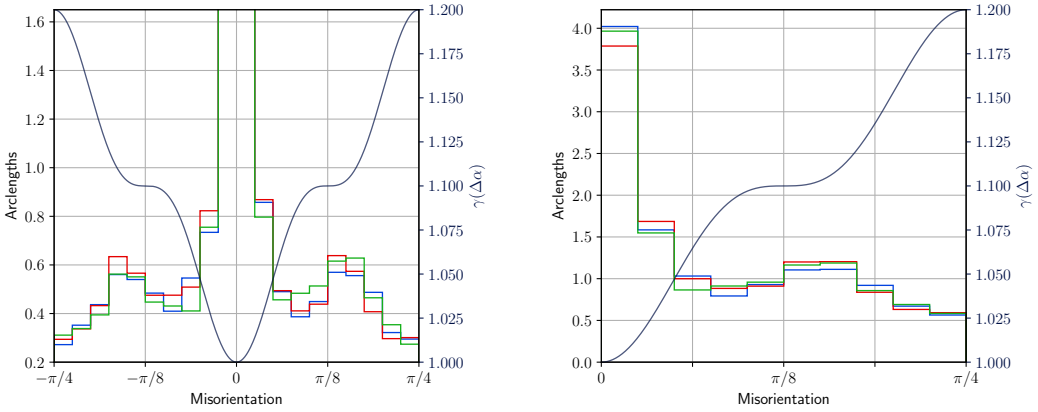


(a) Initial condition.

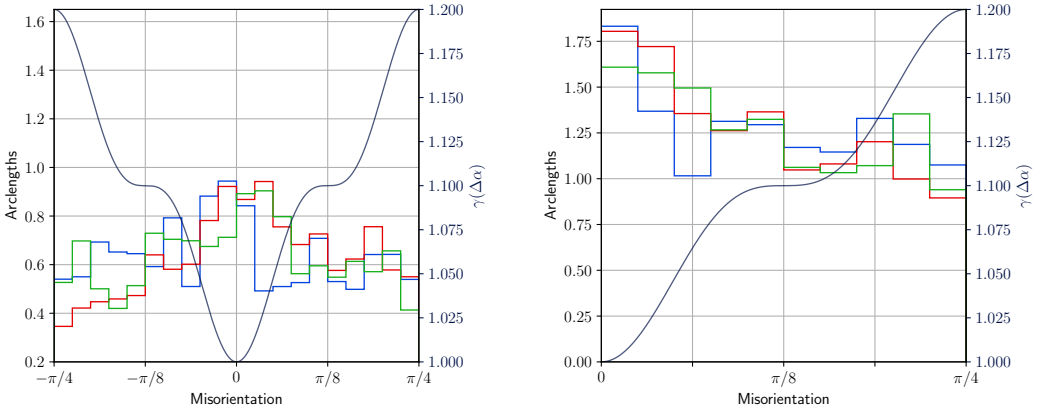


(b) First nucleations.

Figure 5.19: Grain Boundary Character distributions at different stages of grain network evolution and nucleation with $\alpha = \alpha_{\text{nucl}}$.

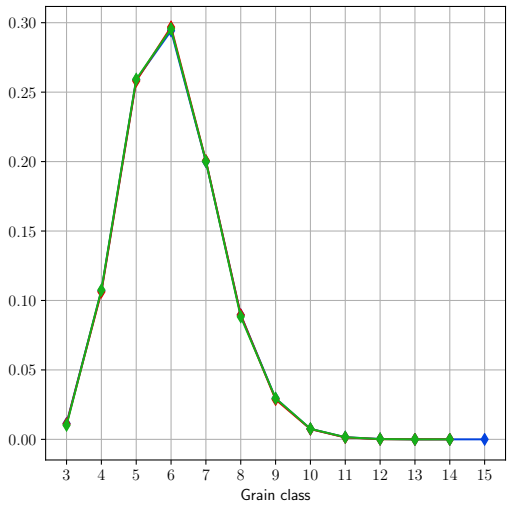


(c) Nucleation stage.

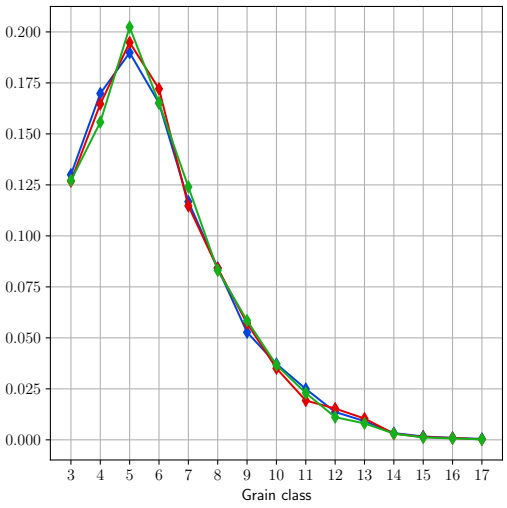


(d) Recrystallization.

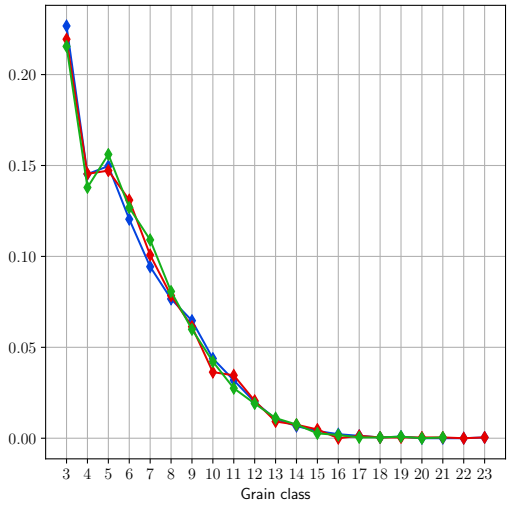
Figure 5.19: Grain Boundary Character distributions at different stages of grain network evolution and nucleation with $\alpha = \alpha_{\text{nucl}}$. (Cont.)



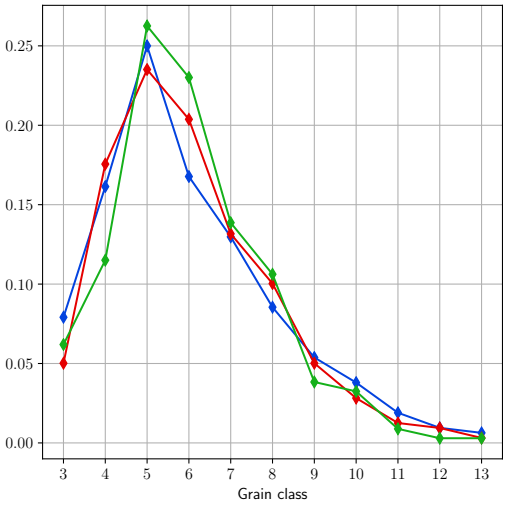
(a) Initial condition.



(b) First nucleations.

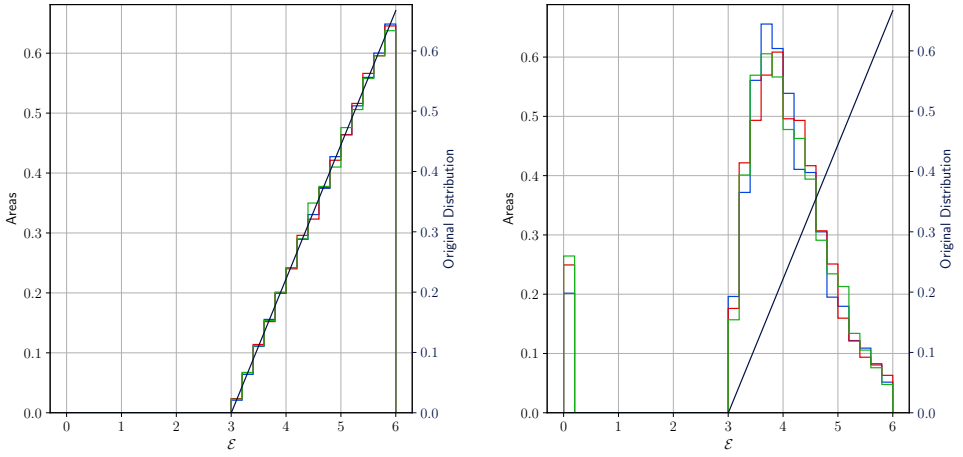


(c) Nucleation stage.



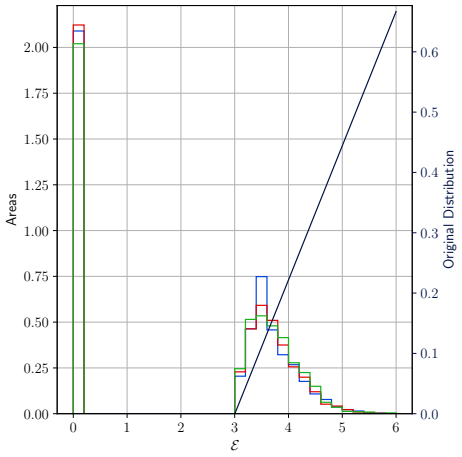
(d) Recrystallization.

Figure 5.20: Average number of sides distributions at different stages of grain network evolution and nucleation with $\alpha = \alpha_{\text{nucl}}$.

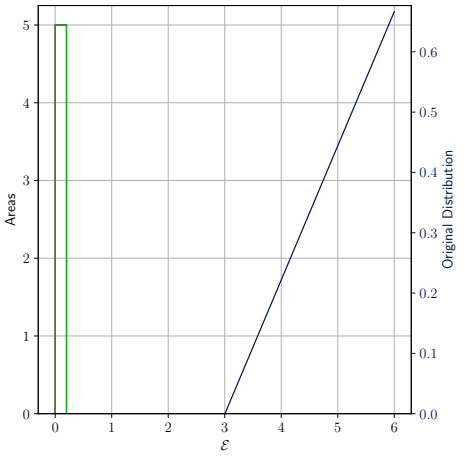


(a) Initial condition.

(b) First nucleations.



(c) Nucleation stage.



(d) Recrystallization.

Figure 5.21: Stored energy distributions at different stages of grain network evolution and nucleation with $\alpha = \alpha_{\text{nucl}}$.

Chapter 6

Parallel Management of Topological Transitions

As discussed in Chapter 2, topological transitions modify the grain structure and data structure must reflect this relations at every time step. The update of this structure is critical, due to the consistency during numerical simulation and implementation performance. We discuss in this chapter the sequential management algorithm of topological transitions in a grain growth numerical simulation and then we propose a parallel management algorithm that aims to handle a great number of grains.

6.1 Sequential Management

The Coupled Model and Stored Energy Model developed in Chapter 4 and 5, respectively, rely on the grain structure of vertices, boundaries and grains explained in Chapter 2.

In a sequential implementation of these models the transitions can be detected by estimating the time that a boundary will collapse, called extinction time (t_{ext}). If the extinction time lies between the current time t and one time step after $t + \Delta t$, i.e. $[t, t + \Delta t]$, the boundary is considered to flip and we say this boundary is a candidate for flipping. A simple algorithm to solve the transitions within the time step is to sort the candidate boundaries by their extinction time in decreasing order. We can iteratively solve the conflicts by labeling those candidates involved in transitions of other candidates, the labeled boundaries are inhibited to flip this time step. Once all the conflicts has been solved the transitions are performed. Algorithm 6.1 summarizes the described management.

Due to the need of more significant statistics, hundreds of thousands of grains must be simulated. The sequential implementation of these models in CPU treats grain structure evolution and topological transitions sequentially. It is possible to parallelize the structure evolution in CPU and still solve the topological transitions

Algorithm 6.1 Sequential Management of Topological Transitions

```
1: procedure SEQUENTIAL
2:     Γ ← Clear flip and inhibited state for each boundary
3:     Γ ← Update t_ext for each boundary
4:     Γ_flip ← Boundaries to flip with t_ext ∈ [0, Δt]
5:     Γ_flip ← Sort boundaries by t_ext in decreasing order
6:     Γ_tmp ← ∅,    Empty list of boundaries visited
7:     X_tmp ← ∅,    Empty list of vertices visited
8:     for each Γ ∈ Γ_flip do
9:         {x_i, x_j} ← Vertices of Γ
10:        if X_tmp ∩ {x_i, x_j} = ∅ then
11:            Γ_tmp ← Γ_tmp ∪ Γ
12:            X_tmp ← X_tmp ∪ {x_i, x_j}
13:        else
14:            return Γ_tmp
15:        end if
16:    end for
17: end procedure
```

in the traditional way from Algorithm 6.1 but this process might become inefficient as we deal with large simulations.

We choose the implement these models in GPU and therefore we expect to handle topological transitions taking advantage of such architecture, that is, to avoid whenever is possible the sequential treatment of the structure.

The parallel implementation in GPU is straightforward when we speak of evolving the data structure: in the CUDA programming language [17, 18] we can launch one thread per vertex or boundary to compute the velocities and then update the positions. The problem of solve the topological transitions arise since the mechanism to solve inconsistencies is sequential and parallelism over data structure will generate race conditions. For example, naive implementation of the sequential algorithm in GPU with CUDA would cause that each thread working with a boundary will try to modify the lists of boundaries visited. Thus the algorithm must be restated in a parallel way.

6.2 Parallel Polling System

Since the concurrent access to data structure in a parallel implementation makes the sequential approach of managing topological transitions a bottleneck, a management system is proposed based on local information given by vertices and boundaries. First we need to compute the extinction time for each boundary and label the candidates for flipping. Instead of sorting the boundaries as the sequential algorithm, each vertex

will store local information on which boundary has the lowest extinction time. We say that each vertex *voted* for a boundary. Next, each boundary counts the obtained votes by referring to its vertices. The vote counting is meaningful, two votes means that a boundary can flip since all their neighbor boundaries have not collided yet, and earns the right to label neighbor boundaries to not flip in this time step. Other boundaries will obtain one or zero votes. This polling is repeated until no further labeling is possible, that is, when the previous set of candidates is the same as the next iteration.

Figure 6.1 shows a brief example of the polling system. Consider a neighborhood of boundaries with their extinction times already computed. Figure 6.1a shows the vertex poll. Each vertex in this neighborhood will vote for their boundary with lowest t_{ext}. We can see that some boundaries obtained zero, one or two votes. The case of a boundary with two votes is special because implies that two vertices (pink and green) decided that their shared boundary is a good candidate and therefore no other boundary in the immediate neighborhood is a flipping candidate where they obtained at most one vote. Figure 6.1b shows the final counting for each boundary. Boundaries with zero votes remain in black, boundaries with one votes are cyan and boundaries with two votes are red. Notice that although cyan boundaries might be able to flip, the priority is assigned to boundaries with two votes and they label their neighbor boundaries as inhibited as shown in Figure 6.1c. Inhibited boundaries, in gray, can't be voted at next iteration. Related vertices must vote between the remaining non-inhibited boundaries, as shown in Figure 6.1d.

This procedure can be seen as a fixed point iteration over a polling function $\text{Poll}(\mathcal{S})$, where the input of the system $\mathcal{S}$ is the inhibited state of each boundary. When the polling function, given an inhibited state, returns the same inhibited state, we have found a fixed point of the function and thus we converged to a feasible solution. Algorithm 6.3 summarizes this idea.

The advantage of this system is that the vertices votes and boundaries counting can be performed in parallel since data is accessed as read-only. No race condition exists here since we avoided possible near flippings. One disadvantage is the computational cost of perform the polling iteratively until convergence, but the number of topological transitions over time tend to decrease, thus this algorithm is in practice fast enough.

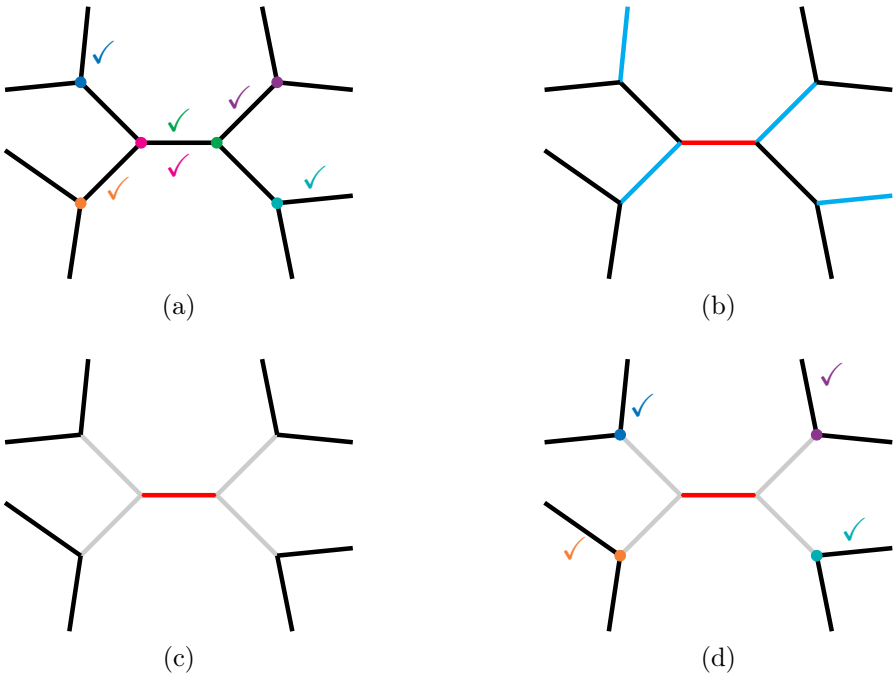


Figure 6.1: Polling system scheme.

Algorithm 6.2 Polling Routine for Inhibit Boundaries

procedure POLL($\mathcal{S}$)
    $\mathbf{\Gamma}_{\text{flip}} \leftarrow$ Clear boundaries count to 0 and vertices votes
    $\mathbf{\Gamma}_{\text{flip}} \leftarrow$ Vertices votes for their uninhibited boundary with lowest t_{ext}
    $\mathbf{\Gamma}_{\text{flip}} \leftarrow$ Boundaries counts the votes received
    $\mathbf{\Gamma}_{\text{flip}} \leftarrow$ Select uninhibited boundaries with two votes
    $\mathbf{\Gamma}_{\text{flip}} \leftarrow$ Inhibit neighbor boundaries
    return $\mathcal{S}$ ▷ New inhibited state of boundaries
end procedure

Algorithm 6.3 Parallel Polling System for Managing Topological Transitions

procedure POLLINGSYSTEM(boundaries $\mathbf{\Gamma}$)
    $\mathbf{\Gamma} \leftarrow$ Clear flip and inhibited state for each boundary
    $\mathbf{\Gamma} \leftarrow$ Update t_{ext} for each boundary
    $\mathbf{\Gamma}_{\text{flip}} \leftarrow$ Boundaries to flip with $t_{\text{ext}} \in [0, \Delta t]$
    $\mathcal{S}_0 \leftarrow$ Initial inhibited state of boundaries
    for $i : 1, \ldots, n$ **do**
        $\mathcal{S}_i \leftarrow$ POLL($\mathcal{S}_{i-1}$)
        if $\mathcal{S}_i = \mathcal{S}_{i-1}$ **then**
            return
        end if
    end for
end procedure

Chapter 7

Implicit-Transition Model for Three-dimensional Grain Growth

In a three-dimensional setting, grain boundaries are surfaces of polyhedra instead of curves delimiting a plain grain. Triple junctions are now triple lines where three grains meet. The single point where four grains meet are called quadruple junctions. Formally, consider a cube unit domain $[0, 1]^3 \subset \mathbb{R}^3$ with periodic boundary conditions. Notation is similar to the two dimensional setting. Grains are still defined as a set $\boldsymbol{\mathcal{G}}$ of N disjoint regions as (2.1), that is, a set of polyhedra. Let $\boldsymbol{\Gamma}$ the set of grain boundaries delimiting the grains as in (2.2), but now these are polygonal surfaces. Instead of triple junctions, we have the set $\boldsymbol{\mathcal{X}}$ of M quadruple junctions.

A classic approach for generating initial conditions for grain growth simulation consists in the Voronoi tessellation of a domain from a random set of initial points [1, 2, 12, 13, 30, 32]. The proposed model extends the idea of vertex-driven models to three dimensions by maintaining grain surfaces flat and triple lines straight during grain structure evolution. This implies that vertices can't be moved individually as vertex model works, otherwise grain surfaces will no longer remain flat. In order to impose these restrictions it is proposed to move the generator points used for the generation of the Voronoi tessellation instead. This implies that every time we move the generator points, we need to build a new tessellation. As the tessellation always gives flat surfaces, the imposed restriction of flat faces and straight triple lines are always accomplished [26].

The motivation for this model is that two dimensional topological transitions are already difficult to implement and check. Chapter 6 shows how cumbersome can be this process even if we introduce sophisticated mechanisms. In a three dimensional setting the possibilities for topological transitions increases considerably since grain boundaries are now surfaces instead of curves [2]. This model handles implicitly topological transitions, that is, to avoid modifications, consistency checking and repairs to underlying data structure, only relying in how the structure is generated via tessellations.

The tessellation is build using a certain group of points in the domain. Let $\mathcal{P}$ the set of generator points of a tessellation such that:

$$\mathcal{P} = \mathcal{P}(t) = \{\mathbf{P}^{(1)}, \mathbf{P}^{(2)}, \ldots, \mathbf{P}^{(N)}\}.$$

Grain structure evolves when generator points moves and each new tessellation after generator motion is considered a new state of the grain structure. This continuous motion and tessellations are stable in front of small changes of the generator points [23]. Grain removal is assumed to happen when a grain decreases its volume until a certain minimum value is reached. The grain is removed by simply removing the related generator point.

The total energy of the system is inspired in the total energy of a two dimensional grain structure from (2.8) where instead of integrating along curve grain boundaries, we integrate over grain surfaces.

$$E(t) = \sum_{k=1}^{K} \int_{\Gamma^{(k)}} \sigma_k \, dA, \tag{7.1}$$

where $\sigma^{(k)}$ is the grain boundary energy per unit of area, analogously to γ in the two dimensional setting. In order to simulate the grain structure evolution we must ensure that (7.1) decreases. The following velocity equation is proposed for the generator points:

$$\dot{\mathbf{P}}^{(\mathcal{G})} = \sum_{m=1}^{M} \gamma^{(m,l)} \mathbf{T}^{(m,l)}, \tag{7.2}$$

where m and l indicates quadruple junctions sharing a triple line, $\gamma^{(m,l)}$ is an energy term related to the triple line and $\mathbf{T}^{(m,l)}$ is the unit tangent vector to the triple line. Consider $\boldsymbol{\mathcal{X}}_{\mathcal{G}}$, $\mathcal{G} = 1, \ldots, N$ the set of all the quadruple junctions that belongs to a grain $\mathcal{G}$. Each quadruple junction in $\boldsymbol{\mathcal{X}}_{\mathcal{G}}$ has four triple lines and thus four neighbor junctions, but only three of them lies in $\mathcal{G}$, the other triple junction belongs to an adjacent grain $\mathcal{G}'$, this is the considered triple line in (7.2) between m and l and generates an unit tension $\mathbf{T}^{(m,l)}$ that will affect the motion of the generator point $\mathcal{P}^{(\mathcal{G})}$. Algorithm 7.1 summarizes the proposed method.

7.1 Numerical Experiments

In order to test the proposed model, two numerical experiments were performed. The first is related to how the evolution equation does indeed minimize the total energy of the grain structure. The second experiment shows that grains obtained in tessellations over time evolve and gain or lose faces through topological transitions that are not explicitly handled. Isotropic setting was chosen thus $\gamma^{(m,l)} = 1$ for all triple lines.

Algorithm 7.1 Implicit-transition Model for Grain Growth

procedure IMPLICIT_GRAIN_GROWTH($N_0, \Delta t$)
  $t \leftarrow 0$
  $N(0) \leftarrow N_0$
  $\mathcal{P}(t) \leftarrow N_0$ random points in unitary first octant.
  while 80% of the grains has not been removed **do**
    $\boldsymbol{\mathcal{G}}(t) \leftarrow$ Voronoi($\mathcal{P}(t)$). ▷ Grain structure
    $\boldsymbol{\mathcal{G}}(t) \leftarrow \boldsymbol{\mathcal{G}}(t) \setminus \{\mathcal{G} \in \boldsymbol{\mathcal{G}}(t) \mid \text{Volume}(\mathcal{G}) < V_{\text{ext}}\}$
    $\mathbf{F}(t) \leftarrow$ Velocity for each generator point.
    $\mathcal{P}(t + \Delta t) = \mathcal{P}(t) + \dot{\mathcal{P}}(t) \cdot \Delta t$
    $t = t + \Delta t$
  end while
  return $\boldsymbol{\mathcal{G}}(t)$
end procedure

7.1.1 Energy Minimization

Recall that the proposed energy equation in (7.2) is not derived from the total energy in (7.1). Total energy over time is shown in Figure 7.1a. This model still achieves a decreasing energy behavior. Although the outcome is not monotonically decreasing, it is asymptotically decreasing.

We found out that if we move the generator points using a linear combination of the the unit outer normal vectors of the faces $\mathbf{N}$, that is

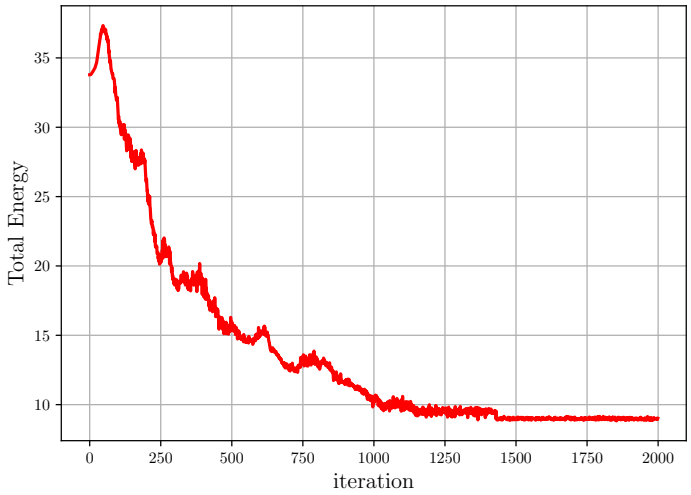
$$\dot{\mathbf{P}}^{(\mathcal{G})} = \sum_{\Gamma^{(k)} \in \mathcal{G}} \sigma_k \mathbf{N}_k, \tag{7.3}$$

we can effectively decrease the total energy of the system monotonically during some range of time in simulation. Figure 7.1b shows the total energy over time for this evolution equation with $\sigma_k = 1$. This is a new proposal that needs to be studied further.

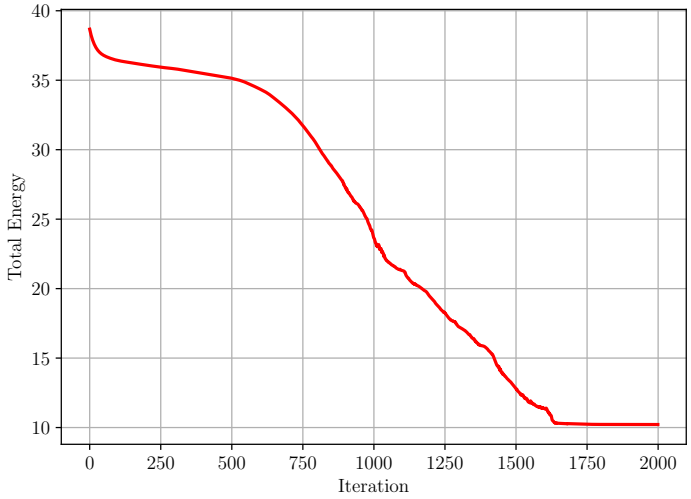
7.1.2 Topological Transitions

The algorithm successfully handles triple lines flippings, surface transitions, and grain removals. Notice that the topology of the grain structure could be complex but the algorithm is still able to manage the topological changes since this is done by Voronoi tessellation itself. A grain boundary flipping is shown in Figure 7.2. A triple line begins to shrink (Figure 7.2a), then it collapses into a single point (Figure 7.2b), and finally this unstable configuration evolves generating a new triple line (Figure 7.2c).

Figure 7.3 shows a surface transition. A surface begins to reduce its area (Figure 7.3a, 7.3b) until it becomes a vertex (Figure 7.3c).

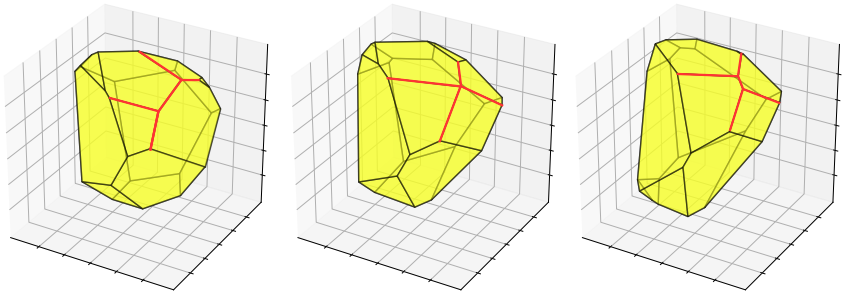


(a) Total energy of a three-dimensional simulation with the Implicit-transition model.



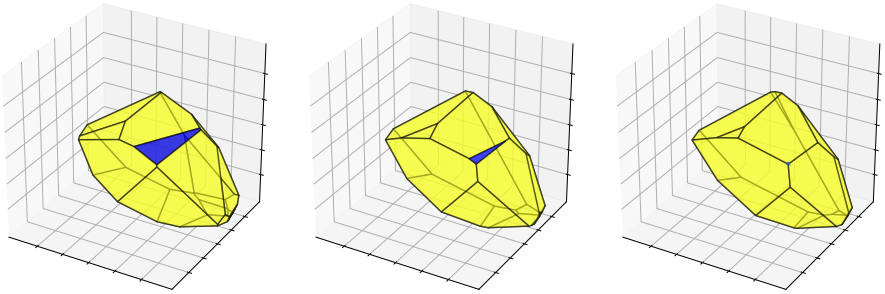
(b) Total energy of a three-dimensional simulation with the Implicit-transition model with the normal vectors evolution equation.

Figure 7.1: Total energy of three-dimensional Models: Implicit-transition Model and evolution with normal vectors.



(a) (b) (c)

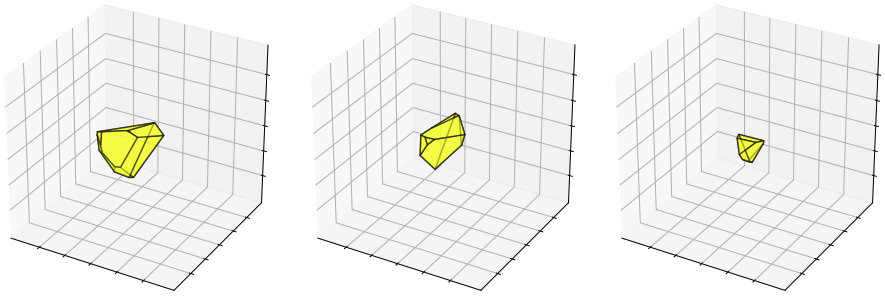
Figure 7.2: Flipping scheme on a single three-dimensional grain. Some triple lines and neighbor grains are omitted.



(a) (b) (c)

Figure 7.3: Surface transition in a three-dimensional grain. Triple lines of other related grains are omitted.

Finally grains removal are also observed, see Figure 7.4. The grain reduces its volume from $V \approx 5 \cdot 10^{-4}$ to $V \approx 1.2 \cdot 10^{-4}$ until the threshold V_{ext} is reached with $V \approx 9.6 \cdot 10^{-5}$. In experiments V_{ext} is set between 10^{-4} and 10^{-5}.



(a) (b) (c)

Figure 7.4: Grain removal stages. The grain loses boundaries (surfaces) until its volume is less than V_{ext}. Then the grain is removed by means of removing its generator point.

Chapter 8

Generation of Two-dimensional statistics from a Three-dimensional Grain Growth Model

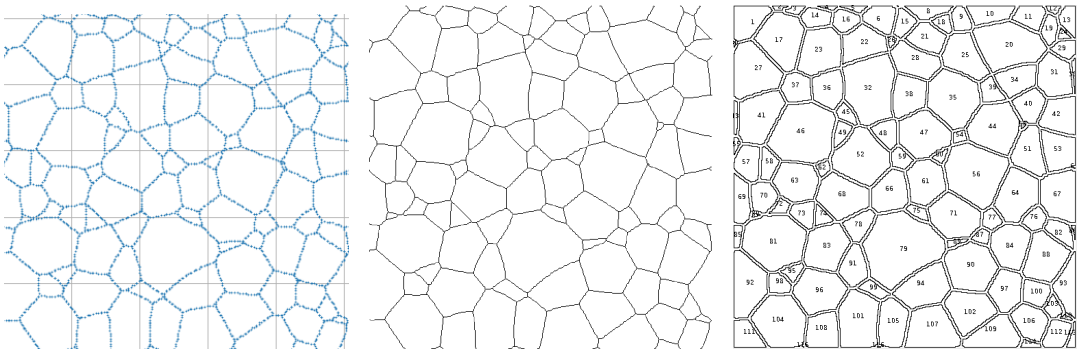
This chapter is focused in obtaining two-dimensional statistics from an existing three-dimensional model. Esedoğlu et al. [8] presented an algorithm for simulating mean curvature motion of network interfaces in $\mathbb{R}^d$ with arbitrary surface tensions σ. An specific application of this algorithm is precisely grain growth in polycrystalline materials where the interest is focused in $d = 3$ and the two-dimensional statistics that can be generated from slices of the resulting grain structure, as if it were a real bulk of material.

8.1 Generation of statistics

The code that implements the algorithm in [8] is available online and implemented in MATLAB at [7]. It receives as input a three-dimensional Voronoi tessellation and an array of random orientations, number of iterations, time step size and a grid size. The output is the final grain configuration and the related orientations.

The procedure consists first in perform the numerical simulation and export the resulting grains in `.mat` format. Further post-processing of data is done in Python. The Scipy library allows to import the `.mat` output and store it as Numpy arrays. This output then is interpreted to build the gran surfaces. The next step is to generate the slice of the original cube where each grain has to be intersected with a plane.

A larger grid is desirable for simulations, but this requires more memory to store the grain configuration. Therefore the initial cut is expanded by some factor to obtain a larger slice. Knowing the plane we want to obtain, we recover the discrete data that lies in this plane, obtaining a clean two dimensional description as shown in Figure 8.1a.



(a) (b) (c)

Figure 8.1: Discrete and continuous representations of the two-dimensional slide obtained from the three-dimensional simulation.

In order to generate a representation with complete boundaries, i.e., without discontinuities along boundaries, the obtained data is processed with two image analysis tools: *dilation* and *skeletonize*. Dilation operation allows the structure to fulfill the spaces between elements in grain boundaries, but this generates very thick boundaries. Skeletonize operation allows to obtain the skeleton of the dilated image, obtaining a clean picture of grain boundaries as shown in Figure 8.1b. Under the same idea of reduce computation, several cuts of this kind are extracted from the cube over the x, y and z-axis.

Using the image analysis software ImageJ [29] we can obtain statistics of grain structure. An internal processing must be done before computing statistics, the image must be converted to 8-bit and binarized to obtain a black-white image easier to analyze. ImageJ can detect grains obtain their areas as shown in Figure 8.1c. Notice that the obtained areas does not consider periodic boundary conditions.

8.2 Numerical Experiments

Experiments were performed with 1,000 initial grains and 1,000 random orientations with distribution $\alpha \sim \text{Unif}(0, 2\pi)$, running 10 time-steps of size $\Delta t = 5 \cdot 10^{-4}$ with a computational cubic grid of size 128^3, which left at the end approximately 500 grains. In order to extract statistics, six runs of the simulation were performed, each time with different initial conditions. ImageJ allows to obtain grain areas easily.

Figure 8.2 shows the grain relative area distribution after simulation. In this simulation the surface tensions -or grain boundary energy- are all equal to 1, which is an isotropic case. It is clear that the distribution is not log-normal since accumulates small grains which can be seen at the left of the log scale plot.

Figure 8.3 shows the grain relative area distribution after simulation. In this simulation the surface tensions, which are the analogue to the grain boundary energy

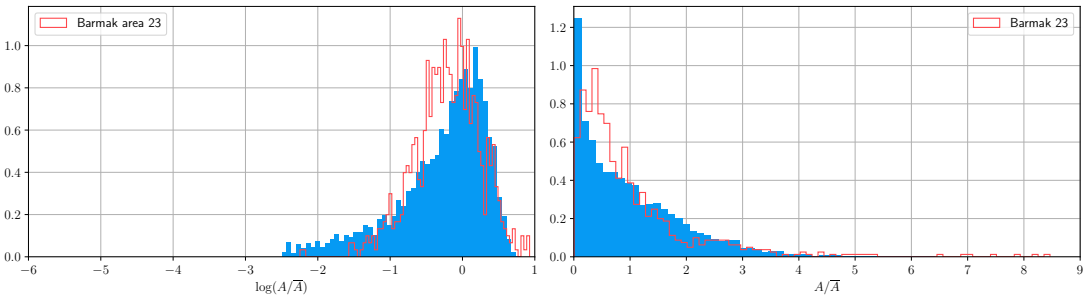


Figure 8.2: Distribution of relative areas for two-dimensional cuts with surface tensions equal to 1.

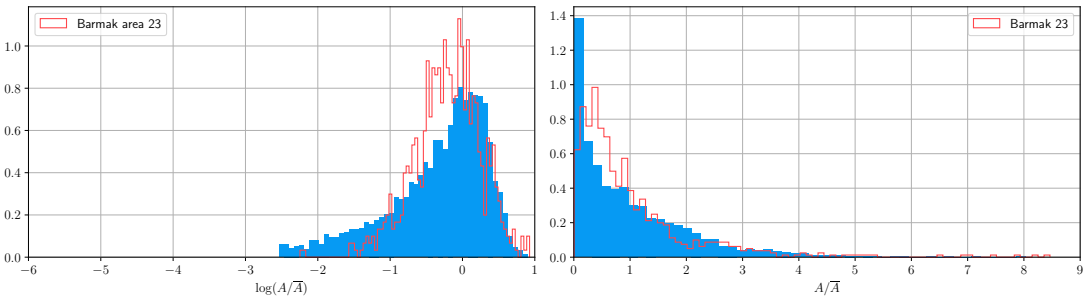


Figure 8.3: Distribution of relative areas for two-dimensional cuts with surface tensions obeying Read-Shockley function.

in two dimensions, are given by the Read-Shockley function:

$$\gamma(\theta) = \begin{cases} \dfrac{\theta}{\theta_1}\left[1 - \log\left(\dfrac{\theta}{\theta_1}\right)\right] & \theta \leq \theta_1 \\ 1 & \theta > \theta_1 \end{cases}$$

with $\theta_1 = 15$ is a fixed value and thus only one degree of freedom is used. Again the distribution shows a tail which implies the existence of small grains in the network. We also observe the decrease of grains with relative size close to 1 (bins near 0 in log scale).

The results of the statistics may be affected in part due to image resolution and also by the error introduced by the technique used to obtain the grain area data, which required fine tuning and ignored the periodic boundary condition which may have lead to introduce artificial grain areas.

Chapter 9

Conclusions

The present Thesis studied two-dimensional and three-dimensional models of grain growth and nucleation via mathematical modeling and analysis, numerical computations and statistics. Figure 9.1 presents an overview about the addressed topics.

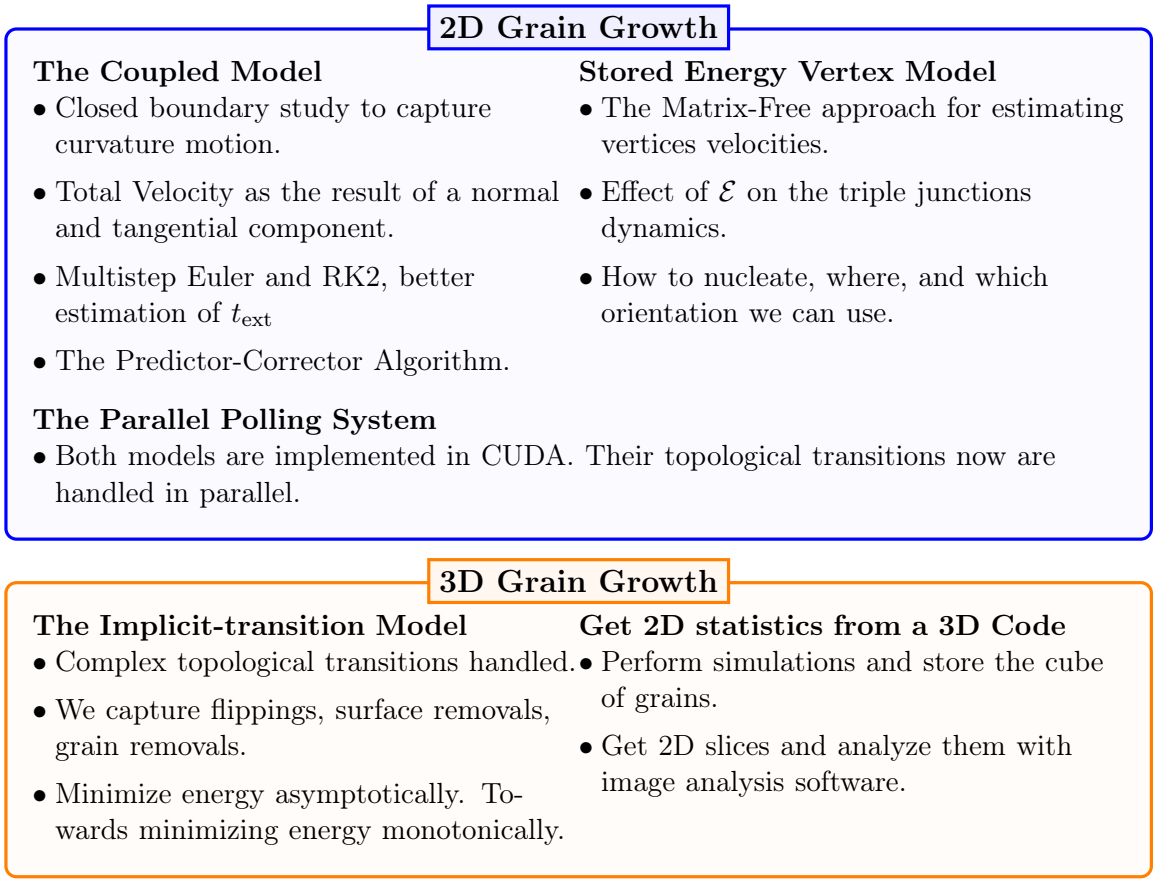


Figure 9.1: Overview of the addressed topics in this Thesis.

In relation to the grain structure, this study delved into its regular topology of a graph in a torus, remarking the Euler's formula between the number of vertices M, boundaries K and grains N of the form $M - K + N = 0$. It also shows that this relation is preserved under the topological transitions of flipping and grain removal and thus along the numerical simulations of the models that was implemented.

The Coupled Model, first presented in [27], was improved to capture the curvature based motion and also to help the interior points - which are an abstract element for grain boundaries - to preserve the stability along the simulation. The curvature based motion was recovered by studying motion in regular and non-regular closed boundaries theoretically and numerically. This yielded that the velocity of the interior points needs a correction proportional to $1/\left\|\mathbf{l}_i\right\|$. This effectively makes small grains to be removed faster, as expected by curvature motion.

On the other hand the interior points stability is not preserved using the vertices and interior points motion equations presented in (4.4) and (4.6). We developed a tangential component of the velocity that helps the interior points to move in a direction of energy minimization and also to preserve the initial equispaced positions. The total velocity of the interior points therefore is described by a normal component and a tangential component. We also developed multisteps algorithms that helps to compute the boundaries extinction time more precisely, avoiding delays in topological transitions. Since the curvature of the boundaries may increase a lot near its collapse, the velocities of the interior points grows and the extinction time of boundaries is not reliable. In order to control the velocities and knowing a candidate steady state given by the straight line between the vertices of the boundary, a predictor-corrector algorithm was developed to scale the velocities, stabilize the motion and prevent such growth that also delays flippings.

Numerical experiments shows that given the set of parameters that propitiate curvature motion we recover the associated statistics. Dihedral angles are around $2\pi/3$ and relative areas distribution shows few small grains, although it is not a log-normal distribution. Under anisotropic grain boundary energy, the GBCD is recovered succesfully. Von Neumann-Mullins relation is also approximated.

An extended Vertex Model was developed, the Continuous Stored Energy Vertex Model, which considers the introduction of an intragranular energy which allows a grain network to nucleate, that is, to create small grains and let them grow. We analyzed when a nucleated grain can grow and we determined a criteria to choose a vertex to introduce the new grain such that will grow. This grain effectively adds more energy to the system, but this grain helps to energy minimization by its growth. The simulations recover successfully a stage of grain growth followed by nucleations that will eventually replace all the original grains in the system and will evolve as a grain growth model. We also tested two values of orientations for the nucleated grains, orientation zero for all grains nucleated or a local optimized orientation that minimized the added energy to the system in function of the neighbor grains. Statistics shows that the stages of grain growth and nucleation can be identified clearly,

and that the GBCD is recovered and also feels the consequence of choosing certain orientation to nucleate.

The Coupled Model as well as the Stored Energy Vertex Model were implemented in GPU, and efforts were directed to parallelize both models using the approach of defining the basic units of works, which ultimately were vertices and boundaries. Operations, such as compute boundaries velocities and move their positions, or compute vertices energies, are carried in parallel for each boundary and vertex respectively. A demanding task to be parallelized was the management of topological transition, a sequential algorithm that had to be posed again to be programmed in parallel to take advantage of parallelism. The final algorithm was the Parallel Polling System, which is essentially a fixed point iteration that ban all unsafe flippings that, when executed in parallel, lead to race conditions.

Given the acquired knowledge of topological transitions in two dimensional grain growth, an extension of the Vertex Model in three dimensions was modeled with the objective of avoid topological transitions since in three dimensions they becomes more difficult to handle. This lead to an Implicit-transition Model [26] which defines the evolution equations of the centroids of the Voronoi tessellation instead of the quadruple junctions since moving each quadruple junction by itself breaks the planar grain surfaces that represents the extension of the Vertex Model to three dimensions. To avoid the topological transitions at each time step the new grain configuration is given by a new tessellation. We observed that this effectively decreases the energy asymptotically -but not monotonically, as sometimes the energy of the system increases- and we also observe the existence of continuous motion and topological transitions such as flippings, grain removals and grain surfaces becoming triple lines.

A three-dimensional model was studied to obtain two-dimensional statistics. The model, based on mean curvature motion, was simulated and several outputs were obtained which consisted in cubes of grains with periodic boundary conditions. Several slices in x,y and z-axis were extracted and analyzed using image analysis tools. The results of the analysis are grain areas with units relative to the size of the image given, therefore the areas are normalized and then relative grain area distributions were obtained. The distribution from slices showed a tail towards small grains, which clearly deviates from a log-normal distribution and experimental data provided. This might be also related not by the model output itself but by the error introduced by the technique used to obtain the grain areas which required fine tuning and ignored the periodic boundary conditions.

9.1 Future Work

The Coupled Model as well as the Stored Energy Vertex Model were implemented in GPU with CUDA. Efforts were made to reduce computation time, for example reducing global memory access and allocation, but there is still room for optimization

for this architecture. It is important to obtain a performance profile of the codes in order to identify potential bottlenecks and improve the code.

The Coupled Model has the flexibility to manage more data structure related to vertices, boundaries and grains. For example, the introduction of stored energy and nucleation might be interesting and challenging. Nucleation sites study shall be revisited to include interior points as possible candidates sites.

Three-dimensional models inspired in the Implicit-transition Model can be developed further, for example, to look for an evolution equation of the Voronoi centroids that minimize monotonically the energy, for example the presented equation in (7.3) which decreases the energy monotonically during a transient state. This proposal needs to be studied further and we are currently working in this model.

The election of a Δt for numerical simulations is still an issue since the current method to obtain stable results is to try several values. If we were able to set an optimal Δt we would be in a good setting to automatize even more the simulations and statistics extractions. The Periodic Hausdorff Distance used in [16] it is a good choice to analyze the convergence of the grain structure and determine Δt.

Bibliography

[1] K. Barmak, E. Eggeling, D. Kinderlehrer, R. Sharp, S. Ta'asan, A. Rollett, and K. Coffey. Grain growth and the puzzle of its stagnation in thin films: The curious tale of a tail and an ear. *Progress in Materials Science*, 58(7):987–1055, aug 2013.

[2] L. A. Barrales-Mora, G. Gottstein, and L. S. Shvindlerman. Three-dimensional grain growth: Analytical approaches and computer simulations. *Acta Materialia*, 56(20):5915–5926, 2008.

[3] J. W. Barrett, H. Garcke, and R. Nürnberg. A parametric finite element method for fourth order geometric evolution equations. *Journal of Computational Physics*, 222(1):441–467, mar 2007.

[4] J. W. Barrett, H. Garcke, and R. Nürnberg. Numerical approximation of gradient flows for closed curves in $\mathbb{R}^d$. *IMA Journal of Numerical Analysis*, 30(1):4–60, jan 2010.

[5] M. Bernacki, R. Logé, and T. Coupez. Level set framework for the finite-element modelling of recrystallization and grain growth in polycrystalline materials. *Scripta Materialia*, 64(6):525–528, mar 2011.

[6] J. G. Brons and G. B. Thompson. A comparison of grain boundary evolution during grain growth in fcc metals. *Acta Materialia*, 61(11):3936–3944, 2013.

[7] S. Esedoğlu. Motion of grain boundaries in polycrystalline materials. `http://www.math.lsa.umich.edu/~esedoglu/Research/grains/grains.html`, 2015. Accessed: 2018-09-30.

[8] S. Esedoğlu and F. Otto. Threshold Dynamics for Arbitrary Surface Tensions. *Communications on Pure and Applied Mathematics*, 68(5):808–864, 2015.

[9] A. Ferro and M. Fortes. The Elimination of Grains and Grain Boundaries in Grain Growth. *Interface Science*, 5(4):263–278, 1997.

[10] C. Herring. Surface tension as a motivation for sintering. *The physics of powder metallurgy*, 27(2):143–179, 1951.

[11] D. Kinderlehrer, J. Lee, I. Livshits, S. Ta'asan, and P. Yu. Multiscale modeling and simulation of grain boundary evolution. In *44th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, page 1611, 2003.

[12] D. Kinderlehrer, I. Livshits, and S. Ta'asan. A Variational Approach to Modeling and Simulation of Grain Growth. *SIAM Journal on Scientific Computing*, 28(5):1694–1715, 2006.

[13] E. A. Lazar, J. K. Mason, R. D. MacPherson, and D. J. Srolovitz. A more accurate three-dimensional grain growth algorithm. *Acta Materialia*, 59(17):6837–6847, 2011.

[14] K. Mikula and D. Sevcovic. Evolution of Plane Curves Driven by a Nonlinear Function of Curvature and Anisotropy. *SIAM Journal of Applied Mathematics*, 61(5):1473–1501, jan 2001.

[15] W. W. Mullins. Two-dimensional motion of idealized grain boundaries. *Journal of Applied Physics*, 27(8):900–904, 1956.

[16] C. A. Muñoz. Análisis de Código en GPU de Crecimiento de Granos. Bachelor's thesis, Universidad Técnica Federico Santa María, 2017.

[17] J. Nickolls, I. Buck, M. Garland, and K. Skadron. Scalable Parallel Programming with CUDA. *Queue*, 6(2):40–53, Mar. 2008.

[18] NVIDIA Corporation. NVIDIA Accelerated Computing, CUDA Zone. `https://developer.nvidia.com/cuda-zone`, 2018. Accessed: 2018-09-23.

[19] J. Orend, F. Hagemann, F. B. Klose, B. Maas, and H. Palkowski. A new unified approach for modeling recrystallization during hot rolling of steel. *Materials Science and Engineering A*, 647(2015):191–200, 2015.

[20] K. Piękoś, J. Tarasiuk, K. Wierzbanowski, and B. Bacroix. Development of Two-Dimensional Vertex Type Model. *Materials Science Forum*, 467-470(I):653–658, oct 2004.

[21] K. Piękoś, J. Tarasiuk, K. Wierzbanowski, and B. Bacroix. Generalized vertex model of recrystallization–application to polycrystalline copper. *Computational Materials Science*, 42(4):584–594, 2008.

[22] K. Piękoś, J. Tarasiuk, K. Wierzbanowski, and B. Bacroix. Stochastic vertex model of recrystallization. *Computational Materials Science*, 42(1):36–42, 2008.

[23] D. Reem. The geometric stability of voronoi diagrams with respect to small changes of the sites. In *Proceedings of the twenty-seventh annual symposium on Computational geometry*, pages 254–263. ACM, 2011.

[24] A. D. Rollett and P. Manohar. The Monte Carlo Method. In D. Raabe, F. Roters, F. Barlat, and L.-Q. Chen, editors, *Continuum Scale Simulation of Engineering Materials: Fundamentals - Microstructures - Process Applications*, chapter 4, pages 76–113. John Wiley & Sons, Wiley, 2006.

[25] F. Sausset and G. Tarjus. Periodic boundary conditions on the pseudosphere. *Journal of Physics A: Mathematical and Theoretical*, 40(43):12873, 2007.

[26] A. H. Sazo and C. E. Torres. An implicit-transition model for numerical simulation of 3d grain growth. In *2017 36th International Conference of the Chilean Computer Science Society (SCCC)*. IEEE, 2017.

[27] A. H. J. Sazo. Implementación Numérica-Computacional avanzada en GPU y Análisis de un Modelo 2D Acoplado de Grain Growth. Bachelor's thesis, Universidad Técnica Federico Santa María, 2016.

[28] A. H. J. Sazo, C. E. Torres, P. Ibarra, A. Sanhueza, F. Casas, M. Emelianenko, and D. Golovaty. Derivation and Numerical Implementation of a 2D Grain Growth Coupled Model. Work in progress.

[29] C. A. Schneider, W. S. Rasband, and K. W. Eliceiri. NIH Image to ImageJ: 25 years of image analysis. *Nature methods*, 9(7):671–675, 2012.

[30] M. Syha and D. Weygand. A generalized vertex dynamics model for grain growth in three dimensions. *Modelling and Simulation in Materials Science and Engineering*, 18(1):015010, 2010.

[31] G. B. Thomas. *Cálculo, varias variables*. Pearson Education, 12 edition, 2010.

[32] C. E. Torres, M. Emelianenko, D. Golovaty, D. Kinderlehrer, and S. Ta'asan. Numerical Analysis of the Vertex Models for Simulating Grain Boundary Networks. *SIAM Journal on Applied Mathematics*, 75(2):762–786, 2015.

[33] L. N. Trefethen. *Spectral Methods in MATLAB*, volume 10. SIAM, 2000.

[34] B. Van der Boor. Curvature-driven grain growth. Master thesis, Delft University of Technology, 2016.

Appendix A

Tools and Technical Specifications

Here are listed the several tools and environments for software developing and testing that were used during this work.

A.1 Operating Systems

The two main environments used were CentOS 7 and Fedora 28. All software listed was extensively used in Fedora 28. CentOS 7 is the main distribution at UTFSM HPC Cluster [1] and secondary Laboratory PC.

A.2 Programming Languages

- CUDA C/C++ release 8.0, V8.0.61
- GCC 5.3.1 (This is for compatibility with CUDA 8.0)
- Python 3.6.5 (Anaconda)
- Python 2.7.15 (Anaconda)
- Anaconda 4.5.11

A.3 Simulation Software

- MATLAB 2016b
- ImageJ 1.50h

[1] `www.hpc.utfsm.cl`

A.4 Numerical Libraries

- Numpy 1.15.1 (Python2 and Python3)
- Pymesh 0.2.1 (Python3)
- Scipy 1.1.1 (Python3)
- Sympy 1.1.1 (Python3)
- Pyvoro 1.3.2 (Python2)
- Voro++ 0.4.5 (C++)

A.5 Art and Text Libraries

- Matplotlib 2.2.2 (Python3)
- TiKz 3.0.1a
- Mayavi 4.5.0 (Python2)
- LaTeX 2_ε

A.6 Hardware

Numerical simulations were run in three environments. Massive simulations (over 50000 grains) were run at CCTVal cluster. Nodes have NVIDIA Tesla M-2050 and NVIDIA Tesla K20m. Smaller simulations and debugging tests were run at Laboratory workstation and Personal Laptop. PC has a NVIDIA Geforce GTX 750 Ti. Personal laptop has NVIDIA Geforce 920m. Specific hardware (CPU, RAM, HDD) for each environment is listed at Table A.1

CCTVal cluster	
CPU	Intel(R) Xeon(R) CPU E5-2643 v2 @ 3.50GHz
RAM	32 GB
HDD	100 GB
GPU	NVIDIA Tesla M-2050 and NVIDIA Tesla K20m
Laboratory workstation	
CPU	AMD FX(tm)-4300 Quad-Core Processor
RAM	16 GB
HDD	Western Digital 500 GB + 1 TB
GPU	NVIDIA GeForce GTX 750 Ti
Personal laptop	
CPU	Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz
RAM	12 GB
HDD	Intel 545s Series SATA III 2.5" SSD 512 GB
GPU	NVIDIA GeForce 920m

Table A.1: Hardware specifications for work environments.