

**Universidade Federal Fluminense**

**JONATHAS LUIS GROETARES FERREIRA**

**Simulação orientada a objetos de transformações de  
fase em metais**

VOLTA REDONDA

2018

**JONATHAS LUIS GROETARES FERREIRA**

# **Simulação orientada a objetos de transformações de fase em metais**

Dissertação apresentada ao Programa de Pós-graduação em Modelagem Computacional em Ciência e Tecnologia da Universidade Federal Fluminense, como requisito parcial para obtenção do título de Mestre em Modelagem Computacional em Ciência e Tecnologia. Área de Concentração: Métodos matemáticos e computacionais aplicados à engenharia e ciência

Orientador:

Paulo Rangel Rios

Coorientador:

Weslley Luiz da Silva Assis  
Tiago Araújo Neves

UNIVERSIDADE FEDERAL FLUMINENSE

VOLTA REDONDA

2018

Ficha catalográfica automática - SDC/BEM  
Gerada com informações fornecidas pelo autor

F383s Ferreira, Jonathas Luis Groetares  
Simulação orientada a objetos de transformações de fase em metais / Jonathas Luis Groetares Ferreira ; Paulo Rangel Rios, orientador ; Weslley Luiz da Silva Assis, coorientador. Volta Redonda, 2018.  
70 f. : il.

Dissertação (mestrado)-Universidade Federal Fluminense, Volta Redonda, 2018.

DOI: <http://dx.doi.org/10.22409/PPG-MCCT.2018.m.13980271706>

1. Autômato Celular. 2. Cone Causal. 3. Transformação de fase. 4. Recristalização. 5. Produção intelectual. I. Rios, Paulo Rangel, orientador. II. Assis, Weslley Luiz da Silva, coorientador. III. Universidade Federal Fluminense. Escola de Engenharia Industrial e Metalúrgica de Volta Redonda. IV. Título.

CDD -

# Simulação orientada a objetos de transformações de fase em metais

Jonathas Luis Groetares Ferreira

Dissertação apresentada ao Programa de Pós-graduação em Modelagem Computacional em Ciência e Tecnologia da Universidade Federal Fluminense, como requisito parcial para obtenção do título de Mestre em Modelagem Computacional em Ciência e Tecnologia. Área de Concentração: Métodos matemáticos e computacionais aplicados à engenharia e ciência.

Aprovada por:

Paulo Rangel Rios

Prof. Paulo Rangel Rios, Ph.D. / EEIMVR-UFF(Presidente)

Yoisell Rodríguez Núñez

Prof. Yoisell Rodríguez Núñez, D.Sc. / EEIMVR-UFF

Mabelle Biancardi Oliveira de Medeiros

Mabelle Biancardi Oliveira de Medeiros, D.Sc. / CEFET-RJ

Volta Redonda, 26 de Novembro de 2018.

*Aos pesquisadores do Núcleo de Modelamento Microestrutural da EEIMVR.*

# Agradecimentos

Agradeço sobretudo a Deus, Criador de todas as coisas e em quem estão escondidos todos os tesouros da sabedoria e do conhecimento. Agradeço à minha família, especialmente à minha mãe Luisa Helena Groetares. Agradeço aos amigos que dentro e fora do ambiente acadêmico têm me apoiado de diversos modos. Agradeço aos servidores da Universidade Federal Fluminense pelo gentil atendimento ao longo dos anos. Agradeço aos professores que me deram as bases essenciais à minha carreira e me inspiram para que eu possa usar o conhecimento de forma útil a todos.

“*Nanos gigantium humeris insidentes.*”

*Bernardus Cartonensis*

# Resumo

O estudo de transformações de fase em metais é realizado por meio da simulação computacional com uso do paradigma da orientação à objetos. A Programação Orientada a Objetos permite que seja feita uma abstração do que ocorre na realidade e o programa é construído em partes que representam os entes reais do fenômeno. Essa abordagem possibilita, entre outras coisas, que o programa seja desenvolvido por grupos de pesquisadores de forma mais eficiente. Neste trabalho são implementadas dois tipos de nucleação e o crescimento é simulado pelos métodos Autômato Celular e Cone Causal. O programa realiza a exportação de arquivos que podem ser usados para visualização da microestrutura simulada e estudos estequiométricos. Os resultados gerados na simulação têm bom acordo com o que é observado na Natureza e com as teorias sobre o tema.

# Abstract

The study of phase transformations in metals is carried out through computational simulation using the object orientation paradigm. Object Oriented Programming allows an abstraction to be made of what actually occurs, and the program is built into parts that represent the real entities of the phenomenon. This approach allows, among other things, that the program be developed by groups of researchers more efficiently. Two types of nucleation are implemented and the growth is simulated by the Cellular Automata and Causal Cone methods. The program performs the export of files that can be used for visualization of the simulated microstructure, stoichiometric studies and statistical analyzes. The results generated in the simulation have good agreement with what is observed in Nature and with the theories on the subject.

# **Palavras-chave**

1. Autômato Celular
2. Cone Causal
3. Crescimento
4. Nucleação
5. Recristalização

# Glossário

AC	:	Autômato Celular
CC	:	Cone Causal
$G$	:	Velocidade da interface de um grão
$\langle G \rangle$	:	Velocidade média dos contornos de grão
JMAK	:	Referência aos pesquisadores Johnson, Mehl, Avrami e Kolmogorov
$N_V$	:	Número de núcleos por unidade de volume
$P$	:	Ponto em $\mathbb{R}^3$ pertencente a uma aresta de um poliedro de Voronoi
POO	:	Programação Orientada a Objetos
$R$	:	Raio de um cone causal
$S_{V\alpha\beta}$	:	Fração entre a área interfacial das fases $\alpha$ e $\beta$ e o volume do espécime
$S_{V\beta\beta}$	:	Fração entre a área interfacial dos grãos transformados e o volume do espécime
$t$	:	Tempo
$u$	:	Parâmetro em equação de reta
$V(t)$	:	Velocidade radial de um cone causal
$\mathbf{V}_1$ e $\mathbf{V}_2$	:	Par de pontos em $\mathbb{R}^3$ representando vértices de um poliedro de Voronoi
$V_V$	:	Fração volumétrica
$\mathbf{x}$	:	Vetor posição
$\alpha$	:	Fase matriz
$\beta$	:	Nova fase sendo transformada
$\sigma$	:	Parâmetro que depende da geometria de crescimento do Cone Causal ou do Autômato Celular
$\omega$	:	Parâmetro de ajuste para o método Autômato Celular ou Cone Causal

# Sumário

<b>Lista de Figuras</b>	<b>xí</b>
<b>1 Introdução</b>	<b>13</b>
1.1 Objetivos . . . . .	15
1.1.1 Objetivo geral . . . . .	15
1.1.2 Objetivos específicos . . . . .	15
<b>2 Nucleação e Crescimento de Fases em metais</b>	<b>16</b>
2.1 Recristalização de ligas monofásicas . . . . .	18
2.1.1 O Processo de Nucleação na Recristalização . . . . .	19
2.1.2 O Processo de Crescimento na Recristalização . . . . .	20
2.1.3 As Leis da Recristalização . . . . .	20
2.2 Simulação Computacional de Transformações de Fase . . . . .	21
<b>3 Métodos de Crescimento</b>	<b>23</b>
3.1 O método Autômato Celular . . . . .	23
3.2 O método Cone Causal . . . . .	25
<b>4 A programação Orientada a Objetos</b>	<b>29</b>
<b>5 Desenvolvimento do Código</b>	<b>32</b>
5.1 Descrição do problema . . . . .	32
5.2 Descrição das classes . . . . .	33

5.2.1	Classe Dados . . . . .	33
5.2.2	Classe Matriz . . . . .	33
5.2.3	Classe Nucleação . . . . .	34
5.2.4	Classe Autômato . . . . .	35
5.2.5	Classe Cone . . . . .	36
5.2.6	Classe Saída . . . . .	37
<b>6</b>	<b>Resultados e Discussão</b>	<b>38</b>
<b>7</b>	<b>Conclusões e Trabalhos Futuros</b>	<b>47</b>
7.1	Conclusões . . . . .	47
7.2	Trabalhos Futuros . . . . .	48
<b>Apêndice A – Desenvolvimento de Software para Determinação de Vértices e Arestas de Grãos</b>		<b>50</b>
<b>Apêndice B – Modo de uso do ParaView</b>		<b>55</b>
B.1	Visualização com arquivos .vtk . . . . .	55
B.2	Visualização com arquivos .csv . . . . .	56
<b>Apêndice C – Script na linguagem R usado no trabalho</b>		<b>58</b>
<b>Referências</b>		<b>66</b>

# Listas de Figuras

2.1	Curva esquemática típica para uma transformação de fase que segue a equação de JMAK (Adaptada) [26]. . . . .	17
2.2	Recristalização de cobre deformado durante recozimento in-situ em Microscópio eletrônico de varredura [16]. . . . .	19
2.3	Fluxograma de uma simulação de nucleação e crescimento. . . . .	22
3.1	Esquema de vizinhanças e condições de contorno usadas neste trabalho. Os retângulos destacados simbolizam a vizinhança devido às condições de contorno periódicas (Adaptado [2]). . . . .	25
3.2	Representação gráfica do cone causal (Adaptada [11]). . . . .	27
3.3	Esquema da evolução e interação de grãos pelo método CC. . . . .	28
5.1	Fluxograma do processo. . . . .	33
5.2	Esquema de crescimento 2D de AC com vizinhança de Von Neumann: (a) → (b). A cor vermelha representa uma célula já transformada e a azul, uma célula que será transformada na iteração atual. . . . .	35
5.3	Esquema do efeito das condições de contorno periódicas na área de influência de um cone causal em um determinado tempo. (a) Área convencional e (b) efeito das condições de contorno. . . . .	36
6.1	Crescimento por Autômato Celular a partir de nucleação uniforme. Transformação de (a) 10% , (b) 50% e (c) 100%. . . . .	39
6.2	Crescimento por Cone Causal a partir de nucleação uniforme. Transformação de (a) 10% , (b) 50% e (c) 100%. . . . .	39
6.3	Crescimento por Autômato Celular a partir de nucleação periódica. Transformação de (a) 10% , (b) 50% e (c) 100%. . . . .	40
6.4	Crescimento por Cone Causal a partir de nucleação periódica. Transformação de (a) 10% , (b) 50% e (c) 100%. . . . .	41

---

6.5	Comparação de microestrutura gerada pelo método (a) AC e (b) CC. Vista da face que aponta para o sentido positivo do eixo z. . . . .	41
6.6	Esquema explicativo dos pontos que surgem em áreas de outras cores na microestrutura de AC. . . . .	41
6.7	Comparação das microestruturas obtidas por simulação de (a) AC e (b) CC com (c) um modelo de microestrutura de austenita (adaptada [3]) e com (d) uma micrografia real de austenita recristalizada (adaptada [36]). . . . .	42
6.8	Evolução temporal de $V_V$ das simulações pelos métodos AC e CC com nucleação uniforme comparadas ao calculado pela Eq. 2.1. . . . .	43
6.9	Evolução temporal de $S_{V\alpha\beta}$ das simulações pelos métodos AC e CC com nucleação uniforme comparadas ao calculado pela Eq. 2.3. . . . .	44
6.10	Comparação entre o caminho microestrutural das simulações de AC e CC com nucleação uniforme. . . . .	44
6.11	Comparação entre a velocidade média de crescimento dos contornos de grãos das simulações de AC e CC com nucleação uniforme. . . . .	45
A.1	Representação dos vértices calculados e dos núcleos (esferas maiores) gerados em uma distribuição (a) uniforme e (b) periódica. . . . .	51
A.2	Representação dos vértices e arestas calculados com auxílio da biblioteca Voro++ e dos núcleos (esferas maiores) gerados em uma nucleação (a) e (b) uniforme e (c) periódica. Em (a) foram exibidas apenas as arestas encontradas, enquanto em (b) e (c) também são exibidos os vértices . . . . .	53
A.3	Representação dos vértices e arestas calculados com auxílio da biblioteca Voro++ e dos núcleos gerados em uma nucleação uniforme em 2D. . . . .	54
B.1	Exemplo de arquivo .vtk. . . . .	56
B.2	Exemplo de arquivo .csv. . . . .	57
C.1	Exemplo de arquivo para entrada de dados referentes a $V_V$ e $S_{V\alpha\beta}$ no Script de R. A primeira coluna refere-se à iteração e a segunda ao valor calculado pelos métodos da simulação. . . . .	58

# Capítulo 1

## Introdução

O trabalho científico é caracterizado pelo compartilhamento de ideias, descobertas e métodos a fim de se chegar a um maior entendimento sobre determinado assunto. Embora tenha havido diversos cientistas que se notabilizaram por terem trabalhado de forma isolada, certamente foi a herança deixada por cada pesquisador à comunidade que permitiu o desenvolvimento do conhecimento sem a necessidade de que cada nova geração tivesse de redescobrir a ciência.

Hoje, o crescimento na produção científica e o maior acesso a esta produção faz com que os pesquisadores tenham que ter uma grande eficiência em seus métodos de trabalho a fim de estarem sempre apresentando resultados relevantes para a comunidade. Sendo assim, o trabalho em grupos de pesquisa e até mesmo a continuidade de trabalhos de um pesquisador devem ser feitos de modo que se perca o mínimo de tempo adaptando-se o que já se tem ao que se deseja ter.

No contexto da Computação Científica, essa necessidade é pronunciada pelo fato de que já existe uma grande quantidade de algoritmos e códigos em várias linguagens para o estudo dos mais diversos problemas, geralmente disponíveis de forma bem acessível. Logo, para a maioria dos casos, não é preciso voltar ao início do desenvolvimento, mas sim explorar o que já existe criando melhorias e adaptações para se chegar ao resultado desejado. Além disso, é importante para o pesquisador elaborar seu trabalho tornando-o mais receptível a alterações ou acréscimos que ele mesmo implementará ou que serão fornecidos por colaboradores e sucessores. É ainda sempre importante levar em conta que os problemas cada vez mais devem ser encarados de forma multidisciplinar, requerendo conhecimento de diferentes domínios.

A Programação Orientada a Objetos é um paradigma de programação que atende essa

demandando ao ter como um de seus princípios o reaproveitamento do código à medida que se acrescentam melhorias ao programa. Em síntese, trata-se de um método de programação que modela entes do mundo real por meio de componentes reutilizáveis, os objetos. O uso dessa metodologia pode tornar grupos de desenvolvimento muito mais produtivos do que é possível com o uso de técnicas anteriores, como a programação estruturada [13].

Desta forma, foi desenvolvido um código seguindo a orientação a objetos para o estudo das transformações de fase em metais. O programa simula a nucleação e o crescimento de uma nova fase em uma fase matriz. Embora seja um tema já bastante desenvolvido, ainda não é comum tratá-lo com o uso deste paradigma de programação visando à facilitação de novos implementos e do trabalho em grupo. Teve-se como objetivo, além de realizar as simulações de transformação de fase, implementar métodos para cálculos úteis ao estudo metalúrgico do fenômeno. Este projeto atende à demanda do Núcleo de Modelamento Microestrutural da Universidade Federal Fluminense, sendo o primeiro programa deste laboratório que aplica a orientação a objetos no desenvolvimento de ferramentas para o estudo das transformações de fase ao mesmo tempo que simplifica o trabalho em grupo e possibilita um maior reaproveitamento de código à medida que novas pesquisas vão sendo introduzidas.

A aplicação da programação orientada a objetos permitiu a implementação de diferentes formas de nucleação e dos métodos Autômato Celular e Cone Causal para a realização do crescimento. Além disso, diversos algoritmos para cálculos relacionados ao estudo foram acrescentados à simulação. A grande vantagem observada foi a maior praticidade no desenvolvimento do programa e a possibilidade de criação de novos algoritmos, o que seria extremamente complexo sem o uso da orientação a objetos.

Neste trabalho, primeiramente são apresentados no Cap. 2 os principais conceitos metalúrgicos sobre Nucleação e Crescimento de fases, assim como sobre Recristalização. Também é feita uma abordagem sobre a simulação computacional desses fenômenos. Em seguida, é feita uma descrição no Cap. 3 dos métodos Autômato Celular e Cone Causal e de como eles se aplicam às simulações de Transformações de Fase. A partir da introdução desses conceitos é feita no Cap. 4 uma análise sobre as características da Programação Orientada a Objetos e de como ela pode ser útil no estudo científico. A segunda parte do trabalho é direcionada para a apresentação da estrutura do programa desenvolvido para as simulações (Cap. 5) e para a apresentação e discussão dos resultados obtidos com seu uso (Cap. 6). É feita então uma conclusão do trabalho (Cap. 7) e, por fim, o Apêndice A apresenta o desenvolvimento de um software para estudo dos vértices e arestas dos grãos,

o Apêndice B registra um breve tutorial sobre o uso do programa ParaView usado na criação das imagens das microestruturas e o Apêndice C traz o script na linguagem R usado no trabalho para criação de gráficos.

## 1.1 Objetivos

### 1.1.1 Objetivo geral

Desenvolver um programa seguindo o paradigma da orientação a objetos para a simulação de transformações de fase em metais.

### 1.1.2 Objetivos específicos

- Identificar e implementar as etapas necessárias para a simulação de nucleação e crescimento de fases;
- Implementar e comparar diferentes formas de nucleação e crescimento de fases;
- Propiciar formas de visualização dos resultados da simulação em programas apropriados e compará-las com microestruturas de materiais reais;
- Implementar métodos para cálculos relevantes ao estudo de transformações de fase;
- Manter o código preparado para novas implementações e aplicações.

## Capítulo 2

# Nucleação e Crescimento de Fases em metais

Em geral, as transformações de fase são divididas em dois processos: Nucleação e Crescimento [26]. A nucleação consiste em rearranjos atômicos que permitem a formação de pequenos agregados de uma fase mais estável ( $\beta$ ) na fase matriz ( $\alpha$ ). Esses núcleos da nova fase se desenvolvem na fase matriz enquanto for cineticamente favorável.

Enquanto a nucleação é a formação de uma interface entre a matriz e a nova fase, o crescimento é a migração dessa interface [26]. Desta forma, deve haver uma força motriz para a transformação da nova fase associada à diferença de potencial químico através da interface e um mecanismo de migração viável (ativado ou não pela temperatura). Quando a migração é termicamente ativada e há necessidade de transporte de massa para que a reação ocorra, a cinética é geralmente controlada por processos de difusão. A nucleação pode ser totalmente concluída antes do crescimento ou pode haver o surgimento de novos núcleos enquanto outros continuam crescendo.

A nucleação e o crescimento podem ser matematicamente modelados pela teoria de Johnson, Mehl [17], Avrami [4, 5, 6] e Kolmogorov [18] (JMAK). Essa teoria leva à Equação 2.1 que representa o grau de recristalização (ou fração volumétrica da fase transformada)  $V_V$  em função do tempo, considerando que a taxa de crescimento dos grãos  $G$  é constante e que toda nucleação ocorre no início da transformação (saturação de sítios). Neste trabalho não é considerado o caso de nucleação durante o crescimento, embora esteja incluso nos trabalhos de JMAK.

$$V_V(t) = 1 - \exp(-\sigma N_V G^3 t^3) \quad (2.1)$$

Na Equação 2.1,  $\sigma$  é um parâmetro que depende da forma dos grãos,  $N_V$  é o número de núcleos por unidade de volume e  $t$  é o tempo. O parâmetro  $\sigma$  será definido no Cap. 5 segundo as características dos modelos utilizados neste trabalho.

A Fig. 2.1 apresenta um típico exemplo de uma curva da Eq. 2.1 para valores arbitrários de  $\sigma$ ,  $N_V$  e  $G$ . Pode-se perceber por esse gráfico como a equação de JMAK prevê o comportamento cinético de uma transformação de fase. O formato sigmoidal do gráfico indica que é esperado que a transformação inicie-se lentamente, passando por um período de rápido crescimento da porção transformada até que chegue ao estágio final do processo com uma lenta transformação das regiões restantes. Este comportamento é observado em numerosos tipos de transformações de fase e de recristalização [26].

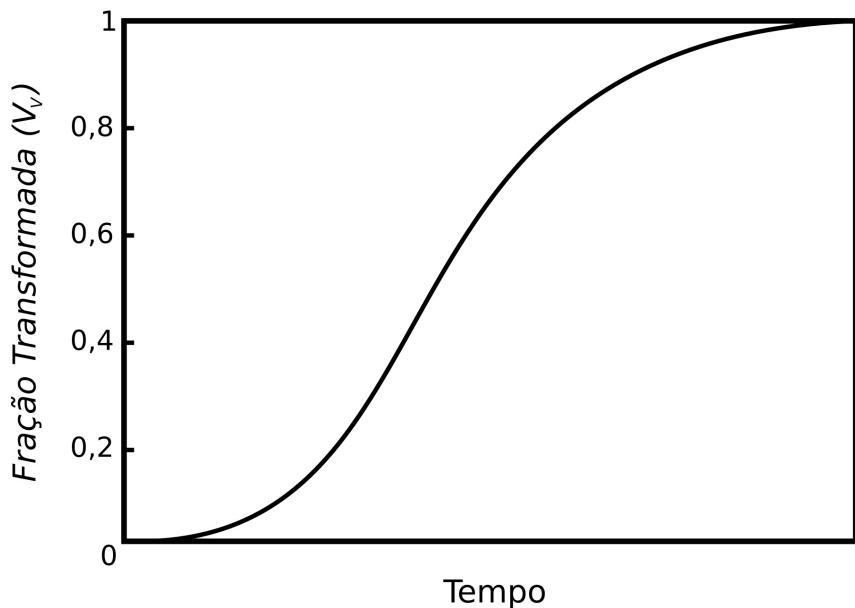


Figura 2.1: Curva esquemática típica para uma transformação de fase que segue a equação de JMAK (Adaptada) [26].

O cálculo da velocidade média dos contornos de grão  $\langle G \rangle$  pode ser feito com o uso da equação [27]:

$$\langle G \rangle = \frac{1}{S_{V\alpha\beta}} \frac{dV_V}{dt}, \quad (2.2)$$

onde  $S_{V\alpha\beta}$  é a área interfacial por unidade de volume entre as regiões transformada ( $\beta$ ) e não transformada ( $\alpha$ ).

Tem-se também uma importante expressão para o cálculo de  $S_{V\alpha\beta}$  em função de  $t$  [27]:

$$S_{V\alpha\beta}(t) = 3\sigma\omega N_V G^3 t^2 \exp(-\sigma N_V G^3 t^3). \quad (2.3)$$

Na equação 2.3,  $\omega$  é um parâmetro que depende da geometria do método de crescimento e será definido no Cap. 5. Os parâmetros  $\sigma$  e  $\omega$  não são encontrados na literatura citada no trabalho, todavia optou-se por utilizá-los tendo em vista que as equações aqui utilizadas diferem minimamente entre si para cada método de crescimento em estudo e essa diferença pode ser simplificada com o uso desses parâmetros.

Além disso, pode ser obtido o caminho microestrutural (Eq. 2.4), uma expressão de  $S_{V\alpha\beta}$  em função de  $V_V$  que gera uma curva característica para cada tipo de nucleação e crescimento [34].

$$S_{V\alpha\beta}(V_V) = 3\omega(1 - V_V) (\sigma N G^3)^{\frac{1}{3}} \left[ \ln\left(\frac{1}{1 - V_V}\right) \right]^{\frac{2}{3}} \quad (2.4)$$

As equações apresentadas nesta seção serão usadas ao longo do trabalho como forma de comparação entre os métodos de crescimento testados. Vale ressaltar que elas só são válidas quando os núcleos são todos gerados no tempo inicial e quando eles obedecem a uma distribuição uniforme no espaço.

## 2.1 Recristalização de ligas monofásicas

A recristalização em ligas metálicas monofásicas acontece de forma semelhante, porém a nucleação de uma nova fase, isto é, uma região livre de deformação, não ocorre devido a rearranjos atômicos. Sua formação deve-se a mecanismos como a migração de contornos de grão induzidas por deformação, ou por migração de subgrãos ou ainda por coalescimento de subgrãos [16].

O processo de recristalização envolve a formação de novos grãos livres de deformação e o seu subsequente crescimento até que todo material esteja recristalizado. A microestrutura é dividida em regiões recristalizadas e não recristalizadas e a fração recristalizada aumenta de 0 a 1 com o progresso da transformação como pode ser visto na Fig. 2.2. Nessa imagem é visto como os grãos recristalizados (de aspecto liso) vão cobrindo toda microestrutura deformada (de aspecto rugoso) à medida que a transformação ocorre.

O progresso da recristalização com o tempo segue um comportamento semelhante ao já discutido na Fig. 2.1.

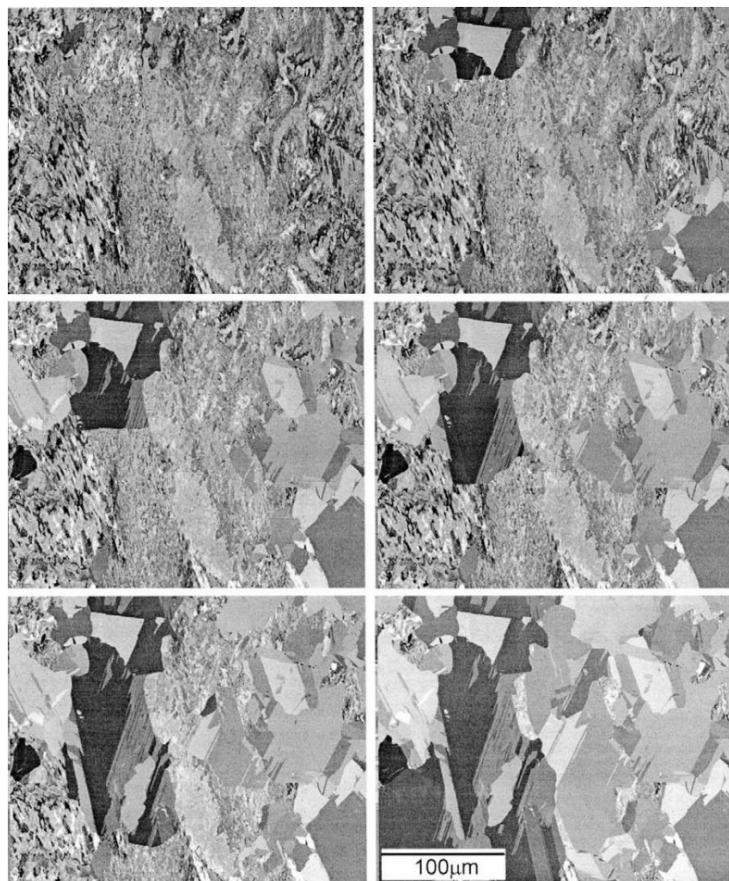


Figura 2.2: Recristalização de cobre deformado durante recozimento in-situ em Microscópio eletrônico de varredura [16].

### 2.1.1 O Processo de Nucleação na Recristalização

É difícil definir o que seria um “núcleo de recristalização”. Humphreys e Hatherly [16] o definem como sendo “um grão de baixa energia interna crescendo em um material deformado ou que sofreu recuperação do qual é separado por um contorno de grão de alto ângulo”. Em outras palavras, pode-se dizer que um núcleo de recristalização é um novo grão livre de deformação que surge e cresce em um material deformado ou que sofreu recuperação.

O termo recuperação refere-se a “mudanças em um material deformado que ocorrem antes da recristalização” [16]. Durante a recuperação, sob temperaturas abaixo das necessárias para que ocorra a recristalização, o material deformado à frio pode voltar às propriedades físicas e mecânicas, tais como resistividade e resistência mecânica, anteriores à deformação devido ao aniquilamento ou realinhamento do excessos de distorções, sendo este último processo chamado de poligonização [25]. A recuperação não ocorre apenas em materiais deformados, mas pode ocorrer sempre que houver situações de não equilíbrio como alta concentração de defeitos de linha ou de ponto [16].

### 2.1.2 O Processo de Crescimento na Recristalização

Uma vez que os novos grãos surgem, eles crescerão até completar toda área deformada acessível ou enquanto houver energia disponível. A densidade de discordâncias do cristal resulta em uma energia armazenada que é a força motriz para o crescimento dos grãos recristalizados. Uma densidade de discordâncias típica de um metal é da ordem de  $10^{15} m^{-2}$  e fornece uma pressão motriz da ordem de  $1 MPa$  [16]. Essa pressão e a consequente velocidade de crescimento dos grãos podem não ser constantes durante toda recristalização devido à concorrência com o fenômeno da recuperação.

### 2.1.3 As Leis da Recristalização

Burke e Turnbull [8], considerando a recristalização como um fenômeno de nucleação e crescimento que é controlado por um processo termicamente ativado cuja força motriz advém da energia armazenada na deformação, elaboraram 7 leis sobre o comportamento dessa transformação na maioria dos metais:

1. Uma deformação mínima é necessária para causar a recristalização.
2. Quanto menor o grau de deformação, maior a temperatura requerida para causar a recristalização.
3. O aumento do tempo de recozimento diminui a temperatura necessária para a recristalização.
4. O tamanho de grão final depende mais do grau de deformação e menos da temperatura de recozimento, sendo menor quanto maior o grau de deformação e quanto menor a temperatura de recozimento.
5. Quanto maior o tamanho de grão original, maior a quantidade de deformação é requerida para temperatura e tempo de recristalização equivalentes.
6. A quantidade de trabalho a frio necessário para provocar um equivalente endurecimento por deformação aumenta com o aumento da temperatura de trabalho.
7. O aquecimento continuado após a recristalização estar completa causa o aumento do tamanho dos grãos.

## 2.2 Simulação Computacional de Transformações de Fase

Até a metade do século passado a pesquisa científica se dividia em dois paradigmas principais: o experimental e o teórico. O modelo experimental consiste em reproduzir em um ambiente controlado algum fenômeno natural. Através disso, os cientistas podem quantificar o fenômeno e provar ou rejeitar hipóteses formuladas. As observações realizadas podem confirmar ou dar origem a teorias que constituem a mais avançada forma de organização do conhecimento científico [21].

Com o surgimento e popularização dos computadores, a simulação ou modelamento dos fenômenos naturais permitiu a reprodução do comportamento de inúmeros sistemas por meios computacionais, constituindo assim um terceiro paradigma de pesquisa. Embora tanto o paradigma experimental como o simulacional consistam em reproduzir algum aspecto natural, em um experimento a Natureza conduz e controla todos os acontecimentos, cabendo ao cientista apenas a preparação das condições iniciais, o controle do meio e a análise dos resultados, enquanto que na simulação, o fenômeno é reproduzido de uma forma artificial, cabendo ao cientista modelar computacionalmente não só o objeto da pesquisa como também todas as leis naturais requeridas [21].

Modelamentos computacionais são um meio mais barato e rápido de testar hipóteses, podem ser executados ainda que os meios para realização de experimentos reais não sejam disponíveis por questões financeiras ou tecnológicas, geralmente permitem a predição ou o alcance de resultados de forma mais rápida e possibilitam uma análise de dados mais vasta [21].

O estudo teórico analítico da cinética de transformações de fase pode se tornar demasiadamente complexo devido às várias hipóteses que precisam ser consideradas. Além disso, certas aproximações podem comprometer a exatidão do resultado e sua aplicação prática. Dessa forma, simulações computacionais são uma alternativa de grande valor, já que podem confirmar teorias e testar hipóteses de forma muito além da abordagem analítica e experimental.

Assim como acontece na Natureza, a simulação de uma transformação de fase é dividida em etapas de nucleação e crescimento. Primeiramente, realiza-se a discretização do domínio onde ocorrerá a transformação e, em seguida, determinam-se a quantidade e as posições dos núcleos segundo a hipótese em estudo. A nucleação pode ser totalmente concluída antes do crescimento (saturação de sítios) ou pode continuar durante ele. Já no estágio de crescimento, são utilizados métodos iterativos para produzir o desenvolvimento

dos núcleos em função do tempo. Essas etapas podem ser representadas em um fluxograma conforme o da Fig. 2.3. Nota-se que caso haja nucleação durante o crescimento, essas etapas serão executadas alternando-se simultaneamente até que toda transformação se conclua.



Figura 2.3: Fluxograma de uma simulação de nucleação e crescimento.

As simulações computacionais das transformações de fase têm atingido boa concordância com os modelos analíticos e com as observações experimentais tornando-se uma importante ferramenta de pesquisa [10, 15, 27].

# Capítulo 3

## Métodos de Crescimento

### 3.1 O método Autômato Celular

Um dos métodos usados para a simulação de transformações de fase é o Autômato Celular (AC). Um AC consiste de uma malha de células que possuem estados que podem variar no tempo. Trata-se de um modelo geral de computação com tempo, espaço e estado discretos [21]. Esse método é caracterizado pela geometria das células, pelo número e tipo de estados que as células podem possuir, pelo tipo de vizinhança entre as células e pelas regras de transição de estados em função do tempo e dos vizinhos [15]. A cada iteração toda malha é atualizada de acordo com as regras de transição.

Naumov [20] apresenta de modo mais formal a definição do AC: um autômato celular  $A$  é um conjunto de 4 objetos  $\{M, Z, V, f\}$ , onde:

- $M$ : a *malha*, um espaço métrico finito de células. O termo “estado da malha” designa o conjunto do estado de todas as células.
- $Z$ : o *conjunto finito dos possíveis estados das células*. Geralmente caracterizados por valores numéricos. É assumido como definições sinônimas “o estado da célula é  $s$ ” e “a célula contém/armazena o valor  $s$ ” [21].
- $V$ : a *descrição da vizinhança da célula*. Para cada célula, sua vizinhança é definida como o conjunto de células que a influenciam. Geralmente é descrita como uma faixa de distância, sendo na maioria dos casos um subconjunto dos vizinhos mais próximos da célula. O conjunto exato dos reais vizinhos de uma célula  $c$ , que podem ser encontrados segundo a descrição de  $V$ , é denotado como  $V(c)$ .
- $f$ : a *função de transição*. Ela determina o estado subsequente de cada célula de

acordo com sua vizinhança. Essa função pode ser definida como um algoritmo computacional ao invés de serem usadas expressões matemáticas. Podem ainda ser referidas como regras ou leis de um AC.

Embora em algumas simulações como, por exemplo, fenômenos físicos, os estados de cada célula pudessem ser representados por um intervalo contínuo, na prática, o conjunto  $Z$  permanece sendo finito pois tais valores são representados computacionalmente por variáveis de ponto flutuante que constituem um conjunto finito de valores [20].

A malha  $M$  é um *array* de células, cada qual contendo um estado pertencente a  $Z$  [20]. Malhas podem ter uma ou múltiplas dimensões. Com relação ao formato das células, deve-se preferir aquelas que tenham formas simples que ao serem replicadas preencham todo espaço. Em duas dimensões, podem-se ter uma malha de triângulos, quadrados ou hexágonos. Já em três dimensões, são exemplos as malhas de cubos, de prismas triangulares e de prismas hexagonais.

A vizinhança deve ser do mesmo tipo para todas as células (excetuando-se as particularidades nas regiões de contorno). Portanto, o conjunto  $V(c)$  pode ser visto como uma coleção de deslocamentos a partir da célula atualmente considerada.

Ademais, há três propriedades básicas que designam um AC clássico [21]:

- *Localidade*: As regras de um AC devem ser locais e todos os membros da vizinhança devem estar a uma distância finita da célula.
- *Homogeneidade*: o sistema deve ser similar para todas as células.
- *Sincronicidade*: todas as células devem atualizar seus estados simultaneamente ao fim de cada iteração, após novos valores serem computados para cada uma delas. Ou seja, a ordem em que cada célula é calculada não pode influenciar no estado final da malha.

Em linhas gerais, o processamento do método é bastante simples. Em cada iteração, o estado de cada célula é calculado com base no estado das células vizinhas na iteração anterior. Em alguns casos, pode-se usar um sorteio para definir qual vizinho definirá o estado da célula.

Com relação à simulações computacionais, primeiramente modela-se o próprio AC, com seus estados, malha, vizinhança e regras de transição. Em seguida, modela-se o fenômeno equiparando suas características com a do AC.

Simulações em grande escala resultam em enorme carga computacional. O processo de determinação dos vizinhos das células e a atualização dos estados de cada uma delas são tarefas relativamente simples, porém dado o grande número de células geralmente utilizado, completar todo este trabalho torna-se bastante custoso. Pode-se considerar que o tempo de processamento aumenta linearmente com o aumento do número de células da matriz. No entanto, como o cálculo de cada célula só depende do estado de sua vizinhança na iteração anterior, o método AC pode ser paralelizado o que proporciona excelentes ganhos de desempenho.

No tocante ao problema em estudo, o método AC, ao contrário da abordagem de JMAK, proporciona uma descrição da transformação dependente do tempo e do espaço. Ele não é restrito a premissas estatísticas de valores médios nem a processos homogêneos. Geralmente modelos matemáticos dependentes do tempo e do espaço são descritos por equações diferenciais parciais que, em muitos casos, precisam ser resolvidas por métodos numéricos. O AC não necessita de nenhuma adaptação para um algoritmo numérico, o que faz dele um método simples e eficaz para a descrição de fenômenos físicos [15].

Na Fig. 3.1 pode ser visto um esquema das vizinhanças de Von Neumann e condições de contorno periódicas usadas neste trabalho na simulação de AC. A continuidade de uma face da malha é seu lado oposto. Maiores informações sobre o funcionamento e características do método são dadas na descrição de sua implementação no Cap. 5.

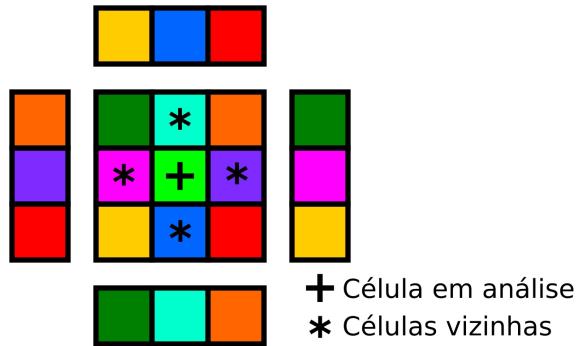


Figura 3.1: Esquema de vizinhanças e condições de contorno usadas neste trabalho. Os retângulos destacados simbolizam a vizinhança devido às condições de contorno periódicas (Adaptado [2]).

## 3.2 O método Cone Causal

Outro interessante método para a descrição de transformações de fase é o Cone Causal (CC) [10]. A ideia de uma ordem causal é um conceito básico da Física e é intuitivamente

percebido na Natureza. Uma vez que um evento acontece em um dado ponto do espaço-tempo, haverá um raio de influência possível máximo sobre as regiões do espaço ao redor que poderá aumentar com o passar do tempo. Analogamente, um dado ponto no espaço-tempo só sofrerá a influência de eventos que ocorram dentro de um determinado raio que se expande com o tempo. Pode-se usar a analogia de uma gota caindo em um lago: o efeito do choque da gota na superfície do lago não afetará todo lago instantaneamente, mas uma onda se propagará causando uma perturbação na superfície que dependerá da distância com relação ao ponto onde a gota caiu e do tempo desde que tal evento ocorreu.

No caso da simulação de transformações de fase, pode-se tomar que a partir do momento que um núcleo surge em uma célula da malha, outras células serão afetadas por este núcleo com o decorrer do tempo. Considerando que os grãos da nova fase crescem em formato esférico com velocidade radial  $V = V(t)$ , o raio do grão no tempo  $t$  cujo núcleo surgiu no tempo  $t_0$  na posição  $\mathbf{x}'$  será dado por

$$R(\mathbf{x}', t_0) = \int_{t_0}^t V(t') dt'. \quad (3.1)$$

Se o comprimento do raio  $R$  superar a distância entre  $\mathbf{x}$  (um ponto qualquer da malha) e  $\mathbf{x}'$  (um núcleo),

$$R(\mathbf{x}', t_0)^2 - |\mathbf{x} - \mathbf{x}'|^2 \geq 0, \quad (3.2)$$

o ponto  $\mathbf{x}$  terá se transformado no tempo  $t$ . O cone causal será o conjunto de todos os pontos que satisfazem essa desigualdade. Sendo assim, trata-se da região do espaço-tempo em que ao menos um evento de nucleação aconteceu tal que o ponto  $x$  será transformado no tempo  $t$  [19]. A Fig. 3.2 dá uma representação gráfica para o cone causal sendo ignorada uma dimensão espacial.

A superfície do cone causal, i.e., a esfera de raio  $R$  no tempo  $t$ , é formada pelos pontos  $(\mathbf{x}, t)$  do grão correspondente ao núcleo  $(\mathbf{x}', t_0)$  que se transformam no tempo  $t$ , satisfazendo a igualdade da Eq. 3.2. Sendo assim, é possível criar um método de crescimento geométrico baseado nesta teoria. O método CC aplicado a um domínio discreto realizará a transformação do estado de um ponto sempre que a Eq. 3.2 for satisfeita com relação ao núcleo mais próximo. É importante destacar que, para evitar sobreposição de grãos, um ponto da malha só sofrerá a influência de um único núcleo que esteja mais próximo.

Sendo um modelo de malhas e estados semelhante ao do AC (Cap. 3.1), o CC con-

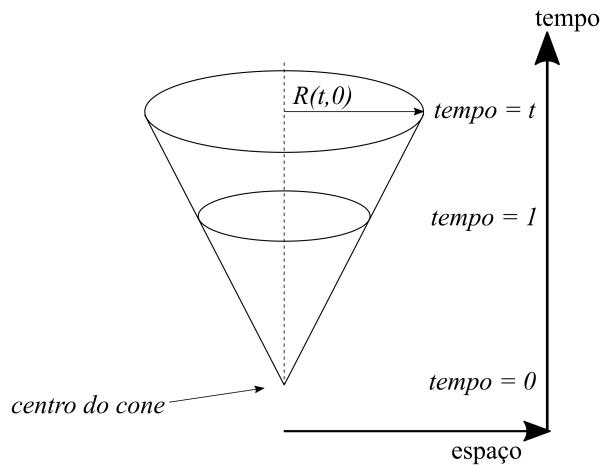


Figura 3.2: Representação gráfica do cone causal (Adaptada [11]).

sidera que o espécime parte do estado não transformado no tempo 0. Uma vez que os núcleos sejam inicializados nas suas posições de acordo com cada método de Nucleação, cones de transformação crescerão a partir de cada ponto da malha com uma taxa pré-determinada a cada iteração. Quando esta região em crescimento atinge algum núcleo, o ponto central do cone passa a ter o mesmo estado do núcleo alcançado e não mais transforma. O CC é um modelo predominantemente determinístico, ou seja, para o mesmo conjunto de núcleos, o método CC resultará sempre na mesma microestrutura final, salvo nas regiões equidistantes a dois ou mais núcleos onde um sorteio decidirá qual núcleo causará a transformação daquela célula.

Uma outra maneira de entender como ocorre a transformação é imaginar um sistema tridimensional onde os eixos  $x$  e  $y$  representam a dimensão espacial e o eixo  $z$  representa o tempo (semelhante ao proposto pela Fig. 3.2). Sendo assim, a região transformada é representada por um círculo que se expande com o passar do tempo. A região do espécime pode ser delimitada por algum retângulo perpendicular ao eixo  $z$ . Se várias dessas regiões transformadas estiverem distribuídas lado a lado, pode-se intuir que a partir de algum plano  $t = t_i$  as primeiras regiões irão se tocar e, no plano  $t = t_f$  em que não houver pontos da região do espécime que não sejam correspondentes a algum dos núcleos, os segmentos de reta resultantes das interseções dos círculos representarão (de forma aproximada) os contornos de grão de uma microestrutura convencional de metal recristalizado. Em duas dimensões, esses segmentos são arestas de uma rede de Voronoi, ou, em três dimensões, as interseções das esferas resultaram nas faces de poliedros de Voronoi [22]. Um esquema dessa evolução é apresentado na Fig. 3.3.

Vale ressaltar que os cones são centrados em cada célula da malha, isto é, existe um

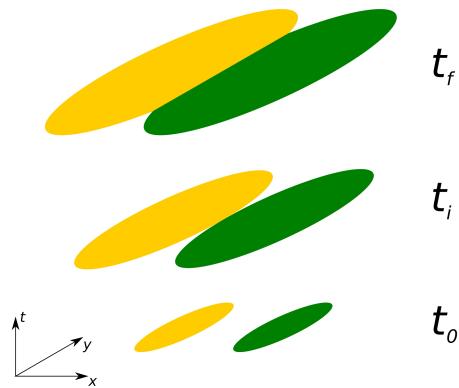


Figura 3.3: Esquema da evolução e interação de grãos pelo método CC.

cone causal para cada ponto da malha (exceto os que representem núcleos ou regiões já transformadas). Esses cones se expandem resultando na transformação da célula quando seu cone atinge algum núcleo. No entanto, o efeito observado leva a supor que os cones crescem a partir dos núcleos. Na verdade, como os cones das células mais próximas de algum núcleo o atingirão primeiro, seguidos dos cones das células um pouco mais afastadas e assim por diante, o que se observa é uma região transformada circular (ou esférica) crescendo a partir de um núcleo a cada iteração.

# Capítulo 4

## A programação Orientada a Objetos

A programação orientada a objetos (POO) é uma forma de criar uma arquitetura de software que possibilita flexibilidade por meio de um design modular [31].

Zimmermann *et al.* [37] descrevem a POO como um paradigma para a construção de programas que permite melhorias na velocidade de desenvolvimento, na facilitação da manutenção, na confiabilidade e na reusabilidade do código. Segundo estes autores, objetos são dispositivos capazes de realizar funções pré-definidas como armazenar informações, executar uma tarefa ou dar acesso a outro objeto. Portanto, objetos são definidos pelos seus atributos (dados que armazena, equivalentes a constantes ou a variáveis) e pelos seus métodos (ações que executa, equivalentes a funções). Os métodos permitem que as classes se comuniquem e trabalhem em conjunto. Ou seja, um objeto consiste de dados que são fortemente relacionados a operações relativas a ele [32]. Essas operações referem-se aos métodos e a comunicação entre os objetos é chamada de *mensagem*.

Um objeto é criado a partir de uma *classe*. Classes são tipos abstratos de dados que funcionam na construção de um objeto tal como a planta de uma casa atua em sua edificação [13], ou seja, elas definem como será o objeto, quais são os atributos e os métodos que cada tipo de objeto terá (embora não necessariamente atribuam valores aos atributos).

Dois outros recursos da POO que propiciam a reutilização do código são a *herança* e o *polimorfismo*. A herança permite que sejam criadas classes com base nas já existentes, reaproveitando atributos e métodos [13]. Classes abaixo na hierarquia podem herdar métodos e atributos das classes superiores [32]. Já o polimorfismo permite que a mesma mensagem ative diferentes métodos quando endereçadas a objetos de classes diferentes relacionadas por herança [13, 37].

Finalmente, objetos têm seus dados encapsulados, isto é, somente o próprio objeto pode manipular seus atributos. O acesso ou modificação dos dados de um objeto deve ser feito mediante a ativação de um de seus métodos [37].

Goldberg e Robson [14] apresentam três definições da POO:

- Uma *visão*, ou um modo de organizar uma descrição do sistema. A ideia central na visão orientada a objetos é que o sistema pode ser construído por conjunto de objetos, cada um com atributos e comportamentos (métodos).
- Um *conjunto de técnicas de programação* que incluem a manipulação de objetos, atributos e comportamentos.
- Um *grande e complexo sistema* que permite a criação de aplicações usando objetos com técnicas para manipulação de objetos, atributos e métodos.

A POO não é um paradigma recente, já havendo trabalhos sobre conceitos primitivos deste tema pelo menos desde os anos 1950 [32]. O paradigma anterior à POO e ainda muito frequentemente utilizado no desenvolvimento de softwares acadêmicos é a Programação Procedural (PP). A PP é um método de desenvolvimento linear que geralmente produz uma série de rotinas e sub-rotinas. É um método que pode funcionar bem caso haja somente um desenvolvedor pois este se tornará familiarizado com o código. Todavia, quando vários desenvolvedores trabalham no mesmo projeto, pode se tornar bastante difícil entender o código e realizar alterações em todos os trechos interligados sem provocar danos [31].

Já na POO, cada comportamento do programa está contido em uma única classe, diminuindo o risco de interferências inesperadas em outras partes do código. Isso também possibilita que desenvolvedores implementem alterações ou reusem o código muitas vezes sem a necessidade de conhecer e entender toda a aplicação.

Uma das primeiras aplicações dos conceitos da POO foi no desenvolvimento do míssil balístico intercontinental nuclear norte-americano Minuteman no final dos anos 1950 [32]. Os objetos descreviam aspectos físicos do míssil, como motores, tanque de combustível e tubeiras, e elementos abstratos como o controle da trajetória de voo. Além disso, os objetos podiam atualizar e armazenar os dados de seus próprios atributos. Cada componente foi criado por técnicos que eram especialistas apenas em sua própria parte do projeto e determinavam para os demais projetistas que informações necessitariam para sua porção do desenvolvimento. Os dados de cada componente eram encapsulados de modo

que só poderiam ser acessados por meio de métodos pré-estabelecidos. Os componentes podiam ser testados separadamente via simulações. Esse exemplo, ainda que não tenha utilizado linguagens propriamente orientadas a objetos, ilustra bem como funciona um projeto de POO.

Assim como no projeto do míssil, a POO aplicada à simulações de Transformações de Fase pode ter representações de entidades físicas (grãos, malha, microestrutura, etc.) e abstratos (representações da nucleação e crescimento das fases, realizações de cálculos estequiométricos, etc.). Também há o uso de objetos para partes secundárias como leitura e escrita de arquivos e armazenamento de dados durante a simulação.

A POO pode ser implementada em diferentes graus de completude e em várias linguagens de programação.

# Capítulo 5

## Desenvolvimento do Código

As classes usadas neste trabalho foram escolhidas de acordo com as tarefas necessárias para a simulação de transformações de fase. Esta seção apresenta o detalhamento do problema e as classes criadas para sua solução. O programa foi escrito na linguagem C++. Algumas estruturas de repetição das classes Matriz, Autômato e Cone foram paralelizadas com uso da biblioteca OpenMP [12].

O código foi predominantemente desenvolvido com o software Microsoft Visual Studio Community 2017<sup>1</sup> (compilador Visual C++ 2017) e está disponível para download<sup>2</sup>. O computador utilizado para execução foi um Intel Core i5 7th Gen com 8GB de memória DDR4 e com sistema operacional Microsoft Windows 10.

### 5.1 Descrição do problema

As transformações de fase são divididas em processos de nucleação e crescimento. No entanto, em termos computacionais, é preciso que se definam os parâmetros da transformação (tamanho do domínio, número e posição dos núcleos, se o crescimento vai ser realizado por AC ou CC, etc.) e então se construa o espaço discreto, uma malha onde a transformação ocorrerá. Uma vez que o espaço esteja construído, ocorrerão a nucleação — a marcação das posições dos núcleos no espaço — e o crescimento — a marcação das redondezas dos núcleos conforme o método de crescimento utilizado. Finalmente, devem ser feitos cálculos e gravação de resultados para se avaliar como ocorreu a transformação.

A Fig. 5.1 apresenta o fluxograma do processo.

---

<sup>1</sup><https://visualstudio.microsoft.com/>

<sup>2</sup><https://github.com/jlgf7/TransFase>

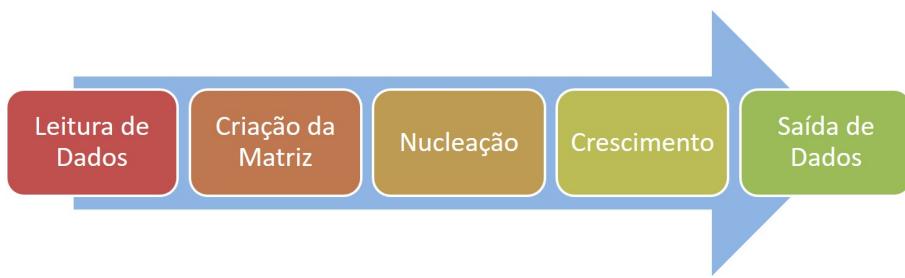


Figura 5.1: Fluxograma do processo.

Cada uma das etapas representadas pelo fluxograma (Fig. 5.1) foram implementadas como classes seguindo o paradigma da POO.

## 5.2 Descrição das classes

### 5.2.1 Classe Dados

A classe *Dados* é responsável pela coleta e armazenamento dos parâmetros que determinarão as tarefas que o programa realizará. Tais parâmetros são obtidos pela leitura de um arquivo de texto padronizado contendo os dados relacionados ao tipo e tamanho da malha, tipo de nucleação, número de núcleos, tipo de vizinhança, critérios de parada e informações para o registro dos resultados. Um objeto armazena esses dados e os torna acessíveis a qualquer momento sem necessidade de releitura do arquivo fonte.

### 5.2.2 Classe Matriz

Esta classe cria o domínio onde serão realizadas as transformações. Neste trabalho, foi implementada uma malha tridimensional cúbica cuja menor unidade é a célula geometricamente semelhante a um cubo. Cada um destes cubos equivale a uma unidade de volume para a simulação e a união ordenada dessas células forma a matriz que representa o espaço discreto. As células armazenam valores inteiros para identificar o estado da transformação, sendo zero o valor para o estado não transformado.

Além disso, um objeto é criado contendo todas as variáveis relativas à malha: a matriz de estados, variáveis para armazenamento dos dados estereológicos e variáveis necessárias para os métodos de crescimento. Esta classe também possui métodos para realização de medições estereológicas que são realizadas durante o crescimento: fração volumétrica ( $V_V$ ), fração de área por unidade de volume entre a interface transformada e a não transformada ( $S_{V\alpha\beta}$ ), fração de área interfacial por unidade de volume entre grãos transformados ( $S_{V\beta\beta}$ )

e caminho microestrutural.

O cálculo de  $V_V$  na simulação (correspondente à Eq. 2.1) é feito com um método que conta as células com estado diferente de zero e divide pelo total de células da matriz. Com relação ao cálculo de  $(S_{V\alpha\beta})$  (correspondente à Eq. 2.3) é usado um método que percorre a malha seguindo todas as retas possíveis na direção do eixo  $x$  com sentido positivo. Sempre que se identifica que a célula atual tem estado 0 e a anterior não (ou vice-versa), há uma interface entre a fase  $\alpha$  (igual a zero) e  $\beta$  (diferente de zero) e é feita uma adição em um contador. Repete-se o processo nas direções dos eixos  $y$  e  $z$  e deve-se considerar as condições de contorno periódicas. A  $S_{V\alpha\beta}$  é obtida pela divisão do total de interfaces identificadas pelo total de células da malha. Já a  $S_{V\beta\beta}$  é calculada de forma análoga à  $S_{V\alpha\beta}$ , no entanto são identificadas as interfaces entre os grãos, ou seja, entre células com estados diferentes e as interfaces com células com estado zero são desprezadas.

A velocidade média dos contornos de grão  $\langle G \rangle$  é calculada para cada simulação com uso da Eq. 2.2 que aplicada ao caso discreto é escrita como:

$$G[i+1] = 2 * \frac{V_V[i+1] - V_V[i]}{S_{V\alpha\beta}[i+1] + S_{V\alpha\beta}[i]}. \quad (5.1)$$

No entanto, o cálculo de  $\langle G \rangle$  não é realizado diretamente no programa, podendo ser feito em qualquer software que processe os dados de  $V_V$  e  $S_{V\alpha\beta}$  exportados.

### 5.2.3 Classe Nucleação

A classe *Nucleação* chama os métodos necessários para a modificação do objeto da classe *Matriz* conforme o tipo de nucleação e o número de núcleos armazenados no objeto da classe *Dados*. Os tipos de nucleação atualmente implementados são: Uniforme e Periódica. Em ambas, todos os núcleos surgem em um tempo arbitrário  $t = 1$ .

A nucleação Uniforme utiliza a biblioteca *random* de C++ para a geração de números pseudoaleatórios com uma distribuição uniforme [33]. Desta forma, as posições na matriz são sorteadas e cada célula distinta escolhida recebe um valor sequencial de 1 a  $n$ , onde  $n$  é o número total de núcleos previamente determinado na entrada de dados.

Já a nucleação Periódica divide os núcleos em posições equidistantes, de acordo com o número de núcleos determinado. Os valores de cada núcleo também seguem um valor sequencial de 1 a  $n$ .

Após concluída a nucleação, os primeiros cálculos estereológicos são realizados por

métodos da classe *Matriz*.

Também foi incluído um método para salvar a posição dos núcleos e para importar a posição de núcleos gerados por este programa ou por similares.

#### 5.2.4 Classe Autômato

Esta é a classe responsável pela execução do método AC, criada para realizar o crescimento a partir dos núcleos previamente formados pela classe *Nucleação*. O AC utilizado neste trabalho possui uma malha tridimensional de células cúbicas, já construída na classe *Matriz*, com vizinhança de Von Neumann. Esse tipo de vizinhança é definido pelas células imediatamente ao lado da célula em análise nas direções axiais (Fig. 5.2). A célula pode ter um dos dois estados *transformado* (representado pelos valores de 1 a  $n$ ) ou *não transformado* (representado pelo valor 0). Além disso, são assumidas condições de contorno periódicas, isto é, as células localizadas nos planos das extremidades da malha têm como vizinhas as células do plano da extremidade oposta.

O método AC necessita que duas dessas matrizes sejam construídas: a primeira para armazenar os dados antes do começo do passe de transformação e a segunda para armazenar os dados durante o passe. Antes da iteração seguinte, a primeira matriz recebe os valores da segunda e o processo continua.

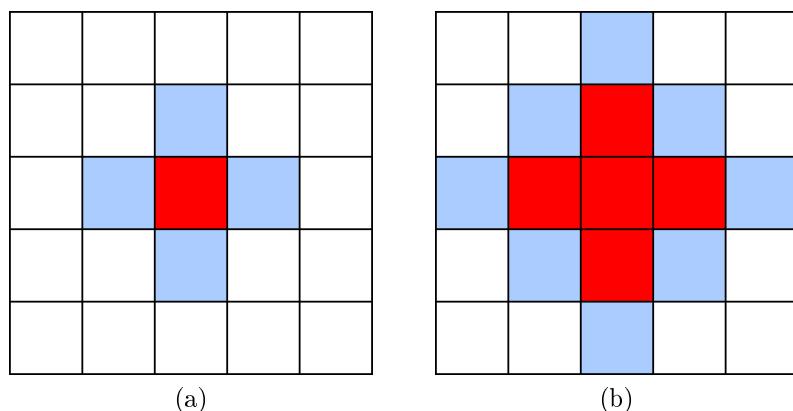


Figura 5.2: Esquema de crescimento 2D de AC com vizinhança de Von Neumann: (a) → (b). A cor vermelha representa uma célula já transformada e a azul, uma célula que será transformada na iteração atual.

Durante cada iteração, o algoritmo testa as células ainda não transformadas da malha e, se alguma de suas vizinhas estiver no estado transformado, a célula sob análise recebe o valor correspondente ao estado dessa vizinha. Casos em que há mais de uma vizinha transformada são facilmente resolvidos uma vez que é sempre escolhido o valor da primeira

vizinha transformada encontrada e a ordem de verificação das vizinhas é aleatória. O novo valor da célula é armazenado em uma matriz auxiliar e não interferirá durante aquela iteração na transformação das demais células até que uma nova iteração comece com os valores atualizados.

### 5.2.5 Classe Cone

Se o método CC for determinado na entrada de dados, será esta a classe que realizará o crescimento. O ponto crítico deste método é a determinação das distâncias  $d_i$  entre uma determinada célula e os núcleos. Essa célula será transformada quando a Eq. 3.2 for satisfeita. Embora em um espaço convencional esse cálculo seja simples, o uso de condições de contorno periódicas o torna bem mais complexo, pois é preciso considerar que um cone que cresce próximo aos limites da malha continuará crescendo nos lados opostos como esquematizado pela Fig. 5.3.

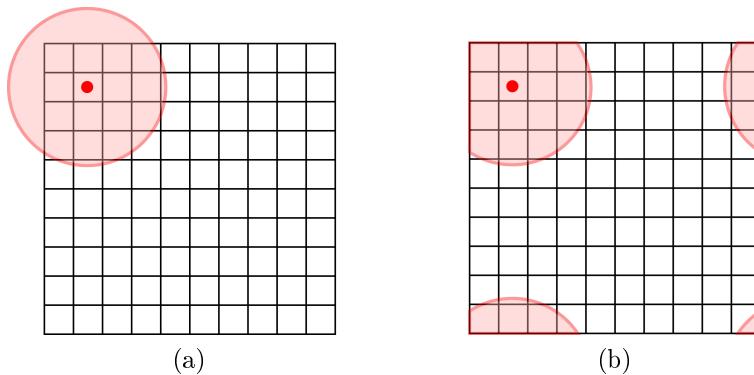


Figura 5.3: Esquema do efeito das condições de contorno periódicas na área de influência de um cone causal em um determinado tempo. (a) Área convencional e (b) efeito das condições de contorno.

A cada iteração o cone cresce e novas células são transformadas recebendo o valor correspondente ao núcleo alcançado pelo cone. Como, uma vez transformada, uma célula não poderá receber o estado de um outro núcleo, ou seja, seu cone parará de crescer, interfaces surgirão entre as regiões transformadas e o processo continuará até que todas as células sejam modificadas ou que seja determinada uma parada arbitrária. Se uma célula estiver equidistante a dois ou mais núcleos, um sorteio decidirá qual núcleo será o responsável por sua transformação.

### 5.2.6 Classe Saída

Sempre que for preciso extrair algum resultado de um cálculo ou mesmo que se queira exportar os valores da malha, um método da classe *Saída* será usado. Esta classe lê o objeto da classe *Matriz* e prepara seus dados para serem salvos em formatos compatíveis para o uso em programas gráficos ou estatísticos. Os dados podem ser salvos durante ou após as transformações. Foi implementada a escrita dos arquivos .dat (com a formatação usada pelo programa Tecplot e com a usada para leitura de scripts em R C) e .vtk (usado pelo programa ParaView B).

# Capítulo 6

## Resultados e Discussão

A observação da microestrutura dos materiais é uma das mais importantes análises para a compreensão das causas de diversas propriedades macroscópicas e da fenomenologia que resultou em tais características. Sendo assim, uma das funções principais deste programa é gerar uma microestrutura compatível com o que é observado em laboratório tendo a vantagem de se poder testar hipóteses e inferir com base nos resultados o que acontece nos materiais reais.

Foram feitas simulações para os dois tipos de nucleação apresentados, uniforme e periódica, e para os dois métodos de crescimento, AC e CC. Todas as simulações foram realizadas em malhas cúbicas de dimensão  $320 \times 320 \times 320$  e com 64 núcleos. Optou-se por manter a mesma dimensão da matriz e o mesmo número de núcleos para todas as simulações. A escolha desses valores não é totalmente arbitrária, mas atende a particularidades da nucleação periódica que, por questões de simetria, apresenta melhores resultados para matrizes cujos números de linhas, colunas e cotas são iguais, cujo número de núcleos é um número cúbico e cuja dimensão do lado da matriz é um múltiplo do número de núcleos.

A Fig. 6.1 traz o desenvolvimento da microestrutura para nucleação uniforme e crescimento por AC. Estas e as demais imagens das microestruturas foram produzidas pelo programa ParaView [1] a partir dos arquivos de dados gerados nas simulações.

Pode-se perceber pela Fig. 6.1 o formato octaédrico dos grãos (inerente à vizinhança definida para o método AC) nas etapas iniciais do crescimento. Tal geometria é compatível com o esquema de crescimento em 2D do AC visto na Fig. 5.2. No entanto, à medida que os grãos vão crescendo e começam a tocar uns nos outros (*impingement*) até que os últimos espaços vazios sejam preenchidos, a forma resultante torna-se semelhante ao que é observado em microestruturas de metais recristalizados, o que será discutido mais à

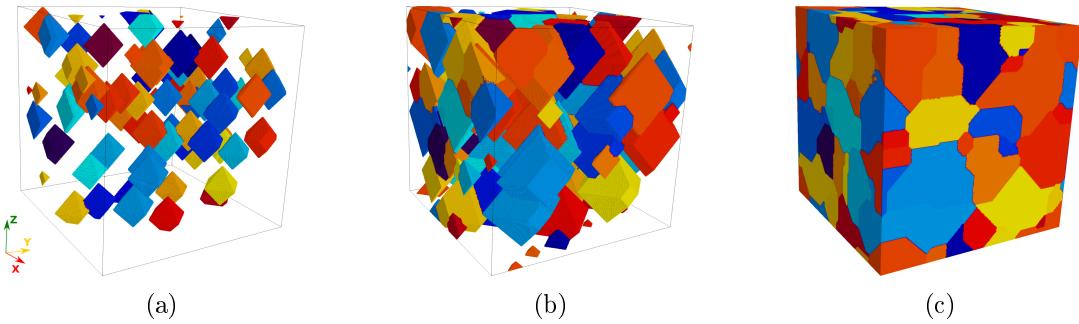


Figura 6.1: Crescimento por Autômato Celular a partir de nucleação uniforme. Transformação de (a) 10% , (b) 50% e (c) 100%.

frente. Pode-se observar ainda, o efeito das condições de contorno periódicas nos grãos que crescem próximo às fronteiras: vários octaedros têm parte de seu volume continuando a crescer na extremidade oposta do espaço da transformação.

Semelhantemente, a Fig. 6.2 apresenta o desenvolvimento da microestrutura para nucleação uniforme e crescimento por CC. Optou-se por usar as mesmas posições dos núcleos que foram sorteadas para a simulação com AC a fim de melhor comparar os dois métodos de crescimento. Ao invés do formato octaédrico típico do AC, o CC promove um crescimento com grãos em formato esférico. Novamente, é visto o efeito das condições de contorno periódicas, esquematizado na Fig. 5.3, nas esferas que crescem nas fronteiras e que parecem ter sofrido um corte cuja seção complementar surge na extremidade oposta do espaço. Além disso, os grãos da microestrutura final do CC são todos convexos, o que é característico dos poliedros de Voronoi cujo volume é uma região do espaço em que cada ponto está mais próximo de um determinado núcleo do que de qualquer outro [30].

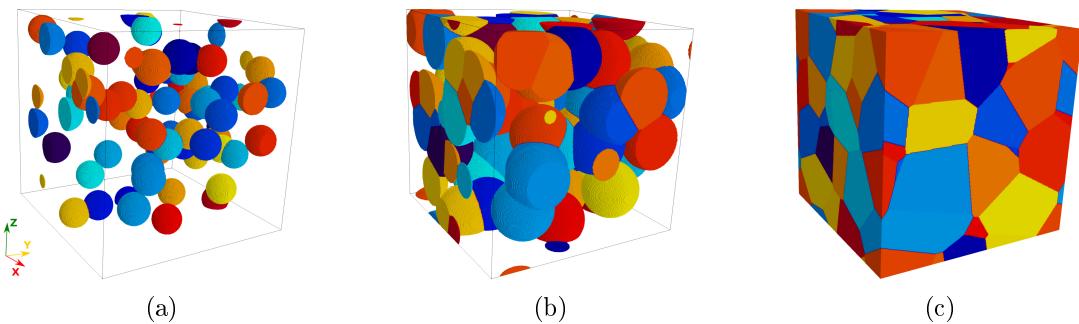


Figura 6.2: Crescimento por Cone Causal a partir de nucleação uniforme. Transformação de (a) 10% , (b) 50% e (c) 100%.

Já a Fig. 6.3 traz os resultados da simulação de crescimento por AC de uma nucleação periódica. A organização equidistante dos núcleos faz com que o crescimento comece produzindo uma forma octaédrica que, após o *impingement* dos grãos, vai se transformando

em uma forma cúbica. A condição de contorno periódica é percebida pelo aspecto pontilhado da face voltada para a direção  $-y$ . No entanto, pode ser visto que as faces voltadas para as direções  $x$  e  $z$  apresentam aspecto liso. Isso se deve à relação entre a dimensão dos lados da matriz e o número de núcleos em cada direção. Por exemplo, nesta simulação pode-se tomar a fileira de núcleos  $N_1 : (280, 40, 0)$ ,  $N_2 : (280, 120, 0)$ ,  $N_3 : (280, 200, 0)$  e  $N_4 : (280, 280, 0)$ . Se for traçada uma reta que ligue esses núcleos, os pontos dessa reta que farão parte das faces da matriz serão  $P_1 : (280, 0, 0)$  e  $P_2 : (280, 319, 0)$ . Logo, se tratando de um espaço discreto e com condições de contorno periódicas, o ponto  $P_1$  é equidistante a  $N_1$  e a  $N_4$  (ou seja, receberá aleatoriamente o valor do estado de  $N_1$  ou de  $N_4$ ), enquanto que o ponto  $P_4$  é mais próximo de  $N_4$  que de  $N_1$  (portanto receberá o valor de  $N_4$ ). Isso acontece de forma análoga a todos os outros pontos contidos nas faces com relação aos demais núcleos e implica no aspecto uniforme de determinadas faces (todos os pontos só possuem cada um um único núcleo mais próximo) e no disforme de outras (pontos equidistantes cada um a dois núcleos).

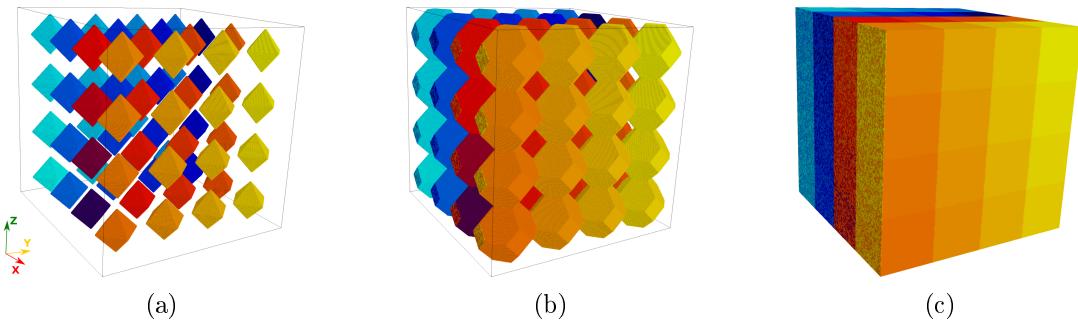


Figura 6.3: Crescimento por Autômato Celular a partir de nucleação periódica. Transformação de (a) 10% , (b) 50% e (c) 100%.

Outrossim, a Fig. 6.4 apresenta a evolução da microestrutura resultante da simulação de crescimento por CC a partir de nucleação periódica. A microestrutura final é muito semelhante à vista na Fig. 6.3 sendo a razão do aspecto pontilhado de algumas faces e as comparações com a simulação por AC análogas ao já exposto nos outros casos.

Embora haja alguma semelhança entre as microestruturas finais do AC e CC, percebe-se contornos dos grãos mais lineares no CC que os vistos no crescimento por AC. A Fig. 6.5 permite a visualização de uma das faces da malha gerada em ambos os métodos. É possível perceber como os contornos de grão da superfície gerada pelo método AC são mais irregulares com regiões contendo pontilhados de uma cor dentro da outra. Isso deve-se às muitas irregularidades na superfície final do grão.

Como esta imagem é de uma região de fronteira, ela traz na verdade o corte em 2D

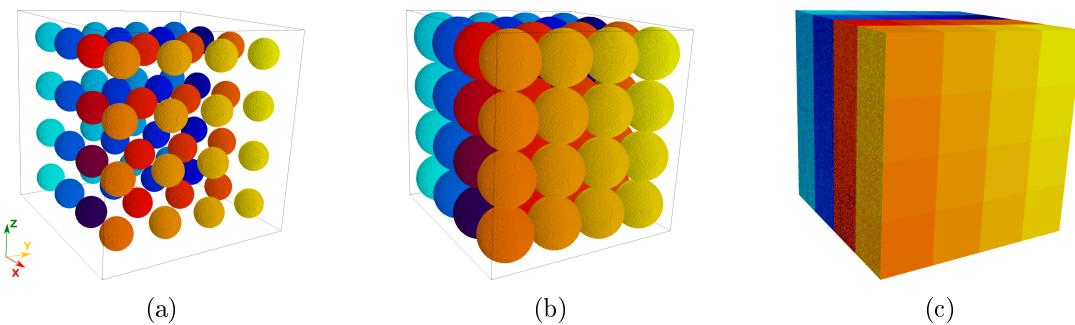


Figura 6.4: Crescimento por Cone Causal a partir de nucleação periódica. Transformação de (a) 10% , (b) 50% e (c) 100%.

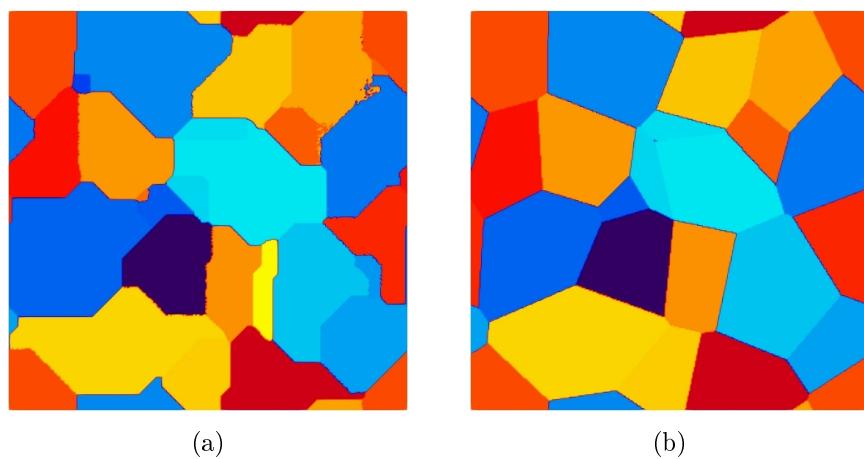


Figura 6.5: Comparação de microestrutura gerada pelo método (a) AC e (b) CC. Vista da face que aponta para o sentido positivo do eixo z.

de grãos que cresceram para além do limite da malha (condição de contorno periódica) e portanto, há rugosidades de um grão não convexo encrustadas no grão vizinho conforme esquema da Fig. 6.6.

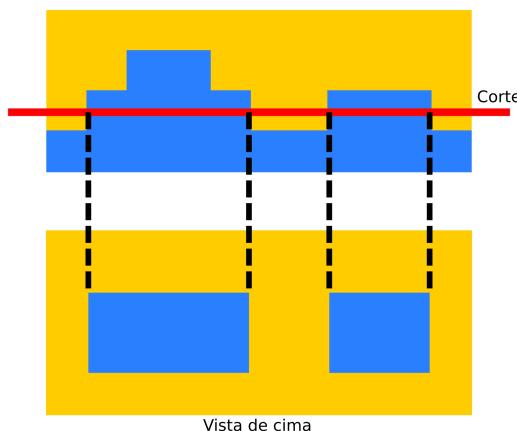


Figura 6.6: Esquema explicativo dos pontos que surgem em áreas de outras cores na microestrutura de AC.

Quando comparadas a microestruturas observadas na Natureza, como o exemplo na Fig. 6.7 com relação à fase austenita do aço, observa-se bom acordo. Em todos os casos apresentados nessa figura são vistos grãos aproximadamente equiaxiais e de tamanhos variados. No entanto, os contornos lineares encontrados na simulação por CC (Fig. 6.7b) são mais semelhantes ao que é visto no modelo de grão austenítico (Fig. 6.7c) e em uma micrografia real (Fig 6.7d). No caso desta última, grãos exageradamente grandes podem ser na verdade conjuntos de grãos que não tiveram seus contornos totalmente revelados sob ataque químico.

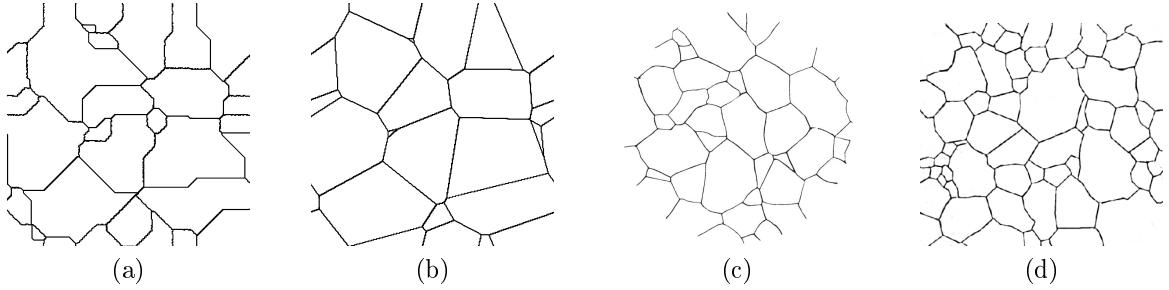


Figura 6.7: Comparação das microestruturas obtidas por simulação de (a) AC e (b) CC com (c) um modelo de microestrutura de austenita (adaptada [3]) e com (d) uma micrografia real de austenita recristalizada (adaptada [36]).

Além desses resultados onde é vista a microestrutura gerada na transformação, o programa gera dados para o estudos estereológicos. Todas as comparações a seguir são feitas para nucleação uniforme visto que as equações apresentadas no trabalho não se referem à situação de nucleação periódica. Além disso, alguns dos pontos relativos aos valores medidos nas simulações foram omitidos a fim de deixar a leitura dos gráficos mais clara. Os seguintes gráficos e cálculos necessários para criá-los foram feitos com uso da linguagem R [24]. Ressalta-se que os valores de tempo exibidos nos gráficos seguintes referem-se aos passos de iteração e, portanto, são adimensionais. Optou-se por manter a nomenclatura “tempo” ao invés de “passos” para facilitar a correspondência com a variável tempo encontrada nas equações referentes a cada gráfico.

A primeira comparação é feita relativa à fração volumétrica de região transformada em cada iteração do método de crescimento. A Fig. 6.8 traz a evolução da fração volumétrica medida na simulação e a comparação com a equação de JMAK (Eq. 2.1). Na literatura, encontra-se que o parâmetro  $\sigma$  dessa equação tem o valor  $4/3$  para o método AC [27] e  $4\pi/3$  para o método CC [35]. O tempo corresponde ao número de iterações efetuadas pelos métodos de crescimento. Como a equação de JMAK foi desenvolvida sob o pressuposto de núcleos aleatórios, ela não pode ser aplicada à nucleação periódica.

Como visto na Fig. 6.8, o crescimento por CC ocorre em menos tempo que o por

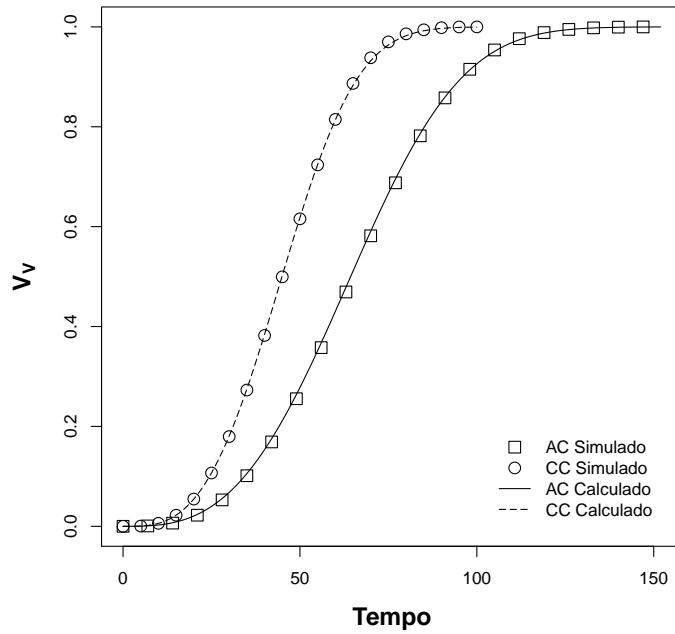


Figura 6.8: Evolução temporal de  $V_V$  das simulações pelos métodos AC e CC com nucleação uniforme comparadas ao calculado pela Eq. 2.1.

AC. De fato, o CC precisou de 101 iterações para concluir a transformação, enquanto o AC, 153. Percebe-se ainda boa correlação entre os valores calculados pela Eq. 2.1 e os medidos na simulação, com erro quadrático médio de  $2.3 \times 10^{-5}$  no AC e  $6.9 \times 10^{-6}$  no CC.

Para os valores de  $S_{V\alpha\beta}$ , a Fig. 6.9 traz a comparação entre os valores calculados pela Eq. 2.3 e os medidos na simulação da nucleação. O parâmetro  $\omega$  é igual a 3 no AC e  $3/2$  para o CC. A curva de  $S_{V\alpha\beta}$  é crescente enquanto a área superficial livre dos grãos é maior que a área superficial entre os grãos que se tocam. Porém, à medida que a transformação se completa e resta menos volume para ser transformado,  $S_{V\alpha\beta}$  decresce até zero. Já os menores valores no cálculo e na simulação de CC devem-se à menor relação entre área e volume em uma esfera (CC) que em um octaedro (AC).

Quanto ao caminho microestrutural, a Fig. 6.10, apresenta a comparação entre os valores calculados pela Eq. 2.4 a partir dos valores já calculados na Eq. 2.1 e Eq. 2.3 e a partir dos valores de  $V_V$  e  $S_{V\alpha\beta}$  medidos na simulação. Como nestes casos somente é feita a comparação para a nucleação uniforme, a diferença entre as curvas do AC e CC são apenas devido aos menores valores de  $S_{V\alpha\beta}$  no CC já comentados anteriormente.

Com relação à velocidade média de crescimento dos contornos de grão  $\langle G \rangle$ , apli-

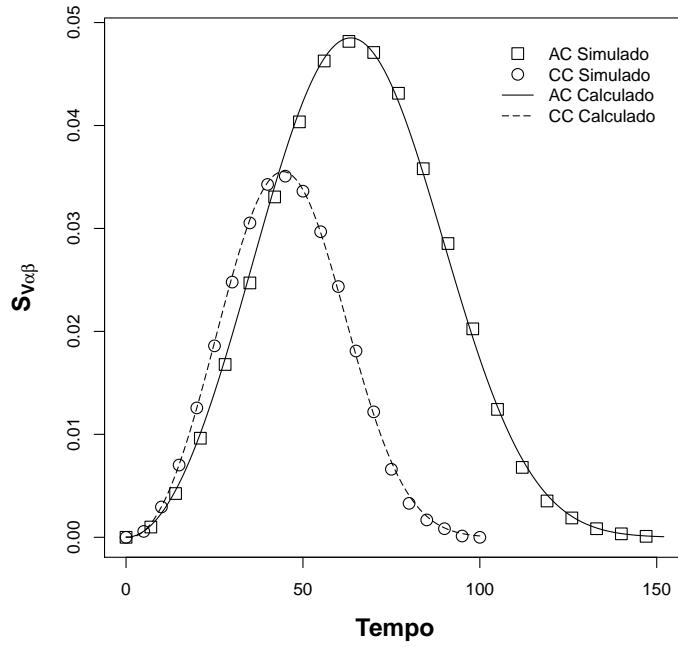


Figura 6.9: Evolução temporal de  $S_{V\alpha\beta}$  das simulações pelos métodos AC e CC com nucleação uniforme comparadas ao calculado pela Eq. 2.3.

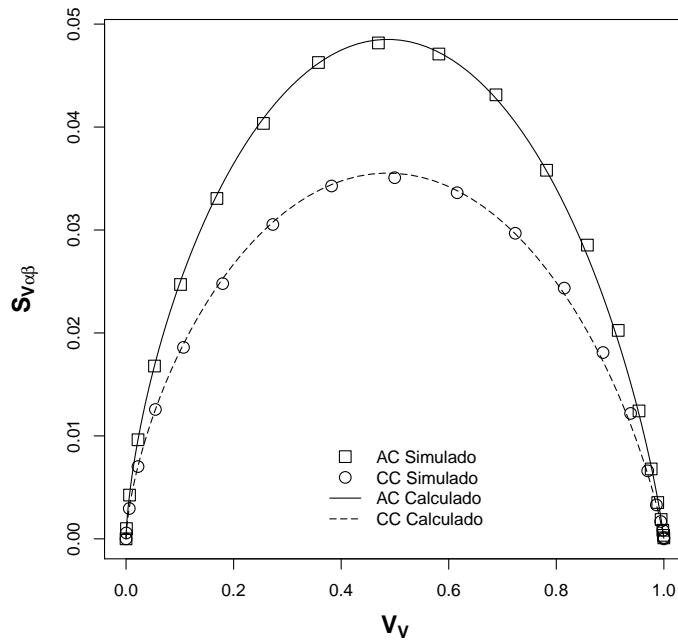


Figura 6.10: Comparação entre o caminho microestrutural das simulações de AC e CC com nucleação uniforme.

cando as Eqs. 2.1 e 2.3 à Eq. 2.2 com a devida substituição dos parâmetros chega-se ao valor de  $1/3$  para crescimento por AC e  $2/3$  para o crescimento por CC. Já com relação

aos valores medidos na simulação,  $G$  é igual a  $0.336 \pm 0.004$  para o AC e  $0.656 \pm 0.011$  para o CC o que mostra boa concordância entre os valores teóricos e medidos além de estarem de acordo com o trabalho de outros autores [27]. A Fig. 6.11 traz a evolução de  $\langle G \rangle$  no tempo. A velocidade inicial medida na simulação por CC é a mesma da simulação por AC e, em seguida, tem um rápido crescimento mantendo-se aproximadamente constante no restante da simulação. O valor coincidente na medição inicial deve-se ao fato de que o crescimento a partir de um núcleo que ocupa apenas uma célula é o mesmo na primeira iteração para ambos os métodos de crescimento. Após a primeira iteração, CC cresce com o dobro da velocidade de AC. A discrepância entre os valores finais da simulação com relação ao valor calculado indica que nos últimos passos a simulação deixa de ser tão similar ao fenômeno, porém, sendo isso restrito a um pequeno intervalo, não há prejuízo relevante para o estudo.

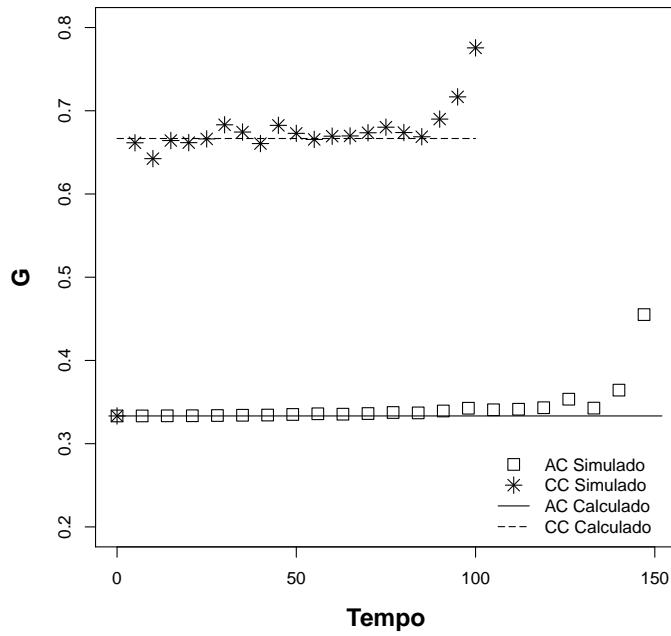


Figura 6.11: Comparação entre a velocidade média de crescimento dos contornos de grãos das simulações de AC e CC com nucleação uniforme.

Com relação às vantagens e desvantagens dos métodos analisados, cabem algumas observações. O algoritmo de ambos os métodos pode ser facilmente paralelizado. Embora a paralelização não tenha sido um dos objetivos deste trabalho, trata-se de uma técnica indispensável para simulações que envolvam uma grande quantidade de cálculos como é o caso de simulações de malhas com grande dimensão e muitos núcleos. Além disso, ambos

Finalmente, faz-se necessário discorrer sobre o comportamento do programa durante

a simulação. Devido à forma como foi implementado, o método CC é mais lento no início (enquanto calcula as distâncias entre as células e os núcleos), no entanto, como o cálculo é totalmente realizado antes do primeiro incremento de raio, as iterações intermediárias poderiam ser suprimidas levando diretamente à microestrutura final apenas com o uso de um incremento que fosse suficientemente grande. Embora isso possa ser útil se o interesse for apenas o estudo da microestrutura final, perde-se o registro das variações durante o crescimento. Já por sua natureza, o método AC necessita invariavelmente que sejam realizadas todas as iterações.

Em teste com 10 simulações com nucleação uniforme de 64 núcleos em uma malha  $320 \times 320 \times 320$  e com crescimento pelo método AC, o tempo médio de cada simulação (criação da matriz + nucleação + crescimento) foi de  $547,5 \pm 6,4\text{s}$ . Já para as mesmas configurações de nucleação e malha, o teste com 10 simulações com o crescimento pelo método CC resultou em um tempo médio de  $643,5 \pm 4,7\text{s}$ . O maior número de cálculos efetuados pelo método CC justifica seu maior tempo de processamento apesar de que este método conclui a transformação em menos iterações que o AC. Uma das vantagens dos métodos de nucleação e crescimento estudados é que podem ser facilmente paralelizados. Embora a paralelização não tenha sido um dos objetivos deste trabalho e apenas uma abordagem inicial tenha sido realizada, trata-se de uma técnica indispensável para simulações que envolvam uma grande quantidade de cálculos como é o caso de simulações de malhas com grande dimensão e muitos núcleos.

# Capítulo 7

## Conclusões e Trabalhos Futuros

### 7.1 Conclusões

A simulação de uma transformação de fase em metais foi implementada com sucesso possibilitando o estudo qualitativo e quantitativo do tema. O uso do paradigma da Orientação a Objetos mostrou-se útil ao permitir que o fenômeno fosse dividido em partes intuitivas e possibilitar que melhorias futuras sejam realizadas de modo muito mais direto e com menos riscos ao comprometimento do restante do código.

A escolha das classes se mostrou adequada permitindo que métodos operacionais como a leitura e escrita de arquivos fossem preparados a fim de atender os métodos que de fato abstraem os aspectos naturais simulados e retornar as medidas necessárias ao estudo.

A classe para simulação da Nucleação possibilitou a implementação da distribuição aleatória uniforme dos núcleos, semelhante a muitas formas de nucleação observadas na Natureza, e da distribuição periódica, uma forma de nucleação que evidenciou como o programa pode ser usado para a exploração de situações que são praticamente impossíveis de serem simuladas experimentalmente. A possibilidade de importação da posição de núcleos possibilita que resultados de outros programas sejam aproveitados e a possibilidade de salvar os núcleos gerados por este programa permite que vários métodos sejam aplicados às mesmas condições iniciais e assim melhor comparados.

Os métodos Autômato Celular e Cone Causal foram apresentados em conjunto possibilitando a percepção de suas vantagens e desvantagens. Ambos geraram microestruturas que, ao serem comparadas com microestruturas reais, mostram-se bastante semelhantes. Estabelecer qual deles é o melhor depende muito do tipo de estudo que se pretende realizar. Por exemplo, foi percebido que o método Autômato Celular é o mais rápido e

pode ser preferível em termos de velocidade. Todavia, o método Cone Causal pode ser adaptado para gerar a microestrutura final em apenas uma iteração após a realização dos cálculos de distância entre células e núcleos e pode ser preferível se o interesse do estudo for apenas na microestrutura final sem levar em conta seu desenvolvimento. É de grande destaque ainda a concordância entre as medições realizadas na simulação e os valores preditos analiticamente. Além disso, cabe realçar que as equações usadas nas previsões não são fruto de ajustes para se aproximar dos valores da simulação, mas são expressões baseadas nos aspectos geométricos desenvolvidos durante as transformações, isto é, em regras de transição baseadas nos fenômenos naturais, o que acentua a compatibilidade entre os resultados analíticos e os obtidos pelo modelo.

Sendo assim, ressalta-se a importância das simulações computacionais como forma de confirmação e até descoberta de conceitos teóricos. O programa desenvolvido pode realizar simulações que possibilitam estudos confiáveis que seriam muito mais onerosos e, em alguns casos, quase impraticáveis se executados por meio de experimentos convencionais em laboratório.

## 7.2 Trabalhos Futuros

O programa desenvolvido é base para diversas aplicações em estudos na área de Metallurgia e ainda em outras, sempre que fenômenos naturais puderem ser simulados de forma análoga à nucleação e crescimento de grãos. Pode-se enunciar algumas das melhorias que podem ser realizadas e propostas de estudos possíveis a partir deste desenvolvimento:

1. Implementação de interface gráfica para uso deste software;
2. Implementação de novas formas de nucleação (com outras distribuições aleatórias, em *clusters*, em vértices e arestas, etc.) e da nucleação durante o crescimento. Implementação de nucleação de fases distintas;
3. Implementação de crescimento com velocidade variável;
4. Implementação de novos tipos de arquivos de saída para outros programas de análise;
5. Melhoria do código tornando-o ainda mais adequado ao paradigma de orientação à objetos;
6. Paralelização de todas as estruturas do código adequadas para o recebimento dessa técnica e melhoria dos algoritmos usados visando ganhos de desempenho;

7. Inclusão do método de Monte Carlo para o estudo do crescimento de grãos após a conclusão da recristalização.

## APÊNDICE A - Desenvolvimento de Software para Determinação de Vértices e Areias de Grãos

A determinação da posição de vértices, arestas e faces dos grãos nas simulações é de grande interesse pois, entre outras coisas, permite o estudo da nucleação nesses sítios – fenômeno que acontece na Natureza nos contornos de grão. No entanto, embora visualmente seja uma tarefa simples na maioria dos casos determinar onde estão esses contornos, computacionalmente há importantes dificuldades nessa identificação, principalmente em simulações em três dimensões.

Uma das dificuldades está no caráter discreto da malha. Por causa disso, geralmente o que se pode considerar como sendo os vértices, as arestas e as faces dos poliedros que representam os grãos não são pontos, retas e planos perfeitos, respectivamente. Todavia, a aproximação dessas regiões torna-se melhor com o aumento da relação entre o tamanho da malha e o número de grãos.

No entanto, apesar dessa limitação, ainda é natural pensar em um tratamento geométrico desse problema de identificação.

Uma possível abordagem está em primeiramente determinar os planos que compõem as faces dos grãos e então, pela interseção deles, calcular as arestas e vértices. Para tanto, surge como primeiro desafio o modo pelo qual esses planos podem ser determinados.

Inicialmente, por meio do método para o cálculo de  $S_V$ , é encontrado o conjunto de grãos que fazem interseção com cada grão. Esse conjunto de vizinhos será útil para a determinação das interfaces. O mesmo método pode ser usado para identificação das faces, pois ele consiste em detectar a mudança no estado de células vizinhas, o que acusa a interface entre grãos ou a interface entre região transformada e não transformada.

Uma vez que a posição dos núcleos é conhecida (Cap. 5), toma-se um par qualquer de núcleos  $N_0$  e  $N_1$ . Pode-se traçar uma reta entre esses núcleos e estabelecer ali um novo

sistema de coordenadas  $\nu$ , com  $N_0$  estando na posição  $\nu = 0$  e  $N_1$  na posição  $\nu = \nu_1$ . Como neste trabalho o crescimento dos raios dos grãos tem velocidade constante e igual para todos os grãos, eles crescerão com velocidade  $\dot{\rho}_0$  e  $\dot{\rho}_1$ , com  $\dot{\rho}_1 = -\dot{\rho}_0$ . Assim, após simples cálculos, é visto que as interfaces dos grãos se encontrará no ponto  $\nu = \frac{\nu_1}{2}$ . Raciocínio análogo pode ser feito para o caso de velocidades variáveis ou diferentes entre os grãos.

Sendo encontrada a posição de interseção, traça-se um plano perpendicular ao segmento  $\overline{N_0N_1}$ . Esse plano corresponde à interface entre estes dois grãos. Repetindo esse processo para todos os pares de vizinhos, um conjunto de planos é gerado. Assim, as interseções desses planos resultarão nos vértices e arestas.

Desse modo, o método acima foi implementado de forma orientada a objetos. Foram criadas as classes Ponto, Plano, Grão e Microestrutura. Essas classes contêm os atributos para representação dessas entidades no espaço e métodos para os cálculos geométricos. Seus nomes indicam intuitivamente o que elas representam. Além disso, a classe Microestrutura fornece uma forma de armazenar e fazer as interações entre os objetos da classe Grão, fornecendo o cálculo dos vértices. Um exemplo de resultado para simulações de nucleação uniforme e periódica com 64 núcleos em uma malha de  $128 \times 128 \times 128$  células pode ser visto na Fig. A.1.

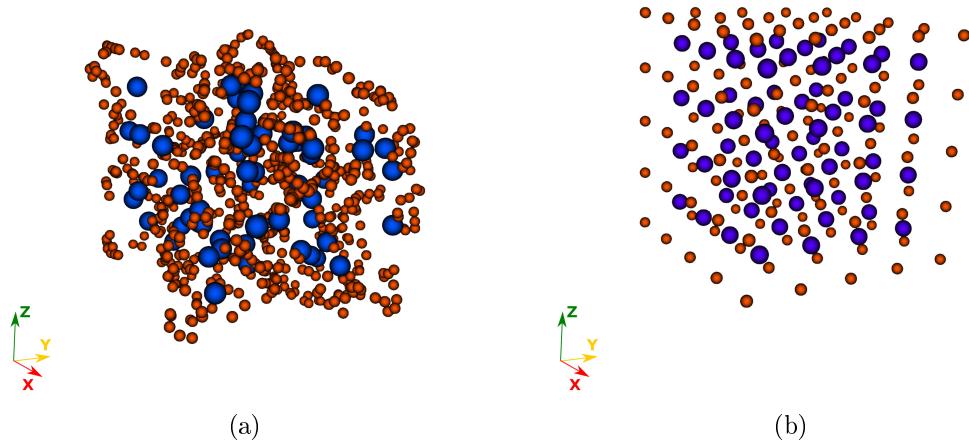


Figura A.1: Representação dos vértices calculados e dos núcleos (esferas maiores) gerados em uma distribuição (a) uniforme e (b) periódica.

Uma das dificuldades encontradas nessa implementação está na natureza de ponto flutuante das grandezas envolvidas. Logo, por questões de representação numérica e do espaço discreto, em alguns casos, vários vértices são encontrados ao redor do que seria um vértice “verdadeiro”. Por isso, como visto na Fig. A.1, a identificação dos vértices funciona bem na nucleação periódica devido à sua simetria, porém na nucleação uniforme

há agrupamentos de pontos próximos ao que seriam os vértices. Além disso, não é trivial a determinação de quais pares de vértices pertencem a mesma aresta e o método para determinar as posições das arestas não havia sido concluído.

No entanto, ainda em busca de solucionar este problema de identificação de vértices e arestas, foi feita uma adaptação da biblioteca Voro++ para esta aplicação [28]. A biblioteca Voro++ é disponível na linguagem C++ e utiliza um método parecido com o anteriormente explanado, porém é característica do encapsulamento das classes que se possa usar de suas funcionalidades sem o completo entendimento de como a classe calcula seus resultados. Portanto, foge ao escopo deste trabalho o detalhamento de como esta biblioteca funciona, estando este disponível na documentação.

A biblioteca Voro++ constrói uma rede de poliedros de Voronoi a partir da entrada das posições dos núcleos. Isso é feito considerando ou não condições de contorno periódicas. Ela retorna, entre outras coisas, os vértices dos poliedros e uma representação ordenada par a par dos vértices que compõem os segmentos que formam as arestas. No entanto, esses pontos possuem representação puramente geométrica sem ligação direta com as células da malha.

A fim de aproveitar os resultados obtidos com Voro++ e fazer a correspondência entre os os valores retornados por ela com as células da malha, foi criado um programa que usa essa biblioteca como ferramenta para gerar uma lista das células relacionadas aos vértices e arestas. O sistema de coordenadas da biblioteca é ajustado com base no sistema de coordenadas da malha usada na simulação. As coordenadas de cada vértice são usadas diretamente na identificação das células correspondentes, em alguns casos sendo necessário um arredondamento simples. Todavia, a identificação das arestas ainda constitui um desafio maior.

O problema da identificação das arestas foi então solucionado com o uso do conceito de equações paramétricas de retas. Dado um par de vértices ( $\mathbf{V}_1$  e  $\mathbf{V}_2$ ) já indicado por Voro++ como sendo pertencente a mesma aresta, uma equação da reta que passa por eles será da forma:

$$\mathbf{P} = \mathbf{V}_1 - u(\mathbf{V}_1 - \mathbf{V}_2), \quad 0 \leq u \leq 1. \quad (\text{A.1})$$

O parâmetro  $u$  estabelece a posição do ponto  $\mathbf{P}$ , a partir de um incremento sobre  $\mathbf{V}_1$  na direção  $(\mathbf{V}_1 - \mathbf{V}_2)$ . Para  $u = 0$ ,  $\mathbf{P} = \mathbf{V}_1$  e para  $u = 1$ ,  $\mathbf{P} = \mathbf{V}_2$ . Assim, resta estipular um valor incremental para ser aplicado a  $u = 0$  de modo a identificar as células do intervalo

entre  $\mathbf{V}_1$  e  $\mathbf{V}_2$  que sejam pertencentes a uma aresta. Tomando um incremento  $k = \frac{1}{\|\mathbf{V}_1 - \mathbf{V}_2\|}$ , garante-se que não haverá descontinuidade no caminho formado pelo conjunto de células que ligam  $\mathbf{V}_1$  e  $\mathbf{V}_2$  (A notação  $\|\mathbf{V}_1 - \mathbf{V}_2\|$  refere-se à norma Euclidiana, isto é, ao comprimento da aresta que liga os vértices  $\mathbf{V}_1$  e  $\mathbf{V}_2$ ). Isso é suficiente pois a cada iteração será adicionado um segmento de comprimento  $l = \|k(\mathbf{V}_1 - \mathbf{V}_2)\| = k\|\mathbf{V}_1 - \mathbf{V}_2\| = \frac{1}{\|\mathbf{V}_1 - \mathbf{V}_2\|}\|\mathbf{V}_1 - \mathbf{V}_2\| = 1$ , partindo de  $\mathbf{V}_1$  até  $\mathbf{V}_2$ . Como cada célula da malha tem dimensão unitária, a projeção de  $l$  sobre os eixos será  $\leq 1$ , logo, executando-se as iterações, uma série contínua de células será identificada como parte de uma aresta.

Um exemplo do resultado desta aplicação é exposto pela Fig. A.2 para as mesmas simulações de nucleação usadas na Fig. A.1.

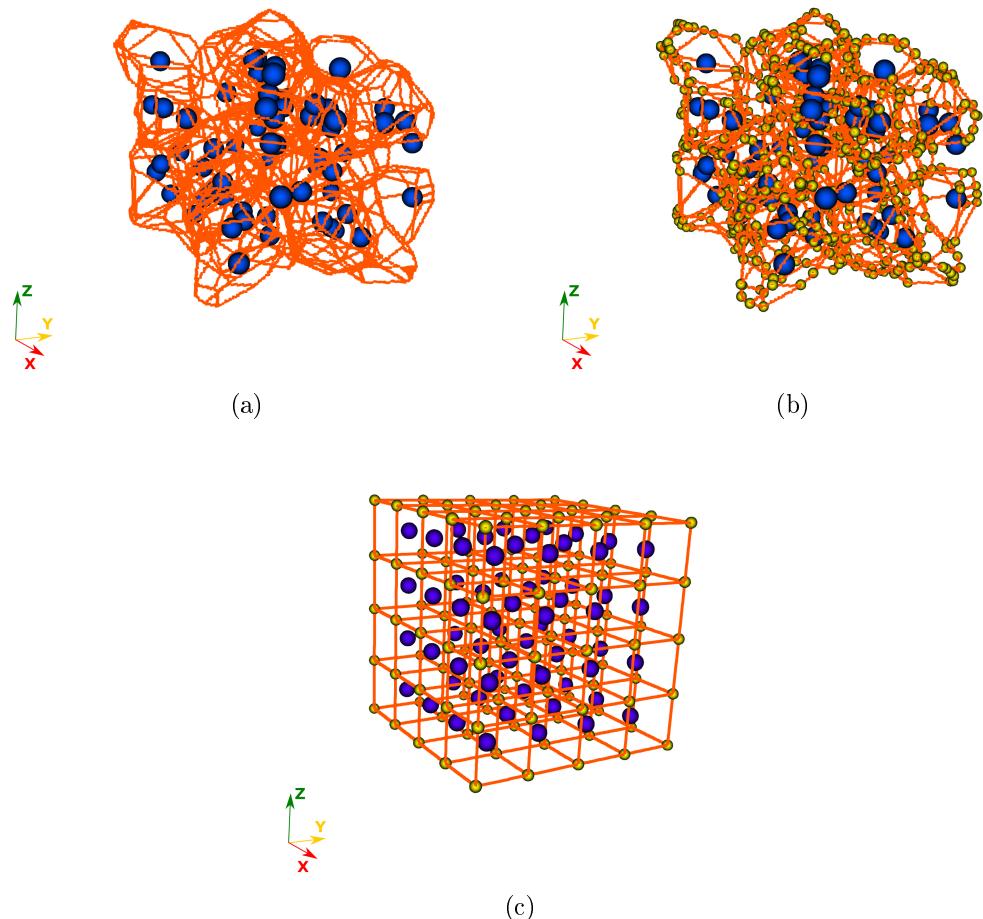


Figura A.2: Representação dos vértices e arestas calculados com auxílio da biblioteca Voro++ e dos núcleos (esferas maiores) gerados em uma nucleação (a) e (b) uniforme e (c) periódica. Em (a) foram exibidas apenas as arestas encontradas, enquanto em (b) e (c) também são exibidos os vértices

Em duas dimensões é possível uma melhor percepção da estrutura de Voronoi como

pode ser visto na Fig. A.3 de uma simulação de nucleação Uniforme com 64 núcleos em uma malha  $128 \times 128$ . Pode-se notar as arestas em posições equidistantes a cada par de núcleos vizinhos. Os aspecto não linear das arestas deve-se à condição discreta da malha.

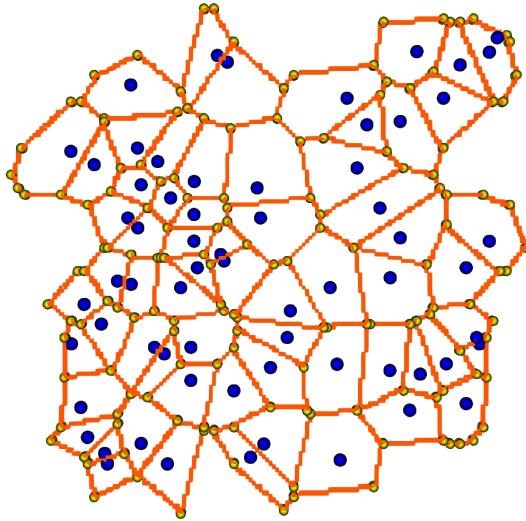


Figura A.3: Representação dos vértices e arestas calculados com auxílio da biblioteca Voro++ e dos núcleos gerados em uma nucleação uniforme em 2D.

Ao contrário do que era possível anteriormente, o programa atual pode determinar os vértices com maior precisão e sem pontos repetidos em todos os modos de nucleação. Além disso, agora a determinação das arestas já pode ser realizada. Portanto, a partir dessas determinações das posições das arestas e vértices dos grãos, tornam-se possíveis inúmeros trabalhos explorando formas de nucleações de novas fases nesses sítios.

## APÊNDICE B - Modo de uso do ParaView

ParaView é um software de código aberto e multiplataforma para análise e visualização de dados [1]. As visualizações da microestrutura simulada do Cap. 6 e do Ap. A foram feitas com esse programa. A seguir, será apresentado um breve tutorial para a criação das imagens usadas neste trabalho no qual foi utilizada a versão 5.4.1 64-bit de ParaView. Para informações complementares convém consultar a página de formatos de dados suportados por ParaView [23] e a sua documentação [7].

### B.1 Visualização com arquivos .vtk

VTK (*Visualization Toolkit*) é uma ferramenta para computação gráfica, consistindo em um conjunto de bibliotecas desenvolvidas em C++ com aplicação em outras linguagens [29]. ParaView suporta a leitura do formato de arquivos fornecido por VTK (.vtk) que, no caso deste trabalho, é escrito pela classe “Saída”. Abaixo há um exemplo ilustrativo de como o arquivo .vtk deve ser escrito.

As três primeiras linhas do arquivo .vtk contêm o cabeçalho e o indicativo da codificação usada. A quarta linha refere-se a estrutura como os dados devem ser representados. Neste trabalho foi usada uma estrutura que gerasse uma malha equivalente àquela usada nas simulações. A quinta linha indica as dimensões da malha; no exemplo da Fig. B.1 trata-se de uma representação com dimensão  $2 \times 2 \times 2$ . Na sexta linha há a indicação da quantidade de pontos representados e o formato dos dados (inteiro). Da sétima à décima quarta linha são escritos as coordenadas de cada ponto. A partir da décima quinta linha, são indicados os valores relativos aos estados de cada ponto (equivalentes ao estado de cada célula na simulação), importando saber que o valor 8 nas linhas quinze e dezessete são referentes ao número de pontos e “S” nesta mesma linha é o nome da variável que representa os estados. Já a linha dezoito contém os valores dos estados de cada um desses 8 pontos.

```

# vtk DataFile Version 3.0
vtk output
ASCII
DATASET STRUCTURED_GRID
DIMENSIONS 2 2 2
POINTS 8 int
0 0 0
0 0 1
0 1 0
0 1 1
1 0 0
1 0 1
1 1 0
1 1 1
POINT_DATA 8
FIELD Stat 1
S 1 8 int
3 2 2 4 2 3 0 2

```

Figura B.1: Exemplo de arquivo .vtk.

No programa ParaView, a leitura deve ser feita do seguinte modo: “File” > “Open” > Selecione o arquivo. Em seguida, acione “Apply”. Em “Coloring” selecione o nome da variável de estado (neste caso, “S”) e em “Representation”, selecionar alguma das opções e observar qual o resultado é mais próximo do desejado.

## B.2 Visualização com arquivos .csv

O formato .csv (“Comma Separated Value”) é amplamente usado em muitos contextos tendo sido popularizado pelo seu uso no programa Microsoft Excel [9]. Embora tenha-se usado o formato .vtk para a representação de microestruturas do Cap. 6, foi mais prático representar os vértices e arestas do Ap. A por meio do formato .csv. A Fig. B.1 ilustra como o arquivo .csv deve ser escrito para um exemplo análogo ao da seção anterior. Neste caso, apenas deve-se incluir na primeira linha os nomes das coordenadas separados por vírgula ou ponto e vírgula, sendo seguida nas demais linhas pelas coordenadas de cada ponto.

A sua leitura no ParaView é feita da seguinte forma: “File” > “Open” > Selecione o arquivo. Na opção “Field Delimiter Characters” escreva “,” ou “;” conforme estiver no arquivo .csv usado e acione “Apply”. Em seguida, selecione “Filters” > “Alphabetical” >

```
x; y; z
0; 0; 0;
0; 0; 1;
0; 1; 0;
0; 1; 1;
1; 0; 0;
1; 0; 1;
1; 1; 0;
1; 1; 1
```

Figura B.2: Exemplo de arquivo .csv.

“Table to Points”. Em “X Column”, “Y Column” e “Z Column” selecione as variáveis adequadas e acione “Apply”. Clique na área onde a imagem deveria aparecer e, em seguida, acione o símbolo de um olho ao lado do item recém criado “TableToPoints” no “Pipeline Browser”. Neste trabalho foi usada a opção “Point Gaussian” em “Representation”. Além disso, é importante mencionar que ParaView suporta a abertura e manipulação de mais de um arquivo ao mesmo tempo para compor uma única imagem. Dessa forma, para criação das imagens que contêm núcleos, arestas e vértices (ex. Fig. A.2) basta abrir consecutivamente os arquivos contendo suas posições e realizar as manipulações apresentadas acima em cada um deles identificando-os no “Pipeline Browser”.

## APÊNDICE C - Script na linguagem R usado no trabalho

R é uma linguagem interpretada de código aberto que fornece ferramentas práticas para a realização de cálculos e criação de gráficos [24]. Os gráficos do Cap. 6 foram criados com auxílio dessa linguagem com base nos dados gerados na simulação. O script usado tanto calcula analiticamente os valores de  $V_V$ ,  $S_{V\alpha\beta}$  e caminho microestrutural, como também realiza operações com os valores calculados pelos métodos da simulação. São utilizados quatro arquivos de entrada: “VvAC.dat”, “SvabAC.dat”, “VvCC.dat” e “SvabCC.dat”, que contêm os valores calculados durante a simulação de  $V_V$  e  $S_{V\alpha\beta}$  nos métodos AC e CC para comparação. Os arquivos devem ser formatados conforme o exemplo da Fig. C.1.

```

0 0
1 1.95313e-06
2 1.36719e-05
3 4.88281e-05
4 0.000123047
5 0.000251953
6 0.000451172
7 0.000736328
8 0.00112305
9 0.00162695

```

Figura C.1: Exemplo de arquivo para entrada de dados referentes a  $V_V$  e  $S_{V\alpha\beta}$  no Script de R. A primeira coluna refere-se à iteração e a segunda ao valor calculado pelos métodos da simulação.

Segue abaixo o script utilizado no trabalho.

```

1 # Script para geracao de graficos de Vv, Svab, Caminho Microestrutural
2 # e Velocidade Media dos Contornos de Grao
3 # Autor: Jonathas Luis Groetares Ferreira
4 # Mestrado em Modelagem Computacional em Ciencia e Tecnologia – MCCT

```

```

5 # Nucleo de Modelamento Microestrutural – NMM
6 #
7 # Observacao: Verificar se os arquivos para leitura estao
8 # na pasta do sistema
9 #
10 #
11
12 # Limpeza do Ambiente
13 rm(list = ls())
14
15 # AUTOMATO CELULAR
16
17 # Leitura da Fracao Volumetrica
18 frac.volAC <- read.table("VvAC.dat")
19
20 # Leitura da Svab
21 frac_areABAC <- read.table("SvabAC.dat")
22
23 # Transformacao em Matriz
24 MVVAC <- data.matrix(frac.volAC)
25 MSvabAC <- data.matrix(frac_areABAC)
26
27 # Atribuicao dos valores para vetores
28 tempoAC <- MVVAC[,1] # O tempo do AC e maior que o do CC
29 VVAC <- MVVAC[,2]
30 SvabAC <- MSvabAC[,2]
31
32 # Velocidade dR/dt
33 GGAC = 1 # 1 celula por unidade de tempo
34
35 # Calculo da constante N (Numero de nucleos por unidade de volume)
36 NAC <- VVAC[2]
37
38 # Calculo da Vv
39 # O tempo do CC e t-1
40 Vv_calcAC <- 1 - exp(-(4/3)*NAC*(GGAC^3)*((tempoAC)^3))
41
42 # Calculo da Svab
43 Sv_calcAC <- 12*NAC*(GGAC^3)*((tempoAC)^2)*
44 exp(-(4/3)*NAC*(GGAC^3)*((tempoAC)^3)) #O tempo do CC e t-1
45
46 # Calculo de Caminho Microestrutural
47 SVVAC <- (3*GGAC*(36*NAC)^(1/3))*
```

```

48 (1-Vv_calcAC)*((log(1/(1-Vv_calcAC)))^(2/3))
49
50 # Calculo de velocidade de interface G
51 tamanhoAC <- length(VVAC) - 1
52 GAC <- 0
53 for (i in 1:tamanhoAC)
54 {
55   GAC[i] <- 2*(-VVAC[i] + VVAC[i+1]) / (SvabAC[i]+SvabAC[i+1])
56 }
57 GAC[tamanhoAC+1] <- GAC[tamanhoAC]
58
59 # G teorico
60 GteoAC <- rep(1/3,tamanhoAC+1)
61
62 ######
63 # Remocao de pontos excessivos
64
65 # Quinta parte (100/20)
66 intervaloAC <- seq(1,length(tempoAC),by = length(tempoAC)%/20)
67 temporedAC <- tempoAC[intervaloAC]
68 VVredAC <- VVAC[intervaloAC]
69 SvabredAC <- SvabAC[intervaloAC]
70
71 # Este ponto e mais representativo
72 intervaloAC[length(intervaloAC)] = length(GAC)-2
73 GredAC <- GAC[intervaloAC]
74
75 # CONE CAUSAL
76
77 # Leitura da Fracao Volumetrica
78 frac_volCC <- read.table("VvCC.dat")
79
80 # Leitura da Svab
81 frac_areNRCC <- read.table("SvabCC.dat")
82
83 # Transformacao em Matriz
84 MVVCC <- data.matrix(frac_volCC)
85 MSvabCC <- data.matrix(frac_areNRCC)
86
87 # Atribuicao dos valores para vetores
88 tempoCC <- MVVCC[,1]
89 VVCC <- MVVCC[,2]
90 SvabCC <- MSvabCC[,2]

```

```

91
92 # Velocidade dR/dt
93 GCC = 1 # 1 celula por unidade de tempo
94
95 # Calculo da constante N (Numero de nucleos por unidade de volume)
96 NCC <- VVCC[ 2 ]
97
98 # Calculo da Vv
99 #ajuste tempo - 1
100 Vv_calcCC <- 1 - exp(-(4/3)*pi*(GCC^3)*NCC*((tempoCC-1)^3))
101
102 # Calculo da Svab
103 Sv_calcCC <- 4*pi*1.5*NCC*(GCC^3)*((tempoCC-1)^2)*
104 exp(-(4/3)*NCC*(GCC^3)*pi*((tempoCC-1)^3)) #ajuste (tempo - 1)
105
106 # Calculo de Caminho Microestrutural
107 SVVCC <- 1.5*GCC*((3*12*pi*NCC)^(1/3))*(1-Vv_calcCC)*
108 ((log(1/(1-Vv_calcCC)))^(2/3))
109
110 # Calculo de velocidade de interface G
111 tamanhoCC <- length(VVCC) - 1
112 GCC <- 0
113 for ( i in 1:tamanhoCC )
114 {
115   GCC[ i ] <- 2*(-VVCC[ i ] + VVCC[ i+1 ]) / (SvabCC[ i ]+SvabCC[ i+1 ])
116 }
117 GCC[tamanhoCC+1] <- GCC[tamanhoCC]
118
119 # G teorico
120 GteoCC <- rep(1/3 * 2,tamanhoCC+1) # Ajuste (* 2)
121 #####
122 ##### # Remocao de pontos excessivos
123
124
125 # Quinta parte (100/20)
126 intervaloCC <- seq(1,length(tempoCC),by = length(tempoCC)%/20)
127 temporedCC <- tempoCC[intervaloCC]
128 VVredCC <- VVCC[intervaloCC]
129 SvabredCC <- SvabCC[intervaloCC]
130
131 # Este ponto e mais representativo
132 intervaloCC[ length(intervaloCC) ] = length(GCC)-2
133 GredCC <- GCC[intervaloCC]

```

```

134
135 ##### GRAFICOS #####
136 # GRAFICOS #
137 #####
138
139 # O Grafico do AC e maior que o do CC
140
141 # Grafico de Vv
142
143 # png(file = "grafVV.png", bg = "white")
144 pdf(file = "grafVV.pdf", bg = "white")
145
146 par(mar = c(5,5,4,2))
147
148 # AC Calculada
149 grafVV <- plot(tempoAC,Vv_calcAC, type = "l", xlab = "Time",
150 ylab = expression(bold(V[V])), col="black", cex.lab= 1.4,
151 font.lab = 2, lty = 1)
152
153 # AC Simulada
154 grafVV <- points(temporedAC,VVredAC, col="black", cex = 1.5, pch = 0)
155
156 # CC Calculada
157 grafVV <- lines(tempoCC,Vv_calcCC, col = "black", lty = 5) # longdash
158
159 # CC Simulada
160 grafVV <- points(temporedCC,VVredCC, col="black", cex = 1.5, pch = 1)
161
162 legend(x = 105, y = 0.2,
163 legend=c("Simulated CA", "Simulated CC", "Calculated CA",
164 "Calculated CC"), lty=c(NA, NA, 1, 5), lwd=1:1, bty="n",
165 pch=c(0,1,NA,NA), col=c("black","black", "black","black"),
166 pt.cex = 1.5)
167
168
169 dev.off()
170 #####
171 #####
172
173 # Grafico de Svab
174
175 # png(file = "grafSvab.png", bg = "white")
176 pdf(file = "grafSvab.pdf", bg = "white")

```

```

177
178 par( mar = c(5,5,4,2))
179
180 # AC Calculada
181 grafVv <- plot(tempoAC,Sv_calcAC, type = "l", xlab = "Time",
182   ylab = expression(bold(S[paste("V", alpha, beta)])), col="black",
183   cex.lab= 1.4, font.lab = 2)
184
185 # AC Simulada
186 grafVv <- points(temporedAC, SvabredAC, col="black", cex = 1.5, pch = 0)
187
188 # CC Calculada
189 grafVv <- lines(tempoCC,Sv_calcCC, col = "black", lty = 5) #longdash
190
191 # CC Simulada
192 grafVv <- points(temporedCC,SvabredCC, col="black", cex = 1.5, pch = 1)
193
194 legend(x = 105, y = 0.049,
195   legend=c("Simulated CA", "Simulated CC", "Calculated CA",
196             "Calculated CC"), lty=c(NA, NA, 1, 5), lwd=1:1, bty="n",
197             pch=c(0,1,NA,NA), col=c("black", "black", "black", "black"),
198             pt.cex = 1.5)
199
200
201 dev.off()
202
203 ######
204
205 # Grafico de Caminho Microestrutural
206
207 # png(file = "caminho.png", bg = "white")
208 pdf(file = "caminho.pdf", bg = "white")
209
210 par( mar = c(5,5,4,2))
211
212 # AC Calculada
213 grafVv <- plot(Vv_calcAC, SVVAC, type = "l", xlab = expression(bold(V[V])), 
214   ylab = expression(bold(S[paste("V", alpha, beta)])), col="black",
215   cex.lab= 1.4, font.lab = 2)
216
217 # AC Simulada
218 grafVv <- points(VVredAC, SvabredAC, col = "black", cex = 1.5, pch = 0)
219

```

```

220 # CC Calculada
221 grafVv <- lines(Vv_calcCC,SVVCC, col = "black", lty = 5) #longdash
222
223 # CC Simulada
224 grafVv <- points(VVredCC,SvabredCC, col="black", cex = 1.5, pch = 1)
225
226 legend(x = 0.37, y = 0.01,
227   legend=c("Simulated CA", "Simulated CC", "Calculated CA",
228   "Calculated CC"), lty=c(NA, NA, 1, 5), lwd=1:1, bty="n",
229   pch=c(0,1,NA,NA), col=c("black","black", "black","black"),
230   pt.cex = 1.5)
231
232
233 dev.off()
234
235 #####
236
237 # Grafico de velocidade de interface G
238
239 # Neste caso, grafico principal devera ser feito com o CC
240
241 # png(file = "velocidade.png", bg = "white")
242 pdf(file = "velocidade.pdf", bg = "white")
243
244 par(mar = c(5,5,4,2))
245
246 # CC Calculada
247 grafVv <- plot(tempoCC, GteoCC, type = "l", lty = 5, xlab = "Time",
248   ylab = "G", col = "black", cex.lab= 1.4, font.lab = 2,
249   ylim = c(0.2, 0.8), xlim = c(0, 150))
250
251 # CC Simulada
252 grafVv <- points(temporedCC, GredCC, col = "black", cex = 1.5, pch = 8)
253
254 # AC Calculada
255 grafVv <- lines(tempoAC, GteoAC, col = "black", lty = 1) # longdash
256
257 # AC Simulada
258 grafVv <- points(temporedAC, GredAC, col="black", cex = 1.5, pch = 0)
259
260 legend(x = 105, y = 0.3,
261   legend=c("Simulated CA", "Simulated CC", "Calculated CA",
262   "Calculated CC"), lty=c(NA, NA, 1, 5), lwd=1:1, bty="n",
263   pch=c(0,1,NA,NA), col=c("black","black", "black","black"),
264   pt.cex = 1.5)

```

```
263 pch=c(0,8,NA,NA), col=c("black","black", "black","black"),  
264 pt.cex = 1.5)  
265  
266  
267 dev.off()
```

Script C.1: Script na linguagem R para criação de gráficos usados no trabalho.

# Referências

- [1] AHRENS, J., LAW, C., GEVECI, B., JAMES AHRENS, CHARLES L. B. GEVECI. ParaView: An end user tool for large data visualization. *The visualization handbook 836* (2005), 717–731.
- [2] ASSIS, W. L. Investigação do efeito da nucleação, da velocidade de crescimento e da distribuição da energia armazenada na recristalização pelo método do autômato celular em três dimensões. Dissertação de Mestrado, Universidade Federal Fluminense, 2006.
- [3] ASTM INTERNATIONAL. *ASTM E112-12, Standard Test Methods for Determining Average Grain Size*. West Conshohocken, PA, 2012.
- [4] AVRAMI, M. Kinetics of Phase Change. I General Theory. *Journal of Chemical Physics* 7 (dezembro de 1939), 1103–1112.
- [5] AVRAMI, M. Kinetics of Phase Change. II Transformation-Time Relations for Random Distribution of Nuclei. *The Journal of Chemical Physics* 8, 2 (1940), 212–224.
- [6] AVRAMI, M. Kinetics of Phase Change. III Granulation, Phase Change, and Microstructure. *The Journal of Chemical Physics* 9, 2 (1941), 177–184.
- [7] AYACHIT, U. *The ParaView Guide: A Parallel Visualization Application*. Kitware, 2015.
- [8] BURKE, J., TURNBULL, D. Recrystallization and grain growth. *Progress in Metal Physics* 3 (1952), 220–292.
- [9] BURTON, D. How to: The Comma Separated Value (CSV) File Format. <http://creativyst.com/Doc/Articles/CSV/CSV01.htm>. Accessed: 2018-10-27.
- [10] CAHN, J. W. The Time Cone Method for Nucleation and Growth Kinetics on a Finite Domain. *Materials Research Society Symposium - Proceedings 398* (1996), 425–437.
- [11] CURIEL, E., BOKULICH, P. Lightcones and causal structure. In *The Stanford Encyclopedia of Philosophy*, E. N. Zalta, Ed., fall 2012 ed. Metaphysics Research Lab, Stanford University, 2012.
- [12] DAGUM, L., MENON, R. OpenMP: an industry standard API for shared-memory programming. *IEEE computational science and engineering* 5, 1 (1998), 46–55.
- [13] DEITEL, H., DEITEL, P., PTR., P. H. *The Complete C++2 Training Course: The Ultimate Cyber Classroom*. How to program series. Prentice Hall PTR Interactive, 2001.

- [14] GOLDBERG, A., ROBSON, D. *Smalltalk-80: The Language*. Addison-Wesley series in computer science. Addison-Wesley, 1989.
- [15] HESSELBARTH, H. W., GÖBEL, I. R. Simulation of recrystallization by cellular automata. *Acta metallurgica et materialia* 39, 9 (1991), 2135–2143.
- [16] HUMPHREYS, F., HATHERLY, M. *Recrystallization and Related Annealing Phenomena*. Elsevier Science, 2012.
- [17] JOHNSON, W. A., MEHL, R. F. Reaction kinetics in processes of nucleation and growth. *Transactions of the American Institute of Mining, Metallurgical and Petroleum Engineers* 135 (1939), 416–442.
- [18] KOLMOGOROV, A., PETROVSKII, I., PISKUNOV, N. Study of a diffusion equation that is related to the growth of a quality of matter and its application to a biological problem. *Moscow University Mathematics Bulletin* 1 (1937), 1–26.
- [19] MOLODOV, D. *Microstructural Design of Advanced Engineering Materials*. Wiley, 2013.
- [20] NAUMOV, L. Generalized Coordinates for Cellular Automata Grids. *Computational Science — ICCS 2003 SE - 94* 2658 (2003), 869–878.
- [21] NAUMOV, L. *Modelling with cellular automata: problem solving environments and multidimensional applications*. PhD thesis, Faculty of Science (FNWI), Universiteit van Amsterdam, 2011.
- [22] O'ROURKE, J. *Computational Geometry in C*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1998.
- [23] PARAVIEW WIKI. ParaView data formats. [https://www.paraview.org/Wiki/ParaView/Data\\_formats](https://www.paraview.org/Wiki/ParaView/Data_formats). Accessed: 2018-10-27.
- [24] R CORE TEAM. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2018.
- [25] REED-HILL, R., ABBASCHIAN, R. *Physical Metallurgy Principles*. The PWS-Kent series in engineering. PWS Publishing Company, 1992.
- [26] RIOS, P., PADILHA, A. *Transformações De Fase*. Artliber, 2007.
- [27] RIOS, P. R., OLIVEIRA, V. T. D., PEREIRA, L. D. O., PEREIRA, M. R., CASTRO, J. A. A. D. Cellular automata simulation of site-saturated and constant nucleation rate transformations in three dimensions. *Materials Research* 9 (06 2006), 223 – 230.
- [28] RYCROFT, C. H. VORO++: A three-dimensional Voronoi cell library in C++. *Chaos* 19, 4 (2009).
- [29] SCHROEDER, W., MARTIN, K., LORENSEN, B. *The Visualization Toolkit (4th ed.)*. Kitware, 2006.
- [30] SEGA, M., JEDLOVSZKY, P., MEDVEDEV, N. N., VALLAURI, R. Free volume properties of a linear soft polymer: A computer simulation study. *Journal of Chemical Physics* (2004).

- [31] SMITH, B. *Object-Oriented Programming*. Apress, Berkeley, CA, 2011, p. 1–25.
- [32] TEN DYKE, R. P., KUNZ, J. C. Object-Oriented Programming. 465–478.
- [33] VAN WEERT, P., GREGOIRE, M. *C++ Standard Library Quick Reference*. Apress, 2016.
- [34] VILLA, E., RIOS, P. R. Transformation kinetics for nucleus clusters. *Acta Materialia* 57, 13 (2009), 3714–3724.
- [35] VILLA, E., RIOS, P. R. On modelling recrystallization processes with random growth velocities of the grains in materials science. *Image Analysis and Stereology* 31, 3 (2012), 149–162.
- [36] ZHOU, M., XU, G., WANG, L., YUAN, Q. The varying effects of uniaxial compressive stress on the bainitic transformation under different austenitization temperatures. 119.
- [37] ZIMMERMANN, T., DUBOIS-PÈLERIN, Y., BOMME, P. Object-oriented finite element programming: I. Governing principles. *Computer Methods in Applied Mechanics and Engineering* 98, 2 (1992), 291–303.