UNIVERSIDADE FEDERAL DE SÃO CARLOS
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA E ENGENHARIA DE MATERIAIS

SIMULAÇÃO COMPUTACIONAL EM ESCALA MICROESTRUTURAL DE COMPÓSITOS CERÂMICOS

Bruno Luchini

São Carlos
2017

UNIVERSIDADE FEDERAL DE SÃO CARLOS
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA E ENGENHARIA DE MATERIAIS

SIMULAÇÃO COMPUTACIONAL EM ESCALA MICROESTRUTURAL DE COMPÓSITOS CERÂMICOS

Bruno Luchini

Dissertação apresentada ao Programa de Pós-Graduação em Ciência e Engenharia de Materiais como requisito parcial à obtenção do título de MESTRE EM CIÊNCIA E ENGENHARIA DE MATERIAIS

Orientador: Dr. Rodrigo Bresciani Canto
Coorientador: Dr. Victor Carlos Pandolfelli
Agência Financiadora: CNPq

São Carlos
2017

DEDICATÓRIA

Dedico este trabalho a todos que de alguma maneira fazem do mundo um lugar melhor.

VITAE DO CANDIDATO

Engenheiro Mecânico pela Universidade Federal de São Carlos (2014).

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Ciência e Engenharia de Materiais

Folha de Aprovação

Assinaturas dos membros da comissão examinadora que avaliou e aprovou a Defesa de Dissertação de Mestrado do candidato Bruno Luchini, realizada em 13/02/2017:

Prof. Dr. Rodrigo Bresciani Canto
UFSCar

Prof. Dr. José de Anchieta Rodrigues
UFSCar

Prof. Dr. Ricardo Afonso Angélico
USP

AGRADECIMENTOS

Agradeço ao Prof. Dr. Rodrigo Bresciani Canto pela orientação e incentivo, contagiando a todos do grupo de pesquisa com sua visão otimista da ciência. Agradeço ao Prof. Dr. Victor Carlos Pandolfelli pela orientação e desafios propostos, provando-me que com determinação e trabalho é possível inovar.

Agradeço ao Prof. Dr. Ricardo Afonso Angélico pela amizade e pelo total apoio na realização deste trabalho. Se o mesmo alcançou seus objetivos, muito deve-se às reuniões e trocas de experiência com ele.

Agradeço aos amigos/técnicos Jhosefer e Rafael, pelo excelente trabalho desenvolvido na montagem e manutenção do LSC (Laboratório de Simulação Computacional).

Agradeço aos amigos do grupo de pesquisa: Antônio, Araldo, Caiuã, Filipe, Otávio, Rafael e Vinicius pela amizade e parcerias de trabalho desenvolvidas ao longo deste mestrado, pelas idas ao nem sempre excelente RU (restaurante universitário) e pelas pausas para o café.

Minha gratidão a todos os amigos que contribuíram de maneira direta ou indireta neste trabalho: Lucas, Doug, Costela, Balu, Sid, Sanclair, Tígano, Morais, Borela, Michael, Fausto, Come-pão, Fábio, Garutti, Baiano, Negresco...

Agradeço de forma especial aos meus pais Cássia e Rogerio, por tudo o que sou. Por confiarem em mim e sempre me incentivarem a buscar conhecimento. Agradeço aos meus avós e tios pelo exemplo de honestidade e aos meus primos e meu irmão Caique por serem minha força-motriz.

Agradeço enormemente à Fernanda Silveira Montilha (Ferzoca) pelas constantes leituras e sugestões neste trabalho, principalmente me acalmando e me dando força quando as dificuldades apareciam. À ela meu maior agradecimento, à ela meu maior amor, à ela meu coração.

RESUMO

O estudo de compósitos cerâmicos assistido por simulação computacional encontra-se em plena expansão. É de grande interesse da indústria cerâmica o desenvolvimento de modelos confiáveis de materiais o suficiente para reduzir os custos com protótipos e explorar virtualmente uma infinidade de possibilidades de composições e solicitações termomecânicas. A diferença entre as propriedades termomecânicas dos constituintes de compósitos pode induzir o surgimento de defeitos quando estes são submetidos a variações de temperatura. A modelagem computacional adequada destes sistemas nestas condições pode auxiliar no planejamento da composição do sistema, uma vez que é possível avaliar a influência da concentração de determinado constituinte no comportamento global do compósito. O presente trabalho objetiva analisar desde a influência do raio e fração volumétrica de inclusões até o efeito das diferenças de propriedades termomecânicas entre as fases no comportamento de compósitos cerâmicos submetidos a variações de temperatura. Analisar também, a aplicação de elementos coesivos na simulação da fissuração de sistemas cerâmicos. Como resultados principais, apresenta-se não só a compreensão dos efeitos de determinadas propriedades termomecânicas, mas também a construção de modelos em elementos finitos que podem ser utilizados por terceiros na investigação do comportamento termomecânico de sistemas cerâmicos.

Palavras-chave: Compósitos cerâmicos; Simulação computacional; Trincas inter-inclusões; Elementos finitos; Raio crítico; Elementos coesivos.

COMPUTATIONAL SIMULATION IN MICROSTRUCTURAL SCALE OF CERAMIC COMPOSITES

ABSTRACT

The study of ceramic composites assisted by computer simulation is well spread nowadays. It is an interest of the ceramic industry the development of materials models trustfully enough to reduce cost with prototypes and virtually explore an infinitude of materials compositions and thermomechanical loads. The mismatch of thermal and mechanical properties among the composite's phases may induce decohesions or cracks after temperature variation. The computer modeling of this behavior could auxiliate the planning of the systems' composition, as it becomes possible to evaluate the influence of the concentration of an specific constituent on the global behavior. The present dissertation aims to analyze the role of the geometrical feature, such as inclusion radius and volume fraction, in the composite behavior submitted to temperature variation. It was also analyzed the application of coesive elements to simulate the cracking phenomena in ceramic systems. The main results of this dissertation were not only the thermomechanical properties influence on the global behavior of ceramic systems, but also the construction of finite element models that might be usefull to others reseachers on the investigation of the thermomechancial behavior of distincts ceramic systems.

Keywords: Ceramic composites; Computer simulation; Inter-inclusion cracking; Finite elements; Critical radius; Cohesive elements.

PUBLICAÇÕES

- LUCHINI, B; SCIUTI, V.F.; ANGÉLICO, R.A.; CANTO, R.B.; PANDOLFELLI, V.C. (2016). Thermal expansion mismatch inter-inclusion cracking in ceramic systems. **Ceramics International**, 42(10), 12512-12515. doi:10.1016/j.ceramint.2016.05.013.

- LUCHINI, B; SCIUTI, V.F.; ANGÉLICO, R.A.; CANTO, R.B.; PANDOLFELLI, V.C. (2017) Critical inclusion size prediction in refractory ceramics via finite element simulations. **Journal of the European Ceramic Society**, 37(1), 315-321. doi:10.1016/j.jeurceramsoc.2016.08.008

- LUCHINI, B; SCIUTI, V.F.; ANGÉLICO, R.A.; CANTO, R.B.; PANDOLFELLI, V.C. Previsão do raio crítico de inclusões em cerâmicas refratárias via simulação computacional In: **Congresso Brasileiro de Cerâmica**, Águas de Lindóia. Anais do 60º CBC, 2016.

- LUCHINI, B; SCIUTI, V.F.; ANGÉLICO, R.A.; CANTO, R.B.; PANDOLFELLI, V.C. Efeito da fração volumétrica local no surgimento de trincas inter-inclusões em compósitos cerâmicos submetidos à variação de temperatura In: **Congresso Brasileiro de Cerâmica**, Águas de Lindóia. Anais do 60º CBC, 2016.

x

ÍNDICE DE ASSUNTOS

FOLHA DE APROVAÇÃO i

AGRADECIMENTOS iii

RESUMO v

ABSTRACT vii

PUBLICAÇÕES ix

ÍNDICE DE ASSUNTOS xi

ÍNDICE DE TABELAS xv

ÍNDICE DE FIGURAS xvii

LISTA DE SIGLAS E ABREVIATURAS xx

LISTA DE SíMBOLOS xxii

1 INTRODUÇÃO 1
1.1 Justificativas do trabalho 2
1.2 Objetivos 3

2 REVISÃO BIBLIOGRÁFICA 5
2.1 Elasticidade linear 5
2.1.1 Lei de Hooke 6
2.1.2 Materiais isotrópicos 7
2.1.3 Interpretação física de coeficientes de elasticidade 8
2.1.3.1 Módulo de Young e coeficiente de Poisson 8
2.1.3.2 Módulo de cisalhamento 9
2.1.3.3 Módulo de incompressibilidade 10
2.1.4 Efeitos termoelásticos 10
2.1.5 Energia de deformação 11

2.2 Modelos analíticos para previsão do raio crítico de inclusões em compósitos cerâmicos 12
2.3 Trincas inter-inclusões 16
2.4 Método dos elementos finitos 18
2.4.1 Materiais compósitos 25

3 MATERIAIS E MÉTODOS 29
3.1 Estudo 1: Modelo computacional para a determinação do raio crítico de inclusões 29
3.1.1 Materiais 29
3.1.2 Métodos 31
3.2 Estudo 2: Modelo computacional para estudo da origem das trincas inter-inclusões 34
3.2.1 Materiais 34
3.2.2 Métodos 34
3.3 Estudo 3: Estudo da influência das propriedades termomecânicas e fração volumétrica no perfil de distribuição de energia elástica armazenada na região entre as inclusões e no nível de energia total armazenado no compósito 36
3.3.1 Análise 1: variação de propriedades da matriz e das inclusões 36
3.3.1.1 Materiais 36
3.3.1.2 Métodos 37
3.3.2 Análise 2: variação de propriedades da inclusão mantendo-se fixas as da matriz 38
3.3.2.1 Materiais 38
3.3.2.2 Métodos 39
3.4 Estudo 4: Modelagem via elementos coesivos da fissuração espontânea de sistemas cerâmicos 40
3.4.1 Análise 1: Estudo do comportamento de um único elemento coesivo . 40
3.4.1.1 Materiais 40
3.4.1.2 Métodos 41

3.4.2 Análise 2: Estudo do efeito do tamanho de grão na fissuração espontânea de sistemas cerâmicos 42
3.4.2.1 Materiais 42
3.4.2.2 Métodos 44
3.4.3 Análise 3: Estudo do efeito da distribuição de orientações de grãos no comportamento de sistemas cerâmicos submetidos a variação de temperatura 45
3.4.3.1 Materiais 46
3.4.3.2 Métodos 46

4 RESULTADOS E DISCUSSÕES 53
4.1 Estudo 1: Comparação entre raios críticos de inclusões de diversos modelos 53
4.2 Estudo 2: Perfil de distribuição de energia entre inclusões para diferentes frações volumétricas 59
4.3 Estudo 3: Efeito das propriedades termomecânicas no perfil de distribuição de energia entre inclusões dispersas em matriz contínua 62
4.3.1 Análise 1: variação de propriedades da matriz e das inclusões 62
4.3.2 Análise 2: variação de propriedades da inclusão mantendo-se fixa as da matriz 64
4.4 Estudo 4: Modelagem via elementos coesivos da fissuração espontânea de sistemas cerâmicos 67
4.4.1 Análise 1: Estudo do comportamento de um único elemento coesivo . 67
4.4.2 Análise 2: Estudo do efeito do tamanho de grão na fissuração espontânea de sistemas cerâmicos 67
4.4.3 Análise 3: Estudo do efeito da distribuição de orientações de grãos no comportamento de sistemas cerâmicos submetidos à variação de temperatura 70

5 CONCLUSÕES 75
5.1 Estudo 1: Comparação entre raios críticos de inclusões de diversos modelos 75

5.2 Estudo 2: Perfil de distribuição de energia entre inclusões para diferentes frações volumétricas 75
5.3 Estudo 3: Efeito das propriedades termomecânicas no perfil de distribuição de energia entre inclusões dispersas em matriz contínua 76
5.4 Estudo 4: Modelagem via elementos coesivos da fissuração espontânea de sistemas cerâmicos 77
5.5 Conclusões gerais do trabalho 77

6 SUGESTÕES DE TRABALHOS FUTUROS 79

7 REFERÊNCIAS BIBLIOGRÁFICAS 81

ÍNDICE DE TABELAS

Tabela 2.1 Relação entre as constantes de Lamé constantes elásticas. Adaptado de Thorne & Wallace [7] 8

Tabela 3.1 Parâmetros usados nos modelos MEF e para previsão do raio crítico de inclusões. 31
Tabela 3.2 Propriedades termomecânicas da matriz 39
Tabela 3.3 Propriedades mecânicas elementos estruturais 40
Tabela 3.4 Propriedades mecânicas elemento coesivo 41
Tabela 3.5 Propriedades da cordierita, extraído de Bubeck [48] 43
Tabela 3.6 Propriedades do elemento coesivo, extraído de Fertig & Nickerson [44] 44

Tabela 4.1 Comparação dos raios críticos de inclusão obtidos via MEF (M1, M3 e Mn) e modelo analítico de Davidge e Green [9] e Liu e Winn [19] com dados experimentais da literatura. 53
Tabela 4.2 Região mais solicitado para diferentes ℓ/d 61
Tabela 4.3 Coeficientes da superfície de transição interface/inter-inclusão 64
Tabela 4.4 Coeficientes superfície energia de deformação 66
Tabela 4.5 Coeficientes de expansão térmica globais nas direções x e y para os distintos desvios padrões 71

ÍNDICE DE FIGURAS

Figura 2.1 Cubo infinitesimal representativo. Adaptado de Kaselow [6]) 6
Figura 2.2 Esquema de um ensaio de tração pura. Adaptado de Tessier-Doyen [8] 9
Figura 2.3 Esquema de um corpo submetido a uma pressão hidrostática 10
Figura 2.4 Curva de resposta de um estado uniaxial de tensões para um material linear elástico 11
Figura 2.5 Modelo de duas inclusões (de raio a) envoltas por matriz finita (de raio b) interseccionadas em um ponto (A), baseado em Liu & Winn [19] 15
Figura 2.6 a) Microfissura inter-inclusão, Joliff et al. [20] b) Modelo esquemático para o cálculo de fração volumétrica local, em que a é o raio das inclusões e b o raio da esfera inscrita em um dodecaédro rômbico 17
Figura 2.7 Processo de análise por elementos finitos. (Adaptado de Bathe [32]) 19
Figura 2.8 Desenho esquemático - zona coesiva. Adaptado de Park & Paulino [40]. 20
Figura 2.9 Esquema representativo de uma zona coesiva com “malha casada”. Adaptado do manual do software ABAQUS [39]. 21
Figura 2.10 Esquema representativo de uma zona coesiva com “malha restrita”. Adaptado do manual do software ABAQUS [39]. 21
Figura 2.11 Esquema representativo de uma zona coesiva com “malha 1/2 restrita 1/2 contato”. Adaptado do manual do software ABAQUS [39]. 21
Figura 2.12 Modelo EF extraído de Fertig & Nickerson [44] 23
Figura 2.13 Comportamento TS de um elemento coesivo genérico. Adaptado do manual do software ABAQUS [39]. 24

Figura 3.1 Procedimento para obtenção das variações de temperatura a partir dos dados experimentais de Tessier-Doyen et al. [46]. . . . 30

Figura 3.2 Modelo axissimétrico de inclusões envolta por uma matriz. (a) Modelo M1: única inclusão; (b) Modelo M3: três inclusões; (c) Modelo Mn: modelo com condição de periodicidade. 32

Figura 3.3 Modelo M2 axissimétrico de inclusões envoltas por matriz . 35

Figura 3.4 Esquema do algorítmo desenvolvido para a simulação das 28000 combinações 37

Figura 3.5 Esquema do algorítmo desenvolvido para a simulação das 3375 combinações 39

Figura 3.6 Modelo axissimétrico com um elemento coesivo e dois elementos estruturais 41

Figura 3.7 Orientações (ω) dos grãos de cordierita ($\mu = 0^\circ$ / $\sigma = 15^\circ$) . . 43

Figura 3.8 Distribuição de orientações dos grãos de cordierita ($\mu = 0^\circ$ / $\sigma = 15^\circ$) 44

Figura 3.9 Modelo EF para a simulação da fissuração espontânea de sistemas cerâmicos, baseado em Fertig & Nickerson [44] 45

Figura 3.10 Orientações dos grãos de cordierita ($\mu = 0^\circ$ / $\sigma = 5^\circ$) 47

Figura 3.11 Distribuição de orientações dos grãos de cordierita ($\mu = 0^\circ$ / $\sigma = 5^\circ$) 47

Figura 3.12 Orientações dos grãos de cordierita ($\mu = 0^\circ$ / $\sigma = 15^\circ$) 48

Figura 3.13 Distribuição de orientações dos grãos de cordierita ($\mu = 0^\circ$ / $\sigma = 15^\circ$) 48

Figura 3.14 Orientações dos grãos de cordierita ($\mu = 0^\circ$ / $\sigma = 35^\circ$) 49

Figura 3.15 Distribuição de orientações dos grãos de cordierita ($\mu = 0^\circ$ / $\sigma = 35^\circ$) 49

Figura 3.16 Orientações dos grãos de cordierita ($\mu = 0^\circ$ / $\sigma = 55^\circ$) 50

Figura 3.17 Distribuição de orientações dos grãos de cordierita ($\mu = 0^\circ$ / $\sigma = 55^\circ$) 50

Figura 3.18 Modelo simplificado para o cálculo do coeficiente de dilatação efetivo 51

Figura 4.1 Raio crítico (a_c) em função da fração volumétrica utilizando-se Davidge e Green e MEF para um sistema alumina (i) / borosilicato (m) retirado de Joliff et al. [20], assumindo-se os seguintes valores: E_i = 340 GPa, α_i = 7,6·10^{-6} °C^{-1}, ν_i = 0,24, E_m = 68 GPa, α_m = 4,6·10^{-6} °C^{-1}, ν_m = 0,2 com ΔT = 244 °C. O resultado experimental foi obtido por Joliff et al. [20] 55

Figura 4.2 Comparação da distribuição de tensões máximas principais do modelo de Liu & Winn [19] com o modelo proposto por Jollif [20] para um sistema alumina (i) / borosilicato (m), assumindo-se os seguintes valores: E_i = 340 GPa, α_i = 7,6·10^{-6} °C^{-1}, ν_i = 0,24, E_m = 68 GPa, α_m = 4,6·10^{-6} °C^{-1}, ν_m = 0,2 com ΔT = 270 °C e diferentes frações volumétricas de inclusão 56

Figura 4.3 Comparação da distribuição de tensões mínimas principais do modelo de Liu & Winn [19] com o modelo proposto por Jollif [20] para um sistema alumina (i) / borosilicato (m), assumindo-se os seguintes valores: E_i = 340 GPa, α_i = 7,6·10^{-6} °C^{-1}, ν_i = 0,24, E_m = 68 GPa, α_m = 4,6·10^{-6} °C^{-1}, ν_m = 0,2 com ΔT = 270 °C e diferentes frações volumétricas de inclusão 56

Figura 4.4 Comparação da distribuição de tensões máximas (a) e mínimas (b) principais nos pontos I e M no modelo Mn para um sistema alumina (i) / borosilicato (m), assumindo-se os seguintes valores: E_i = 340 GPa, α_i = 7,6·10^{-6} °C^{-1}, ν_i = 0,24, E_m = 68 GPa, α_m = 4,6·10^{-6} °C^{-1}, ν_m = 0,2 com ΔT = 244 °C e $a = 250\ \mu$ m 58

Figura 4.5 Comparação da distribuição de tensões máximas principais e tensões normais média no modelo Mn para um sistema alumina (i) / borosilicato (m), assumindo-se os seguintes valores: E_i = 340 GPa, α_i = 7,6·10^{-6} °C^{-1}, ν_i = 0,24, E_m = 68 GPa, α_m = 4,6·10^{-6} °C^{-1}, ν_m = 0,2 com ΔT = 244 °C e a = 250 μm 59

Figura 4.6 Energia normalizada para diferentes distâncias entre inclusões 60

Figura 4.7 Tensão máxima principal, tensão mínima principal e energia versus ℓ/d 62
Figura 4.8 Região energeticamente mais solicitada entre duas inclusões após o resfriamento de amostras heterogêneas 63
Figura 4.9 Superfície de transição interface/inter-inclusão analítica . . . 64
Figura 4.10 Trincamento ou não em diversas amostras submetidas a resfriamento 65
Figura 4.11 66
Figura 4.12 Resultados do modelo de um elemento coesivo (curvas tensão x deformação, tensão x tempo e deformação x tempo). 68
Figura 4.13 Configuração deformada dos grãos (D = 25μm).a) Configuração inicial b) Configuração deformada após ΔT de -500$^{\circ}C$ (amplificado em 50x) c) Configuração após retorno à temperatura inicial. 69
Figura 4.14 Configuração deformada após ΔT de -250$^{\circ}C$ (amplificado em 50x). Azul: elementos estruturais, vermelho: elementos coesivos, e branco: elementos coesivos deletados (σ = 15°) 70
Figura 4.15 Configuração deformada após ΔT de -500$^{\circ}C$ para distribuições com μ = 0° e diferentes σ (σ = $5, 15, 35$ e 55°) (amplificado em 100x) 73

LISTA DE SIGLAS E ABREVIATURAS

CFC - Cúbica de Face Centrada

EF - Elementos Finitos

MEF - Método dos Elementos Finitos

TS - Traction-Separation

STD - Standard deviation

LISTA DE SÍMBOLOS

a - Raio da inclusão

a_c - Raio crítico de inclusão

b - Raio da matriz esférica

d - Diâmetro da inclusão

p - Tensão normal média na interface matriz/inclusão

i - Referente à inclusão

m - Referente à matriz

$A1$ - Coeficiente equação função de transição

$A2$ - Coeficiente equação função de transição

$B1$ - Coeficiente equação função de transição

$C1$ - Coeficiente equação função de transição

$D1$ - Coeficiente equação função de transição

$\mathcal{C}$ - Tensor de rigidez

E - Módulo elástico

F - Força arbitrária

G - Módulo de cisalhamento

K - Módulo de incompressibilidade

L_0 - Comprimento inicial do corpo de prova

$M1$ - Modelo com uma inclusão

$M2$ - Modelo com duas inclusões

$M3$ - Modelo com três inclusões

Mn - Modelo com infinitas inclusões

P - Ponto arbitrário

R - Raio de revolução da amostra

S_0 - Área da seção transversal inicial do corpo de prova

U_s - Energia de formação de superfícies na matriz

U_t - Energia total armazenada durante a deformação

δ_{ij} - Delta de Kronecker

$\underline{\underline{\varepsilon}}$ - Tensor de deformações

ε_{ij} - Componente do tensor de deformações

$\varepsilon_{ij}{}^{M}$ - Componente do tensor de deformações de origem mecânica

$\varepsilon_{ij}{}^{T}$ - Componente do tensor de deformações de origem térmica

γ_s - Energia superficial por unidade de área

λ - Constante de Lamé

ℓ - Distância entre inclusões

μ - Constante de Lamé

ν - Coeficiente de Poisson

ω - Orientação do grão

ϕ - Fração volumétrica de inclusões

$\underline{\underline{\sigma}}$ - Tensor de tensões

σ_{ij} - Componente do tensor de tensões

$\bar{\sigma_\rho}$ - Coeficiente de tensão radial

$\bar{\sigma_\theta}$ - Coeficiente de tensão circunferencial

ΔT - Variação de temperatura

1 INTRODUÇÃO

Materiais compostos ou compósitos são oriundos da combinação de dois ou mais materiais com propriedades distintas [1] para cumprir uma função que quando isolados, não teriam significativo êxito. A utilização de compósitos é crescente na história da humanidade. Um dos primeiros indícios de sua aplicação data de 3400 A.C. pelos povos mesopotâmicos, empregando compósitos de madeiras empilhadas em diferentes orientações com fins estruturais [2]. Os compósitos cerâmicos ganharam grande destaque na era pós-Sputnik (1957), com o início da corrida espacial [3], a partir da necessidade de materiais funcionais, capazes de manterem forma e função durante a reentrada de espaçonaves na atmosfera terrestre, com temperaturas ultrapassando 1500°C e leves o suficiente para vencerem a força gravitacional com economia de combustível durante o lançamento.

Os materiais cerâmicos, em sua maioria, apresentam elevada dureza e alta fragilidade. Em muitos casos, a alta fragilidade pode ser compensada aliando-se estes materiais a polímeros ou metais. Entretanto, para determinadas aplicações esta alternativa não é viável, principalmente em casos que envolvem elevadas temperaturas. Este nicho tecnológico pode ser suprido a partir de compósitos cerâmica-cerâmica, os quais são geralmente projetados para aplicações estruturais, cuja heterogeneidade microestrutural é de extrema importância para a otimização das propriedades globais [4].

A grande limitação dos compósitos cerâmicos encontra-se na usual fragilidade – e consequente modo de falha catastrófico – à temperatura ambiente, limitando sua utilização em condições de vibrações extremas ou carregamentos mecânicos com bruscas variações de direção e magnitude. Inspirados na rocha silicosa sedimentar brasileira itacolumita, que devido à uma rede de fissuras apresenta comportamento macroscópico mais flexível que o material denso, alguns pesquisadores dedicam-se a mimetizar esta estrutura em compósitos cerâmicos, o que seria extremamente útil para bases de edifícios em regiões sujeitas a terremotos [5], dentre outras aplicações que ainda não foram identificadas.

Simulações computacionais estão ganhando grande destaque na pesquisa

de materiais compósitos, uma vez que fornecem inúmeras possibilidades de análises em curto tempo e baixo custo em relação à investigação experimental. A partir da automatização de simulações, pode-se analisar o comportamento termomecânico de milhares de compósitos cerâmicos em semanas, o que levariam anos para se obter os resultados experimentais. Deve-se salientar que a simulação computacional não elimina os experimentos físicos, pelo contrário, auxilia a avaliar as melhores opções a se ensaiar. Desta maneira, reduz-se desperdício de material, horas-máquina, horas-homem e se pode chegar a resultados satisfatórios em menor tempo, ressaltando-se que para tal os modelos devem ser robustos e representativos o suficiente, caso contrário, as simulações somam-se ao desperdício.

A partir do contexto acima descrito, o presente trabalho visa somar conhecimento ao vasto campo de estudos em compósitos, principalmente àqueles com partículas dispersas em matriz contínua. Abordando uma estratégia computacional, apresenta-se uma metodologia viável de pesquisa, em que recursos físicos são economizados, priorizando-se cálculos virtuais que servirão de base para a escolha de experimentos a serem realizados futuramente. Além disso, a simulação computacional será utilizada para entender fenômenos complexos no comportamento desta classe de materiais.

1.1 Justificativas do trabalho

A justificativa central do trabalho fundamenta-se na necessidade de modelos computacionais representativos para a previsão do comportamento mecânico e sob fratura de compósitos cerâmicos quando submetidos a variações térmicas. Tem-se em vista que os modelos analíticos para tal análise possuem hipóteses simplificadoras que limitam sua validade a baixas frações volumétricas da fase descontínua. Diversas constatações da literatura apontam para influências geométricas, como raio crítico de inclusão e fração volumétrica de inclusões, além das propriedades dos materiais como determinantes para o comportamento mecânico destes compósitos. A principal justificativa associada à aplicação do MEF nesta previsão encontra-se na economia de recursos materiais e de tempo

quando comparadas a ensaios físicos e na possibilidade de tratar problemas complexos, superando limitações de modelos analíticos. Ensaios virtuais estão em crescente utilização, uma vez que fornecem aos usuários informações importantes e indicam quais ensaios devem ser realizados em função das prováveis respostas a serem obtidas.

1.2 Objetivos

Os objetivos deste trabalho podem ser divididos em objetivos científicos e tecnológicos. Cientificamente, o presente trabalho objetiva analisar, por meio de simulações computacionais via MEF, a influência do raio crítico, distribuição espacial e de tamanho de inclusões, fração volumétrica e das propriedades mecânicas das distintas fases de compósitos cerâmicos bifásicos no comportamento em fratura do material sujeito a variações térmicas. Tecnologicamente, busca-se o desenvolvimento de modelos que poderão ser utilizados por terceiros no auxílio do planejamento de composições em compósitos cerâmicos para distintas finalidades.

2 REVISÃO BIBLIOGRÁFICA

Para um melhor entendimento do tema central desta dissertação, alguns assuntos da literatura foram selecionados e serão discutidos mais detalhadamente a seguir.

2.1 Elasticidade linear

Quando um corpo contínuo é submetido a uma força (tanto interna quanto externa), todo ponto pertencente a este corpo é influenciado. É comum denotar forças internas como reações devido às deformações de carregamentos externos, além das forças volumétricas, por exemplo a força gravitacional, e forças de contato. Forças volumétricas têm seus efeitos associados ao volume do corpo e à sua densidade, enquanto os efeitos das forças externas dependem da área de contato em que ela é aplicada.

Forças externas atuando em um meio contínuo em equilíbrio estático induziriam deformação no mesmo, resultando em uma mudança de forma e/ou dimensões no corpo. Forças internas atuando no meio tentariam resistir a esta deformação. Consequentemente, o meio retornaria à sua forma e dimensões iniciais quando as forças externas fossem removidas. Se este retorno à configuração original é perfeito, o meio é chamado de elástico.

Uma das leis constitutivas mais difundidas é a lei de Hooke. A forma exata do estado de tensões em um ponto arbitrário P contido no meio contínuo é dependente da orientação da força atuante em P em relação ao sistema de coordenadas de referência. Com o intuito de quantificar o estado de tensões em um ponto P resultante da ação de uma força F, adota-se um cubo infinitesimal. As tensões atuantes nos seis lados do cubo podem ser separadas em componentes normais e componentes contidas nos planos das faces. Esta situação é ilustrada na Figura 2.1 para o plano normal ao eixo 2. Para todo o texto a seguir, um plano orientado normal a um eixo i será chamado de plano i.

A tensão σ_{ij} é definida como atuando no plano i com seu versor na direção j. Logo, quando $i = j$ tratam-se de tensões normais e, no caso contrário, de tensões cisalhantes. Do equilíbrio estático tem-se que $\sigma_{ij} = \sigma_{ji}$ e assim, para representar

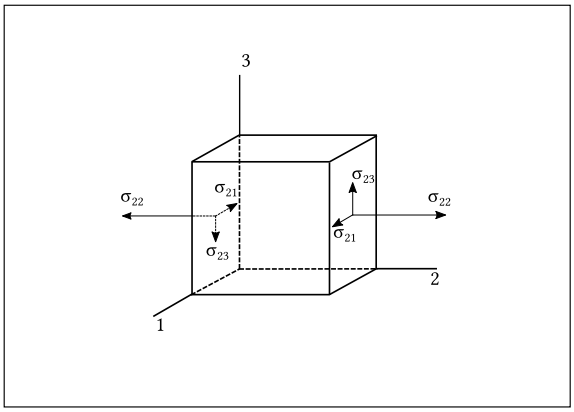

Figura 2.1 Cubo infinitesimal representativo. Adaptado de Kaselow [6])

um estado de tensões em três dimensões, tem-se 6 componentes cisalhantes e 3 componentes normais (equação 2.1).

$$\underline{\underline{\sigma}} = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{bmatrix} \tag{2.1}$$

Conforme mencionado anteriormente, quando um corpo elástico é exposto a tensões, mudanças de dimensões e forma são geradas, originando as deformações. Por definição, deformação é a mudança relativa de uma dimensão de um corpo. Em três dimensões, o tensor de deformações de um ponto P presente no meio contínuo é descrito como:

$$\underline{\underline{\varepsilon}} = \begin{bmatrix} \varepsilon_{11} & \varepsilon_{12} & \varepsilon_{13} \\ \varepsilon_{21} & \varepsilon_{22} & \varepsilon_{23} \\ \varepsilon_{31} & \varepsilon_{32} & \varepsilon_{33} \end{bmatrix} \tag{2.2}$$

2.1.1 Lei de Hooke

A partir da lei de Hooke, assume-se que as deformações são suficientemente pequenas e a dependência entre tensão e deformação dá-se de maneira linear. Para tal, o material é classificado como linear elástico. De uma maneira geral pode-se escrever:

$$\sigma_{ij} = \mathcal{C}_{ijkl}\,\varepsilon_{kl}, \qquad \text{com } i,j,k,l = 1,2,3 \tag{2.3}$$

O tensor de quarta ordem $\mathcal{C}_{ijkl}$ é conhecido como tensor de rigidez e apresenta 81 termos dependentes das constantes elásticas do material. Este tensor correlaciona a deformação de um meio com as respectivas tensões aplicadas.

Cada componente do tensor de tensões σ_{ij} é linearmente dependente dos componentes do tensor de deformações e vice-versa, relacionando-se um termo do tensor de tensões com 9 termos do tensor de deformações. Como o tensor de tensões é simétrico, i.e. $\sigma_{ij} = \sigma_{ji}$, apenas 6 destas equações são independentes.

2.1.2 Materiais isotrópicos

Materiais isotrópicos modelados pela lei de Hooke necessitam de apenas duas constantes elásticas para a caracterização de seu comportamento, chamadas de constantes de Lamé, λ e μ. Nestes materiais as propriedades elásticas de qualquer ponto P pertencente ao sólido são independentes da direção. Relacionam-se as constantes de Lamé com o tensor de rigidez $\mathcal{C}_{ijkl}$ por:

$$\mathcal{C}_{ijkl} = \left[\lambda\delta_{ij}\delta_{kl} + \mu\left(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}\right)\right] \tag{2.4}$$

em que δ é o delta de Kronecker definido como:

$$\delta_{ij} = \begin{cases} 0, & \text{se } i \neq j \\ 1, & \text{se } i = j \end{cases} \qquad i,j = 1,2,3 \tag{2.5}$$

Substituindo-se a equação 2.4 na 2.3, tem-se:

$$\sigma_{ij} = \lambda\delta_{ij}\varepsilon_{kk} + \mu\left(\varepsilon_{ij} + \varepsilon_{ji}\right) \tag{2.6}$$

Escrevendo-se a equação 2.6 na forma matricial, tem-se:

$$\begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{23} \\ \sigma_{13} \\ \sigma_{12} \end{bmatrix} = \begin{bmatrix} 2\mu+\lambda & \lambda & \lambda & 0 & 0 & 0 \\ \lambda & 2\mu+\lambda & \lambda & 0 & 0 & 0 \\ \lambda & \lambda & 2\mu+\lambda & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu \end{bmatrix} \begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{33} \\ \varepsilon_{23} \\ \varepsilon_{13} \\ \varepsilon_{12} \end{bmatrix} \tag{2.7}$$

A combinação das constantes de Lamé com outras constantes elásticas como o módulo de Young E, módulo de incompressibilidade K e o coeficiente de Poisson ν, é descrita na Tabela 2.1:

Tabela 2.1 Relação entre as constantes de Lamé constantes elásticas. Adaptado de Thorne & Wallace [7]

	λ	$\mu(G^*)$	E	ν	K
λ, μ			$\frac{\mu(3\lambda+2\mu)}{\lambda+\mu}$	$\frac{\lambda}{2(\lambda+\mu)}$	$\frac{3\lambda+2\mu}{3}$
λ, E		irracional		irracional	irracional
λ, ν		$\frac{\lambda(1-2\nu)}{2\nu}$	$\frac{\lambda(1+\nu)(1-2\nu)}{\nu}$		$\frac{\lambda(1+\nu)}{3\nu}$
λ, K		$\frac{3(K-\lambda)}{2}$	$\frac{9K(K-\lambda)}{3K-\lambda}$	$\frac{\lambda}{3K-\lambda}$	
μ, E	$\frac{\mu(2\mu-E)}{E-3\mu}$			$\frac{E-2\mu}{2\mu}$	$\frac{\mu E}{3(3\mu-E)}$
μ, ν	$\frac{2\mu\nu}{1-2\nu}$		$2\mu(1+\nu)$		$\frac{2\mu(1+\nu)}{3(1-2\nu)}$
μ, K	$\frac{3K-2\mu}{3}$		$\frac{9K\mu}{3K+\mu}$	$\frac{3K-2\mu}{2(3K+\mu)}$	
E, ν	$\frac{\nu E}{(1+\nu)(1-2\nu)}$	$\frac{E}{2(1+\nu)}$			$\frac{E}{3(1-2\nu)}$
E, K	$\frac{3K(3K-E)}{9K-E}$	$\frac{3EK}{9K-E}$		$\frac{3K-E}{6K}$	
ν, K	$\frac{3K\nu}{1+\nu}$	$\frac{3K(1-2\nu)}{2(1+\nu)}$	$3K(1-2\nu)$		

* A constante de Lamé μ é equivalente ao módulo de cisalhamento G.

2.1.3 Interpretação física de coeficientes de elasticidade

2.1.3.1 Módulo de Young e coeficiente de Poisson

Quando um sólido é submetido à uma solicitação de tração pura (F_y) na direção vertical (y), observa-se uma elongação ΔL nesta direção e uma redução da área da seção transversal, conforme a Figura 2.2.

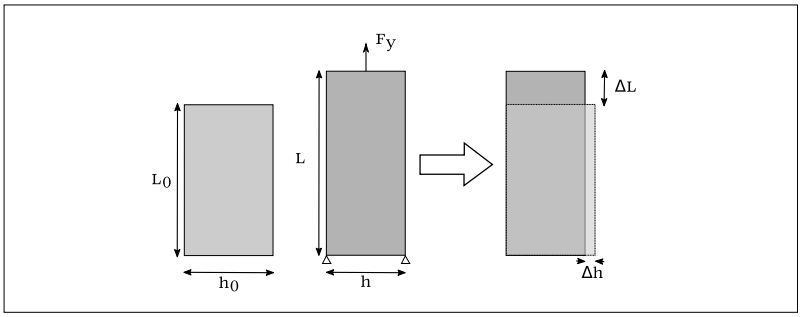

Figura 2.2 Esquema de um ensaio de tração pura. Adaptado de Tessier-Doyen [8]

Assumindo-se pequenas deformações e que o material apresente apenas comportamento linear elástico, o módulo de Young é o fator de proporcionalidade entre a tensão aplicada (σ_y) e a respectiva deformação (ε_y):

$$E = \frac{\sigma_y}{\varepsilon_y} = \frac{F_y}{S_0} \cdot \frac{L_0}{\Delta L} \tag{2.8}$$

em que S_0 é a área de seção transversal inicial.

A contração transversal ε_x oriunda do carregamento é proveniente do efeito de Poisson. Assim, o coeficiente de Poisson caracteriza a razão (em módulo) entre as deformações laterais e a deformação longitudinal. Quando $\nu = 1/2$, tem-se um material incompressível, isto é, seu volume permanece constante durante o carregamento. Se $\nu = 0$, não há redução de área da seção transversal durante um carregamento, neste exemplo. Sendo o material isotrópico, a mesma relação ocorre na direção z.

$$\nu = \frac{-\varepsilon_x}{\varepsilon_y} = \frac{-\varepsilon_z}{\varepsilon_y} \tag{2.9}$$

2.1.3.2 Módulo de cisalhamento

Quando um sólido é submetido a uma solicitação de cisalhamento puro, modificações angulares podem ser observadas. O módulo de cisalhamento é o fator de proporcionalidade entre a tensão de cisalhamento aplicada $\sigma_{ij,(i\neq j)}$ e a

deformação correspondente $\varepsilon_{ij,(i\neq j)}$, por exemplo, para o plano $1-2$, tem-se.

$$G = \frac{\sigma_{12}}{\varepsilon_{12}} \tag{2.10}$$

2.1.3.3 Módulo de incompressibilidade

Quando um material é submetido à uma pressão hidrostática (Fig. 2.3), a relação entre a pressão p (negativa) e a deformação volumétrica $\varepsilon_{vol} = tr(\underline{\underline{\varepsilon}})$ é linear, em que $tr(\cdot)$ é o operador traço. Esta linearidade é definida pelo módulo de incompressibilidade, K, pela equação:

$$p = K\varepsilon_{vol} = K\,\frac{(V - V')}{V} \tag{2.11}$$

em que V é o volume inicial e V' é o volume final.

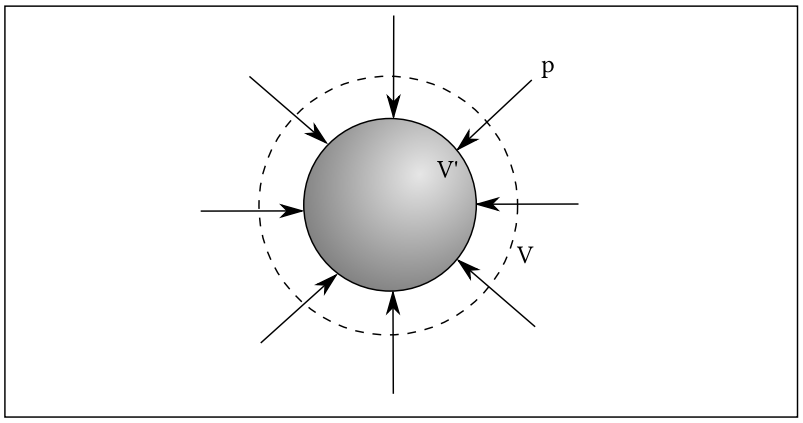

Figura 2.3 Esquema de um corpo submetido a uma pressão hidrostática

2.1.4 Efeitos termoelásticos

Para materiais isotrópicos, as deformações oriundas de variações térmicas possuem caráter dilatacional natural (expansão ou contração térmica) sem causar cisalhamento e são proporcionais à variação de temperatura:

$${\varepsilon_{ij}}^{T} = \alpha\Delta T\delta_{ij} \tag{2.12}$$

Assim, tem-se que a deformação real é o resultado da adição das deformações de origens mecânicas (${\varepsilon_{ij}}^{M}$) e das deformações de origem térmica (${\varepsilon_{ij}}^{T}$).

Reescrevendo a equação 2.3, tem-se:

$$\sigma_{ij} = \mathcal{C}_{ijkl} \left(\varepsilon_{kl} - \alpha \Delta T \delta_{kl} \right) \tag{2.13}$$

em que,

- α [$^{\circ}C^{-1}$]: coeficiente de expansão térmica linear;
- ΔT [$^{\circ}C$]: a variação de temperatura.

2.1.5 Energia de deformação

Considerando-se uma curva tensão-deformação de um material linear elástico submetido a um ensaio de tração uniaxial (Fig. 2.4), tem-se que a energia de deformação por unidade de volume é numericamente igual à área sob a curva, ou seja, $U = \frac{1}{2} \sigma_y \varepsilon_y$. Generalizando-se para um estado de tensão qualquer:

$$U = \frac{1}{2} \underline{\underline{\sigma}} : \underline{\underline{\varepsilon}} = \frac{1}{2} \sigma_{ij} \varepsilon_{ij} \tag{2.14}$$

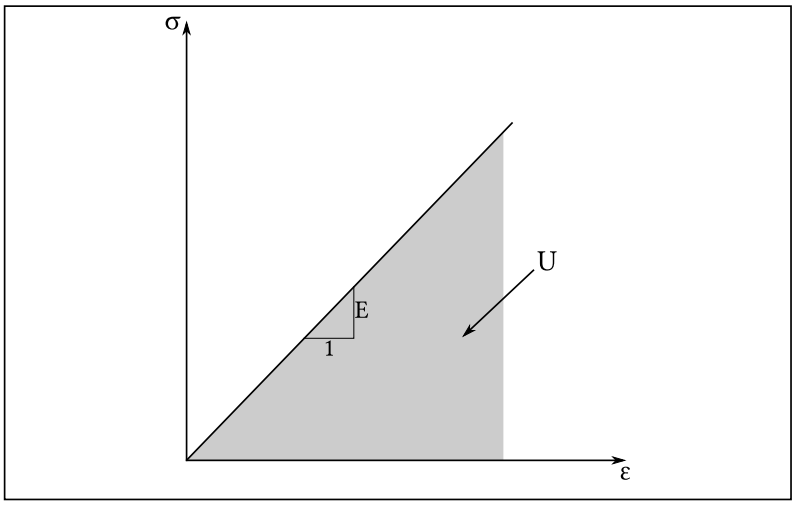

Figura 2.4 Curva de resposta de um estado uniaxial de tensões para um material linear elástico

2.2 Modelos analíticos para previsão do raio crítico de inclusões em compósitos cerâmicos

A diferença entre os coeficientes de expansão térmica da inclusão e da matriz em materiais bifásicos pode induzir microfissurações ou decoesões quando o material é submetido a uma variação de temperatura. Se o coeficiente de expansão térmica da matriz é inferior ao das inclusões, destacamentos podem surgir na interface inclusão / matriz durante o resfriamento [9–17]. Do contrário, para coeficiente de expansão térmica da matriz superior ao das inclusões, microfissuras radiais em relação ao centro da inclusão podem ser nucleadas na matriz devido a tensões trativas nas direções circunferenciais. A nucleação de microfissuras e decoesões ocorre quando a energia total armazenada durante a deformação elástica do material supera sua energia de geração de superfície [9]. Modelos de predição de parâmetros críticos desses materiais devido a tensões térmicas vêm sendo investigados desde os anos 1960 [9, 18].

Selsing [18] calculou as tensões internas em cerâmicas bifásicas considerando as inclusões como esferas. Pautado neste trabalho, Davidge e Green [9] propuseram um modelo analítico que considera as propriedades mecânicas e termomecânicas da matriz e da inclusão, bem como as condições do resfriamento, para o cálculo do raio crítico das inclusões acima do qual ocorrem microfissurações espontâneas. Esses modelos auxiliam na seleção do tamanho das inclusões para a manufatura de componentes, tendo em vista as aplicações do material no qual a presença de microtrincas é desejada ou não.

Os modelos analíticos consideram uma tensão normal média constante ao longo da interface inclusão / matriz [9, 18]. Essa hipótese perde validade para frações volumétricas elevadas, pois a proximidade entre as inclusões pode alterar os campos de tensões em suas vizinhanças [19]. Simulações computacionais podem ser utilizadas na previsão dos campos de deformações e tensões em casos de interação entre inclusões vizinhas. Joliff et al. [20] utilizaram o MEF para investigar a distribuição de tensões entre duas inclusões a partir de um modelo axissimétrico com o intuito de compreender a interação entre as fases. Simula-

ções deste tipo consistem em utilizar a simetria de revolução como uma hipótese simplificadora de cálculo, reduzindo o custo computacional despendido frente a modelos tridimensionais [20].

Modelos analíticos e computacionais proporcionam ferramentas importantes na predição de parâmetros críticos no processamento de cerâmicas refratárias. O MEF já foi aplicado para o estudo de tensões térmicas em compósitos metais / cerâmicos [21, 22] e para compósitos de diferentes materiais cerâmicos [23–26]. Outros trabalhos utilizaram este método para estudo do padrão de fissura induzido por tensões térmicas [27, 28].

Um dos primeiros modelos de previsão de raio crítico foi o desenvolvido por Davidge e Green [9]. Neste é considerada uma única inclusão esférica de raio a, envolta por um volume infinito de matriz. O raio crítico, a_c, é determinado a partir da comparação entre a energia total armazenada durante a deformação, U_t, e a energia de formação de superfícies na matriz, U_s. De acordo com Davidge e Green [9], a energia total armazenada durante a deformação, para esta configuração, pode ser escrita como:

$$U_t = P^2 \, \pi \, a^3 \left[\frac{1 + \nu_m}{E_m} + \frac{2(1 - 2\nu_i)}{E_i} \right] \tag{2.15}$$

sendo:

- P [Pa]: tensão normal média na interface matriz / inclusão;
- E_m, E_i [Pa]: módulos de Young;
- ν_m, ν_i: coeficientes de Poisson;
- a [m]: raio da inclusão.

Os subíndices m e i referem-se à matriz e à inclusão, respectivamente.

Segundo Selsing [18], o valor de P se relaciona com propriedades da matriz e da inclusão pela equação:

$$P = \frac{(\alpha_m - \alpha_i) \, \Delta T}{\dfrac{1 + \nu_m}{2E_m} + \dfrac{1 - 2\nu_i}{E_i}} \tag{2.16}$$

sendo ΔT [°C] a variação da temperatura e α_m e α_i [°C^{-1}] os coeficientes de expansão térmica linear. Neste trabalho de mestrado, será destacado o estudo de fissurações ou decoesões espontâneas durante o resfriamento.

O raio crítico pode ser determinado comparando-se a energia de deformação U_t, Eq. 2.15, com a energia de formação de superfície U_s. Considerando $\alpha_i > \alpha_m$ (decoesão), tem-se que:

$$U_s = 2\,(4\gamma_s \pi a^2) \tag{2.17}$$

sendo γ_s a densidade de energia termodinâmica de formação de superfície da matriz [9] ($J\,m^{-2}$). Considerando a energia total de deformação proporcional a a^3, enquanto a energia de geração de superfície é proporcional a a^2, é possível ser determinado um raio crítico, a_c ao se igualar as energias $U_t(a_c) = U_s(a_c)$:

$$a_c = \frac{8\gamma_s}{P^2\left[\dfrac{1+\nu_m}{E_m} + \dfrac{2(1-2\nu_i)}{E_i}\right]} \tag{2.18}$$

Em cerâmicas com frações volumétricas baixas, a distância média entre inclusões é suficientemente alta para assumir que os campos de deformação e tensão aproximam-se da hipótese de uma inclusão única envolta por volume infinito [9]. Com o aumento da fração volumétrica, para um dado valor de a, as inclusões passam a se dispor mais próximas uma das outras, de modo que os campos de tensões e deformações de inclusões vizinhas interagem mutuamente. Neste caso, a hipótese de tensão normal média constante ao longo da interface entre matriz e inclusão perde validade [19]. Visando investigar a interação entre inclusões e a influência da fração volumétrica na previsão do raio crítico, Liu e Winn [19] propuseram um modelo analítico com duas inclusões e assumiram que os efeitos entre inclusões é calculado como uma combinação linear de efeitos no eixo que cruza o centro de ambas as inclusões (Figura 2.5). A fração volumétrica de inclusões é expressa por $\phi = a^3/b^3$ e o estado de tensão em um ponto genérico P é oriundo da combinação linear de efeitos das inclusões com distância r e $2b - r$ deste ponto. Liu e Winn [19] investigaram a distribuição de

tensões radiais e circunferenciais entre duas inclusões e concluíram que para $b < 2a$ ($\phi > 12.5\ vol.\%$), tem-se alterações significativas nas tensões radiais. Portanto, a hipótese de inclusão isolada envolta em uma matriz de volume infinito perde representatividade com o aumento da fração volumétrica.

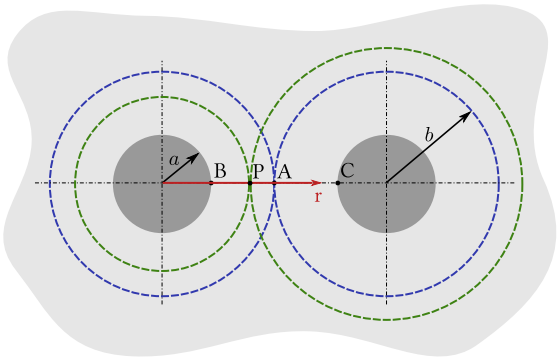

Figura 2.5 Modelo de duas inclusões (de raio a) envoltas por matriz finita (de raio b) interseccionadas em um ponto (A), baseado em Liu & Winn [19]

Para avaliar a distribuição de tensões radiais e circunferenciais entre duas inclusões, Liu e Winn [19] utilizaram os coeficientes de tensão ($\bar{\sigma_\rho}$ e $\bar{\sigma_\theta}$) como parâmetros adimensionais para estudar a distribuição das tensões na matriz. Os coeficientes de tensão são determinados pela razão entre a tensão, radial ou circunferencial, num ponto material $P \in \overline{BC}$ (Figura 2.5), $\sigma(r)$, e a tensão (radial ou circunferencial) na interface inclusão / matriz, $\sigma(a)$. O coeficiente de tensão radial, $\bar{\sigma_\rho}$, e o coeficiente de tensão circunferencial, $\bar{\sigma_\theta}$, podem ser expressos como:

$$\bar{\sigma_\rho} = \frac{\sigma_\rho(r)}{\sigma_\rho(a)} = \left[\left(\frac{a}{r}\right)^3 - \left(\frac{a}{2b-r}\right)^3\right]\frac{1}{1-\phi} \tag{2.19}$$

$$\bar{\sigma_\theta} = \frac{\sigma_\theta(r)}{\sigma_\theta(a)} = \left[\frac{1}{\phi}\left(\frac{a}{r}\right)^3 + \left(\frac{a}{2b-r}\right)^3 + 4\right]\frac{1}{2+(1-\phi)} \tag{2.20}$$

sendo, ϕ a fração volumétrica, os índices ρ e θ referentes às direções radial e circunferencial, respectivamente, e r a posição do ponto P no seguimento $\overline{BC}$, $r \in [a, 2(b-a)]$.

Liu e Winn desenvolveram uma expressão para o cálculo de raio crítico de

inclusões que considera a interação entre inclusões vizinhas:

$$a_c = \frac{3\gamma}{\left[\left({F_\rho}^2 + 2{F_\theta}^2\right)\right] - 2\nu_m F_\theta \left(2F_\rho + F_\theta\right) \frac{2{\sigma_\alpha}^2}{E_m} + \frac{{\sigma_\alpha}^2(1-2\nu_i)^2}{E_i}} \tag{2.21}$$

sendo,

$$F_\rho = \frac{1 - \left(2\phi^{-1/3} - 1\right)^{-2}}{(1-\phi)\left(\phi^{-1/3} - 1\right)} \tag{2.22}$$

$$F_\theta = \frac{1 + \left(2\phi^{-1/3} - 1\right)^{-2}}{4(1-\phi)\left(\phi^{-1/3} - 1\right)} \tag{2.23}$$

$$\sigma_\alpha = \frac{-(\alpha_i - \alpha_m)\Delta T}{\frac{2\phi(1-2\nu_m) + (1+\nu_m)}{2E_m(1-\phi)} + \frac{(1-2\nu_i)}{E_i}} \tag{2.24}$$

σ_α é a tensão normal média na interface matriz-inclusão.

2.3 Trincas inter-inclusões

Modelos numéricos, tanto analíticos [9, 18, 19] quanto computacionais [29], ainda não explicaram satisfatoriamente o surgimento de trincas inter-inclusões durante o resfriamento de compósitos cerâmicos (Figura 2.6-a). Nota-se que a trinca localiza-se aproximadamente no ponto médio entre as inclusões e possui um padrão retilíneo. No trabalho de Liu e Winn [19], foi desenvolvido um modelo analítico para a previsão de distribuição de tensões radiais e circunferenciais entre as inclusões e os autores concluíram que para as frações volumétricas analisadas ($\phi \leq 30\, vol.\%$), a interface matriz-inclusão seria sempre a região mecanicamente mais solicitada durante o resfriamento, não elucidando, desta maneira, o surgimento de trincas inter-inclusões. Joliff et al. [20, 29] analisaram, por meio de simulação computacional utilizando MEF, a distribuição de tensões entre duas inclusões de alumina de mesma dimensão dispersas em matriz vítrea para diferentes frações volumétricas. Afirmaram que seu aumento, ou seja, a aproximação entre as inclusões, não causaria efeito no formato da distribuição de tensões a ponto de explicar o surgimento das trincas inter-inclusões. Desta

maneira, sugeriram que a origem das trincas inter-inclusões não está associada diretamente à distância entre elas, mas sim a defeitos oriundos do processamento. Como a distribuição de pressão durante a compactação de materiais heterogêneos não é uniforme, as regiões inter-inclusões estariam submetidas à menor compactação e, consequentemente, as propriedades mecânicas estariam comprometidas. É importante ressaltar que Jollif et al. [20, 29] limitaram suas simulações para a razão entre distância mínima entre inclusões (ℓ) e o diâmetro de inclusões (d) em $\ell/d = 1/3$, e apenas consideraram tensões radiais.

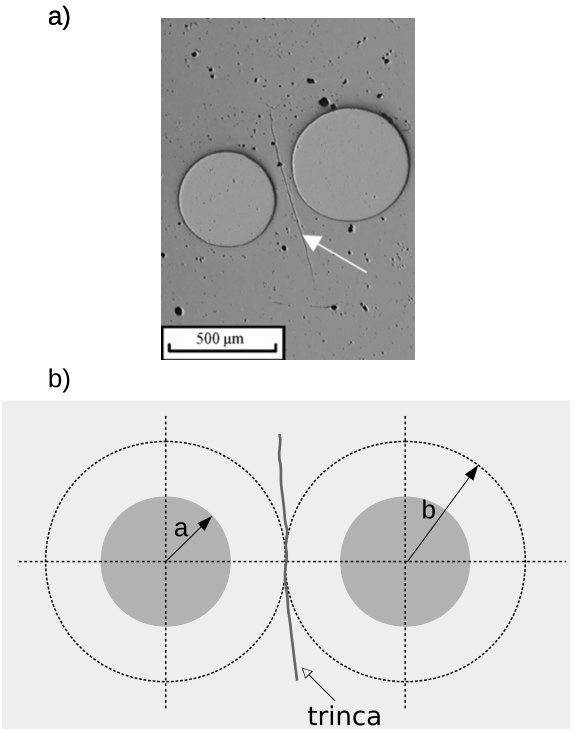

Figura 2.6 a) Microfissura inter-inclusão, Joliff et al. [20] b) Modelo esquemático para o cálculo de fração volumétrica local, em que a é o raio das inclusões e b o raio da esfera inscrita em um dodecaédro rômbico

Com o intuito de relacionar a fração volumétrica de inclusões com a distância média entre elas serão utilizados conceitos discutidos por Rosenfeld [30]. O mesmo estabelece uma relação entre distância e fração volumétrica das inclusões para um empacotamento CFC (cúbica de face centrada), considerando as inclusões fixadas no centro de células dodecaédricas rômbicas em um *honeycomb*. Assim, a relação entre ϕ, a e o raio da esfera inscrita na célula dodecaé-

drica rômbica (b) (Figura 2.6-b) é dada por:

$$b = 0.905\,\phi^{-1/3}\,a \tag{2.25}$$

2.4 Método dos elementos finitos

O método dos elementos finitos é uma técnica numérica para cálculo aproximado de soluções para problemas de valores de contorno. Este método consiste na discretização de um problema relativamente grande em partes menores e mais simples, chamadas de elementos finitos. As equações simplificadas de cada elemento são montadas em um grande sistema de equações que modela o problema todo. O MEF se utiliza de métodos variacionais para encontrar a solução aproximada, minimizando uma função erro associada [31].

Uma vez que os elementos possam ser interconectados de incontáveis maneiras diferentes, permite-se ao usuário do MEF modelar formas geométricas complexas, além de aumentar a possibilidade de aplicação de condições de contorno e carregamentos. O processo de análise de elementos finitos no contexto do desenvolvimento de projeto proposto por Bathe [32] é esquematizado na Figura 2.7.

O desenvolvimento moderno do método dos elementos finitos deu-se nos anos 1940, a partir do trabalho de Hrennik [33] e McHenry [34], que aplicaram o conceito de elementos unidimensionais (barras e vigas) para a solução do cálculo de tensões em meios contínuos.

A maior parte dos trabalhos em elementos finitos até a década de 1960 lidava com pequenas deformações, materiais lineares elásticos e carregamentos estáticos. A partir da década de 1960 começou-se a tratar problemas térmicos, dinâmicos e de flambagem [35–38].

Atualmente o MEF apresenta grande apelo industrial, principalmente devido ao surgimento de códigos comerciais, mais acessíveis e completos. Problemas multifísicos são mais acessíveis de serem tratados, fornecendo aos usuários uma ferramenta de análise poderosa.

Dentre as recentes implementações em códigos comerciais de EF encontram-

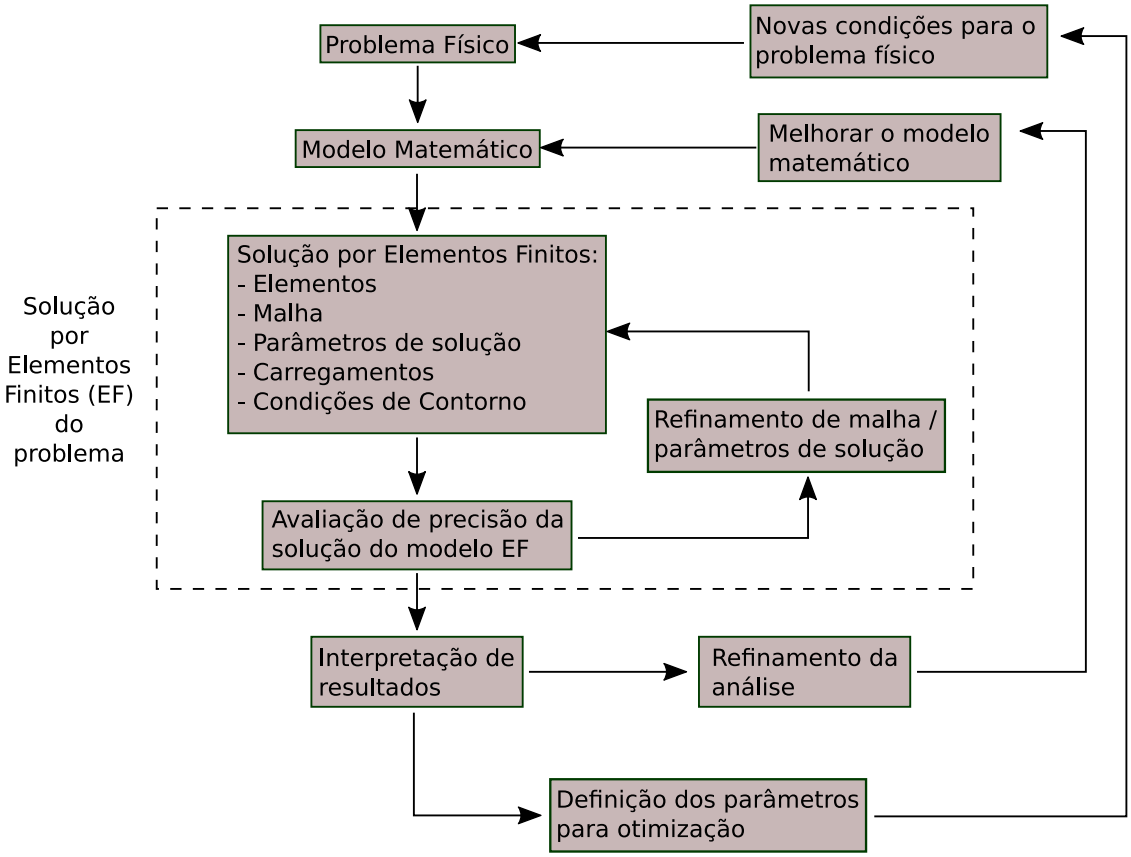

Figura 2.7 Processo de análise por elementos finitos. (Adaptado de Bathe [32])

se os elementos coesivos. Esses foram desenvolvidos para modelar o comportamento de juntas adesivas, interfaces em compósitos e outras situações em que a integridade e a resistência de interfaces são os interesses [39]. Um problema fundamental na simulação de mecanismos de falhas coesivas está na definição das interações coesivas entre as superfícies de fratura. Interações coesivas aproximam o comportamento de fratura progressiva não linear (Figura 2.8).

Interações coesivas são geralmente representadas por uma função de deslocamento (ou separação). Se o deslocamento é maior do que um comprimento característico (δ_n), ocorre danificação das propriedades do elemento. É importante considerar que modelos de zonas coesivas não se limitam a simular a propagação de uma única trinca, também sendo úteis para a previsão do local de nucleação destas, assim como para representar as interações entre as faces da trinca formada.

Segundo o manual de usuários do software $ABAQUS^{TM}$ [39], para a mode-

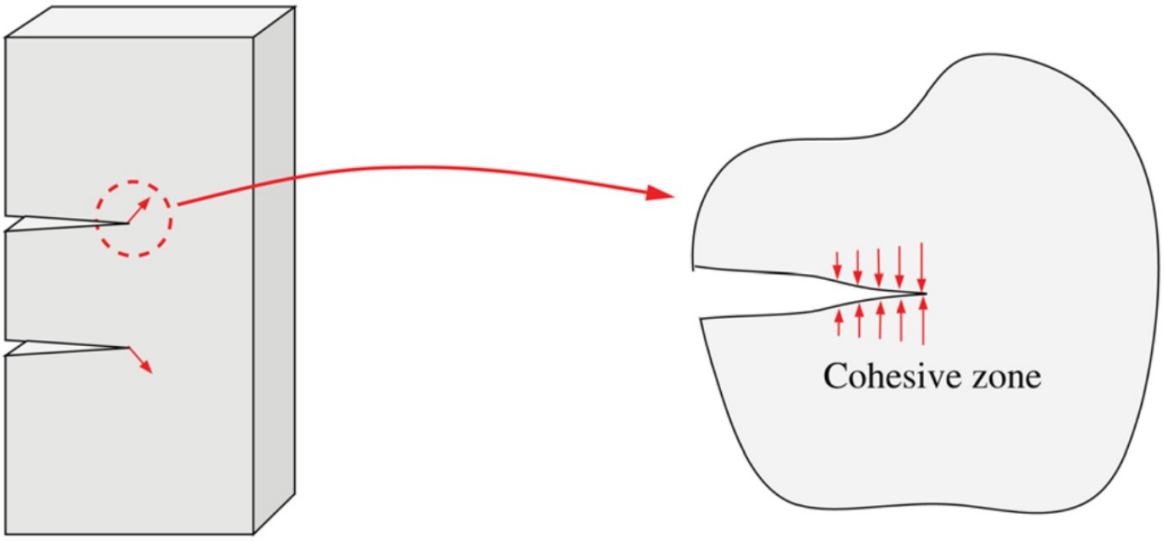

Figura 2.8 Desenho esquemático - zona coesiva. Adaptado de Park & Paulino [40].

lagem de adesivos com uma espessura considerável, deve-se optar pela modelagem do elemento coesivo como um elemento contínuo, e suas propriedades são macroscópicas. Já para a simulação de camadas de adesivos com espessuras muito finas, ou seja, muito menores que as dimensões da superfície de aplicação do adesivo, deve-se optar por modelos constitutivos em termos de tração *versus* movimento relativo da interface, mais conhecido com o termo em inglês *traction-separation* (TS). Pequenos adesivos, tanto em espessura quanto em comprimento, são muito bem representados por elementos 1D.

A ligação entre os elementos coesivos e os elementos contínuos da malha pode ser feita de três maneiras diferentes [39]. A primeira é a chamada de “malha casada”, em que os elementos coesivos e seus vizinhos compartilham os nós (Fig. 2.9). Ela é indicada para resultados iniciais, fornecendo um comportamento global da estrutura.

Outra maneira de se conectar a zona coesiva com os elementos estruturais da malha é por meio de restrições de movimento (Fig. 2.10). Este caso é indicado para a modelagem de propagação de fissuras ou decoesões quando um resultado local mais preciso faz-se necessário, uma vez que a zona coesiva pode possuir um refinamento de malha conforme o nível da precisão que se deseja.

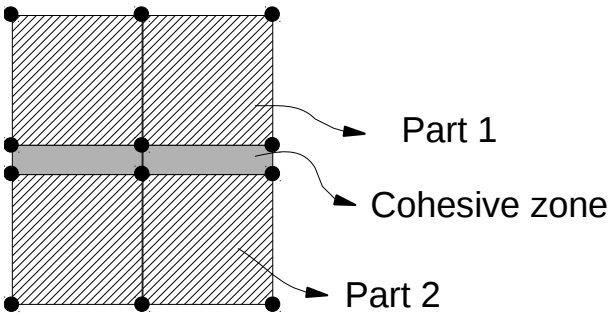

Figura 2.9 Esquema representativo de uma zona coesiva com “malha casada”. Adaptado do manual do software ABAQUS [39].

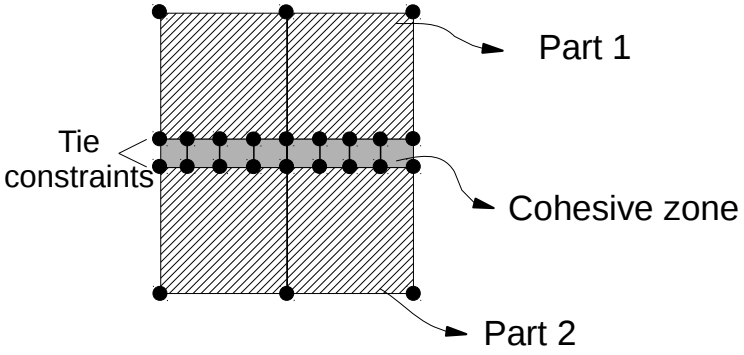

Figura 2.10 Esquema representativo de uma zona coesiva com “malha restrita”. Adaptado do manual do software ABAQUS [39].

A última possibilidade de junção entre a zona coesiva e a malha estrutural consiste na utilização de restrições em um lado da zona coesiva e contato no outro (Fig. 2.11). Este método é mais destinado à simulação de juntas e vedações, quando se tem interesse em estudar vazamentos e destacamentos.

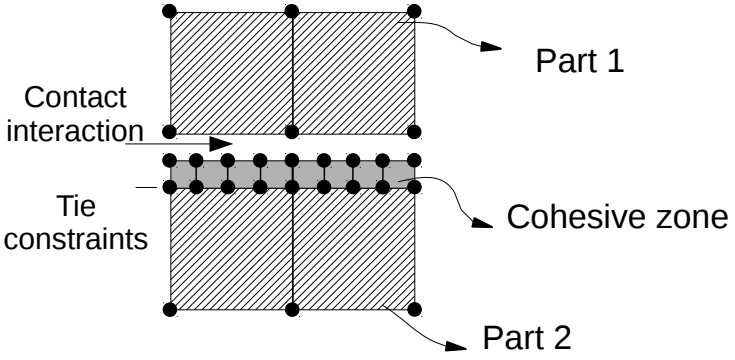

Figura 2.11 Esquema representativo de uma zona coesiva com “malha 1/2 restrita 1/2 contato”. Adaptado do manual do software ABAQUS [39].

Elementos coesivos já foram aplicados anteriormente para a modelagem de fratura em cerâmicas [27,41–43]. No mesmo contexto, Fertig & Nickerson [44] introduziram um modelo analítico mecanístico para a previsão do comportamento do módulo elástico de sistemas cerâmicos submetidos à variação de temperatura. Este modelo serviu para a predição da iniciação e fechamento de microtrincas termicamente induzidas nos contornos de grãos. Neste cenário, entende-se que fechamento de trincas não implica necessariamente em cicatrização – uma trinca fechada é uma trinca que não reduz totalmente o módulo de elasticidade devido a efeitos de atrito nas interfaces, por exemplo.

A modelagem de Fertig e Nickerson foi dividida em duas etapas. Primeiramente, uma versão simplificada de um modelo EF 2D foi estudada para entender qualitativamente as tendências, formas funcionais, sensibilidade de parâmetros e para estabelecer uma referência visual dos mecanismos atuantes na fissuração dos sistemas. Entende-se que estes modelos em EF não são adequados para se extrapolar resultados macroscópicos, sendo portanto, utilizados como guias para a modelagem analítica.

Em seu modelo de EF, Fertig & Nickerson [44] simularam um domínio bidimensional composto de 100 grãos hexagonais (*honeycomb*) de cordierita no software comercial ABAQUS^{TM}. Cada grão possuía uma orientação de material aleatória entre os ângulos -90° e 90° em relação a uma linha vertical. As interfaces entre os grãos apresentavam uma camada de elementos coesivos (Figura 2.12).

Em seu trabalho, Fertig & Nickerson associam a deformação dos elementos coesivos com a propagação da trinca, ou seja, como em materiais cerâmicos os efeitos de rastro nas trincas são muito importantes, ele é representado nas simulações pela degradação das propriedades elásticas dos elementos coesivos. Após a degradação, o material naquela região não possui as propriedades mecânicas do material virgem (lei TS), porém a deleção do elemento não é imediata, como num sistema binário. Pode-se associar esta resistência residual com os efeitos que ainda mantém certa resistência à abertura da trinca, como atrito entre grãos ou até mesmo efeitos de tenacificação encontrado em alguns compó-

sitos cerâmicos. É claro que a total degradação do elemento implica que aquela região não apresenta resistência alguma à abertura da trinca.

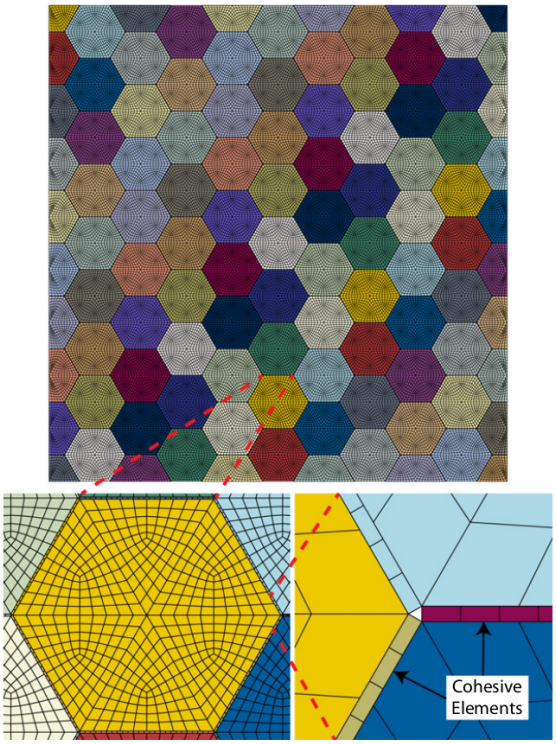

Figura 2.12 Modelo EF extraído de Fertig & Nickerson [44]

A lei tração-separação (TS) é formulada em função de deslocamentos nodais:

$$\{\tau\} = [K]\{\Delta\} \tag{2.26}$$

em que $\{\tau\} = \{\tau_n\, \tau_s\}^T$ é o vetor de tração (n e s correspondem às direções normal e cisalhantes, respectivamente). A variável $\{\Delta\} = \{\Delta_n\, \Delta_s\}^T$ é o deslocamento nodal relativo e a matriz de rigidez é dada por:

$$[K] = \begin{bmatrix} K_n & 0 \\ 0 & K_s \end{bmatrix} \tag{2.27}$$

Assume-se que a falha dos elementos coesivos são caracterizadas pela degradação de sua rigidez, o que permite múltiplos mecanismos de danificação.

Na Figura 2.13 apresenta-se o comportamento TS bilinear de um elemento coesivo. Nota-se que quando há um deslocamento crítico (δ_c), inicia-se a danifi-

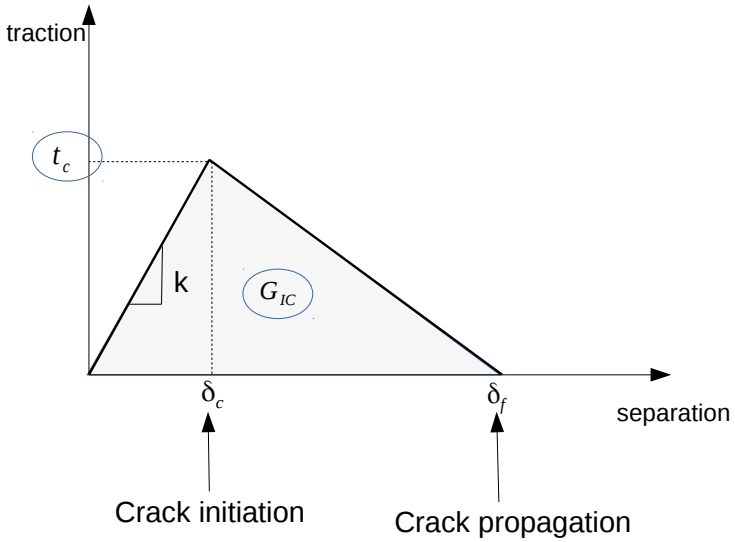

Figura 2.13 Comportamento TS de um elemento coesivo genérico. Adaptado do manual do software ABAQUS [39].

cação do material, havendo a total deleção do elemento em δ_f. Esta figura pode ser associada às três condições possíveis de solicitação (normal e cisalhantes). Logo, o critério de início da danificação do elemento deve ser informado pelo usuário.

Considerando-se que t_n é a tensão normal; t_s e t_t são as tensões cisalhantes; ε_n é a deformação normal; ε_s e ε_t são as deformações cisalhantes; o superíndice (^{o}) refere-se a tensão (t^o) ou deformação (ε^o) críticas; os critérios podem ser:

- Critério de máxima tensão nominal (MAXS), em que uma vez que alguma tensão tenha atingido a tensão crítica, inicia-se a danificação. Este critério foi o utilizado para os modelos tratados nesta dissertação.

$$max\left\{\frac{t_n}{{t^o}_n}; \frac{t_s}{{t^o}_s}; \frac{t_t}{{t^o}_t}\right\} = 1 \tag{2.28}$$

- Critério de máxima deformação nominal (MAXE), em que uma vez que alguma deformação tenha atingido a deformação crítica, inicia-se a danificação, i.e:

$$max\left\{\frac{\varepsilon_n}{{\varepsilon^o}_n}; \frac{\varepsilon_s}{{\varepsilon^o}_s}; \frac{\varepsilon_t}{{\varepsilon^o}_t}\right\} = 1 \tag{2.29}$$

- Critério da tensão nominal quadrática (QUADS), em que faz-se necessário que todas as tensões estejam no nível crítico para iniciar-se o dano. Mais utilizado para danos globais, quando não se possui conhecimento prévio da resposta da estrutura, i.e:

$$\left\{\frac{t_n}{{t_n}^o}\right\}^2 \cdot \left\{\frac{t_s}{{t_s}^o}\right\}^2 \cdot \left\{\frac{t_t}{{t_t}^o}\right\}^2 = 1 \tag{2.30}$$

- Critério da deformação nominal quadrática (QUADE), em que faz-se necessário que todas as deformações estejam no nível crítico para iniciar-se o dano. Mais utilizado para danos globais, quando não se possui conhecimento prévio da resposta da estrutura, i.e:

$$\left\{\frac{\varepsilon_n}{{\varepsilon_n}^o}\right\}^2 \cdot \left\{\frac{\varepsilon_s}{{\varepsilon_s}^o}\right\}^2 \cdot \left\{\frac{\varepsilon_t}{{\varepsilon_t}^o}\right\}^2 = 1 \tag{2.31}$$

2.4.1 Materiais compósitos

Materiais compósitos ou compostos são oriundos da combinação de dois ou mais materiais com propriedades diferentes e utilizados em determinadas aplicações que quando isolados, os materiais constituintes não teriam significativo êxito.

Materiais compósitos muito difundidos em engenharia são:

- concretos
- plásticos reforçados
- compósitos metálicos
- compósitos com fibras de carbono
- compósitos com fibras de vidro
- compósitos cerâmicos

Historicamente os materiais compósitos acompanham a evolução das sociedades. Há 6000 anos atrás, os egípcios retratam a utilização de argila e fibras

naturais na confecção de tijolos. Em 3400 A.C. os povos mesopotâmicos relatam como manufaturar materiais compósitos oriundos do empilhamento de madeira em diferentes orientações com fins estruturais.

Atualmente os materiais compósitos possuem forte apelo inovativo, uma vez que a detenção deste conhecimento pode ser crucial no desenvolvimento de tecnologias aero-espaciais e de defesa, assuntos chaves na autonomia e manutenção de influência dos países.

Concreto é o compósito artificial mais difundido, ele consiste da composição de pedras (agregados) ligadas à uma matriz de cimento. Concretos resistem bem a carregamentos compressivos, embora sejam relativamente frágeis a carregamentos trativos. Para superar essa limitação são utilizadas barras pré-tensionadas de aço, pré-tensionando compressivamente o concreto. Este concreto pré-tensionado com barras de aço apresenta o nome comercial de concreto armado. Muito utilizado em pontes e vigas.

Os agregados apresentam duas principais funções nos concretos, primeiramente eles aumentam o módulo de elasticidade global da estrutura, deixando-a mais rígida. Além disso, eles possuem um papel tenacificador. Ou seja, ele altera o comportamento da propagação de trincas no meio, fazendo com que as mesmas tenham que desviar dos agregados, consumindo mais energia para uma propagação catastrófica.

Polímeros reforçados com fibras incluem polímeros reforçados com fibra de carbono e fibra de vidro. Se estes forem classificados pela matriz, são considerados compósitos termoplásticos. Dos quais podem ser separados em termoplásticos reforçados com fibras curtas ou com fibras longas.

Estes materiais são amplamente aplicados na indústria aero-espacial, automotiva e de defesa. Uma vez que estes materiais possuem a característica de uma ótima relação entre resistência e massa da peça. O que para estas indústrias implica em economia de combustível e um aumento de carga útil que poderá ser transportada.

Os compósitos cerâmicos tratados nesta dissertação são oriundos da dispersão de inclusões e matriz contínua. Tanto inclusões como a matriz são materiais

cerâmicos. Este tipo de compósito ganhou grande destaque na era pós-Sputnik (1957). Uma vez que com a corrida espacial, buscavam-se materiais que mantivessem forma e função na reentrada de espaçonaves à atmosfera terrestre, com temperaturas ultrapassando os 1500°C e que fossem leves o suficiente para houvesse uma maximização da carga útil a ser transportada.

Buscava-se também, uma maneira de contornar a elevada fragilidade dos materiais cerâmicos à temperatura ambiente. O que, muitas vezes, inviabilizava a utilização desta classe de materiais em certas aplicações em que uma falha catastrófica não pudesse ocorrer.

Neste caso, o papel das inclusões consistia em interagir com a trinca, dificultando sua propagação. Assim, para que esta se desenvolvesse a um tamanho crítico, seria necessário um maior consumo de energia. Essa interação com a trinca pode se dar de diversas maneiras, tanto deixando uma tensão residual compressiva nas peças, quanto servindo de obstáculo para a propagação da trinca, fazendo com que esta tenha que desviar das inclusões e consumir mais energia. Além de determinados casos em que há uma reação expansiva das inclusões, deixando uma tensão residual compressiva nas pontas de trincas.

3 MATERIAIS E MÉTODOS

Para facilitar a estruturação desta dissertação, optou-se por dividir o trabalho completo em 4 estudos principais. Esta separação segue uma ordem crescente de complexidade.

3.1 Estudo 1: Modelo computacional para a determinação do raio crítico de inclusões

O estudo 1 apresentou por objetivo principal a determinação do raio crítico de inclusões a partir de simulações computacionais em elementos finitos. Destacam-se as vantagens da simulação computacional frente aos modelos analíticos presentes na literatura, bem como a compreensão da extensão da aplicação do modelo de raio crítico de Davidge e Green [9] e Liu e Winn [19].

3.1.1 Materiais

Buscou-se na literatura exemplos em que materiais cerâmicos compósitos apresentaram fissurações durante o resfriamento devido à diferença dos coeficientes de expansão térmica. Sendo assim, foram escolhidos três exemplos presentes nos trabalhos de:

1. **Davidge e Green [9]**, no qual foi obtido o raio crítico para um sistema com inclusões esféricas de tória envoltas por matriz vítrea (Corning 7740 Pyrex). Os autores realizaram experimentos em amostras considerando o valor da variável a (raio da inclusão) entre 45 e 710 μm e $\phi = 10\, vol.\%$;

2. **Todd e Derby [45]**, no qual foi estudado um sistema de inclusões de carbeto de silício em uma matriz de alumina, semelhante aos experimentos de Davidge & Green [9], porém com inclusões não esféricas e $\phi = 20\ vol.\%$;

3. **Joliff et. al [20]**, em cujo trabalho foram estudados compósitos com inclusões esféricas de alumina envoltas por matriz vítrea de borossilicato, considerando $\phi = 15$, 30 e $45\ vol.\%$ e $a = 250\ \mu$m. Como no trabalho de Joliff et al. [20] não haviam amostras com raios de inclusões distintos, mas sim

três frações volumétricas (ϕ), identificou-se o ΔT em que as microfissurações eram iniciadas em cada amostra a partir da evolução do módulo de Young com a temperatura (Figura 3.1) extraído de [46]. A partir das curvas experimentais de módulo de Young em função da temperatura durante o resfriamento, aproximou-se o comportamento do compósito por três retas ajustadas a partir dos dados experimentais. Assim, o ΔT foi identificado pelo segmento resultante entre as retas que definem o ponto de enrijecimento (2) e aquela que representa o início de fratura (1). Para estes valores de ΔT e ϕ, o raio de $250\ \mu m$ é o raio de inclusão crítico.

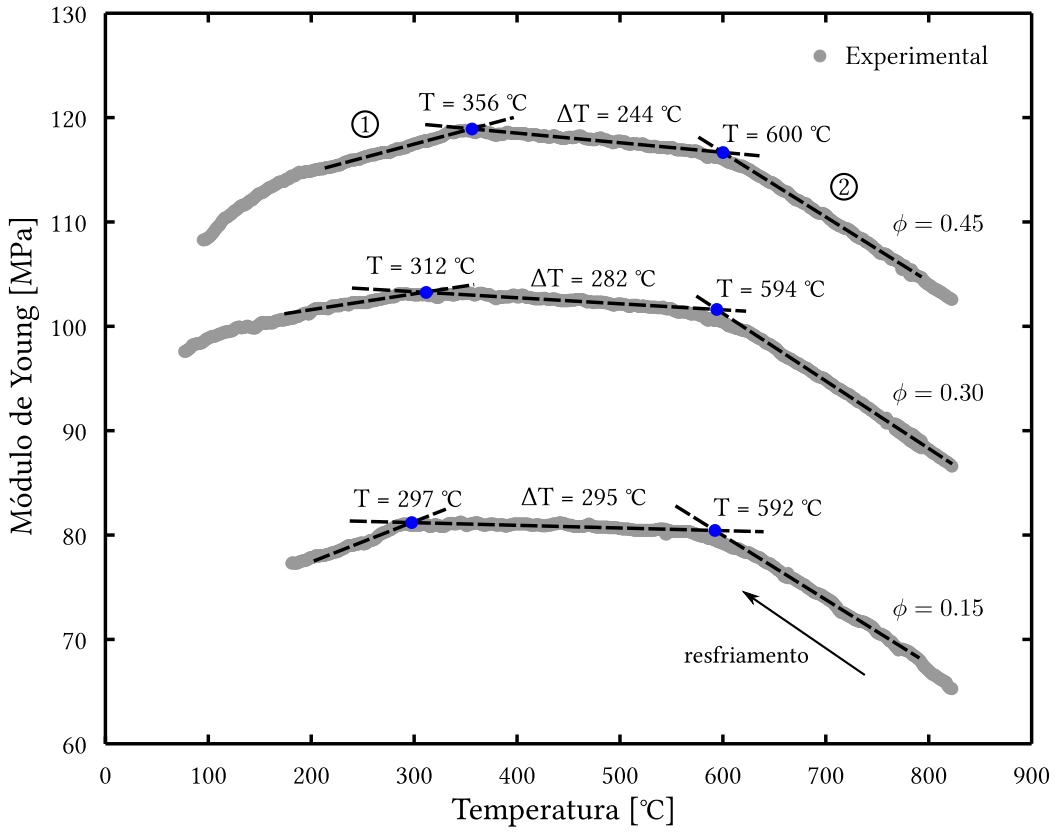

Figura 3.1 Procedimento para obtenção das variações de temperatura a partir dos dados experimentais de Tessier-Doyen et al. [46].

As propriedades mecânicas e termomecânicas destes materiais, bem como as condições de resfriamento dos experimentos, são descritas na Tabela 3.1 [9, 20, 45].

Tabela 3.1 Parâmetros usados nos modelos MEF e para previsão do raio crítico de inclusões.

Ref.	ϕ	γ_s	ΔT	**Matriz**			**Inclusões**		
				α_m	E_m	ν_m	α_i	E_i	ν_i
	[vol. %]	[J m^{-2}]	[$^\circ$ C]	[$10^{-6}\,^\circ$C^{-1}]	[GPa]		[$10^{-6}\,^\circ$C^{-1}]	[GPa]	
[9]	10	4	545	3.6	70	0.20	8.7	250	0.27
			500	5.4					
[45]	20	15	800	8.9	402	0.23	4.4	483	0.17
[20]	15	4	295	4.6	68	0.20	7.6	340	0.24
	30	4	282						
	45	4	244						

Obs.: na referência [9] as inclusões eram de tória em matriz vítrea, em [45] as inclusões eram de carbeto de silício em matriz de alumina, enquanto em [20] as inclusões eram de alumina em matriz vítrea.

3.1.2 Métodos

A fim de simular as condições dos experimentos apresentados na Tabela 3.1, foram desenvolvidos três modelos em elementos finitos (Figura 3.2). Os modelos são axissimétricos com malha de elementos finitos lineares e integração reduzida (CAX4R), e foram desenvolvidos no software Abaqus®.

- **Modelo M1** (Figura 3.2a): este modelo considera uma inclusão isolada em uma matriz "infinita", para que se possa reproduzir em MEF o modelo analítico de Davidge e Green [9]. As dimensões da matriz são uma ordem de grandeza superior àquelas do raio da inclusão, *i.e.* R e $L > 10a$, valor calculado a partir de testes com simulações;

- **Modelo M3** (Figura 3.2b): este modelo considera três inclusões posicionadas no eixo de axissimetria. A distância entre os centros das inclusões, $2b$,

é função da fração volumétrica, a qual é calculada ($b = a \cdot \phi^{-1/3}$) assumindo-se duas esferas concêntricas. A presença das inclusões adjacentes à central permite considerar a influência da proximidade das inclusões vizinhas;

- **Modelo Mn** (Figura 3.2c): este modelo considera uma condição de periodicidade, ou seja, infinitas inclusões alinhadas no eixo de axissimetria. A distância entre inclusões é determinada de forma análoga ao modelo anterior (M3). A condição de periodicidade impõe que o deslocamento na direção z de todos os nós pertencentes à face superior, z^+, possuem valores iguais. De forma análoga, tem-se que todos os nós da face inferior, z^-, têm o mesmo deslocamento na direção z.

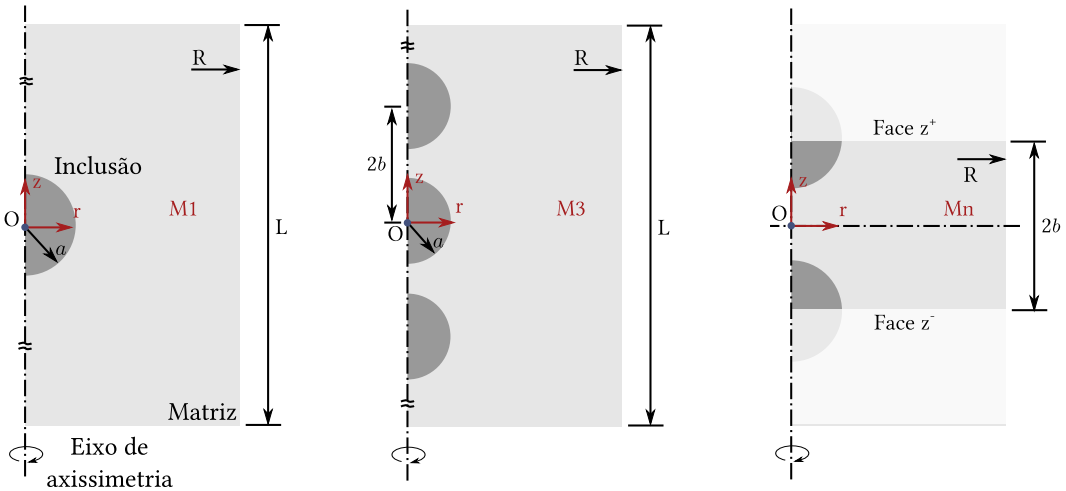

Figura 3.2 Modelo axissimétrico de inclusões envolta por uma matriz.
(a) Modelo M1: única inclusão; (b) Modelo M3: três inclusões; (c) Modelo Mn: modelo com condição de periodicidade.

Nos três modelos, o conjunto matriz / inclusão foi submetido a um resfriamento com variação térmica segundo os ensaios experimentais para cada caso apresentado na Tabela 3.1. Assim, inicialmente trata-se de um problema de análise de tensões residuais oriundas de um processo de resfriamento, o qual proporciona a contração do sistema. Considerou-se que todos os pontos materiais do modelo estão em uma mesma temperatura (regime permanente), isto é, não são considerados efeitos de gradientes térmicos devido a mecanismos de

condução, convecção ou radiação. Os deslocamentos nas direções r e z são restritos no ponto O (Figura 3.2).

O raio crítico foi obtido comparando-se a energia de deformação por inclusão - calculadas via simulações computacionais - com a energia de formação de superfície para o caso de decoesão entre matriz e inclusão. Visto que a energia total de deformação é função do raio da inclusão, $U_t(a)$, bem como a energia de geração de superfície, $U_s(a)$, o problema de determinação do raio crítico resume-se à solução da equação não-linear: $f(a) = U_s(a) - U_t(a) = 0$. O Método de Newton-Raphson foi aplicado na solução do problema, sendo que, o raio da iteração (k+1), especificado como $a_{(k+1)}$, pode ser obtido a partir do raio da iteração (k), e seu respectivo $a_{(k)}$, como:

$$a_{(k+1)} = a_{(k)} - \frac{f(a_{(k)})}{f'(a_{(k)})} \tag{3.1}$$

Em que a derivada $f'(a_{(k)})$ é expressa como:

$$f'(a_{(k)}) = U'_s(a_{(k)}) - U'_t(a_{(k)}) = 16\pi\gamma_s a_{(k)} - U'_t(a_{(k)}) \tag{3.2}$$

A derivada da energia total de deformação foi calculada numericamente com uma aproximação de primeira ordem, *i.e.*:

$$U'_t(a_{(k)}) \approx \frac{U_t(a_{(k)} + \delta a) - U_t(a_{(k)})}{\delta a} \tag{3.3}$$

Uma perturbação $\delta a = a_{(k)} \cdot 10^{-5}$ foi utilizada durante o processo, o qual é interrompido quando $|f(a_{(k)})| < \epsilon = 10^{-3}$. O código em linguagem Matlab™ desta seção encontra-se no Apêndice A.

Além da identificação do raio crítico, o modelo em EF permite comparar a distribuição de tensões com aquela obtida pelo modelo de duas inclusões, proposto por Liu e Winn [19] (Figura 2.5).

3.2 Estudo 2: Modelo computacional para estudo da origem das trincas inter-inclusões

A partir dos resultados do estudo 1, detalhados na Seção 4.1 (pág. 53), houve a necessidade de se investigar via simulação computacional a origem de trincas inter-inclusões. Desta maneira, o objetivo principal do estudo 2 foi buscar subsídios teóricos que expliquem o surgimento deste tipo de trinca.

3.2.1 Materiais

Nestas simulações foram considerados os mesmos compósitos com matriz vítrea e inclusões de alumina. As propriedades mecânicas da alumina e da matriz vítrea foram extraídas de Joliff et al. [20] e encontram-se na Tabela 3.1, pág. 31.

3.2.2 Métodos

Para investigar a origem das trincas inter-inclusões foram desenvolvidos modelos axissimétricos em elementos finitos utilizando-se o software Abaqus®. Os modelos possuem malha com elementos de 4 nós, função de forma linear e integração reduzida (elemento CAX4R da biblioteca do Abaqus®).

Para todos os casos, simulou-se o resfriamento ($\Delta T = 300\,^{\circ}C$) de uma amostra com duas inclusões de alumina (modelo M2) com raio $d/2 = a = 250\,\mu$m dispersas em matriz vítrea, separadas por uma distância ℓ variável e com restrições de deslocamentos nas direções r e z no ponto O (Figura 3.3).

Para a determinação das tensões térmicas oriundas da diferença de contrações das fases após o resfriamento, considerou-se que o sistema matriz-inclusões estivesse em equilíbrio, ou seja, não são considerados efeitos de condução, convecção ou radiação térmica, conforme explicado no modelo de elementos finitos anterior. Todo o conjunto foi submetido à uma variação de temperatura ΔT e as tensões térmicas foram calculadas a partir das diferenças de contrações entre a matriz e as inclusões. Além disso, a interface entre elas foi considerada perfeita e o comportamento dos materiais como linear elástico.

Pautado nos trabalhos de Hasselman [47] e de Davidge e Green [9], nos quais

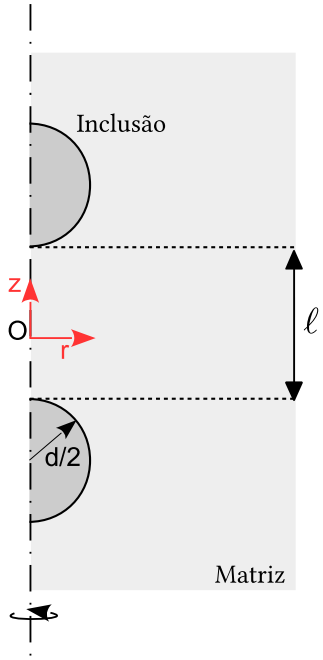

Figura 3.3 Modelo M2 axissimétrico de inclusões envoltas por matriz

a condição para a nucleação de trinca considera que a energia armazenada durante a deformação elástica do compósito supere sua energia de formação de superfícies, calculou-se a contribuição de cada elemento de malha para armazenagem total de energia elástica no material. Deve-se frisar que para as simulações apresentadas foram tomadas precauções para que as dimensões dos elementos fossem mantidas constantes em todos os modelos, ou seja, anulando-se a influência da malha de elementos finitos neste resultado.

O cálculo da energia elástica armazenada ao final do resfriamento (U_t), para cada elemento, foi calculado a partir da Eq. 2.14 (Pág. 11).

3.3 Estudo 3: Estudo da influência das propriedades termomecânicas e fração volumétrica no perfil de distribuição de energia elástica armazenada na região entre as inclusões e no nível de energia total armazenado no compósito

Os objetivos deste estudo podem ser divididos em dois principais aspectos, sendo o primeiro avaliar a influência das propriedades termomecânicas das distintas fases (matriz e inclusão) de um sistema cerâmico na localização da região energeticamente mais solicitada, quando o mesmo é submetido a variação de temperatura. O segundo é efetuar uma análise de mérito de quais propriedades termomecânicas são mais influentes na energia total armazenada durante a deformação de sistemas cerâmicos. A análise de um elevado número destas combinações foi possível graças ao uso da simulação computacional via MEF, que apresenta como vantagem a atual facilidade em se criar códigos computacionais para automatizar as simulações e o pós-tratamento dos resultados.

3.3.1 Análise 1: variação de propriedades da matriz e das inclusões

Nesta etapa do trabalho desejou-se identificar a região energeticamente mais solicitada entre duas inclusões para um grande número de combinações de propriedades termomecânicas do material e, dessa maneira, compreender o papel de cada uma delas no comportamento global do compósito e como estas afetam o perfil de energia entre inclusões. Para tal, fez-se uso de um código de automatização de simulações computacionais pelo MEF.

3.3.1.1 Materiais

O código desenvolvido permitiu a simulação de 28000 combinações de propriedades, mantendo-se sempre a inclusão como sendo mais rígida e com o coeficiente de expansão térmica maior do que a matriz, para esta primeira etapa. Na Figura 3.4 pode-se observar um diagrama de como o código foi desenvolvido. Inicialmente, criou-se o vetor de módulo de elasticidade das inclusões $\underline{E_i}$ contendo 7 pontos $(E_i{}^x)$ linearmente espaçados no intervalo $[100 - 400]\,GPa$. A

partir da criação de $\underline{E_i}$, criou-se o vetor $\underline{E_m}$, que por sua vez apresenta 20 pontos linearmente espaçados $[70 - E_i^x]\,GPa$. O coeficiente de expansão térmica da inclusão foi mantido fixo em $\underline{\alpha_i} = 10 \cdot 10^{-6}\,°C^{-1}$, enquanto o vetor de coeficiente de expansão térmica da matriz, $\underline{\alpha_m}$, apresenta 10 pontos entre os extremos 0.5 e $10 \cdot 10^{-6}\,°C^{-1}$. Por fim, o vetor de fração volumétrica de inclusões $\underline{\phi_i}$ apresenta 20 pontos linearmente espaçados no intervalo $[0.11 - 0.70]$. $\underline{\nu_i}$ e $\underline{\nu_m}$ foram mantidos fixos em 0,24 e 0,20, respectivamente.

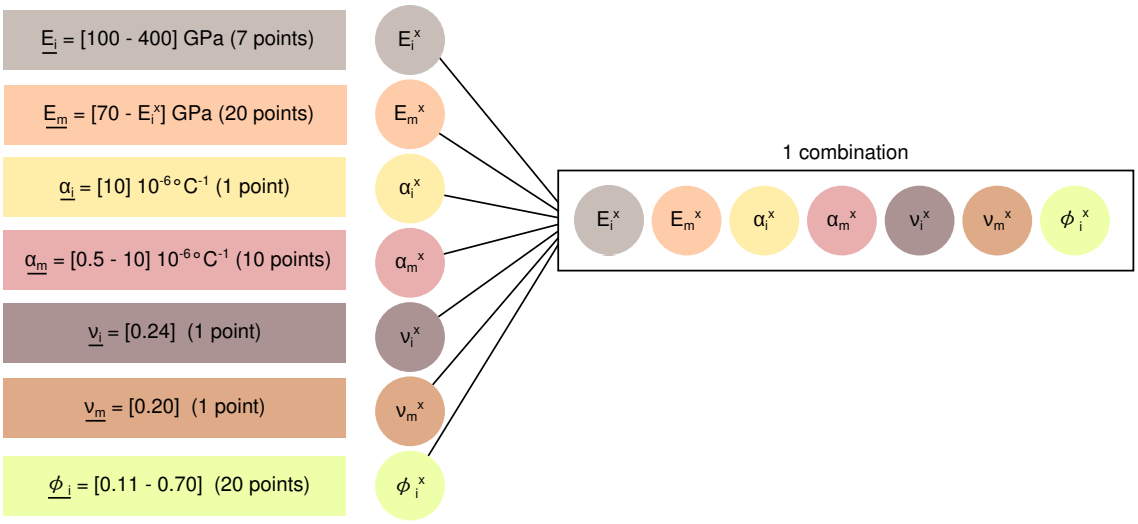

Figura 3.4 Esquema do algorítmo desenvolvido para a simulação das 28000 combinações

A fração volumétrica de inclusões é representada na simulação por meio da distância entre inclusões. Para tal, assumiu-se a hipótese de que as inclusões estejam dispostas de maneira homogênea na matriz e apresentam uma distribuição CFC (cúbica de face centrada) [30] (Eq. 2.25 Pág.18).

Assim, a partir da combinação de todos os pontos dos vetores de propriedades termomecânicas ($\underline{E_i}$, $\underline{E_m}$, $\underline{\nu_i}$, $\underline{\nu_m}$, $\underline{\alpha_i}$ e $\underline{\alpha_m}$) e do vetor de fração volumétrica de inclusões ($\underline{\phi_i}$), foram simuladas as 28000 combinações resultantes.

3.3.1.2 Métodos

Em todos os casos, considerou-se o resfriamento ($\Delta T = 300\,°C$) e foi analisado o sistema em regime permanente, ou seja, todo o sistema variava $300\,°C$, não sendo considerados efeitos de condução, convecção e radiação. O ponto O (Figura 3.3) apresenta restrições de deslocamento nas direções r e z. As si-

mulações foram desenvolvidas no software comercial Abaqus^{TM} e são utilizados elementos do tipo CAX4R. Utilizou-se da axissimetria como hipótese simplificadora, reduzindo o tempo de simulação.

As simulações apresentavam como arquivo de saída o perfil de distribuição de energia entre as inclusões ($U_t(y,p)$) (Eq. 3.4).

$$U_t(y,p) = \frac{1}{2}\sigma(y,p) : \varepsilon(y,p) \tag{3.4}$$

onde: y é a posição do ponto material e $p = \{E_i\ E_m\ \alpha_i\ \alpha_m\ \nu_i\ \nu_m\ \phi_i\ \}^T$

Desta maneira, a partir de um software externo ($Matlab^{TM}$), foi possível efetuar o pós-tratamento destes resultados e identificar qual é a região energeticamente mais solicitada em cada simulação (Eq. 3.5).

$$R(y,p) = filter(U(y,p)) \begin{cases} 0, & U(y_1,p) > U(y_2,p) \\ 1, & U(y_1,p) < U(y_2,p) \end{cases} \tag{3.5}$$

3.3.2 Análise 2: variação de propriedades da inclusão mantendo-se fixas as da matriz

Para uma melhor compreensão dos resultados apresentados, foram realizadas novas combinações, dessa vez mantendo-se as propriedades da matriz fixas (Tabela 3.2). Desta maneira, possuía-se fixado o valor da densidade de energia termodinâmica de formação de superfície (γ_s) e pela Eq. 2.17, pág. 14, foi possível calcular o valor da energia de formação de superfície e compará-lo com o valor de energia armazenada após a deformação do sistema cerâmico, estabelecendo-se então um critério de falha.

3.3.2.1 Materiais

Considerou-se a matriz como sendo vítrea e suas propriedades (Tabela 3.2) foram extraídas de Davidge e Green [9], enquanto as propriedades da inclusão foram variadas segundo a Figura 3.5, totalizando 3375 novas combinações. Novamente foi adotado o modelo M2 (Figura 3.3) com restrições de deslocamentos

no ponto O. Neste caso, considerou-se um $\Delta T = 600° C$, uma vez que eram buscadas combinações em que ocorresse o auto-trincamento.

Tabela 3.2 Propriedades termomecânicas da matriz

Propriedade	Valor
E_m	$70\,GPa$
ν_m	0.20
α_m	$3.6 \cdot 10^{-6}\,°C^{-1}$
γ_s	$4\,Jm^{-2}$

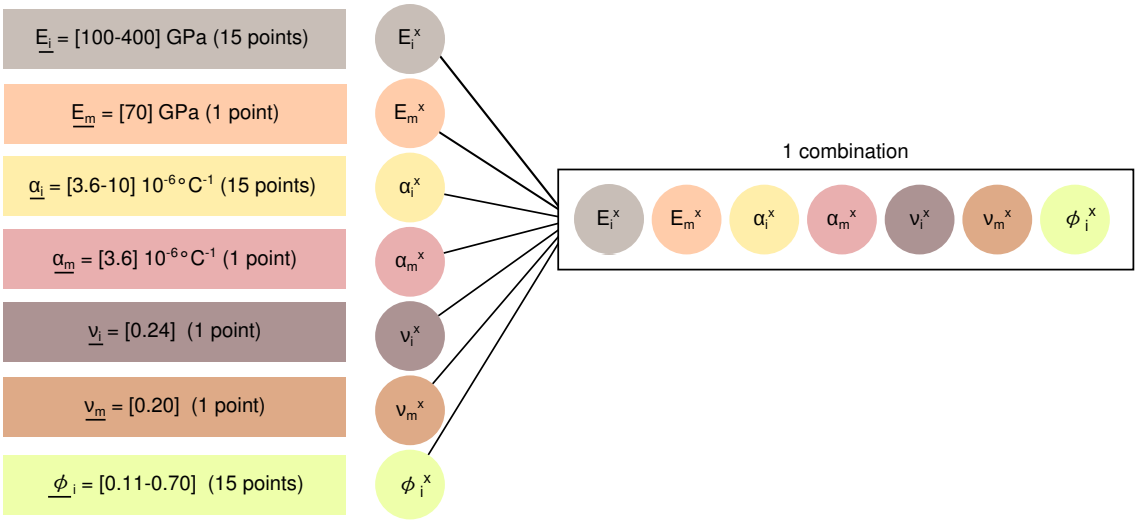

Figura 3.5 Esquema do algorítmo desenvolvido para a simulação das 3375 combinações

3.3.2.2 Métodos

Esta abordagem foi adotada para possibilitar a comparação entre a energia acumulada durante a deformação do conjunto matriz e inclusões, e a energia de formação de superfícies. Assim, caso a primeira supere a última, considera-se que houve trincamento. Para estes casos, foi analisada a região energeticamente mais solicitada e, consequentemente, mais suscetível ao surgimento da trinca.

A partir do trabalho de Davidge e Green [9], pode-se calcular a energia de formação de superfície, U_s, para inclusões esféricas envoltas em matriz contínua (Eq. 2.17 Pág. 14).

Neste caso, a resposta das análises foram, além do perfil de distribuição de energia entre as duas inclusões, a energia total armazenada durante a deforma-

ção do material. Assim, pode-se verificar em quais combinações houveram o auto-trincamento e, nestes casos, em que local a trinca estaria mais suscetível à ser nucleada.

3.4 Estudo 4: Modelagem via elementos coesivos da fissuração espontânea de sistemas cerâmicos

A partir da necessidade de se simular problemas mais próximos à realidade, optou-se por simular a fissuração de um material monofásico policristalino, em que as propriedades dos grãos são dependentes de suas orientações (textura). Este estudo foi dividido em 3 partes, sendo a primeira uma análise com um único elemento coesivo para que se pudesse compreender a implementação deste tipo de elemento no software ABAQUS^{TM}. A segunda fase consistiu da análise do efeito do tamanho de grão na fissuração espontânea de sistemas cerâmicos, enquanto a terceira etapa analisou o efeito da distribuição de orientações dos grãos na fissuração espontânea de sistemas cerâmicos.

3.4.1 Análise 1: Estudo do comportamento de um único elemento coesivo

3.4.1.1 Materiais

As propriedades mecânicas dos elementos estruturais e do elemento coesivo encontram-se nas Tabelas 3.3 e 3.4, respectivamente. Considerou-se os elementos estruturais com comportamento linear elástico e os elementos coesivos com comportamento bilinear em termos de TS. Para simplificação de análise, ambos materiais foram considerados genericamente, não se preocupando, neste momento, com a alocação de valores condizentes com materiais reais.

Tabela 3.3 Propriedades mecânicas elementos estruturais

Propriedade	Valor
E (GPa)	210
ν	0,3

Tabela 3.4 Propriedades mecânicas elemento coesivo

Propriedade	Valor
K_n ($GPa\,m^{-1}$)	28
$K_{t,s}$ ($GPa\,m^{-1}$)	14
G_{IC} (kJ m^{-2})	2.5
${t_n}^o$ (MPa)	75
${t_{t,s}}^o$ (MPa)	35

3.4.1.2 Métodos

Na utilização de EF para a simulação de situações complexas, é uma boa prática fazer uma simulação com um único elemento e diversos carregamentos, para que se tenha certeza que os parâmetros de entrada e a metodologia de simulação são adequados. Neste caso, fez-se um modelo axissimétrico simples, com um elemento coesivo e dois elementos estruturais (Figura 3.6).

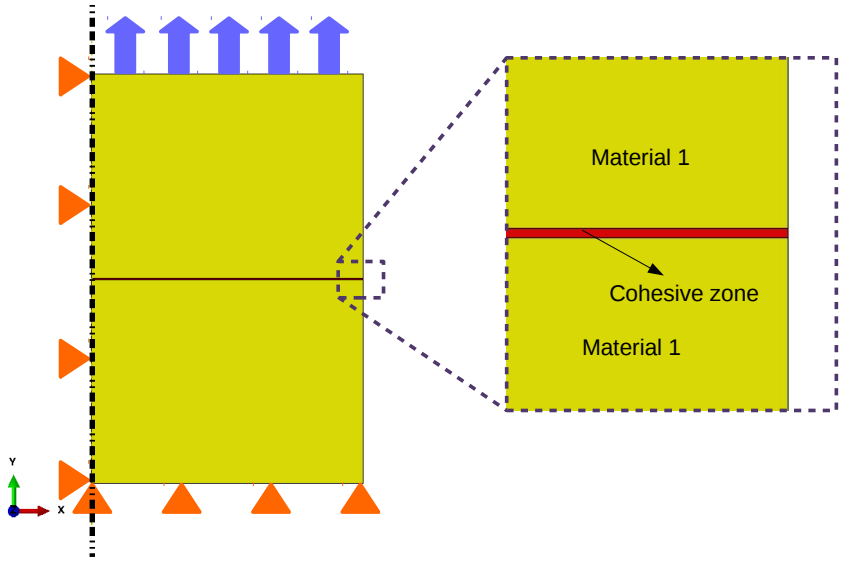

Figura 3.6 Modelo axissimétrico com um elemento coesivo e dois elementos estruturais

Restringiu-se os deslocamentos verticais na face inferior do elemento estrutural de base, bem como os deslocamentos horizontais dos três elementos na face pertencente ao eixo de axissimetria. Foram aplicadas solicitações em deslocamento vertical na face superior do elemento estrutural situado no topo do modelo, variando sentido e magnitude em função do tempo, para que se pudesse anali-

sar as características do comportamento do elemento coesivo. Foi considerado o critério MAXS para o início de degradação.

O acoplamento entre o elemento coesivo e os elementos contínuos de malha foi configurado a partir do modelo de malha restrita, ou seja, os nós não são compartilhados, mas há uma restrição quanto ao deslocamento entre a face superior do elemento coesivo com a face inferior do elemento contínuo superior, e entre a face inferior do elemento coesivo com a face superior do elemento contínuo da base.

3.4.2 Análise 2: Estudo do efeito do tamanho de grão na fissuração espontânea de sistemas cerâmicos

Esta etapa do trabalho baseou-se no artigo escrito por Fertig & Nickerson [44], realizando algumas avaliações que esses autores não abordaram. Tem-se como exemplo o estudo do tamanho do grão no comportamento em fratura do sistema cerâmico analisado.

3.4.2.1 Materiais

Assim como no trabalho de Fertig & Nickerson [44], foram considerados grãos hexagonais de cordierita. A cada grão foi associada uma orientação randômica (ω) entre -90° e 90° em relação a uma linha vertical (Figura 3.7). Para gerar estas orientações randômicas foi escrito um código em MATLAB^{TM} (Apêndice D). Os dados de entrada do código são: número de grãos, orientação média (μ) dos grãos e desvio padrão das orientações (σ). Desta maneira, o código gera uma distribuição de orientações randômica, mas que globalmente satisfaça as variáveis de média e desvio padrão fornecidas pelo usuário (distribuição global segue uma função normal). Nas simulações desta seção, a orientação média foi de 0° e o desvio padrão de 15°. Isso possibilitou a análise do efeito da anisotropia no comportamento em fratura de sistemas cerâmicos, conforme será discutido na Seção 3.4.3, (pág. 45).

Na análise aqui detalhada, a variável de estudo foi o tamanho de grão. Foram adotados três diâmetros distintos: 25, 50 e 200 μm. Os três modelos simulados

possuem as mesmas orientações apresentadas na Figura 3.7. A distribuição de orientações é apresentada na Figura 3.8, em que o eixo *cell orientation* fornece o intervalo de orientações. As propriedades termomecânicas da cordierita são apresentadas na Tabela 3.5 (extraídas do trabalho de Bubeck [48]), e as propriedades dos elementos coesivos são apresentadas na Tabela 3.6 (extraídas do trabalho de Fertig & Nickerson [44]).

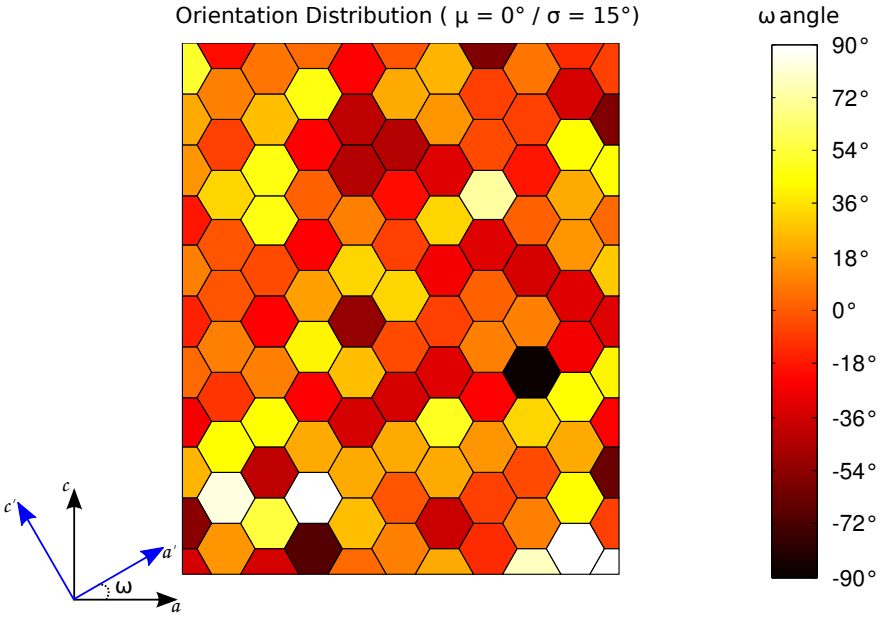

Figura 3.7 Orientações (ω) dos grãos de cordierita ($\mu = 0° \ / \ \sigma = 15°$)

Tabela 3.5 Propriedades da cordierita, extraído de Bubeck [48]

Propriedades da Cordierita	Valor
Módulo de Young$_{a/b}$ (GPa)	148
Módulo de Young$_c$ (GPa)	127
Módulo de Cisalhamento (GPa)	47
Coeficiente de Poisson	0,34
$\alpha_c(10^{-6}K^{-1})$	-1,2
$\alpha_{a/b}(10^{-6}K^{-1})$	3,87

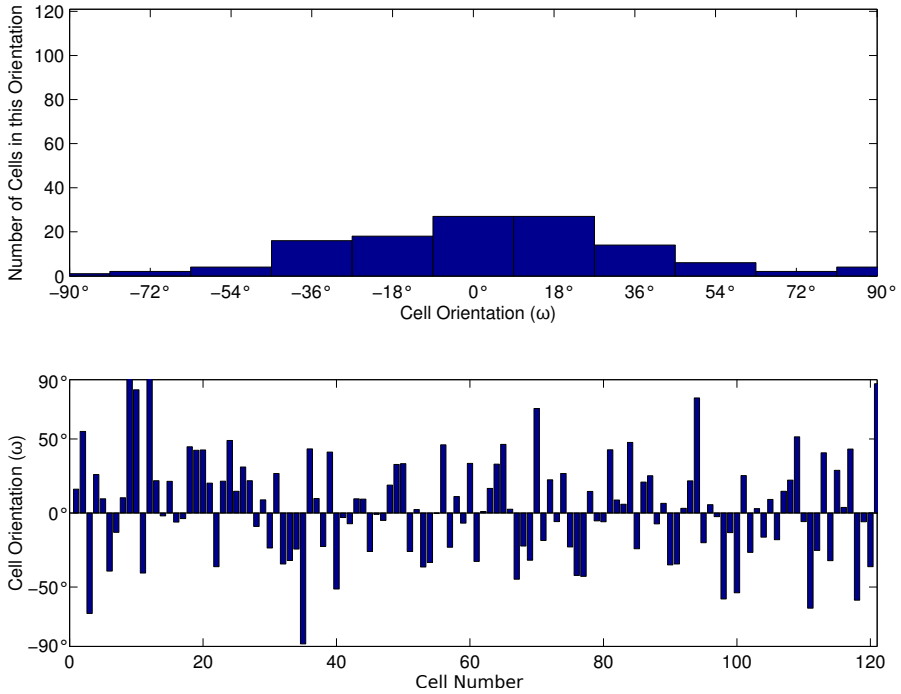

Figura 3.8 Distribuição de orientações dos grãos de cordierita ($\mu = 0° / \sigma = 15°$)

Tabela 3.6 Propriedades do elemento coesivo, extraído de Fertig & Nickerson [44]

Propriedades do coesivo	Valor
K_n (GPa m^{-1})	7,40 $\cdot 10^8$
K_s (GPa m^{-1})	2,35 $\cdot 10^8$
σ_{inic} (MPa)	700
G (J m^{-2})	0.543

3.4.2.2 Métodos

Um modelo bi-dimensional em EF composto de domínios/grãos hexagonais foi criado utilizando-se o software comercial ABAQUS^{TM}. Um material modelo contendo 121 domínios foi construído com o propósito deste estudo. Para tal, foi desenvolvido um código de automatização em linguagem Python (Apêndice D). Uma camada de elementos coesivos (COH2D4) é modelada na interface de cada domínio, estas camadas de elementos coesivos foram vinculadas aos elementos estruturais a partir de restrições do tipo *tie* (seguindo o mesmo princípio da Figura 2.10, pág. 21). As dimensões físicas de cada elemento coesivo foram mantidas constantes sendo 0.2 μm de espessura e 0.5μm de comprimento.

As condições de contorno adotadas podem ser visualizadas na Figura 3.9. Foi restrito o deslocamento na direção *y* dos nós contidos na aresta $\overline{AB}$ e na direção *x*, dos nós contidos na aresta $\overline{DA}$. Quanto aos nós contidos nas arestas $\overline{BC}$ e $\overline{CD}$, foram aplicadas condições de acoplamento para que as arestas $\overline{AB}$ / $\overline{CD}$ e $\overline{DA}$ / $\overline{BC}$ permanecessem paralelas. Aplicou-se uma variação de temperatura homogênea em todo o modelo, com um ΔT = -500$^{\circ}C$, partindo de 500$^{\circ}C$ e resfriando até 0$^{\circ}C$, com a amplitude variando segundo uma senóide. Após atingir 0$^{\circ}C$, a temperatura é novamente alterada até atingir 500$^{\circ}C$. Assim como em todas as simulações discutidas nesta dissertação, o resfriamento é considerado em estado permanente.

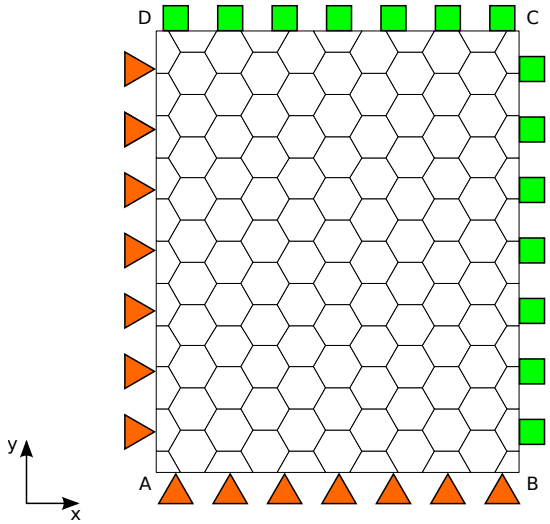

Figura 3.9 Modelo EF para a simulação da fissuração espontânea de sistemas cerâmicos, baseado em Fertig & Nickerson [44]

3.4.3 Análise 3: Estudo do efeito da distribuição de orientações de grãos no comportamento de sistemas cerâmicos submetidos a variação de temperatura

A orientação de grãos é algo de extrema importância no comportamento de materiais não isotrópicos. Assim, entendeu-se que analisar seu efeito no comportamento de sistemas cerâmicos submetidos a variação de temperatura seria interessante, principalmente para materiais como a cordierita, que possuem coeficientes de expansão térmica positivos nas direções a e b e negativo na direção

c do cristal.

3.4.3.1 Materiais

Como o código desenvolvido para a distribuição da orientação de grãos busca respeitar uma distribuição global normal, diminuir o desvio padrão induziria uma distribuição mais homogênea, enquanto aumentar o desvio padrão induziria uma distribuição mais heterogênea de orientações. Assim, esta etapa do trabalho consistiu na comparação de quatro perfis de distribuição de orientação de grãos de cordierita, mantendo-se os diâmetros de grãos fixos em 50μm. Manteve-se a média da distribuição de orientações em 0°, porém, variou-se o desvio padrão. Foram analisados os desvios padrões: 5°, 15°, 35° e 55°. As distribuições espaciais de orientações, bem como seus respectivos gráficos de distribuições são apresentados nas Figuras 3.10 – 3.11, 3.12 – 3.13, 3.14 – 3.15 e 3.16 – 3.17. As propriedades termomecânicas da cordierita e dos elementos coesivos são as mesmas utilizadas na Seção 3.4.2.1 (Tabelas 3.5 e 3.6).

3.4.3.2 Métodos

Utilizou-se o mesmo método apresentado na Seção 3.4.2.2, pág. 44. Salvo que no estudo da anisotropia acrescentou-se uma variável de análise, o coeficiente de expansão térmica global (α). Como duas das condições de contorno do modelo são as equações de acoplamento nas arestas $\overline{BC}$ e $\overline{CD}$, ou seja, ao se deformar o *Honeycomb* mantinha a forma retangular, foi possível com um simples equacionamento de dilatação térmica ($\Delta L = L_0 \cdot \alpha \cdot \Delta T$) encontrar os α efetivos nas direções x e y (Figura 3.18).

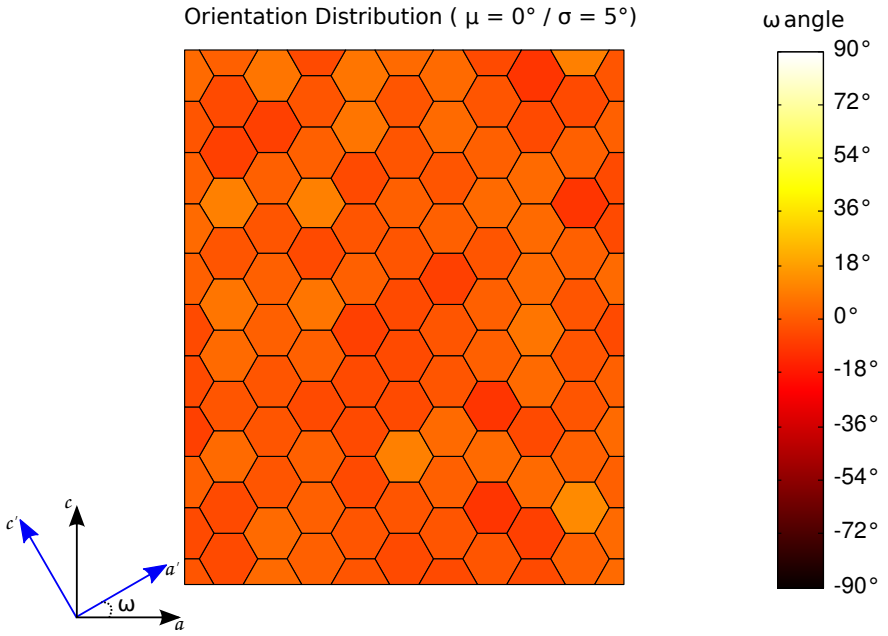

Figura 3.10 Orientações dos grãos de cordierita ($\mu = 0° \ / \ \sigma = 5°$)

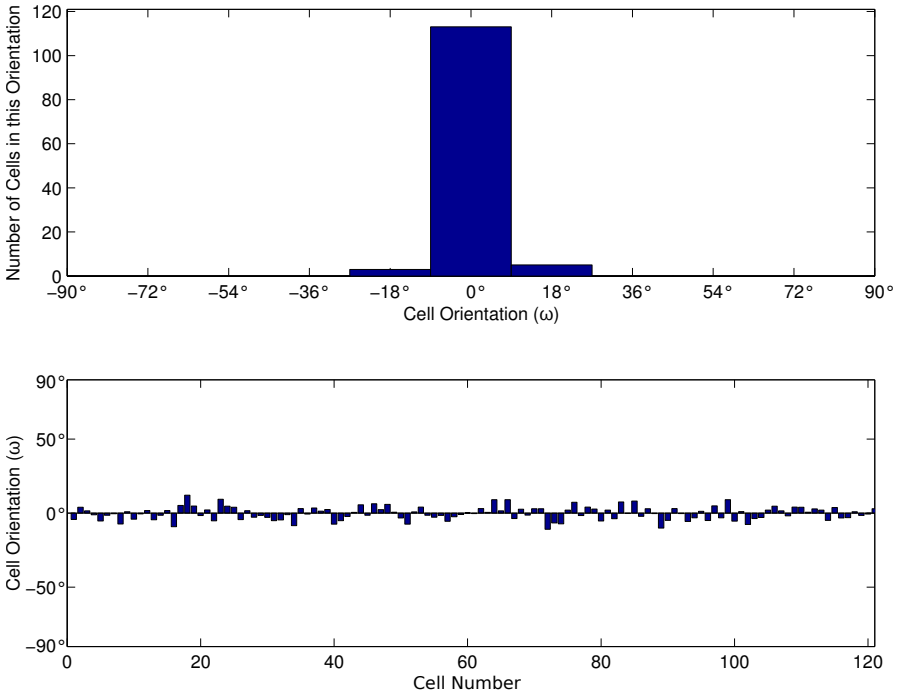

Figura 3.11 Distribuição de orientações dos grãos de cordierita ($\mu = 0° \ / \ \sigma = 5°$)

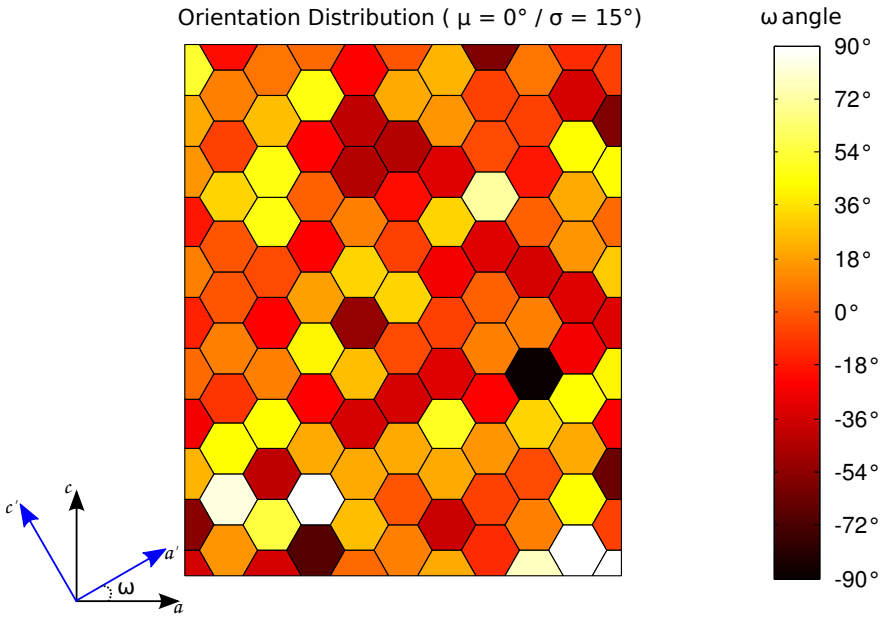

Figura 3.12 Orientações dos grãos de cordierita ($\mu = 0° \ / \ \sigma = 15°$)

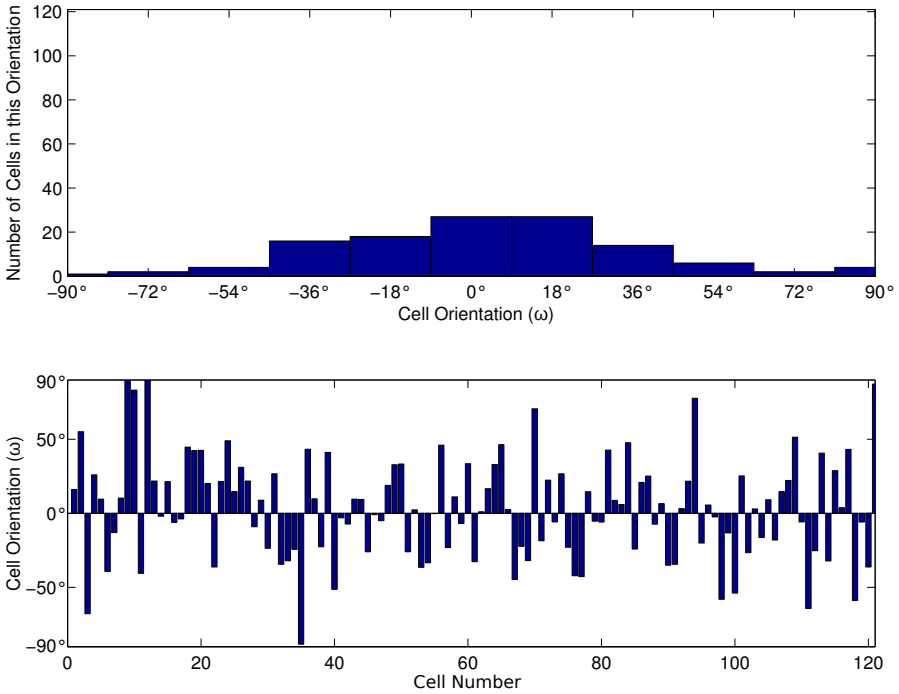

Figura 3.13 Distribuição de orientações dos grãos de cordierita ($\mu = 0° \ / \ \sigma = 15°$)

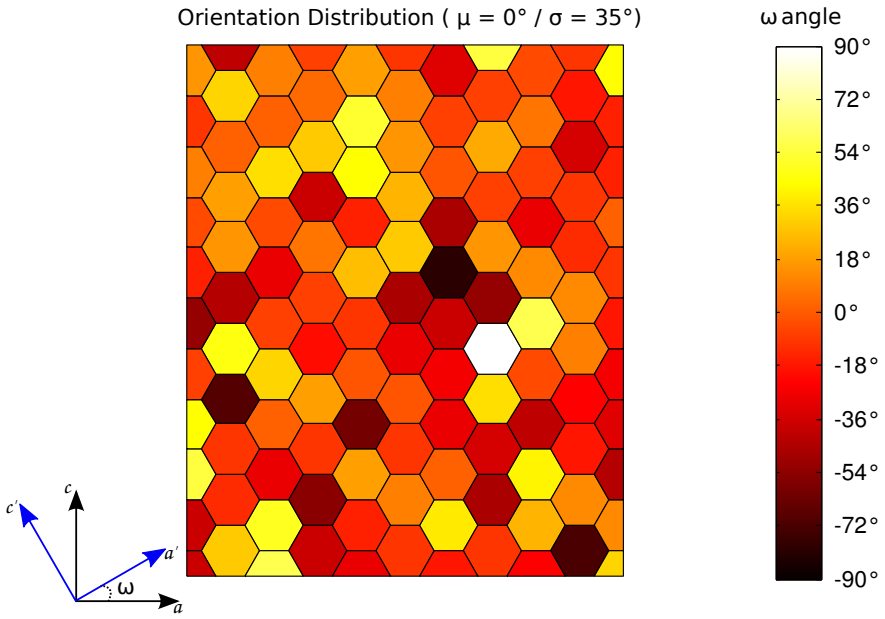

Figura 3.14 Orientações dos grãos de cordierita ($\mu = 0° / \sigma = 35°$)

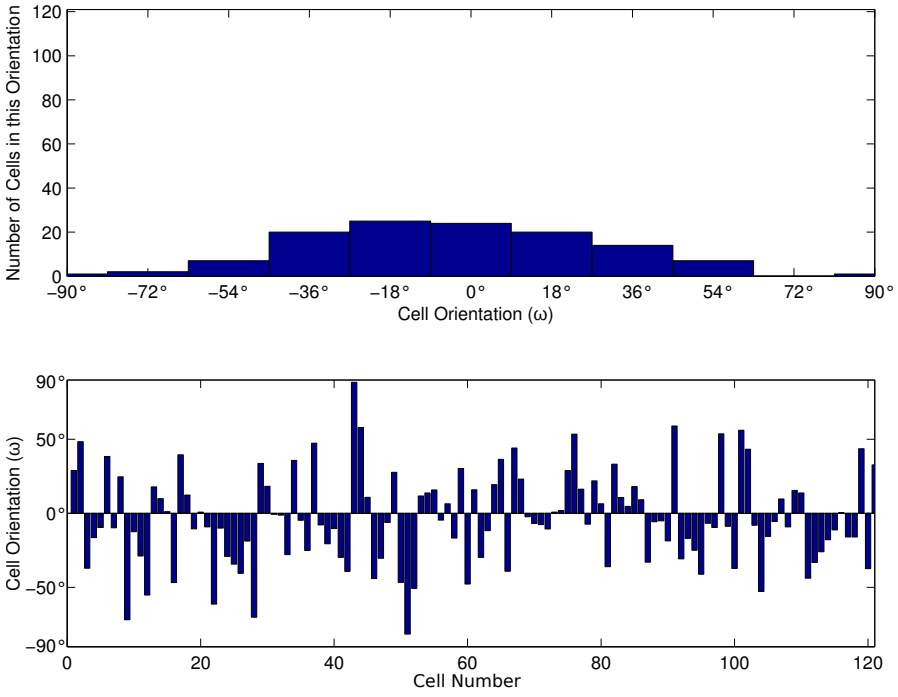

Figura 3.15 Distribuição de orientações dos grãos de cordierita ($\mu = 0° / \sigma = 35°$)

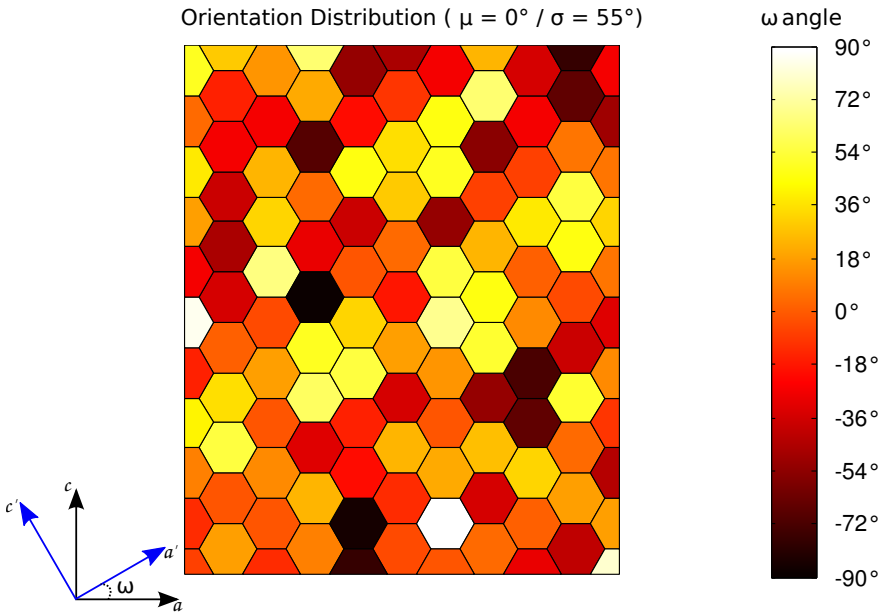

Figura 3.16 Orientações dos grãos de cordierita ($\mu = 0° \ / \ \sigma = 55°$)

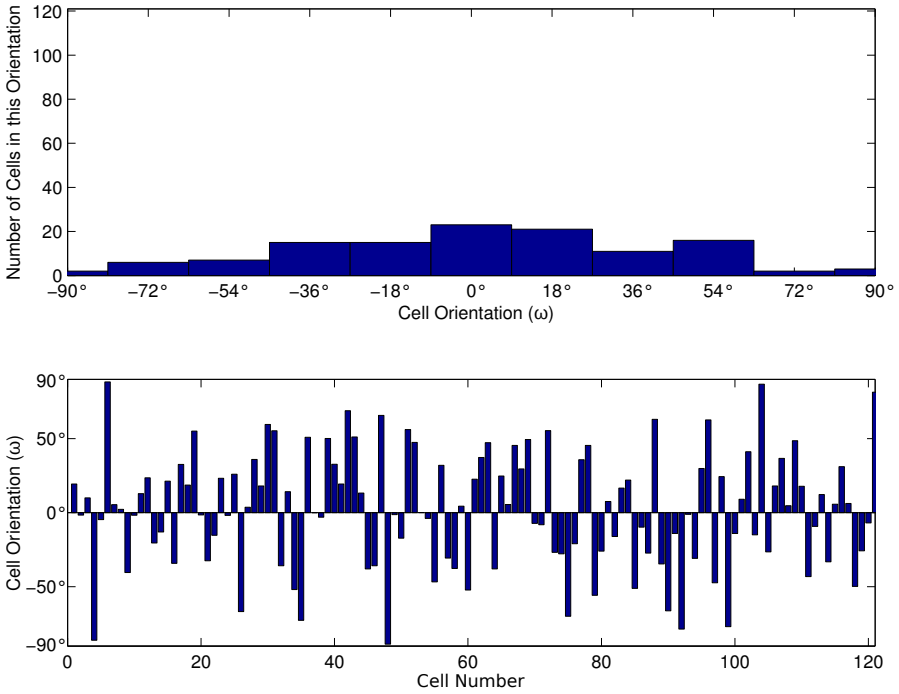

Figura 3.17 Distribuição de orientações dos grãos de cordierita ($\mu = 0° \ / \ \sigma = 55°$)

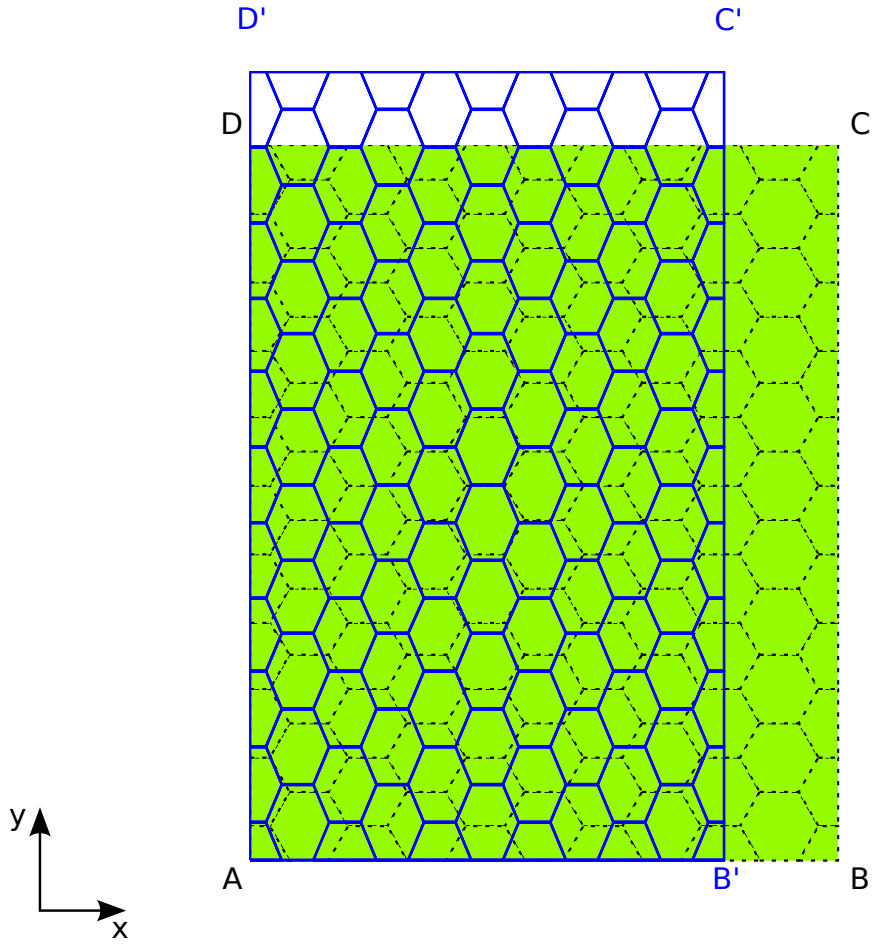

Figura 3.18 Modelo simplificado para o cálculo do coeficiente de dilatação efetivo

4 RESULTADOS E DISCUSSÕES

4.1 Estudo 1: Comparação entre raios críticos de inclusões de diversos modelos

Os modelos axissimétricos em elementos finitos descritos anteriormente permitiram prever o raio crítico em função do cálculo da energia total de deformação, U_t. Na Tabela 4.1 é apresentada uma síntese dos resultados obtidos via MEF (M1, M3 e Mn), os dados experimentais da literatura e aqueles obtidos a partir do modelo analítico de previsão de raio crítico de fratura estabelecido por Davidge e Green [9], Eq. 2.18.

Tabela 4.1 Comparação dos raios críticos de inclusão obtidos via MEF (M1, M3 e Mn) e modelo analítico de Davidge e Green [9] e Liu e Winn [19] com dados experimentais da literatura.

	ϕ [vol.%]	ΔT [C°]	**Raio crítico de inclusão [μm]**					
			Exp.	D e G	Liu e Winn	M1	M3	Mn
[9] (G1)	10	545	[29-41]	22	11	22	22	22
[9] (G2)	10	500	[97-137]	61	32	61	61	61
[45]*	20	800	[7-13]	13	4	13	13	13
[20]	15	295 (Fig. 3.1)	250**	213	76	212	212	211
	30	282 (Fig. 3.1)	250**	234	41	232	231	229
	45	244 (Fig. 3.1)	250**	312	33	310	307	301

*as inclusões em Todd and Derby [45] não são esféricas
**a é fixo em 250 μm, variou-se as frações volumétricas

Os resultados mostram a boa concordância dos modelos analítico e numéricos com os dados experimentais, sobretudo para baixas frações volumétricas. Para os dados de Todd e Derby [45], é possível notar a robustez da predição do modelo em baixas frações, visto que as inclusões não eram esféricas, a matriz não era contínua, e tampouco a distribuição de inclusões era homogênea. Pode-se justificar esta proximidade nos valores pelo fato de que em baixas frações volumétricas de inclusão, a probabilidade destas estarem razoavelmente distanciadas entre si é maior do que para frações altas, aproximando-se, desta forma, da hipótese de partícula isolada de Davidge e Green [9]. Para o caso experimental de maior fração volumétrica (ϕ = 45 vol.$\%$), os modelos M3 e Mn apresentam

aproximações melhores para o raio crítico quando comparados aos resultados dos modelos analíticos. No caso dos experimentos de Joliff et al. [20], a previsão via modelo de Davidge e Green é diferente para cada fração volumétrica devido aos diferentes valores de ΔT (Figura 3.1). Deve-se salientar a má correspondência do modelo de Liu e Winn [19] com os resultados experimentais. Acredita-se que as hipóteses de único ponto de contato e interface matriz-inclusão perfeita sejam os maiores responsáveis por esta discrepância.

A não necessidade pelo MEF de determinadas hipóteses presentes em modelos analíticos, como tensão normal média constante ao longo da interface, motiva sua aplicação em sistemas ditos com altas frações volumétricas. Sendo assim, foi feita uma investigação da aplicação destes modelos na determinação do raio crítico para o material estudado por Joliff et al. [20], variando-se a fração volumétrica. Na Figura 4.1 é apresentada a distribuição de raios críticos em função da fração volumétrica para os diferentes modelos em um sistema com raio de inclusão, a, e variação de temperatura, ΔT, fixos, além do ponto experimental obtido por Joliff et al. [20] para a fração de 45 vol.$\%$. Nota-se a independência da fração volumétrica, tanto para o modelo de previsão de Davidge e Green [9], quanto para o modelo M1. Já para os modelos M3 e Mn, nos quais a fração volumétrica é representada pela distância entre inclusões, esta influência é significativa, sendo mais relevante no modelo Mn. É importante notar que para frações volumétricas inferiores a 30 vol.$\%$, os valores de raio crítico obtidos por todos os modelos são próximos, o que sugere que os modelos M3 e Mn são complementares ao analítico de Davidge e Green [9], uma vez que são sensíveis à proximidade entre as inclusões, ou seja, considerando os efeitos do aumento da fração volumétrica.

Liu e Winn [19] apresentam um equacionamento de raio crítico em seu trabalho, porém, devido ao decaimento exponencial destes valores (vide Tabela 4.1), não foram incluídos na Figura 4.1. Os valores obtidos por estes autores para a mesma faixa de frações volumétricas da Figura 4.1 variam de 291 a 18 μm.

Neste contexto da investigação dos efeitos das hipóteses assumidas em modelos analíticos, comparou-se a distribuição de tensões entre duas inclusões

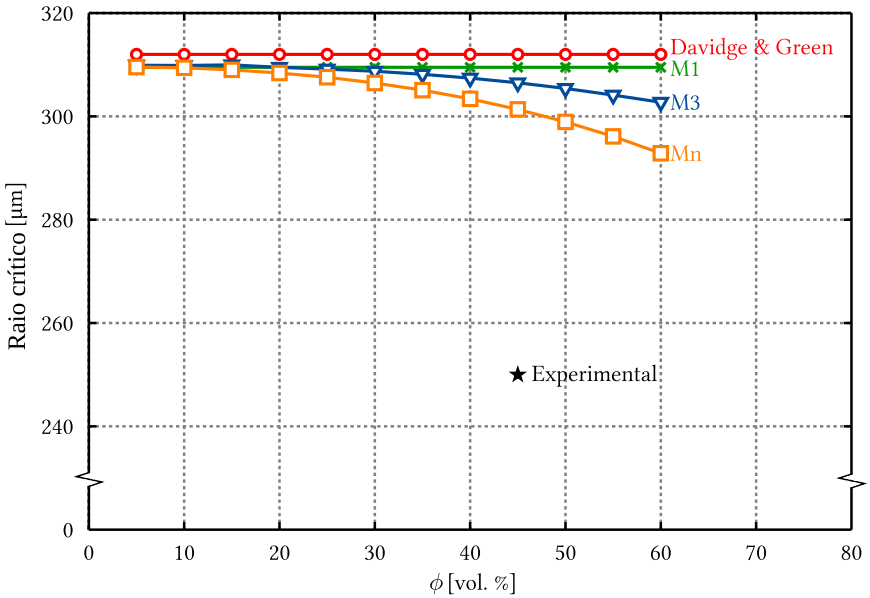

Figura 4.1 Raio crítico (a_c) em função da fração volumétrica utilizando-se Davidge e Green e MEF para um sistema alumina (i) / borosilicato (m) retirado de Joliff et al. [20], assumindo-se os seguintes valores: E_i = 340 GPa, α_i = 7,6·10^{-6} °C^{-1}, ν_i = 0,24, E_m = 68 GPa, α_m = 4,6·10^{-6} °C^{-1}, ν_m = 0,2 com ΔT = 244 °C. O resultado experimental foi obtido por Joliff et al. [20]

proposta por Liu e Winn [19] com aquela obtida pelo modelo Mn. Este modelo analítico assume que haverá uma contribuição linear no valor de tensão e deformação em determinado ponto P no material, a partir dos campos de tensão e deformação das duas inclusões distanciadas r e $2b - r$ deste ponto (Figura 2.5). Esta hipótese simplificadora faz com que o efeito da proximidade entre duas inclusões seja calculado apenas no eixo de axissimetria. Já no modelo em elementos finitos, há uma distribuição contínua de matriz entre as inclusões, permitindo uma avaliação mais realista do efeito do aumento da fração volumétrica na distribuição de tensões.

Para evidenciar os avanços obtidos com o modelo Mn em relação ao modelo proposto por Liu e Winn [19] referente à distribuição de tensões na porção de matriz entre as inclusões, utilizaram-se dados do trabalho de Joliff et al. [20]. Para isso comparou-se diretamente as tensões máximas, σ_{max}, e mínimas, σ_{min}, principais, obtidas via MEF e pelo modelo de Liu e Winn [19], como apresentado nas Figuras 4.2 e 4.3. Para efeito de comparação mais direta, dividiu-se a posição

entre as inclusões, z, pelo raio das inclusões, *i.e.*, a distância de uma inclusão a outra é dada por um múltiplo do raio de inclusão, $n = z/a$.

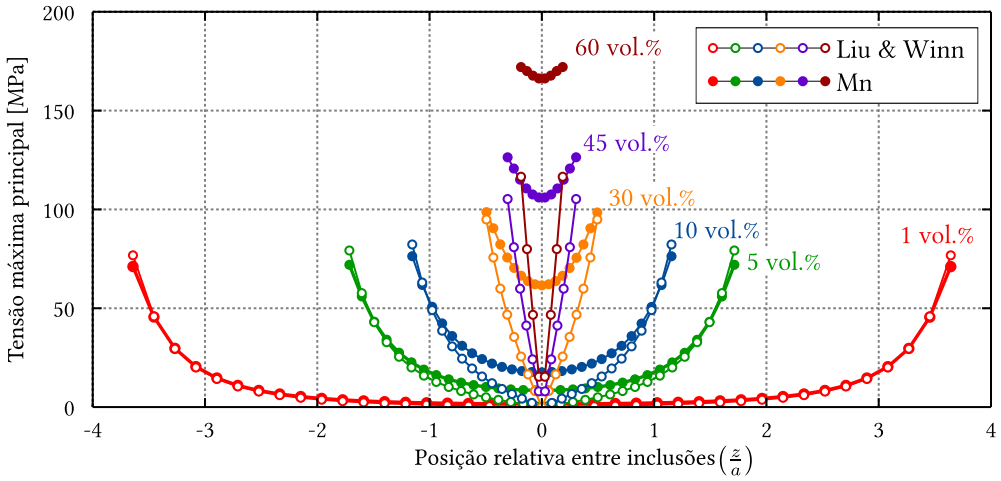

Figura 4.2 Comparação da distribuição de tensões máximas principais do modelo de Liu & Winn [19] com o modelo proposto por Jollif [20] para um sistema alumina (i) / borosilicato (m), assumindo-se os seguintes valores: E_i = 340 GPa, α_i = 7,6·10^{-6} °C^{-1}, ν_i = 0,24, E_m = 68 GPa, α_m = 4,6·10^{-6} °C^{-1}, ν_m = 0,2 com ΔT = 270 °C e diferentes frações volumétricas de inclusão

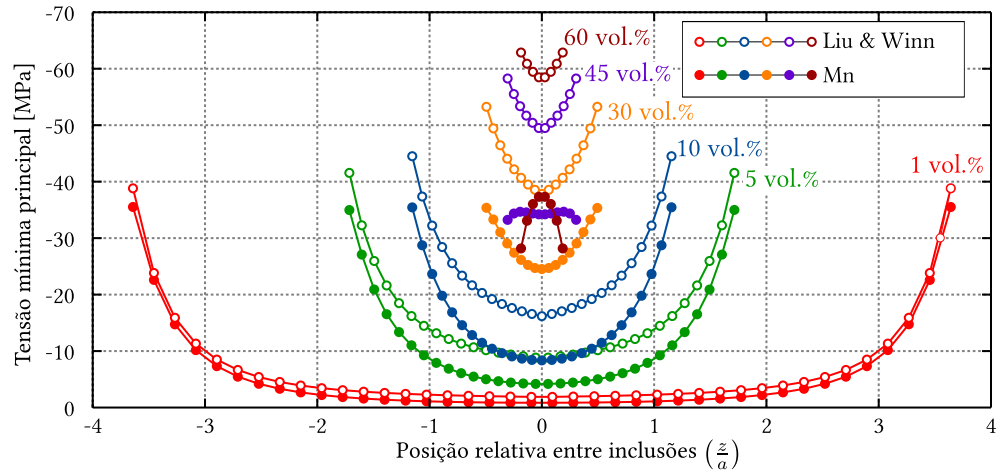

Figura 4.3 Comparação da distribuição de tensões mínimas principais do modelo de Liu & Winn [19] com o modelo proposto por Jollif [20] para um sistema alumina (i) / borosilicato (m), assumindo-se os seguintes valores: E_i = 340 GPa, α_i = 7,6·10^{-6} °C^{-1}, ν_i = 0,24, E_m = 68 GPa, α_m = 4,6·10^{-6} °C^{-1}, ν_m = 0,2 com ΔT = 270 °C e diferentes frações volumétricas de inclusão

É possível notar que a distribuição de tensões máximas e mínimas principais são semelhantes tanto para o modelo analítico como para o modelo em ele-

mentos finitos até a fração volumétrica de 10 vol.%. Entretanto, ao aumentá-la, diferenças significativas podem ser notadas. Na distribuição de tensão máxima principal, nota-se o aumento de tensões no ponto O da Figura 3.2, diferindo-se do resultado do modelo analítico de Liu e Winn [19]. Isto pode ser explicado pela consideração de um único ponto de tangência entre os conjuntos esféricos (matriz e inclusão) no modelo de Liu e Winn [19] (Figura 2.5), enquanto em Mn considera-se a matriz contínua entre as inclusões (Figura 3.2). Na Figura 4.2 é evidente o aumento da diferença entre os modelos quanto às tensões na interface matriz / inclusão com o aumento da fração volumétrica. Isto deve-se tanto pelo cálculo das componentes cisalhantes no Mn, as quais são nulas no modelo analítico, quanto pela consideração de distribuição contínua de material entre as inclusões.

Outra diferença importante nas respostas dos modelos é apresentada na Figura 4.3. Trata-se da inversão do comportamento da distribuição de tensões mínimas principais. Analisando-se as curvas das frações de 30, 45 e 60 vol.%, nota-se a gradual transição entre o posicionamento do maior valor da tensão mínima principal, originalmente na interface matriz / inclusão, para o ponto equidistante entre as duas inclusões. No modelo analítico este fenômeno não é observado, novamente devido à sua hipótese de contribuição linear das duas inclusões.

Esta constatação motivou a realização de um estudo mais aprofundado quanto à influência da fração volumétrica nos estados de tensões no ponto de interface (I) matriz / inclusão ($z = b - a$) e no ponto médio (M) entre duas inclusões ($z = 0$). Na Figura 4.4 são apresentadas as variações das tensões máximas e mínimas principais nestes pontos. Nota-se que as máximas tensões principais nos dois pontos aumentam com a fração volumétrica, sendo que para o ponto médio, este acréscimo é mais sensível à variação de ϕ, aproximando-se da tensão no ponto de interface para altas frações, proporcionando uma maior homogeneidade no campo de tensões máximas principais. Por outro lado, as tensões mínimas principais possuem comportamentos inversos, sendo crescente para o ponto médio e decrescente para o ponto de interface, e interceptam-se em $\phi \approx 45$ vol.%. A somatória destes efeitos faz com que o ponto médio encontre-se mais solicitado

que o ponto de interface à ϕ = 60 vol.%, o que pode favorecer a nucleação de trincas circunferenciais neste ponto, como é notado nas imagens de Joliff et al. [20], apresentada na Figura 2.6-a (Pág. 17).

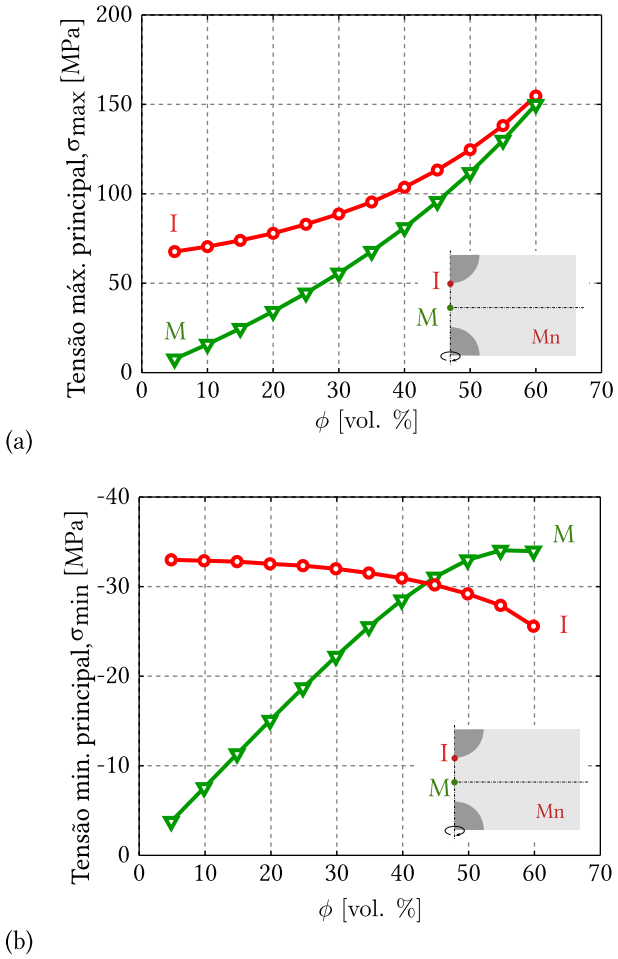

Figura 4.4 Comparação da distribuição de tensões máximas (a) e mínimas (b) principais nos pontos I e M no modelo Mn para um sistema alumina (i) / borosilicato (m), assumindo-se os seguintes valores: E_i = 340 GPa, α_i = 7,6·10^{-6} °C^{-1}, ν_i = 0,24, E_m = 68 GPa, α_m = 4,6·10^{-6} °C^{-1}, ν_m = 0,2 com ΔT = 244 °C e $a = 250\ \mu$ m

Na Figura 4.5 são apresentados os campos de tensão máxima principal e de tensão normal média no modelo Mn em função da fração volumétrica. A tensão normal média foi calculada como $tr(\underline{\underline{\sigma}})/3$, em que $\underline{\underline{\sigma}}$ é o tensor das tensões e $tr(\cdot)$ é o traço do tensor. A partir da análise dos isovalores, é possível concluir que para frações volumétricas baixas (ϕ = 10 vol.%), a distribuição de tensões radiais e circunferenciais é mais concentrada na interface matriz / inclusão. Para frações

volumétricas superiores, as tensões na interface matriz / inclusão distanciam-se da hipótese de tensão normal média homogênea na interface, indicando que as hipóteses do modelo de Davidge e Green [9] começam a perder representatividade para frações volumétricas superiores a 30 vol. $\%$.

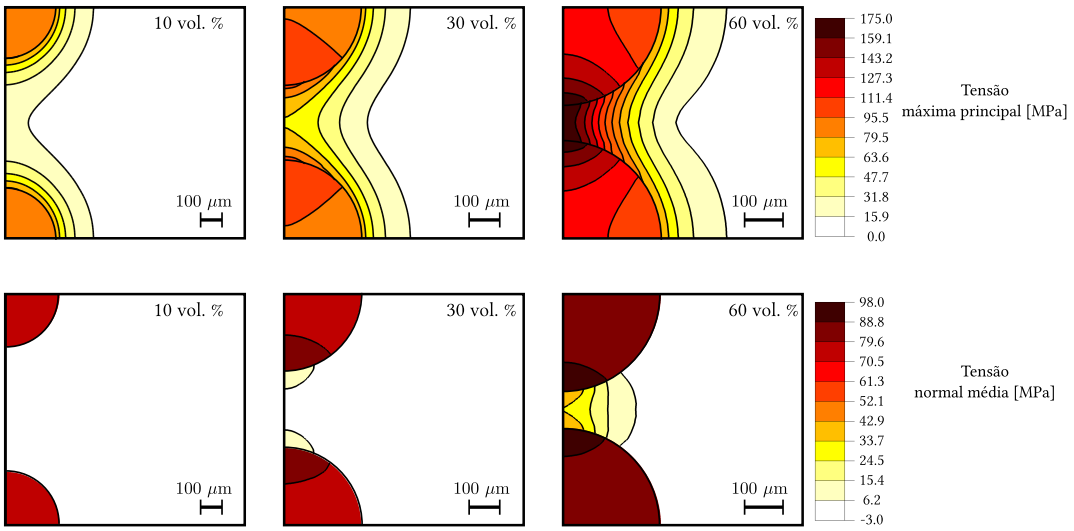

Figura 4.5 Comparação da distribuição de tensões máximas principais e tensões normais média no modelo Mn para um sistema alumina (i) / borosilicato (m), assumindo-se os seguintes valores: E_i = 340 GPa, α_i = 7,6·10^{-6} °C^{-1}, ν_i = 0,24, E_m = 68 GPa, α_m = 4,6·10^{-6} °C^{-1}, ν_m = 0,2 com ΔT = 244 °C e a = 250 μm

4.2 Estudo 2: Perfil de distribuição de energia entre inclusões para diferentes frações volumétricas

Foram realizadas simulações da etapa de resfriamento para diferentes distâncias entre inclusões, representando distintas frações volumétricas (Eq. 2.25, Pág. 18). Na Figura 4.6 é apresentado o perfil de energia normalizada na porção de matriz entre as inclusões (segmento de reta A-B). A normalização fez-se necessária pois à medida que as inclusões se aproximam, sua amplitude aumenta exponencialmente, o que impossibilita a visualização do perfil de energia para todas as simulações em um mesmo gráfico devido ao fator de escala. Tomou-se o valor 1 para quando a energia for máxima e 0 para a mínima em todas as curvas, permitindo, assim, traçá-las em um único gráfico, conservando-se seus perfis.

Como pode ser visualizado na Figura 4.6, para baixas frações volumétricas, as curvas de energia tem perfil "U", possuindo seu máximo na interface matriz-inclusão e seu mínimo no ponto médio entre inclusões. Porém, quando a razão ℓ/d é inferior à 0.18 ($\phi > 45$ vol. $\%$), há uma inversão no perfil de energia, fazendo com que o ponto médio entre as inclusões seja o mais solicitado energeticamente.

Ao investigar a distribuição de tensões radiais entre inclusões para diferentes distâncias, Joliff et al. [29] obtiveram o perfil "U"de tensão radial e não verificaram a inversão deste, concluindo que a proximidade não influenciava este perfil, ou seja, a interface seria sempre a região mais solicitada. Tal fato induziu a propor que a origem das trincas inter-inclusões estivesse associada ao processamento do compósito e não à distância entre as inclusões. Porém, eles realizaram simulações para ℓ/d variando até 0,33 (ϕ = 31 vol.$\%$). Nota-se na Figura 4.6 que a inversão ocorre para ℓ/d entre 0,228 e 0,140 (40 < ϕ < 50 vol.$\%$).

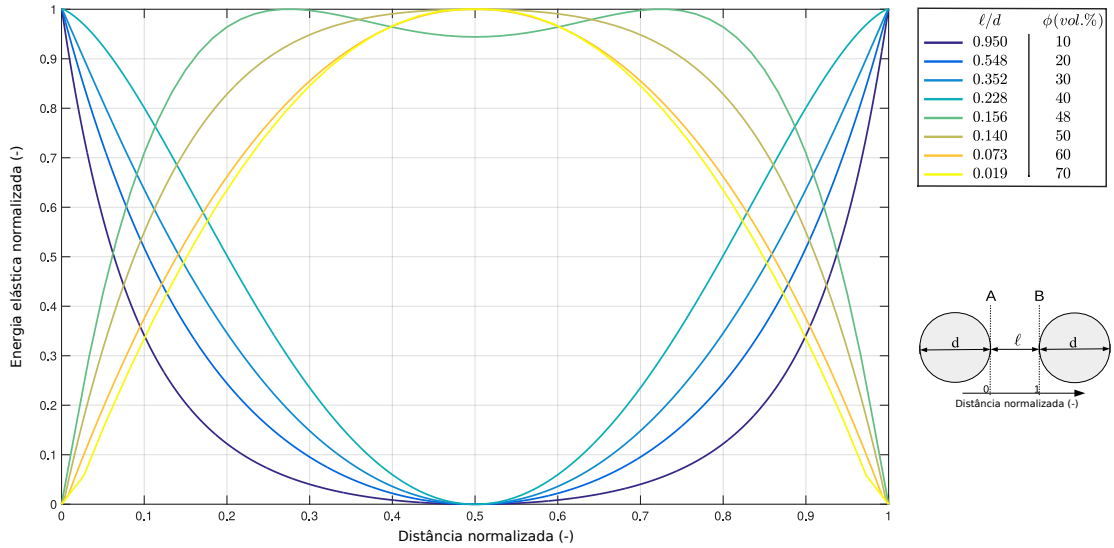

Figura 4.6 Energia normalizada para diferentes distâncias entre inclusões

Para elucidar a inversão de perfil apresentada na Figura 4.6, apresenta-se na Figura 4.7 a comparação entre as tensões máximas e mínimas principais e a energia entre a interface e o ponto médio para diversas relações ℓ/d. As regiões mais solicitadas para distintas relações são indicadas na Tabela 4.2. Nota-se que para $\ell/d > 0,29$ a interface matriz inclusão é o local energeticamente mais soli-

Tabela 4.2 Região mais solicitado para diferentes ℓ/d

	Região mais solicitada		
	Tensão max. prin.	**Tensão min. prin**	**Energia**
$\ell/d > 0,29$	Interface	Interface	Interface
$0,18 < \ell/d < 0,29$	Interface	Ponto médio	Interface
$0,11 < \ell/d < 0,18$	Interface	Ponto médio	Ponto médio
$0,06 < \ell/d < 0,11$	Ponto médio	Ponto médio	Ponto médio
$\ell/d < 0,06$	Ponto médio	Interface	Ponto médio

tado, motivado tanto pela tensão máxima, quanto pela mínima principal. Quando $\ell/d < 0,18$, há a inversão no local energeticamente mais solicitado, primeiramente causada pela mudança no comportamento da tensão mínima principal e, posteriormente, pela inversão da tensão máxima principal. Para $\ell/d < 0,06$ a interface torna-se mais solicitada em tensão mínima principal, mas o comportamento energético é majorado pela tensão máxima principal, que possui seu maior valor no ponto médio entre as inclusões, o que acarreta este local a ser a região energeticamente mais solicitada.

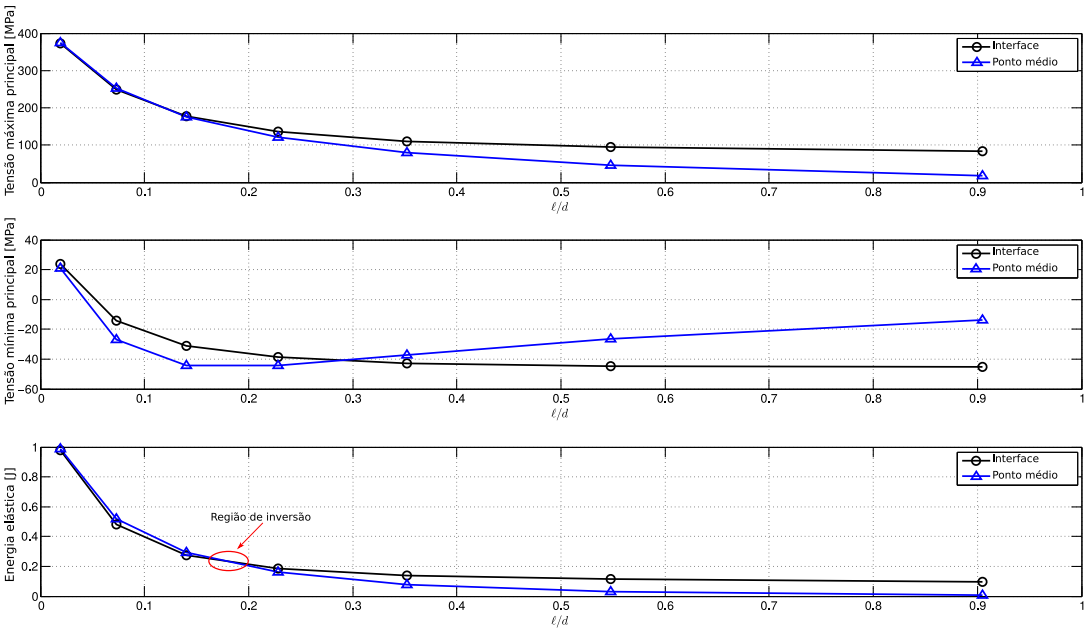

Figura 4.7 Tensão máxima principal, tensão mínima principal e energia versus ℓ/d

4.3 Estudo 3: Efeito das propriedades termomecânicas no perfil de distribuição de energia entre inclusões dispersas em matriz contínua

4.3.1 Análise 1: variação de propriedades da matriz e das inclusões

A partir da simulação das 28000 combinações de sistemas cerâmicos, foi possível identificar a região energeticamente mais solicitada entre as duas inclusões. Com o intuito de facilitar a interpretação dos resultados, esses são plotados a partir das variáveis auxiliares: $E^* = \frac{E_m^x}{E_i^x}$ e $\alpha^* = \frac{\alpha_m^x}{\alpha_i^x}$.

Como pode ser verificado na Fig. 4.8, são identificadas três regiões distintas no gráfico. A primeira, na cor verde, representa a região energeticamente mais solicitada no ponto médio entre as duas inclusões. A segunda, na cor azul, representa a região em que a interface matriz/inclusão é a energeticamente mais solicitada. Por fim, na cor laranja são representadas as regiões intermediárias entre a interface matriz/inclusão e o ponto médio entre duas inclusões, havendo então uma região de transição.

Deve-se ter em mente que na Figura 4.8 não há uma associação direta dos

resultados com o surgimento de trincas nestas regiões. Como já explicado, para que haja o trincamento, a energia elástica acumulada durante a deformação deve ser maior do que a energia de formação de superfícies. A partir da Figura 4.8, entende-se que uma vez que haja energia elástica armazenada o suficiente - e este fator é dependente da variação de temperatura (ΔT) - a expectativa da trinca ser nucleada na região energeticamente mais solicitada é maior do que em outras regiões, conforme discutido em trabalhos recentes [49, 50].

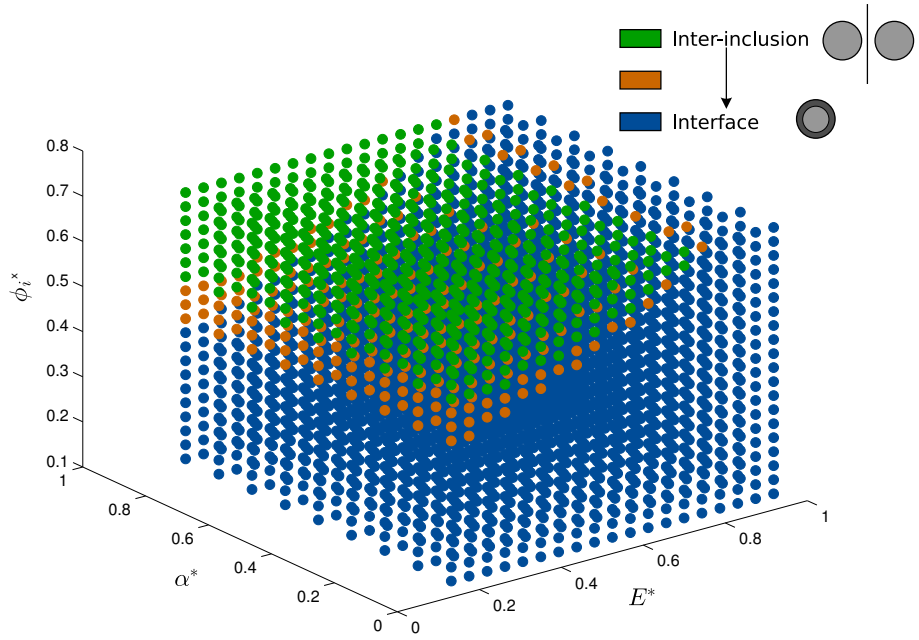

Figura 4.8 Região energeticamente mais solicitada entre duas inclusões após o resfriamento de amostras heterogêneas

Analisando-se a Figura 4.8, pode-se notar que há uma transição contínua entre as regiões inter-inclusões e interfaciais como energeticamente mais solicitadas e esta é dependente de E^*, α^* e ϕ_i^x. Portanto, a partir do software $Matlab^{TM}$, foi possível identificar uma aproximação com o coeficiente de determinação $R^2 \approx 0,99$, que representasse esta superfície de transição.

A função de aproximação da superfície de transição pode ser descrita como:

$$\phi_T = A1 \cdot E^{*2} + A2 \cdot E^* + B1 \cdot E^* \cdot (1 - \alpha^*) + C1 \cdot (1 - \alpha^*) + D1 \qquad (4.1)$$

Ao avaliar a Eq. 4.1, pode-se identificar que a superfície de transição é dependente quadraticamente de E^*, linearmente de α^*, além de ser dependente

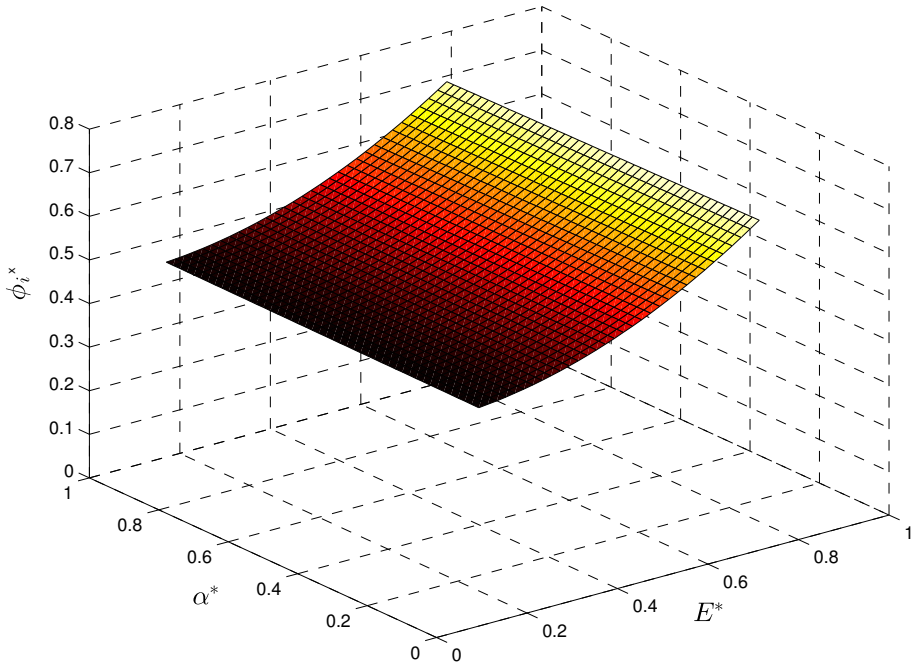

Figura 4.9 Superfície de transição interface/inter-inclusão analítica

Tabela 4.3 Coeficientes da superfície de transição interface/inter-inclusão

Coeficiente	**Valor (intervalo de confiança de 95%)**
A1	0.5133 (0.4689, 0.5577)
A2	-0.0771 (-0.1223, -0.0320)
B1	-0.0289 (-0.0584, 0.0007)
C1	0.0095 (-0.0052, 0.0241)
D1	0.4408 (0.4292, 0.4523)

linearmente da combinação de ambos os fatores. Assim, pode-se concluir que o fato da superfície depender quadraticamente de E^* e seu coeficiente $A1$ ser maior do que os coeficientes $B1$ e $C1$, implica que a razão E^* é mais influente na determinação da região energeticamente mais solicitada se comparada à α^* e ao produto de E^* e α^*.

4.3.2 Análise 2: variação de propriedades da inclusão mantendo-se fixa as da matriz

A partir das 3375 análises MEF, foram obtidos resultados em que a energia total armazenada durante a deformação não superou a energia de formação de superfícies (da Eq. 2.17: $U_s = 6.28\mu J$) para algumas combinações, portanto, não trincaram. Nas demais combinações, ocorreu essa superação de energia, resul-

tando em auto-trincamento.

Conforme pode ser observado na Figura 4.10, para elevados valores de α^* ($\alpha^* \geq 0,6$), não houve auto-trincamento, enquanto para menores valores de α^*, o sistema apresentou energia suficiente para trincar.

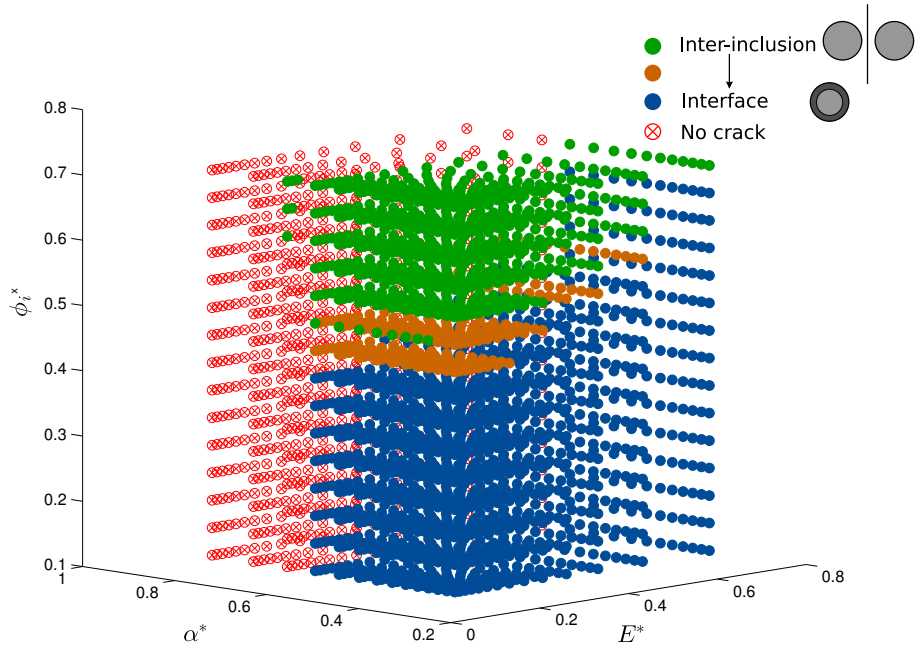

Figura 4.10 Trincamento ou não em diversas amostras submetidas a resfriamento

Com a intenção de compreender melhor a relevância de cada fator (α^* e E^*) na energia total armazenada, fixou-se a fração volumétrica de inclusões em $\phi_i = 0,11$ e, novamente utilizando-se o software $MATLAB^{TM}$, obteve-se uma expressão analítica para a superfície de energia (U_t) (Fig. 4.11), com fator de correlação $R^2 \approx 0,994$.

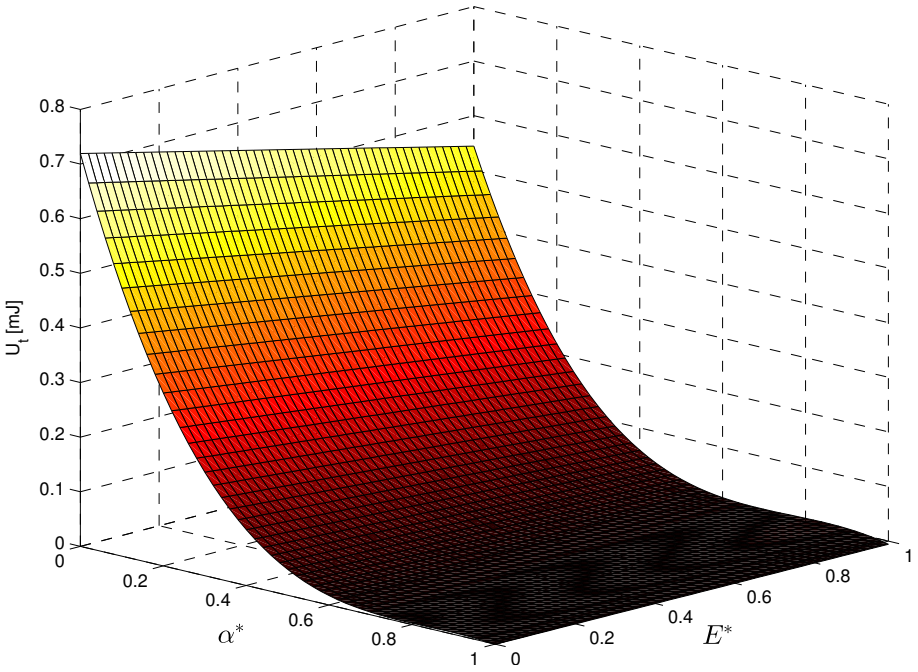

Figura 4.11

A expressão da superfície 4.11 é descrita como:

$$U_t = A11 \cdot E^* + B11 \cdot E^* \cdot \alpha^* + B12 \cdot E^* \cdot \alpha^{*2} + C11 \cdot \alpha^* + C22 \cdot \alpha^{*2} + C33 \cdot \alpha^{*3} + D11 \quad (4.2)$$

Tabela 4.4 Coeficientes superfície energia de deformação

Coeficiente	**Valor (intervalo de confiança de 95 %)**
A11	-0.175 (-0.1945, -0.1555)
B11	0.4364 (0.3726, 0.5003)
B12	-0.2665 (-0.3147, -0.2182)
C11	-2.583 (-2.67, -2.495)
C22	3.091 (2.958, 3.224)
C33	-1.228 (-1.293, -1.163)
D11	0.7194 (0.7008, 0.738)

Analisando a Eq. 4.2, identifica-se que a energia total armazenada após a deformação varia linearmente com E^* e cubicamente com α^*. Além disso, a partir dos coeficientes da Tabela 4.4, fica evidente a maior influência de α^* na energia total de deformação em relação a E^*. Desta maneira, atesta-se o papel de maior relevância da razão dos coeficientes de expansão térmica linear para o auto-trincamento de sistemas cerâmicos.

4.4 Estudo 4: Modelagem via elementos coesivos da fissuração espontânea de sistemas cerâmicos

Uma vez identificada a relação entre propriedades termomecânicas de fases no comportamento de sistemas cerâmicos, buscou-se aumentar a representatividade de modelos e, consequentemente, sua complexidade. Para isso, foi decidido investigar a aplicação de elementos coesivos na simulação de sistemas cerâmicos.

4.4.1 Análise 1: Estudo do comportamento de um único elemento coesivo

Na Figura 4.12 encontram-se os resultados do comportamento do elemento coesivo para as solicitações cíclicas impostas (tensões e deformações na direção y). Observa-se que logo no primeiro carregamento (a) há a danificação do elemento, verificada pela mudança de inclinação nas curvas deformação *versus* tempo e tensão *versus* tempo (a-b). Conforme há o carregamento cíclico controlado com deslocamentos prescritos, novas degradações das propriedades elásticas do elemento coesivo ocorrem. Deve-se notar que, após o primeiro carregamento, quando o elemento já havia sofrido danificação, o sistema é recarregado com a mesma rigidez do término do ciclo anterior, e só volta a se danificar quando a nova tensão crítica é atingida. Na região (h) há a total degradação do elemento coesivo, com sua deleção.

4.4.2 Análise 2: Estudo do efeito do tamanho de grão na fissuração espontânea de sistemas cerâmicos

Na Figura 4.13 são apresentadas as configurações inicial (T = 500°C), intermediária (T = 0°C) e final (T = 500°C) do *honeycomb* com os grãos de diâmetro 25 μm. Nota-se em vermelho os elementos coesivos deformados. Conforme a hipótese de Fertig & Nickerson [44], pode-se considerar que nos elementos coesivos deformados houve propagação de trinca, isto é, as propriedades mecânicas destas regiões foram degradadas. Conforme há o reaquecimento, a maior parte dos elementos coesivos volta às suas dimensões originais, neste caso Fer-

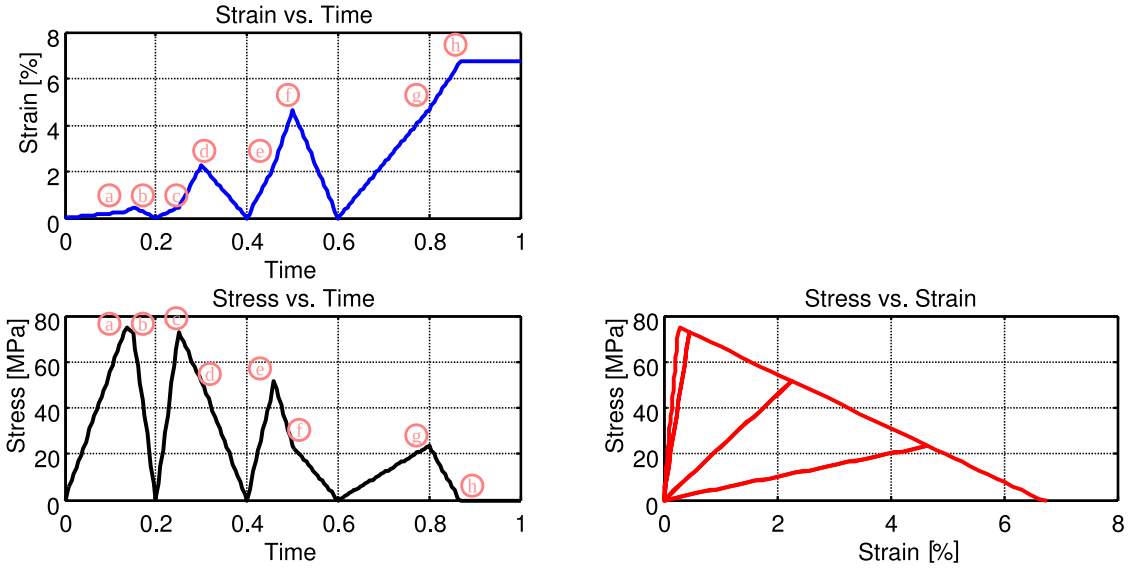

Figura 4.12 Resultados do modelo de um elemento coesivo (curvas tensão x deformação, tensão x tempo e deformação x tempo).

tig & Nickerson [44] sugerem que há a cicatrização da trinca. Para os casos em que há deleção do elemento, ou seja, as propriedades mecânicas foram totalmente degradadas, não existe a possibilidade de o elemento coesivo voltar às dimensões originais, pois já não há elemento coesivo no reaquecimento. A essas trincas, Fertig & Nickerson [44] dizem que são as residuais do tratamento térmico, ou seja, não podem cicatrizar. Na Figura 4.13 é possível notar que a maior parte das trincas foram cicatrizadas no reaquecimento, salvo alguns poucos elementos que foram deletados.

Na Figura 4.14 traz-se a comparação das configurações deformadas para os três diâmetros analisados. Fixou-se o ΔT = -250°C, pois a análise do modelo com D = 200 μm apresentou dificuldades de convergência. Portanto, selecionou-se um valor de ΔT em que a análise apresentou iterações estáveis. Pode-se notar uma maior amplitude de deformação conforme aumenta-se o diâmetro dos grãos. Para o modelo com D = 200 μm, destaca-se na Figura uma região em que houve deleção de elementos. Conclui-se, portanto, que nesta região não haverá cicatrização da trinca após o reaquecimento e assim, tem-se uma relação diretamente proporcional do tamanho de grão com a probabilidade de trincamento. Isto deve-se ao fato de o deslocamento do contorno de grão ser proporcional à di-

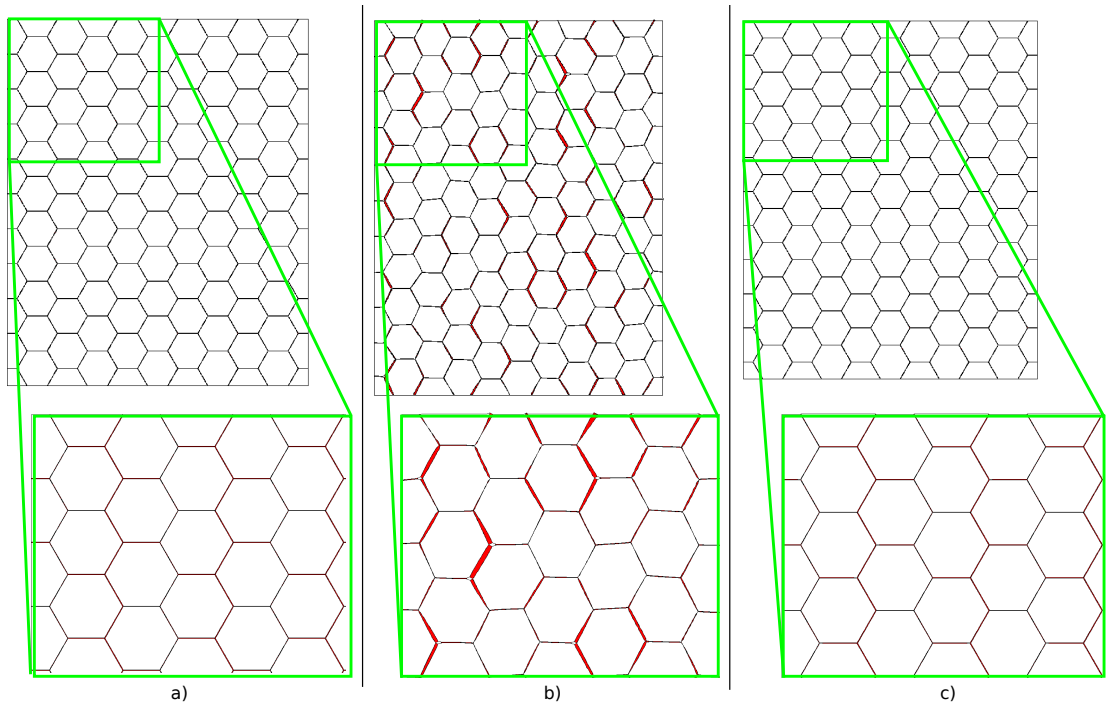

Figura 4.13 Configuração deformada dos grãos (D = 25μm).a) Configuração inicial b) Configuração deformada após ΔT de -500$^{\circ}C$ (amplificado em 50x) c) Configuração após retorno à temperatura inicial.

mensão do grão. Voltando-se à equação de expansão térmica ($\Delta L = L_0 \cdot \alpha \cdot \Delta T$), tem-se que quanto maior L_0, maior será o ΔL, mantendo-se α e ΔT fixos e consequentemente, maior será a tensão no elemento coesivo e maior a probabilidade de sua completa degradação.

É importante ressaltar que fora a deleção de elementos e a diferença na amplitude de deformação dos elementos coesivos, globalmente as regiões em que houveram maior deformação dos elementos coesivos foram as mesmas para os três modelos analisados. Destacando, portanto, que o início da degradação do elemento é causado principalmente pela orientação dos grãos vizinhos. E o estado final de degradação, com possível deleção, é associado ao tamanho de grão.

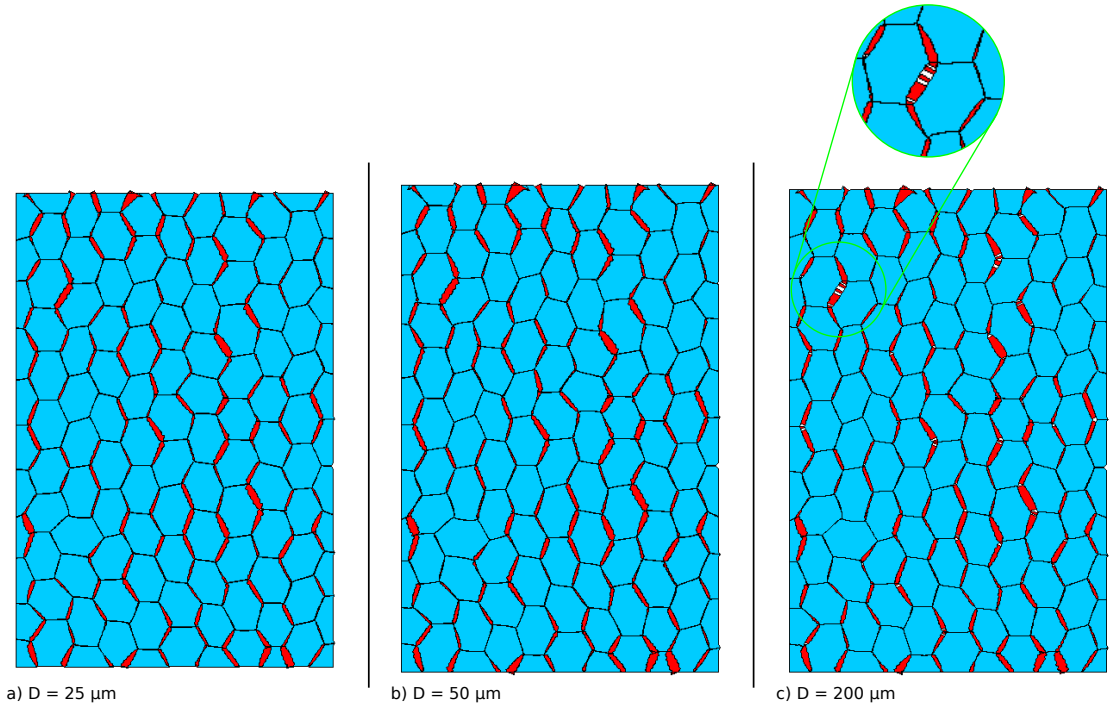

Figura 4.14 Configuração deformada após ΔT de -250$^{\circ}C$ (amplificado em 50x). Azul: elementos estruturais, vermelho: elementos coesivos, e branco: elementos coesivos deletados (σ = 15°)

4.4.3 Análise 3: Estudo do efeito da distribuição de orientações de grãos no comportamento de sistemas cerâmicos submetidos à variação de temperatura

A partir das quatro análises com diferentes distribuições de orientações (Seção 3.4.3, pág. 45), foi possível estimar o efeito da distribuição de orientações de grãos no comportamento global do *honeycomb* quando submetido à variação de temperatura. Conforme descrito na Seção 3.4.3.2 (pág. 46), a condição de periodicidade permitiu calcular facilmente os coeficientes de expansão térmica globais dos *honeycombs*. A Tabela 4.5 apresenta estes resultados. Nota-se que com uma distribuição mais homogênea de orientações, os coeficientes de expansão globais aproximam-se dos valores encontrados para o monocristal, uma vez que grande parte dos grãos encontrava-se com orientações próximas à do monocristal. Conforme é aumentado o desvio padrão da distribuição (σ), ou seja, tornando-se a distribuição mais heterogênea, os coeficientes de expansão térmica afastam-se dos valores do monocristal, porém, aproximam-se entre si.

Ou seja, por mais que o monocristal apresente um comportamento ortotrópico ($\alpha_a = \alpha_b \neq \alpha_c$), o comportamento global do *honeycomb* aproxima-se de um comportamento isotrópico no limite. Esta conclusão foi tirada a partir da tendência exposta na Tabela 4.5, na qual para $\sigma = 55°$, tem-se tanto α_x, como α_y positivos e com valores intermediários aos coeficientes do monocristal.

O fato de os resultados para $\sigma = 15°$ serem mais distantes dos valores do monocristal, se comparados aos resultados para $\sigma = 35°$, demonstram que o programa para distribuição aleatória de orientações deve ser melhor estruturado, uma vez que esperava-se o comportamento inverso. Porém, como a distribuição é aleatória, por mais que o programa busque manter a média e o desvio padrão impostos, pode ser que uma configuração que respeite estes valores globalmente apresente uma configuração mais heterogênea do que outra distribuição com um desvio padrão maior. Uma alternativa a este fenômeno seria aumentar o número de grãos, o que por sua vez aumentará o custo computacional.

Tabela 4.5 Coeficientes de expansão térmica globais nas direções x e y para os distintos desvios padrões

σ	α_x $[K^{-1}]$	α_y $[K^{-1}]$
0°	3.87 $\cdot 10^{-6}$	-1.20 $\cdot 10^{-6}$
5°	3.28 $\cdot 10^{-6}$	-1.00 $\cdot 10^{-6}$
15°	2.50 $\cdot 10^{-6}$	-2.70 $\cdot 10^{-8}$
35°	2.59 $\cdot 10^{-6}$	-1.25 $\cdot 10^{-7}$
55°	2.14 $\cdot 10^{-6}$	3.39 $\cdot 10^{-7}$

Para o desvio padrão 0° foram consideradas as propriedades do monocristal extraídas de Fertig & Nickerson [44]

Quanto ao comportamento global da fissuração, buscou-se apresentá-lo na Figura 4.15. Apela-se ao leitor o uso de sua análise qualitativa, uma vez que ainda não foi implementado um cálculo quantitativo para se comparar a densidade de trincas. Pode-se notar qualitativamente que a distribuição para $\sigma = 5°$ possui a menor densidade de trincas, enquanto a distribuição para $\sigma = 55°$ possui a maior densidade de trincas. As distribuições para $\sigma = 15°$ e $\sigma = 35°$ apresentam um resultado muito próximo, mas isto exige uma melhor abordagem na distribuição da orientação dos grãos.

Extrapolam-se, portanto, os resultados apresentados na Figura 4.15 para a conclusão de que quanto maior a heterogeneidade de distribuição de orientações, maior a probabilidade de trincar o material, considerando-se as mesmas propriedades da interface. Isto deve-se ao fato de quanto mais heterogênea a distribuição, mais fácil será dois grãos vizinhos possuírem orientações bem distintas, o que amplificaria as solicitações nos elementos coesivos.

O tempo de processamento de análises está intimamente vinculado com o grau de não linearidade no sistema. Por exemplo, a análise do modelo com desvio-padrão da distribuição de orientações $\sigma = 5°$ foi efetuada em aproximadamente 2 minutos. Enquanto a análise para o modelo com $\sigma = 55°$ durou aproximadamente 4 horas. Assim, por mais que as análises estejam automatizadas, o custo computacional apresenta-se como uma dificuldade deste tipo de simulação.

A partir de uma análise mais minuciosa da Figura 4.15 é possível notar uma maior tendência ao trincamento nas direções verticais. Isso deve-se a combinação de efeitos tração vertical e compressão horizontal, o que acarreta em tensões cisalhantes nas interfaces.

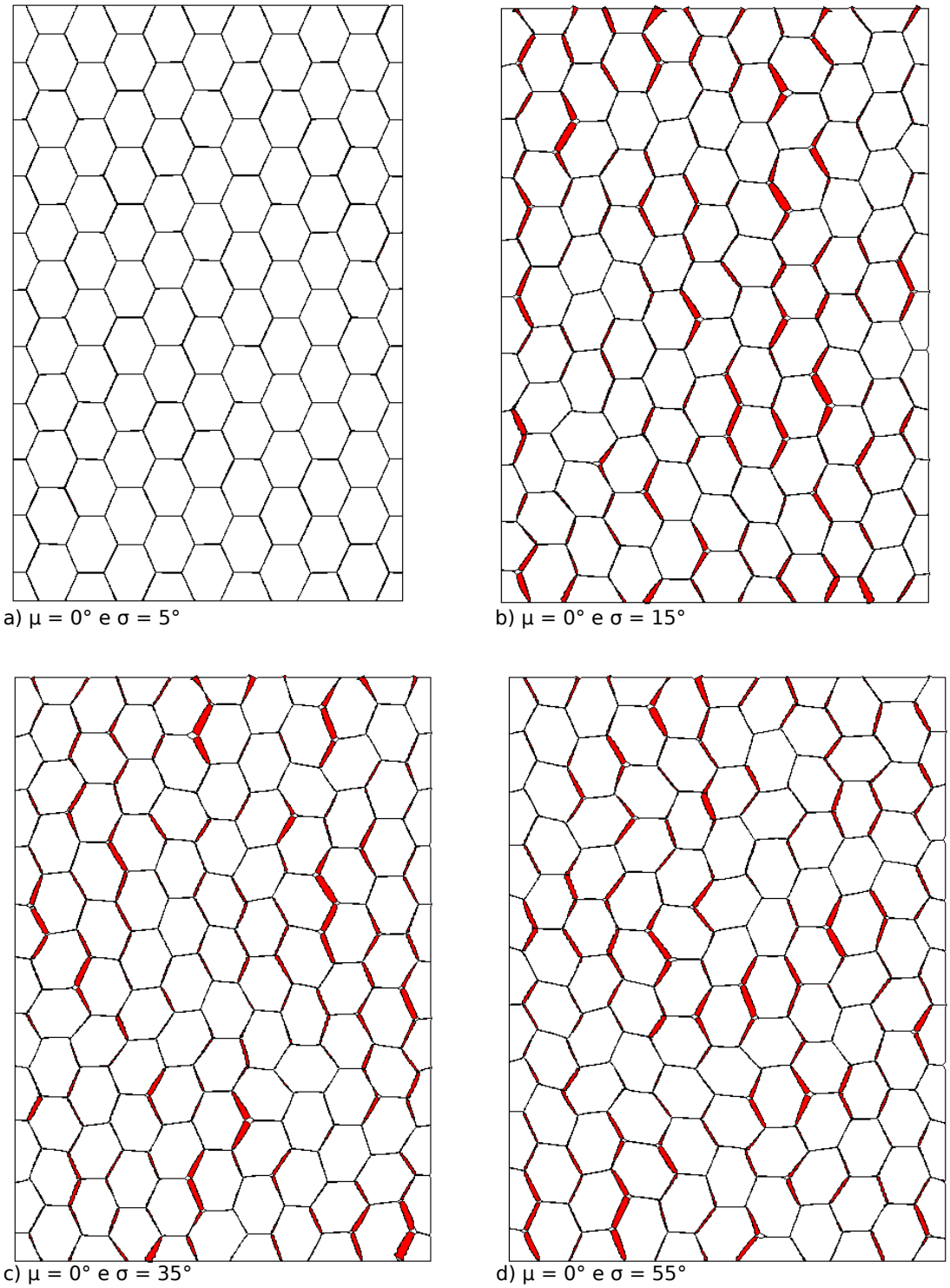

Figura 4.15 Configuração deformada após ΔT de -500°C para distribuições com $\mu = 0°$ e diferentes σ ($\sigma = 5, 15, 35$ e $55°$) (amplificado em 100x)

5 CONCLUSÕES

5.1 Estudo 1: Comparação entre raios críticos de inclusões de diversos modelos

Trabalhos da literatura, como Davidge e Green [9], desenvolveram expressões analíticas para previsão do raio crítico, apresentando uma boa capacidade de predição da ocorrência de microfissuração ou decoesão para materiais com baixas frações volumétricas ($\phi \leq 30$ vol.$\%$). Contudo, em virtude das hipóteses adotadas, como a de tensão normal média constante ao longo da interface matriz / inclusão, o modelo começa a perder representatividade a partir de frações volumétricas superiores ($\phi > 30$ vol.$\%$).

Simulações computacionais via elementos finitos podem ser utilizadas como ferramenta na previsão do raio crítico de inclusão. Os resultados aqui apresentados com os modelos axissimétricos em EF mostram coerência com resultados experimentais da literatura. Para frações volumétricas maiores que 30 vol.$\%$, análises via MEF mostraram que a tensão normal média ao longo da interface matriz / inclusão não é constante, como os modelos analíticos assumem. Adicionalmente, constatou-se a inversão da curva de distribuição de tensões mínimas principais entre inclusões, se comparada a modelos analíticos precedentes. Outra observação importante é a mudança de local de maior solicitação termomecânica durante o resfriamento para sistemas com maiores frações volumétricas. Para baixas frações volumétricas, o local mais solicitado é a interface matriz / inclusão, enquanto para frações volumétricas superiores, passa a ser o ponto médio entre inclusões, fornecendo indícios da explicação da origem de trincas entre estas. Sendo assim, as vantagens das simulações computacionais para melhor previsão do raio crítico de inclusão foram evidenciadas.

5.2 Estudo 2: Perfil de distribuição de energia entre inclusões para diferentes frações volumétricas

Foram realizadas simulações computacionais da etapa de resfriamento de compósitos cerâmicos para investigar o surgimento de trincas inter-inclusões.

Pautados no conceito de surgimento de trinca a partir do acúmulo de energia armazenada, foi possível traçar o perfil de contribuição energética das porções de matriz entre inclusões e constatar uma inversão neste comportamento. Para distância entre inclusões razoavelmente grandes ($\ell/d >$ 0,18), esta distribuição energética possui perfil "U", enquanto para pequenas, ele é invertido, deslocando-se a região mais solicitada para a porção central entre as inclusões. Constatações experimentais na literatura indicando a presença de trincas inter-inclusões para altas frações volumétricas de inclusão podem ser justificadas com a teoria aqui apresentada. É importante entender que o perfil energético é determinado geometricamente, ou seja, em função do raio e distância das inclusões, enquanto a variação de temperatura possui um papel na amplitude da energia. Assim, sob hipótese de interface perfeita entre as inclusões, pode-se afirmar que para razões $\ell/d <$ 0,18 ($\phi >$ 45 vol. $\%$) a região central entre inclusões é energeticamente mais solicitada, o que justificaria o surgimento de trincas nessa região, não previstas por modelos anteriores.

5.3 Estudo 3: Efeito das propriedades termomecânicas no perfil de distribuição de energia entre inclusões dispersas em matriz contínua

A partir da automatização de simulações computacionais, tornou-se possível analisar um grande número de combinações de propriedades termomecânicas, tanto da matriz quanto das inclusões, obtendo-se assim um espaço amostral adequado para compreender o papel de cada parâmetro na determinação da região energeticamente mais solicitada em sistemas cerâmicos submetidos à variação de temperatura. Foi identificada uma superfície de transição, acima da qual tem-se o ponto médio entre duas inclusões como sendo a região mais solicitada, enquanto abaixo desta tem-se a interface matriz e inclusão como região com maiores solicitações. A partir do *fitting* desta superfície, foi possível identificar que esta varia quadraticamente com a razão $\frac{E_m}{E_i}$ e linearmente com a razão $\frac{\alpha_m}{\alpha_i}$. Além disso, com simulações em que se mantiveram fixas as propriedades da matriz e foram variadas as propriedades das inclusões, foi possível obter uma superfície da energia total armazenada durante a deformação em função

das mesmas razões $\left(\frac{E_m}{E_i}, \frac{\alpha_m}{\alpha_i}\right)$. Notou-se que esta superfície varia linearmente com $\frac{E_m}{E_i}$ e cubicamente com $\frac{\alpha_m}{\alpha_i}$. Concluiu-se então que a razão $\frac{E_m}{E_i}$ apresenta papel preponderante na localização da região mais solicitada se comparada a $\frac{\alpha_m}{\alpha_i}$, enquanto a razão $\frac{\alpha_m}{\alpha_i}$ possui maior influência na energia total armazenada durante a deformação frente à $\frac{E_m}{E_i}$. Ou seja, $\frac{\alpha_m}{\alpha_i}$ está mais associado ao auto-trincamento, enquanto $\frac{E_m}{E_i}$ está mais associado à provável localização das trincas.

5.4 Estudo 4: Modelagem via elementos coesivos da fissuração espontânea de sistemas cerâmicos

Simulações computacionais utilizando elementos coesivos mostraram-se extremamente promissoras. Este tipo de metodologia aplicada a sistemas cerâmicos é muito recente e oferece aos pesquisadores um vasto campo a ser explorado. A partir das análises desenvolvidas, foi possível captar a influência do tamanho e distribuição de orientações de grãos de cordierita em seu comportamento em fratura. Os primeiros apresentam relação diretamente proporcional ao trincamento espontâneo das interfaces dos grãos, isto é, grãos maiores proporcionam maiores deslocamentos das interfaces quando sujeitos à variação térmica, o que acarreta em uma maior tensão nos elementos coesivos (interfaces dos grãos) e, consequentemente, maior trincamento. A partir das análises de distribuição de orientações de grãos, conclui-se que quanto mais heterogênea a distribuição, maior a densidade de trincas no sistema. Além disso, foi possível calcular-se a influência da distribuição de orientações no valor dos coeficientes de expansão térmica global do *honeycomb*. Isso evidenciou que um material com distribuição heterogênea de orientações apresenta α's globais intermediários ao do monocristal.

5.5 Conclusões gerais do trabalho

Globalmente o trabalho desenvolvido consiste na aplicação de modelos numéricos na simulação do comportamento de sistemas cerâmicos submetidos a

variação de temperatura. Fez-se inicialmente uma comparação de modelos analíticos com modelos computacionais, concluindo-se que os últimos oferecem uma maior diversidade de análise tendo em vista as hipóteses simplificadoras presentes nos primeiros. A partir deste estudo, novas pesquisas sobre os potenciais da aplicação do MEF na análise de sistemas cerâmicos foram desenvolvidas. Levantou-se uma teoria sobre o surgimento de trincas inter-inclusões, bem como foi analisado a sensibilidade de parâmetros das propriedades termomecânicas no comportamento de sistemas cerâmicos submetidos a variação de temperatura. Por fim, avaliou-se o emprego de elementos coesivos na simulação de auto-trincamento em *honeycombs* cerâmicas. A partir dos primeiros resultados tratados, foi possível identificar a influência do tamanho de grão e da distribuição de orientações destes no comportamento final da peça. Acredita-se que se alcançou uma metodologia eficiente de análise e que poderá ser aplicada para analisar uma infinidade de assuntos.

6 SUGESTÕES DE TRABALHOS FUTUROS

A maneira como este trabalho foi estruturado permite aos leitores, caso queiram, estudar mais detalhadamente cada tópico abordado. Como sugestão de continuação do estudo 1, sugere-se o aumento de complexidade dos modelos analisados. Pode-se variar as propriedades termomecânicas em função da temperatura e obter respostas mais próximas aos ensaios experimentais, bem como fazer simulações tridimensionais e considerar efeitos dependentes do tempo (condução, convecção e radiação). Quanto ao estudo 2, acredita-se que uma bateria de experimentos seja necessária para validar a teoria de trincas inter-inclusões aqui sugeridas. Entende-se que este estudo deva ser feito minuciosamente, uma vez que se trata de um fenômeno complexo. As sugestões de continuação do estudo 3 são no sentido de melhorar as equações analíticas que foram obtidas. Este estudo foi iniciado pelo grupo de pesquisa do Laboratório de Simulação Computacional (LSC) da UFSCar, mas carece de avanços, desejando-se obter um modelo "universal" que contemple a física do problema. Por último, e não menos importante, o estudo 4 mostra-se extremamente fértil para a aplicação em novos problemas. Infelizmente só foi possível avançar neste assunto nas semanas finais deste mestrado, porém diversos temas de análises já se mostram possíveis e acessíveis de serem analisados, como a criação de uma metodologia para avaliar a densidade de trincas em amostras, a avaliação do módulo de elasticidade global em função do trincamento, enfim. Resta aos possíveis pesquisadores explorar melhor os resultados destas análises e compará-los com experimentos.

7 REFERÊNCIAS BIBLIOGRÁFICAS

[1] CHEMISTRY, R. S. of. *Composite Materials*. 2004. Disponível em: <http://www.rsc.org/Education/Teachers/Resources/Inspirational/resources/4.3.1.pdf>.

[2] INCORPORATED, M. M.-B. *History of Composites*. 2016. Disponível em: <http://www.mar-bal.com/language/en/applications/history-of-composites/>.

[3] PALUCKA, T.; BENSAUDE-VINCENT, B. *Composites Overview*. 2002. Disponível em: <http://authors.library.caltech.edu/5456/1/hrst.mit.edu/hrs/materials/public/composites/Composites_Overview.htm>.

[4] TUAN, W.-H.; GILBART, E.; BROOK, R. J. Sintering of heterogeneous ceramic compacts. *Journal of materials science*, Springer, v. 24, n. 3, p. 1062–1068, 1989.

[5] SATO, I.; ICHIKAWA, Y.; SAKANOUE, J.; MIZUTANI, M.; ADACHI, N.; OTA, T. Flexible ceramics in the system kzr2 (po4) 3–kalsi2o6 prepared by mimicking the microstructure of itacolumite. *Journal of the American Ceramic Society*, Wiley Online Library, v. 91, n. 2, p. 607–610, 2008.

[6] KASELOW, A. *The stress sensitive approach: theory and application*. Tese (Doutorado) — Frein University of Berlin, 2004.

[7] THORNE, L.; WALLACE, T. C. Modern global seismology. *Internacional Geophysics Series*, v. 58, p. 375, 1995.

[8] TESSIER-DOYEN, N. *Etude experimentale et numerique du comportement thermomecanique de materiux refractaires modeles*. Tese (Doutorado) — Université de Limoges, 2003.

[9] DAVIDGE, R. W.; GREEN, T. J. The strength of two-phase ceramic/glass materials. *Journal of Materials Science*, Springer, v. 3, n. 6, p. 629–634, 1968.

[10] EVANS, A. The role of inclusions in the fracture of ceramic materials. *Journal of Materials Science*, Kluwer Academic Publishers, v. 9, n. 7, p. 1145–1152, 1974. ISSN 0022-2461.

[11] DAVIDGE, R. W. *Mechanical behaviour of ceramics*. 1. ed. [S.l.]: Syndics of the Cambridge University Press, 1979. ISBN 3257227892.

[12] ITO, Y.; ROSENBLATT, M.; CHENG, L.; LANGE, F.; EVANS, A. Cracking in particulate composites due to thermalmechanical stress. *International Journal of Fracture*, Kluwer Academic Publishers, v. 17, n. 5, p. 483–491, 1981. ISSN 0376-9429.

[13] KRSTIC, V. D. On the fracture of brittle-matrix/ductile-particle composites. *Philosophical Magazine A*, Taylor & Francis, v. 48, n. 5, p. 695–708, 1983.

[14] ORTIZ, M.; MOLINARI, A. Microstructural thermal stresses in ceramic materials. *Journal of the Mechanics and Physics of Solids*, v. 36, n. 4, p. 385 – 400, 1988.

[15] PAULIK, S. W.; ZIMMERMAN, M. H.; FABER, K. T.; FULLER, E. R. Residual stress in ceramics with large thermal expansion anisotropy. *Journal of materials research*, v. 11, p. 2795–2803, 1996. ISSN 2044-5326.

[16] PODREZOV, Y.; LUGOVOY, N.; SLYUNYAEV, V.; MINAKOV, N. Statistical failure model of materials with micro-inhomogeneity. *Theoretical and Applied Fracture Mechanics*, v. 26, p. 35 – 40, 1997. ISSN 0167-8442. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167844296000328>.

[17] SERBENA, F. C.; ZANOTTO, E. D. Internal residual stresses in glass-ceramics: A review. *Journal of Non-Crystalline Solids*, v. 358, n. 6–7, p. 975 – 984, 2012. ISSN 0022-3093. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0022309312000555>.

[18] SELSING, J. Internal stresses in ceramics. *Journal of the American Ceramic Society*, Wiley Online Library, v. 44, n. 8, p. 419–419, 1961.

[19] LIU, D.; WINN, E. J. Microstresses in particulate-reinforced brittle composites. *Journal of Materials Science*, Springer, v. 36, n. 14, p. 3487–3495, 2001.

[20] JOLIFF, Y.; ABSI, J.; GLANDUS, J. C.; HUGER, M.; TESSIER-DOYEN, N. Experimental and numerical study of the thermomechanical behaviour of refractory model materials. *Journal of the European Ceramic Society*, Elsevier, v. 27, n. 2, p. 1513–1520, 2007.

[21] SARKAR, K.; SUND, S. E.; BOSE, D.; YAMANIS, J. Thermal mismatch study in a ceramic to metal joint using the finite element method. *Mathematical and Computer Modelling*, Elsevier, v. 14, p. 842–848, 1990.

[22] NAM, T. H.; REQUENA, G.; DEGISCHER, P. Thermal expansion behaviour of aluminum matrix composites with densely packed sic particles. *Composites Part A: Applied Science and Manufacturing*, Elsevier, v. 39, n. 5, p. 856–865, 2008.

[23] YUTAKA, T.; JUNG-SIN, C. Computational damage mechanics models for brittle microcracking solids based on mesoscopic simulations. *Engineering fracture mechanics*, Elsevier, v. 48, n. 4, p. 483–498, 1994.

[24] CANNILLO, V.; LEONELLI, C.; BOCCACCINI, A. R. Numerical models for thermal residual stresses in al_2o_3 platelets/borosilicate glass matrix composites. *Materials Science and Engineering: A*, Elsevier, v. 323, n. 1, p. 246–250, 2002.

[25] CANNILLO, V.; LEONELLI, C.; MANFREDINI, T.; MONTORSI, M.; VERONESI, P.; MINAY, E. J.; BOCCACCINI, A. R. Mechanical performance and fracture behaviour of glass–matrix composites reinforced with molybdenum

particles. *Composites science and technology*, Elsevier, v. 65, n. 7, p. 1276–1283, 2005.

[26] DIMITRIJEVIĆ, M. M.; MEDJO, B.; HEINEMANN, R. J.; RAKIN, M.; VOLKOV-HUSOVIĆ, T. Experimental and numerical analysis of thermal shock damages to alumina based ceramic disk samples. *Materials & Design*, Elsevier, v. 50, p. 1011–1018, 2013.

[27] PEZZOTTA, M.; ZHANG, Z. L.; JENSEN, M.; GRANDE, T.; EINARSRUD, M.-A. Cohesive zone modeling of grain boundary microcracking induced by thermal anisotropy in titanium diboride ceramics. *Computational Materials Science*, Elsevier, v. 43, n. 3, p. 440–449, 2008.

[28] GILABERT, F. A.; BÓ, M. D.; CANTAVELLA, V.; SÁNCHEZ, E. Fracture patterns of quartz particles in glass feldspar matrix. *Materials Letters*, Elsevier, v. 72, p. 148–152, 2012.

[29] JOLIFF, Y.; ABSI, J.; HUGER, M.; GLANDUS, J. C. Microcracks with unexpected characteristics induced by cte mismatch in two-phase model materials. *Journal of Materials Science*, Springer, v. 43, n. 1, p. 330–337, 2008.

[30] ROSENFELD, J. L. Rotated garnets in metamorphic rocks. *Geological Society of America Special Papers*, Geological Society of America, v. 129, p. 1–102, 1970.

[31] REDDY, J. N. *An introduction to the finite element method*. [S.l.]: McGraw-Hill New York, 1993.

[32] BATHE, K.-J. *Finite element procedures*. [S.l.]: Klaus-Jurgen Bathe, 2006.

[33] HRENNIKOFF, A. Solution of problems of elasticity by the framework method. *Journal of applied mechanics*, v. 8, n. 4, p. 169–175, 1941.

[34] MCHENRY, D. A lattice analogy for the solution of plane stress problems. *Journal of Institution of Civil Engineers*, v. 21, p. 59–82, 1943.

[35] TURNER, M. Large deflections of structures subjected to heating and external loads. *Journal of the Aerospace Sciences*, 1960.

[36] GALLAGHER, R. H. Stress analysis of heated complex shapes. *ARS Journal*, v. 32, n. 5, p. 700–707, 1962.

[37] GALLAGHER, R. H.; PADLOG, J. Discrete element approach to structural instability analysis. *AIAA Journal*, v. 1, n. 6, p. 1437–1439, 1963.

[38] ZIENKIEWICZ, O.; WATSON, M.; KING, I. A numerical method of visco-elastic stress analysis. *International Journal of Mechanical Sciences*, Elsevier, v. 10, n. 10, p. 807–827, 1968.

[39] ABAQUS-6.14. *Abaqus 6.14 Documentation*. Providence, RI, USA., 2014.

[40] PARK, K.; PAULINO, G. H. Cohesive zone models: a critical review of traction-separation relationships across fracture surfaces. *Applied Mechanics Reviews*, American Society of Mechanical Engineers, v. 64, n. 6, p. 060802, 2011.

[41] OSA, M. R. De la; ESTEVEZ, R.; OLAGNON, C.; CHEVALIER, J.; VIGNOUD, L.; TALLARON, C. Cohesive zone model and slow crack growth in ceramic polycrystals. *International journal of fracture*, Springer, v. 158, n. 2, p. 157–167, 2009.

[42] OSA, M. R. D. L.; ESTEVEZ, R.; OLAGNON, C.; CHEVALIER, J.; TALLARON, C. Cohesive zone model for intergranular slow crack growth in ceramics: influence of the process and the microstructure. *Modelling and Simulation in Materials Science and Engineering*, IOP Publishing, v. 19, n. 7, p. 074009, 2011.

[43] NGUYEN, B. N.; KOEPPEL, B. J.; AHZI, S.; KHALEEL, M. A.; SINGH, P. Crack growth in solid oxide fuel cell materials: from discrete to continuum damage modeling. *Journal of the American Ceramic Society*, Wiley Online Library, v. 89, n. 4, p. 1358–1368, 2006.

[44] FERTIG, R. S.; NICKERSON, S. T. Towards prediction of thermally induced microcrack initiation and closure in porous ceramics. *Journal of the American Ceramic Society*, Wiley Online Library, 2015.

[45] TODD, R. I.; DERBY, B. Thermal stress induced microcracking in alumina–20% sic p composites. *Acta materialia*, Elsevier, v. 52, n. 6, p. 1621–1629, 2004.

[46] TESSIER-DOYEN, N.; GLANDUS, J. C.; HUGER, M. Untypical young's modulus evolution of model refractories at high temperature. *Journal of the European Ceramic Society*, Elsevier, v. 26, n. 3, p. 289–295, 2006.

[47] HASSELMAN, D. Elastic energy at fracture and surface energy as design criteria for thermal shock. *Journal of the American Ceramic Society*, Wiley Online Library, v. 46, n. 11, p. 535–540, 1963.

[48] BUBECK, C. Direction dependent mechanical properties of extruded cordierite honeycombs. *Journal of the European Ceramic Society*, Elsevier, v. 29, n. 15, p. 3113–3119, 2009.

[49] LUCHINI, B.; SCIUTI, V. F.; ANGéLICO, R. A.; CANTO, R. B.; PANDOLFELLI, V. C. Critical inclusion size prediction in refractory ceramics via finite element simulations. *Ceramics International*, Wiley Online Library, 2016.

[50] LUCHINI, B.; SCIUTI, V.; ANGÉLICO, R.; CANTO, R.; PANDOLFELLI, V. Thermal expansion mismatch inter-inclusion cracking in ceramic systems. *Ceramics International*, Elsevier, v. 42, n. 10, p. 1512–1515, 2016.

APÊNDICE A

1 Main

```
clear all
close all
clc

R = [21]*1e-3;
vf = 0.10;
DT = 545;
gamma_s = 4;

fa = 1.0;
k = 1;
while (abs(fa(end)) > 1e-3 && k < 10)
    energy_fem(k) = call_abq_model(R(k), vf, DT);
    energy_sup(k) = 2 * 4 * pi * R(k)^2 * gamma_s;
    fa(k) = energy_sup(k) - energy_fem(k);
    fa_da = 2 * 4 * pi * (1.001*R(k))^2 * gamma_s -
        call_abq_model(1.001*R(k), vf, DT);
    deriv = (fa_da - fa(k)) / (0.001 * R(k));
    R(k+1) = R(k) - fa(k) / deriv;
    k = k + 1;
    fa
end
```

2 Call abq model

```
function energy = call_abq_model(R, vf,dT)

unix('rm -f params.py');

fp = fopen('params.py','w');
fprintf(fp, ['R = ' num2str(R) '\n']);
fprintf(fp, ['vf = ' num2str(vf) '\n']);
fprintf(fp, ['dT = ' num2str(dT) '\n']);
fclose(fp);

disp(['----------Computing - R = ' num2str(R*1e3) ' um
    ----------']);
unix('/opt/abaqus/Commands/abaqus cae noGUI=model.py &&
    echo Analysis completed.' );

if exist('energy.txt', 'file')
    fp = fopen('energy.txt');
    data = textscan(fp, '%f %f', 'Headerlines', 3);
```

```
17      fclose(fp);
18
19      energy = data{2};
20      energy = energy(end);
21
22      unix('rm -f energy.txt');
23  else
24      energy = 0.0;
25  end
26
27  end
28
29  unix('rm -f params.py');
30
31  fp = fopen('params.py','w');
32  fprintf(fp, ['E = '  num2str(E) '\n']);
33  fprintf(fp, ['nu = '  num2str(nu) '\n']);
34  fclose(fp);
```

3 Params

```
1  R = 0.021639
2  vf = 0.1
3  dT = 545
```

4 Model

```
1
2  from abaqus import *
3  from abaqusConstants import *
4  session.Viewport(name='Viewport: 1', origin=(0.0, 0.0),
       width=304.006225585938, height=193.694442749023)
5  session.viewports['Viewport: 1'].makeCurrent()
6  session.viewports['Viewport: 1'].maximize()
7  from caeModules import *
8  from driverUtils import executeOnCaeStartup
9  executeOnCaeStartup()
10 session.viewports['Viewport: 1'].partDisplay.
       geometryOptions.setValues(
11      referenceRepresentation=ON)
12 Mdb()
13
14 #*************************************************
15 # PROBLEM PARAMETERS
16 #*************************************************
```

```
17
18 execfile('params.py')
19
20 w = 20.0 * R
21 h = 2.0*R*vf**(-1.0/3.0)
22
23 E_i = 250e6
24 nu_i = 0.27
25 alpha_i = 8.7e-6
26
27 E_m = 70e6
28 nu_m = 0.2
29 alpha_m = 3.6e-6
30
31
32
33
34 #**************************************************
35 # GEOMETRY
36 #**************************************************
37 s = mdb.models['Model-1'].ConstrainedSketch(name='
       __profile__', sheetSize=0.001)
38 g, v, d, c = s.geometry, s.vertices, s.dimensions, s.
       constraints
39 s.sketchOptions.setValues(decimalPlaces=5, viewStyle=
       AXISYM)
40 s.setPrimaryObject(option=STANDALONE)
41 s.ConstructionLine(point1=(0.0, -0.0005), point2=(0.0,
       0.0005))
42 s.FixedConstraint(entity=g[2])
43 s.rectangle(point1=(0.0, -h/2.0), point2=(w, h/2.0))
44 p = mdb.models['Model-1'].Part(name='Part-1',
       dimensionality=AXISYMMETRIC, type=DEFORMABLE_BODY)
45 p = mdb.models['Model-1'].parts['Part-1']
46 p.BaseShell(sketch=s)
47 s.unsetPrimaryObject()
48 p = mdb.models['Model-1'].parts['Part-1']
49 del mdb.models['Model-1'].sketches['__profile__']
50 p = mdb.models['Model-1'].parts['Part-1']
51 f, e, d1 = p.faces, p.edges, p.datums
52 t = p.MakeSketchTransform(sketchPlane=f[0],
       sketchPlaneSide=SIDE1, origin=(0.0, 0.0, 0.0))
53 s1 = mdb.models['Model-1'].ConstrainedSketch(name='
       __profile__',
54       sheetSize=6151.58, gridSpacing=153.78, transform=t)
55 g, v, d, c = s1.geometry, s1.vertices, s1.dimensions, s1.
```

```
    constraints
s1.sketchOptions.setValues(decimalPlaces=5)
s1.setPrimaryObject(option=SUPERIMPOSE)
p = mdb.models['Model-1'].parts['Part-1']
p.projectReferencesOntoSketch(sketch=s1, filter=
    COPLANAR_EDGES)
s1.CircleByCenterPerimeter(center=(0.0, h/2.0), point1=(R
    , h/2.0))
s1.CircleByCenterPerimeter(center=(0.0, -h/2.0), point1=(
    R, -h/2.0))
s1.Line(point1=(-0.1*w, 0), point2=(1.1*w,0.0))
p = mdb.models['Model-1'].parts['Part-1']
f = p.faces
pickedFaces = f.getSequenceFromMask(mask=('[#1 ]', ), )
e1, d2 = p.edges, p.datums
p.PartitionFaceBySketch(faces=pickedFaces, sketch=s1)
s1.unsetPrimaryObject()
del mdb.models['Model-1'].sketches['__profile__']

#**************************************************
# MATERIAL AND SECTION
#**************************************************
mdb.models['Model-1'].Material(name='inclusion')
mdb.models['Model-1'].materials['inclusion'].Elastic(
    table=((E_i, nu_i), ))
mdb.models['Model-1'].materials['inclusion'].Expansion(
    table=((alpha_i, ), ))
mdb.models['Model-1'].Material(name='matrix')
mdb.models['Model-1'].materials['matrix'].Elastic(table
    =((E_m, nu_m), ))
mdb.models['Model-1'].materials['matrix'].Expansion(table
    =((alpha_m, ), ))
mdb.models['Model-1'].HomogeneousSolidSection(name='
    inclusion', material='inclusion', thickness=None)
mdb.models['Model-1'].HomogeneousSolidSection(name='
    matrix', material='matrix', thickness=None)

p = mdb.models['Model-1'].parts['Part-1']
f_inc1 = p.faces.findAt(((0.1*R,-0.9*h/2.0,0.0),))
f_inc2 = p.faces.findAt(((0.1*R,0.9*h/2.0,0.0),))
p.SectionAssignment(region=(f_inc1+f_inc2,), sectionName=
    'inclusion', offset=0.0,
     offsetType=MIDDLE_SURFACE, offsetField='',
         thicknessAssignment=FROM_SECTION)

f_mat1 = p.faces.findAt(((0.9*w,-0.9*h/2.0,0.0),))
```

```
f_mat2 = p.faces.findAt(((0.9*w,0.9*h/2.0,0.0),))
p.SectionAssignment(region=(f_mat1+f_mat2,), sectionName='matrix', offset=0.0,
    offsetType=MIDDLE_SURFACE, offsetField='',
    thicknessAssignment=FROM_SECTION)

#**************************************************
# ASSEMBLY
#**************************************************
a = mdb.models['Model-1'].rootAssembly
a.DatumCsysByThreePoints(coordSysType=CYLINDRICAL, origin=(0.0, 0.0, 0.0),
    point1=(1.0, 0.0, 0.0), point2=(0.0, 0.0, -1.0))
p = mdb.models['Model-1'].parts['Part-1']
a.Instance(name='Part-1-1', part=p, dependent=ON)

#**************************************************
# STEP
#**************************************************
mdb.models['Model-1'].StaticStep(name='cooling', previous='Initial', initialInc=0.1, maxInc=0.1)

#**************************************************
# LOAD
#**************************************************
mdb.models['Model-1'].TabularAmplitude(name='Amp-1', timeSpan=STEP, smooth=SOLVER_DEFAULT, data=((0.0, 1.0), (1.0, 0.0)))
a = mdb.models['Model-1'].rootAssembly
faces = a.instances['Part-1-1'].faces
region = a.Set(faces=faces, name='full_region')
mdb.models['Model-1'].Temperature(name='initial', createStepName='Initial',
    region=region, distributionType=UNIFORM,
    crossSectionDistribution=CONSTANT_THROUGH_THICKNESS, magnitudes=(dT, ))
mdb.models['Model-1'].Temperature(name='initial_temperature',
    createStepName='cooling', region=region,
    distributionType=UNIFORM,
    crossSectionDistribution=CONSTANT_THROUGH_THICKNESS, magnitudes=(dT, ),
    amplitude='Amp-1')

a = mdb.models['Model-1'].rootAssembly
edges = a.instances['Part-1-1'].edges
```

```
126 ed1 = edges.findAt(((0.0,-0.9*h/2.0,0.0),))
127 ed2 = edges.findAt(((0.0,-0.05*h/2.0,0.0),))
128 ed3 = edges.findAt(((0.0, 0.05*h/2.0,0.0),))
129 ed4 = edges.findAt(((0.0, 0.9*h/2.0,0.0),))
130
131 region = a.Set(edges=ed1+ed2+ed3+ed4, name='Set-3')
132 mdb.models['Model-1'].DisplacementBC(name='BC-1',
        createStepName='Initial',
133      region=region, u1=SET, u2=UNSET, ur3=UNSET, amplitude
            =UNSET,
134      distributionType=UNIFORM, fieldName='', localCsys=
            None)
135 vertices = a.instances['Part-1-1'].vertices
136 vert = vertices.findAt(((0.0,0.0,0.0),))
137 region = a.Set(vertices=vert, name='Set-4')
138 mdb.models['Model-1'].DisplacementBC(name='BC-2',
        createStepName='Initial',
139      region=region, u1=UNSET, u2=SET, ur3=UNSET, amplitude
            =UNSET,
140      distributionType=UNIFORM, fieldName='', localCsys=
            None)
141
142 #**************************************************
143 # MESH
144 #**************************************************
145 p = mdb.models['Model-1'].parts['Part-1']
146 f = p.faces
147 pickedRegions = f.getSequenceFromMask(mask=('[#7 ]', ), )
148 p.setMeshControls(regions=pickedRegions, technique=
        STRUCTURED)
149 p.seedPart(size=R/20.0, deviationFactor=0.1,
        minSizeFactor=0.1)
150 p.setMeshControls(regions=pickedRegions, technique=
        STRUCTURED)
151 p.generateMesh()
152 a1 = mdb.models['Model-1'].rootAssembly
153 a1.regenerate()
154
155
156 #**************************************************
157 # INTERACTION
158 #**************************************************
159 a = mdb.models['Model-1'].rootAssembly
160 edges = a.instances['Part-1-1'].edges
161 ed_top1 = edges.findAt(((0.1*R,h/2.0,0.0),))
162 ed_top2 = edges.findAt(((1.1*R,h/2.0,0.0),))
```

```
163 a.Set(edges=ed_top1+ed_top2, name='top_face')
164 a.Set(name='top_nodes', nodes=a.sets['top_face'].nodes)
165
166 edges = a.instances['Part-1-1'].edges
167 ed_top1 = edges.findAt(((0.1*R,-h/2.0,0.0),))
168 ed_top2 = edges.findAt(((1.1*R,-h/2.0,0.0),))
169 a.Set(edges=ed_top1+ed_top2, name='bottom_face')
170 a.Set(name='bottom_nodes', nodes=a.sets['bottom_face'].
        nodes)
171
172 vtx = a.instances['Part-1-1'].vertices
173 top_vertice = vtx.findAt(((0.0, h/2.0, 0.0),))
174 a.Set(vertices=top_vertice, name='top_vertice')
175 a.Set(name='top_ref_node', nodes=a.sets['top_vertice'].
        nodes)
176
177 vtx = a.instances['Part-1-1'].vertices
178 top_vertice = vtx.findAt(((0.0, -h/2.0, 0.0),))
179 a.Set(vertices=top_vertice, name='bottom_vertice')
180 a.Set(name='bottom_ref_node', nodes=a.sets['
        bottom_vertice'].nodes)
181
182 a.SetByBoolean(name='top_nodes2', sets=(a.sets['top_nodes
        '],a.sets['top_ref_node'],), operation=DIFFERENCE)
183 a.SetByBoolean(name='bottom_nodes2', sets=(a.sets['
        bottom_nodes'],a.sets['bottom_ref_node'],), operation=
        DIFFERENCE)
184
185 mdb.models['Model-1'].Equation(name='eq_top', terms
        =((1.0, 'top_nodes2', 2), (-1.0, 'top_ref_node', 2)))
186 mdb.models['Model-1'].Equation(name='eq_bottom', terms
        =((1.0, 'bottom_nodes2', 2), (-1.0, 'bottom_ref_node'
        , 2)))
187
188 #**************************************************
189 # CREATE THE JOB
190 #**************************************************
191 mdb.Job(name='abq_periodic', model='Model-1', description
        ='', type=ANALYSIS,
192     atTime=None, waitMinutes=0, waitHours=0, queue=None,
            memory=90,
193     memoryUnits=PERCENTAGE, getMemoryFromAnalysis=True,
194     explicitPrecision=SINGLE, nodalOutputPrecision=FULL,
            echoPrint=OFF,
195     modelPrint=OFF, contactPrint=OFF, historyPrint=OFF,
            userSubroutine='',
```

```
196       scratch='', resultsFormat=ODB, multiprocessingMode=
              DEFAULT, numCpus=4,
197       numDomains=4, numGPUs=0)
198
199   #*************************************************
200   # SAVE THE MODEL
201   #*************************************************
202   mdb.saveAs(pathName='./abq_periodic')
203
204   #*************************************************
205   # RUN THE JOB
206   #*************************************************
207   mdb.jobs['abq_periodic'].submit(consistencyChecking=OFF)
208   mdb.jobs['abq_periodic'].waitForCompletion()
209
210
211   #*************************************************
212   # GET RESULTS
213   #*************************************************
214
215   o3 = session.openOdb(name='./abq_periodic.odb')
216   session.viewports['Viewport: 1'].setValues(
          displayedObject=o3)
217
218   session.Path(name='path', type=RADIAL, expression=((1E
          -05*R, h/2.0, 1.0), (1E-05*R, h/2.0, 0), (1E-05*R, -h
          /2.0, 0)), circleDefinition=ORIGIN_AXIS,
219       numSegments=100, radialAngle=0, startRadius=0,
              endRadius=CIRCLE_RADIUS)
220
221   pth = session.paths['path']
222
223   session.viewports['Viewport: 1'].odbDisplay.
          setPrimaryVariable(variableLabel='S', outputPosition=
          INTEGRATION_POINT, refinement=(COMPONENT, 'S11'))
224   session.XYDataFromPath(name='data_s11', path=pth,
          includeIntersections=False, projectOntoMesh=False,
          pathStyle=UNIFORM_SPACING, numIntervals=100,
          projectionTolerance=0, shape=UNDEFORMED, labelType=
          NORM_DISTANCE)
225
226   session.viewports['Viewport: 1'].odbDisplay.
          setPrimaryVariable(variableLabel='S', outputPosition=
          INTEGRATION_POINT, refinement=(COMPONENT, 'S12'))
227   session.XYDataFromPath(name='data_s12', path=pth,
          includeIntersections=False, projectOntoMesh=False,
```

```
    pathStyle=UNIFORM_SPACING, numIntervals=100,
    projectionTolerance=0, shape=UNDEFORMED, labelType=
    NORM_DISTANCE)
228
229 session.viewports['Viewport: 1'].odbDisplay.
    setPrimaryVariable(variableLabel='S', outputPosition=
    INTEGRATION_POINT, refinement=(COMPONENT, 'S22'))
230 session.XYDataFromPath(name='data_s22', path=pth,
    includeIntersections=False, projectOntoMesh=False,
    pathStyle=UNIFORM_SPACING, numIntervals=100,
    projectionTolerance=0, shape=UNDEFORMED, labelType=
    NORM_DISTANCE)
231
232 x0 = session.xyDataObjects['data_s11']
233 x1 = session.xyDataObjects['data_s22']
234 x2 = session.xyDataObjects['data_s12']
235
236 session.writeXYReport(fileName='response.txt', appendMode
    =OFF, xyData=(x0, x1, x2))
237
238 session.XYDataFromHistory(name='energy', odb=o3,
    outputVariableName='Strain energy: ALLSE for Whole
    Model', steps=('cooling', ), )
239 energy = session.xyDataObjects['energy']
240 session.writeXYReport(fileName='energy.txt', appendMode=
    OFF, xyData=(energy))
```

APÊNDICE B

1 Analysis control

```
clear all
close all
clc


%% Data

% Volume fraction
PHI = linspace(0.11, 0.70, 20);

% Young modulus
Em_min = 68e3;
Em_max = 100e3;
E_ratio = linspace(Em_min/Em_max, Em_max/Em_max, 20);

% Linear expansion coefficient
alpha_min = 0.5e-6;
alpha_max = 10e-6;
D_alpha = linspace(0, alpha_max-alpha_min, 10);
D_alpha=D_alpha(1,1:10);

ncase = 1;
tot_case = length(PHI) * length(E_ratio) * length(D_alpha
    );
for phi_aux = PHI

    for E_aux = E_ratio
        E1 = Em_max;
        E2 = E1 * E_aux;

        for alpha_aux = D_alpha
            AF1 = 10e-6;
            AF2 = AF1 - alpha_aux;

            % Model parameters
            L1=0.7;
            L2=0.975;
            A=250e-3;
            B=0.905*A*(phi_aux^(-1.0/3.0));
            C=B+A;
            D=0.005;
            F=B-A;
            G=2.0*B;
```

```
43          N1= 0.24;
44          N2= 0.2;
45          TEMP = 300.0;
46
47          % Params to input file
48          params = [L1 L2 A B C D F G E1 N1 AF1 E2 N2
                AF2 TEMP];
49
50          % Write input file
51          write_input_file(params);
52
53          disp('
                ==========================================
                ');
54          disp(['Ncase = ' num2str(ncase) '/' num2str(
                tot_case) ' | Phi = ' num2str(phi_aux) ' |
                 E_ratio = ' num2str(E_aux) ' | D_Alpha =
                ' num2str(alpha_aux) ]);
55          porcent=100*(ncase/tot_case);
56          disp(['...' num2str(porcent) '%']);
57          disp('
                ==========================================
                ');
58
59          % Call abaqus
60
61          unix('rm Jo*');
62          unix('rm aba*');
63          unix('rm energy.txt');
64          unix('/opt/abaqus/Commands/abaqus cae noGUI=
                m2_automatic.py && echo END');
65
66          % Read result
67          fp = fopen('energy.txt');
68          data = textscan(fp, '%f %f', 'headerlines',3)
                ;
69          fclose(fp);
70
71          x_norm = data{1};
72          x_norm = x_norm(1:end-1);
73          energy = data{2};
74          energy = energy(1:end-1);
75
76
77          analysis(ncase).phi = phi_aux;
78          analysis(ncase).N1 = N1;
```

```
                analysis(ncase).N2 = N2;
                analysis(ncase).E1 = E1;
                analysis(ncase).E2 = E2;
                analysis(ncase).AF1 = AF1;
                analysis(ncase).AF2 = AF2;
                analysis(ncase).xnorm = x_norm;
                analysis(ncase).energy = energy;


                disp(' ');

                ncase = ncase + 1;
            end
        end
    end

save('results.mat','analysis');

load handel;
player = audioplayer(y, Fs);
play(player);
```

2 Write input file

```
%% Write input data file
function [] = write_input_file(params)

fp = fopen('input_data.py','w+');

fprintf(fp,'## GEOMETRY \n');
fprintf(fp,'L1 = %6.4e \n', params(1));
fprintf(fp,'L2 = %6.4e \n', params(2));
fprintf(fp,'A = %6.4e \n', params(3));
fprintf(fp,'B = %6.4e \n', params(4));
fprintf(fp,'C = %6.4e \n', params(5));
fprintf(fp,'D = %6.4e \n', params(6));
fprintf(fp,'F = %6.4e \n', params(7));
fprintf(fp,'G = %6.4e \n', params(8));

fprintf(fp,'## MATERIAL \n');
fprintf(fp,'E1 = %6.4e \n', params(9));
fprintf(fp,'N1 = %6.4e \n', params(10));
fprintf(fp,'AF1 = %6.4e \n', params(11));
fprintf(fp,'E2 = %6.4e \n', params(12));
fprintf(fp,'N2 = %6.4e \n', params(13));
fprintf(fp,'AF2 = %6.4e \n', params(14));
```

```
23
24 fprintf(fp,'## CONDICOES \n');
25 fprintf(fp,'TEMP = %6.4e \n', params(15));
26
27 fclose(fp);
28
29 end
```

3 Input data

```
1  ## GEOMETRY
2  L1 = 7.0000e-01
3  L2 = 9.7500e-01
4  A = 2.5000e-01
5  B = 2.5481e-01
6  C = 5.0481e-01
7  D = 5.0000e-03
8  F = 4.8136e-03
9  G = 5.0963e-01
10 ## MATERIAL
11 E1 = 1.0000e+05
12 N1 = 2.4000e-01
13 AF1 = 1.0000e-05
14 E2 = 1.0000e+05
15 N2 = 2.0000e-01
16 AF2 = 5.0000e-07
17 ## CONDICOES
18 TEMP = 3.0000e+02
```

4 M2 automatic

```
1  # -*- coding: mbcs -*-
2  from part import *
3  from material import *
4  from section import *
5  from assembly import *
6  from step import *
7  from interaction import *
8  from load import *
9  from mesh import *
10 from optimization import *
11 from job import *
12 from sketch import *
13 from visualization import *
14 from connectorBehavior import *
```

```
15 #
   ------------------------------------------------------------------------

16
17 ## CALL VARIABLES
18
19 execfile('input_data.py')
20
21 #
   ------------------------------------------------------------------------

22
23
24 ## GEOMETRY
25
26 mdb.models['Model-1'].ConstrainedSketch(name='__profile__
       ', sheetSize=2.0)
27 mdb.models['Model-1'].sketches['__profile__'].
       sketchOptions.setValues(
28       viewStyle=AXISYM)
29 mdb.models['Model-1'].sketches['__profile__'].
       ConstructionLine(point1=(0.0,
30       -1.0), point2=(0.0, 1.0))
31 mdb.models['Model-1'].sketches['__profile__'].
       FixedConstraint(entity=
32       mdb.models['Model-1'].sketches['__profile__'].
             geometry[2])
33 mdb.models['Model-1'].sketches['__profile__'].rectangle(
       point1=(0.0, 0.0),
34       point2=(L1, L2))
35 mdb.models['Model-1'].Part(dimensionality=AXISYMMETRIC,
       name='Part-1', type=
36       DEFORMABLE_BODY)
37 mdb.models['Model-1'].parts['Part-1'].BaseShell(sketch=
38       mdb.models['Model-1'].sketches['__profile__'])
39 del mdb.models['Model-1'].sketches['__profile__']
40 mdb.models['Model-1'].ConstrainedSketch(gridSpacing=0.06,
       name='__profile__',
41       sheetSize=2.4, transform=
42       mdb.models['Model-1'].parts['Part-1'].
             MakeSketchTransform(
43       sketchPlane=mdb.models['Model-1'].parts['Part-1'].
             faces[0],
44       sketchPlaneSide=SIDE1, sketchOrientation=RIGHT,
             origin=(0.0, 0.0,
45       0.0)))
```

```
46 mdb.models['Model-1'].parts['Part-1'].
       projectReferencesOntoSketch(filter=
47      COPLANAR_EDGES, sketch=mdb.models['Model-1'].sketches
           ['__profile__'])
48 mdb.models['Model-1'].sketches['__profile__'].
       ConstructionLine(angle=0.0,
49      point1=(0, B))
50 mdb.models['Model-1'].sketches['__profile__'].
       HorizontalConstraint(
51      addUndoState=False, entity=
52      mdb.models['Model-1'].sketches['__profile__'].
           geometry[7])
53 mdb.models['Model-1'].sketches['__profile__'].
       CircleByCenterPerimeter(center=(
54      0, B), point1=(0, C))
55 mdb.models['Model-1'].sketches['__profile__'].Line(point1
       =(0,
56      B), point2=(L1, B))
57 mdb.models['Model-1'].sketches['__profile__'].Line(point1
       =(D, 0),
58      point2=(D, L2))
59 mdb.models['Model-1'].sketches['__profile__'].Line(point1
       =(L1/2.0, 0.0), point2=(
60      L1/2.0, L2))
61 mdb.models['Model-1'].sketches['__profile__'].Line(point1
       =(0.0, G), point2=(
62      L1, G))
63 mdb.models['Model-1'].parts['Part-1'].
       PartitionFaceBySketch(faces=
64      mdb.models['Model-1'].parts['Part-1'].faces.
           getSequenceFromMask(('[#1 ]',
65      ), ), sketch=mdb.models['Model-1'].sketches['
           __profile__'])
66 del mdb.models['Model-1'].sketches['__profile__']
67
68 #
   ____________________________________________________________________________


69
70 ## MATERIALS
71
72
73 mdb.models['Model-1'].Material(name='Alumina')
74 mdb.models['Model-1'].materials['Alumina'].Elastic(table
       =((E1, N1), ))
75 mdb.models['Model-1'].materials['Alumina'].Expansion(
```

```
    table=((AF1, ), ))
mdb.models['Model-1'].Material(name='glass')
mdb.models['Model-1'].materials['glass'].Elastic(table=((
    E2, N2), ))
mdb.models['Model-1'].materials['glass'].Expansion(table
    =((AF2, ), ))
mdb.models['Model-1'].HomogeneousSolidSection(material='
    Alumina', name=
    'inclusion', thickness=None)

#
    ------------------------------------------------------------------------


## SECTION


mdb.models['Model-1'].HomogeneousSolidSection(material='
    glass', name='matrix',
    thickness=None)
mdb.models['Model-1'].parts['Part-1'].Set(faces=
    mdb.models['Model-1'].parts['Part-1'].faces.
        getSequenceFromMask((
    '[#1620 ]', ), ), name='Set-1')
mdb.models['Model-1'].parts['Part-1'].SectionAssignment(
    offset=0.0,
    offsetField='', offsetType=MIDDLE_SURFACE, region=
    mdb.models['Model-1'].parts['Part-1'].sets['Set-1'],
        sectionName=
    'inclusion', thicknessAssignment=FROM_SECTION)
mdb.models['Model-1'].parts['Part-1'].Set(faces=
    mdb.models['Model-1'].parts['Part-1'].faces.
        getSequenceFromMask(('[#9df ]',
    ), ), name='Set-2')
mdb.models['Model-1'].parts['Part-1'].SectionAssignment(
    offset=0.0,
    offsetField='', offsetType=MIDDLE_SURFACE, region=
    mdb.models['Model-1'].parts['Part-1'].sets['Set-2'],
        sectionName='matrix',
    thicknessAssignment=FROM_SECTION)


#
    ------------------------------------------------------------------------


```

```
107 ## MESH GENERATION
108
109
110 mdb.models['Model-1'].parts['Part-1'].setMeshControls(
        elemShape=QUAD, regions=
111     mdb.models['Model-1'].parts['Part-1'].faces.
            getSequenceFromMask(('[#37f ]',
112     ), ), technique=STRUCTURED)
113 mdb.models['Model-1'].parts['Part-1'].setMeshControls(
        regions=
114     mdb.models['Model-1'].parts['Part-1'].faces.
            getSequenceFromMask(('[#880 ]',
115     ), ), technique=STRUCTURED)
116 mdb.models['Model-1'].parts['Part-1'].seedPart(
        deviationFactor=0.1,
117     minSizeFactor=0.1, size=0.0015)
118 mdb.models['Model-1'].parts['Part-1'].seedEdgeBySize(
        constraint=FINER,
119     deviationFactor=0.1, edges=
120     mdb.models['Model-1'].parts['Part-1'].edges.
            getSequenceFromMask((
121     '[#780000 ]', ), ), minSizeFactor=0.1, size=0.0003)
122 mdb.models['Model-1'].parts['Part-1'].generateMesh()
123
124 #
        ----------------------------------------------------------------------------


125
126 ## ASSEMBLY
127
128
129 mdb.models['Model-1'].rootAssembly.DatumCsysByThreePoints
        (coordSysType=
130     CYLINDRICAL, origin=(0.0, 0.0, 0.0), point1=(1.0,
            0.0, 0.0), point2=(0.0,
131     0.0, -1.0))
132 mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
        name='Part-1-1',
133     part=mdb.models['Model-1'].parts['Part-1'])
134
135 #
        ----------------------------------------------------------------------------


136
137 ## STEP
138
```

```
139 mdb.models['Model-1'].StaticStep(name='Step-1', nlgeom=ON
    , previous='Initial')
140 mdb.models['Model-1'].rootAssembly.Set(edges=
141     mdb.models['Model-1'].rootAssembly.instances['Part
        -1-1'].edges.getSequenceFromMask(
142     ('[#4200000 #1 ]', ), ), name='Set-1')
143
144 #
    ____________________________________________________________________________


145
146 ## BOUNDARY CONDITIONS
147
148 mdb.models['Model-1'].YsymmBC(createStepName='Initial',
    localCsys=None, name=
149     'BC-1', region=mdb.models['Model-1'].rootAssembly.
        sets['Set-1'])
150 mdb.models['Model-1'].rootAssembly.Set(name='Set-2',
    vertices=
151     mdb.models['Model-1'].rootAssembly.instances['Part
        -1-1'].vertices.getSequenceFromMask(
152     ('[#10000 ]', ), ))
153 mdb.models['Model-1'].DisplacementBC(amplitude=UNSET,
    createStepName='Initial',
154     distributionType=UNIFORM, fieldName='', localCsys=
        None, name='BC-2',
155     region=mdb.models['Model-1'].rootAssembly.sets['Set-2
        '], u1=SET, u2=UNSET,
156     ur3=UNSET)
157 mdb.models['Model-1'].rootAssembly.Set(edges=
158     mdb.models['Model-1'].rootAssembly.instances['Part
        -1-1'].edges.getSequenceFromMask(
159     ('[#ffffffff #1 ]', ), ), faces=
160     mdb.models['Model-1'].rootAssembly.instances['Part
        -1-1'].faces.getSequenceFromMask(
161     ('[#1fff ]', ), ), name='Set-3', vertices=
162     mdb.models['Model-1'].rootAssembly.instances['Part
        -1-1'].vertices.getSequenceFromMask(
163     ('[#1fffff ]', ), ))
164 mdb.models['Model-1'].Temperature(createStepName='Initial
    ',
165     crossSectionDistribution=CONSTANT_THROUGH_THICKNESS,
        distributionType=
166     UNIFORM, magnitudes=(TEMP, ), name='Predefined Field
        -1', region=
167     mdb.models['Model-1'].rootAssembly.sets['Set-3'])
```

```
168 mdb.models['Model-1'].TabularAmplitude(data=((0.0, 1.0),
        (1.0, 0.0)), name=
169     'Amp-1', smooth=SOLVER_DEFAULT, timeSpan=STEP)
170 mdb.models['Model-1'].predefinedFields['Predefined Field
        -1'].setValuesInStep(
171     amplitude='Amp-1', stepName='Step-1')
172
173
174 #
        ----------------------------------------------------------------------------
175
176 ## JOB
177
178 mdb.models['Model-1'].fieldOutputRequests['F-Output-1'].
        setValues(variables=(
179     'S', 'EE', 'U'))
180 mdb.Job(atTime=None, contactPrint=OFF, description='',
        echoPrint=OFF,
181     explicitPrecision=SINGLE, getMemoryFromAnalysis=True,
            historyPrint=OFF,
182     memory=90, memoryUnits=PERCENTAGE, model='Model-1',
            modelPrint=OFF,
183     multiprocessingMode=DEFAULT, name='Job-1',
            nodalOutputPrecision=SINGLE,
184     numCpus=4, numDomains=4, numGPUs=0, queue=None,
            resultsFormat=ODB, scratch=
185     '', type=ANALYSIS, userSubroutine='', waitHours=0,
            waitMinutes=0)
186 mdb.jobs['Job-1'].submit(consistencyChecking=OFF)
187
188
189 mdb.jobs['Job-1'].waitForCompletion
190
191 #
        ----------------------------------------------------------------------------
192
193 ## POS-PROCESSING
194
195 ## PATH
196 session.Path(name='Path-1', type=RADIAL, expression=((0,
        0, 0), (F, 0, 0),
197     (0, F, 0)), circleDefinition=ORIGIN_AXIS, numSegments
            =30,
198     radialAngle=0, startRadius=0, endRadius=CIRCLE_RADIUS
```

```
        )
199
200
201 #: session.paths['Path-1']
202 #: Job Job-1: Analysis Input File Processor completed
    successfully.
203 #: Job Job-1: Abaqus/Standard completed successfully.
204 #: Job Job-1 completed successfully.
205 a = mdb.models['Model-1'].rootAssembly
206 o3 = session.openOdb(name='./Job-1.odb')
207
208 o1 = session.openOdb(
209     name='./Job-1.odb')
210 session.viewports['Viewport: 1'].setValues(
    displayedObject=o1)
211 s1f1_S = session.odbs['./Job-1.odb'].steps['Step-1'].
    frames[1].fieldOutputs['S']
212 s1f1_EE = session.odbs['./Job-1.odb'].steps['Step-1'].
    frames[1].fieldOutputs['EE']
213 s1f1_S = session.odbs['./Job-1.odb'].steps['Step-1'].
    frames[1].fieldOutputs['S']
214 s1f1_EE = session.odbs['./Job-1.odb'].steps['Step-1'].
    frames[1].fieldOutputs['EE']
215 s1f1_S = session.odbs['./Job-1.odb'].steps['Step-1'].
    frames[1].fieldOutputs['S']
216 s1f1_EE = session.odbs['./Job-1.odb'].steps['Step-1'].
    frames[1].fieldOutputs['EE']
217 tmpField = 0.5*(s1f1_S.getScalarField(
218     invariant=MAX_PRINCIPAL)*s1f1_EE.getScalarField(
          invariant=MAX_PRINCIPAL)+\
219     s1f1_S.getScalarField(invariant=MID_PRINCIPAL)*
          s1f1_EE.getScalarField(
220     invariant=MID_PRINCIPAL)+s1f1_S.getScalarField(
221     invariant=MIN_PRINCIPAL)*s1f1_EE.getScalarField(
          invariant=MIN_PRINCIPAL))
222 currentOdb = session.odbs['./Job-1.odb']
223 scratchOdb = session.ScratchOdb(odb=currentOdb)
224 sessionStep = scratchOdb.Step(name='Session Step',
225     description='Step for Viewer non-persistent fields',
          domain=TIME,
226     timePeriod=1.0)
227 sessionFrame = sessionStep.Frame(frameId=0, frameValue
    =0.0,
228     description='Session Frame')
229 sessionField = sessionFrame.FieldOutput(name='Energy',
230     description='0.5*(s1f1_S.getScalarField(invariant=
```

```
            MAX_PRINCIPAL)*s1f1_EE.getScalarField(invariant=
            MAX_PRINCIPAL)+s1f1_S.getScalarField(invariant=
            MID_PRINCIPAL)*s1f1_EE.getScalarField(invariant=
            MID_PRINCIPAL)+s1f1_S.getScalarField(invariant=
            MIN_PRINCIPAL)*s1f1_EE.getScalarField(invariant=
            MIN_PRINCIPAL))',
231         field=tmpField)
232 odbname = session.viewports['Viewport: 1'].odbDisplay.
        name
233 frame1 = session.scratchOdbs[odbname].steps['Session Step
        '].frames[0]
234 session.viewports['Viewport: 1'].odbDisplay.setFrame(
        frame=frame1)
235 session.viewports['Viewport: 1'].odbDisplay.display.
        setValues(plotState=(
236     CONTOURS_ON_UNDEF, ))
237 pth = session.paths['Path-1']
238 session.XYDataFromPath(name='energy', path=pth,
        includeIntersections=False,
239     projectOntoMesh=False, pathStyle=UNIFORM_SPACING,
            numIntervals=30,
240     projectionTolerance=0, shape=UNDEFORMED, labelType=
            NORM_DISTANCE)
241 x0 = session.xyDataObjects['energy']
242 session.writeXYReport(fileName='energy.txt', xyData=(x0,
        ))
```

APÊNDICE C

1 Results - Part 1

```
clear all
clc
close all


load ('/media/DADOS/luchini/automatic3/Automatic/
    analysis19.mat')
analysis1=analysis;

for i=1:length(analysis1)

[n, bin] = histc(analysis1(i).xnorm, unique(analysis1(i).
    xnorm));
multiple = find(n > 1);
index    = find(ismember(bin, multiple));
b=length(index)/2;
c=0;
if length(index)>1
for j=1:b
     analysis1(i).xnorm(index(end-c))=[];
     analysis1(i).energy(index(end-c))=[];
     if index>2
     c=c+2;
     end
end
end
i
end
```

2 Results - Part 2

```
clear all
clc
close all

load results2.mat

results=[analysis1 ];

% A=hot(31);
%A=colorbrewer(31,'BuPu');
A=colorbrewer(64,'PRGn');
% A=gray(31);
```

```
13 A=A(16:47,:);p00 =       0.3863; % (0.3745, 0.3981)
14 x = []; y = []; z = [];
15 for i=1:length(results)
16     [a b]=max(results(i).energy);
17     max_position(i)=b;
18     cor(i,:)=A(b,:);
19     delta_E(i)=results(i).E2/results(i).E1;
20     delta_A(i)=1-(results(i).AF2/results(i).AF1);
21     PHI(i)=results(i).phi;
22     if (b>25 && b<31)
23 if (delta_A(i) ~= 0 && delta_E(i) ~= 1)
24 %       plot3(delta_E(i),delta_A(i),results(i).phi,'o','
       MarkerFaceColor',cor(i,:),'MarkerEdgeColor',cor(i,:))
25     plot3(delta_E(i),delta_A(i),results(i).phi,'o','
           MarkerFaceColor','k','MarkerEdgeColor','k')
26 %     if (delta_A(i) ~= 0)
27     x(end+1) = delta_E(i);
28     y(end+1) = delta_A(i);
29     z(end+1) = results(i).phi;
30     end
31      hold on
32 %      set(gca,'Color',[0.0 1 0.0]);
33 end
34 end
```

APÊNDICE D

1 Honeycomb Code

Este programa foi desenvolvido juntamente com o Eng. M.e Caiuã Caldeira de Melo.

2 Input data

```
H = Honeycomb()
H.setEdgeSize(1)
H.setGapSize(0.05)
H.setNumberColumns(8)
H.setNumberRows(6)
H.generateHexs()
H.generateCohesives()
H.generateAssembly()
```

3 Code

```
"""
Funcao: Automatiza a criacao de um honeycomb com gaps
    preenchidos por elementos coesivos
 Autores: Caiua Caldeira de Melo e Bruno Luchini
 Data de criacao: 2/12/2016

 Para executar no abaqus?
 Carrega a classe
  H = Honeycomb()
 Seta o tamanho da aresta do hexagono
  H.setEdgeSize(1)
 Tamanho do gap
  H.setGapSize(0.05)
 Numero de colunas
  H.setNumberColumns(7)
 Numero de linhas
  H.setNumberRows(6)
  H.generateHexs()
  H.generateCohesives()
  H.generateAssembly()
"""
# -*- coding: mbcs -*-
from material import *
from section import *
from assembly import *
from step import *
from interaction import *
from load import *
from mesh import *
from optimization import *
```

```
30 from job import *
31 from sketch import *
32 from visualization import *
33 from connectorBehavior import *
34 import numpy
35
36 class Honeycomb:
37     # Atributos
38     edge = 4 # Lado do hexagono
39     nr = 5 # Numero na linha de hexagonos na aresta
40     nc = 5 # Numero na linha de hexagonos na coluna
41
42     gap = 0.05 # Tamanho do gap para os elementos coesivos
43
44     # Metodos
45     def setEdgeSize(self,edge):
46         self.edge = edge
47     def setGapSize(self,gap):
48         self.gap = gap
49     def setNumberRows(self,nr):
50         self.nr = nr
51     def setNumberColumns(self,nc):
52         self.nc = nc
53
54     def generateHexs(self):
55         # Hexagonos: gera as partes hexagonais do honeycomb
56         for j in range(2,4*self.nr,4):
57             k=0
58             for i in range(2,2*self.nc,2):
59                 k=k+1
60                 mdb.models['Model-1'].ConstrainedSketch(name='
                        __profile__', sheetSize=200.0)
61                 mdb.models['Model-1'].sketches['__profile__'].
                        Line(point1=(0.0, 0.0),
62                         point2=(1.0*self.edge, 0.0))
63                 mdb.models['Model-1'].sketches['__profile__'].
                        Line(point1=(1.0*self.edge, 0.0),
64                         point2=(1.5*self.edge, (sqrt(3)/2)*self.
                            edge))
65                 mdb.models['Model-1'].sketches['__profile__'].
                        Line(point1=(1.5*self.edge, (sqrt(3)/2)*
                        self.edge),
66                         point2=(1.0*self.edge, sqrt(3)*self.edge))
67                 mdb.models['Model-1'].sketches['__profile__'].
                        Line(point1=(1.0*self.edge, sqrt(3)*self.
                        edge),
68                         point2=(0.0, sqrt(3)*self.edge))
69                 mdb.models['Model-1'].sketches['__profile__'].
                        Line(point1=(0.0, sqrt(3)*self.edge),
70                         point2=(-0.5*self.edge, (sqrt(3)/2)*self.
```

```
                edge))
71              mdb.models['Model-1'].sketches['__profile__'].
                    Line(point1=(-0.5*self.edge, (sqrt(3)/2)*
                    self.edge),
72                    point2=(0.0, 0.0))
73              mdb.models['Model-1'].Part(dimensionality=
                    TWO_D_PLANAR, name='Hex-'+str(j+((k-1)%2)
                    *2)+'-'+str(i), type=
74                    DEFORMABLE_BODY)
75              mdb.models['Model-1'].parts['Hex-'+str(j+((k
                    -1)%2)*2)+'-'+str(i)].BaseShell(sketch=
76                    mdb.models['Model-1'].sketches['
                        __profile__'])
77              del mdb.models['Model-1'].sketches['
                    __profile__']
78              # Cria as particoes para melhorar a malha
79              mdb.models['Model-1'].ConstrainedSketch(
                    gridSpacing=0.13, name='__profile__',
80                    sheetSize=5.29, transform=
81                    mdb.models['Model-1'].parts['Hex-'+str(j
                        +((k-1)%2)*2)+'-'+str(i)].
                        MakeSketchTransform(
82                    sketchPlane=mdb.models['Model-1'].parts['
                        Hex-'+str(j+((k-1)%2)*2)+'-'+str(i)].
                        faces[0],
83                    sketchPlaneSide=SIDE1, sketchOrientation=
                        RIGHT, origin=(0.5*self.edge, 0.5*sqrt
                        (3)*self.edge,
84                    0.0)))
85              mdb.models['Model-1'].parts['Hex-'+str(j+((k
                    -1)%2)*2)+'-'+str(i)].
                    projectReferencesOntoSketch(filter=
86                    COPLANAR_EDGES, sketch=mdb.models['Model-1
                        '].sketches['__profile__'])
87              mdb.models['Model-1'].sketches['__profile__'].
                    Line(point1=(-1.0*self.edge,
88                    0), point2=(1.0*self.edge, 0))
89              mdb.models['Model-1'].sketches['__profile__'].
                    HorizontalConstraint(
90                    addUndoState=False, entity=
91                    mdb.models['Model-1'].sketches['
                        __profile__'].geometry[8])
92              mdb.models['Model-1'].sketches['__profile__'].
                    Line(point1=(0.5*self.edge, -0.5*sqrt(3)*
                    self.edge),
93                    point2=(-0.5*self.edge, 0.5*sqrt(3)*self.
                        edge))
94              mdb.models['Model-1'].sketches['__profile__'].
                    Line(point1=(-0.5*self.edge, -0.5*sqrt(3)*
                    self.edge),
```

```
95                      point2=(0.5*self.edge, 0.5*sqrt(3)*self.
                            edge))
96                  mdb.models['Model-1'].parts['Hex-'+str(j+((k
                        -1)%2)*2)+'-'+str(i)].
                        PartitionFaceBySketch(faces=
97                        mdb.models['Model-1'].parts['Hex-'+str(j
                            +((k-1)%2)*2)+'-'+str(i)].faces[0:1],
98                        sketch=mdb.models['Model-1'].sketches['
                            __profile__'])
99                  del mdb.models['Model-1'].sketches['
                        __profile__']
100             # Gera as partes hexagonais do honeycomb da aresta
                    inferior
101             for i in range(4,2*self.nc,4):
102                 mdb.models['Model-1'].ConstrainedSketch(name='
                        __profile__', sheetSize=200.0)
103                 mdb.models['Model-1'].sketches['__profile__'].Line
                        (point1=(-0.5*self.edge, (sqrt(3)/2)*self.edge
                        ),
104                      point2=(1.5*self.edge, (sqrt(3)/2)*self.edge))
105                 mdb.models['Model-1'].sketches['__profile__'].Line
                        (point1=(1.5*self.edge, (sqrt(3)/2)*self.edge)
                        ,
106                      point2=(1.0*self.edge, sqrt(3)*self.edge))
107                 mdb.models['Model-1'].sketches['__profile__'].Line
                        (point1=(1.0*self.edge, sqrt(3)*self.edge),
108                      point2=(0.0, sqrt(3)*self.edge))
109                 mdb.models['Model-1'].sketches['__profile__'].Line
                        (point1=(0.0, sqrt(3)*self.edge),
110                      point2=(-0.5*self.edge, (sqrt(3)/2)*self.edge)
                            )
111                 mdb.models['Model-1'].Part(dimensionality=
                        TWO_D_PLANAR, name='Hex-'+str(0)+'-'+str(i),
                        type=
112                      DEFORMABLE_BODY)
113                 mdb.models['Model-1'].parts['Hex-'+str(0)+'-'+str(
                        i)].BaseShell(sketch=
114                      mdb.models['Model-1'].sketches['__profile__'])
115                 del mdb.models['Model-1'].sketches['__profile__']
116                 # Particoes
117                 mdb.models['Model-1'].ConstrainedSketch(
                        gridSpacing=0.13, name='__profile__',
118                      sheetSize=5.29, transform=
119                 mdb.models['Model-1'].parts['Hex-'+str(0)+'-'+str(
                        i)].MakeSketchTransform(
120                      sketchPlane=mdb.models['Model-1'].parts['Hex-'
                            +str(0)+'-'+str(i)].faces[0],
121                      sketchPlaneSide=SIDE1, sketchOrientation=RIGHT
                            , origin=(-0.5*self.edge, 0.5*sqrt(3)*self
                            .edge, 0.0)))
```

```
122             mdb.models['Model-1'].parts['Hex-'+str(0)+'-'+str(
                    i)].projectReferencesOntoSketch(filter=
123                 COPLANAR_EDGES, sketch=mdb.models['Model-1'].
                        sketches['__profile__'])
124             mdb.models['Model-1'].sketches['__profile__'].Line
                    (point1=(self.edge, 0), point2=(0.5*self.edge,
                     0.5*sqrt(3)*self.edge))
125             mdb.models['Model-1'].sketches['__profile__'].Line
                    (point1=(self.edge, 0), point2=(1.5*self.edge,
                     0.5*sqrt(3)*self.edge))
126             mdb.models['Model-1'].parts['Hex-'+str(0)+'-'+str(
                    i)].PartitionFaceBySketch(faces=
127                 mdb.models['Model-1'].parts['Hex-'+str(0)+'-'+
                        str(i)].faces[0:1],
128                 sketch=mdb.models['Model-1'].sketches['
                        __profile__'])
129             del mdb.models['Model-1'].sketches['__profile__']
130         # Gera as partes hexagonais do honeycomb da aresta
                superior
131         for i in range(2,2*self.nc,4):
132             mdb.models['Model-1'].ConstrainedSketch(name='
                    __profile__', sheetSize=200.0)
133             mdb.models['Model-1'].sketches['__profile__'].Line
                    (point1=(0.0, 0.0),
134                 point2=(1.0*self.edge, 0.0))
135             mdb.models['Model-1'].sketches['__profile__'].Line
                    (point1=(1.0*self.edge, 0.0),
136                 point2=(1.5*self.edge, (sqrt(3)/2)*self.edge))
137             mdb.models['Model-1'].sketches['__profile__'].Line
                    (point1=(1.5*self.edge, (sqrt(3)/2)*self.edge)
                    ,
138                 point2=(-0.5*self.edge, (sqrt(3)/2)*self.edge)
                        )
139             mdb.models['Model-1'].sketches['__profile__'].Line
                    (point1=(-0.5*self.edge, (sqrt(3)/2)*self.edge
                    ),
140                   point2=(0.0, 0.0))
141             mdb.models['Model-1'].Part(dimensionality=
                    TWO_D_PLANAR, name='Hex-'+str(4*self.nr+2)+'-'
                    +str(i), type=
142                 DEFORMABLE_BODY)
143             mdb.models['Model-1'].parts['Hex-'+str(4*self.nr
                    +2)+'-'+str(i)].BaseShell(sketch=
144                 mdb.models['Model-1'].sketches['__profile__'])
145             del mdb.models['Model-1'].sketches['__profile__']
146             # Particoes
147             mdb.models['Model-1'].ConstrainedSketch(
                    gridSpacing=0.13, name='__profile__',
148                 sheetSize=5.29, transform=
149             mdb.models['Model-1'].parts['Hex-'+str(4*self.nr
```

```
                +2)+'-'+str(i)].MakeSketchTransform(
150             sketchPlane=mdb.models['Model-1'].parts['Hex-'
                    +str(4*self.nr+2)+'-'+str(i)].faces[0],
151             sketchPlaneSide=SIDE1, sketchOrientation=RIGHT
                    , origin=(0.0, 0.0, 0.0)))
152         mdb.models['Model-1'].parts['Hex-'+str(4*self.nr
                +2)+'-'+str(i)].projectReferencesOntoSketch(
                filter=
153             COPLANAR_EDGES, sketch=mdb.models['Model-1'].
                    sketches['__profile__'])
154         mdb.models['Model-1'].sketches['__profile__'].Line
                (point1=(0, 0), point2=(0.5*self.edge, 0.5*
                sqrt(3)*self.edge))
155         mdb.models['Model-1'].sketches['__profile__'].Line
                (point1=(self.edge, 0), point2=(0.5*self.edge,
                 0.5*sqrt(3)*self.edge))
156         mdb.models['Model-1'].parts['Hex-'+str(4*self.nr
                +2)+'-'+str(i)].PartitionFaceBySketch(faces=
157             mdb.models['Model-1'].parts['Hex-'+str(4*self.
                    nr+2)+'-'+str(i)].faces[0:1],
158             sketch=mdb.models['Model-1'].sketches['
                    __profile__'])
159         del mdb.models['Model-1'].sketches['__profile__']
160     # Gera as partes hexagonais do honeycomb do lado
        esquerdo
161     for j in range(4,4*(self.nr+1),4):
162         mdb.models['Model-1'].ConstrainedSketch(name='
                __profile__', sheetSize=200.0)
163         mdb.models['Model-1'].sketches['__profile__'].Line
                (point1=(0.5*self.edge, 0.0),
164             point2=(1.0*self.edge, 0.0))
165         mdb.models['Model-1'].sketches['__profile__'].Line
                (point1=(1.0*self.edge, 0.0),
166             point2=(1.5*self.edge, (sqrt(3)/2)*self.edge))
167         mdb.models['Model-1'].sketches['__profile__'].Line
                (point1=(1.5*self.edge, (sqrt(3)/2)*self.edge)
                ,
168             point2=(1.0*self.edge, sqrt(3)*self.edge))
169         mdb.models['Model-1'].sketches['__profile__'].Line
                (point1=(1.0*self.edge, sqrt(3)*self.edge),
170             point2=(0.5*self.edge, sqrt(3)*self.edge))
171         mdb.models['Model-1'].sketches['__profile__'].Line
                (point1=(0.5*self.edge, sqrt(3)*self.edge),
172             point2=(0.5*self.edge, 0.0))
173         mdb.models['Model-1'].Part(dimensionality=
                TWO_D_PLANAR, name='Hex-'+str(j)+'-'+str(0),
                type=
174             DEFORMABLE_BODY)
175         mdb.models['Model-1'].parts['Hex-'+str(j)+'-'+str
                (0)].BaseShell(sketch=
```

```
176                 mdb.models['Model-1'].sketches['__profile__'])
177             del mdb.models['Model-1'].sketches['__profile__']
178             # Particoes
179             mdb.models['Model-1'].ConstrainedSketch(
                    gridSpacing=0.13, name='__profile__',
180                     sheetSize=5.29, transform=
181             mdb.models['Model-1'].parts['Hex-'+str(j)+'-'+str
                    (0)].MakeSketchTransform(
182                     sketchPlane=mdb.models['Model-1'].parts['Hex-'
                        +str(j)+'-'+str(0)].faces[0],
183                     sketchPlaneSide=SIDE1, sketchOrientation=RIGHT
                        , origin=(0.5*self.edge, 0.0, 0.0)))
184             mdb.models['Model-1'].parts['Hex-'+str(j)+'-'+str
                    (0)].projectReferencesOntoSketch(filter=
185                     COPLANAR_EDGES, sketch=mdb.models['Model-1'].
                        sketches['__profile__'])
186             mdb.models['Model-1'].sketches['__profile__'].Line
                    (point1=(0, 0.5*sqrt(3)*self.edge), point2
                    =(0.5*self.edge, sqrt(3)*self.edge))
187             mdb.models['Model-1'].sketches['__profile__'].Line
                    (point1=(0, 0.5*sqrt(3)*self.edge), point2=(
                    self.edge, 0.5*sqrt(3)*self.edge))
188             mdb.models['Model-1'].sketches['__profile__'].Line
                    (point1=(0, 0.5*sqrt(3)*self.edge), point2
                    =(0.5*self.edge, 0.0))
189             mdb.models['Model-1'].parts['Hex-'+str(j)+'-'+str
                    (0)].PartitionFaceBySketch(faces=
190                     mdb.models['Model-1'].parts['Hex-'+str(j)+'-'+
                        str(0)].faces[0:1],
191                     sketch=mdb.models['Model-1'].sketches['
                        __profile__'])
192             del mdb.models['Model-1'].sketches['__profile__']
193         # Gera as partes hexagonais do honeycomb do lado
                direito
194         for j in range(2,4*(self.nr),4):
195             mdb.models['Model-1'].ConstrainedSketch(name='
                    __profile__', sheetSize=200.0)
196             mdb.models['Model-1'].sketches['__profile__'].Line
                    (point1=(0.0, 0.0),
197                     point2=(0.5*self.edge, 0.0))
198             mdb.models['Model-1'].sketches['__profile__'].Line
                    (point1=(0.5*self.edge, 0.0),
199                     point2=(0.5*self.edge, sqrt(3)*self.edge))
200             mdb.models['Model-1'].sketches['__profile__'].Line
                    (point1=(0.5*self.edge, sqrt(3)*self.edge),
201                     point2=(0.0, sqrt(3)*self.edge))
202             mdb.models['Model-1'].sketches['__profile__'].Line
                    (point1=(0.0, sqrt(3)*self.edge),
203                     point2=(-0.5*self.edge, 0.5*sqrt(3)*self.edge)
                        )
```

```
204             mdb.models['Model-1'].sketches['__profile__'].Line
                    (point1=(-0.5*self.edge, 0.5*sqrt(3)*self.edge
                    ),
205                  point2=(0.0, 0.0))
206             mdb.models['Model-1'].Part(dimensionality=
                    TWO_D_PLANAR, name='Hex-'+str(j+((self.nc-1)%
                    2)*2)+'-'+str(2*self.nc), type=
207                  DEFORMABLE_BODY)
208             mdb.models['Model-1'].parts['Hex-'+str(j+((self.nc
                    -1)%2)*2)+'-'+str(2*self.nc)].BaseShell(sketch
                    =
209                  mdb.models['Model-1'].sketches['__profile__'])
210             del mdb.models['Model-1'].sketches['__profile__']
211             # Particoes
212             mdb.models['Model-1'].ConstrainedSketch(
                    gridSpacing=0.13, name='__profile__',
213                  sheetSize=5.29, transform=
214             mdb.models['Model-1'].parts['Hex-'+str(j+((self.nc
                    -1)%2)*2)+'-'+str(2*self.nc)].
                    MakeSketchTransform(
215                  sketchPlane=mdb.models['Model-1'].parts['Hex-'
                        +str(j+((self.nc-1)%2)*2)+'-'+str(2*self.
                        nc)].faces[0],
216                  sketchPlaneSide=SIDE1, sketchOrientation=RIGHT
                        , origin=(0.0, 0.0, 0.0)))
217             mdb.models['Model-1'].parts['Hex-'+str(j+((self.nc
                    -1)%2)*2)+'-'+str(2*self.nc)].
                    projectReferencesOntoSketch(filter=
218                  COPLANAR_EDGES, sketch=mdb.models['Model-1'].
                        sketches['__profile__'])
219             mdb.models['Model-1'].sketches['__profile__'].Line
                    (point1=(0.5*self.edge, 0.5*sqrt(3)*self.edge)
                    , point2=(0.0, 0.0))
220             mdb.models['Model-1'].sketches['__profile__'].Line
                    (point1=(0.5*self.edge, 0.5*sqrt(3)*self.edge)
                    , point2=(-0.5*self.edge, 0.5*sqrt(3)*self.
                    edge))
221             mdb.models['Model-1'].sketches['__profile__'].Line
                    (point1=(0.5*self.edge, 0.5*sqrt(3)*self.edge)
                    , point2=(0.0, sqrt(3)*self.edge))
222             mdb.models['Model-1'].parts['Hex-'+str(j+((self.nc
                    -1)%2)*2)+'-'+str(2*self.nc)].
                    PartitionFaceBySketch(faces=
223                  mdb.models['Model-1'].parts['Hex-'+str(j+((
                        self.nc-1)%2)*2)+'-'+str(2*self.nc)].faces
                        [0:1],
224                  sketch=mdb.models['Model-1'].sketches['
                        __profile__'])
225             del mdb.models['Model-1'].sketches['__profile__']
226         # Gera as partes hexagonais do honeycomb dos vertices
```

```
227             mdb.models['Model-1'].ConstrainedSketch(name='
                    __profile__', sheetSize=200.0)
228             mdb.models['Model-1'].sketches['__profile__'].Line(
                    point1=(0.0, 0.0),
229                     point2=(self.edge, 0.0))
230             mdb.models['Model-1'].sketches['__profile__'].Line(
                    point1=(self.edge, 0.0),
231                     point2=(0.5*self.edge, 0.5*sqrt(3)*self.edge))
232             mdb.models['Model-1'].sketches['__profile__'].Line(
                    point1=(0.5*self.edge, 0.5*sqrt(3)*self.edge),
233                     point2=(0, 0.5*sqrt(3)*self.edge))
234             mdb.models['Model-1'].sketches['__profile__'].Line(
                    point1=(0, 0.5*sqrt(3)*self.edge),
235                     point2=(0.0, 0.0))
236             mdb.models['Model-1'].Part(dimensionality=TWO_D_PLANAR
                    , name='Hex-'+str(0)+'-'+str(0), type=
237                     DEFORMABLE_BODY)
238             mdb.models['Model-1'].parts['Hex-'+str(0)+'-'+str(0)].
                    BaseShell(sketch=
239                     mdb.models['Model-1'].sketches['__profile__'])
240             del mdb.models['Model-1'].sketches['__profile__']
241             mdb.models['Model-1'].ConstrainedSketch(gridSpacing
                    =0.13, name='__profile__',
242                     sheetSize=5.29, transform=
243             mdb.models['Model-1'].parts['Hex-0-0'].
                    MakeSketchTransform(
244                     sketchPlane=mdb.models['Model-1'].parts['Hex-0-0'
                            ].faces[0],
245                     sketchPlaneSide=SIDE1, sketchOrientation=RIGHT,
                            origin=(0.0, 0.0, 0.0)))
246             mdb.models['Model-1'].parts['Hex-0-0'].
                    projectReferencesOntoSketch(filter=
247                     COPLANAR_EDGES, sketch=mdb.models['Model-1'].
                            sketches['__profile__'])
248             mdb.models['Model-1'].sketches['__profile__'].Line(
                    point1=(0.0, 0.0), point2=(0.5*self.edge, 0.5*sqrt
                    (3)*self.edge))
249             mdb.models['Model-1'].parts['Hex-0-0'].
                    PartitionFaceBySketch(faces=
250                     mdb.models['Model-1'].parts['Hex-0-0'].faces[0:1],
251                     sketch=mdb.models['Model-1'].sketches['__profile__
                            '])
252             del mdb.models['Model-1'].sketches['__profile__']
253             if (self.nc % 2) == 1:
254                 mdb.models['Model-1'].ConstrainedSketch(name='
                            __profile__', sheetSize=200.0)
255                 mdb.models['Model-1'].sketches['__profile__'].Line
                            (point1=(0.0, 0.0),
256                             point2=(0.5*self.edge, 0.0))
257                 mdb.models['Model-1'].sketches['__profile__'].Line
```

```
                (point1=(0.5*self.edge, 0.0),
258             point2=(0.5*self.edge, 0.5*sqrt(3)*self.edge))
259         mdb.models['Model-1'].sketches['__profile__'].Line
                (point1=(0.5*self.edge, 0.5*sqrt(3)*self.edge)
                ,
260             point2=(-0.5*self.edge, 0.5*sqrt(3)*self.edge)
                    )
261         mdb.models['Model-1'].sketches['__profile__'].Line
                (point1=(-0.5*self.edge, 0.5*sqrt(3)*self.edge
                ),
262             point2=(0.0, 0.0))
263         mdb.models['Model-1'].Part(dimensionality=
                TWO_D_PLANAR, name='Hex-'+str(4*self.nr+2)+'-'
                +str(2*self.nc), type=
264             DEFORMABLE_BODY)
265         mdb.models['Model-1'].parts['Hex-'+str(4*self.nr
                +2)+'-'+str(2*self.nc)].BaseShell(sketch=
266             mdb.models['Model-1'].sketches['__profile__'])
267         del mdb.models['Model-1'].sketches['__profile__']
268         mdb.models['Model-1'].ConstrainedSketch(
                gridSpacing=0.13, name='__profile__',
269             sheetSize=5.29, transform=
270         mdb.models['Model-1'].parts['Hex-'+str(4*self.nr
                +2)+'-'+str(2*self.nc)].MakeSketchTransform(
271             sketchPlane=mdb.models['Model-1'].parts['Hex-'
                    +str(4*self.nr+2)+'-'+str(2*self.nc)].
                    faces[0],
272             sketchPlaneSide=SIDE1, sketchOrientation=RIGHT
                    , origin=(0.0, 0.0, 0.0)))
273         mdb.models['Model-1'].parts['Hex-'+str(4*self.nr
                +2)+'-'+str(2*self.nc)].
                projectReferencesOntoSketch(filter=
274             COPLANAR_EDGES, sketch=mdb.models['Model-1'].
                    sketches['__profile__'])
275         mdb.models['Model-1'].sketches['__profile__'].Line
                (point1=(0.0, 0.0), point2=(0.5*self.edge,
                0.5*sqrt(3)*self.edge))
276         mdb.models['Model-1'].parts['Hex-'+str(4*self.nr
                +2)+'-'+str(2*self.nc)].PartitionFaceBySketch(
                faces=
277             mdb.models['Model-1'].parts['Hex-'+str(4*self.
                    nr+2)+'-'+ str(2*self.nc)].faces[0:1],
278             sketch=mdb.models['Model-1'].sketches['
                    __profile__'])
279         del mdb.models['Model-1'].sketches['__profile__']
280     else:
281         mdb.models['Model-1'].ConstrainedSketch(name='
                __profile__', sheetSize=200.0)
282         mdb.models['Model-1'].sketches['__profile__'].Line
                (point1=(0.0, 0.0),
```

```
283                 point2=(self.edge, 0.0))
284             mdb.models['Model-1'].sketches['__profile__'].Line
                    (point1=(self.edge, 0.0),
285                 point2=(self.edge, 0.5*sqrt(3)*self.edge))
286             mdb.models['Model-1'].sketches['__profile__'].Line
                    (point1=(self.edge, 0.5*sqrt(3)*self.edge),
287                 point2=(0.5*self.edge, 0.5*sqrt(3)*self.edge))
288             mdb.models['Model-1'].sketches['__profile__'].Line
                    (point1=(0.5*self.edge, 0.5*sqrt(3)*self.edge)
                    ,
289                 point2=(0.0, 0.0))
290             mdb.models['Model-1'].Part(dimensionality=
                    TWO_D_PLANAR, name='Hex-'+str(0)+'-'+str(2*
                    self.nc), type=
291                 DEFORMABLE_BODY)
292             mdb.models['Model-1'].parts['Hex-'+str(0)+'-'+str
                    (2*self.nc)].BaseShell(sketch=
293                 mdb.models['Model-1'].sketches['__profile__'])
294             mdb.models['Model-1'].ConstrainedSketch(
                    gridSpacing=0.13, name='__profile__',
295                 sheetSize=5.29, transform=
296             mdb.models['Model-1'].parts['Hex-0-'+str(2*self.nc
                    )].MakeSketchTransform(
297                 sketchPlane=mdb.models['Model-1'].parts['Hex
                        -0-'+str(2*self.nc)].faces[0],
298                 sketchPlaneSide=SIDE1, sketchOrientation=RIGHT
                        , origin=(0.0, 0.0, 0.0)))
299             mdb.models['Model-1'].parts['Hex-0-'+str(2*self.nc
                    )].projectReferencesOntoSketch(filter=
300                 COPLANAR_EDGES, sketch=mdb.models['Model-1'].
                        sketches['__profile__'])
301             mdb.models['Model-1'].sketches['__profile__'].Line
                    (point1=(self.edge, 0.0), point2=(0.5*self.
                    edge, 0.5*sqrt(3)*self.edge))
302             mdb.models['Model-1'].parts['Hex-0-'+str(2*self.nc
                    )].PartitionFaceBySketch(faces=
303                 mdb.models['Model-1'].parts['Hex-0-'+str(2*
                        self.nc)].faces[0:1],
304                 sketch=mdb.models['Model-1'].sketches['
                        __profile__'])
305             del mdb.models['Model-1'].sketches['__profile__']
306     def generateCohesives(self):
307         # Elementos coesivos: gera os elementos coesivos
308         # Coesive de angulo 0
309         mdb.models['Model-1'].ConstrainedSketch(name='
                __profile__', sheetSize=200.0)
310         mdb.models['Model-1'].sketches['__profile__'].
                rectangle(point1=(0.0, sqrt(3)*self.edge),
311             point2=(self.edge, self.gap + sqrt(3)*self.edge))
312         mdb.models['Model-1'].Part(dimensionality=TWO_D_PLANAR
```

```
        , name='Cohesive000', type=
313         DEFORMABLE_BODY)
314     mdb.models['Model-1'].parts['Cohesive000'].BaseShell(
            sketch=
315         mdb.models['Model-1'].sketches['__profile__'])
316     del mdb.models['Model-1'].sketches['__profile__']
317     # Coesive de angulo 0 metade (para preencher o lado
            direito e esquerdo)
318     mdb.models['Model-1'].ConstrainedSketch(name='
            __profile__', sheetSize=200.0)
319     mdb.models['Model-1'].sketches['__profile__'].
            rectangle(point1=(0.0, sqrt(3)*self.edge),
320         point2=(0.5*self.edge, self.gap + sqrt(3)*self.
                edge))
321     mdb.models['Model-1'].Part(dimensionality=TWO_D_PLANAR
            , name='Cohesive000h', type=
322         DEFORMABLE_BODY)
323     mdb.models['Model-1'].parts['Cohesive000h'].BaseShell(
            sketch=
324         mdb.models['Model-1'].sketches['__profile__'])
325     del mdb.models['Model-1'].sketches['__profile__']
326     # Coesive de angulo 60
327     mdb.models['Model-1'].ConstrainedSketch(name='
            __profile__', sheetSize=200.0)
328     mdb.models['Model-1'].sketches['__profile__'].
            rectangle(point1=(-0.5*self.edge,
329         0.5*sqrt(3)*self.edge), point2=(-0.5*self.edge-
                self.gap, self.edge*(1+0.5*sqrt(3))))
330     mdb.models['Model-1'].sketches['__profile__'].rotate(
            angle=-30.0, centerPoint=(-0.5*self.edge, 0.5*sqrt
            (3)*self.edge), objectList=(
331         mdb.models['Model-1'].sketches['__profile__'].
                geometry[2],
332         mdb.models['Model-1'].sketches['__profile__'].
                geometry[3],
333         mdb.models['Model-1'].sketches['__profile__'].
                geometry[4],
334         mdb.models['Model-1'].sketches['__profile__'].
                geometry[5]))
335     mdb.models['Model-1'].Part(dimensionality=TWO_D_PLANAR
            , name='Cohesive060',
336         type=DEFORMABLE_BODY)
337     mdb.models['Model-1'].parts['Cohesive060'].BaseShell(
            sketch=
338         mdb.models['Model-1'].sketches['__profile__'])
339     del mdb.models['Model-1'].sketches['__profile__']
340     # Coesive de angulo 120
341     mdb.models['Model-1'].ConstrainedSketch(name='
            __profile__', sheetSize=200.0)
342     mdb.models['Model-1'].sketches['__profile__'].
```

```
            rectangle(point1=(-0.5*self.edge,
343             0.5*sqrt(3)*self.edge), point2=(-0.5*self.edge-
                    self.gap, self.edge*(-1+0.5*sqrt(3))))
344         mdb.models['Model-1'].sketches['__profile__'].rotate(
                angle=30.0, centerPoint=(-0.5*self.edge, 0.5*sqrt
                (3)*self.edge), objectList=(
345             mdb.models['Model-1'].sketches['__profile__'].
                    geometry[2],
346             mdb.models['Model-1'].sketches['__profile__'].
                    geometry[3],
347             mdb.models['Model-1'].sketches['__profile__'].
                    geometry[4],
348             mdb.models['Model-1'].sketches['__profile__'].
                    geometry[5]))
349         mdb.models['Model-1'].Part(dimensionality=TWO_D_PLANAR
                , name='Cohesive120',
350             type=DEFORMABLE_BODY)
351         mdb.models['Model-1'].parts['Cohesive120'].BaseShell(
                sketch=
352             mdb.models['Model-1'].sketches['__profile__'])
353         del mdb.models['Model-1'].sketches['__profile__']
354     def generateAssembly(self):
355         # Criar e posicionar os hexagonos e coesivos
356         mdb.models['Model-1'].rootAssembly.DatumCsysByDefault(
                CARTESIAN)
357         l=0
358         for j in range(2,4*self.nr,4):
359             l=l+1
360             k=0
361             for i in range(2,2*self.nc,2):
362                 k=k+1
363                 # Monta hexagono
364                 mdb.models['Model-1'].rootAssembly.Instance(
                        dependent=ON,
365                     name='Instance-'+str(j+((k-1)%2)*2)+'-'+
                            str(i),
366                     part=mdb.models['Model-1'].parts['Hex-'+
                            str(j+((k-1)%2)*2)+'-'+str(i)])
367                 # Monta os tres cohesivos para o hexagono
368                 # Coesivo 0 graus
369                 mdb.models['Model-1'].rootAssembly.Instance(
                        dependent=ON,
370                     name='Instance-Cohesive-000-'+str(j+2+((k
                            -1)%2)*2)+'-'+str(i),
371                     part=mdb.models['Model-1'].parts['
                            Cohesive000'])
372                 # Coesivo 60 graus
373                 mdb.models['Model-1'].rootAssembly.Instance(
                        dependent=ON,
374                     name='Instance-Cohesive-060-'+str(j+1+((k
```

```
                    -1)%2)*2)+'-'+str(i-1),
375                 part=mdb.models['Model-1'].parts['
                        Cohesive060'])
376             # Coesivo 120 graus
377             mdb.models['Model-1'].rootAssembly.Instance(
                    dependent=ON,
378                 name='Instance-Cohesive-120-'+str(j-1+((k
                        -1)%2)*2)+'-'+str(i-1),
379                 part=mdb.models['Model-1'].parts['
                        Cohesive120'])
380             # posiciona as 4 partes (tres coesivos e dois
                    hexagonos)
381             if (k!=1   or  l!=1):
382                 mdb.models['Model-1'].rootAssembly.
                        translate(instanceList=
383                     ('Instance-'+str(j+((k-1)%2)*2)+'-'+
                            str(i),
384                     'Instance-Cohesive-000-'+str(j+2+((k
                            -1)%2)*2)+'-'+str(i),
385                     'Instance-Cohesive-060-'+str(j+1+((k
                            -1)%2)*2)+'-'+str(i-1),
386                     'Instance-Cohesive-120-'+str(j-1+((k
                            -1)%2)*2)+'-'+str(i-1)),
387                     vector=(((1.5*(self.edge)+(sqrt(3)/2)*
                            self.gap))*(k-1),
388                         ((k-1) % 2)*((sqrt(3)/2)*self.edge
                                +0.5*self.gap) + (l-1)*(self.
                                edge*sqrt(3)+self.gap) , 0.0))
389         # Monta elementos da aresta inferior
390         k=-1
391         for i in range(4,2*self.nc,4):
392             k=k+2
393             # Monta hexagono
394             mdb.models['Model-1'].rootAssembly.Instance(
                    dependent=ON,
395                 name='Instance-'+str(0)+'-'+str(i),
396                 part=mdb.models['Model-1'].parts['Hex-'+str(0)
                        +'-'+str(i)])
397             # Coesivo 0 graus
398             mdb.models['Model-1'].rootAssembly.Instance(
                    dependent=ON,
399                 name='Instance-Cohesive-000-'+str(2)+'-'+str(i
                        ),
400                 part=mdb.models['Model-1'].parts['Cohesive000'
                        ])
401             # Coesivo 60 graus
402             mdb.models['Model-1'].rootAssembly.Instance(
                    dependent=ON,
403                 name='Instance-Cohesive-060-'+str(1)+'-'+str(i
                        -1),
```

```
404                 part=mdb.models['Model-1'].parts['Cohesive060'
                        ])
405             # Posiciona as partes coesivas e os hex da aresta
406             mdb.models['Model-1'].rootAssembly.translate(
                    instanceList=
407                 ('Instance-'+str(0)+'-'+str(i),
408                  'Instance-Cohesive-060-'+str(1)+'-'+str(i-1),
409                  'Instance-Cohesive-000-'+str(2)+'-'+str(i)),
410                 vector=(k*(1.5*self.edge + 0.5*sqrt(3)*self.
                        gap), -(0.5*sqrt(3)*self.edge + 0.5*self.
                        gap), 0.0))
411         k=0
412         for i in range(2,2*self.nc,4):
413             k=k+1
414             # Coesivo 0 graus
415             mdb.models['Model-1'].rootAssembly.Instance(
                    dependent=ON,
416                 name='Instance-Cohesive-000-'+str(0)+'-'+str(i
                        ),
417                 part=mdb.models['Model-1'].parts['Cohesive000'
                        ])
418             # Posiciona as partes coesivas e os hex da aresta
419             mdb.models['Model-1'].rootAssembly.translate(
                    instanceList=
420                 ('Instance-Cohesive-000-'+str(0)+'-'+str(i),),
421                 vector=((k-1)*(3*self.edge + sqrt(3)*self.gap)
                        , -(sqrt(3)*self.edge + self.gap), 0.0))
422         # Monta elementos da aresta superior
423         k=-2
424         for i in range(2,2*self.nc,4):
425             k=k+2
426             # Monta hexagono
427             mdb.models['Model-1'].rootAssembly.Instance(
                    dependent=ON,
428                 name='Instance-'+str(4*self.nr+2)+'-'+str(i),
429                 part=mdb.models['Model-1'].parts['Hex-'+str(4*
                        self.nr+2)+'-'+str(i)])
430             # Coesivo 120 graus
431             mdb.models['Model-1'].rootAssembly.Instance(
                    dependent=ON,
432                 name='Instance-Cohesive-120-'+str(4*self.nr+1)
                        +'-'+str(i-1),
433                 part=mdb.models['Model-1'].parts['Cohesive120'
                        ])
434             # Posiciona as partes coesivas e os hex da aresta
435             mdb.models['Model-1'].rootAssembly.translate(
                    instanceList=
436                 ('Instance-'+str(4*self.nr+2)+'-'+str(i),
437                  'Instance-Cohesive-120-'+str(4*self.nr+1)+'-'+
                        str(i-1)),
```

```
438                 vector=((k/2)*((3)*self.edge + sqrt(3)*self.
                        gap), self.nr*(sqrt(3)*self.edge + self.
                        gap), 0.0))
439             # Monta elementos da aresta da esquerda
440             k=-2
441             for j in range(4,4*(self.nr+1),4):
442                 k=k+2
443                 # Monta hexagono
444                 mdb.models['Model-1'].rootAssembly.Instance(
                        dependent=ON,
445                     name='Instance-'+str(j)+'-'+str(0),
446                     part=mdb.models['Model-1'].parts['Hex-'+str(j)
                            +'-'+str(0)])
447                 # Coesivo 0 graus (metade)
448                 mdb.models['Model-1'].rootAssembly.Instance(
                        dependent=ON,
449                     name='Instance-Cohesive-000-'+str(j+2)+'-'+str
                            (0),
450                     part=mdb.models['Model-1'].parts['Cohesive000h
                            '])
451                 # Posiciona as partes coesivas e os hex da aresta
452                 mdb.models['Model-1'].rootAssembly.translate(
                        instanceList=
453                     ('Instance-Cohesive-000-'+str(j+2)+'-'+str(0)
                            ,),
454                     vector=(0.5*self.edge, 0.0, 0.0))
455                 mdb.models['Model-1'].rootAssembly.translate(
                        instanceList=
456                     ('Instance-'+str(j)+'-'+str(0),
457                     'Instance-Cohesive-000-'+str(j+2)+'-'+str(0)),
458                     vector=(-(1.5*self.edge + 0.5*sqrt(3)*self.gap
                            ), 0.5*(sqrt(3)*self.edge + self.gap) +
459                             0.5*k*(sqrt(3)*self.edge + self.gap), 0.0)
                                )
460             # Monta elementos da aresta da direita
461             mdb.models['Model-1'].rootAssembly.DatumCsysByDefault(
                    CARTESIAN)
462             l=0
463             for j in range(2,4*self.nr,4):
464                 l=l+1
465                 # Monta hexagono
466                 mdb.models['Model-1'].rootAssembly.Instance(
                        dependent=ON,
467                     name='Instance-'+str(j+((self.nc-1)%2)*2)+'-'+
                            str(2*self.nc),
468                     part=mdb.models['Model-1'].parts['Hex-'+str(j
                            +((self.nc-1)%2)*2)+'-'+str(2*self.nc)])
469                 # Monta os tres cohesivos para o hexagono
470                 # Coesivo 0 graus (metade)
471                 mdb.models['Model-1'].rootAssembly.Instance(
```

```
                dependent=ON,
472             name='Instance-Cohesive-000-'+str(j+2+((self.
                    nc-1)%2)*2)+'-'+str(2*self.nc),
473             part=mdb.models['Model-1'].parts['Cohesive000h
                    '])
474         # Coesivo 60 graus
475         mdb.models['Model-1'].rootAssembly.Instance(
                dependent=ON,
476             name='Instance-Cohesive-060-'+str(j+1+((self.
                    nc-1)%2)*2)+'-'+str(2*self.nc-1),
477             part=mdb.models['Model-1'].parts['Cohesive060'
                    ])
478         # Coesivo 120 graus
479         mdb.models['Model-1'].rootAssembly.Instance(
                dependent=ON,
480             name='Instance-Cohesive-120-'+str(j-1+((self.
                    nc-1)%2)*2)+'-'+str(2*self.nc-1),
481             part=mdb.models['Model-1'].parts['Cohesive120'
                    ])
482         # posiciona as 4 partes (tres coesivos e hexagono)
483         mdb.models['Model-1'].rootAssembly.translate(
                instanceList=
484             ('Instance-'+str(j+((self.nc-1)%2)*2)+'-'+str
                    (2*self.nc),
485             'Instance-Cohesive-000-'+str(j+2+((self.nc-1)%
                    2)*2)+'-'+str(2*self.nc),
486             'Instance-Cohesive-060-'+str(j+1+((self.nc-1)%
                    2)*2)+'-'+str(2*self.nc-1),
487             'Instance-Cohesive-120-'+str(j-1+((self.nc-1)%
                    2)*2)+'-'+str(2*self.nc-1)),
488              vector=(((1.5*(self.edge)+(sqrt(3)/2)*self.
                    gap))*(self.nc-1),
489                  ((self.nc-1) % 2)*((sqrt(3)/2)*self.edge
                        +0.5*self.gap) + (1-1)*(self.edge*sqrt
                        (3)+self.gap) , 0.0))
490     ## Coloca coesivo no canto inferior direito
491     # Coesivo 0 graus (metade)
492     if (self.nc % 2) == 1:
493         mdb.models['Model-1'].rootAssembly.Instance(
                dependent=ON,
494             name='Instance-Cohesive-000-'+str(0)+'-'+str
                    (2*self.nc),
495             part=mdb.models['Model-1'].parts['Cohesive000h
                    '])
496         mdb.models['Model-1'].rootAssembly.translate(
                instanceList=
497             ('Instance-Cohesive-000-'+str(0)+'-'+str(2*
                    self.nc),),
498              vector=(((1.5*(self.edge)+(sqrt(3)/2)*self.
                    gap))*(self.nc-1),-sqrt(3)*self.edge-self
```

```
                .gap , 0.0))
499         # Monta elementos dos vertices
500         ## Monta hex do vertice inferior esquerdo
501         mdb.models['Model-1'].rootAssembly.Instance(dependent=
                ON,
502             name='Instance-'+str(0)+'-'+str(0),
503             part=mdb.models['Model-1'].parts['Hex-'+str(0)+'-'
                    +str(0)])
504         mdb.models['Model-1'].rootAssembly.Instance(dependent=
                ON,
505             name='Instance-Cohesive-000-'+str(2)+'-'+str(0),
506             part=mdb.models['Model-1'].parts['Cohesive000h'])
507         ### posiciona o elemento coesivo
508         mdb.models['Model-1'].rootAssembly.translate(
                instanceList=
509             ('Instance-Cohesive-000-'+str(2)+'-'+str(0),),
510             vector=(-(1.0*self.edge + 0.5*sqrt(3)*self.gap),
                    -0.5*(sqrt(3)*self.edge + self.gap) , 0.0))
511         ### posiciona o hex
512         mdb.models['Model-1'].rootAssembly.translate(
                instanceList=
513             ('Instance-'+str(0)+'-'+str(0),),
514             vector=(-(self.edge + 0.5*sqrt(3)*self.gap), -0.5*
                    self.gap, 0.0))
515         ## posiciona o hex do vertice superior direito
516         if (self.nc % 2) == 1:
517             mdb.models['Model-1'].rootAssembly.Instance(
                    dependent=ON,
518                 name='Instance-'+str(4*self.nr+2)+'-'+str(2*
                        self.nc),
519                 part=mdb.models['Model-1'].parts['Hex-'+str(4*
                        self.nr+2)+'-'+str(2*self.nc)])
520             # Coesivo 120 graus
521             mdb.models['Model-1'].rootAssembly.Instance(
                    dependent=ON,
522                 name='Instance-Cohesive-120-'+str(4*self.nr+1)
                        +'-'+str(2*self.nc-1),
523                 part=mdb.models['Model-1'].parts['Cohesive120'
                        ])
524             mdb.models['Model-1'].rootAssembly.translate(
                    instanceList=
525                 ('Instance-'+str(4*self.nr+2)+'-'+str(2*self.
                        nc),
526                 'Instance-Cohesive-120-'+str(4*self.nr+1)+'-'+
                        str(2*self.nc-1)),
527                 vector=(((1.5*(self.edge)+(sqrt(3)/2)*self.gap
                        ))*(self.nc-1), self.nr*(sqrt(3)*self.edge
                         + self.gap), 0.0))
528         else:
529             mdb.models['Model-1'].rootAssembly.Instance(
```

```
                    dependent=ON,
530                 name='Instance-'+str(0)+'-'+str(2*self.nc),
531                 part=mdb.models['Model-1'].parts['Hex-'+str(0)
                        +'-'+str(2*self.nc)])
532             mdb.models['Model-1'].rootAssembly.Instance(
                    dependent=ON,
533                 name='Instance-Cohesive-000-'+str(2)+'-'+str
                        (2*self.nc),
534                 part=mdb.models['Model-1'].parts['Cohesive000h
                        '])
535             mdb.models['Model-1'].rootAssembly.Instance(
                    dependent=ON,
536                 name='Instance-Cohesive-060-'+str(1)+'-'+str
                        (2*self.nc-1),
537                 part=mdb.models['Model-1'].parts['Cohesive060'
                        ])
538             mdb.models['Model-1'].rootAssembly.translate(
                    instanceList=
539                 ('Instance-'+str(0)+'-'+str(2*self.nc),),
540                  vector=(((1.5*(self.edge)+(sqrt(3)/2)*self.
                        gap))*(self.nc-1)-0.5*self.edge, -0.5*
                        self.gap, 0.0))
541             mdb.models['Model-1'].rootAssembly.translate(
                    instanceList=
542                 ('Instance-Cohesive-000-'+str(2)+'-'+str(2*
                        self.nc),),
543                  vector=(((1.5*(self.edge)+(sqrt(3)/2)*self.
                        gap))*(self.nc-1), -0.5*(sqrt(3)*self.
                        edge+ self.gap), 0.0))
544             mdb.models['Model-1'].rootAssembly.translate(
                    instanceList=
545                 ('Instance-Cohesive-060-'+str(1)+'-'+str(2*
                        self.nc-1),),
546                  vector=(((1.5*(self.edge)+(sqrt(3)/2)*self.
                        gap))*(self.nc-1), -0.5*(sqrt(3)*self.
                        edge+ self.gap), 0.0))
547     def generateInteractions(self):
548         # Gera as interacoes para o centro do honeycomb
549         for j in range(2,4*self.nr,4):
550             k=0
551             for i in range(2,2*self.nc,2):
552                 k=k+1
553                 index = j+((k-1)%2)*2
554                 mdb.models['Model-1'].Tie(adjust=ON, master=
                        Region(
555                     side1Edges=mdb.models['Model-1'].
                            rootAssembly.instances['Instance-'+str
                            (index)+'-'+str(i)].edges[4:5])
556                     , name='Hex-'+str(index)+'-'+str(i)+'-Coe-
                            '+str(index+2)+'-'+str(i),
```

```
557                 positionToleranceMethod=COMPUTED, slave=
                        Region(
558                 side1Edges=mdb.models['Model-1'].
                        rootAssembly.instances[
559                     'Instance-Cohesive-000-'+str(index+2)+
                            '-'+str(i)].edges[0:1])
560                 , thickness=ON, tieRotations=ON)
561             mdb.models['Model-1'].Tie(adjust=ON, master=
                    Region(
562                 side1Edges=mdb.models['Model-1'].
                        rootAssembly.instances['Instance-'+str
                        (index)+'-'+str(i)].edges[1:2])
563                 , name='Hex-'+str(index)+'-'+str(i)+'-Coe-
                        '+str(index+1)+'-'+str(i-1),
564                 positionToleranceMethod=COMPUTED, slave=
                        Region(
565                 side1Edges=mdb.models['Model-1'].
                        rootAssembly.instances[
566                     'Instance-Cohesive-060-'+str(index+1)+
                            '-'+str(i-1)].edges[1:2])
567                 , thickness=ON, tieRotations=ON)
568             mdb.models['Model-1'].Tie(adjust=ON, master=
                    Region(
569                 side1Edges=mdb.models['Model-1'].
                        rootAssembly.instances['Instance-'+str
                        (index)+'-'+str(i)].edges[5:6])
570                 , name='Hex-'+str(index)+'-'+str(i)+'-Coe-
                        '+str(index-1)+'-'+str(i-1),
571                 positionToleranceMethod=COMPUTED, slave=
                        Region(
572                 side1Edges=mdb.models['Model-1'].
                        rootAssembly.instances[
573                     'Instance-Cohesive-120-'+str(index-1)+
                            '-'+str(i-1)].edges[3:4])
574                 , thickness=ON, tieRotations=ON)
575             mdb.models['Model-1'].Tie(adjust=ON, master=
                    Region(
576                 side1Edges=mdb.models['Model-1'].
                        rootAssembly.instances['Instance-'+str
                        (index)+'-'+str(i)].edges[10:11])
577                 , name='Hex-'+str(index)+'-'+str(i)+'-Coe-
                        '+str(index-2)+'-'+str(i),
578                 positionToleranceMethod=COMPUTED, slave=
                        Region(
579                 side1Edges=mdb.models['Model-1'].
                        rootAssembly.instances[
580                     'Instance-Cohesive-000-'+str(index-2)+
                            '-'+str(i)].edges[2:3])
581                 , thickness=ON, tieRotations=ON)
582             mdb.models['Model-1'].Tie(adjust=ON, master=
```

```
                    Region(
583                     side1Edges=mdb.models['Model-1'].
                            rootAssembly.instances['Instance-'+str
                            (index)+'-'+str(i)].edges[11:12])
584                     , name='Hex-'+str(index)+'-'+str(i)+'-Coe-
                            '+str(index-1)+'-'+str(i+1),
585                     positionToleranceMethod=COMPUTED, slave=
                            Region(
586                     side1Edges=mdb.models['Model-1'].
                            rootAssembly.instances[
587                         'Instance-Cohesive-060-'+str(index-1)+
                                '-'+str(i+1)].edges[3:4])
588                     , thickness=ON, tieRotations=ON)
589                 mdb.models['Model-1'].Tie(adjust=ON, master=
                        Region(
590                     side1Edges=mdb.models['Model-1'].
                            rootAssembly.instances['Instance-'+str
                            (index)+'-'+str(i)].edges[8:9])
591                     , name='Hex-'+str(j)+'-'+str(i)+'-Coe-'+
                            str(j+1)+'-'+str(i+1),
592                     positionToleranceMethod=COMPUTED, slave=
                            Region(
593                     side1Edges=mdb.models['Model-1'].
                            rootAssembly.instances[
594                         'Instance-Cohesive-120-'+str(index+1)+
                                '-'+str(i+1)].edges[1:2])
595                     , thickness=ON, tieRotations=ON)
596             # Interacoes da aresta inferior
597             for i in range(4,2*self.nc,4):
598                 mdb.models['Model-1'].Tie(adjust=ON, master=Region
                        (
599                     side1Edges=mdb.models['Model-1'].rootAssembly.
                            instances['Instance-0-'+str(i)].edges
                            [5:6])
600                     , name='Hex-'+str(0)+'-'+str(i)+'-Coe-1'+'-'+
                            str(i-1),
601                     positionToleranceMethod=COMPUTED, slave=Region
                            (
602                     side1Edges=mdb.models['Model-1'].rootAssembly.
                            instances['Instance-Cohesive-060-1-'+str(i
                            -1)].edges[1:2])
603                     , thickness=ON, tieRotations=ON)
604                 mdb.models['Model-1'].Tie(adjust=ON, master=Region
                        (
605                     side1Edges=mdb.models['Model-1'].rootAssembly.
                            instances['Instance-0-'+str(i)].edges
                            [4:5])
606                     , name='Hex-'+str(0)+'-'+str(i)+'-Coe-2'+'-'+
                            str(i),
607                     positionToleranceMethod=COMPUTED, slave=Region
```

```
                    (
608                 side1Edges=mdb.models['Model-1'].rootAssembly.
                        instances['Instance-Cohesive-000-2-'+str(i
                        )].edges[0:1])
609                 , thickness=ON, tieRotations=ON)
610             mdb.models['Model-1'].Tie(adjust=ON, master=Region
                    (
611                 side1Edges=mdb.models['Model-1'].rootAssembly.
                        instances['Instance-0-'+str(i)].edges
                        [2:3])
612                 , name='Hex-'+str(0)+'-'+str(i)+'-Coe-1'+'-'+
                        str(i+1),
613                 positionToleranceMethod=COMPUTED, slave=Region
                        (
614                 side1Edges=mdb.models['Model-1'].rootAssembly.
                        instances['Instance-Cohesive-120-1-'+str(i
                        +1)].edges[1:2])
615                 , thickness=ON, tieRotations=ON)
616         # Interacoes da aresta superior
617         for i in range(2,2*self.nc,4):
618             mdb.models['Model-1'].Tie(adjust=ON, master=Region
                    (
619                 side1Edges=mdb.models['Model-1'].rootAssembly.
                        instances['Instance-'+str(4*self.nr+2)+'-'
                        +str(i)].edges[1:2])
620                 , name='Hex-'+str(4*self.nr+2)+'-'+str(i)+'-
                        Coe-'+str(4*self.nr+1)+'-'+str(i+1),
621                 positionToleranceMethod=COMPUTED, slave=Region
                        (
622                 side1Edges=mdb.models['Model-1'].rootAssembly.
                        instances['Instance-Cohesive-060-'+str(4*
                        self.nr+1)+'-'+str(i+1)].edges[3:4])
623                 , thickness=ON, tieRotations=ON)
624             mdb.models['Model-1'].Tie(adjust=ON, master=Region
                    (
625                 side1Edges=mdb.models['Model-1'].rootAssembly.
                        instances['Instance-'+str(4*self.nr+2)+'-'
                        +str(i)].edges[6:7])
626                 , name='Hex-'+str(4*self.nr+2)+'-'+str(i)+'-
                        Coe-'+str(4*self.nr)+'-'+str(i),
627                 positionToleranceMethod=COMPUTED, slave=Region
                        (
628                 side1Edges=mdb.models['Model-1'].rootAssembly.
                        instances['Instance-Cohesive-000-'+str(4*
                        self.nr)+'-'+str(i)].edges[2:3])
629                 , thickness=ON, tieRotations=ON)
630             mdb.models['Model-1'].Tie(adjust=ON, master=Region
                    (
631                 side1Edges=mdb.models['Model-1'].rootAssembly.
                        instances['Instance-'+str(4*self.nr+2)+'-'
```

```
                +str(i)].edges[5:6])
632             , name='Hex-'+str(4*self.nr+1)+'-'+str(i-1)+'-
                    Coe-'+str(4*self.nr+1)+'-'+str(i-1),
633             positionToleranceMethod=COMPUTED, slave=Region
                    (
634             side1Edges=mdb.models['Model-1'].rootAssembly.
                    instances['Instance-Cohesive-120-'+str(4*
                    self.nr+1)+'-'+str(i-1)].edges[3:4])
635             , thickness=ON, tieRotations=ON)
636         # Interacoes do lado direito
637         for j in range(2,4*(self.nr),4):
638             index=j+((self.nc+1)%2)*2
639             mdb.models['Model-1'].Tie(adjust=ON, master=Region
                    (
640             side1Edges=mdb.models['Model-1'].rootAssembly.
                    instances['Instance-'+str(index)+'-'+str
                    (2*self.nc)].edges[4:5])
641             , name='Hex-'+str(index)+'-'+str(2*self.nc)+'-
                    Coe-'+str(index-2)+'-'+str(2*self.nc),
642             positionToleranceMethod=COMPUTED, slave=Region
                    (
643             side1Edges=mdb.models['Model-1'].rootAssembly.
                    instances['Instance-Cohesive-000-'+str(
                    index-2)+'-'+str(2*self.nc)].edges[2:3])
644             , thickness=ON, tieRotations=ON)
645             mdb.models['Model-1'].Tie(adjust=ON, master=Region
                    (
646             side1Edges=mdb.models['Model-1'].rootAssembly.
                    instances['Instance-'+str(index)+'-'+str
                    (2*self.nc)].edges[8:9])
647             , name='Hex-'+str(index)+'-'+str(2*self.nc)+'-
                    Coe-'+str(index-1)+'-'+str(2*self.nc-1),
648             positionToleranceMethod=COMPUTED, slave=Region
                    (
649             side1Edges=mdb.models['Model-1'].rootAssembly.
                    instances['Instance-Cohesive-120-'+str(
                    index-1)+'-'+str(2*self.nc-1)].edges[3:4])
650             , thickness=ON, tieRotations=ON)
651             mdb.models['Model-1'].Tie(adjust=ON, master=Region
                    (
652             side1Edges=mdb.models['Model-1'].rootAssembly.
                    instances['Instance-'+str(index)+'-'+str
                    (2*self.nc)].edges[7:8])
653             , name='Hex-'+str(index)+'-'+str(2*self.nc)+'-
                    Coe-'+str(index+1)+'-'+str(2*self.nc-1),
654             positionToleranceMethod=COMPUTED, slave=Region
                    (
655             side1Edges=mdb.models['Model-1'].rootAssembly.
                    instances['Instance-Cohesive-060-'+str(
                    index+1)+'-'+str(2*self.nc-1)].edges[1:2])
```

```
656                 , thickness=ON, tieRotations=ON)
657             mdb.models['Model-1'].Tie(adjust=ON, master=Region
                    (
658                 side1Edges=mdb.models['Model-1'].rootAssembly.
                        instances['Instance-'+str(index)+'-'+str
                        (2*self.nc)].edges[2:3])
659                 , name='Hex-'+str(index)+'-'+str(2*self.nc)+'-
                        Coe-'+str(index+2)+'-'+str(2*self.nc),
660                 positionToleranceMethod=COMPUTED, slave=Region
                        (
661                 side1Edges=mdb.models['Model-1'].rootAssembly.
                        instances['Instance-Cohesive-000-'+str(
                        index+2)+'-'+str(2*self.nc)].edges[0:1])
662                 , thickness=ON, tieRotations=ON)
663         # Interacoes do lado esquerdo
664         for j in range(4,4*(self.nr+1),4):
665             mdb.models['Model-1'].Tie(adjust=ON, master=Region(
666                 side1Edges=mdb.models['Model-1'].rootAssembly.
                        instances['Instance-'+str(j)+'-0'].edges
                        [4:5])
667                 , name='Hex-'+str(j)+'-'+str(0)+'-Coe-'+str(j
                        +2)+'-0',
668                 positionToleranceMethod=COMPUTED, slave=Region(
669                 side1Edges=mdb.models['Model-1'].rootAssembly.
                        instances['Instance-Cohesive-000-'+str(j+2)
                        +'-0'].edges[0:1])
670                 , thickness=ON, tieRotations=ON)
671             mdb.models['Model-1'].Tie(adjust=ON, master=Region(
672                 side1Edges=mdb.models['Model-1'].rootAssembly.
                        instances['Instance-'+str(j)+'-0'].edges
                        [8:9])
673                 , name='Hex-'+str(j)+'-'+str(0)+'-Coe-'+str(j
                        +1)+'-1',
674                 positionToleranceMethod=COMPUTED, slave=Region(
675                 side1Edges=mdb.models['Model-1'].rootAssembly.
                        instances['Instance-Cohesive-120-'+str(j+1)
                        +'-1'].edges[1:2])
676                 , thickness=ON, tieRotations=ON)
677             mdb.models['Model-1'].Tie(adjust=ON, master=Region(
678                 side1Edges=mdb.models['Model-1'].rootAssembly.
                        instances['Instance-'+str(j)+'-0'].edges
                        [7:8])
679                 , name='Hex-'+str(j)+'-'+str(0)+'-Coe-'+str(j
                        -1)+'-1',
680                 positionToleranceMethod=COMPUTED, slave=Region(
681                 side1Edges=mdb.models['Model-1'].rootAssembly.
                        instances['Instance-Cohesive-060-'+str(j-1)
                        +'-1'].edges[3:4])
682                 , thickness=ON, tieRotations=ON)
683             mdb.models['Model-1'].Tie(adjust=ON, master=Region(
```

```
684                 side1Edges=mdb.models['Model-1'].rootAssembly.
                        instances['Instance-'+str(j)+'-0'].edges
                        [2:3])
685                 , name='Hex-'+str(j)+'-'+str(0)+'-Coe-'+str(j
                        -2)+'-0',
686                 positionToleranceMethod=COMPUTED, slave=Region(
687                 side1Edges=mdb.models['Model-1'].rootAssembly.
                        instances['Instance-Cohesive-000-'+str(j-2)
                        +'-0'].edges[2:3])
688                 , thickness=ON, tieRotations=ON)
689             # Interacoes do canto inferior esquerdo
690             mdb.models['Model-1'].Tie(adjust=ON, master=Region(
691                 side1Edges=mdb.models['Model-1'].rootAssembly.
                        instances['Instance-0-0'].edges[3:4])
692                 , name='Hex-0-0-Coe-2-0', positionToleranceMethod=
                        COMPUTED, slave=Region(
693                 side1Edges=mdb.models['Model-1'].rootAssembly.
                        instances['Instance-Cohesive-000-2-0'].edges
                        [0:1])
694                 , thickness=ON, tieRotations=ON)
695             mdb.models['Model-1'].Tie(adjust=ON, master=Region(
696                 side1Edges=mdb.models['Model-1'].rootAssembly.
                        instances['Instance-0-0'].edges[2:3])
697                 , name='Hex-0-0-coe-1-1', positionToleranceMethod=
                        COMPUTED, slave=Region(
698                 side1Edges=mdb.models['Model-1'].rootAssembly.
                        instances['Instance-Cohesive-120-1-1'].edges
                        [1:2])
699                 , thickness=ON, tieRotations=ON)
700             # Interacoes dos cantos direitos
701             if (self.nc % 2) == 1:
702                 mdb.models['Model-1'].Tie(adjust=ON, master=Region
                        (
703                     side1Edges=mdb.models['Model-1'].rootAssembly.
                            instances['Instance-'+str(4*self.nr+2)+'-'
                            +str(2*self.nc)].edges[1:2])
704                     , name='Hex-'+str(4*self.nr+2)+'-'+str(2*self.
                            nc)+'-Coe-'+str(4*self.nr)+'-'+str(2*self.
                            nc),
705                     positionToleranceMethod=COMPUTED, slave=Region
                            (
706                     side1Edges=mdb.models['Model-1'].rootAssembly.
                            instances['Instance-Cohesive-000-'+str(4*
                            self.nr)+'-'+str(2*self.nc)].edges[2:3])
707                                 , thickness=ON, tieRotations=ON)
708                 mdb.models['Model-1'].Tie(adjust=ON, master=Region
                        (
709                     side1Edges=mdb.models['Model-1'].rootAssembly.
                            instances['Instance-'+str(4*self.nr+2)+'-'
                            +str(2*self.nc)].edges[4:5])
```

```
710                     , name='Hex-'+str(4*self.nr+2)+'-'+str(2*self.
                            nc)+'-Coe-'+str(4*self.nr+1)+'-'+str(2*
                            self.nc-1),
711                     positionToleranceMethod=COMPUTED, slave=Region
                            (
712                     side1Edges=mdb.models['Model-1'].rootAssembly.
                            instances['Instance-Cohesive-120-'+str(4*
                            self.nr+1)+'-'+str(2*self.nc-1)].edges
                            [3:4])
713                     , thickness=ON, tieRotations=ON)
714             else:
715                 mdb.models['Model-1'].Tie(adjust=ON, master=Region
                        (
716                     side1Edges=mdb.models['Model-1'].rootAssembly.
                            instances['Instance-'+str(0)+'-'+str(2*
                            self.nc)].edges[2:3]) #Acertar arestas
717                     , name='Hex-'+str(0)+'-'+str(2*self.nc)+'-Coe-
                            '+str(1)+'-'+str(2*self.nc-1),
718                     positionToleranceMethod=COMPUTED, slave=Region
                            (
719                         side1Edges=mdb.models['Model-1'].
                                rootAssembly.instances['Instance-
                                Cohesive-000-'+str(2)+'-'+str(2*self.
                                nc)].edges[0:1])
720                     , thickness=ON, tieRotations=ON)
721                 mdb.models['Model-1'].Tie(adjust=ON, master=Region
                        (
722                     side1Edges=mdb.models['Model-1'].rootAssembly.
                            instances['Instance-'+str(0)+'-'+str(2*
                            self.nc)].edges[3:4])
723                     , name='Hex-'+str(0)+'-'+str(2*self.nc-1)+'-
                            Coe-'+str(2)+'-'+str(2*self.nc),
724                     positionToleranceMethod=COMPUTED, slave=Region
                            (
725                     side1Edges=mdb.models['Model-1'].rootAssembly.
                            instances['Instance-Cohesive-060-'+str(1)+
                            '-'+str(2*self.nc-1)].edges[1:2])
726                     , thickness=ON, tieRotations=ON)
727     # Propriedades dos hexagonos
728     def setHexsProperty(self):
729         distrib = numpy.loadtxt('distrib.txt')
730         mdb.models['Model-1'].Material(name='Material-Hex')
731         mdb.models['Model-1'].materials['Material-Hex'].
                Elastic(table=((148000.0,
732             127000.0, 148000.0, 0.34, 0.34, 0.34, 47000.0,
                    47000.0, 47000.0), ), type=
733             ENGINEERING_CONSTANTS)
734         mdb.models['Model-1'].materials['Material-Hex'].
                Expansion(table=((3.87e-06,
735             -1.2e-06, 3.87e-06), ), type=ORTHOTROPIC)
```

```
736             mdb.models['Model-1'].HomogeneousSolidSection(material
                    ='Material-Hex', name=
737                         'Section-Hex', thickness=None)
738             i=0
739             for key in mdb.models['Model-1'].parts.keys():
740                 if 'Hex' in key:
741                     mdb.models['Model-1'].parts[key].
                            MaterialOrientation(
742                         additionalRotationField='',
                                additionalRotationType=ROTATION_ANGLE,
                                 angle=
743                         distrib[i], axis=AXIS_3, fieldName='',
                                localCsys=None, orientationType=SYSTEM
                                ,
744                         region=Region(faces=mdb.models['Model-1'].
                                parts[key].faces[:]), stackDirection=
                                STACK_3)
745                     mdb.models['Model-1'].parts[key].
                            SectionAssignment(offset=0.0,
746                         offsetField='', offsetType=MIDDLE_SURFACE,
                                 region=Region(
747                         faces=mdb.models['Model-1'].parts[key].
                                faces[:]), sectionName=
748                         'Section-Hex', thicknessAssignment=
                                FROM_SECTION)
749                     i+=1
750         # Propriedades dos coesivos
751         def setCohesiveProperty(self):
752             mdb.models['Model-1'].Material(name='Material-Cohesive
                    ')
753             mdb.models['Model-1'].materials['Material-Cohesive'].
                    Elastic(table=((740000000.0,
754                     335000000.0, 335000000.0), ), type=TRACTION)
755             mdb.models['Model-1'].materials['Material-Cohesive'].
                    MaxsDamageInitiation(table=((
756                     700.0, 700.0, 700.0), ))
757             mdb.models['Model-1'].materials['Material-Cohesive'].
                    maxsDamageInitiation.DamageEvolution(
758                     table=((0.000543, ), ), type=ENERGY)
759             mdb.models['Model-1'].CohesiveSection(material='
                    Material-Cohesive', name=
760                         'Section-Cohesive', outOfPlaneThickness=
                                None, response=TRACTION_SEPARATION)
761             cohesives = ['000','000h','060','120']
762             for cohesive in cohesives:
763                 mdb.models['Model-1'].parts['Cohesive'+cohesive].
                        MaterialOrientation(
764                     additionalRotationField='',
                            additionalRotationType=ROTATION_ANGLE,
                            angle=
```

```
765                     float(cohesive[0:3]), axis=AXIS_3, fieldName='
                            ', localCsys=None, orientationType=SYSTEM,
766                     region=Region(faces=mdb.models['Model-1'].
                            parts['Cohesive'+cohesive].faces[:]),
                            stackDirection=STACK_3)
767                 mdb.models['Model-1'].parts['Cohesive'+cohesive].
                        SectionAssignment(offset=0.0,
768                     offsetField='', offsetType=MIDDLE_SURFACE,
                            region=Region(
769                     faces=mdb.models['Model-1'].parts['Cohesive'+
                            cohesive].faces[:]), sectionName=
770                     'Section-Cohesive', thicknessAssignment=
                            FROM_SECTION)
771         # Gera a malha
772         def generateMesh(self, seed=0.1):
773             for key in mdb.models['Model-1'].parts.keys():
774                 if 'Hex' in key:
775                     mdb.models['Model-1'].parts[key].seedPart(
                            deviationFactor=0.1, minSizeFactor=0.1,
                            size=seed)
776                     mdb.models['Model-1'].parts[key].
                            setMeshControls(regions=
777                         mdb.models['Model-1'].parts[key].faces[:],
                                technique=STRUCTURED)
778                     mdb.models['Model-1'].parts[key].generateMesh
                            ()
779                     mdb.models['Model-1'].parts[key].
                            setElementType(elemTypes=(ElemType(
780                         elemCode=CPS4I, elemLibrary=STANDARD),
781                         ElemType(elemCode=CPS3, elemLibrary=
                                STANDARD)), regions=(
782                         mdb.models['Model-1'].parts[key].faces[:],
                                ))
783                 else:
784                     mdb.models['Model-1'].parts[key].seedPart(
                            deviationFactor=0.1, minSizeFactor=0.1,
                            size=0.5*seed)
785                     mdb.models['Model-1'].parts[key].
                            setMeshControls(regions=
786                         mdb.models['Model-1'].parts[key].faces[:],
                                technique=STRUCTURED, elemShape=QUAD)
787                     mdb.models['Model-1'].parts[key].generateMesh
                            ()
788                     mdb.models['Model-1'].parts[key].
                            setElementType(elemTypes=(ElemType(
789                         elemCode=COH2D4, elemLibrary=STANDARD,
                                elemDeletion=ON, viscosity=0.002),
790                         ElemType(elemCode=UNKNOWN_TRI, elemLibrary
                                =STANDARD)), regions=(
791                         mdb.models['Model-1'].parts[key].faces[:],
```