**UNESP**
**Faculdade de Engenharia do Campus de Guaratinguetá**

**Guaratinguetá**
**2009**

LEONARDO GRANDO

# DESENVOLVIMENTO DE UM PROGRAMA EM JAVA PARA CÁLCULO DA CINÉTICA DE SINTERIZAÇÃO DE VIDROS

Monografia apresentada à Faculdade de Engenharia do Campus de Guaratinguetá, Universidade Estadual Paulista, para a obtenção do grau de Engenheiro de Materiais

Orientador: Prof. Dr. Eduardo Bellini Ferreira

Guaratinguetá
2009

Grando, Leonardo
G754d Desenvolvimento de um software em Java para cálculo da cinética de sinterização de vidros. / Leonardo Grando – Guaratinguetá : [s.n], 2009.
78f. : il.
Bibliografia: f. 56

Trabalho de Graduação em Engenharia de Materiais – Universidade Estadual Paulista, Faculdade de Engenharia de Guaratinguetá, 2009.
Orientador: Prof. Dr. Eduardo Bellini Ferreira

1. Vidro I. Título

CDU 666.1

UNESP **UNIVERSIDADE ESTADUAL PAULISTA**
**Faculdade de Engenharia do Campus de Guaratinguetá**

# DESENVOLVIMENTO DE UM SOFTWARE EM JAVA PARA CÁLCULO DA CINÉTICA DE SINTERIZAÇÃO DE VIDROS

**LEONARDO GRANDO**

ESTA MONOGRAFIA FOI JULGADA ADEQUADA PARA A OBTENÇÃO DO GRAU DE

**"ENGENHEIRO DE MATERIAIS"**

APROVADA EM SUA FORMA FINAL PELA COORDENAÇÃO DO CURSO DE ENGENHARIA DE MATERIAIS

Profa. Dra. Ana Paula Rosifini Alves Claro
Coordenadora do Curso de Graduação em Engenharia de Materiais

**BANCA EXAMINADORA:**

Prof. Dr. Eduardo Bellini Ferreira

Prof. Dr. José Luiz Minatti

Prof. Dr. Luis Rogério de Oliveira Hein

Dezembro de 2009

# DADOS CURRICULARES

LEONARDO GRANDO

| | |
|---|---|
| NASCIMENTO | 18.09.1984 – LARANJAL PAULISTA / SP |
| FILIAÇÃO | Antonio Cesare Grando<br>Teresinha Silvestrin Grando |
| 2005/2009 | Graduação em Engenharia de Materiais<br>UNESP – Campus de Guaratinguetá |

**Dedicatória**

De modo especial, aos meus pais, que foram os incentivadores para que eu concluísse o curso de Engenharia de Materiais, e à minha namorada Patrícia.

## AGRADECIMENTOS

Em primeiro lugar agradeço a Deus, fonte da vida e da graça. Agradeço pela minha vida, minha inteligência, minha família e meus amigos,

ao meu orientador, *Prof. Dr. Eduardo Bellini Ferreira* que jamais deixou de me incentivar. Sem a sua orientação, dedicação e auxílio, o estudo aqui apresentado seria praticamente impossível.

aos meus pais Teresinha Silvestrin Grando e Antonio Cesare Grando, que apesar das dificuldades enfrentadas, sempre incentivaram meus estudos,

aos meus irmãos e irmãs *Daniel Grando, Rosevaldo Grando*, *Andréia Grando Bernardes*, *Rosana Grando Zanettini* e famílias pelo incentivo nos estudos,

a minha namorada *Patrícia Rodrigues Cocato,* amiga e companheira que com carinho sempre esteve ao meu lado me incentivando no estudo para que este trabalho fosse realizado,

ao amigo *Rafael Sérgio de Mattos,* pela troca de conhecimento nos estudos de engenharia,

aos professores do Departamento de Materiais e Tecnologia da UNESP-Guaratinguetá, pelo empenho em ensinar a arte que é a Engenharia de Materiais,

aos professores Dra. Cassilda Maria Ribeiro e Dr. Edson Luiz França Senne pelo apoio na elaboração dos métodos de cálculos numéricos.

às Senhoras *Ana Antunes Bartelega e Ana Laura Bartelega* que sempre me acolheram com alegria em vossas casas,

às funcionárias da Biblioteca do Campus de Guaratinguetá pela dedicação, presteza e principalmente pela vontade de ajudar.

Este trabalho contou com apoio da seguinte entidade

- FAPESP – através do contrato nº 2009/06867-0

"O conhecimento torna a alma jovem e diminui a amargura da velhice. Colhe, pois, a sabedoria. Armazena suavidade para o amanhã."

Leonardo da Vinci

GRANDO, L. **Desenvolvimento de um software em Java para cálculo da cinética de sinterização de vidros.** 2009. 91 f. Monografia para conclusão de curso de Engenharia de Materiais – Faculdade de Engenharia do Campus de Guaratinguetá, Universidade Estadual Paulista, Guaratinguetá, 2009.

## RESUMO

Ferramentas computacionais são cada vez mais utilizadas para a simulação de fenômenos físicos, permitindo economizar tempo e dinheiro no desenvolvimento de novos processos e materiais. O presente trabalho tem como objetivo utilizar a ferramenta JAVA para o desenvolvimento de um programa para cálculos da cinética de sinterização de materiais vítreos para a fabricação de vitrocerâmicas. Por esse processo, vidros especiais são moídos, compactados e sinterizados, com simultânea ou posterior cristalização das partículas. Um recente equacionamento do processo de sinterização de vidros por fluxo viscoso, denominado modelo de Clusters, considera o efeito da cristalização superficial das partículas, que limita a densificação (eliminação de poros) do material, e permite lidar com misturas de partículas de diferentes tamanhos. Foi desenvolvido um programa em linguagem JAVA, baseado no modelo de Clusters, para cálculos da cinética de sinterização de compactos de partículas de vidro em pó com diferentes tamanhos de partículas e com cristalização concorrente. Dados publicados na literatura para vidros no sistema $Al_2O_3$-$B_2O_3$-$SiO_2$ e CaO-MgO-$SiO_2$ foram usados para conferir os resultados do programa, expressos na forma de gráficos, que repetiram com grande precisão os valores da literatura.

**PALAVRAS-CHAVE:** Vidros. Java. Cristalização. Cinética de Sinterização. Modelo de Clusters.

GRANDO, L. **Developing a Java software to compute the glass sintering kinetics.** 2009. 91 f. Monograph (Completion of Material Engineering Course) - Faculdade de Engenharia do Campus de Guaratinguetá, Universidade Estadual Paulista, Guaratinguetá, 2009.

## ABSTRACT

The use of computational tools for the simulation of physical phenomena is increasing, saving time and money in new processes and materials development. This study aims to use the Java tool for developing a program to compute the kinetics of sintering of vitreous material for ceramics manufacturing. Through this process, special glasses are grinded, compacted and sintered, with simultaneous or subsequent particles crystallization. A recent solution for the sintering process of glass by viscous flow, called Cluster Model, considers the effect of surface crystallization of the particles, which limits the densification (pore elimination) of the material, and allows to deal with mixtures of particles of different sizes. We developed a program in JAVA based on the Clusters Model, for calculations of the sintering kinetics of compact particles of glass powder with different particle sizes and concurrent crystallization. Published data for glasses in the systems $Al_2O_3$-$B_2O_3$-$SiO_2$ and CaO-MgO-$SiO_2$ were used to check the results of the program, expressed as graphs, which repeated with great precision the literature values.

**KEYWORDS**: Glass, Java, Crystallization, Sintering Kinetics, Clusters Model.

# LISTA DE FIGURAS

# LISTA DE TABELAS

# LISTA DE ABREVIATURAS E SIGLAS

| | | |
|---|---|---|
| JMAK | - | Johnson-Mehl, Avrami e Kolmogorov |
| UML | - | Unified modeling Language |
| MS | - | Mackenzie-Shuttleworth |
| F | - | Frenkel |

# LISTA DE SÍMBOLOS

| Símbolo | Significado | Unidades |
|---|---|---|
| Tg | Temperatura de transição vítrea | K |
| Lo | Comprimento original de um conglomerado de partículas | M |
| $\Delta L$ | Encolhimento após um tempo de sinterização t | M |
| $\eta(T)$ | Viscosidade, que é uma função da temperatura T | Pa.s |
| $\Gamma$ | Energia superficial vapor – vidro | $J/m^2$ |
| R | Raio da partícula | M |
| $\rho(t)$ | Densidade relativa em função do tempo | |
| $\rho_o$ | Densidade relativa à verde | |
| $a_0$ | Raio inicial dos poros | M |
| Vr | Fração em volume de partículas de um determinado raio | % |
| T | Tempo de sinterização | S |
| T | Temperatura de sinterização | K |
| $\theta(t)$ | Função degrau | |
| $t_{08}$ | Tempo em que a densidade relativa é igual a 0,8 | S |
| $\eta_\infty$ | Viscosidade a uma temperatura infinita | Pa.s |
| $E_V$ | Energia de ativação associada ao transporte molecular por fluxo viscoso | J |
| R | Constante dos gases | J/mol.K |
| T0 | Constante empírica | K |
| Ns | Numero de sítios por unidade de área | $m^{-2}$ |
| U(T) | Taxa linear de crescimento de grãos | m/s |
| $\alpha_s$ | Fração cristalizada da superfície das partículas | % |
| $\rho_F(t)$ | Densidade relativa calculada pelo mecanismo de Frenkel | |

| | |
|---|---|
| $\rho_{MS}(t)$ | Densidade relativa calculada pelo mecanismo de Mackenzie-Shutteworth. |
| $\rho_c(t)$ | Densidade relativa calculada considerando o efeito de cristalização superficial. |
| $\rho_{C,F}(t)$ | Densidade relativa calculada pelo mecanismo de Frenkel considerando o efeito de cristalização superficial. |
| $\rho_{C,MS}(t)$ | Densidade relativa calculada pelo mecanismo de Mackenzie-Shutteworth considerando o efeito de cristalização superficial |
| K | Constante calculada para encontrar valor de $a_0$. |

# SUMÁRIO

# 1. INTRODUÇÃO

Vidro é um material com características de um sólido, embora não-cristalino (amorfo), ou seja, que apresenta somente ordenação atômica de curto alcance e que apresenta o fenômeno da transição vítrea [CALLISTER, 2002]. A transição vítrea corresponde à transição, no aquecimento, do comportamento do material, de um sólido elástico para um líquido viscoso (e vice-versa, no resfriamento) [RAO, 2002]. O estudo da sinterização de materiais vítreos particulados é de grande importância para conhecer e assim permitir a fabricação de materiais vitrocerâmicos por essa rota.

Vitrocerâmicas são materiais inorgânicos policristalinos que podem conter fase vítrea residual e são produzidos a partir da cristalização controlada de um vidro. Tais materiais podem ser fabricados totalmente livres de poros e o processo permite projetar as microestruturas e as propriedades desejadas [SOARES, 2007]. O uso abrange desde usos domésticos até materiais utilizados em engenharia, tais como em microeletrônica, biomateriais, equipamentos científicos, vitrocerâmicas usináveis, etc.

Vitrocerâmicas podem ser fabricadas pela sinterização por fluxo viscoso de partículas vítreas, que ocorre com cristalização superficial concorrente e limitante do processo. [SOARES, 2007] Assim, a cristalização superficial das partículas deve ser controlada para permitir a sinterização e a densificação até o nível desejado, antes que o processo seja barrado. Quando controlada a cristalização, a sinterização de vidros permite a obtenção de vitrocerâmicas por uma rota alternativa, que dispensa o uso de agentes nucleantes [PRADO; ZANOTO; MULLER, 2001], já que a superfície das partículas é local preferencial para a nucleação de cristais e ocorre sempre (mesmo quando a nucleação no volume é improvável). A sinterização de materiais vítreos segue aproximadamente o Modelo de Clusters [PRADO; ZANOTO; MULLER, 2001], que utiliza dois mecanismos distintos de sinterização, de Frenkel ($F$) e de

Mackenzie-Shuttleworth (*MS*), para abranger todo o intervalo de ocorrência do fenômeno.

Duas abordagens são possíveis para o fenômeno da sinterização de vidros, considerando-se ou não o efeito da cristalização superficial na cinética de sinterização das partículas vítreas. Métodos de cálculos numéricos são utilizados para calcular computacionalmente a densidade dos compactos sinterizados.

A partir do modelo de Clusters, implementamos um programa em JAVA para os cálculos correspondentes, onde os dados de entrada são as variáveis dependentes do processo e da composição que afetam a cinética do fenômeno. As variáveis dependentes da composição do vidro são: a viscosidade $\eta(T)$, a tensão superficial $\gamma$ e a taxa de crescimento de cristais $U(T)$. As variáveis dependentes do processo são a distribuição de tamanhos de partículas $r$, a densidade a verde $\rho_o$ do compacto, a densidade de núcleos na superfície das partículas, $N_s$ (onde cristais irão se desenvolver), o tempo $t$ e a temperatura $T$ de sinterização em condições isotérmicas, além de fatores geométricos que dependem do formato e empacotamento das partículas. A resposta do software é a densidade relativa do compacto em função do tempo, nas formas de tabela e gráfico.

A linguagem de programação JAVA é orientada a objeto, robusta e de grande portabilidade (pode ser executada em qualquer sistema operacional, Windows, Linux, etc.) [NETO, 2004]. Essas características foram responsáveis pela escolha da linguagem.

A partir do entendimento do fenômeno de sinterização de vidros com cristalização concorrente, podemos planejar a fabricação de materiais vitrocerâmicos para diversas aplicações, utilizando, como matéria-prima, materiais vítreos com diferentes composições.

O objetivo desse trabalho consistiu em desenvolver um software para o cálculo da densidade de compactos de partículas de vidro submetidos a tratamento de sinterização em temperatura constante, durante o qual ocorre também o fenômeno da cristalização superficial concorrente das partículas.

# 2. DESENVOLVIMENTO

## *2.1 REVISÃO BIBLIOGRÁFICA*

### 2.1.1 Vidros e curva de transição vítrea

O fenômeno da transição vítrea pode ser entendido através da curva do volume específico em função da temperatura dos materiais (Figura 1), a partir da qual também pode ser entendido como ocorre a formação dos materiais vítreos a partir do estado líquido. [PRADO; ZANOTO; MULLER, 2001]

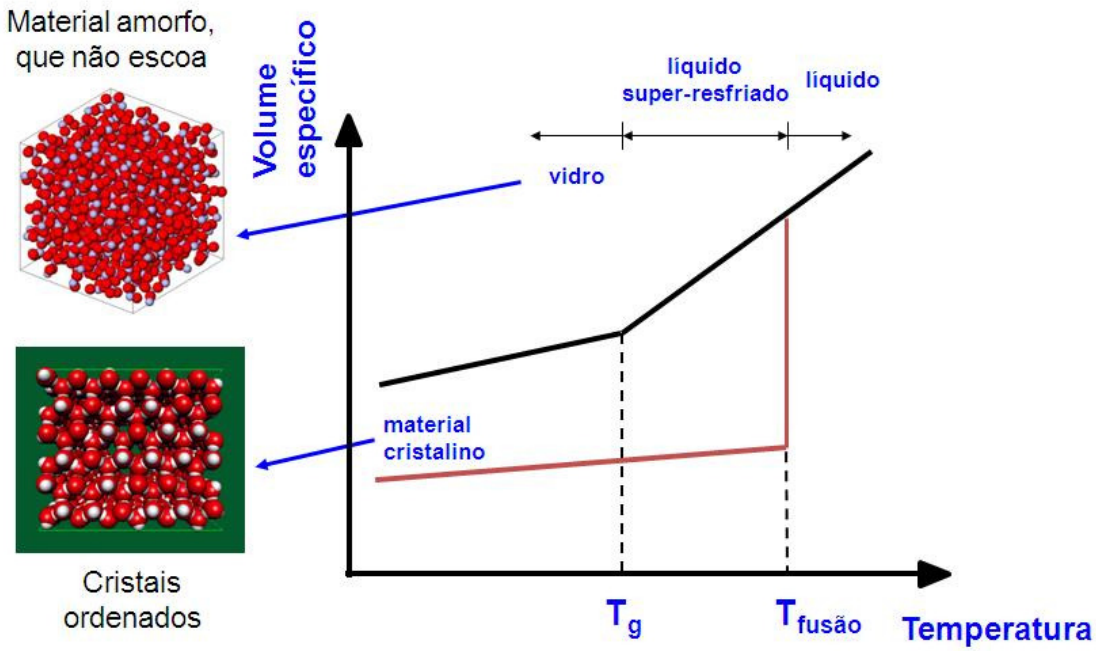



Figura 1 - Representação, pela variação do volume específico com a temperatura, da transição vítrea e dos estados da matéria em função da história térmica.

Quando um material está no estado líquido, em altas temperaturas, seus átomos ou moléculas possuem liberdade de movimentação. Conforme ocorre o resfriamento, o material pode seguir por dois caminhos diferentes, que determinam sua microestrutura e consequentes propriedades, dependendo da taxa de resfriamento. Quando o material é resfriado no equilíbrio, a partir da temperatura de fusão o mesmo adquire uma estrutura organizada, cristalina, ou seja, seus átomos formam uma rede periódica uniforme. Caso não houver tempo suficiente para a cristalização, por outro lado, o material se fixará no estado

vítreo, onde seus átomos ou moléculas estarão dispostos de forma não ordenada, com características estruturais típicas de um líquido, embora não sejam mais observadas movimentações atômicas, a não ser por difusão. [SOARES, 2007]

Assim, para se formar no estado cristalino, partindo do estado líquido, o material é resfriado até uma temperatura onde ocorre a solidificação ($T_f$) e uma contração abrupta de volume pode ser verificada pela linha vertical do gráfico da Figura 1. No estado cristalino, uma contração branda continua com o resfriamento, correspondendo à diminuição da vibração térmica dos átomos no material (contração térmica) em uma estrutura ordenada.

Para a formação do estado vítreo, o caminho de resfriamento do material é diferente e pode ser explicado também partindo do seu estado líquido, mas em uma taxa rápida o suficiente para impedir a organização atômica de longo alcance. No resfriamento rápido, o material entra inicialmente em um estado de líquido super-resfriado, após a temperatura ultrapassar a temperatura de fusão ($T_f$). Segue-se com um resfriamento relativamente rápido até a transição vítrea, que pode ser definida como a passagem de um material líquido super-resfriado, que pode escoar no estado viscoso, para um estado puramente elástico, típico de sólidos, sem ocorrência de cristalização [RAO, 2002]. Assim, o estado vítreo corresponde ao estado de um material que se comporta como sólido, mas com estrutura amorfa e desordenada. A transição vítrea pode ser caracterizada por uma temperatura convencional, $T_g$, obtida na intersecção do prolongamento das curvas do gráfico da Figura 1 nas regiões de líquido super-resfriado e após a formação do vidro – tal temperatura é chamada de *temperatura de transição vítrea*.

### 2.1.2 Vitrocerâmicas sinterizadas

Vitrocerâmicas são cerâmicas obtidas a partir da cristalização controlada de materiais vítreos. Dois possíveis caminhos podem ser usados para a obtenção desse tipo de material cerâmico: a) por cristalização no volume de monólitos de vidro; e b) por cristalização na superfície de partículas vítreas, em compactos que se sinterizam simultaneamente por fluxo viscoso. No primeiro caso, em geral, há

necessidade de se adicionar agentes catalisadores para a formação de cristais no volume do material, na etapa de formulação do vidro. No segundo caso, no entanto, utilizado no presente trabalho, a fabricação de vitrocerâmicas ocorre por sinterização e cristalização de compactos de partículas vítreas, que não necessita de agentes nucleantes [SOARES, 2007] e [PRADO; ZANOTTO, 2000], pois a própria superfície das partículas é propícia para a formação de cristais, que em seguida crescem em direção ao volume das mesmas. [PRADO; ZANOTO; MULLER, 2001]

O fluxograma (Figura 2) mostra como são obtidos materiais vitrocerâmicos a partir de um material vítreo, por sinterização e cristalização de partículas em um compacto.

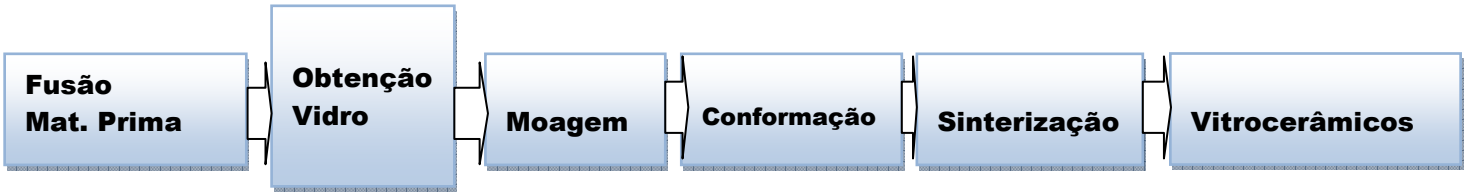



Figura 2 - Fluxograma do processo de fabricação de materiais vitrocerâmicos.

Materiais vitrocerâmicos são formados apenas se ocorrer a cristalização controlada simultaneamente ou após a sinterização.

No fluxograma da Figura 2 temos inicialmente uma etapa de fusão pelo aquecimento da matéria prima, que ao se resfriar rapidamente se transforma em um material vítreo. Esse material vítreo passa então por processos de moagem e conformação de acordo com o formato desejado do objeto vitrocerâmico. Finalmente ocorre novo aquecimento, para sinterização e cristalização controlada das partículas, resultando em um material vitrocerâmico. [SOARES, 2007]

### 2.1.3 Sinterização

O modelo utilizado no presente trabalho para descrever a densificação de materiais vítreos é encontrado na literatura e recebe o nome de Modelo de Clusters. Tal modelo utiliza os equacionamentos de Frenkel (F) e de Mackenzie-Shuttleworth (MS) para a sinterização por fluxo viscoso, para as etapas inicial e final do processo, respectivamente. [PRADO; ZANOTO; MULLER, 2001]

O modelo de Frenkel descreve bem os estágios iniciais de sinterização e densificação isotrópica por fluxo viscoso de partículas esféricas e monodispersas (de mesmos tamanhos), até uma densidade relativa de aproximadamente 0,8 ou uma retração linear de aproximadamente 10% [PRADO; ZANOTO; MULLER, 2001]. A densidade relativa é a razão entre a densidade do conglomerado e a densidade do material sem poros, e pode ser então entendida como uma fração da densidade teórica do material sem poros. A equação que descreve a retração no estágio de Frenkel em relação ao tempo de sinterização ($t$) é a seguinte:

$$\frac{\Delta L}{L_0} = \frac{3\gamma}{8\eta(T)r}t \qquad (1)$$

Onde:

$L_o$ = comprimento original de um conglomerado de partículas

$\Delta L$ = variação de comprimento após um tempo de sinterização $t$

$\eta(T)$ = viscosidade, que é uma função da temperatura $T$

$\gamma$ = energia superficial vapor–vidro

$r$ = raio das partículas.

A Figura 3 mostra esquematicamente o modelo físico considerado para o desenvolvimento da Equação 1, onde o comprimento total das duas partículas sofre uma contração linear de $\Delta L$.

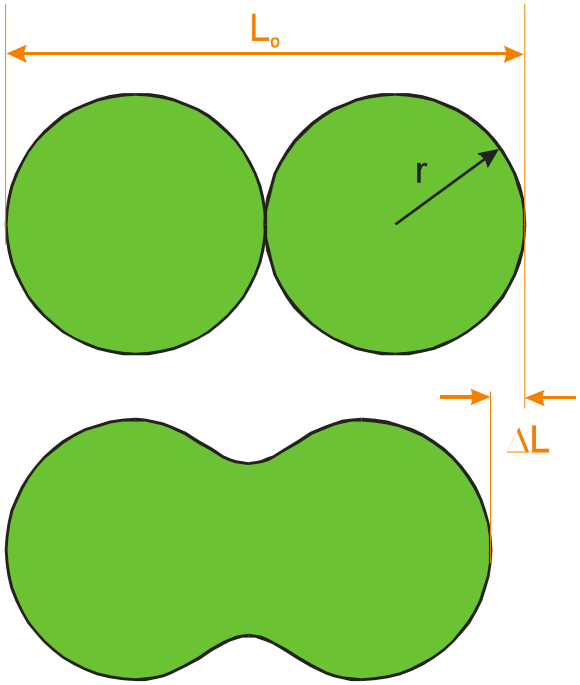



Figura 3 – Modelo de sinterização de partículas esféricas, no estágio inicial do processo.

Já a expressão que relaciona a densidade relativa em função do tempo de sinterização, $\rho_F(t)$, é descrita pela função abaixo, onde $\rho_o$ é a densidade relativa a verde

$$\frac{\rho_F(t)}{\rho_0} = \left(1 - \frac{3\gamma t}{8\eta(T)r}\right)^{-3} \qquad (2)$$

ou

$$\rho_F(t) = \rho_0\left(1 - \frac{3\gamma t}{8\eta(T)r}\right)^{-3} \qquad (3)$$

Para densidades relativas elevadas, ou seja, no estágio final de sinterização, aproximado por uma situação onde poros esféricos isolados e de mesmo tamanho retraem em uma matriz vítrea contínua, o modelo que melhor descreve a densificação do material devido ao fluxo viscoso é a expressão de Mackenzie-Shuttleworth (Equação 4):

$$\frac{d\rho_{MS}(t)}{dt} = \frac{3\gamma}{2a_0\eta(T)}(1 - \rho_{MS}(t)) \qquad (4)$$

Onde $a_0$ é o raio inicial dos poros, considerado constante.

A solução dessa equação diferencial resulta na Equação 5:

$$\rho_{MS}(t) = 1 - \exp\left(\frac{-3\gamma t}{2a_0\eta(T)} + \ln(1-\rho_0)\right) \qquad (5)$$

Que é a equação que descreve a cinética de sinterização de Mackenzie Shutterworth.

A viscosidade $\eta(T)$ nesses casos é calculada pela equação de Vogel-Fulcher-Tamman (VFT) (Equação 6). Para que ocorra fluxo viscoso nos vidros, a temperatura de sinterização deve estar acima da $T_g$, a temperatura de transição vítrea [PRADO; ZANOTO; MULLER, 2001]. Por essa equação observa-se que a viscosidade cai em função do aumento de temperatura e os valores de $\eta_\infty$, $E_V$ e $T_o$ variam de acordo com o material utilizado.

$$\eta(T) = \eta_\infty e^{\frac{Ev}{R(T-T_0)}} \qquad (6)$$

Onde:

$\eta$ = viscosidade

$\eta_\infty$ = viscosidade a uma temperatura infinita

$E_V$ = energia de ativação associada ao transporte molecular por fluxo viscoso

$R$ = constante dos gases

$T$ = temperatura que se deseja calcular a viscosidade

$T_o$ = constante empírica.

A figura 4 mostra esquematicamente o principio básico do modelo de Mackenzie-Shuttleworth, onde se percebem poros isolados, de raio $a_0$, em uma matriz vítrea contínua.

O modelo de Clusters tem como princípio a aproximação de que partículas de mesmo tamanho se agrupam em clusters (Figura 5) e sinterizam-se somente entre si, com as partículas pequenas localizadas nos espaços existentes entre as partículas maiores. As partículas menores sinterizam-se em tais clusters mais rapidamente que as partículas maiores. [PRADO; ZANOTTO, 2000]

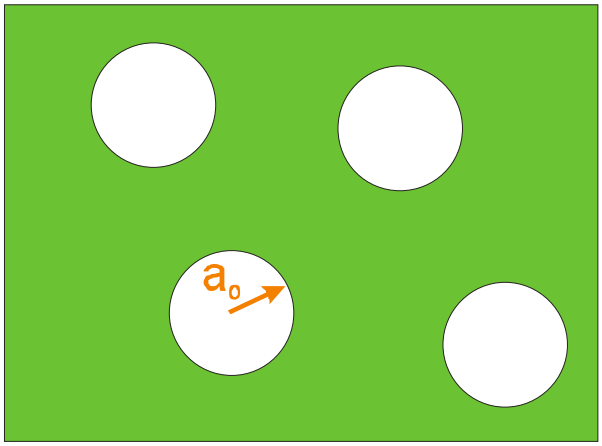



Figura 4 – Matriz vítrea continua com poros de raio $a_0$.

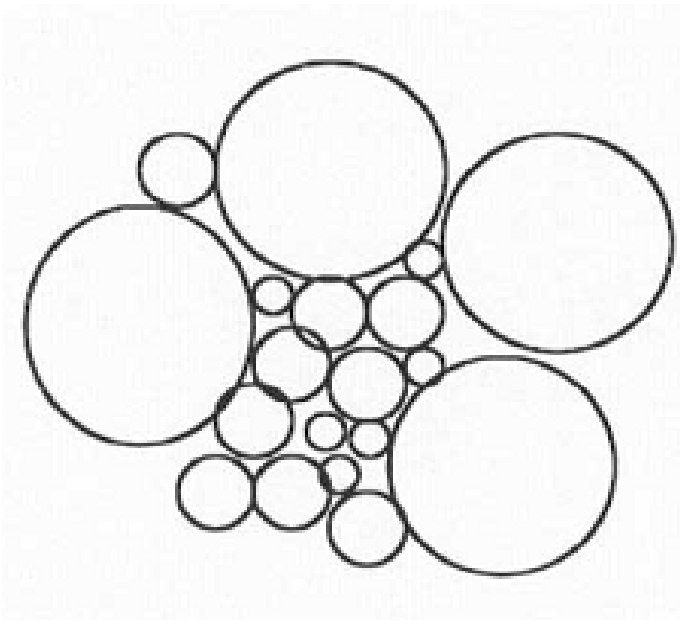


Figura 5 - Representação de compacto de partículas segundo o Modelo de Clusters.

A Figura 6 abaixo mostra a formação de pescoços nos contatos entre as partículas vítreas sinterizadas, as pequenas localizadas nos espaços entre as partículas maiores, e partículas de tamanhos similares sinterizando-se entre si. No entanto, também se observa formação de pescoços entre partículas de diferentes tamanhos.

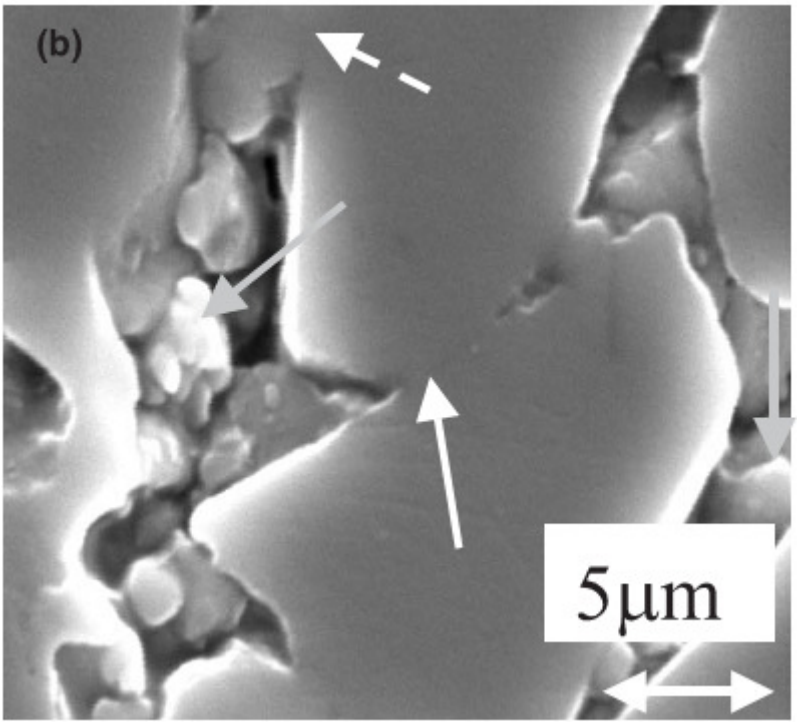



Figura 6 - Micrografia de um compacto de partículas de tamanhos polidispersos, após um encolhimento linear de 8%. Flecha cinza: sinterização entre partículas pequenas, flecha branca: entre partículas grandes, e flechas tracejada: entre partículas grandes e pequenas. [PRADO; ZANOTTO, 2000]

O comportamento da sinterização de vidros pode ser resumido na expressão a seguir [PRADO; ZANOTO; MULLER, 2001], de origem empírica, onde $V_r$ é a fração em volume de partículas de raio $r$:

$$\rho(t) = \sum_r \left[ \left( \rho_F(r,t) \cdot \theta(t_{0,8} - t) + \rho_{MS}(r,t) \cdot \theta(t - t_{0,8}) \right) \cdot V_r \right] \quad (7)$$

A função $\theta(t)$ (degrau) assume os seguintes valores:

$\theta(t) = 1$ se $t > 0$ e (8)
$\theta(t) = 0$ se $t \leq 0$

A função $\theta(t)$ (Equação 8) assume valores binários em relação ao tempo de sinterização ($t$), fazendo com que até uma densidade relativa de 0,8 prevaleça o mecanismo de Frenkel e a partir dessa densidade o mecanismo de Mackenzie–Shuttleworth atua, para um dado tamanho de partículas na distribuição. [PRADO; ZANOTO; MULLER, 2001]

A Figura 7 ilustra o comportamento desses dois mecanismos de forma individual.

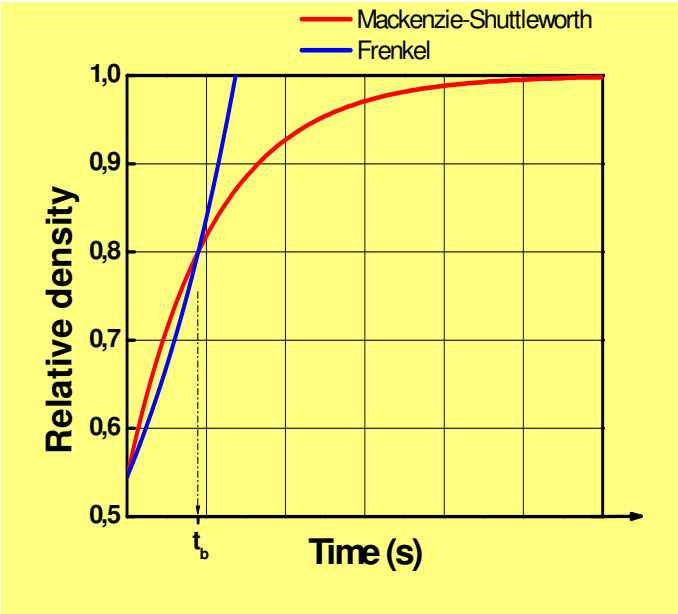



Figura 7 – representação dos mecanismo de sinterização de forma separada. ($t_b = t_{08}$)

O modelo de Cluster, diferente dos mecanismos de Frenkel e de Mackenzie-Shuttleworth, pode ser utilizado para o cálculo da cinética de sinterização de uma distribuição de tamanhos de partículas e em todo o intervalo de densidades, de $\rho_o$ até 100% da densidade teórica.

A Figura 8 mostra uma curva típica resultante de cálculos com o mecanismo de Clusters para um único tamanho de partícula. Até a densidade relativa de 0,8 o modelo de Frenkel é atuante e após MS domina.

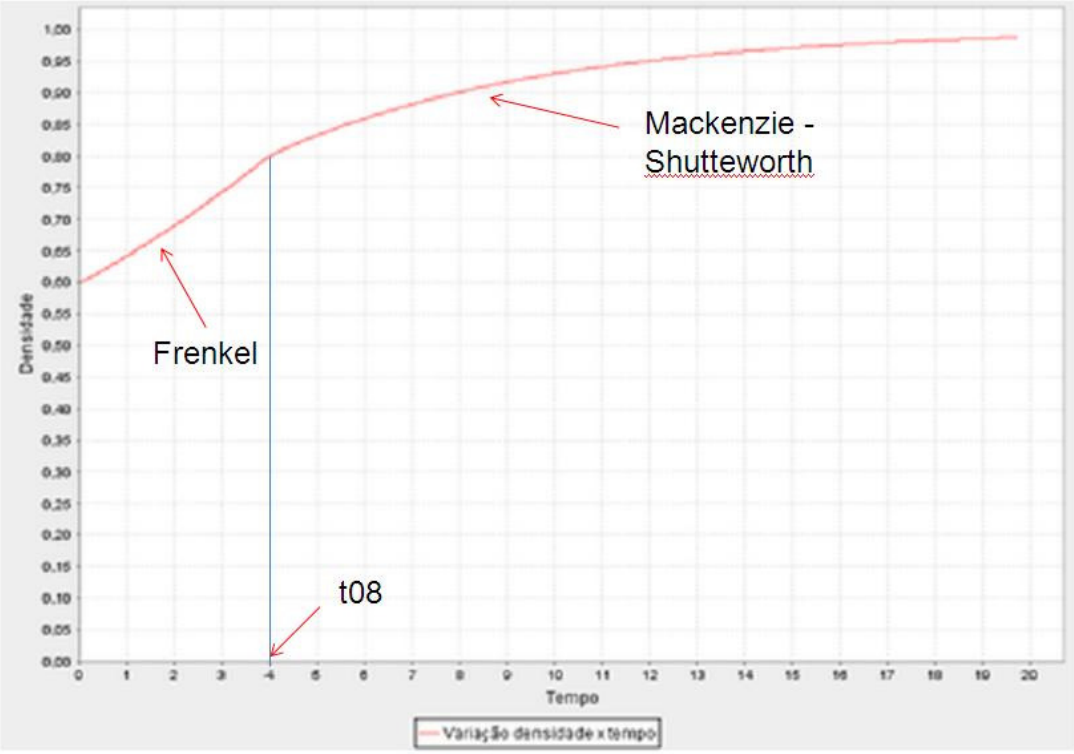



Figura 8 - Gráfico obtido pelo software desenvolvido por esse trabalho, simulando a sinterização do complexo vítreo a 1000 °C (densidade relativa em função do tempo, s).

### 2.1.4 Modelo de Clusters sem cristalização concorrente

Nesse caso não é considerado o fenômeno da cristalização concorrente na superfície das partículas, ocorrendo apenas à sinterização por fluxo viscoso.

**Cálculo de raio inicial de partículas [$a_0$]**

Para cada tamanho inicial de partícula, quando ocorre a passagem do estágio de Frenkel para o estágio de Mackenzie-Shuttleworth, o compacto parcialmente sinterizado apresenta um determinado tamanho inicial de poros, dependente do raio inicial das partículas, que irão posteriormente retrair. Há assim necessidade de se estimar o tamanho inicial dos poros para os cálculos no estágio de MS.

A estimativa de $a_0$ pode ser feita a partir do modelo de Mackenzie–Shuttleworth (MS) [PRADO; ZANOTO; MULLER, 2001]. Após manipulação e

integração da expressão (4) para a densidade $\rho(t)$ de $\rho_0$ até $\rho_1$ e tempo $t$ de $t_0$ a $t_1$, tem-se:

$$\int_{\rho_0}^{\rho_1} \frac{d\rho_{MS}(t)}{1-\rho_{MS}(t)} = \int_{t_0}^{t_1} \frac{3\gamma}{2a_0\eta(T)} dt \qquad (9)$$

Integrando e manipulando a Equação (9) chega-se a:

$$-\ln\left(1-\rho_{MS}(t)\right)\Big|_{\rho_0}^{\rho_1} = \frac{3\gamma}{2a_o\eta(T)} t\Big|_{t_0}^{t_1} \qquad (10)$$

$$\ln\left(1-\rho_0\right) - \ln\left(1-\rho_1\right) = \frac{3\gamma}{2a_o\eta(T)}\left(t_1 - t_0\right) \qquad (11)$$

Para se chegar a uma expressão para o valor de $a_0$, pode-se usar as seguintes condições de contorno: $\rho_0 = \rho_0$ , $\rho_1 = 0{,}8$ , $t_0 = 0$ e $t_1 = t_{0,8}$ na Equação (11), obtendo-se a Equação (12).

$$\ln\left(1-\rho_0\right) - \ln\left(1-0{,}8\right) = \frac{3\gamma}{2a_o\eta(T)}\left(t_{0,8} - 0\right) \qquad (12)$$

Finalmente, obtém-se a Equação (13) para o cálculo do valor de $a_0$ utilizada no presente trabalho.

$$a_0 = \left( \frac{3\gamma}{2\eta(T)} t_{0.8} \frac{1}{\left(\ln(1-\rho_0) - \ln(0{,}2)\right)} \right) \qquad (13)$$

**Cálculo de $t_{08}$**

O tempo $t_{08}$ é definido como o tempo onde a densidade relativa do compacto atinge o valor de 0,8. O cálculo desse tempo é fundamental para determinar quando a sinterização entra no estágio MS, após o estágio F. Para encontrar seu valor, a partir da expressão de Frenkel, Equação (3), quando $\rho_F(t)$ é igual a 0,8, o

tempo $t$ é $t_{08}$, e ocorre a mudança do mecanismo de F para MS para um dado tamanho de partículas $r$:

$$0{,}8 = \rho_0 \left( 1 - \frac{3\gamma t_{08}}{8\eta(T) r} \right)^{-3} \qquad (15)$$

Manipulando a Equação (15), deixando $t_{08}$ em função de $r$ :

$$1 - \left( \frac{\rho_0}{0{,}8} \right)^{\frac{1}{3}} = \frac{3\gamma t_{08}}{8\eta(T) r} \qquad (16)$$

Finalmente chega-se à Equação (17), que representa a equação utilizada no calculo do valor de $t_{08}$ no presente trabalho, no caso em que a cristalização superficial não ocorre significativamente:

$$t_{08}(r) = \frac{8}{3} \left( \frac{(0{,}8^{\frac{1}{3}} - \rho_0^{\frac{1}{3}})}{0{,}8^{\frac{1}{3}}} \right) \frac{\eta(T)}{\gamma} r \qquad (17)$$

### 2.1.5 Modelo de Clusters com cristalização concorrente

O modelo descrito a seguir considera a cristalização que pode ocorrer na superfície de partículas de vidro quando são efetuados tratamentos térmicos para sinterização. O resultado dessa cristalização é a possibilidade de uma densidade relativa inferior a 1 (ou 100% da densidade teórica) quando a cristalização superficial for significativa, já que a cristalização da superfície das partículas não permite a total sinterização, impedindo a total eliminação de poros do interior do compacto. O termo concorrente decorre do fato da cristalização ocorrer simultaneamente à sinterização, restringindo-a [PRADO; ZANOTTO, 2000].

A Figura 9 mostra como ocorre a cristalização na superfície de uma partícula em um compacto durante a sinterização, impedindo sua densificação.

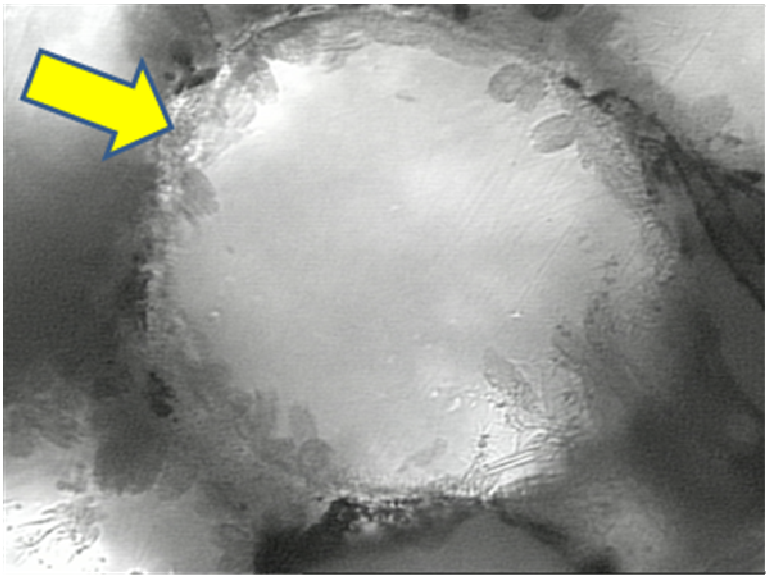


Figura 9 – Cristalização superficial de uma partícula, que limita a densidade final do material, apontada pela flecha.

Mas a cristalização pode ser benéfica, no caso da fabricação de vitrocerâmicas pela rota de sinterização. Assim, o conhecimento de um modelo de sinterização adequado permite o domínio do processo para a fabricação de vitrocerâmicas por essa rota.

Partículas de vidros têm a tendência de se cristalizar pela superfície quando levadas a temperaturas que permitam sua sinterização. E qualquer fração da superfície das partículas que se cristalizar barra proporcionalmente a sinterização por fluxo viscoso, reduzindo a eliminação de poros e por isso a densificação do compacto. [PRADO; ZANOTTO, 2000]

É discutido abaixo o caso mais comum de cristalização de partículas de vidro, que ocorre por nucleação heterogênea de cristais na superfície das mesmas, a partir de um número fixo de sítios por unidade de área, $N_s$, e crescimento a uma taxa linear $U(T)$ em direção ao volume das mesmas. Nesse caso a teoria de JMAK (Johnson, Mehl, Avrami e Kolmogorov) prediz que a fração cristalizada da superfície das partículas, $\alpha_s(t)$, após um tempo $t$ de tratamento isotérmico a uma dada temperatura $T$, é dada pela Equação (18) [PRADO; ZANOTTO, 2000].

$$\alpha_s(t) = 1 - \exp\left(-\pi N_s U(T)^2 t^2\right) \qquad (18)$$

Segundo Müller [PRADO; ZANOTO; MULLER, 2001] e [PRADO; ZANOTTO, 2000], pode-se assumir que a taxa de densificação é progressivamente reduzida pelo processo de cristalização superficial, resultando em uma densidade relativa do compacto $\rho_C(t)$ onde o efeito da cristalização superficial é considerado da seguinte forma:

$$\frac{d\rho_C(t)}{dt} = \frac{d\rho(t)}{dt}\left(1 - \alpha_S(t)\right) \qquad (19)$$

Nesse caso, $\rho(t)$ é dado pelo modelo de F ou MS, dependendo do estágio de sinterização, Equações (3) ou (5), respectivamente, e $\alpha_s(t)$ é dado pela Equação (18). Substituindo-se essas expressões na Equação (19), para os casos de F e MS, podemos obter as densidades para ambos os estágios, considerando a cristalização superficial, da seguinte forma [PRADO; ZANOTTO, 2000]:

$$\rho_{C,F}(t) = \rho_0 + \frac{3C\rho_0}{\eta(T)}\int_0^t \left(1 - \frac{C}{\eta(T)}t'\right)^{-4} \exp\left(-\pi N_s U(T)^2 t'^2\right) dt' \qquad (20)$$

$$\rho_{C,MS}(t) = \rho_0 + (1-\rho_0)\frac{C'}{\eta(T)}\int_0^t \exp\left(\frac{C't'}{\eta(T)}\right)\exp\left(-\pi N_s U(T)^2 t'^2\right) dt' \qquad (21)$$

Sendo:

$$C = \frac{3\gamma}{8r} \qquad \text{e} \qquad C' = \frac{3\gamma}{2a_0}$$

Onde:

$\rho_{C,F}(t)$ = densidade relativa calculada pelo mecanismo de F considerando a cristalização concorrente.

$\rho_{C,MS}(t)$= densidade relativa calculada pelo mecanismo de MS considerando a cristalização concorrente.

De posse dessas equações, podemos determinar a densidade relativa aplicando o modelo de Cluster, agora usando as Equações (20) e (21) na Equação (7), resultando na seguinte fórmula:

$$\rho(t) = \left[ \sum_r \left( \rho_{C,F}(r,t) \cdot \theta(t_{08} - t) + \rho_{C,MS}(r,t) \cdot \theta(t - t_{08}) \right) \cdot V_r \right] \quad (22)$$

Nesse caso, para o cálculo da viscosidade, foi utilizada a seguinte expressão (similar à Eq.6):

$$\eta(T) = A \exp\left( \frac{B}{T - T_0} \right) \quad (23)$$

Onde $A$, $B$ e $T_o$ são constantes para cada tipo de vidro.

## *2.2 DESENVOLVIMENTO COMPUTACIONAL*

Neste capítulo apresentaremos como o programa foi desenvolvido no aspecto computacional.

### 2.2.1 Introdução

Cada vez mais são geradas ferramentas computacionais por causa da grande quantidade de cálculos envolvidos em problemas de pesquisa e desenvolvimento, ou mesmo de rotina. O presente projeto não é diferente, a matemática apresentada pelo modelo de Clusters é agilizada quando se emprega a computação. Para isso, utilizamos a linguagem Java.

A utilização da ferramenta Java como método computacional tem fundamento em suas características intrínsecas, tais como a sua portabilidade e a existência de pacotes disponíveis para a implementação. A portabilidade está relacionada com a forma de execução do Java. Ele é interpretado, o que faz com que sua utilização seja possível em qualquer ambiente computacional, bastando para isso instalar o Java Runtime Environment [NETO, 2004], apesar de isso deixar o tempo de execução do software mais lento.

O Java é uma linguagem orientada a objeto, o que permite a re-utilização de pacotes (códigos já implementados) para a criação de novas implementações. Com a utilização da programação orientada a objeto é possível modular o software a fim de facilitar a manutenção no mesmo. [NETO, 2004]

Os cálculos de densidades de compactos de vidro sinterizados segundo o modelo de Clusters foram executados utilizando os conceitos de orientação a objeto que relacionam as seguintes classes (cujo conceito é definido em 2.2.3):

***Partículas*** – na classe partículas estão contidas as características que descrevem as partículas de vidro, como raio e fração volumétrica, e os métodos que estão relacionados com sua manipulação, os quais estão relacionados a atribuir valores para os raios das partículas e para a fração volumétrica. Esses métodos apresentam a característica de encapsulamento da programação orientada a objeto;

***LGB_Dados*** – nessa classe estão relacionados os métodos utilizados para a utilização dos atributos para o uso da classe *LGB_Cálculos*. São inseridos nessa classe os valores de tensão superficial, densidade do compacto de partículas de vidro a verde, valores de $T_0$, da taxa de crescimento linear de grãos, do número de sítios de cristais;

***LGB_Cálculos*** – nessa classe são feitos todos os cálculos relacionados com a densidade segundo o modelo de Clusters: $t_{08}$, $a_0$ e densidade;

***LGB_Gráfico*** – esta classe é uma interface entre as classes *LGB_Cálculos* e *JFreeChart* (pacote utilizado na implementação de gráficos), utilizada para a apresentação de um gráfico com os resultados finais.

### 2.2.2 Pacote JFreeChart

O pacote *JFreeChart* foi utilizado para a elaboração de gráficos que representam de forma elaborada a cinética de sinterização, ou seja a densidade do compacto em função do tempo para dada temperatura. Sua utilização exemplifica as vantagens do Java como ferramenta computacional, pois sua utilização é fácil, além de permitir os mais variados gráficos. [JFREECHART, 2008]

### 2.2.3 UML – Unified Modeling Language

A UML é uma linguagem padrão para a especificação e elaboração de projetos de software. Pode ser empregada para visualização, especificação, construção e documentação de artefatos que façam uso de sistemas complexos de software. A versão atual da UML é a 3.0. [MATTOS, 2007]

A UML não é uma linguagem de programação, mas sim uma representação precisa e gráfica de como o sistema deve ser implementado. A visualização de toda a estrutura do software e as relações entre classes e objetos pode ser realizada por meio de um conjunto de diferentes representações gráficas que capturam diferentes características do sistema de software. Essas representações gráficas auxiliam na compreensão dos detalhes do projeto. [MATTOS, 2007]

Atender às especificações na elaboração dos modelos é importante para satisfazer as expectativas de qualquer cliente. Quando as especificações do problema estão bem definidas e são atendidas na análise e no projeto, obtém-se um modelo preciso da aplicação em desenvolvimento, o que colabora na implantação robusta de sistemas complexos. [MATTOS, 2007]

Quando se utiliza a UML, é possível obter ao final do projeto uma série de produtos, tais como, documentação de requisitos, arquitetura, projeto, código fonte, planos do projeto, teste, protótipos e versões. O diagrama de caso de uso tem como objetivo detectar as funções de um sistema e seus atores.

Dentre os vários diagramas disponíveis na UML, serão descritos brevemente o de classes e o de seqüência, que foram utilizados neste projeto. [MATTOS, 2007]

A utilização do **Diagrama de Classes** é indispensável para expressar claramente quais as classes relevantes e as relações entre elas. Uma **classe** representa uma abstração de um objeto real ou abstrato no domínio do estudo. O ideal é que o nome de uma classe seja representativo do tipo de objeto que ela representa.

O modelo de uma classe é representado por um retângulo dividido em três partes: o retângulo superior determina o nome da classe; atributos são as características do objeto e são identificados no retângulo logo abaixo do nome da classe; métodos são ações dos objetos e são identificados no retângulo abaixo dos atributos. Dependendo da fase de desenvolvimento, a apresentação do espaço relativo aos atributos e métodos é opcional. [MATTOS, 2007]

A linguagem UML fornece alguns símbolos que interligam as classes, tais como os apresentados na Figura 10.

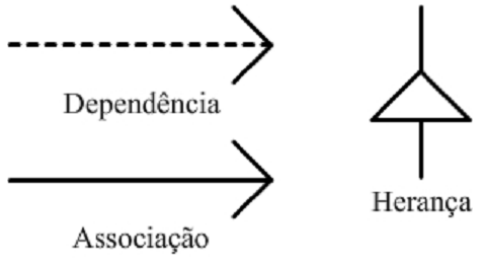



Figura 10 - Símbolos utilizados na UML para o diagrama de Classe

Quando se elabora um Diagrama de Classe fica muito mais fácil identificar os objetos e suas relações.

Outros diagramas de interesse neste trabalho são os de interação. Existem dois tipos de **Diagramas de Interação** na UML: o de **Seqüência** e o de **Colaboração**. Este tipo de diagrama mostra o comportamento dinâmico das classes.

Figura 11 apresenta um exemplo de **Diagrama de Sequência**. Toda interação tem início em um estímulo, representado pela mensagem identificada pelo índice 1. Diagramas de Seqüência exigem esse estímulo que é, na verdade, uma mensagem exigindo do objeto inicial a execução de uma operação. A barra vertical identificada com o número 2 indica o tempo de vida de uma operação.

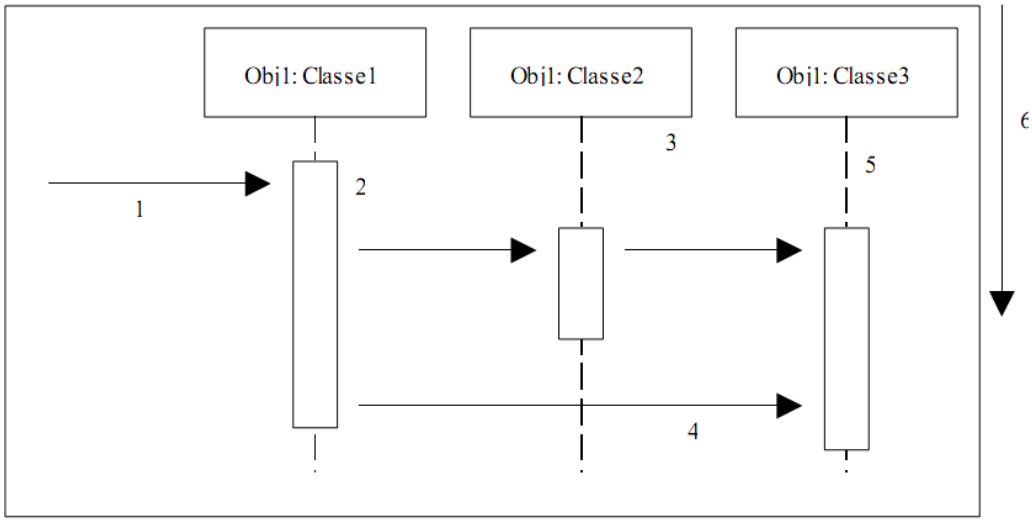



Figura 11 - Diagrama de seqüência [MATTOS, 2007].

Os retângulos apresentados na parte superior do diagrama (índice 3), e a partir dos quais se dá origem às linhas verticais de tempo, linhas de índice 5, representam objetos que tomam parte da ação representada pelo diagrama.

A seta horizontal que faz a ligação entre *Obj1:Classe1* e *Obj1:Classe3*, identificada pelo índice 4, mostra um estímulo síncrono. A representação do eixo y do gráfico é uma forma de mostrar que as mensagens devem ser lidas sequencialmente, respeitando a linha de tempo indicada, identificada pela seta vertical 6.

A seguir mostramos os diagramas de classes e o diagrama de seqüência para o presente caso.

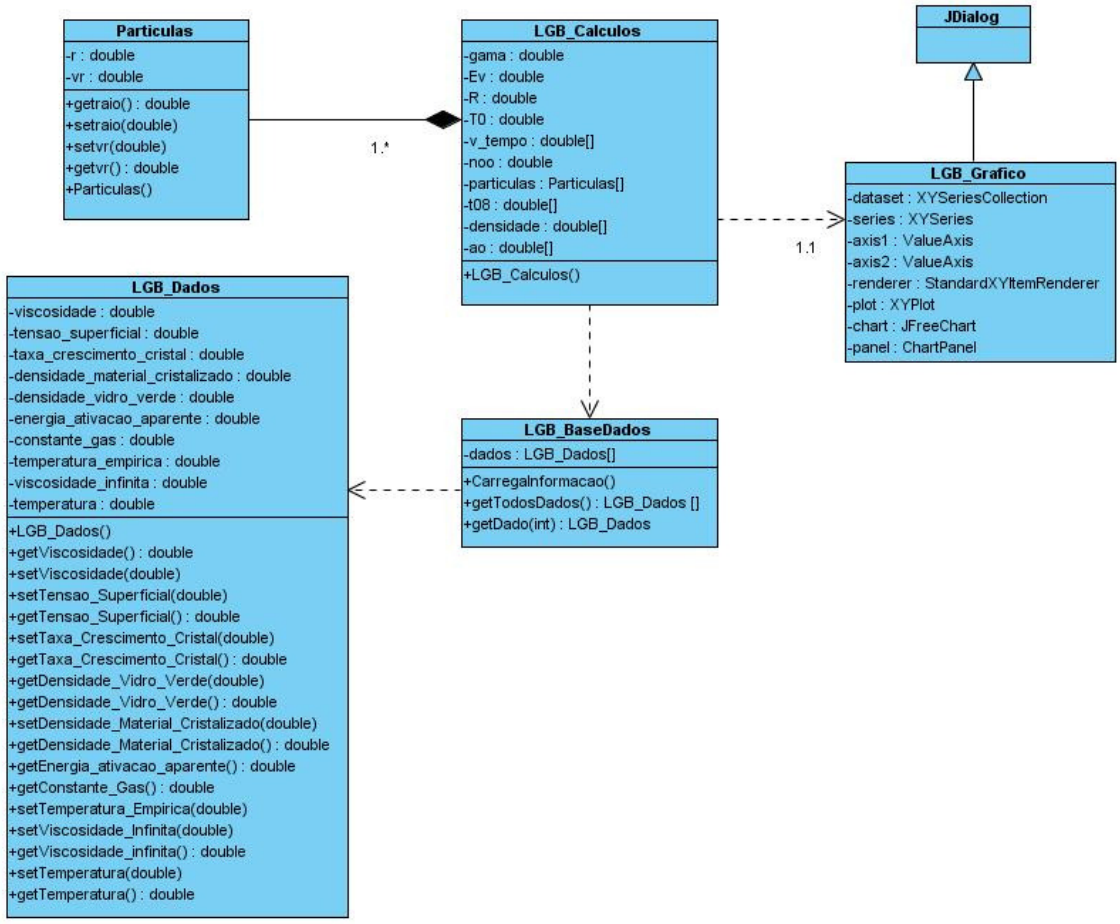



Figura 12 - Diagrama de Classe do presente trabalho.

No caso da relação entre as classes *Particulas* e *LGB_Calculos* ocorre o que se chama de composição ou seja Isto implica em se uma instância da classe deixar de existir, todas as outras associadas por composição, “deixarão” de existir também. O símbolo que representa a composição é um losango preenchido.

A classe *LGB_Gráficos* herda características da classe *JDialog* e a mesma tem dependência única (um único resultado para cada ponto do gráfico) da *LGB_Cálculos*.

A classe *LGB_Cálculos* faz todos os cálculos, a partir da dependência com a classe *LGB_Bases Dados*, que é a classe relacionada com a base de dados que devem ser fornecidos em forma de arquivo de texto. A classe *LGB_BasesDados* tem dependência com a classe *LGB_Dados*, sendo que essa classe define a utilização das constantes para os cálculos. A classe Partículas está relacionada

com um número definido de relações (esse número de relação está definido como o número de classes de partículas encontradas na distribuição de partículas, no caso de uma distribuição), com os *LGB_Cálculos* e essa classe contém os elementos (raio e distribuição volumétrica das partículas) para efetuar os cálculos.

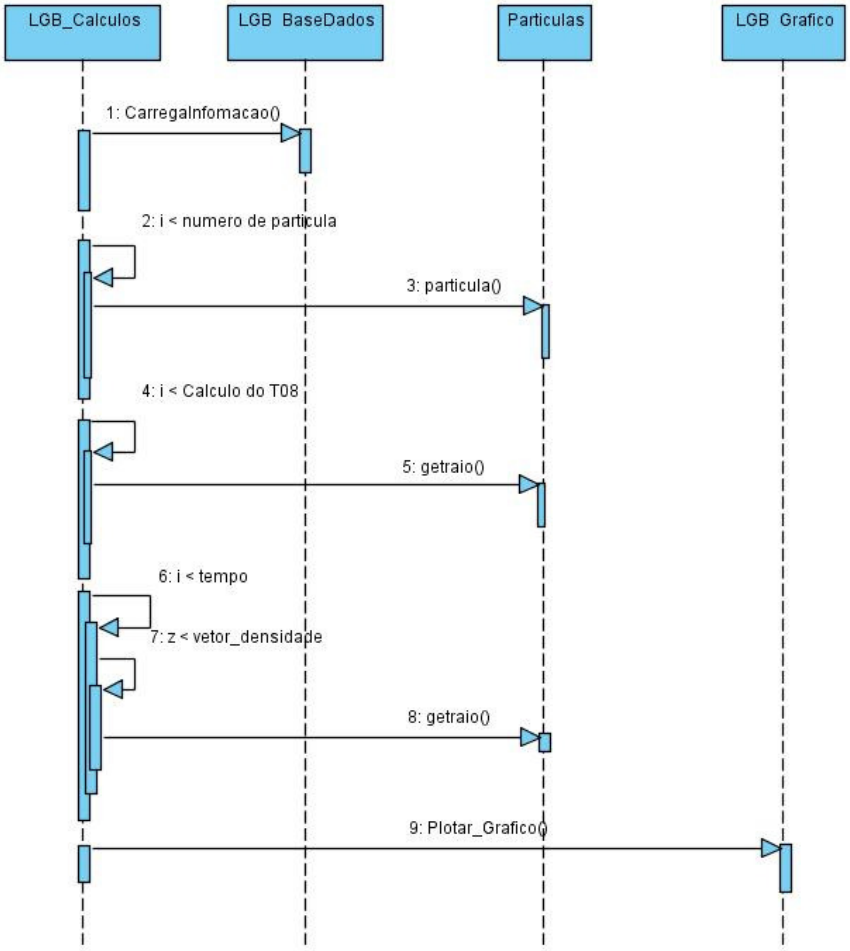



Figura 13 - Diagrama de Seqüência do presente trabalho.

O diagrama de Sequência nesse caso está relacionado em relação à base da classe *LGB_Cálculos* para iniciar as ações e efetuar os cálculos da densidade e a elaboração do gráfico. Cada método mostra às vezes que os mesmos são efetuados e em quais classes tais métodos são disparados.

# 3. METODOLOGIA

Descrição dos cálculos de cada variável, bem como os métodos numéricos utilizados.

## 3.1 Cálculos efetuados

O software recebe os seguintes valores do usuário para o cálculo da cinética de sinterização:

Dados relacionados com a natureza do material:

- A viscosidade para a temperatura de interesse.
- Tempo de cálculo da sinterização.
- Tensão superficial.

Dados relacionados com o tamanho das partículas:

- Número de classes de partículas.
- Valores de raios
- Frações volumétricas de partículas com diferentes raios na distribuição.

Dados relacionados com o fenômeno da cristalização concorrente (quando ocorrer):

- Número de sítios por área.
- Velocidade de crescimento de cristais na temperatura de tratamento.

De posse desses dados o software calcula:

- O valor do tempo em que a densidade relativa atinge o valor de 0,8 para cada valor de raio, de acordo com o modelo de Frenkel.
- O valor do raio inicial dos poros isolados no compacto de partículas para cada raio presente na distribuição de tamanhos.

- O valor da densidade pelo modelo de Clusters, para cada valor de tempo, em nosso caso a cada um segundo (1 s), até o valor final de tempo especificado para o cálculo da sinterização.

## *3.2 Metodologia dos cálculos.*

### 3.2.1 Método analítico (sem cristalização concorrente)

Para o cálculo da densidade considerando a não ocorrência de cristalização concorrente, apenas substituem-se no programa as fórmulas já descritas e efetua-se o cálculo segundo a Equação (7), não sendo necessária nenhuma aproximação numérica para a solução das equações e obtenção dos valores. Uma melhor discussão desse cálculo pode ser encontrada em [GRANDO; MATTOS, 2008].

### 3.2.2 Método com cálculo numérico (com cristalização concorrente)

Para o cálculo dos valores de $t_{08}$, $a_0$ e da densidade do compacto sinterizado propriamente dita, quando ocorre cristalização concorrente, recorre-se a métodos numéricos para a solução de raízes de funções e integrais, descritos no **Anexo 1** ao final desse trabalho, quando não se encontram soluções analíticas. O cálculo da densidade é efetuado pelo modelo de Clusters segundo a Equação (16).

De forma resumida a lógica do cálculo da densidade pode ser definida de acordo com a Figura 14.

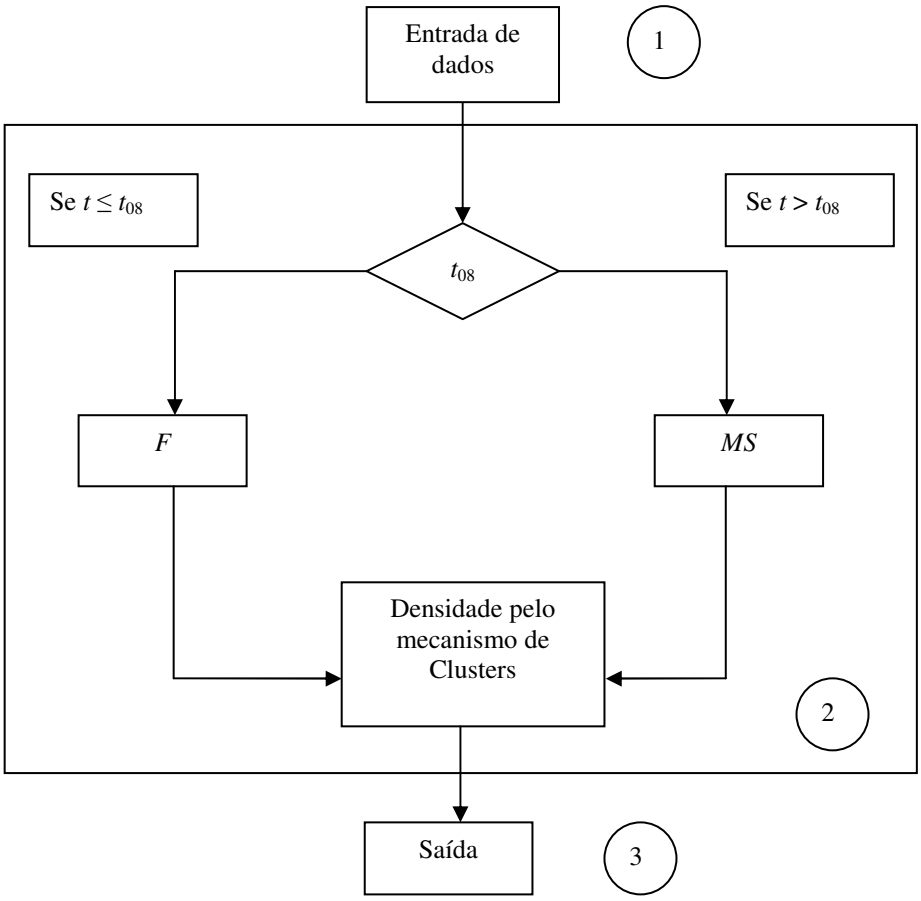



Figura 14 – Fluxograma do funcionamento do programa desenvolvido

Os valores calculados resultantes podem ser expressos em um gráfico e na forma de arquivo de texto, devido à capacidade da linguagem Java se comunicar com outros softwares.

# 4. RESULTADOS

O software desenvolvido foi testado para dois sistemas vítreos, um em que a cristalização da superfície é insignificante e o fluxo viscoso procede sem barreiras, e outro em que a cristalização é limitante da densificação por fluxo viscoso. Os resultados são dados abaixo.

## *4.1 Validação do software sem cristalização concorrente.*

Nessa primeira análise, usamos os dados apresentados em um artigo de Prado e colaboradores [PRADO; ZANOTO; MULLER, 2001] referentes a um vidro de composição (% em peso) 71,70 $SiO_2$, 8,33 $B_20_3$, 8,56 $Al_2O_3$, 1,00 MgO,

2,67 CaO e 7,44 $Na_2O$ para os cálculos com nosso programa e comparação dos resultados. Esse vidro apresenta baixa taxa de cristalização na temperatura de sinterização. Os valores considerados constantes e utilizados pelo software estão descritos na Tabela 1.

Tabela 1 – valores utilizados para o cálculo de densidade sem cristalização concorrente.

| **Constante** | **Valor (unidade)** |
|---|---|
| $\rho_o$ | 0,6 |
| $\gamma$ | 0,326 J/m$^2$ |
| $\eta_\infty$ | 0,018909 Pa.s ($e^{-3.9681} Pa.s$) |
| $E_V$ | 83.700 J/mol |
| $R$ | 8,31 J/mol.K |
| $T_0$ | 562,48 K |

A Figura 15 mostra a distribuição granulométrica de partículas utilizada por Prado *et al.* [PRADO; ZANOTO; MULLER, 2001]. A partir dessa figura foram obtidos valores de fração volumétrica de diferentes classes de diâmetros de partículas, listados na Tabela 2, resultando em 31 classes de tamanhos de partículas.

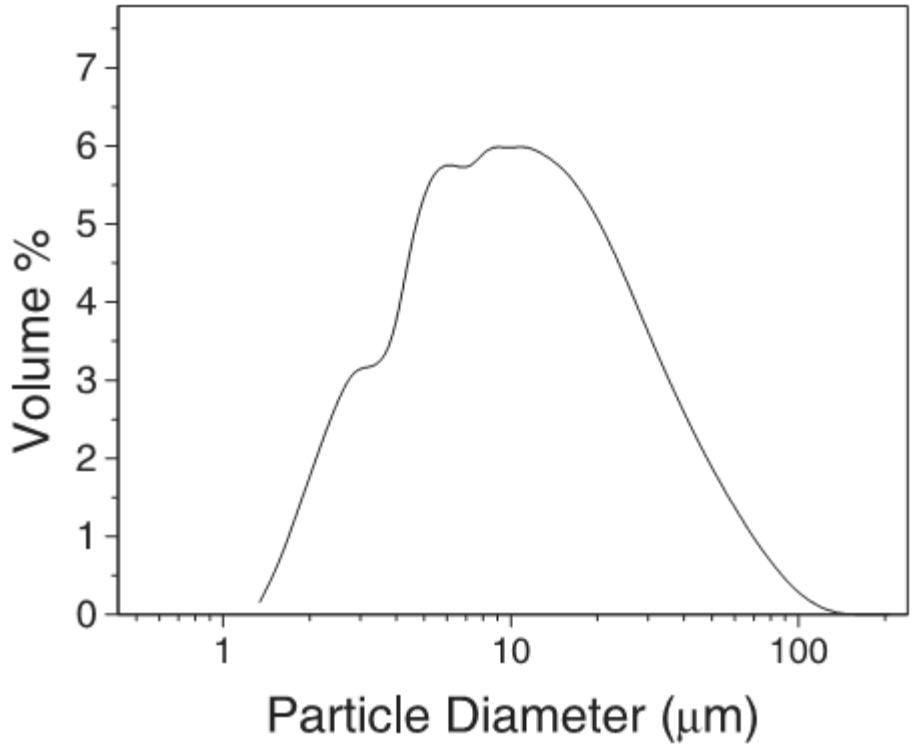



Figura 15 - Distribuição continua de partículas de materiais vítreos [PRADO; ZANOTO; MULLER, 2001].

Tabela 2 - Classe de partículas e suas respectivas porcentagens volumétricas.

| Raio (m) | Fração volumétrica (%) |
|---|---|
| 7,1E-07 | 0,34 |
| 7,9E-07 | 0,97 |
| 8,5E-07 | 1,14 |
| 9,5E-07 | 1,72 |
| 1,0E-06 | 1,95 |
| 1,1E-06 | 2,58 |
| 1,2E-06 | 2,75 |
| 1,3E-06 | 3,15 |
| 1,4E-06 | 3,15 |
| 1,6E-06 | 3,15 |
| 1,7E-06 | 3,44 |
| 2,0E-06 | 4,59 |
| 2,4E-06 | 5,62 |
| 2,9E-06 | 5,68 |
| 3,5E-06 | 5,85 |
| 4,2E-06 | 6,02 |
| 5,0E-06 | 6,02 |
| 6,0E-06 | 6,02 |
| 7,2E-06 | 5,96 |
| 8,6E-06 | 5,62 |

| | |
|---|---|
| 1,0E-05 | 5,16 |
| 1,2E-05 | 4,59 |
| 1,4E-05 | 3,84 |
| 1,7E-05 | 3,09 |
| 2,1E-05 | 2,58 |
| 2,5E-05 | 1,89 |
| 3,0E-05 | 1,26 |
| 3,6E-05 | 0,86 |
| 4,3E-05 | 0,57 |
| 5,1E-05 | 0,28 |
| 6,1E-05 | 0,05 |

A distribuição acima, quando representada em um gráfico, Figura 16, reproduz a distribuição encontrada no artigo [PRADO; ZANOTO; MULLER, 2001]:

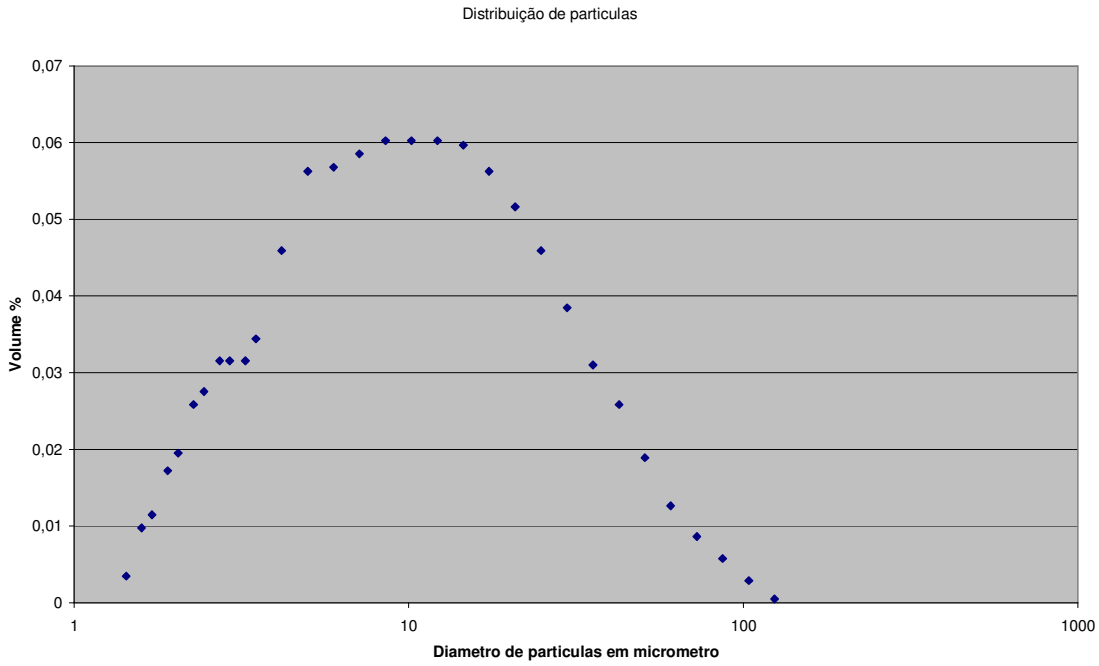



Figura 16 - Distribuição de partículas discretizada.

Prado et al. [PRADO; ZANOTO; MULLER, 2001] calcularam a cinética de densificação (densidade relativa em função do tempo de tratamento) de compactos de partículas de vidro nas seguintes temperaturas: 959, 966, 980, 989, 999 e 1017 K, e obtiveram o gráfico da Figura 17, que será utilizado para a validação dos resultados do presente trabalho.

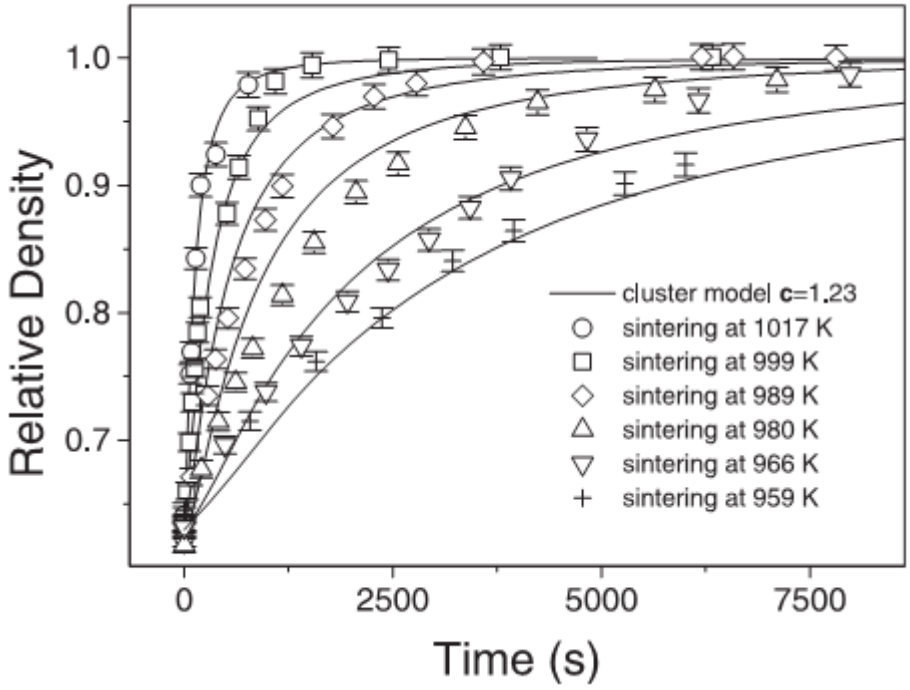
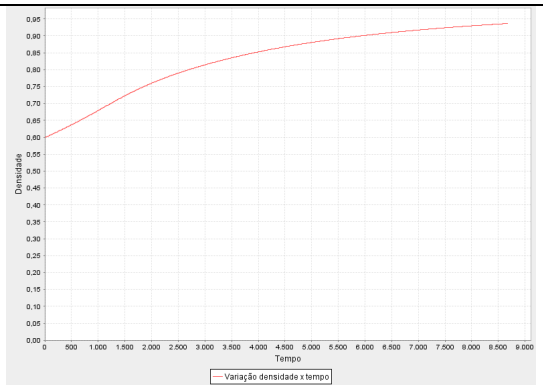


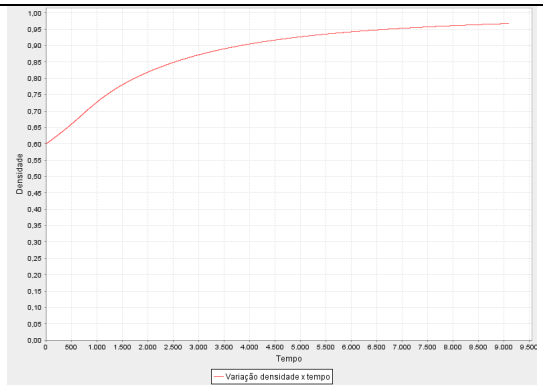

Figura 17 – Cinética de sinterização de vidro borosilicato . [PRADO; ZANOTO; MULLER, 2001]

Verificou-se no artigo [PRADO; ZANOTO; MULLER, 2001] que o valor da tensão superficial variou de 0,327 J/m$^2$ para 959 K a 0,325 J/m$^2$ para 1017 K. Para 966, 980, 989 e 999 K foi utilizado o valor de 0,326 J/m$^2$.

Os gráficos resultantes do programa desenvolvido são mostrados nas Figuras 18 a 20, para as temperaturas indicadas. As Tabelas 3 a 5 mostram os resultados obtidos comparados com valores listados no artigo [PRADO; ZANOTO; MULLER, 2001], para os tempos de 2500, 5000 e 7500 s de sinterização.




959 K

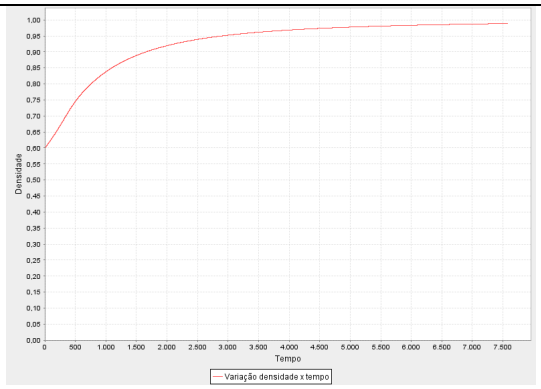



966 K

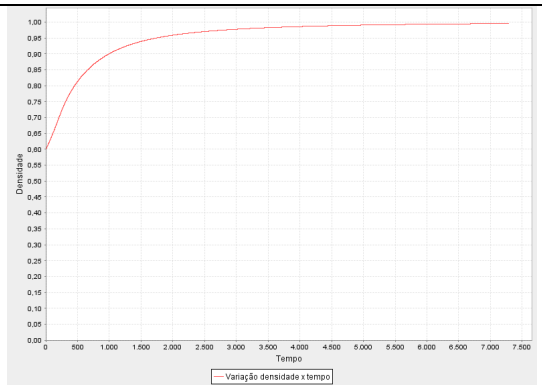
Figura 18 – Valores de densidades encontrados para as temperaturas de 959 K e 966 K.

Tabela 3 - Comparação entre os valores encontrados na literatura e pelo software para 959 K e 966 K.

| Temperatura (K) | Fonte | Tempo (s) | | |
|---|---|---|---|---|
| | | 2500 | 5000 | 7500 |
| 959 | literatura | 0,8 | 0,87 | 0,9 |
| | software | 0,79 | 0,87 | 0,93 |
| 966 | literatura | 0,85 | 0,92 | 0,95 |
| | software | 0,85 | 0,93 | 0,96 |




980 K

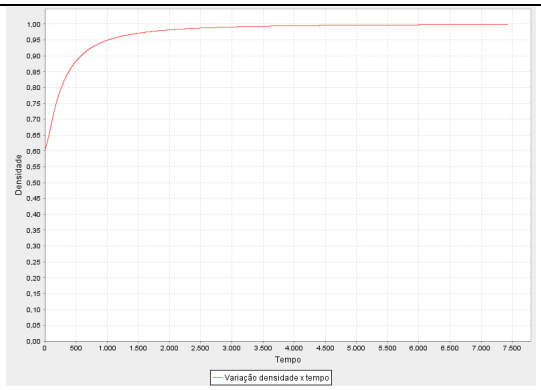



989 K

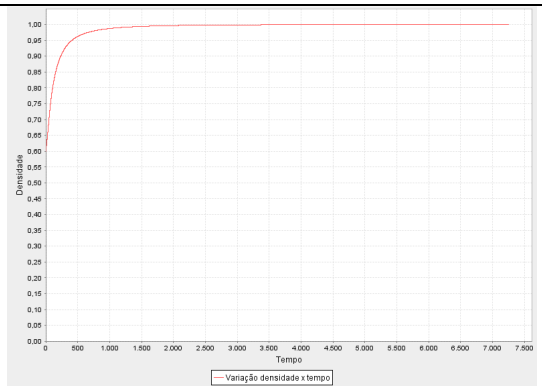
Figura 19 – Valores de densidades encontrados para as temperaturas de 980 K e 989 K.

Tabela 4 – Comparação dos valores encontrados na literatura e pelo software para 980 K e 989K

| Temperatura (K) | Fonte | Tempo (s) | | |
|---|---|---|---|---|
| | | 2500 | 5000 | 7500 |
| 980 | literatura | 0,93 | 0,96 | 0,98 |
| | software | 0,94 | 0,97 | 0,99 |
| 989 | literatura | 0,96 | 0,98 | 0,99 |
| | software | 0,97 | 0,99 | 1 |




999 K




1017 K

Figura 20 – Valores de densidades encontrados para as temperaturas de 999 K e 1017 K.

Tabela 5 - Comparação dos valores encontrados na literatura e pelo software 999 K e 1017 K.

| Temperatura (K) | Fonte | Tempo (s) | | |
|---|---|---|---|---|
| | | 2500 | 5000 | 7500 |
| 999 | literatura | 0,97 | 0,98 | 1 |
| | software | 0,98 | 0,99 | 1 |
| 1017 | literatura | 1 | 1 | 1 |
| | software | 1 | 1 | 1 |

Observa-se grande correspondência entre os valores resultantes do software e da literatura [PRADO; ZANOTO; MULLER, 2001], o que valida o programa desenvolvido para cálculos de sinterização por fluxo viscoso para uma distribuição de tamanhos de partículas.

## *4.2 - Validação com cristalização concorrente*

Para a validação do programa para o caso de sinterização por fluxo viscoso com cristalização concorrente, foram utilizados os dados de viscosidade e taxa de crescimento de cristais de um vidro com a mesma composição do diopsídio ($CaO.MgO.2SiO_2$) do artigo [NASCIMENTO; FERREIRA; ZANOTTO, 2004] e cálculos para comparação efetuados escrevendo-se o

modelo de Clusters no software de cálculos matemáticos Mathcad. Esse software possui embutidos os métodos de cálculo numérico e solução de raízes de funções que foram necessários programar no software desenvolvido neste trabalho. Os dados para o vidro de diopsídio e demais valores utilizados são dados da Tabela 6.

Tabela 6 – Valores utilizados para o cálculo no caso com cristalização concorrente

| Variável | Valor (unidade) |
|---|---|
| $\rho_o$ | 0,6 |
| $\gamma$ | 0,371 J/m$^2$ |
| $T_o$ | 750,9 K |
| $A$ | 5,32x10$^{-5}$ Pa.s |
| $B$ | 9,12x10$^3$ K |
| $U$ | 10$^{-9}$ m/s |
| $N_s$ | 10$^{11}$ m$^{-2}$ |
| $R$ | 5x10$^{-5}$ m |

Na validação foram utilizados dois valores de temperaturas, 1098,15 K e 1120 K. O Mathcad já tem incluso ferramentas computacionais desenvolvidas e testada por vários anos, resultando em credibilidade para podendo assim comparar os cálculos efetuados pelo mesmo com o aqui desenvolvido. Essas ferramentas computacionais são escolhidas de forma dinâmica, o que permite dizer se foi feita boa escolha do método numérico de integração (Método de Simpson).

Para a temperatura de 1098,15 K e o tempo de 2000 s, foram obtidos graficamente os valores de densidade relativa, descritos na Figura 22.

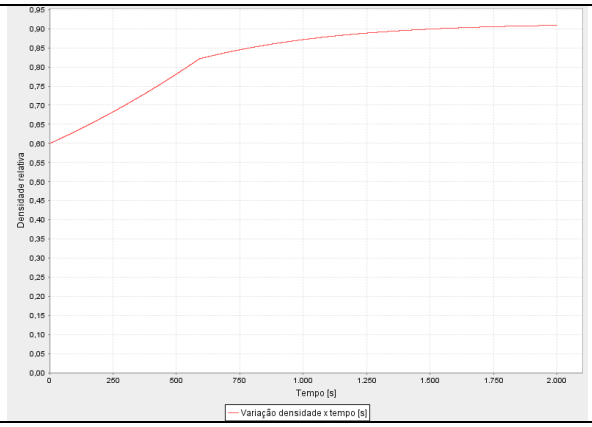
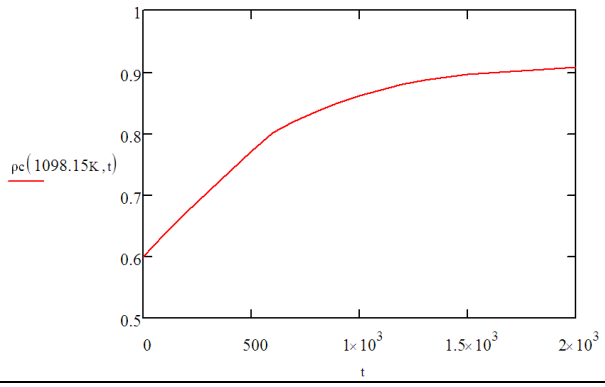
| Java | MathCad |
| --- | --- |
|  |  |



Figura 21 – Comparação entre os valores de densidades encontrados pelos softwares para as temperaturas de 1098.15K e 2000 segundo de tempos

Agora para a temperatura de 1120 K e o tempo de 500 s, conforme a figura 22:

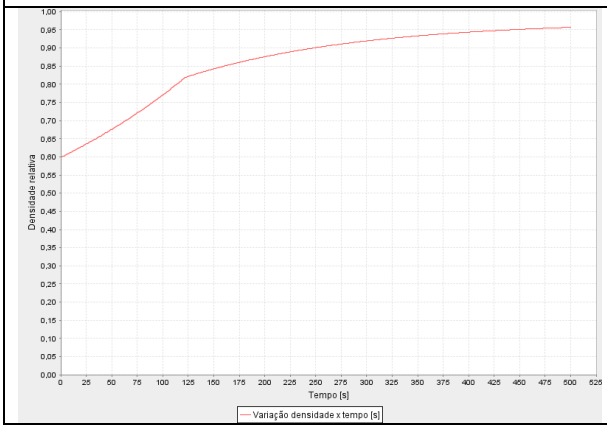
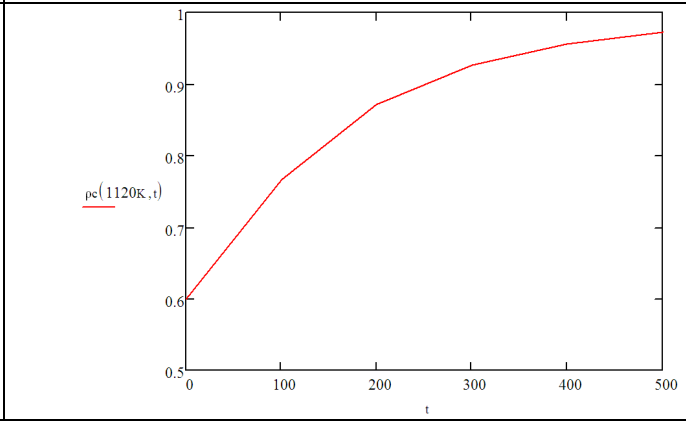
| Java | MathCad |
| --- | --- |
|  |  |



Figura 22 – Comparação entre os valores de densidades encontrados pelos softwares para as temperaturas de 1120K e 500 segundo de tempo

Comparando os valores obtidos em apenas um gráfico para cada caso e para o tempo de 2000 s podemos ter os gráficos comparativos para cada resultado (Figuras 23 e 24).

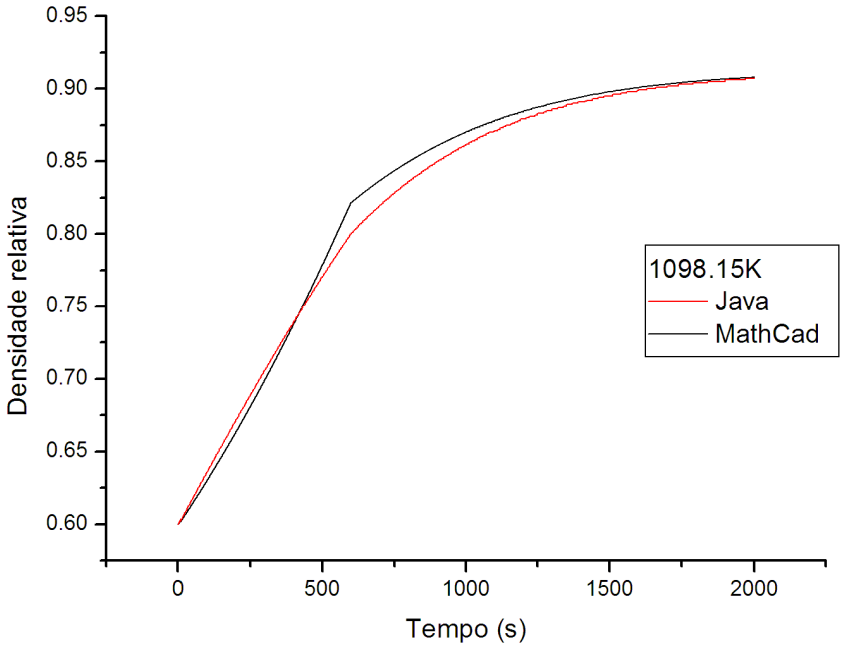



Figura 23 – Comparação entre os valores de densidades encontrados pelos programas para as temperaturas de 1098,15 K e 2000 s de tempo.

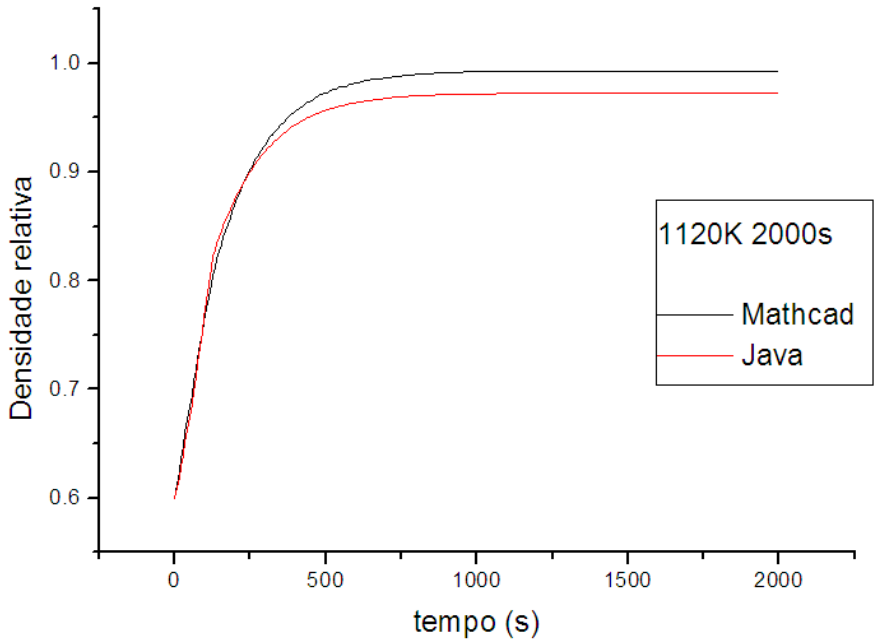



Figura 24 – Comparação entre os valores de densidades encontrados pelos softwares para as temperaturas de 1120 K e 2000 s de tempo.

Observando as Figuras 23 e 24, verifica-se a proximidade dos resultados esperados nesse caso em que considera a cristalização concorrente. A pequena

variação dos valores, principalmente no momento em que o mecanismo de MS está em funcionamento, deve-se possivelmente à diferença de valores de $a_0$ e $t_{08}$, sendo que uma pequena variação desses valores pode causar uma grande variação no resultado final, já que o método de Simpson é composto de um grande número de iterações e de somatórias. Então como sugestão para futuros trabalhos, seria interessante aprimorar o método de raiz descrito no Anexo 2, para que se obtenham valores de $a_0$ e $t_{08}$ com maior precisão.

Como sugestão para a continuidade desse trabalho, sugerimos a construção de uma interface gráfica, para facilitar o uso pelo usuário final, e uso de outras ferramentas, como o Método de Elementos Finitos para simulações do problema de sinterização de vidros.

# 5. CONCLUSÃO

Foi desenvolvido no trabalho apresentado um software em JAVA para o cálculo da sinterização de materiais vítreos em duas situações, sem cristalização e com cristalização superficial das partículas.

No caso em que se considera uma distribuição de partículas e sem a ocorrência de cristalização, houve grande correspondência dos valores encontrados na literatura e obtidos pelo software desenvolvido. Pequenos erros devem-se, possivelmente, à diferença de aproximações entre os dois programas.

Já no segundo caso, onde foram utilizadas ferramentas de cálculos numéricos para aproximar uma solução para o fenômeno da cristalização concorrente, foram obtidos resultados semelhantes entre diferentes fontes de resultados (MathCad e o programa em Java). Pequenas variações nos resultados, principalmente quando se trata do mecanismo de Mackenzie-Shuttleworth, foram observadas. Essas pequenas variações devem-se possivelmente ao grande número de cálculos realizados para estimar os valores das integrais pelo método de Simpson, que como sabemos é uma somatória de vários valores das funções em determinado intervalo.

Como conclusão final, foi verificado que é possível utilizar o software desenvolvido para estimar a densidade de compactos de partículas de vidro sinterizados utilizando o mecanismo de Clusters.

# 6. BIBLIOGRAFIA


BURDEN, R. L.; FAIRES, J. D. **Análise Numérica**. Pioneira Thomson Learning, 2003.

CALLISTER, W. D. **Ciência e engenharia de materiais: uma introdução**. 5.ed. Rio de Janeiro: LTC, 2002. 589p.

GRANDO, L.; MATTOS, R. S. **Desenvolvimento de um software em java para estudo da sinterização de materiais vítreos**, Monografia para a disciplina de Pesquisa e Desenvolvimento e Caracterização de Materiais do Curso de Engenharia de Materiais. – UNESP – Guaratinguetá, 2008.

JFreeChart – URL: **http://www.jfree.org/jfreechart/** - Último acesso em Setembro de 2008

MATTOS, R. S. **Implementação do fenômeno magnetodinâmico com aplicação do Método dos Elementos Finitos**. 2007. 73f. Dissertação (Trabalho de Conclusão de Curso Engenharia de Computação) – Universidade Brás Cubas, Mogi das Cruzes. 2007.

NASCIMENTO, M. L. F.; FERREIRA; E. B.; ZANOTTO, E. D. **Kinetics and mechanisms of crystal growth and diffusion in a glass-forming liquid** Journal of Chemical Physics, 121 (2004) 8924-8928

NETO, O. M. **Entendendo e dominando o JAVA** 1.ed. São Paulo: Digerati Books, 2004. 384p.

PRADO, M. O; ZANOTTO, E. D. **Concurrent sintering and crystallization of glass spheres** Int. Symp. on Crystallization in Glasses & Liquids. / Glastechn. Ber. Glass Sci. Tech. 73 C1 (2000) 194-196

PRADO, M. O.; ZANOTO, E, D.; MULLER, R. **Model for sintering polydispersed particles**. Journal of Non-Crystalline Solids, v. 279, p. 169-178, 2001.

RAO, R. J. **Strutural chemistry of glass**. Elsevier, 2002. 567p.

SOARES, V. L. **Vitrocerâmicas do sistema Li2O – Al2O3-SiO2(LAS) via sinterização com cristalização concorrente**. 2007. 160 f. Dissertação (Mestrado em Engenharia de Materiais) – Universidade Federal de São Carlos, São Carlos, 2007.

# 7. ANEXOS

## *Anexo 1 - Métodos numéricos*

Integrar numericamente uma função $y = f(x)$ num dado intervalo [a,b] consiste, em geral, em integrar um polinômio $P_n(x)$ que aproxime $f(x)$ no dado intervalo. [BURDEN; FAIRES, 2003]

Em particular, se $y = f(x)$ for dada por uma tabela ou, o que é o mesmo, por um conjunto de pares ordenados (x0, f(x0)), (x1, f(x1)),..., (xn, f(xn)) (onde os xi podem ser supostos em ordem crescente), x0 = a e xn = b, podemos usar como polinômio de aproximação para a função $y = f(x)$, no intervalo [a,b], o seu polinômio de interpolação. [BURDEN; FAIRES, 2003]

O polinômio de interpolação para a função $y = f(x)$ no intervalo [a,b], a = x0, b = xn, é um polinômio de aproximação para f(x) em qualquer sub-intervalo [xi,xj], $0 \le i \le n$, $0 \le j \le n$ de [a,b]. Podemos então usar o polinômio $P_n(x)$ para integrar $f(x)$ em qualquer desses sub-intervalos. [BURDEN; FAIRES, 2003]

As vantagens de se integrar um polinômio que aproxima $y = f(x)$ ao invés de $f(x)$, são principalmente as seguintes:[BURDEN; FAIRES, 2003]

a) $f(x)$ pode ser uma função difícil de integrar ou para a qual a integração seja impossível, como no nosso caso, onde as funções presentes que descrevem o modelo de cluster considerando a cristalização superficial são de integração analítica impossível.

b) a solução analítica do resultado da integral é conhecida, mas seu cálculo só pode ser obtido aproximadamente.

c) a função é conhecida apenas em pontos discretos obtidos através de experimentos.

**Método de Simpson:**

Considerando *n=2*, isto é, queremos obter uma fórmula para integrar *f(x)* entre três pontos consecutivos $x_0$, $x_1$ e $x_2$ usando polinômio do segundo grau. Temos: [BURDEN; FAIRES, 2003]

$$\int_{x_o}^{x_n} f(x)dx \cong \sum_{k=0}^{2} f_k h C_k^2 \quad (24)$$

Podemos provar por ferramentas matemáticas [BURDEN; FAIRES, 2003], que fogem ao escopo desse trabalho, que a seguinte relação procede:

$$\int_{x_0}^{x_N} f(x)dx \cong \frac{h}{3}[f(x_0) + 4f(x_1) + f(x_2)] \quad (25)$$

A fórmula de Newton-Cotes acima é conhecida como Regra 1/3 de Simpson.

A generalização da regra 1/3 de Simpson para integração ao longo de um intervalo [*a,b*] é feita dividindo [*a,b*] num número par *2N* de subintervalos de amplitude $h = \frac{b-a}{2N}$, de tal forma que $x_0 = a$, $x_{2N} = b$. Temos então:

$$\int_{x_0}^{x_{2N}} f(x)dx = \int_{x_0}^{x_2} f(x)dx + \int_{x_2}^{x_4} f(x)dx + \ldots + \int_{x_{2N-2}}^{x_{2N}} f(x)dx \quad (26)$$

Usando a regra de 1/3 de Simpson em cada sub-intervalo [ *xj, xj+2], j = 0,2,...,2N-2* ], obtemos:

$$\int_{x_0}^{x_{2N}} f(x)dx \cong \frac{h}{3}\left[f(x_0) + 4f(x_1) + f(x_2)\right] + \frac{h}{3}\left[f(x_2) + 4f(x_3) + f(x_4)\right]$$
$$+ \ldots + \frac{h}{3}\left[f(x_{2N-2}) + 4f(x_{2N-1}) + f(x_{2N})\right]. \quad (27)$$

Na expressão anterior vemos que, com exceção de *f* calculada nos pontos *x0* e *x2N*, o cálculo de *f* nos pontos extremos de cada sub-intervalo de integração aparece duas vezes. Portanto, podemos escrever:

$$\int_{x_0}^{x_{2N}} f(x)dx \cong \frac{h}{3}\left[f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + 2f(x_4) + \ldots + 2f(x_{2N-2}) + 4f(x_{2N-1}) + f(x_{2N})\right]$$
(28)

Que resultara na Regra 1/3 de Simpson Generalizada, lembrando que o valor de 2N deve ser um numero par. [BURDEN; FAIRES, 2003]

No presente trabalho, no caso em que é necessário o uso do método composto de Simpson, o valor de 2N é igual a 6 e o intervalo h é de 0,2 segundo, já que o intervalo de integração escolhido foi de 1 segundo. Foi observado que variando os valores de 2N, pouca variação ocorre no resultado final, por isso foi-se escolhido esse valor, para ganho de tempo computacional, que comparado aos cálculos executados pelo Mathcad, tem-se grande vantagem quanto ao período de tempo.

## *Anexo 2 - Cálculo do valor de $t_{08}$ e $a_0$ para o caso de presença de cristalização concorrente*

Para o cálculo do valor de $t_{08}$ para cada valor de raio de partícula, considera-se a função de Frenkel (Equação 20), no momento em que o valor de sua densidade relativa é de 0,8.

A mesma pode ser representada da seguinte forma:

$$\rho_{C,F}(t) = \rho_0 + \frac{3C\rho_0}{\eta(T)} \int_0^t \left(1 - \frac{C}{\eta(T)} t'\right)^{-4} e^{-\pi N s U(T)^2 t'^2} dt' \quad (29)$$

Para $\rho_{C,F} = 0{,}8$

$$0{,}8 = \rho_0 + \frac{3C\rho_0}{\eta(T)} \int_0^t \left(1 - \frac{C}{\eta(T)} t'\right)^{-4} e^{-\pi N s U(T)^2 t'^2} dt' \quad (30)$$

Ou melhor

$$0{,}8 - \rho_0 = \frac{3C\rho_0}{\eta(T)} \int_0^t \left(1 - \frac{C}{\eta(T)} t'\right)^{-4} e^{-\pi N s U(T)^2 t'^2} dt' \quad (31)$$

Pode-se dizer que o valor de $t_{08}$ é encontrado quando a igualdade acima é estabelecida.

Para encontrar o valor do $t_{08}$ para um determinado raio, é variado o valor do tempo em ordem crescente de 0 s até o valor que tal igualdade é obtida. Este valor corresponde ao mesmo valor utilizado para determinar $a_0$ e ao valor de tempo a partir do qual o mecanismo de MS toma o lugar do mecanismo de Frenkel.

Podemos demonstrar graficamente o processo para encontrar $t_{08}$

Considerando os dados do diopsidio:

$\rho_0 = 0{,}6$

$\gamma = 0{,}371$ J/m$^2$

$\eta(1098{,}15\ \text{K}) = 1{,}38 \times 10^{7}$ Pa.s

$U(1098{,}15\ \text{K}) = 10^{-9}$ m/s

$N_s = 10^{-12}$ m$^{-2}$

$r = 0{,}00005$ m

Assim podemos considerar a Equação (20) da forma:

$$B = I(t) \tag{32}$$

onde:

$$I(t) = \frac{3C\rho_0}{\eta(T)} \int_0^t \left(1 - \frac{C}{\eta(T)} t'\right)^{-4} e^{-\pi N s U (T)^2 t'^2} dt' \tag{33}$$

$$B = 0{,}8 - \rho_0 \tag{34}$$

E graficamente:

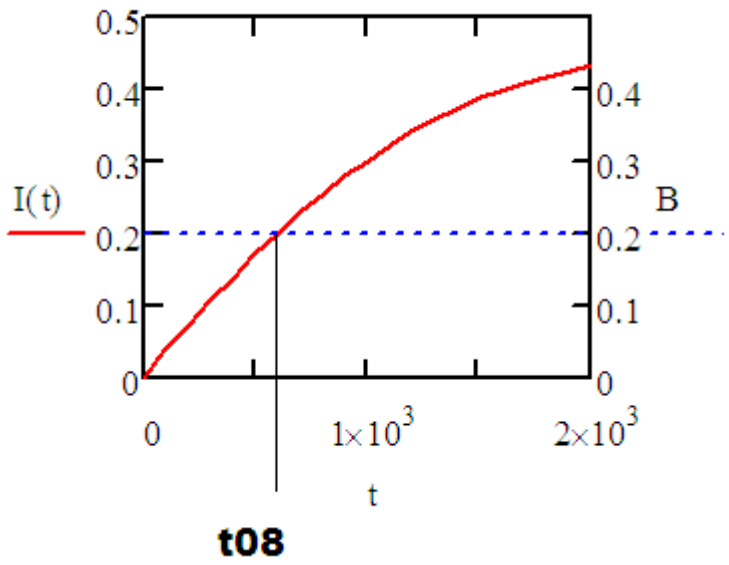



Figura 25 – Representação do método de localização de raiz para $t_{08}$

A intersecção entre as duas curvas é o valor de $t_{08}$

Para o cálculo de $I(t)$ foi utilizada a regra de 1/3 de Simpson e foi variando o valor do t para que a tal igualdade procedesse.

Assim, pode-se obter o valor de $t_{08}$ para cada valor de raio.

**Cálculo do valor de $a_0$**

Para o cálculo do valor de $a_0$ para cada raio de partícula da distribuição, é necessário o uso da equação de MS (Equação 21) no momento em que a densidade relativa é igual a 0,8.

$$\rho_{C,MS}(t) = \rho_0 + (1-\rho_0)\frac{3\gamma}{2\eta(T)a0}\int_0^t e^{\left(\frac{-3\gamma'}{2\eta(T)a0}\right)} e^{-\pi NsU(T)^2 t'^2} dt' \tag{35}$$

$$0{,}8 = \rho_0 + (1-\rho_0)\frac{3\gamma}{2\eta(T)a0}\int_0^t e^{\left(\frac{-3\gamma'}{2\eta(T)a0}\right)} e^{-\pi NsU(T)^2 t'^2} dt' \tag{36}$$

$$0{,}8 - \rho_0 = (1-\rho_0)\frac{3\gamma}{2\eta(T)a0}\int_0^t e^{\left(\frac{-3\gamma'}{2\eta(T)a0}\right)} e^{-\pi NsU(T)^2 t'^2} dt' \tag{37}$$

Outra consideração feita para tal aproximação do valor de a0 para cada raio de partícula da distribuição foi referente ao valor dos poros, que eles nunca sejam maiores do que os valores do raio em questão. E o valor de a0 é uma fração do valor do raio sempre menor que 1 e maior do que zero.

$$0 \le a0 \le r \tag{38}$$

Ou melhor

$$a0 = (k.r) \tag{39}$$

onde k = 0 a 1.

$$0{,}8-\rho_0=(1-\rho_0)\frac{3\gamma}{2\eta(T)kr}\int_0^t e^{\left(\frac{-3\gamma'}{2\eta(T)kr}\right)}e^{-\pi NsU(T)^2t'^2}dt' \qquad (40)$$

No software a variação de *k* é feita ao passo de 0,001, resultando em possíveis 1000 valores do mesmo e o tempo utilizado na integral é o $t_{08}$ já explicado anteriormente.

$$D(k)=(1-\rho_0)\frac{3\gamma}{2\eta(T)kr}\int_0^{t08} e^{\left(\frac{-3\gamma'}{2\eta(T)kr}\right)}e^{-\pi NsU(T)^2t'^2}dt' \qquad (41)$$

Da mesma forma de $t_{08}$, para o calculo da integral $D(k)$ é utilizado o método de Simpson simples de integração numérica.

Graficamente:

Considerando os mesmos valores citados acima para t08 e considerando o tempo de integração o t08 para determinado raio, que no nosso caso é de:

$t_{08}$(0,00005 m)=598 s e

$$B = 0{,}8 - \rho_0 \qquad (42)$$

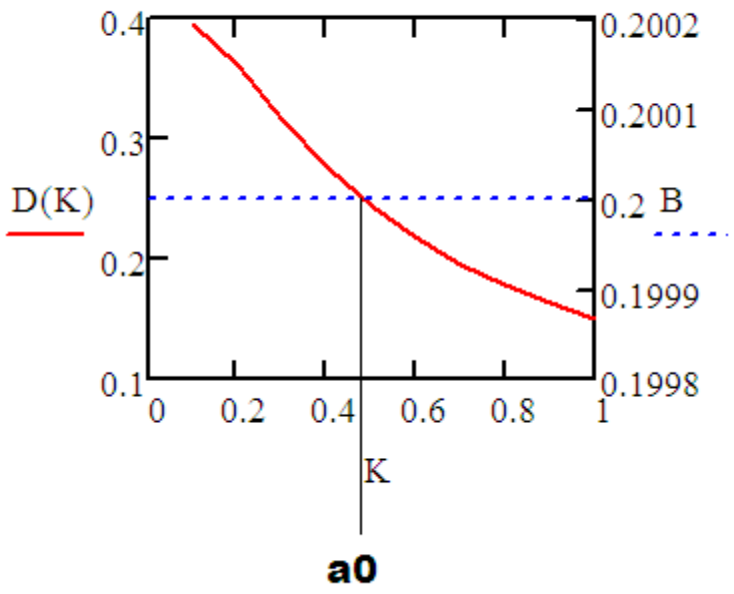



Figura 26 – Representação do método de localização de raiz para $a_0$

Conhecendo o valor de $k$ podemos determinar o valor de $a_0$ para cada raio. E multiplicando o valor de $k$ encontrado para esse raio pelo mesmo, já que para cada valor de raiz teremos outro de $a_0$.

# *Anexo 3 – Códigos fontes dos programas desenvolvidos*

**Classe Main**

```
package vidgrand;
public class Main {
   public static void main(String[] args) {
            new LGB_Calculos();}}
```

**Classe LGB_Dados**

```
package vidgrand;
public class LGB_Dados {
   double viscosidade;
   double tensao_superficial;
   double taxa_crescimento_cristal;
   double densidade_vidro_verde;
   double densidade_material_cristalizado;
   double energia_ativacao_aparente;
   double constante_gas;
   double temperatura_empirica;
   double viscosidade_infinita;
   double temperatura;
   LGB_Dados()
   {
      viscosidade=0.;
      tensao_superficial=0.;
      taxa_crescimento_cristal=0.;
      densidade_vidro_verde=0.6;
      densidade_material_cristalizado=0.;
      energia_ativacao_aparente=83700.;
      constante_gas=8.31;
      temperatura_empirica=0.;
      viscosidade_infinita=0.;
      temperatura=0.;
   }
      void setViscosidade(double valor)
   {viscosidade = valor;}
      double getViscosidade()
   {return viscosidade;}
      void setTensao_Superficial(double valor)
   {tensao_superficial = valor;}
   double getTensao_Superficial()
   {return tensao_superficial;}
   void setTaxa_Crescimento_Cristal(double valor)
   {taxa_crescimento_cristal = valor;}
   double getTaxa_Crescimento_Cristal()
   {return taxa_crescimento_cristal;}
   void getDensidade_Vidro_Verde(double valor)
   {densidade_vidro_verde = valor;}
   double getDensidade_Vidro_Verde()
   {return densidade_vidro_verde;}
   void setDensidade_Material_Cristalizado(double valor)
   {densidade_material_cristalizado = valor;}
   double getDensidade_Material_Cristalizado()
   {return densidade_material_cristalizado}
    double getEnergia_ativacao_aparente()
   {return energia_ativacao_aparente; }
      double getConstante_Gas(){
     return constante_gas;}

   void setTemperatura_Empirica(double valor)
```

```
   {temperatura_empirica = valor;}
      double getTemperatura_Empirica()
   { return temperatura_empirica}
     void setViscosidade_Infinita(double valor)
   { viscosidade_infinita = valor;}
      double getViscosidade_infinita()
   { return viscosidade_infinita}
   void setTemperatura(double valor)
   {temperatura = valor:}
      double getTemperatura()
   { return temperatura;}}
```

**Classe LGB_BaseDados**

```
package vidgrand;
import vidgrand.Particulas;
public class LGB_BaseDados extends Particulas {
  int quant_registro;
   LGB_Dados dados[];
   LGB_BaseDados()
   {}}
    {return dados;}
      LGB_Dados getDado(int i)
   {LGB_Dados aux=null;
      if(quant_registro>0&&i<quant_registro)
         aux = dados[i];
      return aux;}}
```

**Classe Particulas**

```
package vidgrand;
public class Particulas {
   double r;
   double vr;
   public Particulas()
   {  r =  0;
      vr = 0;}
      public void setraio (double pr)
   {  r = pr;}
      public double getraio()
   {   return r;}
      public void setvr (double pvr)
   {       vr = pvr ;  }
      public double getvr()
   {  return vr;}}
```

**Classe LGB_Grafico**

```
package vidgrand;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.axis.NumberAxis;
import org.jfree.chart.axis.ValueAxis;
import org.jfree.chart.plot.XYPlot;
import org.jfree.chart.renderer.xy.StandardXYItemRenderer;
import org.jfree.data.time.Hour;
import org.jfree.data.time.TimeSeries;
import org.jfree.data.time.TimeSeriesCollection;
import org.jfree.data.xy.XYSeries;
import org.jfree.data.xy.XYSeriesCollection;
public class LGB_Grafico extends javax.swing.JDialog {
   public LGB_Grafico(java.awt.Frame parent, boolean modal) {
      super(parent, modal);
      initComponents();}
   public void PlotarGrafico(String titulo, String eixoX,String eixoY,double[] densidade, double[] tempo )
   {       XYSeriesCollection dataset = new XYSeriesCollection();
```

```
        XYSeries series = new XYSeries(titulo);
        for (int j = 0; j < densidade.length; j++) {
           series.add(tempo[j], densidade[j]);    }
           dataset.addSeries(series);
         ValueAxis axis1 = new NumberAxis(eixoX);
        ValueAxis axis2 = new NumberAxis(eixoY);
         StandardXYItemRenderer renderer = new StandardXYItemRenderer();
         XYPlot plot = new XYPlot(dataset, axis1, axis2, renderer);
         JFreeChart chart = new JFreeChart(plot);
         ChartPanel panel = new ChartPanel(chart);
         setSize(800, 600);
        getContentPane().add(panel);  }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {
        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
        pack();
    }// </editor-fold>
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                LGB_Grafico dialog = new LGB_Grafico(new javax.swing.JFrame(), true);
                dialog.addWindowListener(new java.awt.event.WindowAdapter() {
                    public void windowClosing(java.awt.event.WindowEvent e) {
                        System.exit(0);
                    }
                });
                dialog.setVisible(true);
            }
        });
    }
}
```

**Classe LGB_Calculos**

```
package vidgrand;
import javax.swing.JOptionPane;
public class LGB_Calculos extends LGB_Dados {
   Particulas [] particulas;
   double [] t08;
   double [] densidade;
   double [] ao;
   double [] er;
   //public LGB_Calculos(double ppo, double pgama, int pnum, int iteracoes,double variacao,double
ptemperatura)
   public LGB_Calculos()
   {
      // Valores para testes
      int numero_particulas=31; // Alterar para classes de tamanho
      double po = 0.6;
      //double ni = 61400000.;
      double ni=0; // visco
      //double ni = 559578339.3;
      double gama = 0.325;
      double t = 0.0000;
      double Ev = 83700;
      double R = 8.31;
      double T0 = 562.48;
      double var_tempo =0.;
      int tempo = 0;
      double []v_tempo;
```

```
        double noo = 0.018909;
        int K_tempo=0;
        double T = 0;
        double ax, ax1, ax2 = 0.;
        double ax2er =0.;
        double ax2er1 = 0.;
        double c = 1.23;
        particulas = new Particulas[numero_particulas];
        for(int i=0;i<numero_particulas;i++)
        {
           particulas[i] = new Particulas();
        }
        particulas[0].setraio(0.000000714511);
        particulas[0].setvr(0.003443);
      particulas[1].setraio(0.000000795281);
      particulas[1].setvr(0.009754);
        particulas[2].setraio(0.000000854138);
        particulas[2].setvr(0.011475);
        particulas[3].setraio(0.000000950692);
        particulas[3].setvr(0.017213);
           particulas[4].setraio(0.00000102105);
        particulas[4].setvr(0.019508);
        particulas[5].setraio(0.00000113647);
      particulas[5].setvr(0.02582);
         particulas[6].setraio(0.00000122058);
        particulas[6].setvr(0.027541);
         particulas[7].setraio(0.0000013586);
        particulas[7].setvr(0.031557);
particulas[8].setraio(0.00000145910);
        particulas[8].setvr(0.031557);
      particulas[9].setraio(0.00000162404);
      particulas[9].setvr(0.031557);
        particulas[10].setraio(0.00000174424);
        particulas[10].setvr(0.034426);
        particulas[11].setraio(0.00000208509);
        particulas[11].setvr(0.045902);
particulas[12].setraio(0.00000249255);
        particulas[12].setvr(0.05623);
      particulas[13].setraio(0.00000297964);
      particulas[13].setvr(0.056803);
        particulas[14].setraio(0.00000356191);
        particulas[14].setvr(0.058525);
        particulas[15].setraio(0.00000425797);
        particulas[15].setvr(0.060246);
particulas[16].setraio(0.00000509005);
        particulas[16].setvr(0.060246);
      particulas[17].setraio(0.00000608473);
      particulas[17].setvr(0.0602459);
        particulas[18].setraio(0.00000727379);
        particulas[18].setvr(0.059672);
        particulas[19].setraio(0.00000869521);
        particulas[19].setvr(0.05623);
particulas[20].setraio(0.0000103944);
        particulas[20].setvr(0.051639);
      particulas[21].setraio(0.0000124256);
      particulas[21].setvr(0.045902);
        particulas[22].setraio(0.0000148538);
        particulas[22].setvr(0.038443);
        particulas[23].setraio(0.0000177565);
        particulas[23].setvr(0.030984);
```

```
particulas[24].setraio(0.0000212264);
       particulas[24].setvr(0.02582);
    particulas[25].setraio(0.0000253744);
    particulas[25].setvr(0.018934);
      particulas[26].setraio(0.000030330);
      particulas[26].setvr(0.012623);
      particulas[27].setraio(0.0000362606);
      particulas[27].setvr(0.008607);
particulas[28].setraio(0.0000433465);
       particulas[28].setvr(0.005738);
    particulas[29].setraio(0.0000518172);
    particulas[29].setvr(0.002869);
      particulas[30].setraio(0.0000619432);
      particulas[30].setvr(0.000492);
       String tempo1 = JOptionPane.showInputDialog("Digite o numero de iteração!");
       tempo = Integer.parseInt(tempo1);
      String K_tempo1 = JOptionPane.showInputDialog("Digite digite a constante do tempo");
       K_tempo = Integer.parseInt(K_tempo1);
       String var_temperatura = JOptionPane.showInputDialog("Digite a temperatura!");
       T = Double.parseDouble(var_temperatura);
       v_tempo = new double[tempo];
       // Criação de t08
       t08 = new double[numero_particulas];
       ao = new double[numero_particulas];
       er = new double[numero_particulas];
       double tmenor = 10000000000000000.;
       //calculo viscosidade
       ni = noo * Math.exp(Ev/(R*(T-T0)));
       System.out.println("Viscosidade: " + ni);
       //calculo auxiliar er
       for(int i=0;i<numero_particulas;i++)
            {   ax2er1 = particulas[i].getvr()/Math.pow(particulas[i].getraio(), c);
              ax2er += ax2er1 ;}
       System.out.println("ax2er: "+ ax2er);
            for(int i=0;i<numero_particulas;i++)
       {   t08[i] = (2.666666667*((Math.pow(0.8, 0.3333333)-Math.pow(po, 0.3333333))/Math.pow(0.8,
0.3333333))*(ni/gama)*particulas[i].getraio());
          System.out.println("Valor t08["+i + "]: "+t08[i]);
          System.out.println("Valor do raio: " + particulas[i].getraio());
                  //calculo ao
          ax= (double) (Math.log(1-po)-Math.log(0.2));
          ax1= (double)(Math.pow(0.8, 0.333333333333333)-Math.pow(po, 0.3333333333333333));
          ax2= (double)Math.pow(0.8, 0.333333333333333);
                  ao[i] = (double)((4/ax)*(ax1)/(ax2))*particulas[i].getraio();  //
          System.out.println("ao: " + ao[i]);
          //calculo t08
          if(tmenor> t08[i]){
            tmenor = t08[i];
            System.out.println(">>tmenor :<< " + tmenor );    }
          er[i]  =(double)(1/Math.pow(particulas[i].getraio(), c)/ax2er);}
       for(int i=0;i<numero_particulas;i++){
            System.out.println("fator de forma: " + er[i]);
}
       System.out.println("tmenor :" + tmenor );
       var_tempo = tmenor/K_tempo;
       densidade = new double[tempo];
       // impletação do cluster
       double aux,aux2;
```

```
      int z=0;
      for(int i=0;i<tempo;i++)
      {
         for(z=0;z<t08.length;z++)
         {
            if( t < t08[z]   ) //&& t<((8*ni*particulas[z].getraio())/3*gama)  //&& densidade[i] > 0.6
            {
               // Frenkel
               densidade[i] += (double)(po*Math.pow((1-((3*gama*t)/(8*ni*particulas[z].getraio()))), -
3))*particulas[z].getvr()*er[z];
              //  System.out.println("Frenkel");}
            else
            {
               // MS
               aux = (double)(-3*gama*t);
               densidade[i] += (double)(1 - Math.exp((aux/(2*ni*ao[z]))+Math.log(1-po)))
*particulas[z].getvr()*er[z] ;  //particulas[z].getvr()*er[i]
            }
         }
         t += (double) var_tempo;
         v_tempo[i] = t;
         System.out.println("densidade " + densidade[i]);}
      LGB_Grafico grafico = new LGB_Grafico(null,true);
      grafico.setTitle("Gráfico ('Densidade x Tempo')");
      grafico.PlotarGrafico("Variação densidade x tempo", "Tempo", "Densidade", densidade, v_tempo );
      grafico.setLocationRelativeTo(null);
      grafico.setVisible(true); }}
```

## Método cálculo numérico

**Classe Main**

```
package vidgrand;
import java.io.IOException;
public class Main {
   public static void main(String[] args) throws IOException {
      new LGB_Calculos() ;
   }
}
```

**Classe LGB_Dados**

```
package vidgrand;
public class LGB_Dados {
   double viscosidade;
   double tensao_superficial;
   double taxa_crescimento_cristal;
   double densidade_vidro_verde;
   double densidade_material_cristalizado;
   double energia_ativacao_aparente;
   double constante_gas;
   double temperatura_empirica;
   double viscosidade_infinita;
   double temperatura;
   LGB_Dados()
   {
      viscosidade=0.;
      tensao_superficial=0.;
      taxa_crescimento_cristal=0.;
      densidade_vidro_verde=0.6;
      densidade_material_cristalizado=0.;
      energia_ativacao_aparente=83700.;
      constante_gas=8.31;
      temperatura_empirica=0.;
```

```
        viscosidade_infinita=0.;
        temperatura=0.;  }
        void setViscosidade(double valor)
    {    viscosidade = valor;}
        double getViscosidade()
    {       return viscosidade;}
        void setTensao_Superficial(double valor)
    {       tensao_superficial = valor;  }
        double getTensao_Superficial()
    {       return tensao_superficial;}
        void setTaxa_Crescimento_Cristal(double valor)
    {    taxa_crescimento_cristal = valor;    }
        double getTaxa_Crescimento_Cristal()
    { return taxa_crescimento_cristal; }
        void getDensidade_Vidro_Verde(double valor)
    {    densidade_vidro_verde = valor;}
        double getDensidade_Vidro_Verde()
    {      return densidade_vidro_verde;  }
        void setDdensidade_Material_Cristalizado(double valor)
    {       densidade_material_cristalizado = valor; }
        double getDdensidade_Material_Cristalizado()
    {  return densidade_material_cristalizado; }
    double getEnergia_ativacao_aparente()
    { return energia_ativacao_aparente;  }
        double getConstante_Gas()
    { return constante_gas;  }
        void setTemperatura_Empirica(double valor)
    {  temperatura_empirica = valor; }
        double getTemperatura_Empirica()
    {  return temperatura_empirica; }
        void setViscosidade_Infinita(double valor)
    {viscosidade_infinita = valor; }
    double getViscosidade_infinita()
    { return viscosidade_infinita; }
    void setTemperatura(double valor)
    {temperatura = valor; }
        double getTemperatura()
    {  return temperatura; }}
```

**Classe LGB_BaseDados**

```
package vidgrand;
import vidgrand.Particulas;
public class LGB_BaseDados extends Particulas {
   int quant_registro;
    LGB_Dados dados[];
    LGB_BaseDados()
    {   }
    }
     {  return dados;}
        LGB_Dados getDado(int i)
    {
        LGB_Dados aux=null;
        if(quant_registro>0&&i<quant_registro)
           aux = dados[i];
        return aux;}}
```

**Classe Particulas**

```
package vidgrand;
public class Particulas {

    double r;
    double vr;
```

```
    public Particulas()
    {
       r =  0;
       vr = 0;  }
       public void setraio (double pr)
    {
       r = pr ; }
       public double getraio()
    {
       return r; }
       public void setvr (double pvr)
    {
       vr = pvr; }
       public double getvr()
    {
       return vr;}}
```

**Classe LGB_Grafico**

```
package vidgrand;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.axis.NumberAxis;
import org.jfree.chart.axis.ValueAxis;
import org.jfree.chart.plot.XYPlot;
import org.jfree.chart.renderer.xy.StandardXYItemRenderer;
import org.jfree.data.time.Hour;
import org.jfree.data.time.TimeSeries;
import org.jfree.data.time.TimeSeriesCollection;
import org.jfree.data.xy.XYSeries;
import org.jfree.data.xy.XYSeriesCollection;
public class LGB_Grafico extends javax.swing.JDialog {
    public LGB_Grafico(java.awt.Frame parent, boolean modal) {
       super(parent, modal);
       initComponents();}
    public void PlotarGrafico(String titulo, String eixoX,String eixoY,double[] densidade, double[] tempo )
    {
       XYSeriesCollection dataset = new XYSeriesCollection();
       XYSeries series = new XYSeries(titulo);
       for (int j = 0; j < densidade.length; j++) {
          series.add(tempo[j], densidade[j]);          }
          dataset.addSeries(series);
         ValueAxis axis1 = new NumberAxis(eixoX);
       ValueAxis axis2 = new NumberAxis(eixoY);
         StandardXYItemRenderer renderer = new StandardXYItemRenderer();
         XYPlot plot = new XYPlot(dataset, axis1, axis2, renderer);
         JFreeChart chart = new JFreeChart(plot);
         ChartPanel panel = new ChartPanel(chart);
       setSize(800, 600);
       getContentPane().add(panel);
    }
    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

       setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
       pack();
```

```
    }// </editor-fold>
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                LGB_Grafico dialog = new LGB_Grafico(new javax.swing.JFrame(), true);
                dialog.addWindowListener(new java.awt.event.WindowAdapter() {
                    public void windowClosing(java.awt.event.WindowEvent e) {
                        System.exit(0);}});
                dialog.setVisible(true);}});}}
```

**Classe LGB_Calculos**

```
package vidgrand;
import java.io.*;
import javax.swing.*;
public class LGB_Calculos extends LGB_Dados  {
   Particulas [] particulas;
   double [] t08;
   double [] densidade;
   double [] ao;
   double [] er;
   double [] F;
   double [] ka0;
   double [] ko;
   double [] auxf1;
   double [] auxf2;
   double [] auxms1;
   String [] x;
   String [] x1;
   double [] densidade1;
   double [] densidade2;
ptemperatura)
   public LGB_Calculos() throws IOException
   {
      // Valores para testes
      int numero_particulas = 1; // Alterar para classes de tamanho era 31
      double po = 0;
      //double ni = 61400000.;
      double ni=0; // visco
      //double ni = 559578339.3;
      double gama = 0.371;
      double t = 0.0000;
      //double Ev = 83700;
      //double R = 8.31;
      double T0 = 750.88;
      double var_tempo =0.;  //interacao
      int segundos = 0;
      double []v_tempo;
      //double noo = 0.018909;
      double K_tempo=0;
      double T = 0;
      //double ax, ax1, ax2 = 0.;
      //double ax2er =0.;
      //double ax2er1 = 0.;
      double Ns = Math.pow(10, 11);  //densidade dos nucleos
      double U = Math.pow(10, -9);  //taxa de crescimento dos cristais
      double t8 =0;
      //relacionado cluster
      double densidadeteste = 0;
      double pa = 0;
      String name = JOptionPane.showInputDialog("Digite seu nome e data");
```

```
      String nome = JOptionPane.showInputDialog(" digite o nome do arquivo com a extensão (exemplo =
teste.txt), o arquivo estara na pasta onde esta o software");
       File outputFile = new File (nome);
       FileWriter out =new FileWriter(outputFile);
      //nova viscosidade VFT
      double A = 0.00005321;
      double B = 9120.885;
      int m = 0;
      int n= 0;
      double V = 0;
      particulas = new Particulas[numero_particulas];
     for(int i=0;i<numero_particulas;i++)
      {particulas[i] = new Particulas();}

      particulas[0].setraio(0.00005);
      particulas[0].setvr(1);
      //entrada de dados via usuario
      String tempo1 = JOptionPane.showInputDialog("[t] Digite o tempo de simulação [s]");
      segundos = Integer.parseInt(tempo1);
      String var_temperatura = JOptionPane.showInputDialog("[T] Digite a temperatura em [K]");
      T = Double.parseDouble(var_temperatura);
      String densidadeinicial1 = JOptionPane.showInputDialog("Digite o valor da densidade relativa
inicial [po]");
      po= Double.parseDouble(densidadeinicial1);
      String gama1 = JOptionPane.showInputDialog(" [gama] Digite o valor da Tensão Superficial
[J/m^2]");
      gama= Double.parseDouble(gama1);
      /*
      String NS1 = JOptionPane.showInputDialog(" [Ns] Digite o valor do numero de sites por area [m^-
2]");
      Ns= Double.parseDouble(NS1);
      String U1 = JOptionPane.showInputDialog(" [u] Digite o valor da taxa de crescimento de cristal
linear [m/s]");
      U= Double.parseDouble(U1);
            n = 10;
      m = n-1;
      System.out.println(" m     " + m);
      v_tempo = new double[segundos];
      // Criação de t08
      t08 = new double[numero_particulas];
      ao = new double[numero_particulas];
      ko = new double[numero_particulas];
      ka0 = new double[1001];
      auxf1=new double[numero_particulas];
      auxf2=new double[numero_particulas];
      auxms1=new double[numero_particulas];
      x = new String[segundos];
      x1= new String[segundos];
      F = new double[2*n];
      double tmenor = 10000000000000000.;
      System.out.println(">>>>>calculo da viscosidade<<<<<<<");
      ni = A*Math.exp(B/(T-T0));
      System.out.println("Viscosidade: " + ni);
      //para calculo do vetor de a0
      for(int a1=1; a1<=1000; a1++)
          {
            ka0[a1]=a1*0.001;
            //System.out.println(k0t[a1] +"  " + a1);
          }
      //calculos utilizados no programa para simplificar codigo
```

```
        System.out.println(" >>>>>>>>>>    Calculos auxiliares de f(i) <<<<<<<<<<<  " );
        for (int u=0; u<numero_particulas; u++)
        {
             auxf1[u]=((9*gama*po)/(8*ni*particulas[u].getraio()));
            System.out.println("auxf1      "  +  auxf1[u]    );
             auxf2[u] = ((3*gama)/(8*ni*particulas[u].getraio()));
            System.out.println("auxf2      "  +  auxf2[u]    );
            //comsiderando a0 como aproximadamente r
             auxms1[u] = ((3*gama)/(2*ni*particulas[u].getraio()));
            System.out.println("auxMS1      "  +  auxms1[u]    );
        }
            //precisa analise para implementar distribuicao
        //sempre inserir valor de raio maior por ultimo
            double auxf3 = (-Math.PI*Ns*U*U);
            System.out.println("auxf3      "  +  auxf3    );
            //inicializacao valores de a0
        double func1 = 0.;
        double func = 0.;
        double a0_1 = 0.;
        double a0_2 = 0.;
        double a0_3 = 0.;
        double k0 = 0;
        double auxa01 = 0.;
        double auxa02 = 0.;
        auxa01 = (1-po)*((1.5*gama)/ni);
        auxa02 = -1.5*gama/ni;
       pa = 0.8-po;
  //    System.out.println(">>>>>calculo de t08 e seu menor valor<<<<<<<");
      double t081 = 0;
         for(int i=0;i<numero_particulas;i++)
     {
           //aproximacao...4 vezes o valor do t08 padrao
           double tempo_total = 4 * (2.666666667*((Math.pow(0.8, 0.3333333)-Math.pow(po,
0.3333333))/Math.pow(0.8, 0.3333333))*(ni/gama)*particulas[i].getraio());
        for ( t8=0; t8<tempo_total; t8++ ) {
         double t_081 = auxf1[i];
         double t_082 = auxf1[i]*(1-auxf2[i]*(t8/2)) * Math.exp(auxf3*(t8/2)*(t8/2));
         double t_083 = auxf1[i]*(1-(auxf2[i]*t8))*Math.exp(auxf3*t8*t8);
         t081 = (t8/6)*(t_081 + t_083 + 4 * t_082);
         if (t081 <= pa) {
         t08[i]=t8;  }}
       //calculo de a0
        for (int a=0 ; a<=1000;a++) {
         a0_1 = auxa01/(particulas[i].getraio()*ka0[a]);
         a0_2 =
(auxa01/(particulas[i].getraio()*ka0[a]))*Math.exp((auxa02*(t08[i])/2)/(particulas[i].getraio()*ka0[a])*M
ath.exp(auxf3*(t08[i])/2*(t08[i])/2));
         a0_3 =
(auxa01/(particulas[i].getraio()*ka0[a]))*Math.exp((auxa02*(t08[i]))/(particulas[i].getraio()*ka0[a])*Mat
h.exp(auxf3*(t08[i])*(t08[i])));
         func1 = (t08[i]/6)*(a0_1 + a0_3 + 4 * a0_2);
         //System.out.println(">>>>>>>>>integral para ao<<<<<<<<<<" + func1);
         func =  func1-pa;
         if (func > 0) {
         k0 = a * 0.001;
         }
         }
         ko[i] = k0;
        }
      K_tempo = (1/auxf2[0]);
```

```
   System.out.println("********Valor   tempo8 " + t08[0]);
   System.out.println("ultimo valor de ka0   " + ko[0]);
        //  System.out.println("********Valor t08["+0 + "]: "+ t08[0]);
        //  System.out.println("Valor do raio: " + particulas[i].getraio());
          //calculo do menor valor de t08
          //verificar tmenor e imprime onde localiza
        /*
          if(tmenor> t08[i]){
            tmenor = t08[i];
            System.out.println(">>tmenor :<< " + tmenor );
            //calculo er
          }
      //criar valores vetor ka0
//for(int i=0;i<numero_particulas;i++){
//}
       //calculo do var_tempo valor relacionado com a variacao da iteracao dos calculos
       densidade = new double[segundos];
       densidade1= new double[segundos];
       densidade2= new double[segundos];
       //calculos auxiliares
                // calculo de t08
                // esse calculo sera executado para a determinacao de t08...
                //e usara as iteracoes para determinacao do mesmo
      var_tempo = Math.pow(auxf2[0], -1)/K_tempo ;
//var_tempo = Math.pow(auxf2[0], -1)/10000 ;
       V = var_tempo/n;
       System.out.println("     Var_tempo  " + var_tempo + "  variacao da iteracao de f(i)   " + V );
       //mt usada para contar as iteracoes v sem uso ate o momento
       int j=0;
       int z=0;
       int mt = 0;
       int v=0;
       int l=0;
       //inicializacao e calculos referentes a integracao
       //m2 = valor de m dividido por dois e h inverso de m
       double m2 = 0;
       double h = 0;
       m2 = m/2;
       //double t_break=0;
//cluster
        //densidade2[0]=0;
        System.out.println(">>>>>>>>>>inicio dos calculos da densidade<<<<<<<<<" );
       for(int i=0;i<segundos;i++)
       {
         //densidade[i] = densidade[i];
          for(z=0;z<t08.length;z++)
          {
           //inicializacao dos f(i)
//implementacao do metodo de clusters
                //Frenkel
          if (t<= t08[z]) {
                for(j=0;j<=m ;j++)
                {  F[j] =0;
{                  F[j]= (auxf1[z]*Math.pow(1-auxf2[z]*(t+j*V), -
4)*Math.exp(auxf3*(t+j*V)*(t+j*V)))*particulas[z].getvr();}}}
//implementacao do metodo de clusters
                //MS
          else
          {               for(j=0;j<=m ;j++)
               {                  F[j] =0;
```

```
                {                   F[j]= ((1-po)*(auxms1[z]/ko[z])*Math.exp((-
auxms1[z]/ko[z])*(t+j*V))*Math.exp(auxf3*(t+j*V)*(t+j*V)))*particulas[z].getvr();}}}
                        //somatorios
               //soma impar
               double soma_impar =0;
                for (v=0; v<=(m2-1);v++)
               {
                  soma_impar+=F[1+2*v];                }
              //somar indices pares
               double soma_par = 0;
               if (n>=2)
                  for (v=1; v<=(m2-1);v++)
               {
                soma_par+=F[2*v];
                 //calculo da densidade de frenkel pelo metodo de Simpson
}
               else
                  soma_par = 0;
              // System.out.println(">>>>soma_impar<<<<" + soma_impar );
              // System.out.println(">>>>>>soma_par<<<<<<"+soma_par);
               //segunda mudanca
               //densidade1[i] +=po + auxf1*((V/3)*(F[0] + F[m] + 4*(soma_impar) +2 * (soma_par)) *
particulas[z].getvr()); }
         t += (double) var_tempo;
         v_tempo[i] = t;
         mt = mt + 1;
         //System.out.println(mt + "     densidade1  " + densidade1[i] +  "     tempo  " + v_tempo[i] );
         //System.out.println(mt + "     densidade  " + densidade[i] +  "     tempo  " + v_tempo[i] );
             for (l=1; l<=(i);l++)
         {
//modificacao
            densidade2[i] = densidade2[l-1] + densidade1[l] ;           }
        densidadeteste=densidade2[i];
      }
      //somar po para densidade 2
       out.write( "Vidgrand V.1 log..Bem vindos!" +  System.getProperty("line.separator") );
       out.write( "Esse software calcula a cinetica de sinterizacao de materiais vítreos usando o modelo de
Clusters " + System.getProperty("line.separator") );
       out.write( "Para maiores informações: lgrando@gmail.com e ebferreira@feg.unesp.br" +
System.getProperty("line.separator") );
       out.write( "UNESP - CAMPUS DE GUARATINGUERÁ - (DMT)"  +
System.getProperty("line.separator") );
       out.write( System.getProperty("line.separator") );
       out.write( "Operador e data:  " + name + System.getProperty("line.separator") );
       out.write( "densidade relativa e tempo em segundos (s) " + System.getProperty("line.separator") );
       out.write( "numero de classes de particulas  " + numero_particulas +
System.getProperty("line.separator") );
       for(int p=0; p<numero_particulas;p++)
       {
       out.write( "Particula " + String.valueOf(p) + " raio  " + String.valueOf(particulas[p].getraio())  +
System.getProperty("line.separator") );
       //out.write( "Vr " + particulas[p].setvr  + System.getProperty("line.separator") );
      out.write( "Particula " + String.valueOf(p) + " Vr  " + String.valueOf(particulas[p].getvr())  +
System.getProperty("line.separator") );

      out.write( System.getProperty("line.separator") );
       }
out.write("     densidade  " + System.getProperty("line.separator"));
       for(int i=0;i<segundos;i++)
     {
```

```
        densidade[i] = densidade2[i] + po;
      System.out.println( densidade[i] +  "     tempo  " + v_tempo[i] );
       x[i] = String.valueOf(densidade[i]);
      out.write( x[i]  +  System.getProperty ("line.separator"));
      }
      out.close();
       LGB_Grafico grafico = new LGB_Grafico(null,true);
       grafico.setTitle("Gráfico ('Densidade x Tempo')");
       grafico.PlotarGrafico("Variação densidade x tempo [s]", "Tempo [s]", "Densidade relativa",
densidade, v_tempo );
       grafico.setLocationRelativeTo(null);
       grafico.setVisible(true);
System.exit(0);}}
```