

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA

CAIO GUIMARÃES MAIOLI

**CONVERSÃO TERMOQUÍMICA EM LEITO FIXO:
CARACTERÍSTICAS, MODELAGEM E
IMPLEMENTAÇÃO DO SOLVER
*BIOMASSGASIFICATIONFOAM***

VITÓRIA, ES
2016

CAIO GUIMARÃES MAIOLI

**CONVERSÃO TERMOQUÍMICA EM LEITO FIXO:
CARACTERÍSTICAS, MODELAGEM E
IMPLEMENTAÇÃO DO SOLVER
*BIOMASSGASIFICATIONFOAM***

Dissertação de mestrado apresentada ao Programa de Pós Graduação em Engenharia Mecânica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para a obtenção do título de Mestre em Engenharia Mecânica.

Orientador: Prof. Dr. Márcio Ferreira Martins

VITÓRIA, ES

2016

Agradecimentos

Agradeço aos meus pais, meu irmão Davi e demais amigos e familiares, que me acompanharam por toda esta etapa, dando muita força e incentivo.

Agradeço enormemente ao meu irmão Artur, cujas trocas de conhecimento, ideias e experiências tornaram possíveis a conclusão deste trabalho.

Ao meu orientador Prof. Dr. Márcio Ferreira Martins, pelas orientações, conselhos e apoio.

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - CAPES - pelo apoio financeiro prestado.

Resumo

Os processos de conversão termoquímica são meios para converter os sólidos em portadores de energia mais convenientes (carvão), ou transformá-los em combustíveis líquidos e gasosos, ou ainda em calor ou outros produtos diversos. Muitos estudos vêm sendo realizados com o objetivo de tornar os processos termoquímicos mais eficientes e economicamente viáveis. A aplicação dos modelos de Dinâmica dos Fluidos Computacional (CFD) ajuda a otimizar o projeto e a operação dos reatores termoquímicos, como os de leito fixo. Os softwares de CFD gratuitos oferecem uma ferramenta econômica para a realização desses estudos. Um desses softwares que vem ganhando destaque é o OpenFOAM. Foi publicado recentemente um solver para o OpenFOAM chamado `biomassGasificationFoam`, para a simulação da gaseificação de biomassa em leito fixo. Neste trabalho foi feita uma avaliação do solver `biomassGasificationFoam` e das bibliotecas auxiliares que o acompanham, para generalizar a sua aplicabilidade em diversos processos de conversão termoquímica de sólidos em leito fixo. Seus modelos de transporte de calor e massa em um meio poroso reativo foram verificados e validados com testes propostos, equações analíticas e experimentos. Detalhes do código foram alterados e melhorias foram implementadas. Identificou-se uma superestimação do processo de difusão das espécies gasosas, causada por uma hipótese adotada na equação de conservação dessas espécies. O modelo cinético se mostrou adequado para processos de decomposição térmica do sólido onde a ordem da reação é unitária. Porém, esse modelo não se mostrou tão adequado para reações homogêneas do sólido onde a ordem da reação é diferente de um e para reações heterogêneas. No conjunto, o `biomassGasificationFoam` apresenta um potencial interessante, uma ferramenta útil para a simulação de diversos processos de conversão termoquímica de sólidos em leito fixo. Por fim, foram fornecidas informações sobre o *software* OpenFOAM e, principalmente, sobre o `biomassGasificationFoam`, com o intuito de servir como documentação para auxiliar futuros usuários interessados no seu uso.

Palavras-chaves: conversão termoquímica, OpenFOAM, `biomassGasificationFoam`, cinética química.

Abstract

The thermochemical conversion processes are ways to convert solids into more convenient energy carriers (charcoal), or turn them into liquid and gaseous fuels, or also heat or other diverse products. Many studies have been conducted aiming to make the thermochemical processes more efficient and economically viable. The application of Computational Fluid Dynamics (CFD) models helps to optimize the design and operation of the thermochemical reactors, such as fixed bed reactors. The free CFD softwares offer a cost-effective tool for these studies. One such software that is gaining prominence is OpenFOAM. An solver for OpenFOAM called biomassGasificationFoam was recently published for the simulation of biomass gasification in fixed bed. In this work, it was done an evaluation of the solver biomassGasificationFoam and the auxiliary libraries that accompany it, in order to generalize their applicability in various solids thermochemical conversion processes in fixed bed. Its models for heat and mass transport in a reactive porous medium were verified and validated with proposed tests, analytical equations and experiments. Some code details were changed and improvements were implemented. It was identified an overestimation of the diffusion process of the gaseous species caused by a hypothesis adopted in these species conservation equation. The kinetic model showed to be suitable for solid thermal decomposition processes where the reaction order is unitary. However, this model was not so suitable for solid homogeneous reactions where the reaction order is different from one and for heterogeneous reactions. In overall, the biomassGasificationFoam presents an interesting potential, an useful tool for the simulation of many solids thermochemical conversion processes in fixed bed. Finally, informations were provided about the software OpenFOAM, and, especially, about the biomassGasificationFoam, in order to serve as documentation to help future users interested in its use.

Key- Words: *thermochemical conversion, OpenFOAM, biomassGasificationFoam, chemical kinetics.*

Lista de ilustrações

Figura 1 – Etapas da conversão termoquímica de sólidos	25
Figura 2 – Zonas de temperatura ou regimes de reação para a conversão de sólidos	40
Figura 3 – Reator TGA simplificado de geometria cúbica com um amostra hexagonal de madeira no centro	50
Figura 4 – Comparação da perda de massa durante o experimento TGA e a simulação correspondente	51
Figura 5 – Taxa de perda de massa em cinco pontos da amostra	51
Figura 6 – Variação na massa dos componentes do sólido	52
Figura 7 – Fração mássica dos gases	52
Figura 8 – Massa normalizada de partícula de palha e madeira durante a combustão	53
Figura 9 – Massa dos componentes sólidos durante a combustão de madeira	54
Figura 10 – Massa dos componentes sólidos durante a combustão de palha	54
Figura 11 – Liberação de CH_4 durante a combustão	55
Figura 12 – Malha correspondente à câmara de combustão, com a biomassa representada em azul	55
Figura 13 – Variação da temperatura do sólido em uma simulação na qual a ignição é desativada aos 800 s	56
Figura 14 – Variação na temperatura da biomassa com o uso da resistência de aquecimento	57
Figura 15 – Variação da fração mássica de metano com o uso da resistência de aquecimento	58
Figura 16 – Variação da massa de carvão com o uso da resistência de aquecimento	58
Figura 17 – Estrutura de um caso no OpenFOAM	63
Figura 18 – Geometria do caso <i>cavity</i>	65
Figura 19 – Numeração do bloco para geração da malha do caso <i>cavity</i>	65
Figura 20 – <i>blockMeshDict</i> para o caso <i>cavity</i>	66
Figura 21 – Arquivo <i>U</i> do tutorial <i>cavity</i>	69
Figura 22 – Arquivo <i>transportProperties</i> para o tutorial <i>cavity</i>	71
Figura 23 – Arquivo <i>controlDict</i> para o tutorial <i>cavity</i>	72
Figura 24 – Arquivo <i>fvSchemes</i> para o tutorial <i>cavity</i>	74
Figura 25 – Exemplo de um arquivo <i>fvSolution</i>	76
Figura 26 – Exemplo de um arquivo <i>decomposeParDict</i>	81
Figura 27 – Janela do ParaView	82
Figura 28 – Campo de pressão para o tutorial <i>cavity</i>	83
Figura 29 – Campo de velocidades para o tutorial <i>cavity</i>	83
Figura 30 – Exemplo de um arquivo <i>probesDict</i>	84

Figura 31 – Exemplo de um arquivo <i>sampleDict</i>	85
Figura 32 – Estrutura do código <i>biomassGasificationFoam</i> : <i>solver</i> e bibliotecas	87
Figura 33 – Arquivo <i>exportLibSrc</i>	91
Figura 34 – Diretório <i>biomassGasificationFoam_installPack_1.0</i> com o arquivo <i>exportLibSrc</i>	91
Figura 35 – Trecho final do arquivo <i>.bashrc</i> com as linhas acrescentadas destacadas	91
Figura 36 – Erro devido à permissão do arquivo <i>Allwmake</i>	92
Figura 37 – Erro ao compilar o conjunto de bibliotecas <i>solid</i> , dos modelos termofísicos	92
Figura 38 – Arquivo <i>files</i> alterado	93
Figura 39 – Esquema do <i>loop</i> do <i>solver biomassGasificationFoam</i>	102
Figura 40 – Arquivo <i>thermophysicalProperties</i> do caso <i>TGA_test</i>	111
Figura 41 – Arquivo <i>chem.inp.frag</i> do caso <i>oneCellTest</i>	112
Figura 42 – Arquivo <i>thermophysicalProperties</i> do caso tutorial do <i>reactingFoam</i>	113
Figura 43 – Arquivo <i>reactions</i> do caso tutorial do <i>reactingFoam</i>	114
Figura 44 – Trecho do arquivo <i>chemistryProperties</i> , mostrando a variável <i>solidReactionEnergyFromEnthalpy</i>	115
Figura 45 – Arquivo <i>heatTransferProperties</i> do caso <i>TGA_test</i>	119
Figura 46 – Trecho do arquivo <i>solidThermophysicalProperties</i> do caso <i>TGA_test</i>	121
Figura 47 – Trecho do arquivo <i>therm.dat</i> do caso <i>TGA_test</i> , destacando os parâmetros para o CO_2 e para o CO	125
Figura 48 – Trecho do arquivo <i>chemistryProperties</i> do caso <i>TGA_test</i> , demonstrando os parâmetros para o H_2O	126
Figura 49 – Trecho do arquivo <i>YEqn.H</i> modificado	130
Figura 50 – Comparação dos resultados para a difusão obtidos com o uso do coeficiente de difusão efetivo D^* e com a viscosidade dinâmica μ para uma temperatura de 300 K	132
Figura 51 – Comparação dos resultados para a difusão obtidos com o uso do coeficiente de difusão efetivo D^* e com a viscosidade dinâmica μ para uma temperatura de 1000 K	133
Figura 52 – Comparação dos resultados para a difusão obtidos com o uso do coeficiente de difusão D_{AB} e com a viscosidade dinâmica μ para uma temperatura de 300 K	134
Figura 53 – Comparação dos resultados para a difusão obtidos com o uso do coeficiente de difusão D_{AB} e com a viscosidade dinâmica μ para uma temperatura de 1000 K	134
Figura 54 – Frações mássicas para a simulação da advecção para uma malha com 100 divisões	136
Figura 55 – Frações mássicas para a simulação da advecção para uma malha com 1000 divisões	136

Figura 56 – Croqui da célula de combustão de Martins (2008)	137
Figura 57 – Perfis de temperatura para $\alpha = 1,3 \text{ W m}^{-2} \text{ K}^{-1}$	139
Figura 58 – Perfis de temperatura para $\alpha = 0,1 \text{ W m}^{-2} \text{ K}^{-1}$	139
Figura 59 – Perfis de temperatura para $\alpha = 10 \text{ W m}^{-2} \text{ K}^{-1}$	140
Figura 60 – Evolução da temperatura do sólido para o código original	145
Figura 61 – Evolução da temperatura do sólido para o código com a modificação proposta	145
Figura 62 – Evolução das frações mássicas para os três Códigos diferentes	159
Figura 63 – Comparação da evolução das frações mássicas para o Código Original e o encontrado por Hared et al. (2007) para uma taxa de aquecimento de $10 \text{ }^\circ\text{C/min}$	160
Figura 64 – Comparação da evolução das frações mássicas encontradas pelo Código Original e por Zanoni, Massard e Martins (2012)	164
Figura 65 – Comparação da evolução das frações mássicas encontradas pelo Código Alterado e por Zanoni, Massard e Martins (2012)	165
Figura 66 – Evolução das frações mássicas para a combustão do xisto betuminoso obtidas com os três Códigos diferentes	166

Lista de tabelas

Tabela 1 – Características dos processos de pirólise	22
Tabela 2 – Unidades básicas para definir as dimensões no OpenFOAM	70
Tabela 3 – Principais sub-dicionários utilizados no <i>fvSchemes</i>	73
Tabela 4 – Alguns <i>solvers</i> fornecidos com o OpenFOAM	79
Tabela 5 – Parâmetros para a simulação da difusão do CO ₂	132
Tabela 6 – Parâmetros para a simulação da advecção	135
Tabela 7 – Parâmetros para a simulação da transferência de calor	138
Tabela 8 – Parâmetros cinéticos para a pirólise de uma madeira impregnada com ácido fosfórico	158
Tabela 9 – Composição inicial e coeficientes estequiométricos para a combustão do xisto betuminoso	162
Tabela 10 – Parâmetros cinéticos para a combustão do xisto betuminoso obtidos com uma taxa de aquecimento de 3 K/min	163

Lista de abreviaturas e siglas

CFD	Dinâmica dos Fluidos Computacional (<i>Computational Fluid Dynamics</i>)
IMFT	Instituto de Mecânica dos Fluidos de Toulouse
CVFEM	<i>Control Volume Finite Element Method</i>
PET	Polietileno tereftalato
PEAD	Polietileno de alta densidade
PEBD	Polietileno de baixa densidade
PVC	Policloreto de polivinila
PP	Polipropileno
PS	Poliestireno
DAEM	<i>Distributed Activation Energy Model</i>
FG-DVC	<i>Functional Group - Depolymerization, Vaporization, Cross Linking</i>
CPD	<i>Chemical Percolation Devolatilization</i>
NMR	<i>Nuclear Magnetic Resonance</i>
FTIR	<i>Fourier Transform Infra-Red</i>
SI	Sistema Internacional
USCS	<i>United States Customary System</i>
RAS	<i>Reynolds-averaged stress</i>
LES	<i>Large-eddy simulation</i>
TVD	<i>Total Variation Diminishing</i>
NV	<i>Normalised Variable</i>
PISO	<i>Pressure-Implicit Split-Operator</i>
SIMPLE	<i>Semi-Implicit Method for Pressure-Linked Equations</i>
MPI	<i>Message Passing Interface</i>
EDO	Equação Diferencial Ordinária
HTC	<i>Heat Transfer Coefficient</i>

Lista de Símbolos

Nomenclatura

A	Fator pré-exponencial [s^{-1}]
C_p	Calor específico [$J\ kg^{-1}\ K^{-1}$]
D_{AB}	Coefficiente de difusão binária [$m^2\ s^{-1}$]
D^*	Coefficiente de difusão mássica efetivo [$m^2\ s^{-1}$]
E	Energia de ativação [$kJ\ mol^{-1}$]
h_f	Entalpia de formação [$J\ kg^{-1}$]
H_r	Energia das reações química [$W\ m^{-3}$]
h_r	Calor de reação [$J\ kg^{-1}$]
h_s	Entalpia sensível [$J\ kg^{-1}$]
k	Condutividade térmica [$W\ m^{-1}\ K^{-1}$] Taxa de reação [s^{-1}]
K	Tensor de resistência viscosa [m^{-2}]
c	Massa das espécies [kg]
W	Massa molar [$kg\ mol^{-1}$]
p	Pressão [Pa]
S^{rad}	Energia de radiação [$W\ m^{-3}$]
T_a	Temperatura de ativação [K]
R	Taxa de consumo ou produção das espécies [$kg\ m^{-3}\ s^{-1}$]
t	Tempo [s]
u	Velocidade [$m\ s^{-1}$]
Y	Fração mássica

Letras gregas

α	Coefficiente de transferência de calor [$W\ m^{-2}\ K^{-1}$]
----------	--

Γ	Energia necessária para aquecer ou resfriar os gases produzidos nas reações do sólido para a temperatura dos gases [W m^{-3}]
Σ	Área superficial específica dos poros [$\text{m}^2 \text{m}^{-3}$]
γ	Fração de vazio
μ	Viscosidade dinâmica [Pa s]
Ω	Taxa de reação [$\text{kg m}^{-3} \text{s}^{-1}$]
ρ	Massa específica [kg m^{-3}]
ν	Viscosidade cinemática [$\text{m}^2 \text{s}^{-1}$]
ω	Taxa de produção ou consumo das espécies referente às reações homogêneas na fase gasosa [$\text{kg m}^{-3} \text{s}^{-1}$]
ξ	Razão mássica entre produtos e substratos sólidos

Sobrescritos

<i>comb</i>	Combustão
<i>evap</i>	Evaporação
<i>gaseif</i>	Gaseificação
G	Gás
<i>pir</i>	Pirólise
S	Sólido

Sumário

1	INTRODUÇÃO	15
2	REVISÃO BIBLIOGRÁFICA	18
2.1	A Conversão Termoquímica de Sólidos	18
2.1.1	Craqueamento térmico	18
2.1.1.1	Pirólise	19
2.1.1.2	Liquefação	20
2.1.1.3	Torrefação	20
2.1.1.4	Considerações	21
2.1.2	Processos heterogêneos de oxirredução	22
2.1.2.1	Combustão	23
2.1.2.2	Gaseificação	23
2.2	Modelos de Conversão de Massa	24
2.2.1	Secagem	24
2.2.1.1	Modelo de temperatura constante	25
2.2.1.2	Modelo algébrico	26
2.2.1.3	Modelo cinético	27
2.2.1.4	Modelo de transporte	27
2.2.2	Pirólise	30
2.2.2.1	Modelos cinéticos	30
2.2.2.1.1	Modelo global de taxa única	31
2.2.2.1.2	Modelos de reações múltiplas de simples estágio	32
2.2.2.1.3	Modelos semi-globais de dois estágios	34
2.2.2.1.4	Modelo de energia de ativação distribuída – DAEM	35
2.2.2.2	Modelos de rede	37
2.2.3	Reações Heterogêneas	38
2.3	Estudos em CFD da Conversão Termoquímica de Combustíveis Sólidos	44
2.4	Introdução ao <i>solver</i> biomassGasificationFoam	48
3	O SOFTWARE OPENFOAM	59
3.1	Escolha do OpenFOAM como pacote CFD	60
3.2	Estrutura de um caso	62
3.3	Pré-processamento	63
3.3.1	Geração da malha	64
3.3.1.1	blockMesh	64

3.3.1.2	snappyHexMesh	67
3.3.1.3	Importação de malhas	67
3.3.1.4	checkMesh	68
3.3.2	Definindo o caso	68
3.3.2.1	Condições iniciais e de contorno	68
3.3.2.2	Propriedades físicas	70
3.3.2.3	Controle do tempo e da escrita dos dados (<i>controlDict</i>)	71
3.3.2.4	Esquemas numéricos (<i>fvSchemes</i>)	73
3.3.2.5	Controle da solução (<i>fvSolution</i>)	76
3.4	Solvers	78
3.4.1	Paralelização	80
3.5	Pós-Processamento	82
4	BIOMASSGASIFICATIONFOAM: CARACTERÍSTICAS, MODELAGEM E IMPLEMENTAÇÃO	86
4.1	Introdução	86
4.2	Instalação	88
4.3	Campo de Porosidade Dinâmico	93
4.4	Hipóteses da Modelagem	96
4.5	Equações de Conservação	97
4.5.1	Conservação do Momento	97
4.5.2	Continuidade	98
4.5.3	Conservação das Espécies para a Fase Sólida	98
4.5.4	Conservação das Espécies para a Fase Gasosa	98
4.5.5	Conservação da Energia para a Fase Sólida	99
4.5.6	Conservação da Energia para a Fase Gasosa	100
4.6	Implementação do Solver biomassGasificationFoam	100
4.7	biomassGasificationMedia: Bibliotecas Auxiliares	104
4.8	Modelos Termofísicos e Termoquímicos	105
4.8.1	Cinética Química e Estequiometria	106
4.8.1.1	Fase Sólida	106
4.8.1.2	Fase Gasosa	111
4.8.2	Calor de Reação	114
4.8.3	Transferência de Calor	116
4.8.4	Radiação	119
4.8.5	Propriedades Físicas da Fase Sólida	120
4.8.6	Propriedades Físicas da Fase Gasosa	123
4.9	heterogeneousPyrolysisModel	126

5	O TRANSPORTE DE CALOR E MASSA EM MEIO POROSO RE- ATIVO	128
5.1	Transporte de Espécies Gasosas	128
5.1.1	Difusão	128
5.1.2	Advecção	134
5.2	Transferência de Calor Entre Gás e Sólido	135
5.3	Modelo Termoquímico das Reações na Fase Sólida	139
5.3.1	Calor de Reação	142
5.3.2	Propostas de Alteração no Código para a Cinética Química	146
5.3.2.1	Ordem da reação com relação ao gás	147
5.3.2.2	ODESolidHeterogeneousChemistryModel	152
5.3.3	Função <i>solve(t0, deltaT)</i> e Função <i>calculate()</i>	155
5.3.4	Testes para a Cinética Química	157
5.3.4.1	Pirólise de um material sólido	157
5.3.4.2	Combustão de um material sólido	161
6	CONSIDERAÇÕES FINAIS	167
	REFERÊNCIAS	169
	APÊNDICES	178
	APÊNDICE A – CLASSE ODESOLIDHETEROGENEOUSCHEMIS- TRYMODEL: MUDANÇAS PROPOSTAS	179
A.1	Função <i>omega(c, T, p, updateC0)</i>	179
A.2	Função <i>omega(R, c, T, p, pf, cf, lRef, pr, cr, rRef)</i>	181
A.3	Função <i>solve(t0, deltaT)</i>	182
A.4	Função <i>jacobian(t, c, dcdt, dfdc)</i>	186
	APÊNDICE B – MUDANÇAS PARA USAR A FUNÇÃO CALCUL- LATE()	191
B.1	Arquivo <i>ODESolidHeterogeneousChemistryModel.C</i> - Função <i>cal- culate()</i>	191
B.2	Arquivo <i>volPyrolysis.C</i> - Função <i>evolveRegion()</i>	193
	ANEXOS	194
	ANEXO A – ARQUIVO BIOMASSGASIFICATIONFOAM.C	195

1 Introdução

Todo sólido que apresente carbono em sua composição possui uma conseqüente energia química armazenada, com enorme potencial para ser transformada em formas de energia essenciais à vida humana, como energia térmica, elétrica e mecânica. Os processos de conversão termoquímica são meios para converter os sólidos em portadores de energia mais convenientes (carvão), ou transformá-los em combustíveis líquidos e gasosos, ou ainda em calor ou outros produtos diversos - produtos químicos, materiais sintetizados de uso final, ou materiais inertes.

Diversos tipos de reatores foram desenvolvidos para realizarem os processos de conversão termoquímica, entre eles os reatores de leito fixo, que encontram grande aplicabilidade pelo mundo. Segundo [Duffy \(2012\)](#), a combustão em leito fixo é o mais simples e comum método de converter a energia química armazenada em biomassa em uma forma mais útil (calor).

Os reatores de leito fixo são caracterizados pela presença de um leito poroso por onde escoar uma mistura gasosa, reativa ou não. Essa interação resulta em complexos processos químicos e de transporte de calor e massa. Nesse âmbito, o desenvolvimento de ferramentas numéricas como a Dinâmica dos Fluidos Computacional (CFD) oferece um meio eficaz de quantificar esses processos sob condições variadas de operação, buscando otimizar o projeto e a operação desses reatores ([WANG; YAN, 2008](#)).

Um pequeno inconveniente surge pelos pacotes CFD disponíveis não apresentarem os modelos que descrevem os complexos processos da conversão termoquímica de sólidos em leito fixo como um recurso integrado, cabendo ao usuário desenvolver e implementar esses modelos. Isso leva à existência de vários códigos pessoais (in-house) na literatura, o que limita o acesso amplo aos mesmos. Implementar os modelos em pacotes CFD existentes reduz o esforço de desenvolvimento e aumenta a acessibilidade.

Nesse meio, o *software* OpenFOAM (Open Field Operation And Manipulation) vem recebendo crescente destaque. Ele é um pacote computacional gratuito, de código aberto, que permite ao usuário o acesso a todo o seu código, possibilitando a implementação de novos modelos e a sua adaptação para a solução de um problema específico. Os códigos desenvolvidos para o OpenFOAM podem ser mais facilmente compartilhados, por meio de sua extensa comunidade.

Recentemente, [Kwiatkowski et al. \(2013\)](#) desenvolveram um *solver* para o OpenFOAM chamado `biomassGasificationFoam`. A proposta inicial é a sua utilização para a simulação da gaseificação de biomassa em leito fixo. Os modelos implementados nesse *solver* demonstram um potencial para a sua aplicação não apenas para processos de gaseificação

de biomassa, mas também para vários outros processos de conversão termoquímica em leito fixo de diferentes materiais sólidos, como pirólise e combustão, e também para qualquer processo envolvendo um meio poroso reativo, como catálise ou filtração. Em teoria, esse *solver* poderia ser aplicado até para processos ocorrendo em um meio poroso não reativo, como queimadores porosos, por exemplo.

Apesar do OpenFOAM se mostrar uma ferramenta poderosa, uma desvantagem no seu uso é a falta de documentação detalhada, que contribui para tornar lenta a sua curva de aprendizado. Ele acaba requerendo um conhecimento ao menos básico da linguagem de programação C++ para uma melhor compreensão de seus códigos fontes.

Perante o contexto descrito acima, os objetivos deste trabalho são:

- Geral
 - Avaliar o *solver biomassGasificationFoam* para generalizar sua aplicabilidade em diversos processos de conversão termoquímica de sólidos em leito fixo.
- Específicos
 - Verificar e validar com testes propostos, equações analíticas e experimentos os modelos de transporte de calor e massa em um meio poroso reativo do *solver biomassGasificationFoam*.
 - Identificar erros e possibilidades de melhorias no *solver biomassGasificationFoam*.
 - Implementar as melhorias no código para viabilizar o uso generalizado do *solver biomassGasificationFoam*.
 - Fornecer informações sobre o *software* OpenFOAM e, principalmente, sobre o *solver biomassGasificationFoam*, com o intuito de servir como documentação para auxiliar futuros usuários interessados no seu uso.

Este trabalho foi dividido em seis capítulos, sendo o primeiro este capítulo introdutório. O Capítulo 2 apresenta uma revisão da literatura sobre os processos de conversão termoquímica de sólidos e os modelos de conversão de massa que descrevem as transformações sofridas pelos sólidos expostos à esses processos. Também apresenta alguns estudos da conversão termoquímica de sólidos baseados na Dinâmica dos Fluidos Computacional (CFD). Por fim, faz uma breve introdução sobre o *solver biomassGasificationFoam*, sua validação realizada pelo seu autor, e comenta sobre os trabalhos que o utilizaram como ferramenta computacional.

O Capítulo 3 se dedica a apresentar o *software* OpenFOAM, descrevendo suas características, vantagens, ferramentas e o seu potencial.

O Capítulo 4 descreve o *solver* `biomassGasificationFoam`, suas características, modelagem e implementação. Nele são fornecidas informações sobre a instalação desse *software*, hipóteses da modelagem, equações de conservação, modelos termofísicos e termoquímicos, e sobre como suas bibliotecas são implementadas.

O Capítulo 5 apresenta os testes realizados com o intuito de avaliar e validar os modelos de transporte de calor e massa em meio poroso reativo implementados no `biomassGasificationFoam`. São analisados o transporte das espécies da fase gasosa, tanto difusivo quanto advectivo; a transferência de calor entre as fases sólida e gasosa e a sua influência na temperatura do leito; e o modelo termoquímico das reações em fase sólida, a implementação do calor de reação e da cinética química.

O Capítulo 6 apresenta as considerações finais deste trabalho e as sugestões para trabalhos futuros.

2 Revisão Bibliográfica

2.1 A Conversão Termoquímica de Sólidos

Sólidos que apresentam carbono e matéria orgânica em sua composição possuem uma energia química que pode ser transformada em formas mais relevantes, como energia térmica e elétrica. Para tal, é necessário aplicar técnicas para convertê-los em portadores de energia mais apropriados, como combustíveis líquidos, gasosos e sólidos, ou, em alguns casos, converter a energia química diretamente em calor. Isso pode ser feito por dois caminhos principais, que são a conversão bioquímica e a conversão termoquímica (BASU, 2013).

Na conversão bioquímica, as moléculas de biomassa são quebradas em moléculas menores por enzimas ou bactérias. Esse processo não requer muita energia externa, porém é bem mais lento que a conversão termoquímica. Os principais caminhos para a conversão bioquímica são a fermentação, digestão aeróbica e anaeróbica, e hidrólise enzimática ou ácida.

Os processos de conversão termoquímica transformam o sólido por meio de tratamentos térmicos. Baseado em seu mecanismo de atuação, eles podem ser divididos em duas vertentes. A primeira seriam os processos por craqueamento térmico, onde a ação pura de energia térmica, sem a ação de agentes oxidantes, causa a degradação do material orgânico. Nesse ramo se encontram a pirólise, a liquefação e a torrefação. A segunda vertente seriam os processos nos quais ocorrem reações heterogêneas de oxidação e redução entre o sólido e gases como O_2 , CO_2 , H_2 e H_2O . Nesse ramo se encontram os processos de combustão e gaseificação. Segundo Grønli (1996), os principais produtos dos processos de conversão termoquímica podem ser gases, líquidos, carvão (sólido) ou calor, dependendo do método empregado. A seguir, serão ressaltadas as principais características de cada método.

2.1.1 Craqueamento térmico

O fornecimento de energia térmica, com conseqüente aumento de temperatura, atua nas ligações químicas das cadeias carbônicas, causando o seu rompimento e produzindo materiais de menor peso molecular. Os processos envolvendo craqueamento térmico são muito aplicados e estudados para a conversão de biomassa. Porém, deve-se salientar que também encontram aplicações para outros sólidos, como carvão mineral e plásticos como polietileno e poliestireno. A seguir serão comentados os processos de pirólise, liquefação e torrefação.

2.1.1.1 Pirólise

A pirólise é a decomposição térmica do combustível sólido, liberando alcatrão (hidrocarbonetos de alto peso molecular) e gases leves não condensáveis, ficando a fase condensada com excesso de carbono, normalmente referenciado como coque (CUNHA, 2010). Segundo Basu (2013), na pirólise, moléculas grandes de hidrocarbonetos da biomassa são quebradas em moléculas menores. A pirólise rápida produz principalmente combustível líquido, conhecido como bio-óleo, enquanto a pirólise lenta produz gás e carvão sólido. A pirólise se mostra promissora para a conversão de biomassa residual em combustíveis líquidos úteis. De acordo com Di Blasi (2008), vários fatores afetam a taxa de pirólise e as proporções, composição e propriedades dos tipos de produtos. Temperatura, pressão e taxa de aquecimento são os principais parâmetros operacionais. Além disso, as propriedades da biomassa (composição química, teor de cinzas e sua composição, dimensão e forma das partículas, massa específica, teor de umidade, etc.), também desempenham um papel importante.

Os gases de pirólise compreendem monóxido de carbono, dióxido de carbono, metano, e menores quantidades de hidrogênio e hidrocarbonetos de C_2 (DI BLASI, 2008). Eles podem ser usados para geração de energia ou calor, ou podem ser sintetizados para produção de metanol ou amônia (GRØNLI, 1996).

Os líquidos gerados na pirólise consistem principalmente de hidrocarbonetos poliaromáticos, compostos aromáticos oxigenados tais como o fenol, e água, originada a partir tanto do teor de umidade do combustível sólido, quanto das reações de decomposição (HALLGREN, 1996; DI BLASI, 2008). Esses líquidos podem ser aprimorados em combustíveis líquidos de hidrocarbonetos de maior qualidade para motores de combustão, ou usados diretamente para a produção de energia ou calor (GRØNLI, 1996).

Biocarvão é o produto sólido da pirólise de biomassa. Ele é composto principalmente por carbono ($\sim 85\%$), mas também pode conter oxigênio e hidrogênio. É caracterizado por uma grande área superficial dos poros. Ao contrário dos combustíveis fósseis, a biomassa contém pouca cinza inorgânica (BASU, 2013). Esse carvão é muito útil como combustível renovável, principalmente para culinária doméstica e churrasco, pode ser aprimorado em carvão ativado e utilizado como redutor na indústria metalúrgica. Antal e Grønli (2003) também citam outras utilidades do biocarvão. Devido à sua inerente porosidade, que resulta em alta área superficial, ele se torna um preferencial adsorvente para o tratamento de água e ar. Um leito compactado de carvão carbonizado conduz eletricidade quase tão bem quanto um leito compactado de partículas de grafite. Consequentemente, ele pode ser utilizado para formar eletrodos. Existe também um amplo uso do carvão para correção de solos, o que funciona como um sequestro de carbono e reduz o efeito estufa.

As características do carvão resultante dependem enormemente das condições de

pirólise, como taxa de aquecimento e temperatura final de pirólise, fazendo com que essas condições sejam um dos parâmetros mais importantes ao se estudar a reatividade do carvão e sua estrutura porosa (BARRIO, 2002). Altas taxas de aquecimento contribuem para o aumento da reatividade do carvão, tornando-o mais adequado para tratamentos térmicos posteriores, como combustão e gaseificação.

2.1.1.2 Liquefação

Liquefação é um processo termoquímico que ocorre a baixa temperatura (250–400 °C) e alta pressão (5–20 MPa), no qual a biomassa é convertida em três produtos, isto é, uma fração de bio-óleo (produto alvo), uma fração de gás e uma fração de resíduo sólido, com uso de água ou outro solvente adequado (HUANG; YUAN, 2015). O objetivo principal é maximizar a produção de líquido, que possui uma qualidade superior em relação ao formado pela pirólise quanto ao poder calorífico superior (35–40 MJ/kg comparado a 20–25 MJ/kg) e menor quantidade de oxigênio (GRØNLI, 1996).

2.1.1.3 Torrefação

A torrefação é um tratamento termoquímico com uma temperatura de operação entre (200 – 300 °C). É realizada a condições atmosféricas, na ausência de oxigênio e é caracterizada por baixas taxas de aquecimento (< 50 °C/min) (BERGMAN et al., 2005). Segundo Luengo, Felfi e Bezzon (2006), Felfi, Luengo e Soler (2000), nestas condições a umidade é removida e a hemicelulose é degradada, causando a liberação de ácido acético, frações de fenol e outros compostos de baixo poder calorífico. Também ocorre uma pequena despolimerização da lignina e da celulose. Deste processo, resulta um material intermediário entre a biomassa e o carvão, com altos rendimentos energéticos. O objetivo fundamental da torrefação é concentrar a energia da biomassa em um produto formado em curto tempo, baixas taxas de aquecimento e temperaturas moderadas, permitindo reter no próprio produto os voláteis de maior poder calorífico.

Basu (2013) define também a torrefação como sendo um processo termoquímico em um ambiente inerte ou com oxigênio limitado, aonde a biomassa é lentamente aquecida até uma faixa de temperatura especificada e mantida lá por um tempo estabelecido, de modo a resultar na quase completa degradação da sua hemicelulose enquanto maximiza a quantidade de massa e energia do produto sólido. A torrefação altera a estrutura química da biomassa para aumentar o seu teor de carbono, reduzindo o teor de oxigênio.

As reações de decomposição fazem com que a biomassa fique completamente seca e perca a sua estrutura tenaz e fibrosa. Com isto, a moabilidade dessa biomassa é melhorada significativamente. Além disso, a torrefação aumenta o poder calorífico da biomassa e transforma sua natureza higroscópica para que se torne um material hidrofóbico (BERGMAN; KIEL, 2005).

A biomassa torrificada pode ser utilizada como um combustível mais conveniente para a gaseificação. Também serve como combustível para aquecimento doméstico. Industrialmente, pode ser usada na queima em caldeiras para produção de vapor para geração de energia elétrica, ou na co-combustão com carvão mineral. Ela apresenta características que a tornam interessante para a aplicação como reductor na indústria metalúrgica, substituindo o coque mineral em alto-fornos para redução da emissão de carbono (LUENGO; FELFLI; BEZZON, 2006; BASU, 2013).

2.1.1.4 Considerações

Uma certa dúvida surge quanto à utilização e significado do termo “pirólise”. Segundo Basu (2013), o termo “pirólise” é comumente empregado em um sentido mais restritivo para o processo térmico que visa a produção de extratos líquidos a partir da biomassa. Porém, o termo “pirólise” significa decomposição térmica ou alteração química provocada pelo calor. Nessa definição estão contidos os processos de carbonização, torrefação e a pirólise para produção de líquidos.

No sentido mais amplo, a pirólise pode ser classificada como lenta ou rápida, baseada na taxa de aquecimento. Na pirólise lenta, o tempo de residência dos gases condensáveis na zona de pirólise é da ordem de minutos. Esse processo visa a produção de carvão e pode ser dividido em carbonização e torrefação. A torrefação ocorre em uma faixa de temperaturas baixa e estreita (200 – 300 °C), enquanto a carbonização ocorre em temperaturas mais elevadas e em uma faixa mais ampla. Na pirólise rápida, o tempo de residência dos gases condensáveis na zona de pirólise é da ordem de segundos ou milissegundos. Esse processo visa a produção de líquido. Nessa classificação existem ainda a pirólise *flash* e a pirólise ultra-rápida. Esta última envolve um aquecimento extremamente rápido, e visa principalmente a produção de gases. A Tabela 1 apresenta as principais características desses processos.

Esses processos de pirólise são bastante estudados para a degradação de biomassa, porém apresentam aplicações para outros sólidos como carvão mineral e plásticos. O uso dos processos de pirólise para o reaproveitamento de resíduos sólidos vem sendo estudado por diversos autores. Sharuddin et al. (2016) fizeram uma revisão dos estudos existentes sobre pirólise de plásticos presentes nos resíduos sólidos, como polietileno tereftalato (PET), polietileno de alta densidade (PEAD), polietileno de baixa densidade (PEBD), policloreto de polivinila (PVC), polipropileno (PP), poliestireno (PS) e também a pirólise de misturas desses plásticos. O objetivo desses trabalhos é o aproveitamento de resíduos plásticos, transformando-os em combustível líquido. Uma revisão também foi feita da influência dos parâmetros do processo no produtos finais, sendo eles gases, líquidos e sólidos. Esses parâmetros incluem temperatura, pressão, tipo de reator, tempo de residência, uso de catalisadores e do gás fluidizante.

Tabela 1 – Características dos processos de pirólise

Processo de pirólise	Tempo de residência	Taxa de aquecimento	Temperatura final (°C)	Produtos
Torrefação	10 - 60 min	Muito pequena	280	Biomassa torrificada
Carbonização	Dias	Pequena	> 400	Carvão
Rápida	< 2 s	Alta	~ 500	Bio-óleo
Flash	< 1 s	Alta	< 650	Bio-óleo, gás e produtos químicos
Ultra-rápida	< 0,5 s	Muito alta	~ 1000	Gás e produtos químicos

Fonte: Adaptado de [Basu \(2013\)](#)

[Martínez et al. \(2013\)](#) apresentam o estado da arte da pirólise de pneus, mostrando o efeito dos parâmetros operacionais (taxa de aquecimento, temperatura, pressão, tempo de residência e gás de fluidização) na distribuição e nas propriedades físicas e químicas dos produtos (gás, líquido e sólido).

[Miranda et al. \(2013a\)](#) e [Miranda et al. \(2013b\)](#) fizeram estudos para avaliar os possíveis caminhos do mecanismo da pirólise de pneus e também da mistura de pneus com plásticos. Diferentes modelos cinéticos foram aplicados, associados à dados experimentais, e os parâmetros cinéticos foram avaliados.

2.1.2 Processos heterogêneos de oxirredução

Os sólidos também podem sofrer conversão termoquímica por meio de reações heterogêneas com agentes oxidantes e redutores. Os exemplos desses processos são a combustão e a gaseificação. O produto útil da combustão é a energia térmica contida nos gases gerados. Gases esses que não apresentam mais um poder calorífico, apenas um calor sensível. Os produtos da gaseificação são gases de composição química que ainda possuem um poder calorífico utilizável, podendo servir como combustível ou serem transformados em outros produtos químicos.

Quando consideramos os processos completos que ocorrem nos reatores de combustão e gaseificação, ambos apresentam como etapas iniciais a desumidificação e a degradação térmica do sólido, podendo os gaseificadores também apresentarem produtos líquidos (alcatrão) provenientes da pirólise.

2.1.2.1 Combustão

Combustão é definida como a oxidação completa do combustível. O termo pode se referir à reação heterogênea entre o oxigênio e o carvão, ou às reações homogêneas entre o oxigênio e os gases e alcatrão combustíveis produzidos nos processos de pirólise e gaseificação (DUFFY, 2012). Essas reações são exotérmicas, e o calor liberado é, atualmente, a maior fonte de consumo de energia pelo homem (BASU, 2013).

As finalidades da combustão de biomassa são numerosas. Ela pode ser utilizada para fornecer calor para cozinhar, ou para aquecimento industrial ou residencial. Pode ser queimada em caldeiras para a produção de vapor, que pode ser usado em turbinas à vapor para geração de eletricidade. O calor dos gases quentes pode ser usado como fonte para processos. A biomassa pode ser queimada em conjunto com materiais residuais ou combustíveis fósseis, no processo conhecido como co-combustão. Pode servir também como fonte de energia para motores Stirling.

O processo de combustão da biomassa é o mais conhecido dos processos termoquímicos de conversão, e sua pesquisa está focada atualmente na emissão de NO_x e, para algumas biomassas, na emissão de SO_x (BARRIO, 2002).

2.1.2.2 Gaseificação

Gaseificação converte combustíveis fósseis e não fósseis em gases e produtos químicos úteis. Ele requer um meio para reação (agente de gaseificação), que pode ser gases ou água supercrítica. Os agentes gasosos incluem ar, oxigênio, vapor, ou uma mistura desses. Atualmente, gaseificação de combustíveis fósseis é mais comum que de combustíveis não fósseis, como biomassa, para produção de gás de síntese (BASU, 2013).

Segundo Duffy (2012), a gaseificação é a degradação térmica do combustível na presença de um agente oxidante. O termo gaseificação se refere às reações heterogêneas entre o carvão gerado na pirólise e reagentes outros que não o oxigênio, principalmente CO_2 e H_2O .

De acordo com Grønli (1996), a gaseificação pode também ser referida como pirólise por oxidação parcial. Seu objetivo é maximizar a formação de produtos gasosos, e geralmente ocorre a temperaturas entre 800°C e 1100°C . A mistura gasosa produzida contém CO , CO_2 , H_2 , H_2O , CH_4 , N_2 (se ar for usado como agente de gaseificação) e vários contaminantes como pequenas partículas de carvão, e pequenas quantidades de cinzas e alcatrão. Esse gás pode ser transformado em metanol por síntese, queimado em caldeiras para produção de água quente e vapor, ou queimado em turbinas à gás ou motores de combustão interna para produção de energia elétrica. Nesse último caso, o gás combustível precisa ser primeiramente limpo dos contaminantes.

O poder calorífico e a composição do gás produzido em um gaseificador depende

fortemente da natureza e quantidade do agente de gaseificação utilizado. Gaseificação com ar produz gases de baixo poder calorífico, 4 a 7 MJ/Nm³, principalmente devido ao efeito de diluição do nitrogênio. Vapor produz gases de médio poder calorífico, de 10 a 18 MJ/Nm³. Oxigênio produz gases com o maior poder calorífico, de 12 a 28 MJ/Nm³ (BASU, 2013).

2.2 Modelos de Conversão de Massa

A conversão termoquímica de sólidos envolve alguns processos distintos. Esse processo é complexo quando comparado com combustíveis gasosos, e é geralmente aceito que ocorre conforme os mecanismos ilustrados na Figura 1 (DUFFY, 2012).

Inicialmente, as partículas de sólido sofrem aquecimento e perda inicial de massa devido à secagem. O fornecimento de calor causa também a degradação térmica e liberação dos voláteis, resultando em produtos gasosos, líquidos e um resíduo sólido rico em carbono. Se esse sólido se encontrar em um ambiente propício, reações heterogêneas de oxidação e redução (combustão e gaseificação) irão ocorrer. Ao final desses processos, o que resta do material sólido é a matéria inorgânica (cinzas). Essas etapas da conversão termoquímica podem ocorrer simultaneamente.

2.2.1 Secagem

Sólidos porosos como carvão e biomassa geralmente apresentam umidade. A água pode estar presente de três formas (DUFFY, 2012):

- Água líquida livre que se encontra no interior dos poros do sólido. O transporte é devido ao escoamento capilar pelos vazios, e a energia para a sua evaporação é o calor latente de evaporação.
- Água adsorvida na estrutura porosa do sólido. O transporte ocorre por difusão, e a energia para a evaporação compreende o calor latente de evaporação além do calor de adsorção. Para madeira, acredita-se que essa água se encontra ligada às fibras de celulose e hemicelulose por ligações de hidrogênio (BELLAIS, 2007; DUFFY, 2012).
- Vapor de água resultante da evaporação da água contida no sólido. O transporte é ocorre tanto por convecção quanto por difusão.

Duffy (2012) comenta que a secagem ocorre inicialmente por difusão da água a partir da superfície da partícula, mas aumenta rapidamente quando a temperatura de evaporação, aproximadamente 100 °C, é atingida. A secagem começa na superfície da partícula pela presença de maiores temperaturas, mas a frente de secagem move para o seu interior

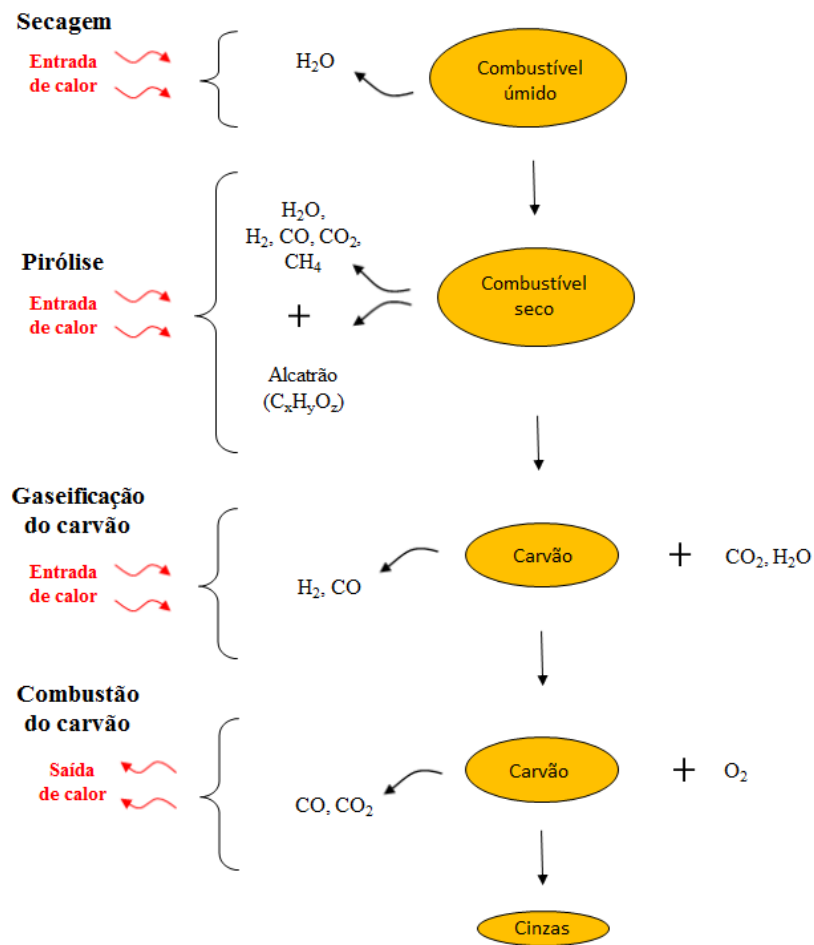


Figura 1 – Etapas da conversão termoquímica de sólidos

Fonte: Adaptado de [Duffy \(2012\)](#)

devido à transferência de calor e de massa. Quatro métodos para a modelagem da secagem em CFD serão destacados.

2.2.1.1 Modelo de temperatura constante

Esse modelo considera que a secagem começa quando o sólido atinge uma temperatura pré-definida, T_{evap} , geralmente considerada 100 °C. Enquanto o sólido estiver a essa temperatura, todo calor transferido para ele é utilizado para vaporizar qualquer umidade presente, sem fazer distinção entre as diferentes formas de água ([DUFFY, 2012](#)). A taxa de evaporação é dada por ([PETERS; BRUCH, 2003](#)):

$$\frac{d\rho_{um}}{dt} = \begin{cases} \frac{(T_s - T_{evap})\rho_s c_{p,s}}{\Delta H_{evap} \Delta t}, & \text{se } T_s \geq T_{evap} \\ 0, & \text{se } T_s < T_{evap} \end{cases} \quad (2.1)$$

Onde $d\rho_{um}/dt$ é a perda de massa de umidade, T_s é a temperatura da fase sólida, T_{evap} é a temperatura de evaporação pré-definida, ρ_s é a massa específica da fase sólida, $c_{p,s}$ é o calor específico da fase sólida, ΔH_{evap} é o calor latente de evaporação, e Δt é o intervalo de tempo.

Duffy (2012) ressalta que a evaporação da umidade a partir de uma determinada temperatura causa uma função degrau no tempo para simulações transientes, o que pode levar a instabilidade. Para melhorar a estabilidade pode-se aplicar um fator à Equação 2.1. Esse fator também implica que apenas parte do calor recebido é dedicado para o processo de evaporação, enquanto o resto é usado para aquecer o sólido acima da temperatura de evaporação (COLLAZO et al., 2012). Como fator para essa taxa de evaporação, Yang et al. (2007) utilizam a fração mássica de umidade inicial presente no sólido. Collazo et al. (2012) utilizam um fator de 0,5.

Bryden, Ragland e Rutland (2002) comentam que esse modelo, também chamado de modelo de balanço de energia, é simples e fácil de ser implementado numericamente. Sua principal desvantagem é que a zona de secagem é reduzida a uma superfície móvel infinitamente fina e pode não modelar corretamente a secagem e pirólise de partículas pequenas, onde a espessura da zona de secagem não é desprezível comparado à espessura do material. Duffy (2012) cita como outra desvantagem desse modelo o fato de ele desconsiderar a secagem devido ao transporte e difusão da água adsorvida, que irá ocorrer abaixo da temperatura definida de evaporação.

2.2.1.2 Modelo algébrico

Esse modelo, exposto por Alves e Figueiredo (1989), utiliza uma expressão algébrica para a temperatura como função do conteúdo de umidade (BRYDEN; RAGLAND; RUTLAND, 2002). Segundo Duffy (2012), nesse modelo a secagem é determinada por uma combinação entre a transferência de calor e o equilíbrio vapor-líquido. Similarmente ao modelo de temperatura constante, quando o sólido atinge T_{evap} , todo calor transferido é utilizado para vaporizar a umidade presente. Porém, T_{evap} não é considerada constante, e sim uma função do conteúdo de umidade, Y_{um} . Quando Y_{um} é maior que 14,4% em base seca, pode-se assumir T_{evap} igual a 100 °C, com descontinuidade e erros desprezíveis (ALVES; FIGUEIREDO, 1989). Para esse caso, o modelo comporta-se de forma idêntica ao modelo de temperatura constante (DUFFY, 2012). Quando Y_{um} for menor que 14,4% em base seca, T_{evap} é calculada pela expressão dada por Alves e Figueiredo (1989):

$$T_{evap} = 1/(2,13 \times 10^{-3} + 2,778 \times 10^{-4} \ln(Y_{um}) + 9,997 \times 10^{-6} [\ln(Y_{um})]^2 - 1,461 \times 10^{-5} [\ln(Y_{um})]^3) \quad (2.2)$$

Segundo [Alves e Figueiredo \(1989\)](#), esse modelo negligencia a difusão de água adsorvida, a difusão de vapor e ar e também gradientes de pressão no interior do sólido. O movimento da água livre também não é retratado. Essas considerações restringem a validade do modelo para casos com temperatura maior que 150 °C, conteúdo de umidade inicial menor que ~45%, e dimensões da amostra na direção longitudinal não muito maiores que a dimensão na direção transversal.

2.2.1.3 Modelo cinético

Esse modelo, visto em [Chan, Kelbon e Krieger \(1985\)](#) e [Krieger-Brockett e Glaister \(1988\)](#), se tornou um dos modelos mais utilizados na literatura. Ele representa a secagem como uma reação química ([Equação 2.3](#)) cuja cinética é calculada usando uma expressão de Arrhenius ([BRYDEN; RAGLAND; RUTLAND, 2002](#)).



A taxa de evaporação é dada por:

$$\frac{d\rho_{um}}{dt} = -k_{um} \cdot \rho_{um} \quad (2.4)$$

onde ρ_{um} é a massa específica de umidade na fase sólida, e k_{um} é dado pela equação de Arrhenius:

$$k_{um} = A_{um} e^{-E_{um}/RT} \quad (2.5)$$

onde A_{um} é o fator pré-exponencial, E_{um} é a energia de ativação, R é a constante universal dos gases, e T é a temperatura.

[Peters e Bruch \(2003\)](#) comentam que essa abordagem apresenta a vantagem de considerar uma energia de ativação que contabiliza tanto uma temperatura de evaporação dependente da temperatura quanto uma resistência variável da água adsorvida, além de ser numericamente estável. No entanto, as constantes cinéticas, A_{um} e E_{um} , somente são válidas para as condições nas quais elas foram derivadas ([DUFFY, 2012; PETERS; BRUCH, 2003](#)).

2.2.1.4 Modelo de transporte

O modelo de transporte, cuja uma das origens é o trabalho de [Ouelhazi, Arnaud e Fohr \(1992\)](#), é o mais completo modelo de secagem, e também foi estudado por [Grønli](#)

(1996). Uma descrição detalhada desse modelo, assim como das considerações aplicadas, pode ser encontrada nesses dois trabalhos. Duffy (2012) fez uma descrição sucinta desse modelo.

Segundo Duffy (2012), o modelo de transporte é o mais sofisticado dos modelos de secagem e tenta capturar os detalhados mecanismos de transporte por trás do processo de secagem. Eles incluem o transporte da água adsorvida, da água líquida, e do vapor de água dentro da partícula, seguido da difusão a partir da superfície externa. Para a conservação da água adsorvida, o transporte é tratado como um processo de difusão:

$$\frac{\partial \rho_{um,a}}{\partial t} + \frac{\partial(\rho_{um,a} u_{um,a})}{\partial x} = S_{um,a} \quad (2.6)$$

$$\rho_{um,a} u_{um,a} = \rho_{ss} D_a \frac{\partial}{\partial x} \left(\frac{\rho_{um,a}}{\rho_{ss}} \right) \quad (2.7)$$

onde o subscrito a refere-se à água adsorvida, $u_{um,a}$ é a velocidade, D_a é o coeficiente de difusão, ρ_{ss} é a massa específica do sólido seco, e $S_{um,a}$ é a taxa de desorção da água.

Para a conservação da água líquida, o transporte é devido à capilaridade:

$$\frac{\partial \rho_{um,l}}{\partial t} + \frac{\partial(\rho_{um,l} u_{um,l})}{\partial x} = S_{um,l} \quad (2.8)$$

onde o subscrito l refere-se à água líquida, $S_{um,l}$ é a taxa de evaporação, e a velocidade $u_{um,l}$ é dada pela Lei de Darcy, desconsiderando o efeito da gravidade:

$$u_{um,l} = - \frac{K_{i,l} K_{r,l}}{\mu_{um}} \frac{\partial P_l}{\partial x} \quad (2.9)$$

onde $K_{i,l}$ é a permeabilidade intrínseca da água, $K_{r,l}$ é a permeabilidade relativa da água, μ_{um} é a viscosidade dinâmica da água líquida, e P_l é pressão de líquido, que está relacionada com a pressão de gás pela capilaridade:

$$P_l = P_g - P_c \quad (2.10)$$

onde P_g é a pressão de gás e P_c é a pressão capilar.

Para a conservação do vapor de água, o transporte é devido à combinação da convecção do gás com a difusão do vapor no gás:

$$\frac{\partial Y_{vap,g}\rho_g}{\partial t} + \frac{\partial(Y_{vap,g}\rho_g u_g)}{\partial x} = \frac{\partial}{\partial x} \left(\rho_g D_{ef} \frac{\partial Y_{vap,g}}{\partial x} \right) + S_{um,g} \quad (2.11)$$

onde $Y_{vap,g}$ é a fração mássica de vapor de água na fase gasosa, ρ_g é a massa específica da fase gasosa, u_g é a velocidade superficial da fase gasosa, D_{ef} é uma difusividade efetiva, e $S_{um,g}$ representa as fontes de vapor de água na fase gasosa. A velocidade superficial do gás é descrita pela Lei de Darcy:

$$u_g = -\frac{K_{i,g}K_{r,g}}{\mu_g} \frac{\partial P_g}{\partial x} \quad (2.12)$$

onde $K_{i,g}$ é a permeabilidade intrínseca do gás, $K_{r,g}$ é a permeabilidade relativa do gás, μ_g é a viscosidade dinâmica do gás, e P_g é pressão de gás.

A perda de umidade ocorre pela superfície da partícula, sendo modelada como uma transferência de massa do tipo:

$$\frac{\partial \rho_{um}}{\partial t} = S_{um,a} + S_{um,l} + S_{um,g} = h_m A (\rho_{um,sup} - \rho_{um,\infty}) \quad (2.13)$$

onde $\rho_{um,sup}$ é a massa específica de umidade na superfície, $\rho_{um,\infty}$ é a massa específica de umidade no ambiente, A é a área superficial, e h_m é o coeficiente de transferência de massa.

A água livre e a água adsorvida apresentam níveis distintos de energia. A água livre é tratada como água líquida, e portanto, a energia requerida para realizar a mudança de estado é simplesmente o calor de evaporação. No entanto, a vaporização da água adsorvida requer uma energia adicional, o que é contabilizado incluindo o calor de dessorção:

$$Y_{um} \geq Y_{um,psf} \quad \Delta H_{tot} = \Delta H_{evap} \quad (2.14)$$

$$Y_{um} < Y_{um,psf} \quad \Delta H_{tot} = \Delta H_{evap} + \Delta H_{des} \quad (2.15)$$

onde Y_{um} é a fração mássica de umidade, $Y_{um,psf}$ é o ponto de saturação da fibra ($\approx 0,3$ para madeira), ΔH_{tot} é o calor total da reação, e ΔH_{des} é o calor de dessorção.

De acordo com [Duffy \(2012\)](#), o modelo de transporte é o método mais detalhado de se representar a secagem, mas também é o mais custoso computacionalmente. O formato da partícula e as propriedades do material devem ser finamente avaliados por toda a partícula,

e equações de conservação adicionais devem ser resolvidas para cada tipo de água presente nela. Além disso, a complexidade adicional introduz um número significativo de variáveis de parâmetros adicionais, tais como as permeabilidades intrínseca e relativa, a pressão capilar, etc., muitas das quais são difíceis de se determinar experimentalmente. Discrepâncias nos valores desses parâmetros e variáveis podem diminuir a representatividade do modelo.

2.2.2 Pirólise

Os métodos para a modelagem da degradação térmica de sólidos e a liberação dos voláteis podem ser classificados de forma mais ampla em duas classes. Na classe dos modelos cinéticos, a taxa de perda de massa do sólido é correlacionada com a temperatura conforme a partícula é aquecida para determinar a taxa de reação, tipicamente expressa por meio de uma equação de Arrhenius (DUFFY, 2012). A classe dos modelos de rede ou estruturais (*network models*) apresenta uma abordagem mais mecanicista, baseada em uma descrição físico-química da estrutura do carvão (EATON et al., 1999; ARENILLAS et al., 2001).

2.2.2.1 Modelos cinéticos

Os modelos cinéticos, também referenciados como modelos empíricos, descrevem a conversão das diversas substâncias por meio de reações químicas, cujas cinéticas são calculadas baseadas em parâmetros empíricos obtidos de dados experimentais, como representado na [Equação 2.16](#):



onde k é a taxa de reação, dada pela Equação de Arrhenius ([Equação 2.17](#)), e representa a dependência da taxa de conversão com a temperatura.

$$k = Ae^{-E/RT} \quad (2.17)$$

onde A é o fator pré-exponencial, E é a energia de ativação, R é a constante universal dos gases, e T é a temperatura.

A taxa de conversão de uma determinada espécie i é então dada pela [Equação 2.18](#) (CONESA et al., 2001):

$$\frac{dm_i}{dt} = kf(m_i) \quad (2.18)$$

onde m_i é a massa da espécie i . A função $f(m_i)$ mostra que a taxa de conversão é proporcional à massa presente (CONESA et al., 2001). Uma das formas mais simples para essa função é considerá-la como uma lei de potência da massa:

$$\frac{dm_i}{dt} = km_i^n \quad (2.19)$$

Ou em função das frações mássicas:

$$\frac{dY_i}{dt} = kY_i^n \quad (2.20)$$

onde Y_i é a fração mássica da espécie i , e n é a ordem da reação.

Mais informações sobre a função $f(m_i)$ pode ser encontrada em Conesa et al. (2001), Font et al. (2001) e Baker (1978).

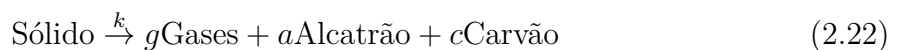
De acordo com Di Blasi (1993), os estudos cinéticos podem ser classificados em três grupos: os modelos globais de taxa única; os modelos de reações múltiplas de simples estágio; e os modelos semi-globais de dois estágios. Mais informações sobre esses modelos em Di Blasi (1993), Di Blasi (1998), Di Blasi (2008), Moghtaderi (2006), Wang e Yan (2008), Duffy (2012), Peters e Bruch (2003), Bellais (2007), Cunha (2010), Burnham e Braun (1999), Antal e Varhegyi (1995), Antal (1985).

2.2.2.1.1 Modelo global de taxa única

Esse modelo propõe um esquema cinético simples para descrever a degradação térmica do sólido, por meio de uma reação global de passo único:



onde v é o coeficiente estequiométrico ou a fração mássica de voláteis presente no sólido. Se a composição dos voláteis resultantes da degradação for conhecida, eles podem ser desaglomerados, por exemplo, em alcatrão mais gases:



Segundo Duffy (2012), a maior vantagem desse modelo é a sua simplicidade, e como resultado, a maioria dos dados publicados de pirólise são para esse modelo. A quantidade

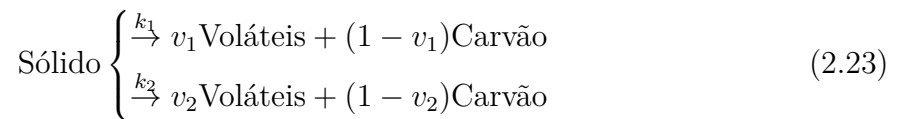
final de voláteis produzida é, contudo, válida apenas para condições correspondentes às condições do experimento, o que limita a sua aplicabilidade. Também afirma que o modelo não pode prever a taxa de geração de espécies individuais. Portanto, a menos que uma abordagem de espécies aglomerada seja utilizada, a distribuição das espécies deve ser determinada previamente.

Di Blasi (1998) também comenta que, por assumir uma proporção constante entre voláteis e carvão, esse modelo não permite a predição da dependência com as condições de reação para a quantidade produzida de cada produto.

2.2.2.1.2 Modelos de reações múltiplas de simples estágio

Na realidade, a quantidade total de voláteis produzida é dependente da evolução e do histórico de temperatura do sólido, e foi observado que ela cresce com a taxa de aquecimento. Assumir previamente a quantidade final de voláteis, como no modelo de taxa única, é ignorar esse efeito (DUFFY, 2012). Para tentar contornar esse problema, alguns modelos de reações múltiplas foram propostos.

O primeiro modelo é o de duas reações paralelas concorrentes (KOBAYASHI; HOWARD; SAROFIM, 1977; UBHAYAKAR et al., 1977):

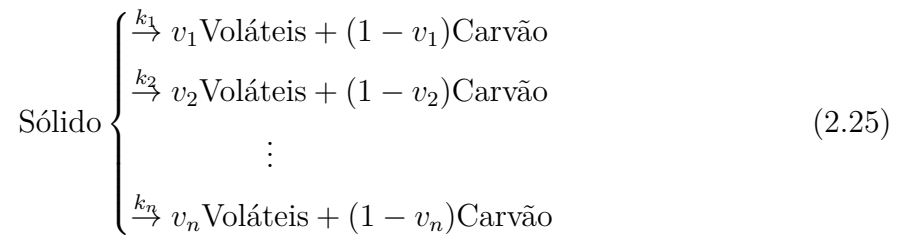


$$\begin{aligned} k_1 &= A_1 e^{-E_1/RT} \\ k_2 &= A_2 e^{-E_2/RT} \end{aligned} \quad (2.24)$$

Segundo Cunha (2010), nesse modelo, dois fatores de produção total de voláteis são utilizados nas reações competitivas. Uma energia de ativação é muito maior que a outra, a reação de baixa produção é favorecida em baixas temperaturas, enquanto que a reação de alta produção é favorecida em altas temperaturas. Com esse método, a porcentagem de voláteis liberados do sólido se torna dependente da temperatura, o que corresponde às observações experimentais (EATON et al., 1999; KOBAYASHI; HOWARD; SAROFIM, 1977).

Duffy (2012) comenta que, apesar de ser uma evolução do modelo de taxa única, os coeficientes estequiométricos v_i para cada produto individual ainda precisam ser previamente especificados para cada reação competitiva.

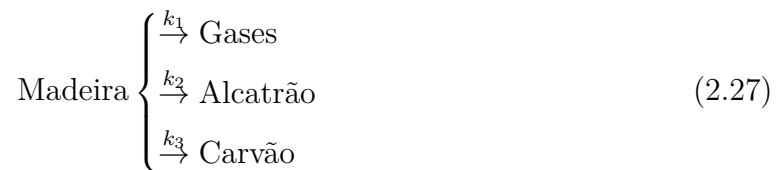
Esta abordagem pode ser ainda estendida para incluir n reações:



Um segundo modelo é o de reações paralelas baseadas na distribuição do produto, onde o sólido decompõe diretamente em cada produto de acordo com um conjunto de reações paralelas independentes (MOGHTADERI, 2006):

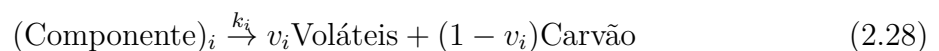


Shafizadeh e Chin (1977) propuseram a utilização desse modelo para a degradação de madeira em gases leves, alcatrão e carvão. Uma abordagem similar foi feita por Turner e Mann (1981).



Duffy (2012) comenta que a maior vantagem desse modelo é que os coeficientes estequiométricos não precisam ser assumidos previamente. A quantidade de cada produto é determinada pela evolução de temperatura do sólido, já que a formação de cada produto é descrita por meio de reações competitivas dependentes da temperatura.

Um terceiro modelo do grupo de reações múltiplas de simples estágio é o modelo de reações paralelas baseadas na composição do sólido, também chamado de modelo de superposição por Cunha (2010).



A biomassa pode ser considerada composta principalmente de hemicelulose, celulose e lignina. Segundo Di Blasi (2008), na maioria dos casos, o número de pseudo-componentes é três, correspondendo à hemicelulose, celulose e lignina. Porém, em alguns casos, um número maior de reações pode ser levada em consideração para a decomposição dos componentes, principalmente da hemicelulose e da lignina. Um exemplo é o trabalho de

Alves e Figueiredo (1989), que utilizaram seis reações, sendo uma para a celulose, uma para a hemicelulose e quatro para a lignina.

A Equação 2.28 mostrou a decomposição de cada componente i por meio de um modelo global de taxa única (Equação 2.21). Porém, vale ressaltar que outros modelos cinéticos de pirólise podem ser utilizados para cada componente individual i , como a Equação 2.23 ou a Equação 2.27.

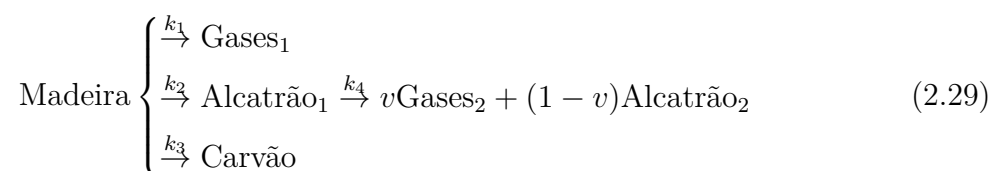
Segundo Duffy (2012), esse modelo possui a vantagem de que, uma vez que a taxa de decomposição de cada componente for conhecida, a taxa de pirólise de diferentes sólidos pode ser calculada simplesmente ajustando-se a composição do sólido, ou seja, as frações mássicas de cada componente. Entretanto, esse método herda as desvantagens do modelo particular aplicado a cada componente individual.

2.2.2.1.3 Modelos semi-globais de dois estágios

Os modelos anteriores trataram apenas da degradação primária do sólido, onde ele se decompõe em materiais voláteis e um resíduo sólido (carvão). Porém, segundo Di Blasi (2008), em temperaturas elevadas e tempo de residência suficientemente longo, reações secundárias dos vapores de alcatrão primário tornam-se ativas. Essas reações secundárias podem ocorrer homoganeamente na fase gasosa, ou heterogeneamente na superfície do carvão. Em particular, as reações heterogêneas podem dar origem à gaseificação exotérmica do carvão pela redução dos voláteis ricos em oxigênio produzidos na pirólise primária (DI BLASI, 1993). Essas reações secundárias heterogêneas não serão tratadas neste trabalho, mas informações e estudos sobre os efeitos catalíticos de alguns materiais para a degradação do alcatrão podem ser encontrados em Sutton, Kelleher e Ross (2001) e El-Rub, Bramer e Brem (2004).

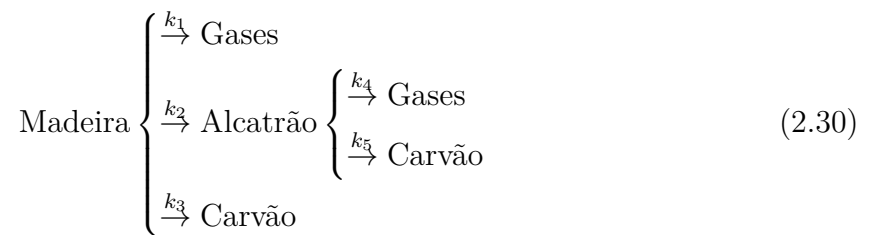
Morf, Hasler e Nussbaumer (2002) comentam que processos como craqueamento, oxidação parcial, repolimerização e condensação podem ocorrer tanto nos processos homogêneos quanto nos heterogêneos. De acordo com Di Blasi (2008), a composição química complexa dos alcatrões produzidos exigiria um grande número de reações químicas para descrever os detalhes dessas transformações. Para evitar essa complexidade, os modelos propostos tratam essas reações de maneira mais simplificada.

Chan, Kelbon e Krieger (1985) propuseram um modelo baseado no esquema de degradação de Shafizadeh e Chin (1977) (Equação 2.27):



Esse modelo considera o craqueamento do alcatrão, que é a degradação de alcatrões de alto peso molecular em gases de baixo peso molecular, por meio de uma reação secundária, transformando o alcatrão primário em gases e alcatrão secundário. Segundo [Duffy \(2012\)](#), a desvantagem dessa abordagem é que o coeficiente estequiométrico para a reação secundária, v , deve ser assumido previamente.

O esquema de degradação de [Shafizadeh e Chin \(1977\)](#) pode ser ampliado para incluir a reação de repolimerização do alcatrão, formando carvão, como visto em [Thurner e Mann \(1981\)](#), [Di Blasi e Russo \(1993\)](#) e [Di Blasi \(1996\)](#):



Esse modelo utiliza reações secundárias competitivas para o craqueamento (formação de gases) e a repolimerização (formação de carvão).

2.2.2.1.4 Modelo de energia de ativação distribuída – DAEM

O modelo de energia de ativação distribuída (DAEM - *Distributed Activation Energy Model*) ([VAND, 1943](#); [PITT, 1962](#); [ANTHONY et al., 1975](#)) trata a evolução da pirólise por um conjunto de reações paralelas e independentes de simples estágio, de forma similar aos modelos de reações múltiplas de simples estágio.

De acordo com [Rostami, Hajaligol e Wrenn \(2004\)](#), o DAEM assume que a evolução de um dado produto envolve um número infinito de reações químicas independentes. Cada reação j contribui para a formação de um produto i de acordo com:

$$\frac{dY_{i,j}}{dt} = -k_j Y_{i,j} \quad (2.31)$$

onde Y_i refere-se à fração mássica da espécie i que não reagiu, ou seja, remanescente no sólido, e k_j é a taxa da reação j na forma de Equação de Arrhenius ([Equação 2.17](#)).

Rearranjando a [Equação 2.31](#), temos:

$$\frac{dY_{i,j}}{Y_{i,j}} = -k_j dt \quad (2.32)$$

Integrando a [Equação 2.32](#), temos a fração mássica remanescente para o tempo t :

$$Y_{i,j} = Y_{i,j,0} \exp\left(-\int_0^t k_j dt\right) \quad (2.33)$$

onde $Y_{i,0}$ é a fração mássica inicial da espécie i no material, antes do início da pirólise ([ROSTAMI; HAJALIGOL; WRENN, 2004](#)).

O DAEM assume um número infinito de equações. As reações individuais, i , são substituídas por uma dependência contínua com a energia de ativação, E , por meio de uma curva de distribuição das energias de ativação, $f_i(E)$, para representar as diferenças nas energias de ativação das diferentes reações ([DUFFY, 2012](#)). A fração da espécie i disponível como fonte pela reação j , $Y_{i,j,0}$, é representada como a fração da espécie i com energia de ativação entre E e $E + dE$:

$$dY_i = Y_{i,0} f_i(E) dE \quad (2.34)$$

Assim, a quantidade total da espécie i remanescente no tempo t é dada pela integral ([ROSTAMI; HAJALIGOL; WRENN, 2004](#)):

$$Y_i = Y_{i,0} \int_0^\infty \exp\left(-\int_0^t k(E) dt\right) f_i(E) dE \quad (2.35)$$

ou então, expandindo-se a taxa de reação $k(E)$:

$$Y_i = Y_{i,0} \int_0^\infty \exp\left(-A_i(E) \int_0^t e^{-E/RT} dt\right) f_i(E) dE \quad (2.36)$$

onde $A_i(E)$ é o fator pré-exponencial dependente da energia de ativação ([MIURA, 1995](#)). Uma suposição comum é a de que o fator pré-exponencial $A_{i,j}$ de todas as reações j possuem o mesmo valor, $A_{i,0}$, o que simplifica a análise ([PLEASE; MCGUINNESS; MCELWAIN, 2003; MIURA, 1995](#)).

A curva de distribuição, $f_i(E)$, é geralmente assumida como uma distribuição Gaussiana com uma energia de ativação média $E_{i,0}$ e um desvio padrão σ_i ([ROSTAMI; HAJALIGOL; WRENN, 2004; MIURA, 1995](#)).

$$f_i(E) = \frac{1}{\sigma_i(2\pi)^{1/2}} \exp\left(\frac{-(E - E_{i,0})^2}{2\sigma_i^2}\right) \quad (2.37)$$

Assim, a quantidade total da espécie i se torna:

$$Y_i = \frac{Y_{i,0}}{\sigma_i(2\pi)^{1/2}} \int_0^\infty \exp\left(-A_0(E) \int_0^t e^{-E/RT} dt - \frac{(E - E_{i,0})^2}{2\sigma_i^2}\right) dE \quad (2.38)$$

Segundo Miura (1995), a atribuição de uma distribuição Gaussiana para $f_i(E)$ nem sempre reflete as situações reais. Além disso, a suposição de um fator pré-exponencial $A_{i,0}$ constante pode não ser válida quando $f_i(E)$ se estender à uma ampla gama de valores para a energia de ativação E . Sendo assim, alguns estudos foram feitos para a estimação de $A_i(E)$ e $f_i(E)$, como Miura (1995), Maki, Takatsuno e Miura (1997), Miura e Maki (1998). Além da distribuição Gaussiana, outros estudos utilizam distribuições diferentes, como a Weibull (LAKSHMANAN; WHITE, 1994), e a Gamma (DING et al., 2005).

Uma outra dificuldade desse modelo é a integração dupla na Equação 2.36 ou 2.38. Algumas abordagens matemáticas foram propostas para lidar com essa dupla integração: como uma integração no tempo para uma taxa de aquecimento constante e o uso de uma função de erro complementar para representar a integral em E (SUUBERG, 1983); o uso de uma técnica de aproximação com função degrau para a determinação da distribuição da energia de ativação (DU; SAROFIM; LONGWELL, 1990); o uso de uma fórmula de quadratura Gauss-Hermite para fazer a integração (VÁRHEGYI; SZABÓ; ANTAL, 2002). Mais informações também em Burnham e Braun (1999), Please, McGuinness e McElwain (2003), Rostami, Hajaligol e Wrenn (2004), Cai e Liu (2008).

2.2.2.2 Modelos de rede

Segundo Duffy (2012), os modelos de rede (*network models*) são os modelos mais sofisticados disponíveis atualmente para pirólise, e são baseados em suposições de como a estrutura do sólido muda conforme ele é aquecido. Essa estrutura é representada por grupos aromáticos ligados por pontes estáveis e instáveis, e outros grupos periféricos (EATON et al., 1999). Conforme a temperatura aumenta, as ligações entre esses grupos são aleatoriamente quebradas, causando a degradação do sólido. Métodos estatísticos são aplicados para determinar o resultado da quebra das ligações, isto é, se carvão é formado, se gases leves são liberados, ou se um grupo se rompe da estrutura principal para formar alcatrão. Os métodos para modelar a formação de gás, alcatrão e carvão variam entre os modelos (DUFFY, 2012).

Na literatura pode-se encontrar três modelos de rede, que foram desenvolvidos para a decomposição de carvão mineral:

- *Functional Group - Depolymerization, Vaporization, Cross Linking Model* (FG-DVC) (SOLOMON et al., 1988; SOLOMON et al., 1990)

- FLASHCHAIN (NIKSA; KERSTEIN, 1991)
- *Chemical Percolation Devolatilization Model* (CPD) (GRANT et al., 1989; FLETCHER et al., 1990; FLETCHER et al., 1992)

A estrutura química do combustível pode ser caracterizada por NMR (*Nuclear Magnetic Resonance*), FTIR (*Fourier Transform Infra-Red*) ou outros procedimentos padronizados (CUNHA, 2010).

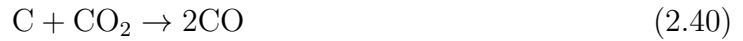
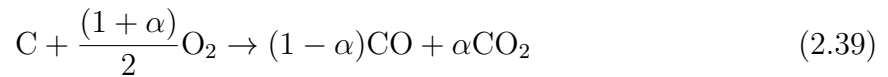
De acordo com Eaton et al. (1999), os três modelos incluem modelos de rede, caracterização da estrutura, reações de despolimerização, reações de reticulação (*crosslinking*), e a formação de carvão, alcatrão e gases não-condensáveis. No entanto, suas abordagens e detalhes na aplicação e modelagem diferem. Uma das principais diferenças é o modelo de rede empregado para interpretar as inter-relações características da estrutura macromolecular do carvão, alcatrão e coque. O modelo FG-DVC utiliza uma estrutura Bethe de dois parâmetros. O FLASHCHAIN usa um modelo de cadeia direta, sem ligações cruzadas, para aproximar o impacto qualitativo da rede macromolecular. O modelo CPD usa a teoria de percolação e uma estrutura Bethe tridimensional para aproximar uma rede de carvão onde as frações das pontes intactas e números de coordenação são determinados de dados de NMR (CUNHA, 2010; EATON et al., 1999). Uma comparação detalhada desses três modelos pode ser encontrada em Smith et al. (1994).

Esses modelos de rede tiveram origem para o estudo da pirólise de carvão mineral. Porém, pela pirólise de biomassa ser análoga à pirólise de carvão mineral, eles foram adaptados para serem utilizados com biomassa, embora modificações sejam necessárias para contabilizar a estrutura diferente e as proporções significativamente maiores de hidrogênio e oxigênio na biomassa (DUFFY, 2012). O modelo FG-DVC foi modificado resultando no bio-FGDVC (CHEN, 1998), e o modelo FLASHCHAIN resultou no bio-FLASHCHAIN (NIKSA, 2000).

Eaton et al. (1999) comentam que, apesar dos modelos estruturais serem mais complicados que os modelos cinéticos empíricos, apresentam maior flexibilidade, confiabilidade e aplicabilidade. Porém, também requerem uma substancial entrada de dados do usuário (DUFFY, 2012).

2.2.3 Reações Heterogêneas

Uma reação heterogênea é aquela que envolve reagentes em diferentes estados físicos (CUNHA, 2010). Para o caso da conversão termoquímica de sólidos, as reações heterogêneas correspondem às reações entre o resíduo carbônico (carvão ou coque) e reagentes gasosos, como O₂ (combustão), ou CO₂ e H₂O (gaseificação). Se o carvão for tomado como composto por carbono puro, essas reações tornam-se:



As reações heterogêneas entre gás e sólido são governadas por um complexo acoplamento entre fenômenos de transporte e cinética química (LAURENDEAU, 1978), como concentração do gás reagente, temperatura, pressão, estrutura e composição do carvão, e área superficial ativa total. O esquema geral de reação pode ser descrito por três processos básicos (LAURENDEAU, 1978):

- Difusão de massa e calor pela camada limite em torno do sólido;
- Difusão de massa e calor dentro da estrutura porosa do sólido;
- Reação dos gases com a superfície do sólido.

Por causa dessa dependência da reação tanto com a taxa de difusão quanto com a taxa de reação, vários pesquisadores, tais como (WALKER; RUSINKO; AUSTIN, 1959), postularam a existência das três diferentes zonas de temperatura ou regimes, nos quais um ou mais processos diferentes controlam a velocidade de reação global (HONG, 2000). A Figura 2 mostra essas três zonas ou regimes de reação.

A Zona I ocorre quando a velocidade da reação é lenta comparada com a difusão (baixas temperaturas e partículas pequenas), e a reação química controla a taxa global de reação (HONG, 2000). Nesta zona a energia de ativação observada é igual ao valor verdadeiro e a ordem aparente de reação, n , é igual à ordem verdadeira, m . Neste regime, a concentração do oxidante na superfície da partícula é praticamente a mesma do escoamento não perturbado (CUNHA, 2010). A conversão ocorre ao longo da partícula com alterações na densidade, mas com um tamanho constante (tempo de conversão e taxa da reação independente do tamanho da partícula) (DI BLASI, 2009).

Na Zona II, a taxa de difusão das espécies gasosas nos poros torna-se comparável à taxa de reação química, levando a uma penetração limitada de gás no carvão (DI BLASI, 2009). Assim, a taxa de reação é controlada tanto pela reação química quanto pela difusão nos poros (HONG, 2000). A energia de ativação observada é metade do valor verdadeiro e a ordem aparente de reação, n , é relacionada à ordem verdadeira da seguinte forma: $n = (m + 1)/2$ (CUNHA, 2010).

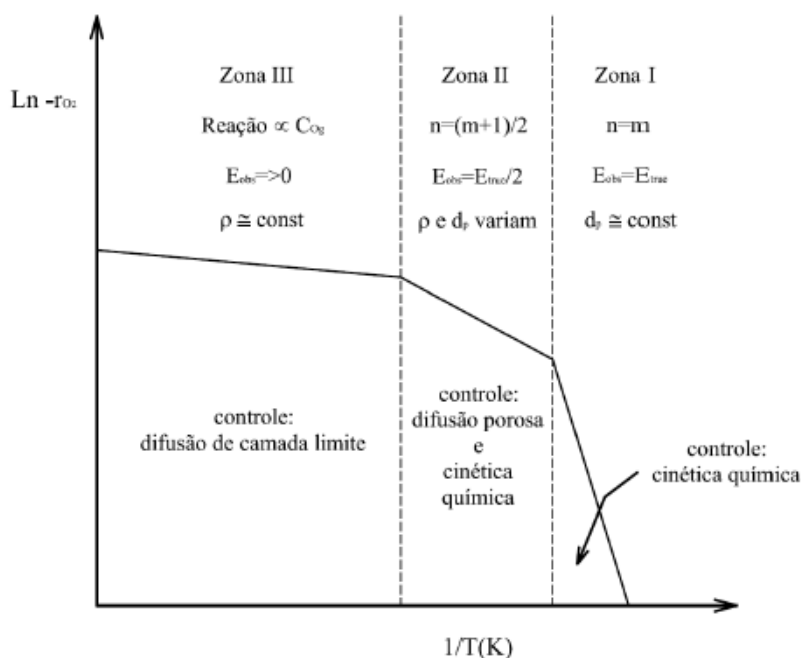


Figura 2 – Zonas de temperatura ou regimes de reação para a conversão de sólidos

Fonte: Cunha (2010)

A Zona III, que ocorre a altas temperaturas, é caracterizada por limitações de transferência de massa na camada limite da partícula (HONG, 2000). A taxa de reação química é tão rápida que a concentração do reagente se aproxima de zero, e a taxa global é limitada pela taxa de difusão dos reagentes (DUFFY, 2012). Nesta zona, a energia de ativação observada é baixa e a ordem da reação é unitária (CUNHA, 2010). O carvão se contrai a uma densidade constante, e a taxa de reação é proporcional à superfície externa da partícula e à um coeficiente de transferência de massa (DI BLASI, 2009).

Os modelos de oxidação do carvão podem ser divididos em duas categorias: modelos globais e modelos intrínsecos (SMITH et al., 1994). Segundo Hong (2000), os modelos globais consideram as partículas de carvão insensíveis aos efeitos da difusão nos poros ou então agregam tais efeitos nas constantes da taxa de reação química. Estes modelos são altamente empíricos, baseiam a taxa de reação global na área superficial externa da partícula e na concentração de oxidante na superfície desta. Os modelos intrínsecos relacionam a taxa de oxidação de carvão com a área superficial ativa envolvida na reação e consideram o perfil não-uniforme da concentração de oxidante no interior da partícula. Eles também necessitam de modelos da estrutura porosa para descrever a difusão gasosa pela complexa estrutura porosa e para modelar a concentração local de oxidante na área superficial ativa.

Uma equação de taxa de n -ésima ordem empírica é comumente utilizada para descrever a cinética de reações heterogêneas gás-sólido. A forma mais comum é uma lei

de potência em relação à pressão parcial do gás reagente (HURT; CALO, 2001; HONG, 2000; DI BLASI, 2009; BARRIO, 2002; BASU, 2013). Para as reações de combustão (Equação 2.39) e gaseificação (Equação 2.40 e Equação 2.41), a taxa de reação será:

$$r_{O_2} = k_1 P_{O_2}^{n_1} = A_1 e^{-E_1/RT} P_{O_2}^{n_1} \quad (2.42)$$

$$r_{CO_2} = k_2 P_{CO_2}^{n_2} = A_2 e^{-E_2/RT} P_{CO_2}^{n_2} \quad (2.43)$$

$$r_{H_2O} = k_3 P_{H_2O}^{n_3} = A_3 e^{-E_3/RT} P_{H_2O}^{n_3} \quad (2.44)$$

onde P_{O_2} , P_{CO_2} e P_{H_2O} são as pressões parciais de oxigênio, dióxido de carbono e vapor de água, respectivamente. Cada reação tem um valor para a energia de ativação, E , o fator pré-exponencial, A , e a ordem de reação, n .

Segundo Cunha (2010), a equação de taxa de n -ésima ordem fornece uma base para a estimativa da taxa de oxidação de coque e tem se mostrada adequada para uso prático em pressão atmosférica numa faixa estreita de temperatura e, portanto, não pode ser aplicada indiscriminadamente. Hurt e Calo (2001) também comentam que esse modelo é inapropriado como uma forma geral, porém encontra uso prático em limitadas faixas de temperatura e pressão. De acordo com Hong (2000), essa equação é comumente utilizada em modelos computacionais devido à sua simplicidade. Entretanto, a equação de taxa de n -ésima ordem não considera explicitamente os efeitos da difusão nos poros na cinética. Esses efeitos são implicitamente incluídos na energia de ativação e no fator pré-exponencial observados.

Um modelo mais elaborado para a cinética química de reações gás-sólido inclui mecanismos fundamentais que governam as reações de superfície. Esses mecanismos são baseados na teoria do sítio ativo, que descrevem a quimissorção (adsorção) de reagentes na superfície, migração de compostos intermediários e a dessorção de produtos a partir da superfície. Um modelo cinético de Langmuir-Hinshelwood é então utilizado para descrever a taxa de reação (LAURENDEAU, 1978; HONG, 2000).

A gaseificação do carvão com dióxido de carbono (Equação 2.40) pode ser representada pelo seguinte mecanismo (LAURENDEAU, 1978; DI BLASI, 2009; BARRIO; HUSTAD, 2001):



onde k_1 , k_2 e k_3 são as taxas de reação de Arrhenius, C_f representa um sítio de carbono disponível e $C(O)$ um sítio ocupado por um oxigênio atômico, também chamado de complexo de carbono-oxigênio. A presença de CO produz um efeito inibidor através da diminuição da concentração em estado estacionário de $C(O)$ pela [Equação 2.45b](#) ([DI BLASI, 2009](#); [BARRIO](#); [HUSTAD, 2001](#)). A taxa de gaseificação usando a cinética de Langmuir-Hinshelwood será ([DI BLASI, 2009](#); [BARRIO](#); [HUSTAD, 2001](#); [BASU, 2013](#)):

$$r_{CO_2} = \frac{k_1 P_{CO_2}}{1 + (k_2/k_3) P_{CO} + (k_1/k_3) P_{CO_2}} \quad (2.46)$$

onde P_{CO_2} e P_{CO} são as pressões parciais de CO_2 e CO , respectivamente.

A gaseificação do carvão com vapor de água ([Equação 2.41](#)) é mais complexa que a gaseificação com CO_2 por causa da ação de H_2 , CO_2 e CO devido ao equilíbrio da reação de *water-gas shift* $CO + H_2O = CO_2 + H_2$ ([BARRIO et al., 2001](#)). Basicamente, existem dois mecanismos para a gaseificação com vapor de água: o mecanismo de troca de oxigênio e o mecanismo de inibição por hidrogênio. Os passos envolvidos são ([BARRIO et al., 2001](#); [HÜTTINGER](#); [MERDES, 1992](#); [DI BLASI, 2009](#)):



O mecanismo de troca de oxigênio consiste dos passos descritos nas [Equações 2.47a](#), [2.47b](#) e [2.47c](#). O mecanismo tradicional de inibição por hidrogênio é baseado nas [Equações 2.47a](#), [2.47c](#), [2.47d](#) e [2.47e](#). Uma segunda versão do mecanismo de inibição por hidrogênio é baseado nas [Equações 2.47a](#), [2.47c](#), [2.47f](#) e [2.47g](#). A taxa de reação para os modelos

apresentados é semelhante, com a exceção da dependência sobre a pressão parcial de hidrogênio (BARRIO et al., 2001; DI BLASI, 2009):

$$r_{H_2O} = \frac{k_1 P_{H_2O}}{1 + (k_1/k_3) P_{H_2O} + f(P_{H_2})} \quad (2.48)$$

onde P_{H_2O} e P_{H_2} são as pressões parciais de H_2O e H_2 , respectivamente. $f(P_{H_2})$ depende do mecanismo escolhido. Para o mecanismo de troca de oxigênio, ela será a [Equação 2.49](#). Para o mecanismo tradicional de inibição por hidrogênio, pela formação do complexo $C(H)_2$, ela será a [Equação 2.50](#). Para a segunda versão do mecanismo de inibição por hidrogênio, pela formação do complexo $C(H)$, ela será a [Equação 2.51](#).

$$f(P_{H_2}) = \frac{k_2}{k_3} P_{H_2} \quad (2.49)$$

$$f(P_{H_2}) = \frac{k_4}{k_5} P_{H_2} \quad (2.50)$$

$$f(P_{H_2}) = \frac{k_6}{k_7} P_{H_2}^{0,5} \quad (2.51)$$

Para a combustão do carvão ($C + O_2 \rightarrow CO/CO_2$), [Hurt e Calo \(2001\)](#) modelaram mecanismos semi-globais baseados em leis cinéticas de Langmuir-Hinshelwood. O mais básico considera as seguintes reações:



onde a taxa de reação é dada por:

$$r_{O_2} = \frac{k_1 k_2 P_{O_2}}{k_1 P_{O_2} + k_2} \quad (2.53)$$

[Hurt e Calo \(2001\)](#) também propuseram um mecanismo cinético de triplo passo, que inclui uma reação entre oxigênio gasoso e o complexo de superfície $C(O)$ ([Equação 2.54b](#)). Nesse mecanismo, as três reações são as seguintes:



onde a taxa de reação é dada por:

$$r_{O_2} = \frac{k_1 k_2 P_{O_2}^2 + k_1 k_3 P_{O_2}}{k_1 P_{O_2} + k_3/2} \quad (2.55)$$

Nessa seção, tratou-se da modelagem das reações heterogêneas que ocorrem entre um gás e um sólido durante a conversão desse sólido. Segundo [Klose e Wölki \(2005\)](#), a conversão do carvão é mais complicada que a sua devolatização, pois esse é um processo heterogêneo onde a superfície é o local onde ocorrem as reações químicas. Mais informações sobre a modelagem de reações heterogêneas podem ser encontradas em [Laurendeau \(1978\)](#), [Walker, Rusinko e Austin \(1959\)](#), [Smoot e Smith \(1985\)](#), [Smith et al. \(1994\)](#), [Hong \(2000\)](#), [Di Blasi \(2009\)](#), [Hurt e Calo \(2001\)](#), [Bews et al. \(2001\)](#), [Barrio \(2002\)](#), [Basu \(2013\)](#), [Williams et al. \(2002\)](#), [Eaton et al. \(1999\)](#), [Smith \(1982\)](#), [Hurt e Haynes \(2005\)](#), [Di Blasi, Buonanno e Branca \(1999\)](#).

2.3 Estudos em CFD da Conversão Termoquímica de Combustíveis Sólidos

Como destacado por [Wang e Yan \(2008\)](#), as técnicas de modelagem baseadas na Dinâmica dos Fluidos Computacional (CFD) têm sido bastante difundidas no ramo da conversão termoquímica. Pesquisadores têm utilizado CFD para simular e analisar a performance de equipamentos de conversão termoquímica, como leitos fluidizados, leitos fixos, fornos de combustão, caldeiras e fornos rotativos. A seguir serão comentados alguns desses trabalhos.

[Fletcher et al. \(2000\)](#) desenvolveram um detalhado modelo CFD para simular o escoamento e as reações em um gaseificador de biomassa de leito arrastado utilizando o pacote CFX. As partículas de biomassa são modeladas por uma abordagem Lagrangiana conforme entram no gaseificador, sofrem devolatização e por fim gaseificação. A química é introduzida por um modelo cinético global envolvendo CH_4 , H_2 , CO , CO_2 , H_2O , N_2 e O_2 , e considera devolatização, conversão do carvão por reações heterogêneas e as reações

químicas na fase gasosa. Uma validação foi feita em relação à composição dos gases de saída.

Lathouwers e Bellan (2001a, 2001b) propuseram um modelo matemático para descrever a dinâmica de misturas reativas gás-sólido, e o aplicaram na simulação de pirólise de biomassa em leito fluidizado. Foi analisada a resposta do reator em relação à formação de alcatrão em função de vários parâmetros. Os resultados indicam que a temperatura do gás é o fator que mais influencia a quantidade de alcatrão formado, enquanto a temperatura e natureza da biomassa, e a velocidade de fluidização têm um impacto menor. Simulações paramétricas demonstram a capacidade de modelos CFD para identificar as condições ótimas de operação para o reator e auxiliam no processo de aumento de escala para reatores industriais.

Zhou, Flamant e Gauthier (2004a, 2004b) acoplaram um método discreto de partículas com o modelo de turbulência LES para descrever a combustão de carvão em um reator de leito fluidizado borbulhante. O modelo cinético para a combustão do carvão foi baseado em uma abordagem por energia de ativação distribuída em função da taxa de aquecimento. Foram estudados os efeitos do diâmetro das partículas, da temperatura do leito e da velocidade de entrada do gás. A presença de grandes partículas reativas no leito fluidizado pode afetar significativamente a velocidade do gás e das partículas e suas intensidades turbulentas. A temperatura das partículas excede a temperatura do leito, o que se mostra de acordo com os dados experimentais da literatura.

Yu et al. (2007) desenvolveram um modelo baseado na teoria cinética de fluxo granular e acoplaram com um modelo químico de vários estágios para estudar a gaseificação de carvão em um gaseificador de leito fluidizado. A composição dos gases de saída se mostrou em boa concordância com dados experimentais para várias condições operacionais.

Rostami, Murthy e Hajaligol (2004) modelaram o processo de combustão *smoldering* de um leito de combustível poroso. O modelo desenvolvido era transiente, bidimensional e axissimétrico, baseado nos princípios de conservação para a combustão de um bastão cilíndrico de combustível sólido poroso. A transferência de calor foi representada por duas equações de energia, uma para a fase gasosa e outra para a fase sólida. O material inicialmente sofria pirólise, que consistia de um conjunto de reações de primeira ordem, e em sequência o resíduo carbônico resultante sofria combustão. Cálculos sob condições transientes foram feitos para várias condições de operação e de contorno, assim como propriedades físicas do combustível e porosidade do leito. As simulações capturaram o desenvolvimento de uma região de combustão que se movia a uma taxa constante. O pico de temperatura e a velocidade da frente de combustão se mostraram dentro das faixas dos dados experimentais. Porém, os autores concluem que melhorias ainda precisam ser feitas para o modelo, principalmente com relação à perdas de calor.

Luo, Wang e Cen (2005) desenvolveram um modelo para a pirólise de madeira em

reatores de leito fluidizado, simulando os efeitos dos principais parâmetros operacionais na distribuição dos produtos da pirólise. O objetivo era prever o processo de pirólise rápida para produção de bio-óleo. O modelo mostrou que a temperatura de reação desempenha um papel fundamental na pirólise de madeira. Ele mostrou também que partículas menores que $500 \mu\text{m}$ conseguem atingir uma alta taxa de aquecimento, atingindo os requerimentos de uma pirólise rápida.

Lu (2006) utilizou experimentação em um reator de leito arrastado e modelagem numérica para analisar a pirólise e combustão de biomassa. O modelo era transiente e unidimensional, e simulava os processos de secagem, pirólise, e combustão e gaseificação do resíduo carbônico para partículas de vários formatos. O modelo contabilizava o encolhimento e o inchamento da partícula. As simulações para três formatos de partículas concordaram razoavelmente com os dados experimentais.

Saidi et al. (2007) desenvolveram um modelo tridimensional para simulação numérica da combustão *smoldering* de um combustível poroso que consistia de celulose ou tabaco (cigarro), embrulhado em um papel poroso e envolvido por ar ambiente. O modelo incluía o efeito de forças de empuxo no escoamento, e também tratamento separado para as fases sólida e gasosa, de forma a contabilizar o não equilíbrio térmico. As mudanças na porosidade do sólido devido à pirólise e à oxidação do carvão, e os efeitos de porosidade na permeabilidade do leito e difusividade dos gases foram incluídas no modelo. Segundo os autores, os resultados da simulação reproduziram razoavelmente as principais características observadas experimentalmente para a combustão de cigarro. O mecanismo de difusão se mostrou importante na diluição das espécies gasosas transportadas através do cigarro.

Gerun et al. (2008) desenvolveram um modelo CFD bidimensional axissimétrico para a zona de oxidação de um gaseificador *downdraft* de dois estágios para estudar o impacto da formação de alcatrão na temperatura e no perfil de velocidade. As simulações correspondem satisfatoriamente aos dados experimentais sobre perfil de temperatura e concentração de alcatrão.

Martins (2008) fez um estudo experimental e numérico da propagação de uma frente de combustão *smoldering* de xisto betuminoso em configuração co-corrente. O código utilizado foi desenvolvido pela equipe do Instituto de Mecânica dos Fluidos de Toulouse (IMFT) (LAPENE et al., 2007; LAPENE et al., 2008). O não equilíbrio térmico local foi considerado no modelo, resultando em duas equações para o transporte de energia, uma para a fase gasosa e uma para a fase sólida. As reações químicas consideradas foram as de combustão do carbono fixo e descarbonatação. Um estudo paramétrico foi realizado, variando-se a velocidade do ar e o tamanho da partícula. Esse estudo confirmou que a propagação da frente de combustão é controlada pelo fornecimento de O_2 , e demonstrou que o tamanho da partícula não influenciou na propagação da frente. As abordagens experimental e numérica combinadas tornaram possível avaliar também a espessura das

diferentes zonas de reação.

Cunha (2010) apresenta um modelo matemático apropriado para simular os principais processos termoquímicos que ocorrem na combustão e gaseificação de partícula sólida, em regime de baixo número de Reynolds. As equações de conservação de massa, energia e momentum (geometria bidimensional), transiente, são resolvidas numericamente por um método aprimorado de volumes finitos baseado em elementos finitos chamado *Control Volume Finite Element Method* (CVFEM). O modelo proposto foi validado e aplicado no estudo da secagem, pirólise e combustão de partícula de turfa e biomassa, mostrando excelente concordância com resultados experimentais e teóricos publicados na literatura.

Duffy (2012) desenvolveu um modelo numérico detalhado, transiente e bidimensional, para a predição da combustão em leito fixo de combustíveis sólidos. O modelo é validado utilizando dados existentes para taxas de ignição e perfis de espécies em um leito de biomassa. O modelo é então aplicado para investigar os fatores que influenciam o efeito de canalização no leito. Por fim, o modelo é estendido para uma geometria tridimensional para examinar medidas para a redução do desgaste da grelha, investigando a influência do formato da grelha, fluxo de entrada de ar e a espessura da camada de cinzas.

Gao et al. (2012) criaram um modelo de um gaseificador ciclone baseado no pacote de CFD Fluent. Modelos para a pirólise de serragem e combustão dos voláteis e do carvão foram adicionados ao modelo padrão. Ele fornece informações sobre a temperatura do gás no gaseificador e sobre a composição dos gases de saída. Os resultados mostraram a temperatura do gás e as concentrações seguem as tendências dos dados experimentais, porém, o modelo sub prevê as concentrações de CO e CO₂.

Jakobs et al. (2012) desenvolveram um modelo de um gaseificador de leito arrastado de alta pressão. Equações estacionárias para o balanço de massa, energia, momento e espécies foram solucionadas por volumes finitos. Foi analisado a qualidade de atomização dos injetores de fluido em função da velocidade do gás e pressão no reator.

Janajreh e Shrah (2013) investigaram a eficiência de conversão em um gaseificador *downdraft* de pequena escala utilizando madeira. A investigação experimental do campo de temperatura no gaseificador foi acompanhada de simulação numérica utilizando CFD com um modelo Lagrangiano para a evolução das partículas. A temperatura média dada pelo modelo CFD foi maior comparada à temperatura medida experimentalmente.

Uma revisão da literatura sobre a modelagem CFD de processos de conversão termoquímica de biomassa e carvão, e dos trabalhos publicados nessa área podem ser encontrados em Wang e Yan (2008) e Pepiot, Dibble e Foust (2010). Uma extensa revisão dos trabalhos sobre processos de combustão de biomassa foi exposto por Duffy (2012), e também por Yin, Rosendahl e Kaer (2008).

2.4 Introdução ao *solver* biomassGasificationFoam

O *solver* biomassGasificationFoam (Kwiatkowski et al. (2013), Kwiatkowski et al. (2014)) foi desenvolvido para o *software* de CFD de código livre OpenFOAM (versão 2.1.1) para a simulação abrangente dos processos físicos e termoquímicos de gaseificação e pirólise de biomassa em leito fixo.

Os *solvers* fornecidos na distribuição padrão do OpenFOAM para a modelagem de pirólise (fireFoam (FM GLOBAL, 2016)), reações químicas (reactingFoam) ou gaseificação em leito fluidizado (coalChemistryFoam) não são adequados para a simulação de processos termoquímicos em leito fixo, como a gaseificação de biomassa. Esses programas disponíveis não consideram a mútua influência entre o fluxo reativo e o sólido e as reações volumétricas que ocorrem no material poroso.

O fireFoam é um dos mais avançados *solvers* que incluem processos térmicos do sólido, e o seu foco é a modelagem de fogos e incêndios. Nele, o meio no qual ocorre a pirólise está ligado ao domínio computacional principal, onde o fluxo de gás é calculado, e interage com a fase gasosa apenas por meio das condições de contorno do domínio. Com essa abordagem, não há uma descrição natural do material poroso com o fluxo gasoso que flui através dele. No fireFoam, a pirólise é implementada como um fenômeno de superfície, e não um fenômeno volumétrico, o que limita a sua aplicação.

O reactingFoam é um *solver* para escoamentos reativos transientes, laminares ou turbulentos, onde ocorrem reações químicas. Nele, essas reações são volumétricas, porém ele considera apenas a fase gasosa. Assim, ele não se aplica aos processos de conversão termoquímica de sólidos. O coalChemistryFoam foi desenvolvido para a simulação de gaseificação em leito fluidizado, não se aplicando para processos de conversão em leito fixo.

Não existia então um *solver* para OpenFOAM adequado para a simulação da conversão termoquímica de sólidos em leito fixo. Interessados no estudo da gaseificação de biomassa, Kwiatkowski et al. (2013) desenvolveram o biomassGasificationFoam. Segundo Kwiatkowski et al. (2014), o objetivo principal desse código é proporcionar um ambiente computacional abrangente para uma ampla gama de aplicações envolvendo gases e sólidos reagentes, sendo ele um *solver* integrado capaz de modelar a conversão térmica, incluindo evaporação, pirólise, gaseificação e combustão, de vários materiais sólidos.

O código referido é composto de um *solver* (biomassGasificationFoam) e um conjunto de bibliotecas complementares (biomassGasificationMedia), baseados no código do reactingFoam e expandido para incluir a fase sólida porosa e os seus processos termoquímicos. O modelo inclui o escoamento da mistura gasosa reativa e a complexa transferência de massa e energia entre a fase gasosa e a fase sólida porosa. O Método dos Volumes Finitos é utilizado no código. O conjunto de equações governantes solucionadas compreende equações para ambas as fases, sendo elas a conservação de massa das espécies

gasosas e dos componentes sólidos, conservação do momento na fase gasosa, a conservação da energia para as duas fases, e a equação da continuidade para a fase gasosa.

[Kwiatkowski et al. \(2014\)](#) comentam que as principais funcionalidades implementadas no `biomassGasificationFoam` são as seguintes:

- Reações volumétricas no interior do meio poroso (a porosidade varia com a decomposição do sólido);
- Reações homogêneas e heterogêneas com o calor de reação definido diretamente ou derivado a partir das entalpias de formação;
- Uma definição flexível da biomassa e suas propriedades termoquímicas (essa flexibilidade permite tratar de diversos sólidos além da biomassa);
- Mecanismos cinéticos customizáveis para pirólise e gaseificação;
- Transferência de calor e massa entre os gases e sólidos no interior e no entorno do meio poroso;
- Escoamento transiente no meio poroso;
- Equações de energia separadas para as fases sólida e gasosa de forma a considerar os efeitos do não-equilíbrio térmico.

A validação do `solver biomassGasificationFoam` por [Kwiatkowski et al. \(2013\)](#) foi feita pela simulação de um teste de análise termogravimétrica (TGA) de uma madeira (*Rubinia pseudoaccacia*), cujos resultados foram comparados à um teste experimental. Um teste TGA, por meio da medição da massa, quantifica a decomposição térmica de um material sólido em condições controladas. A temperatura inicial foi de 300 K, com uma taxa de aquecimento constante de 10 K/min, finalizando quando atingisse a temperatura de 1300 K. O ambiente fornecido foi de nitrogênio puro, e portanto, apenas os processos de secagem e pirólise ocorreram.

As simulações foram realizadas para uma geometria simplificada do reator TGA e da amostra, ambos modelados como cubos, como mostrado na [Figura 3](#). O reator foi considerado com um lado de 8 mm de comprimento. O domínio computacional completo foi composto de 20 x 20 x 20 células. O meio poroso (amostra) foi distribuído em 8 x 8 x 8 células.

O mecanismo de pirólise utilizado foi o mecanismo de reações múltiplas de simples estágio baseado na composição do sólido (comentado na [subseção 2.2.2.1.2](#) e exemplificado na [Equação 2.28](#)), levando em conta que a madeira é composta dos componentes celulose, hemicelulose e lignina. Cada um desses componentes se decompõe em um conjunto de

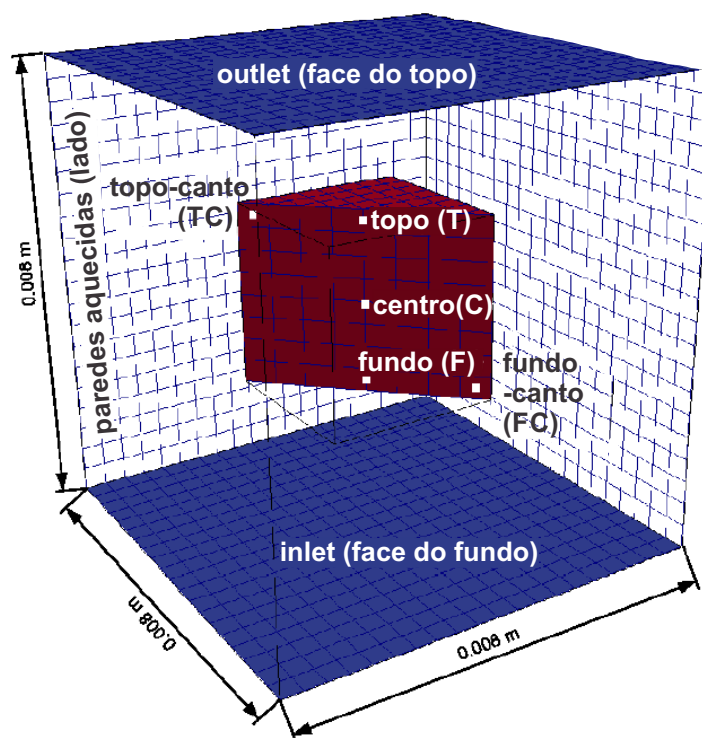


Figura 3 – Reator TGA simplificado de geometria cúbica com um amostra hexagonal de madeira no centro

Fonte: Adaptado de Kwiatkowski et al. (2013)

voláteis e um resíduo sólido carbônico. Esses voláteis são considerados apenas como os gases H_2 , CO , CO_2 e CH_4 . O calor de reação foi determinado pelas entalpias de formação dos produtos e dos reagentes.

A comparação entre a perda de massa durante a secagem e pirólise no experimento TGA e a correspondente simulação está apresentado na Figura 4. O resultado apresenta algumas discrepâncias, mas tem uma concordância qualitativa. Segundo Kwiatkowski et al. (2013), as principais razões para essas discrepâncias são a cinética imperfeita e simplificada da pirólise e a variabilidade da composição da madeira, que não pode ser determinada com exatidão. É possível perceber que uma pequena quantidade de carvão continua sofrendo devolatização em temperaturas elevadas, acima de 800 K, efeito esse que não está incluso na cinética de pirólise utilizada.

A Figura 5 apresenta a taxa de perda de massa nos cinco pontos descritos na Figura 3. Os picos na curva correspondem à evaporação e pirólise da hemicelulose, da celulose e da lignina, respectivamente. Todos os processos são finalizados em aproximadamente 800 K. A taxa de perda de massa nos cinco pontos são praticamente indistinguíveis.

A Figura 6 ilustra como as massas dos componentes do sólido (umidade, hemicelulose, celulose e lignina) decaem com o processo enquanto a massa de carvão aumenta. A reprodução dessa conhecida sequência de decomposição é possível com o uso da cinética

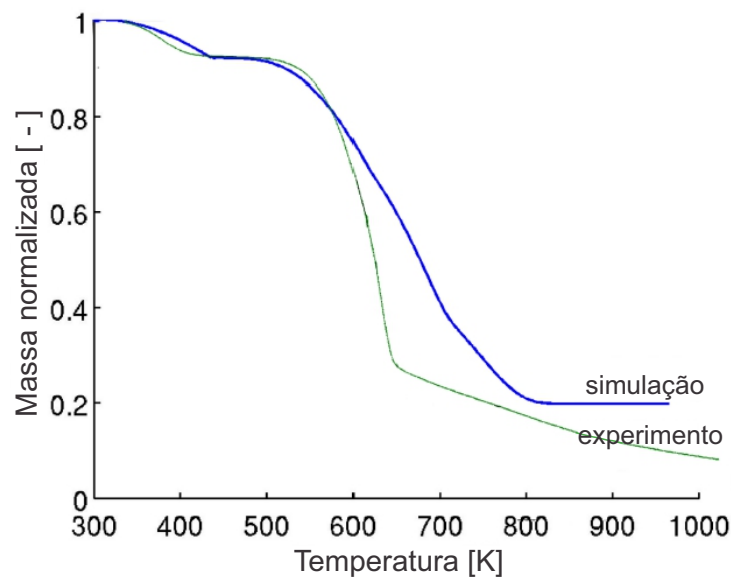


Figura 4 – Comparação da perda de massa durante o experimento TGA e a simulação correspondente

Fonte: Adaptado de Kwiatkowski et al. (2013)

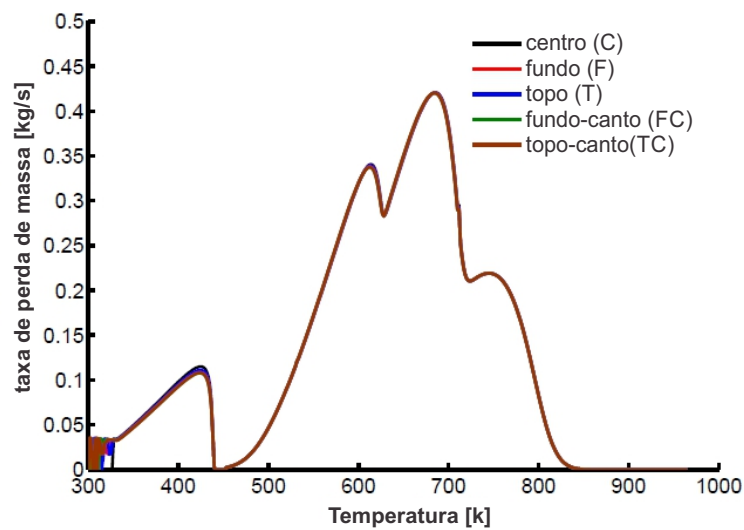


Figura 5 – Taxa de perda de massa em cinco pontos da amostra

Fonte: Adaptado de Kwiatkowski et al. (2013)

química separadamente para cada componente do sólido.

A composição dos gases produzidos está representada na [Figura 7](#). Os valores apresentados são relativos, pelo processo acontecer em um reator cheio com nitrogênio. De acordo com [Kwiatkowski et al. \(2013\)](#), o *solver* com a cinética utilizada reproduz adequadamente as frações mássicas relativas e os picos do processo.

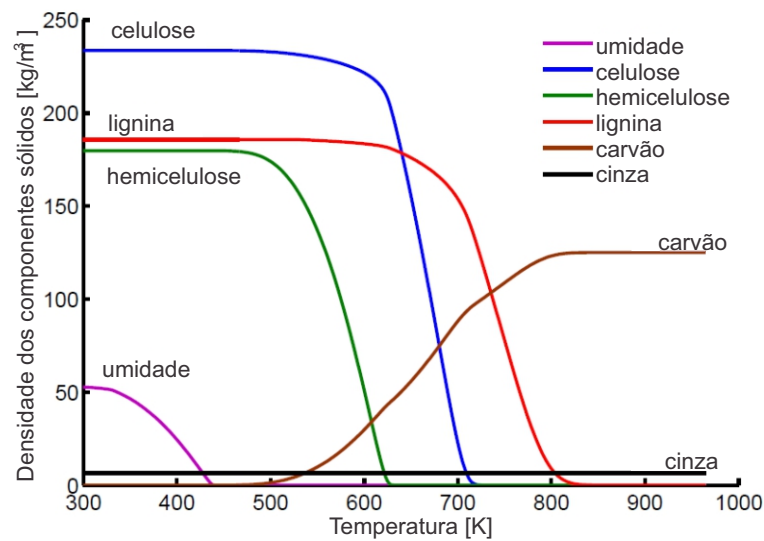


Figura 6 – Variação na massa dos componentes do sólido

Fonte: Adaptado de Kwiatkowski et al. (2013)

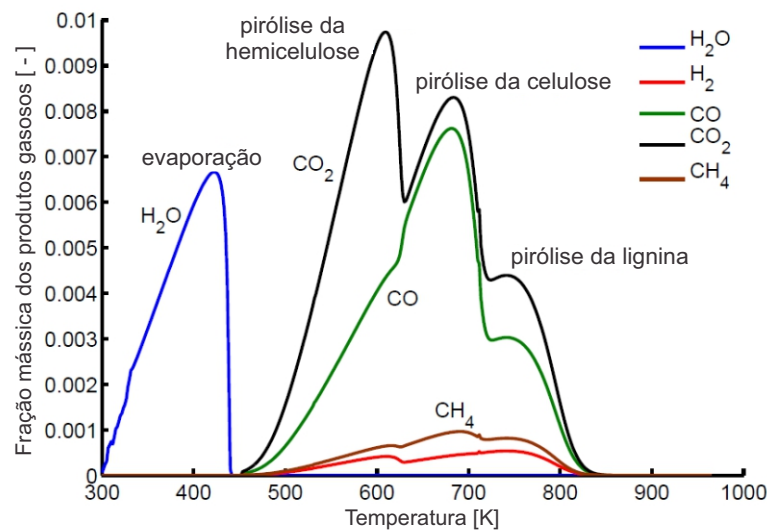


Figura 7 – Fração mássica dos gases

Fonte: Adaptado de Kwiatkowski et al. (2013)

Dernbecher, Tabet e Ortwein (2015) utilizaram o *solver* biomassGasificationFoam para simular a combustão de palha durante uma análise termogravimétrica (TGA), cuja geometria e malha foram as mesmas usadas por Kwiatkowski et al. (2013). Também foi simulada a combustão da mesma madeira de Kwiatkowski et al. (2013), para comparação dos resultados. Para tal, modificaram as propriedades do sólido e adicionaram reações para a gaseificação do carvão, combustão do carvão e combustão homogênea da fase gasosa (aproximada por uma reação global de simples estágio para a combustão do metano). O

sólido (palha) também foi considerado como composto por celulose, hemicelulose e lignina, porém com as correspondentes frações mássicas adequadas para esse caso.

Como meio para o reator TGA foi utilizado ar (79% de nitrogênio e 21% de oxigênio). A temperatura inicial foi de 300 K, com uma taxa de aquecimento de 1 K/s. Ao atingir 1300 K, essa temperatura foi mantida constante. A [Figura 8](#) mostra a comparação da massa normalizada da madeira e da palha durante a análise termogravimétrica. A evaporação é o primeiro processo a acontecer, e ocorre quase simultaneamente para as duas amostras. Durante a pirólise da hemicelulose e da celulose, a palha decompõe mais rapidamente que a madeira. Durante a combustão do carvão resultante, a massa de madeira cai mais rapidamente. Por ter um menor conteúdo de cinza, a massa restante da madeira é menor.

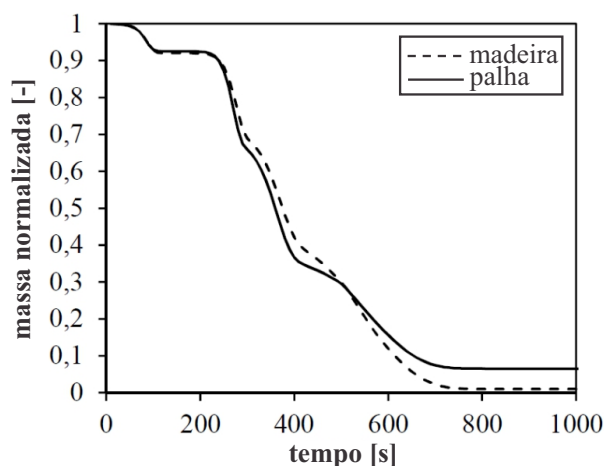


Figura 8 – Massa normalizada de partícula de palha e madeira durante a combustão

Fonte: Adaptado de [Dernbecher, Tabet e Ortwein \(2015\)](#)

Na [Figura 9](#) é mostrada a decomposição da madeira em um ponto no centro da partícula. A [Figura 10](#) mostra a decomposição da palha. Pode-se ver que a palha possui um menor conteúdo de lignina e mais cinzas que a madeira. Em torno de 500 segundos (800 K), a curva de massa para o carvão (char) começa a diminuir, mostrando que a partir desse ponto o consumo de carvão por combustão e gaseificação é maior que a sua formação pelos processos de pirólise.

Por último, foi estudado o perfil dos gases CH_4 , H_2 , CO e CO_2 liberados durante a combustão da madeira e da palha. A [Figura 11](#) mostra um exemplo desses gráficos para a liberação de CH_4 . É possível ver os picos de liberação desse gás nos processos de pirólise da hemicelulose, da celulose e da lignina, respectivamente.

[Dernbecher, Tabet e Ortwein \(2015\)](#) concluíram que o *solver biomassGasification-Foam* oferece muitos recursos e tem importantes mecanismos implementados, mostrando-se útil como código CFD para a combustão de biomassa no OpenFOAM. É possível simular a combustão de vários tipos de biomassa mudando apenas os dados das propriedades. No

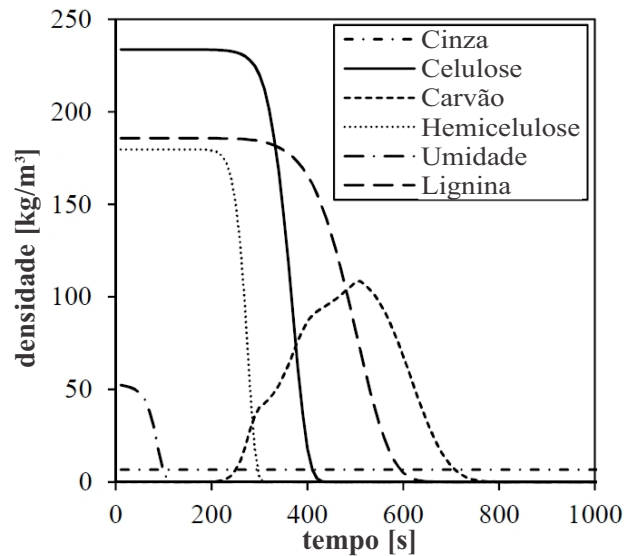


Figura 9 – Massa dos componentes sólidos durante a combustão de madeira

Fonte: Adaptado de [Dernbecher, Tabet e Ortwein \(2015\)](#)

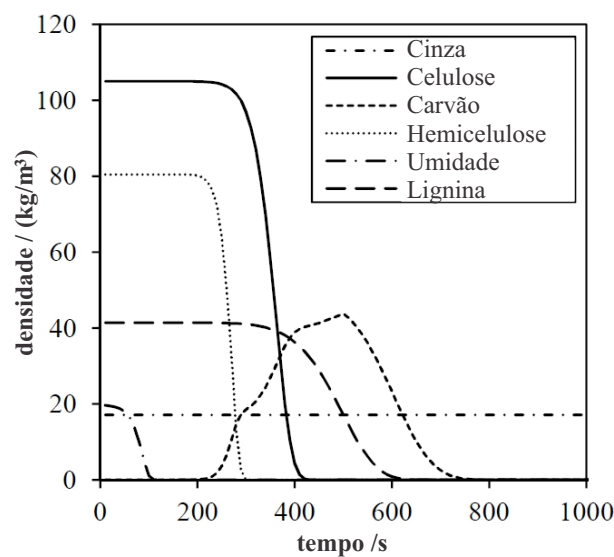
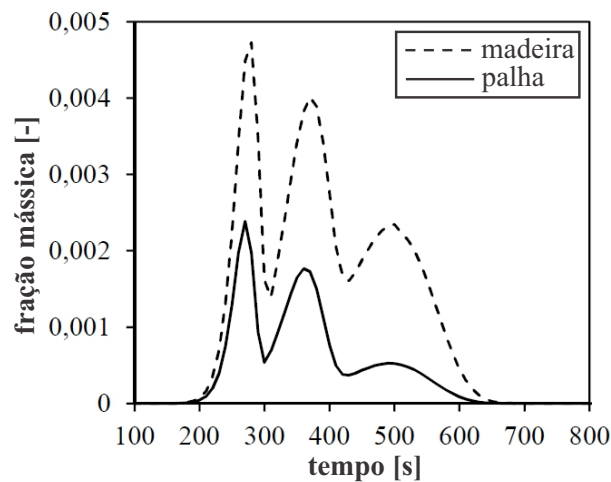


Figura 10 – Massa dos componentes sólidos durante a combustão de palha

Fonte: Adaptado de [Dernbecher, Tabet e Ortwein \(2015\)](#)

entanto, na visão dos autores, melhorias em relação aos modelos de pirólise e mecanismos de combustão e gaseificação ainda são necessárias.

Uma aplicação do `biomassGasificationFoam` para um processo mais complexo que uma análise termogravimétrica foi feita por [Barbosa \(2014\)](#), que utilizou esse *solver* para simular a combustão em uma caldeira a *pellets* de biomassa, com o objetivo de realizar a modelagem do processo, estudar o comportamento dessas caldeiras para otimização do seu projeto e condições de operação, bem como criar uma base para avaliar a validade de

Figura 11 – Liberação de CH_4 durante a combustãoFonte: Adaptado de [Dernbecher, Tabet e Ortwein \(2015\)](#)

tecnologias emergentes no mercado.

A geometria escolhida para a câmara de combustão da caldeira foi a de um corpo cilíndrico, com admissão de ar pela parte inferior e saída pela face superior, como mostrado na [Figura 12](#), onde a parte preenchida com biomassa está representada em azul.

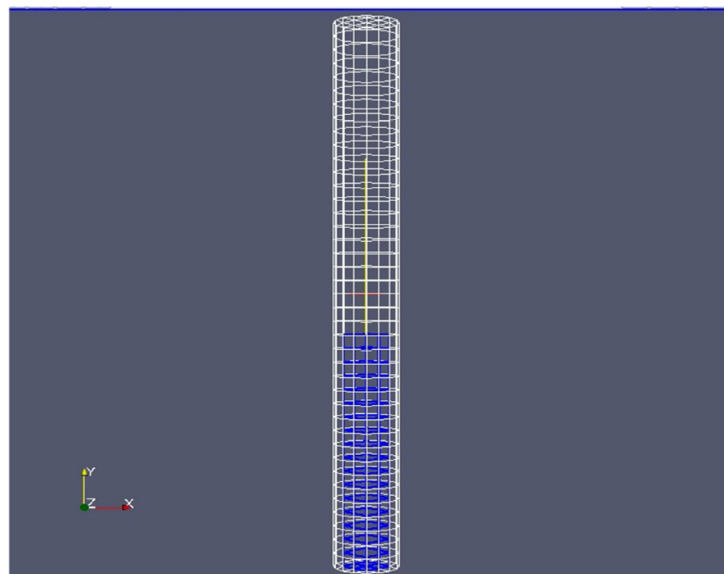


Figura 12 – Malha correspondente à câmara de combustão, com a biomassa representada em azul

Fonte: [Barbosa \(2014\)](#)

A biomassa utilizada na simulação foi a mesma madeira usada por [Kwiatkowski et al. \(2013\)](#), composta pelas respectivas frações mássicas de celulose, hemicelulose, lignina, umidade e cinzas. As reações consideradas foram as de desumidificação, pirólise dos três sub-componentes da madeira, gaseificação do carvão resultante da pirólise, e combustão

em fase gasosas dos gases liberados. A energia relacionada à essas reações foi calculada por meio das entalpias de formação dos produtos e dos reagentes.

Buscando uma aplicação mais realista, [Barbosa \(2014\)](#) adaptou o código de ignição usado pelo *XiFoam* para realizar a ignição da biomassa, que permite ativar a ignição por um período de tempo pré-estabelecido e com uma força determinada (por meio do aumento da entalpia nas células escolhidas). Dessa forma, quando a ignição fosse desligada esperava-se que a combustão continuasse de forma autossustentável. Contudo, verificou-se que isso não acontecia, a combustão parava sempre que desligava a fonte de ignição. Assim, [Barbosa \(2014\)](#) estudou outros dois métodos para realizar a ignição da biomassa. O primeiro era a alimentação de uma corrente de ar quente, e a segunda era o uso de uma resistência de aquecimento. Porém, esses dois métodos também se mostraram incapazes de obter a autossustentabilidade do processo de combustão.

O efeito da insustentabilidade da combustão é exemplificado na [Figura 13](#), que mostra a variação da temperatura do sólido no tempo para uma simulação na qual a ignição é desativada aos 800 segundos. A partir desse tempo, a temperatura do sólido cai rapidamente.

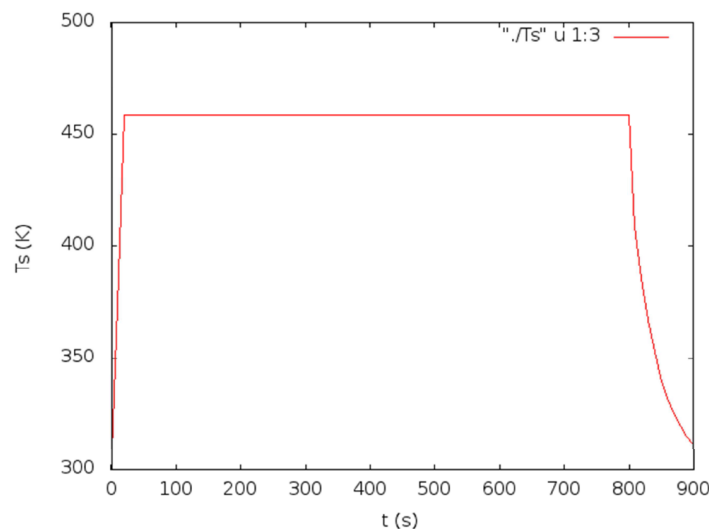


Figura 13 – Variação da temperatura do sólido em uma simulação na qual a ignição é desativada aos 800 s

Fonte: [Barbosa \(2014\)](#)

[Barbosa \(2014\)](#) enfrentou problemas de convergência e instabilidade nas suas simulações, atribuídos a problemas numéricos que ocorrem na resolução das equações de conservação de energia, principalmente devido às diferenças nas propriedades térmicas do sólido e do gás. Para tentar ultrapassar os problemas de convergência e de autossustentação da combustão foram testadas várias hipóteses, que incluíram diferentes composições da biomassa, diferentes cinéticas, calores de reação, parâmetros de transferência de calor, velocidades do ar de alimentação, malhas, esquemas de resolução numérica do sistema

de equações diferenciais, e diferentes parâmetros desses esquemas de resolução. Todas essas alternativas não apresentaram melhores resultados. Em seu trabalho, os resultados mostrados correspondem apenas aos obtidos com os modelos cinéticos mais realistas.

Alguns dos resultados obtidos para o caso da ignição com uso da resistência de aquecimento são mostrados a seguir. A Figura 14 mostra a variação da temperatura do sólido (T_s) na parte inferior, intermediária e superior da biomassa. Na célula inferior, que está em contacto com a resistência (curva 1:2), a temperatura aumenta rapidamente, estabilizando nos 650 K ao fim de 20.000 s. As frações de metano, hidrogênio e dióxido de carbono são praticamente nulas, inferiores a 5×10^{-4} , como mostrado na Figura 15 para o metano. A fração de monóxido de carbono, embora pequena, é superior às dos outros voláteis. Estas pequenas frações mássicas dos gases que resultam da reação de pirólise da biomassa indicam que as reações de combustão em fase gasosa não ocorrem, ou ocorrem em pequena extensão. Esse resultado é consistente com os valores da massa de carvão (Figura 16) e cinzas, que dão uma indicação das extensão das reações na fase sólida, e em que se verifica que a massa de carvão é praticamente nula.

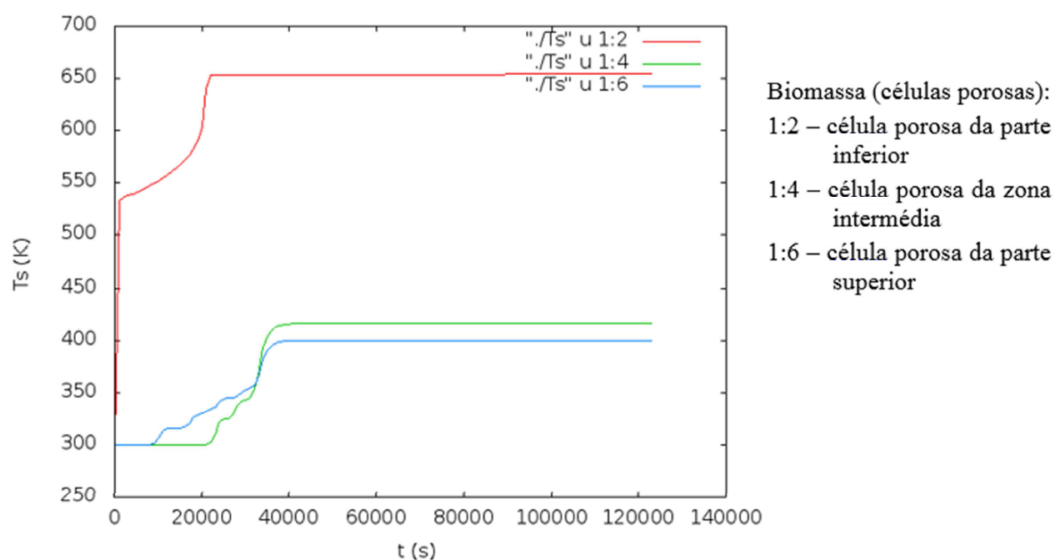


Figura 14 – Variação na temperatura da biomassa com o uso da resistência de aquecimento

Fonte: Barbosa (2014)

Por fim, com o uso do `biomassGasificationFoam`, Barbosa (2014) não conseguiu um resultado satisfatório para a simulação da combustão em uma caldeira a *pellets* de biomassa, e sugeriu uma análise mais profunda da modelagem desse *solver* e como ela está implementada. Para isso, seria necessária uma análise do código e conhecimento de programação em C++.

Esses estudos demonstram algumas das características que tornam o `biomassGasificationFoam` uma ferramenta interessante para a simulação de processos de conversão

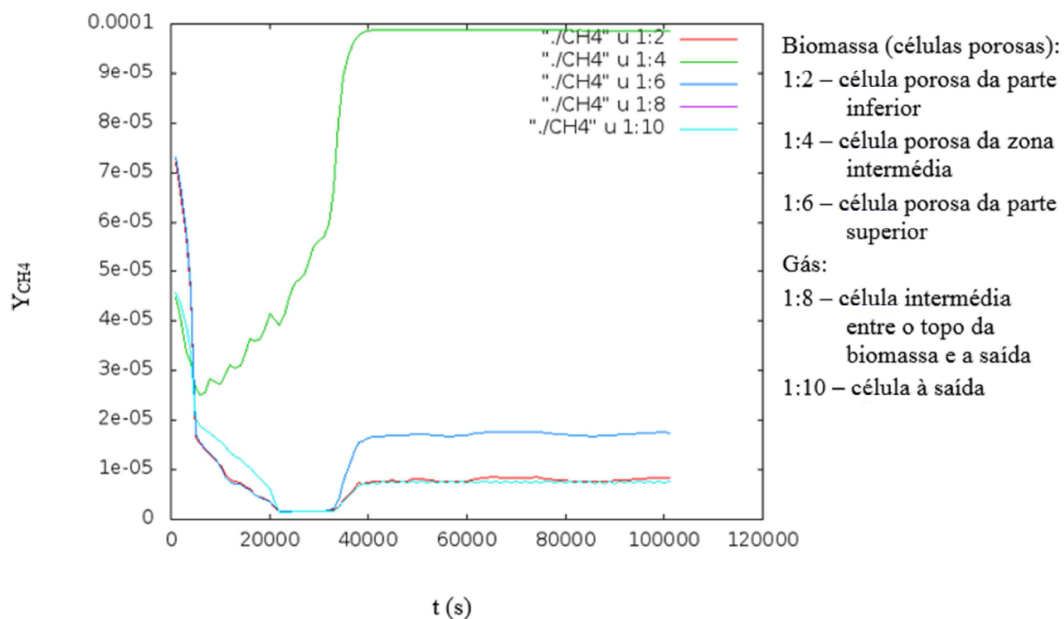


Figura 15 – Variação da fração mássica de metano com o uso da resistência de aquecimento

Fonte: Barbosa (2014)

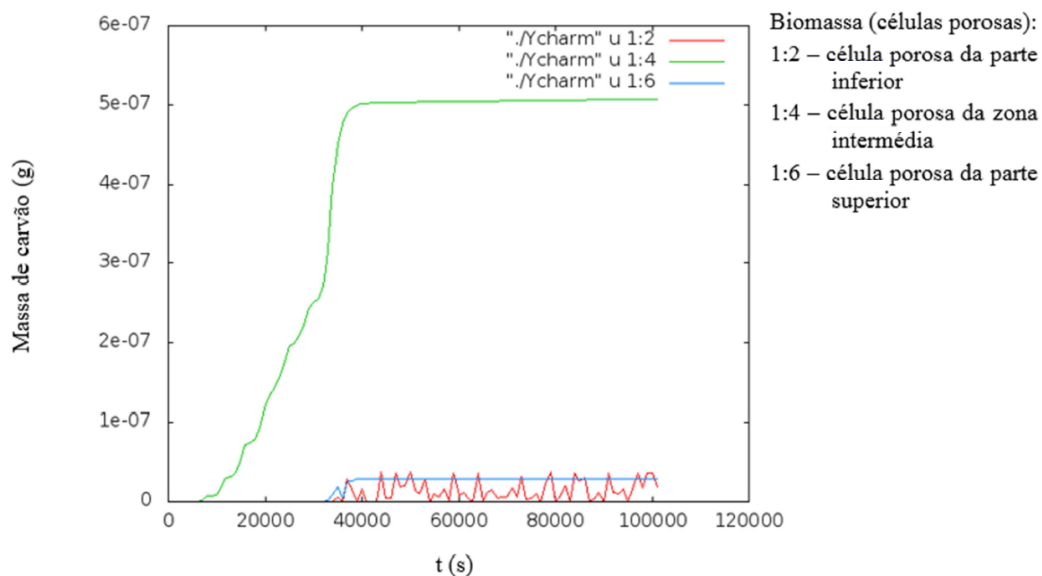


Figura 16 – Variação da massa de carvão com o uso da resistência de aquecimento

Fonte: Barbosa (2014)

termoquímica de sólidos em leito fixo, como a flexibilidade para a definição da cinética química e para a definição do sólido. Porém, algumas dificuldades encontradas por Barbosa (2014) levantam a necessidade de uma análise mais detalhada dos modelos implementados nesse *solver*, como o calor de reação, a cinética química, e a transferência de calor entre as fases sólida e gasosa.

3 O *software* OpenFOAM

OpenFOAM (Open Field Operation And Manipulation) é um pacote de simulação gratuito e de código aberto, criado para resolver problemas da mecânica dos meios contínuos, principalmente para a dinâmica dos fluidos computacional. A tecnologia central do OpenFOAM é um conjunto de módulos escritos em C++, que são usados para criar aplicativos executáveis. Estes podem ser: *solvers*, projetados para resolver problemas específicos de engenharia; utilitários, para realizar tarefas de pré e pós-processamento, que abrangem desde a manipulação dos dados à visualização, construção e processamento de malhas; e bibliotecas de expansão usadas pelos *solvers* e utilitários, como uma biblioteca de modelos físicos.

O desenvolvimento do OpenFOAM teve início em 1993, com Henry Weller e Hrvoje Jasak, alunos de doutorado da Imperial College, em Londres. O interesse deles era criar um código geral, de qualidade, confiável e eficiente, que pudesse ser utilizado no futuro por outras pessoas. Ambos haviam trabalhado com códigos de CFD escritos em Fortran, mas sentiram que essa linguagem não era mais apropriada para o tipo de trabalho que queriam fazer, podendo torná-lo algo que ninguém mais utilizaria, assim fazendo sua atenção se voltar para uma linguagem mais nova, o C++. Tudo isso levou os dois a unirem esforços para desenvolver o código do FOAM (Field Operation And Manipulation).

Por alguns anos, o FOAM foi desenvolvido em uma linha comercial pela empresa Nabla Ltd, com a ideia de fornecer ao usuário final uma ferramenta para simulações CFD de processos fisicamente complexos e oferecendo serviços de consultoria sobre o uso do software e implementação de novos solvers com modelagem complexa. Contudo, os desenvolvedores chegaram à conclusão de que a abordagem comercial não era a ideal. Assim, em 12 de dezembro de 2004, o código do FOAM se tornou de domínio público sobre a GPL (General Public License) e o programa passou a ser chamado de OpenFOAM (referente ao código ser aberto). Segundo os desenvolvedores, esta nova fase de desenvolvimento do código começa com o fim de sua abordagem comercial, reconhecendo que o OpenFOAM é essencialmente uma ferramenta de pesquisa e que o futuro do código só pode ser garantido através de colaborações e interações com o meio acadêmico (SILVA, 2008).

A partir dessa data, o OpenFOAM foi distribuído e desenvolvido pela empresa OpenCFD Ltd. Em 15 de agosto de 2011, a OpenCFD foi comprada pela Silicon Graphics International (SGI). Então, em 12 de setembro de 2012, o grupo ESI adquiriu a OpenCFD Ltd da SGI, e atualmente distribui o OpenFOAM pela OpenFOAM Foundation.

3.1 Escolha do OpenFOAM como pacote CFD

O fato de o OpenFOAM ser um pacote computacional de código aberto permite que o usuário tenha acesso a todo o seu código, não apresentando “caixas pretas” como os *softwares* comerciais. Assim, o usuário pode desenvolver novos *solvers* e utilitários, como também implementar novas bibliotecas, como condições de contorno e funções de parede. Isso torna possível que o OpenFOAM seja adaptado para as necessidades específicas de determinados casos, tornando-o uma ferramenta de simulação para praticamente qualquer tipo de problema de engenharia. Atualmente, o pacote padrão do OpenFOAM já vem com uma vasta quantidade de *solvers* e utilitários, que abrangem a maioria dos problemas clássicos, como escoamento de fluidos, transferência de calor, fluidos multifásicos, combustão, análise de tensões mecânicas e até eletromagnetismo, porém a adaptação é possível para problemas inovadores.

O OpenFOAM é distribuído gratuitamente sob a licença GNU General Public License, podendo ser executado em distribuições Linux (Debian, Ubuntu, CentOS, Fedora, etc.) tanto em versões de 32 bits como de 64 bits. Sendo assim, requer que o usuário tenha conhecimentos básicos de Linux, o que pode aumentar a dificuldade para iniciantes.

Um grande diferencial do OpenFOAM comparado a outros *softwares* gratuitos é que ele fornece ferramentas de pré e pós-processamento dos dados. Apesar de já existirem vários programas específicos para geração de malha e visualização de dados, o usuário deve despende de um grande esforço para interligar estes com o código CFD, o que é muito facilitado com o uso do OpenFOAM.

Outro ponto importante é a enorme capacidade para processamento em paralelo, visto que a licença não é limitada a certa quantidade de aplicativos ou núcleos. Sendo assim, pode-se aproveitar toda a capacidade de grandes *clusters* sem que usuários tenham que dividir o número de licenças, como em *softwares* comerciais.

Silva (2007, 2008) listou uma série de vantagens na utilização do OpenFOAM:

- Código aberto e escrito em C++. A grande maioria dos manuais dos pacotes CFD comerciais contém dados incompletos sobre a modelagem e implementação numérica. Ao analisar o código, o usuário é capaz de obter todas estas informações. E mais, ele pode alterar o código existente (ou criar um novo) para atender às necessidades do problema.
- Ferramentas gratuitas de geração de malha e visualização de dados incorporados ao pacote.
- Generalidade da malha (estruturada ou não-estruturada), inclusive com a possibilidade de importar malhas de outros programas (gratuitos e comerciais).

- Ampla faixa de aplicação na engenharia, incluindo escoamentos turbulentos, com troca de calor, multifásico, etc.
- Multi-plataforma (Linux, Solaris, MacOS, Linux 64 bits, etc). Existe uma versão adaptada para Windows.
- Possibilidade de executar simulações de grande porte em um *cluster* de computadores. Implementações cada vez mais eficientes de paralelismo estão sendo incorporadas ao OpenFOAM.
- Ferramentas de exportação de resultados para visualização em outros programas gráficos, o que facilita a obtenção dos resultados utilizando o *software* que mais agrade ou se encaixe nas necessidades do usuário.
- Desenvolvimento tende a ser realizado pelo meio acadêmico. A tecnologia mais avançada em termos de modelagem CFD, técnicas numéricas (geração de malha, métodos de discretização, solução de sistemas lineares, etc) e computação científica (visualização e paralelismo) provêm do meio acadêmico.

O OpenFOAM também apresenta algumas dificuldades e desvantagens em seu uso. Vários motivos tornam longa a sua curva de aprendizado e ele não possui uma interface gráfica de interação com o usuário. A falta de uma documentação detalhada é um outro ponto de dificuldade, principalmente para usuários iniciantes, e acaba requerendo um conhecimento ao menos básico da linguagem de programação C++ para uma melhor compreensão de seus códigos fontes. O OpenFOAM é escrito em uma linguagem C++ de alto nível, com uma programação orientada ao objeto. Se o usuário quiser se aprofundar nesses códigos, seja para um melhor entendimento, para realizar mudanças ou implementar novas funcionalidades, poderá ser necessário compreender as várias características dessa linguagem, como classes, hierarquia de classes e herança, funções virtuais, *overloading* de operadores, ponteiros, referências e *templates*.

Pelo fato de o OpenFOAM ser utilizado no Linux, usuários que não estejam familiarizados com esse sistema operacional podem sentir mais uma dificuldade inicial quanto ao seu uso, como por exemplo, para a instalação, compilação de códigos e comandos no terminal.

O OpenFOAM aceita malhas estruturadas e não-estruturadas, tetraédricas ou hexaédricas. Porém, como os outros programas de simulação, demonstra uma certa sensibilidade à malha, sendo importante sempre garantir a sua qualidade.

Ele também fornece ao usuário uma boa liberdade e variedade de escolha quanto aos métodos de discretização e parâmetros de solução, ficando a cargo do usuário fazer as escolhas mais adequadas para o seu problema. Isso pode se mostrar bastante complicado,

pois algumas escolhas podem levar à não-convergência da solução, e geralmente o usuário necessita de várias tentativas para chegar à melhor configuração. Assim, o OpenFOAM requer do usuário um conhecimento mais profundo dos métodos numéricos envolvidos no processo de solução, e novamente a falta de documentação detalhada pode ser um problema.

Se o leitor desejar um conhecimento mais profundo dos métodos numéricos implementados no OpenFOAM, é aconselhável a leitura do trabalho de [Jasak \(1996\)](#), um dos idealizadores do OpenFOAM. As principais fontes de informação sobre o OpenFOAM são os dois manuais que veem em seu pacote, o Guia do Usuário ([OpenFOAM User Guide \(2012\)](#)) e o Guia do Programador ([OpenFOAM Programmer's Guide \(2012\)](#)), presentes no diretório *doc* do local de instalação do OpenFOAM (`$WM_PROJECT_DIR/doc`). Existe também um manual não oficial criado por Gerhard Holzinger chamado “OpenFOAM A little User-Manual” ([Holzinger \(2016\)](#)).

Alguns outros trabalhos recomendados são os de [Weller et al. \(1998\)](#), [Silva \(2007\)](#), [Silva \(2008\)](#), [Jasak, Jemcov e Tukovic \(2007\)](#) e [Jasak \(2009\)](#).

Algumas páginas na internet também se mostram bastante úteis para a obtenção de informações e dicas:

- [<http://openfoam.org/>](http://openfoam.org/). Site oficial da OpenFOAM Foundation.
- [<https://openfoamwiki.net/index.php/Main_Page>](https://openfoamwiki.net/index.php/Main_Page). É uma página do tipo *Wiki*, ou seja, uma enciclopédia gratuita e comunitária. Nela estão contidas não apenas informações sobre o OpenFOAM, mas também sobre ferramentas desenvolvidas pela comunidade para o OpenFOAM, como o *pyFoam* e o *swak4Foam*.
- [<http://www.cfd-online.com/>](http://www.cfd-online.com/). Site que contém muitas informações sobre CFD, incluindo um excelente fórum, no qual se encontram dicas e discussões valiosas.

3.2 Estrutura de um caso

Cada caso a ser simulado no OpenFOAM segue uma estrutura de diretórios que contém os arquivos que configuram o mesmo. Estes arquivos possuem as informações necessárias para simular o caso, como a descrição da geometria, detalhes da malha e condições de contorno, parâmetros sobre os métodos numéricos, assim como as propriedades físicas do problema. A estrutura de diretórios pode ser vista na [Figura 17](#):

O diretório raiz de um caso irá conter os subdiretórios *system*, *constant* e os diretórios de tempo:

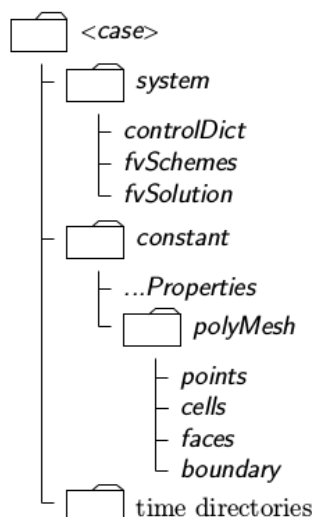


Figura 17 – Estrutura de um caso no OpenFOAM

Fonte: [OpenFOAM User Guide \(2012\)](#)

- *Constant*: deve conter os arquivos de propriedades físicas pertinentes ao caso, por exemplo, *transportProperties*, *turbulenceProperties* e *thermophysicalProperties*. A descrição completa da geometria e da malha deve ser incluída no subdiretório *polyMesh*.
- *System*: contém os arquivos que estão associados com o procedimento de solução do caso. Aqui devem estar contidos pelo menos três arquivos: *controlDict*, onde se define os parâmetros de controle da simulação, como os tempos de início e término da simulação, intervalo de tempo e controle de escrita de dados; *fvSchemes*, onde se seleciona os procedimentos de discretização usados na solução do problema; e *fvSolution*, que seleciona os métodos para resolver o sistema de equações lineares e suas tolerâncias, assim como outros parâmetros de controle do algoritmo de solução.
- *Diretórios de tempo*: contém os arquivos individuais de dados para os campos das variáveis tratadas no caso. Estes dados podem ser os valores iniciais e as condições de contorno que o usuário deve especificar para definir o problema (diretório “0”), ou os resultados da simulação, escritos em arquivo pelo OpenFOAM. O nome de cada diretório de tempo refere-se ao instante simulado em que os dados foram escritos.

3.3 Pré-processamento

O pré-processamento no OpenFOAM consiste na definição dos arquivos contendo o controle das condições de simulação e as propriedades físicas e modelos adicionais do problema. Nessa etapa também é definido o domínio computacional e realizada a geração

da malha. A modificação dos parâmetros da simulação deve ser feita editando-se os arquivos do caso, por meio de programas editores de texto como o *gedit*.

3.3.1 Geração da malha

O OpenFOAM sempre opera em um sistema de coordenadas cartesianas de três dimensões, sendo que todas as geometrias são geradas em 3D. Por padrão, ele resolve o caso em três dimensões, mas pode ser instruído para resolver em duas especificando-se uma condição de contorno especial “*empty*” para o contorno normal à dimensão na qual não é necessária nenhuma solução.

Para gerar a geometria, o OpenFOAM não apresenta um editor CAD. Porém, ele possui dois utilitários para gerar a malha, que são o `blockMesh` e o `snappyHexMesh`. Outra opção é a importação de malhas geradas em outros programas. Vale a pena comentar que, com o lançamento do OpenFOAM versão 2.3.0, foi lançado um novo utilitário para geração de malhas, o `foamyHexMesh`. Entretanto, ele não será comentado neste texto.

3.3.1.1 `blockMesh`

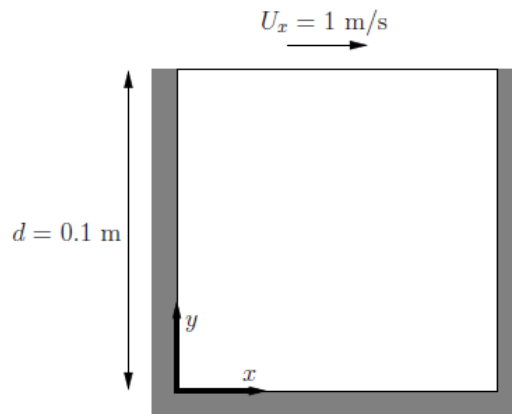
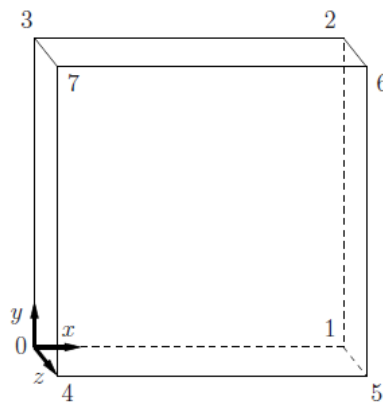
O `blockMesh` é um gerador de malha fornecido com o OpenFOAM, que tem a capacidade de gerar malhas estruturadas a partir das informações sobre a geometria e sobre os contornos do domínio contidas em um arquivo de configuração chamado *blockMeshDict*, localizado no diretório *constant/polyMesh*.

O princípio por trás do `blockMesh` é decompor a geometria do domínio em um conjunto de um ou mais blocos hexaédricos tridimensionais. As arestas dos blocos podem ser linhas retas ou arcos e cada bloco é definido por oito vértices (um para cada canto do hexaedro). Os vértices são numerados e escritos em uma lista, formando pontos no espaço tridimensional. É possível gerar blocos com menos de oito vértices fazendo-se o colapso de um ou mais pares de vértices em outro.

Para ilustrar o uso do `blockMesh`, será analisado um tutorial do OpenFOAM chamado *cavity*, disponível no diretório *\$FOAM_RUN/tutorials/incompressible/icoFoam/cavity*.

A [Figura 18](#) representa a geometria do problema. Todos os contornos são paredes, sendo que a parede superior move-se na direção x com uma velocidade de 1 m/s, enquanto as outras três são estacionárias. A [Figura 19](#) representa a estrutura do bloco com a numeração dos vértices para a malha.

A [Figura 20](#) mostra o arquivo *blockMeshDict* para o caso *cavity*. A palavra-chave (*keyword*) *convertToMeters* especifica um fator de escala pelo qual todas as coordenadas dos vértices na descrição da malha são multiplicados, o significa que, para esse exemplo, as coordenadas estão em decímetros.

Figura 18 – Geometria do caso *cavity*Fonte: [OpenFOAM User Guide \(2012\)](#)Figura 19 – Numeração do bloco para geração da malha do caso *cavity*Fonte: [OpenFOAM User Guide \(2012\)](#)

A palavra-chave *vertices* lista todos os vértices de todos os blocos por suas coordenadas cartesianas. A nomenclatura dos vértices é feita conforme a ordem em que são declarados, começando a partir do “0” conforme mostrado na [Figura 19](#).

A palavra-chave *blocks* contém a definição dos blocos, que é feita por uma série de entradas. A primeira é o identificador do formato do bloco, que é sempre *hex* por ser sempre um hexaedro. Em seguida vem a declaração dos vértices que compõem o bloco e um vetor que indica o número de células em cada direção do plano cartesiano. Esse exemplo contém apenas um bloco com 20 células na direção *x*, 20 na direção *y* e uma na direção *z*. No OpenFOAM, problemas 2D são tratados como 3D, onde a dimensão que se deseja ignorar contém apenas uma célula de comprimento. A última entrada fornece a razão de expansão das células para cada direção do bloco, o que permite que a malha seja refinada em direções específicas. Essa razão representa o comprimento da última célula de uma aresta do bloco dividido pelo comprimento da primeira célula dessa aresta.

```

/*-----* C++ *-----*/
FoamFile
{
  version      2.0;
  format       ascii;
  class        dictionary;
  object       blockMeshDict;
}
// *****

convertToMeters 0.1;

vertices
(
  (0 0 0)
  (1 0 0)
  (1 1 0)
  (0 1 0)
  (0 0 0.1)
  (1 0 0.1)
  (1 1 0.1)
  (0 1 0.1)
);

blocks
(
  hex (0 1 2 3 4 5 6 7) (20 20 1) simpleGrading (1 1 1)
);

edges
(
);

boundary
(
  movingWall
  {
    type wall;
    faces
    (
      (3 7 6 2)
    );
  }
  fixedWalls
  {
    type wall;
    faces
    (
      (0 4 7 3)
      (2 6 5 1)
      (1 5 4 0)
    );
  }
  frontAndBack
  {
    type empty;
    faces
    (
      (0 3 2 1)
      (4 5 6 7)
    );
  }
);

mergePatchPairs
(
);
// *****

```

Figura 20 – *blockMeshDict* para o caso *cavity*Fonte: Tutorial *cavity* do OpenFOAM

A palavra-chave *edges* define as arestas dos blocos, que são consideradas retas por padrão. Contudo, é possível criar arestas curvas, por exemplo, utilizando-se a entrada *arc* para descrever um arco circular. A lista *edges* é requerida apenas se a geometria tiver arestas curvas, se não pode ser deixada sem entradas, como no exemplo.

A palavra-chave *boundary* lista os contornos da malha, que são declarados com os nomes *movingWall*, *fixedWalls* e *frontAndBack* no exemplo. Para cada contorno, é especificado seu tipo como *patch*, *wall*, ou *empty*, e também as faces que o compõe. Cada face é definida por uma lista de 4 vértices. Deve-se tomar bastante cuidado ao declarar esses vértices. A ordem com que eles devem ser declarados deve ser tal que o vetor área aponte para fora da superfície, o que pode ser verificado aplicando-se a “regra da mão direita”. Em outras palavras, olhando a superfície da face pelo lado de fora, a ordem dos vértices deve seguir um sentido anti-horário.

Com todos os parâmetros do dicionário *blockMeshDict* definidos, para se criar a malha basta executar o comando `blockMesh` em um terminal a partir do diretório raiz do caso. O `blockMesh` lê esse dicionário, gera a malha e escreve seus dados em uma série de arquivos como *points*, *faces*, *cells*, *boundaries*, *neighbour*, *owner*, localizados no diretório *constant/polyMesh* do caso.

3.3.1.2 snappyHexMesh

O `snappyHexMesh` difere do jeito tradicional de gerar a malha em CFD. Este método utiliza um procedimento automático para criar uma malha ortogonal hexaédrica ao redor ou dentro de determinada superfície geométrica. A superfície é dada em formato STL e pode ser gerada por um sistema CAD ou por uma outra ferramenta de pré-processamento. Este método permite rápida e robusta geração de malha de geometrias complexas.

Existem vários parâmetros para o ajuste da criação da malha. A densidade inicial da malha é definida através da criação de um bloco que cobre todo o domínio, sendo essa a malha de refinamento nível 0. O nível de refinamento para a área da superfície pode ser tanto para todo o modelo quanto para diferentes componentes da superfície separadamente. A adaptação da espessura da camada de superfície e a máxima não-ortogonalidade permitida das células da superfície podem ser definidos, entre muitos outros parâmetros. O processo de geração da malha é iterativo e, portanto, os parâmetros computacionais também precisam ser definidos.

A definição dos parâmetros e o processo de geração da malha são descritos em detalhes no [OpenFOAM User Guide \(2012\)](#), capítulo 5.4.

[Kortelainen \(2009\)](#) fez um interessante trabalho onde compara o `snappyHexMesh` do OpenFOAM com outros dois *softwares* de geração de malha gratuitos, o SALOME e o Gmsh.

3.3.1.3 Importação de malhas

Muitas vezes nos deparamos com casos que apresentam malhas complexas e não estruturadas. O OpenFOAM permite importar malhas (estruturadas ou não) geradas em

outros *softwares*, comerciais ou gratuitos, convertendo-as para o formato utilizado por ele. Para tal, deve-se executar um comando específico no terminal no diretório raiz do caso. Alguns dos utilitários conversores de malha são:

- `fluentMeshToFoam`: converte um arquivo `.msh` do Fluent;
- `star4ToFoam`: converte arquivos de malha do STAR-CD/PROSTAR;
- `gambitToFoam`: converte arquivos de malha do GAMBIT;
- `ideasUnvToFoam`: converte arquivos de malha no formato *I-Deas unv*;
- `gmshToFoam`: converte um arquivo de malha `.msh` escrito pelo Gmsh.

Uma tabela com todos os utilitários conversores de malha do OpenFOAM está disponível no capítulo 3.6 do [OpenFOAM User Guide \(2012\)](#), e mais informações em seu capítulo 5.5.

3.3.1.4 `checkMesh`

Um utilitário de grande importância é o `checkMesh`. Ele faz um diagnóstico da sua malha, relatando diversos aspectos das propriedades topológicas e geométricas, como ortogonalidade, assimetria, orientação ou número de regiões.

É altamente aconselhável que o usuário execute o `checkMesh` sempre que importar malhas de outros *softwares* para a detecção de possíveis erros oriundos da geração da malha ou de seu processo de conversão.

Ele é executado no terminal a partir do comando `checkMesh` dentro do diretório raiz do caso.

3.3.2 Definindo o caso

Além da geração da malha, faz parte também do pré-processamento a definição do caso estudado. Os parâmetros que definem as condições iniciais e de contorno, as propriedades físicas, os procedimentos de discretização das equações governantes e as características do processo de solução dessas equações devem ser editados em arquivos específicos, que serão detalhados a seguir.

3.3.2.1 Condições iniciais e de contorno

As condições iniciais e de contorno são inseridas em arquivos com o nome da variável, por exemplo, p para a pressão e U para a velocidade, localizados no diretório “0” do caso estudado. A [Figura 21](#) mostra o arquivo U para o tutorial *cavity* do OpenFOAM:

```

/*-----*- C++ -*/
|=====|
| \ \ \ \ | F i e l d | OpenFOAM: The Open Source CFD Toolbox
| \ \ \ \ | O p e r a t i o n | Version: 2.1.1
| \ \ \ \ | A n d | Web: www.OpenFOAM.org
| \ \ \ \ | M a n i p u l a t i o n |
|=====|
FoamFile
{
  version      2.0;
  format       ascii;
  class        volVectorField;
  object       U;
}
// *****

dimensions      [0 1 -1 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
  movingWall
  {
    type        fixedValue;
    value       uniform (1 0 0);
  }

  fixedWalls
  {
    type        fixedValue;
    value       uniform (0 0 0);
  }

  frontAndBack
  {
    type        empty;
  }
}
// *****

```

Figura 21 – Arquivo *U* do tutorial *cavity*Fonte: Tutorial *cavity* do OpenFOAM

O sub-dicionário *dimensions* especifica as dimensões do campo da variável. Como garantia contra a implementação de uma operação sem sentido, o OpenFOAM atribui dimensões para o campo e para as propriedades físicas, e executa uma verificação da dimensão em qualquer operação tensorial. A definição da dimensão é feita por sete escalares limitados por colchetes, que indicam a potência de cada uma das unidades básicas de medição, ordenadas conforme a [Tabela 2](#). Ela fornece as unidades básicas no Sistema Internacional (SI) e no *United States Customary System* (USCS). Pela [Figura 21](#) pode-se ver que o arquivo *U* trata de um campo de velocidade com dimensão m/s.

O sub-dicionário *internalFields* declara a condição inicial do campo interno da geometria, que pode ser *uniform*, descrito por um único valor, ou *nonuniform*, onde o valor de todos os pontos do campo devem ser especificados. Para o exemplo, o fluido está inicialmente em repouso, e portanto com velocidade nula nas três direções.

O *boundaryField* é um sub-dicionário que contém um conjunto de entradas com os nomes de cada uma das superfícies de contorno listadas no arquivo *boundary*, localizado no diretório *constant/polyMesh*. Para cada contorno deve ser declarado o tipo da condição de contorno e o seu valor. No *cavity*, existem três superfícies de contorno: *movingWall*

Tabela 2 – Unidades básicas para definir as dimensões no OpenFOAM

Nº	Propriedade	Unidade SI	Unidade USCS
1	Massa	quilograma (kg)	libra-massa (lbm)
2	Comprimento	metro (m)	pé (ft)
3	Tempo	segundos (s)	
4	Temperatura	Kelvin (K)	Rankine (°R)
5	Quantidade	Quilograma-mol (kgmol)	libra-mol (lbmol)
6	Corrente	ampere (A)	
7	Intensidade luminosa	candela (cd)	

corresponde à superfície superior, que se movimenta com velocidade constante de 1 m/s na direção x , como visto na [Figura 18](#); *fixedWalls* corresponde às paredes fixas (laterais e inferior), e considerando-se a condição de não escorregamento na parede, a velocidade é nula; *frontAndBack* corresponde às faces normais ao plano xy . Apesar de o caso ser em duas dimensões, o fato de o OpenFOAM sempre trabalhar em três dimensões faz com que seja necessário declarar esse contorno usando o tipo especial *empty*, para que a direção z seja desconsiderada nos cálculos.

3.3.2.2 Propriedades físicas

As propriedades físicas para um caso são armazenadas em dicionários cujos nomes apresentam o sufixo *...Properties*, localizados no diretório *constant*. Para o exemplo *cavity*, a única propriedade que precisa ser determinada é a viscosidade cinemática, que é armazenada no dicionário *transportProperties*, mostrado na [Figura 22](#). As entradas necessárias são o nome da propriedade, sua dimensão e seu valor. Pode-se concluir que, para o fluido utilizado no *cavity*, sua viscosidade cinemática (representada no OpenFOAM pela palavra *nu*) é de 0,01 m²/s.

Dependendo das características do problema, outros dicionários de propriedades serão necessários. Por exemplo, para um escoamento turbulento, deve-se especificar o tipo de modelagem de turbulência, *Reynolds-averaged stress* (RAS) ou *Large-eddy simulation* (LES), em um dicionário chamado *turbulenceProperties*. Dentro desses dois tipos de modelagem, pode-se escolher o modelo de turbulência, como $\kappa-\epsilon$ ou $\kappa-\omega$ *SST* para o RAS, em dicionários chamados *RASProperties* e *LESProperties*.

Propriedades como densidade e módulo de elasticidade são especificadas no dicionário *mechanicalProperties*, assim como propriedades como condutividade térmica e coeficiente de expansão térmica são especificados no dicionário *thermalProperties*.

```

/*-----*- C++ -*/
|=====|
| \ \ \ \ | F i e l d | OpenFOAM: The Open Source CFD Toolbox
| \ \ \ \ | O p e r a t i o n | Version: 2.1.1
| \ \ \ \ | A n d | Web: www.OpenFOAM.org
| \ \ \ \ | M a n i p u l a t i o n |
|=====|
FoamFile
{
  version      2.0;
  format       ascii;
  class        dictionary;
  location     "constant";
  object       transportProperties;
}
// *****

nu              nu [ 0 2 -1 0 0 0 ] 0.01;

// *****

```

Figura 22 – Arquivo *transportProperties* para o tutorial *cavity*

Fonte: Tutorial *cavity* do OpenFOAM

3.3.2.3 Controle do tempo e da escrita dos dados (*controlDict*)

Os dados de entrada para o controle de tempo (como tempo inicial, tempo final e intervalo de tempo) e escrita dos dados calculados são definidos no arquivo *controlDict*, localizado no diretório *system* para um determinado caso. Assim, o arquivo *controlDict* especifica os parâmetros de controle para a simulação.

Na [Figura 23](#) pode-se ver esses parâmetros para o tutorial *cavity*. Em *application* deve-se especificar o *solver* que resolverá o problema, que para esse caso é o *icoFoam*.

O sub-dicionário *startFrom* controla o instante de tempo em que se inicia a simulação. Há três opções possíveis: *firstTime*, que inicializa o caso a partir do diretório que representa o menor tempo; *startTime*, onde o usuário define qual é o valor do ponto inicial; e *latestTime*, que inicia a simulação a partir do último diretório de tempo criado, sendo útil para continuar simulações que foram interrompidas. O tempo aqui e em todos os demais locais é dado em segundos.

No sub-dicionário *stopAt* controla-se o instante de tempo em que se encerra a simulação. A opção *endTime* define o instante de tempo exato para o término da simulação. Já *writeNow* e *noWriteNow* são imposições de interrupção após concluir-se o atual intervalo de tempo, porém no primeiro caso escrevendo o resultado final e no segundo sem escrever o resultado final. Por fim, a opção *nextWrite* é usada quando o usuário quer esperar mais alguns passos de tempo antes de interromper a simulação, mais precisamente, esperar até a próxima escrita programada de dados para interromper.

O sub-dicionário *deltaT* controla o intervalo de tempo da simulação.

Outro sub-dicionário importante é o *writeControl*, que controla a escrita de dados. Por exemplo, com a opção *timeStep*, os dados serão escritos de acordo com o número de

```

/*-----* C++ -*-----*/
|=====|
| \ \ \ \ | F i e l d | | OpenFOAM: The Open Source CFD Toolbox
| \ \ \ \ | O p e r a t i o n | | Version: 2.1.1
| \ \ \ \ | A n d | | Web: www.OpenFOAM.org
| \ \ \ \ | M a n i p u l a t i o n | |
|-----|
/*-----*/

FoamFile
{
    version      2.0;
    format       ascii;
    class        dictionary;
    location     "system";
    object       controlDict;
}
// ***** //

application      icoFoam;

startFrom        startTime;

startTime        0;

stopAt           endTime;

endTime          0.5;

deltaT           0.005;

writeControl     timeStep;

writeInterval    20;

purgeWrite       0;

writeFormat      ascii;

writePrecision   6;

writeCompression off;

timeFormat       general;

timePrecision    6;

runTimeModifiable true;

// ***** //

```

Figura 23 – Arquivo *controlDict* para o tutorial *cavity*

Fonte: Tutorial *cavity* do OpenFOAM

intervalos de tempo definido no sub-dicionário *writeInterval*. Com a opção *clockTime*, os dados serão escritos de acordo com o número de segundos em tempo real também definido no sub-dicionário *writeInterval*. Para a escrita dos dados, o OpenFOAM cria diretórios de tempo, similarmente ao diretório “0”, com as informações das variáveis para aquele determinado instante.

Mais informações sobre as entradas do dicionário *controlDict* podem ser encontradas no capítulo 4.3 do [OpenFOAM User Guide \(2012\)](#).

Para o caso *cavity*, a simulação irá começar no tempo zero e acabar em 0,5 segundos,

com um intervalo de tempo de 0,005 segundos. Os dados serão escritos a cada 20 intervalos de tempo.

3.3.2.4 Esquemas numéricos (*fvSchemes*)

O arquivo *fvSchemes* contém informações sobre os esquemas numéricos que serão utilizados para a discretização dos diferentes termos das equações governantes do problema. Os termos que tipicamente necessitam de serem atribuídos um esquema numérico incluem derivações, por exemplo, o gradiente ∇ , e interpolações de valores de um ponto para outro. Nesse quesito, o OpenFOAM oferece uma ampla possibilidade de escolha para o usuário.

O conjunto de termos para o qual esquemas numéricos devem ser especificados são subdivididos dentro do dicionário *fvSchemes* nas categorias listadas na [Tabela 3](#).

Tabela 3 – Principais sub-dicionários utilizados no *fvSchemes*

Sub-dicionário	Categoria de termos matemáticos
<i>interpolationSchemes</i>	Interpolação ponto-a-ponto dos valores
<i>snGradSchemes</i>	Componente do gradiente normal à face da célula
<i>gradSchemes</i>	Gradiente ∇
<i>divSchemes</i>	Divergente $\nabla \cdot$
<i>laplacianSchemes</i>	Laplaciano ∇^2
<i>timeScheme</i>	Derivadas primeira e segunda do tempo, $\partial/\partial t$, $\partial^2/\partial t^2$
<i>fluxRequired</i>	Campos que requerem a geração de um fluxo

Um exemplo do arquivo *fvSchemes* para o tutorial *cavity* pode ser visto na [Figura 24](#).

No sub-dicionário *ddtSchemes* escolhe-se o método de discretização para a primeira derivada no tempo, se for um problema transiente. Para um caso em regime permanente, basta selecionar a opção *steadyState*. Algumas outras opções disponíveis são os métodos de *Euler* e *CrankNicholson*. Para a derivada segunda no tempo ($\partial^2/\partial t^2$), especificado no sub-dicionário *d2dt2Schemes*, apenas o método *Euler* é disponível.

No sub-dicionário *interpolationSchemes* define-se o método de interpolação de uma certa variável, tipicamente do centro da célula para a face. Existem esquemas de interpolação que se encaixam em uma categoria mais geral, como o esquema *linear* (diferenças centrais). Existem outros que são utilizados principalmente em conjunto com o esquema *Gaussian* para a discretização de termos convectivos (divergentes), como os esquemas baseados nos métodos *upwind*, *Total Variation Diminishing* (TVD) e *Normalised Variable* (NV). Entretanto, é altamente improvável que o usuário iria adotar qualquer um dos esquemas específicos de convecção para interpolações gerais dos campos no sub-dicionário *interpolationSchemes*.

de ψ varia entre 0 e 1, onde $\psi = 0$ corresponde à *uncorrected* e $\psi = 1$ corresponde à *corrected*.

Os esquemas para a discretização do gradiente são definidos no sub-dicionário *gradSchemes*. Basicamente há três métodos disponíveis: *Gauss*, *leastSquares* e *fourth*. Os dois primeiros são de segunda ordem e o terceiro é de quarta ordem. Para as opções *leastSquares* e *fourth*, o esquema de discretização é suficiente para especificar o esquema completamente, por exemplo, “*grad(p) leastSquares*”. Porém, o método de *Gauss* requer que o usuário defina ainda o tipo de interpolação, isto é, *Gauss <interpolationScheme>*, dos quais *linear* é a opção mais comum, por exemplo, “*grad(p) Gauss linear*”. Versões limitadas de qualquer um dos três esquemas básicos podem ser selecionados precedendo o esquema de discretização por *cellLimited* (ou *faceLimited*), por exemplo, “*grad(p) cellLimited Gauss linear 1*”.

O sub-dicionário *laplacianSchemes* contém o esquema de discretização para os termos de laplaciano. O esquema de *Gauss* é a única opção de discretização disponível e exige a seleção de um esquema de interpolação para o coeficiente de difusão, e um esquema para o gradiente normal à superfície, ou seja, *Gauss <interpolationScheme> <snGradScheme>*. O mais comum é a utilização do *Gauss linear* com *corrected* ou *limited*. Por exemplo, para o caso *cavity* foi usado: “*laplacian(nu, U) Gauss linear corrected*”.

No sub-dicionário *divSchemes* são especificados os esquemas de discretização para os termos de divergente. Novamente, o esquema de *Gauss* é a única opção de discretização disponível, sendo também necessário selecionar um esquema de interpolação. Portanto, as entradas são do tipo *Gauss <interpolationScheme>*. Por exemplo, para o divergente no caso *cavity* foi utilizado “*div(phi, U) Gauss upwind*”, onde $\phi = \rho U$. Esse sub-dicionário representa a discretização do termo convectivo, e requer uma atenção especial por parte do usuário. Para um caso específico, existem métodos que são mais adequados que outros, podendo causar a não convergência da solução.

O sub-dicionário *fluxRequired* lista as variáveis que são acopladas com o cálculo do fluxo, ou seja, dependem do fluxo para serem determinadas. Na dinâmica dos fluidos, a principal variável nessa lista é a pressão (quando há o acoplamento pressão-velocidade). Há casos, como em análise de sólidos por exemplo, onde pode ser necessário calcular o fluxo de calor.

O arquivo *fvSchemes* é um dos mais complicados arquivos de configuração do OpenFOAM. Ele apresenta uma quantidade enorme de métodos de interpolação que podem ser usados nos termos de divergente, gradiente e laplaciano, sendo que uma escolha errada pode levar à não convergência da solução. Essa variedade e liberdade de aplicação é um grande diferencial do OpenFOAM para os *softwares* comerciais. No entanto, isso requer do usuário um conhecimento mais profundo dos métodos numéricos para sua correta aplicação. Mais informações podem ser encontradas no capítulo 4.4 do [OpenFOAM User](#)

Guide (2012).

3.3.2.5 Controle da solução (*fvSolution*)

Os métodos de solução das equações discretizadas, suas tolerâncias e alguns parâmetros para o algoritmo de solução (correção pressão-velocidade e de ortogonalidade da malha), são controlados a partir do dicionário *fvSolution* no diretório *system*.

O *fvSolution* contém um conjunto de sub-dicionários que são específicos para o *solver* sendo executado. No entanto, existe um pequeno conjunto de sub-dicionários que cobrem a maior parte dos utilizados pelos *solvers* padrões. Estes incluem *solvers*, *relaxationFactors*, *PISO* e *SIMPLE*.

A Figura 25 demonstra um exemplo para o arquivo *fvSolution*.

```

/*-----*-----*-----*-----*-----*-----*-----*-----*-----*
|
|   \ \ \ \ \   F i e l d   |   OpenFOAM: The Open Source CFD Toolbox
|   / / / / /   O p e r a t i o n   |   Version: 2.1.1
|   / / / / /   A n d   |   Web: www.OpenFOAM.org
|   / / / / /   M a n i p u l a t i o n   |
|
|-----*-----*-----*-----*-----*-----*-----*-----*-----*
FoamFile
{
  version      2.0;
  format       ascii;
  class        dictionary;
  location     "system";
  object       fvSolution;
}
// *****

solvers
{
  p
  {
    solver          PCG;
    preconditioner  DIC;
    tolerance       1e-06;
    relTol          0;
  }

  U
  {
    solver          PBiCG;
    preconditioner  DILU;
    tolerance       1e-05;
    relTol          0;
  }
}

PISO
{
  nCorrectors      2;
  nNonOrthogonalCorrectors 0;
  pRefCell         0;
  pRefValue        0;
}

// *****

```

Figura 25 – Exemplo de um arquivo *fvSolution*

Fonte: Tutorial *pitzDaily* do OpenFOAM

O primeiro sub-dicionário, *solvers*, refere-se ao método de cálculo numérico para resolver o sistema de equações lineares provenientes da discretização das equações governantes. É importante destacar que aqui *solvers* não se refere ao aplicativo *solver*, que descreve o conjunto de equações e algoritmos para solucionar um problema particular, e sim ao método de solução da matriz do sistema de equações lineares. Entre os algoritmos implementados no OpenFOAM, pode-se citar os métodos de Gauss-Seidel, Multigrid algébrico e variantes do gradiente conjugado. Mais informações, como os métodos de solução e pré-condicionadores, podem ser encontrados no capítulo 4.5.1 do [OpenFOAM User Guide \(2012\)](#).

Os métodos de solução de matrizes esparsas implementados no OpenFOAM são iterativos e, portanto, baseiam-se em reduzir o resíduo das equações até um valor pré-estabelecido. O método de solução é interrompido quando o resíduo se torna menor que a tolerância especificada em *tolerance*, ou a razão entre os resíduos da iteração atual e da inicial for menor que a tolerância relativa especificada em *relTol*. Um método opcional de se interromper a solução é especificando-se o número máximo de iterações usando a opção *maxIter*. A tolerância relativa do *solver* limita a melhoria relativa da solução inicial para a final. Em simulações transientes, é comum definir a tolerância relativa do *solver* como zero para forçar a solução a convergir para a tolerância do *solver* em cada intervalo de tempo.

Um segundo sub-dicionário do *fvSolution* que é frequentemente usado no OpenFOAM é o *relaxationFactors*. Ele controla a sub-relaxação, uma técnica usada para melhorar a estabilidade de uma simulação, principalmente na resolução de problemas de estado estacionário. A sub-relaxação funciona limitando-se a quantidade em que uma variável muda de uma iteração para a outra. O fator de sub-relaxação α pode assumir valores entre zero e um, sendo que $\alpha = 1$ não apresenta sub-relaxação e $\alpha = 0$ representa uma solução que não aplica correção e não muda em nada com sucessivas iterações. Uma escolha ideal para o α é aquele que é pequeno o suficiente para garantir uma simulação estável, mas grande o suficiente para mover o processo iterativo para a frente rapidamente.

Os algoritmos PISO (*Pressure-Implicit Split-Operator*) e SIMPLE (*Semi-Implicit Method for Pressure-Linked Equations*) estão implementados no OpenFOAM para resolver o acoplamento pressão-velocidade presente nas equações de escoamento de fluidos, sendo o PISO utilizado para problemas transientes e o SIMPLE para problemas estacionários. Os dois algoritmos são baseados em procedimentos iterativos, avaliando a solução em dado instante de tempo e, então, corrigindo-a. SIMPLE só faz uma correção enquanto PISO requer mais do que uma, mas normalmente não mais do que quatro. O usuário deve então especificar o número de correções no sub-dicionário PISO pela opção *nCorrectors*. Uma correção adicional para levar em conta a não-ortogonalidade da malha está disponível em ambos SIMPLE e PISO, sendo que o número de correções não-ortogonais é especificada na opção *nNonOrthogonalCorrectors*, variando de zero para malhas ortogonais até valores

mais altos, como dez para malhas muito não-ortogonais.

Um outro parâmetro importante é o *residualControl*, onde se define o valor para o resíduo inicial no qual considera-se que a simulação atingiu a convergência e portanto será interrompida. Esse parâmetro é muito útil para se definir uma condição para o término de simulações em regime permanente.

3.4 Solvers

As simulações no OpenFOAM são realizadas por arquivos executáveis chamados *solvers*, que lêem as informações referentes ao caso (malha e condições de simulação) e resolvem problemas específicos da mecânica do contínuo. Os *solvers* são resultado da compilação dos arquivos fonte, e a solução numérica depende de como as bibliotecas do OpenFOAM são usadas para montar o algoritmo de solução. Na essência, os *solvers* leem as informações da simulação (fornecidas nas etapas descritas anteriormente), resolvem as equações através de metodologias de solução específicas para cada caso, e geram arquivos de resultados para pós-processamento (SILVA, 2007).

A Tabela 4 mostra alguns dos *solvers* fornecidos com o OpenFOAM, que abrangem áreas como escoamento compressível e incompressível, escoamento multifásico, combustão, transferência de calor, escoamento com trajetória de partículas, eletromagnetismo, análise de tensão em sólidos, entre outros. Uma lista completa dos *solvers* pode ser encontrada no capítulo 3.5 do [OpenFOAM User Guide \(2012\)](#).

Para realizar a simulação deve-se executar o *solver* em um terminal no diretório raiz do caso. Por exemplo, para uma simulação de combustão em fase gasosa, que utilize o *solver reactingFoam*, basta digitar seu nome no terminal:

```
reactingFoam
```

Na tela do terminal irão aparecer informações sobre a evolução da simulação, como os resíduos das equações, número de *Courant*, e tempo computacional.

Qualquer aplicativo pode ser executado como um processo em *background*, ou seja, que não necessita de estar completo para que o usuário possa executar outros comandos no terminal. Se o usuário quiser executar o *solver reactingFoam* em *background* e escrever o progresso da simulação que estaria na tela do terminal em um arquivo chamado **log**, deve executar o comando:

```
reactingFoam > log &
```

Com a simulação rodando em um processo em *background* nada será escrito na tela do terminal. Se o usuário quiser acompanhar também o progresso da simulação no terminal pode executar o seguinte comando:

Tabela 4 – Alguns *solvers* fornecidos com o OpenFOAM

Solver	Descrição
laplacianFoam	soluciona uma equação de Laplace simples, por exemplo, para a difusão térmica em sólidos
potencialFoam	escoamento potencial que pode ser usado para gerar campos iniciais para códigos que usam Navier-Stokes
icoFoam	escoamento transiente e laminar de fluidos Newtonianos incompressíveis
simpleFoam	escoamento permante, laminar ou turbulento, para fluidos Newtonianos incompressíveis
pisoFoam	escoamento transiente de fluidos incompressíveis
sonicFoam	solver transiente para escoamento supersônico, laminar ou turbulento, de um gás compressível
cavitatingFoam	solver para cavitação transiente
engineFoam	solver para motores de combustão interna
reactingFoam	solver para combustão com reações químicas
magneticFoam	solver para o campo magnético gerado por magnetos permanentes
solidDisplacementFoam	solver transiente para deformação linear-elástica de um corpo sólido

```
tail -f log
```

Isso fará com que seja escrito no terminal o final do arquivo **log**, sendo constantemente atualizado, como se fosse um processo executado normalmente. A vantagem em se fazer isso é ter esse progresso escrito em arquivo de texto, no caso de uma posterior avaliação ser necessária. Um cuidado deve ser tomado para simulações longas, onde esse arquivo **log** pode se tornar muito grande, e abri-lo para uma simples visualização pode se mostrar extremamente custoso.

Uma dica que se mostrou muito útil para lidar mais facilmente com o terminal para a execução dos aplicativos (*solvers* e utilitários) do OpenFOAM é a instalação do pacote *Nautilus Open Terminal*. Ele permite abrir o terminal diretamente a partir de uma pasta qualquer, clicando com o botão direito e em seguida clicando em “Abrir em um terminal”. Com isso, o usuário pode navegar pelos seus casos, configurar seus arquivos e em seguida vai no diretório raiz do caso, clica com o botão direito e abre o terminal, que já virá naquele diretório. Isso evita que o usuário abra o terminal sempre no diretório raiz do computador e tenha que ir navegando pelas pastas até o diretório do caso estudado, o que se mostra

um processo maçante. No Ubuntu, esse pacote pode ser instalado com o comando:

```
sudo apt-get install nautilus-open-terminal
```

3.4.1 Paralelização

Um fator de grande importância no OpenFOAM é a sua enorme capacidade para simulações em paralelo. Devido ao fato dele ser um *software* gratuito, não há limitações quanto ao número de licenças para serem distribuídas entre os processadores ou núcleos.

O método de processamento em paralelo utilizado pelo OpenFOAM é conhecido como decomposição do domínio, em que a malha e os campos associados são divididos em subdomínios e atribuídos aos diferentes processadores para solução. A comunicação entre os processadores é feita utilizando-se o protocolo de comunicação MPI (*Message Passing Interface*), sendo que, por padrão, o OpenFOAM é fornecido com a biblioteca de domínio público *openMPI*, mas qualquer outra pode ser utilizada.

A malha e os campos são decompostos usando-se o utilitário `decomposePar`. Eles são divididos de acordo com um conjunto de parâmetros, entre eles o número de processadores e o método de decomposição, definidos em um dicionário chamado *decomposeParDict*, que deve estar localizado no diretório *system* do caso de interesse.

O OpenFOAM fornece quatro métodos para a decomposição do domínio: *simple*, *hierarchical*, *scotch* e *manual*. Informações detalhadas sobre esses métodos podem ser encontradas no capítulo 3.4 do [OpenFOAM User Guide \(2012\)](#). Um exemplo de um arquivo *decomposeParDict* para o caso tutorial *damBreak* é mostrado na [Figura 26](#). Os métodos *simple*, *hierarchical* e *manual* requerem a entrada de coeficientes para a sua utilização. O método *scotch* não requer nenhuma entrada adicional, e é um método automático de decomposição, apresentando uma maior simplicidade para utilização.

O primeiro passo é decompor o caso estudado executando o utilitário `decomposePar` em seu diretório raiz. Isso fará com que um conjunto de subdiretórios seja criado, sendo um para cada processador. Os diretórios são nomeados *processorN*, onde $N=0,1,2,\dots$ representa o número de um processador. Neles estarão contidos os diretórios de tempo que contêm as descrições dos campos decompostos, e um diretório *constant/polyMesh* contendo a descrição da malha decomposta.

Em seguida, para rodar um caso em paralelo utiliza-se um comando com a seguinte sintaxe:

```
mpirun -hostfile <machines> -np <nProcs> <foamExec> <otherArgs>  
-parallel > log &
```

Onde *<nProcs>* é o número de processadores, *<foamExec>* é o executável como, por exemplo, um *solver*, e *<machines>* é um arquivo que contém o nome das máquinas

```

/*-----* C++ *-----*/
|=====|
| \ \ / | F i e l d | OpenFOAM: The Open Source CFD Toolbox
| \ \ / | O p e r a t i o n | Version: 2.1.1
| \ \ / | A n d | Web: www.OpenFOAM.org
| \ \ / | M a n i p u l a t i o n |
|-----*/
FoamFile
{
    version      2.0;
    format       ascii;
    class        dictionary;
    location     "system";
    object       decomposeParDict;
}
// *****

numberOfSubdomains 4;

method          simple;

simpleCoeffs
{
    n             ( 2 2 1 );
    delta        0.001;
}

hierarchicalCoeffs
{
    n             ( 1 1 1 );
    delta        0.001;
    order        xyz;
}

manualCoeffs
{
    dataFile     "";
}

distributed     no;

roots           ( );
// *****

```

Figura 26 – Exemplo de um arquivo *decomposeParDict*

Fonte: Tutorial *damBreak* do OpenFOAM

utilizadas caso o problema esteja sendo rodado em múltiplas máquinas em uma rede.

Para um computador com um processador de 8 núcleos, onde se queira utilizar o *solver simpleFoam* por exemplo, o comando será:

```
mpirun -np 8 simpleFoam -parallel > log &
```

Por fim, depois de um caso ter sido executado em paralelo, ele pode ser reconstruído para pós-processamento. O caso é reconstruído por meio da fusão dos conjuntos de diretórios de tempo de cada *processorN* em um único conjunto de diretórios de tempo. O utilitário *reconstructPar* executa tal reconstrução a partir do comando:

```
reconstructPar
```

3.5 Pós-Processamento

A principal ferramenta de pós-processamento fornecida com o OpenFOAM é o utilitário `paraFoam`, que é um *script* que iniciará automaticamente o *software* livre de visualização de dados ParaView. Esse *software* utiliza o *Visualisation ToolKit* (VTK) como processamento de dados e renderização de imagens, podendo portanto, ler qualquer arquivo de dados no formato VTK. O OpenFOAM fornece também o utilitário `foamToVTK`, que converte os dados gerados no formato nativo do OpenFOAM para o formato VTK, podendo então utilizar o ParaView para abrir os arquivos VTK ou qualquer outro *software* que faça a leitura de arquivos VTK.

Alguns usuários podem sentir a necessidade de usar outros *softwares* de visualização de dados para o pós-processamento. É possível converter os resultados fornecidos pelo OpenFOAM para formatos lidos por outros *softwares* como FLUENT, Fieldview, Ensignt, e Tecplot, utilizando ferramentas fornecidas junto com o OpenFOAM. Alguns exemplos são os utilitários `foamDataToFluent`, `foamToFieldview` e `foamToEnsignt`. Mais informações de como converter os dados para serem lidos em outros *softwares* no capítulo 6 do [OpenFOAM User Guide \(2012\)](#).

O Paraview opera com uma estrutura baseada em árvore na qual os dados podem ser filtrados a partir do módulo do caso para se criar conjuntos de sub-módulos. Um ponto forte do ParaView é que o usuário pode criar uma série de sub-módulos e mostrar apenas o que ele quer para criar a imagem ou animação desejada.

A [Figura 27](#) mostra a janela de interface do ParaView para o caso *cavity*.

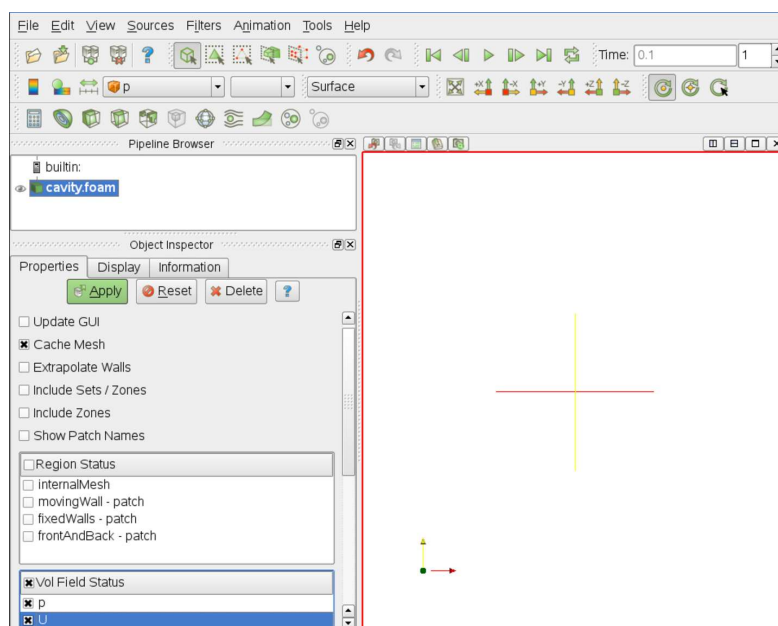


Figura 27 – Janela do ParaView

Fonte: [OpenFOAM User Guide \(2012\)](#)

Em *Pipeline Browser* são listados os módulos abertos no ParaView, onde os módulos selecionados são destacados em azul. A imagem para o módulo selecionado pode ser ativada ou desativada clicando-se no botão em forma de “olho” ao lado do respectivo módulo. O painel *Properties* contém as seleções de entrada para o caso, como regiões da malha e campos de variáveis. O painel *Display* controla a representação visual do módulo escolhido como as cores, por exemplo. O painel *Information* fornece estatísticas do caso, como a geometria e o tamanho da malha.

No ParaView estão incluídas diversas ferramentas para visualização de dados em CFD, como a criação de gráficos de contorno, vetores e linhas de fluxo. Ainda é possível criar animações para analisar a evolução dos resultados. As [Figura 28](#) e [Figura 29](#) mostram a representação da pressão e da velocidade para o tutorial *cavity* do OpenFOAM, para um tempo de 0,5 segundos, criadas no ParaView.

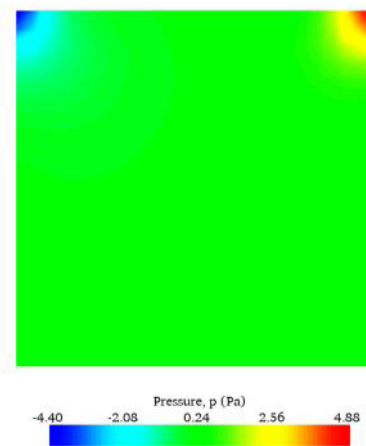


Figura 28 – Campo de pressão para o tutorial *cavity*

Fonte: OpenFOAM User Guide (2012)

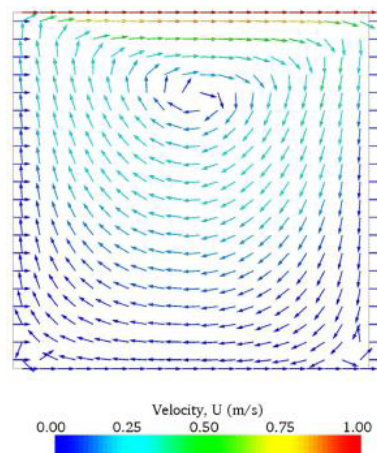


Figura 29 – Campo de velocidades para o tutorial *cavity*

Fonte: OpenFOAM User Guide (2012)

O OpenFOAM fornece também dois utilitários para que o usuário possa fazer uma amostragem dos dados resultantes da simulação. O primeiro é o `probeLocations`, que extrai os valores das variáveis em função do tempo para um determinado ponto do domínio. Para utilizá-lo, é necessário a presença do dicionário `probesDict` no diretório `system` do caso estudado. Nele deve-se definir os campos e os pontos para a extração dos dados, conforme exemplificado na Figura 30. Após a simulação ser concluída, basta utilizar o comando `probeLocations` em um terminal no diretório raiz do caso. Será criado então um diretório chamado `probes`, que irá conter os arquivos resultantes.

```

/*-----* C++ *-----*/
|=====|
| \\ \\ / | F i e l d | OpenFOAM: The Open Source CFD Toolbox
| \\ \\ / | O p e r a t i o n | Version: 2.1.1
| \\ \\ / | A n d | Web: www.OpenFOAM.org
| \\ \\ / | M a n i p u l a t i o n |
|=====|
FoamFile
{
  version      2.0;
  format       ascii;
  class        dictionary;
  object       probesDict;
}
// *****

// Fields to be probed. runTime modifiable!
fields
(
  p
);

// Locations to be probed. runTime modifiable!
probeLocations
(
  (0.0254 0.0253 0.0)
  (0.0508 0.0253 0.0)
  (0.0762 0.0253 0.0)
  (0.1016 0.0253 0.0)
  (0.1270 0.0253 0.0)
  (0.1524 0.0253 0.0)
  (0.1778 0.0253 0.0)
);
// *****

```

Figura 30 – Exemplo de um arquivo `probesDict`

Fonte: OpenFOAM

O segundo é o utilitário `sample`, que faz a amostragem dos dados dos campos tanto por uma linha 1D quanto por um plano 2D. Os locais para realizar a amostragem dos dados deve ser especificado em um dicionário chamado `sampleDict`, localizado no diretório `system` do caso. Esse dicionário pode conter as seguintes entradas, dependendo da necessidade do usuário: `interpolationScheme`, onde se define o esquema para a interpolação dos dados; `sets`, são as localizações no domínio onde os campos são amostrados por uma linha 1D; `surfaces`, são as localizações no domínio onde os campos são amostrados por uma superfície 2D; `setFormat`, onde se define o formato de saída para os dados das linhas; `surfaceFormat`,

define o formato de saída para os dados das superfícies; e *fields*, onde se escolhe os campos a serem amostrados. A Figura 31 mostra um exemplo de um arquivo *sampleDict* para o caso *plateHole*, localizado no diretório $\$FOAM_TUTORIALS/solidDisplacementFoam$. Para esse caso os dados para o campo *sigmaxx* (σ_{xx}) serão amostrados em uma linha definida no sub-dicionário *sets* para cada intervalo de tempo nos quais definiu-se a escrita dos dados. Assim, um diretório chamado *sets* será criado, que irá conter sub-diretórios correspondentes aos diretórios de tempo, onde estarão os arquivos resultantes.

```

/*-----* C++ *-----*/
|=====|
| \ \ \ \ | F i e l d           | OpenFOAM: The Open Source CFD Toolbox
| \ \ \ \ | O p e r a t i o n   | Version: 2.1.1
| \ \ \ \ | A n d               | Web:      www.OpenFOAM.org
| \ \ \ \ | M a n i p u l a t i o n |
|-----|
FoamFile
{
  version      2.0;
  format       ascii;
  class        dictionary;
  location     "system";
  object       sampleDict;
}
// *****

interpolationScheme cellPoint;

setFormat      raw;

sets
(
  leftPatch
  {
    type        uniform;
    axis        y;
    start       ( 0 0.5 0.25 );
    end         ( 0 2 0.25 );
    nPoints     100;
  }
);

fields         ( sigmaxx );

// *****

```

Figura 31 – Exemplo de um arquivo *sampleDict*

Fonte: Tutorial *plateHole* do OpenFOAM

4 biomassGasificationFoam: Características, Modelagem e Implementação

4.1 Introdução

O OpenFOAM é um *software* de simulação gratuito, de código aberto, escrito em linguagem C++, fazendo uso da linguagem orientada ao objeto e da estrutura de classes. Como discutido no [Capítulo 3](#), os módulos escritos em C++ são utilizados para construir *solvers*, utilitários e bibliotecas.

Os *solvers* são programas executáveis contendo algoritmos que integram as funcionalidades de selecionadas bibliotecas e organizam os cálculos em cada passo de tempo. Cada solver é preparado para resolver um tipo de problema, por exemplo, o `reactingFoam`, para a solução de problemas de escoamento de fluidos onde ocorrem reações químicas. Os utilitários também são programas executáveis. Porém, designados para realizarem tarefas de pré e pós-processamento. As bibliotecas são utilizadas pelos *solvers* e utilitários, podendo ser divididas em dois grupos principais: numéricas e físicas. As bibliotecas numéricas implementam as ferramentas para a solução numérica, como métodos de discretização, caracterização do domínio e da malha, solução dos sistemas de equações e paralelização. As bibliotecas físicas implementam os modelos matemáticos para os processos físicos e químicos, como modelos de turbulência, propriedades físicas, cinética química, entre outros.

Uma importante característica que o OpenFOAM herda da estrutura de classes da linguagem C++ é a possibilidade de substituir ou adaptar as bibliotecas existentes para adequá-las ao problema em estudo.

Todas essas características fazem do OpenFOAM um ambiente de programação altamente flexível e transparente para o desenvolvimento de novos *solvers* e adaptação de novas bibliotecas. Foi nesse ambiente que [Kwiatkowski et al. \(2013\)](#) desenvolveram o *solver* `biomassGasificationFoam` e suas bibliotecas complementares `biomassGasificationMedia`, para a simulação dos processos físicos e termoquímicos da gaseificação e pirólise de biomassa.

O código desenvolvido integra modelos de secagem, pirólise, combustão, gaseificação e o complexo escoamento em meio poroso. Segundo [Kwiatkowski et al. \(2013\)](#), as mais importantes novas funcionalidades implementadas são o escoamento transiente em meio poroso (com alteração da porosidade), uma definição flexível da biomassa e suas propriedades, transferência de massa e calor entre as fases sólida e gasosa, reações homogêneas e heterogêneas (o calor de reação é definido diretamente ou baseado nas entalpias de formação), e mecanismos cinéticos customizáveis de pirólise e gaseificação.

Essas características levantam o interesse para a aplicação desse código não apenas para

processos de gaseificação de biomassa, mas também para vários outros processos de conversão termoquímica de sólidos em leito fixo, como processos de pirólise e combustão de sólidos em caldeiras ou células de combustão.

O código do `biomassGasificationFoam` é fornecido contendo quatro componentes:

- `biomassGasificationFoam` - é o *solver* propriamente dito, que apresenta um algoritmo para sequenciar a solução e integrar as demais bibliotecas;
- `biomassGasificationMedia` - conjunto de bibliotecas adicionais;
- `setPorosity` - utilitário para a criação do campo de porosidade (será comentado na seção 4.3). Utilizado no pré-processamento;
- `totalMassBiomassGasificationFoam` - utilitário para a integração da massa da fase sólida por todo o domínio computacional, apresentando dados para a perda de massa total do sólido no tempo. Utilizado no pós-processamento.

Os três primeiros componentes serão comentados de forma mais aprofundada nas próximas seções. O último, `totalMassBiomassGasificationFoam`, define um utilitário de pós-processamento chamado `totalMass`, que fornece informações apenas sobre a mudança no tempo da massa total do meio poroso, e escreve os dados em um arquivo de texto chamado “*total-Mass.txt*”. Para usar essa ferramenta, basta executar o seguinte comando em um terminal no diretório raiz do caso:

```
totalMass
```

A Figura 32 apresenta um esquema da estrutura do código (o *solver* e as bibliotecas) conforme dada por Kwiatkowski et al. (2013).

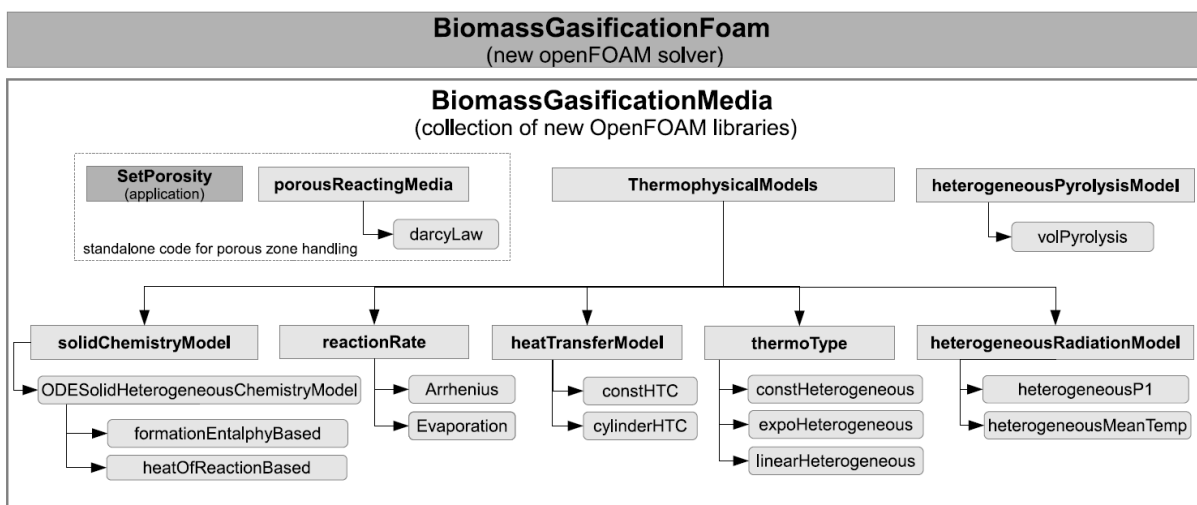


Figura 32 – Estrutura do código `biomassGasificationFoam`: *solver* e bibliotecas

Fonte: Kwiatkowski et al. (2013)

4.2 Instalação

Nesta seção serão tratadas informações sobre a instalação do pacote *biomassGasificationFoam*, visando auxiliar futuros pretendentes à sua utilização. As informações aqui fornecidas e os problemas encontrados são referentes à versão do código disponibilizada na época deste trabalho (2016).

Ele foi desenvolvido para ser utilizado com a versão do OpenFOAM 2.1.1. Também testamos que ele funciona com a versão do OpenFOAM 2.1.x, que é uma versão de repositório. Portanto, para sua utilização, o usuário deve ter instalado uma dessas versões do OpenFOAM. Essas versões e informações sobre sua instalação podem ser encontradas no endereço eletrônico da OpenFOAM Foundation <<http://openfoam.org/>>. A instalação do `biomassGasificationFoam` não se mostrou possível com versões um pouco mais novas, como 2.2.X e 2.3.X, devido à mudanças nas bibliotecas do OpenFOAM, o que poderia se mostrar interessante pelas novas funcionalidades que foram implementadas nessas versões. Por exemplo, a versão 2.3.X alterou as bibliotecas de modelos termofísicos, e também inseriu vários novos solucionadores de sistemas de Equações Diferenciais Ordinárias (EDO), que poderiam se mostrar mais adequados para a solução da cinética química dos casos simulados com o `biomassGasificationFoam`, diminuindo problemas de convergência. Uma sugestão para trabalhos futuros seria justamente a adaptação do `biomassGasificationFoam` para funcionar com versões mais novas do OpenFOAM. Porém, essa tarefa pode se mostrar de difícil execução, requerindo do usuário um conhecimento profundo da linguagem de programação C++ e sua implementação no OpenFOAM.

Primeiramente, o código `biomassGasificationFoam` pode ser obtido por *download* na página eletrônica <<https://biomassgasification.eu/biomassGasificationFoam>>, juntamente com o artigo de Kwiatkowski et al. (2013).

O local de instalação do OpenFOAM, a pasta de uso pessoal do usuário e as variáveis de ambiente que os definem serão importantes no processo de instalação. Não existe um diretório obrigatório no qual deve ser instalado o OpenFOAM, ficando a escolha do usuário. Ele pode ser, por exemplo, os diretórios `/opt`, `/opt/OpenFOAM`, `/usr/local/OpenFOAM` ou `$HOME/OpenFOAM`, sendo esse último uma opção para apenas um usuário. Para uso neste trabalho, o OpenFOAM foi instalado no diretório `/opt`, resultando no local de instalação `/opt/OpenFOAM-2.1.1`. Assim sendo, se o leitor instalou em um local diferente, deve fazer as considerações e mudanças necessárias. Esse diretório de instalação pode ser verificado com a variável de ambiente `$WM_PROJECT_DIR`, executando o seguinte comando:

```
echo $WM_PROJECT_DIR
```

que retorna o diretório `/opt/OpenFOAM-2.1.1`. A pasta de uso pessoal do usuário fica no diretório `$HOME/OpenFOAM/<usuário>-2.1.1` e é atribuída à variável de ambiente `$WM_PROJECT_USER_DIR`. O comando:

```
echo $WM_PROJECT_USER_DIR
```

retorna o diretório `/home/caio/OpenFOAM/caio-2.1.1` para o caso do autor. Todas as variáveis de ambiente definidas podem ser vistas com o comando:

env

Dando sequência ao processo de instalação do código, é aconselhável que o usuário copie e cole o arquivo baixado (*biomassGasificationFoam_installPack_1.0*) na sua pasta de uso pessoal e extraia o arquivo *.zip* nesse mesmo local, obtendo o diretório *biomassGasificationFoam_installPack_1.0*. Nesse diretório estarão contidos os seguintes itens:

- *biomassGasificationFoam_installPack_1.0* - Diretório de mesmo nome contendo os códigos do *solver*, utilitários e bibliotecas, e arquivos necessários para a instalação;
- *oneCellTest* - Caso teste onde o domínio é reduzido a apenas uma célula, com o objetivo de analisar a cinética de determinado processo;
- *TGA_test* - Caso teste de análise termogravimétrica para a pirólise de uma madeira que foi utilizado para a validação do código (comentado na [seção 2.4](#));
- *README* - arquivo de texto README.

Os dois casos testes podem ser executados pelo usuário e servem como exemplos e fontes de informação. Já o primeiro diretório (*biomassGasificationFoam_installPack_1.0*) contém os códigos e arquivos para a instalação, apresentando os seguintes itens:

- *biomassGasificationFoam* - Conjunto de códigos do *solver* `biomassGasificationFoam`, para a sua execução e compilação;
- *biomassGasificationMedia* - Conjunto de códigos da biblioteca auxiliar `biomassGasificationMedia`;
- *setPorosity* - Códigos do utilitário `setPorosity`;
- *totalMassBiomassGasificationFoam* - Códigos do utilitário `totalMassBiomassGasificationFoam`;
- *biomassGasificationMediaDirectories* - Arquivo de texto que define as variáveis de ambiente para o `biomassGasificationFoam`;
- *install* - Arquivo de texto que funciona como *script* para a instalação do `biomassGasificationFoam`;
- *practical_issues.pdf* - Arquivo *.pdf* com instruções básicas de instalação;
- *README* - Arquivo de texto com instruções de instalação e informações básicas dos códigos.

Antes de executar o *script install* para realizar a instalação, é necessário declarar as variáveis de ambiente, tanto do OpenFOAM quanto do `biomassGasificationFoam`. Para as variáveis do OpenFOAM, basta executar o seguinte comando em um terminal (levando em consideração o local de instalação escolhido pelo autor):

```
source /opt/OpenFOAM-2.1.1/etc/bashrc
```

Para as variáveis do *biomassGasificationFoam*, o usuário deve abrir um terminal no diretório *biomassGasificationFoam_installPack_1.0*, comentado acima (que contém os arquivos *install* e *biomassGasificationMediaDirectories*) e executar o comando:

```
source biomassGasificationMediaDirectories
```

Esse comando executa o arquivo de texto *biomassGasificationMediaDirectories* como se fosse um programa. Os usuários podem testar se as variáveis de ambiente necessárias para a instalação estão corretamente definidas com os seguintes comandos:

```
echo $WM_PROJECT_DIR
```

deve retornar o diretório `/opt/OpenFOAM-2.1.1`.

```
echo $FOAM_HGS
```

deve retornar o diretório `/opt/OpenFOAM-2.1.1/biomassGasificationMedia`.

```
echo $LIB_SRC
```

deve retornar o diretório `/opt/OpenFOAM-2.1.1/src`. É extremamente provável que esse comando não retorne nenhum diretório, indicando que a variável `$LIB_SRC` não está definida, fazendo com que o usuário tenha que defini-la manualmente. Esse é um primeiro inconveniente. Um segundo inconveniente é que, se o usuário quiser futuramente alterar os códigos, terá que recompilar o *solver* ou as bibliotecas que foram alteradas, e toda vez que ele for fazer esse procedimento terá que definir novamente as variáveis de ambiente (com o arquivo *biomassGasificationMediaDirectories* e a variável `$LIB_SRC`).

Para contornar esses inconvenientes e polpar tempo, o autor propõe então a seguinte solução: definir essas variáveis de modo automático toda vez que o terminal for aberto, colocando os comandos de definição no arquivo `~/.bashrc`. Esse arquivo fica localizado na pasta *home* do usuário (`/home/<usuário>`, também designado por `$HOME` ou pelo símbolo `~`). Ele é um arquivo oculto, por isso seu nome é precedido por um ponto, e é executado toda vez que um terminal é aberto. Esse procedimento já deve ter sido realizado pelo usuário ao instalar o OpenFOAM, acrescentando uma linha ao final desse arquivo que define as variáveis de ambiente do OpenFOAM.

Primeiramente, crie um arquivo que realizará a exportação da variável `$LIB_SRC`. Um exemplo utilizado pelo autor é um arquivo de texto chamado *exportLibSrc* com o conteúdo mostrado na [Figura 33](#). Coloque esse arquivo no diretório *biomassGasificationFoam_installPack_1.0*, juntamente com o arquivo *biomassGasificationMediaDirectories*, como mostrado na [Figura 34](#). Em seguida, utilize o seguinte comando em um terminal para abrir o arquivo `~/.bashrc` e editá-lo:

```
gedit ~/.bashrc
```

Ao final do arquivo, acrescente as linhas com os comandos que realizarão a execução dos arquivos *biomassGasificationMediaDirectories* e *exportLibSrc*, como mostra a [Figura 35](#), sendo esses comandos específicos para cada computador e diretório de instalação utilizado. Portanto, o usuário deve fazer a adaptação para o seu caso.

omassGasificationFoam_installPack_1.0/biomassGasificationMedia/thermophysicalModels). Se isso não for feito, o usuário terá problemas de permissão como o destacado na [Figura 36](#).

```
caio@caio-notebook:~/OpenFOAM/cao-2.1.1/biomassGasificationFoam_installPack_1.0/biomassGasificationFoam_installPack_1.0$ ./install
./install: 3: ./install: source: not found
mkdir: é impossível criar o diretório "/opt/OpenFOAM-2.1.1": Arquivo existe
mkdir: é impossível criar o diretório "/opt/OpenFOAM-2.1.1/applications": Arquivo existe
cp: impossível obter estado de "bashrc": Arquivo ou diretório não encontrado
wmakeInInclude: linking include files to ./lnInclude
Making dependency list for source file porousReactingZone.C
SOURCEporousReactingZone.C ; g++ -m64 -Dlinux64 -DWM_DP -Wall -Wextra -Wno-unused-parameter -Wold-style-cast -Wnon-virtual-dtor -O3 -DNoRepository -ftemplate-depth-100
-I/opt/OpenFOAM-2.1.1/src/triSurface/lnInclude -I/opt/OpenFOAM-2.1.1/src/meshTools/lnInclude -I/opt/OpenFOAM-2.1.1/src/finiteVolume/lnInclude -lnInclude -I. -I/opt/Open
FOAM-2.1.1/src/OpenFOAM/lnInclude -I/opt/OpenFOAM-2.1.1/src/OSspecific/POSIX/lnInclude -fPIC -c $SOURCE -o Make/linux64GccDPopt/porousReactingZone.o
In file included from /opt/OpenFOAM-2.1.1/src/OpenFOAM/lnInclude/Field.H:360:8,
      from /opt/OpenFOAM-2.1.1/src/OpenFOAM/lnInclude/scalarField.H:38,
      from /opt/OpenFOAM-2.1.1/src/OpenFOAM/lnInclude/vectorField.H:38,
      from /opt/OpenFOAM-2.1.1/src/meshTools/lnInclude/coordinateSystem.H:134,
      from porousReactingZone.H:63,
      from porousReactingZone.C:26:
/opt/OpenFOAM-2.1.1/src/OpenFOAM/lnInclude/Field.C: In member function 'void Foam::Field<Type>::operator=(const Foam::VectorSpace<Form, Cmpt, nCmpt>&)':
/opt/OpenFOAM-2.1.1/src/OpenFOAM/lnInclude/Field.C:680:42: warning: typedef 'VSType' locally defined but not used [-Wunused-local-typedefs]
    typedef VectorSpace<Form, Cmpt, nCmpt> VSType;
      ^
'/opt/OpenFOAM-2.1.1/biomassGasificationMedia/lib/libporousReactingZone.so' is up to date.
./install: 23: ./install: ./Allwmake: Permission denied
wmakeInInclude: linking include files to ./lnInclude
Making dependency list for source file pyrolysisModel/heterogeneousPyrolysisModel/heterogeneousPyrolysisModel.C
could not open file solidChemistryModel.H for source file pyrolysisModel/heterogeneousPyrolysisModel/heterogeneousPyrolysisModel.C
could not open file basicSolidThermo.H for source file pyrolysisModel/heterogeneousPyrolysisModel/heterogeneousPyrolysisModel.C
could not open file psiChemistryModel.H for source file pyrolysisModel/heterogeneousPyrolysisModel/heterogeneousPyrolysisModel.C
could not open file heatTransferModel.H for source file pyrolysisModel/heterogeneousPyrolysisModel/heterogeneousPyrolysisModel.C
Making dependency list for source file pyrolysisModel/heterogeneousPyrolysisModel/heterogeneousPyrolysisModelNew.C
```

Figura 36 – Erro devido à permissão do arquivo *Allwmake*

Com as permissões liberadas, para o usuário executar o *script install* basta usar o seguinte comando em um terminal no diretório *biomassGasificationFoam_installPack_1.0*, que contém o arquivo *install*:

```
./install
```

Esse *script* realizará a instalação do *biomassGasificationFoam*, ou seja, a compilação do *solver*, das bibliotecas auxiliares e dos utilitários. Devido a um erro em um arquivo de compilação de uma biblioteca, o usuário irá se deparar com o erro “Sem regra para processar o alvo *reaction/Reactions/solidHeterogeneousReaction.dep*”, destacado na [Figura 37](#).

```
caio@caio-notebook:~/OpenFOAM/cao-2.1.1/biomassGasificationFoam_installPack_1.0/biomassGasificationFoam_installPack_1.0$ ./install
./install: 3: ./install: source: not found
mkdir: é impossível criar o diretório "/opt/OpenFOAM-2.1.1": Arquivo existe
mkdir: é impossível criar o diretório "/opt/OpenFOAM-2.1.1/applications": Arquivo existe
cp: impossível obter estado de "bashrc": Arquivo ou diretório não encontrado
wmakeInInclude: linking include files to ./lnInclude
Making dependency list for source file porousReactingZone.C
SOURCEporousReactingZone.C ; g++ -m64 -Dlinux64 -DWM_DP -Wall -Wextra -Wno-unused-parameter -Wold-style-cast -Wnon-virtual-dtor -O3 -DNoRepository -ftemplate-depth-100
-I/opt/OpenFOAM-2.1.1/src/triSurface/lnInclude -I/opt/OpenFOAM-2.1.1/src/meshTools/lnInclude -I/opt/OpenFOAM-2.1.1/src/finiteVolume/lnInclude -lnInclude -I. -I/opt/Open
FOAM-2.1.1/src/OpenFOAM/lnInclude -I/opt/OpenFOAM-2.1.1/src/OSspecific/POSIX/lnInclude -fPIC -c $SOURCE -o Make/linux64GccDPopt/porousReactingZone.o
In file included from /opt/OpenFOAM-2.1.1/src/OpenFOAM/lnInclude/Field.H:360:8,
      from /opt/OpenFOAM-2.1.1/src/OpenFOAM/lnInclude/scalarField.H:38,
      from /opt/OpenFOAM-2.1.1/src/OpenFOAM/lnInclude/vectorField.H:38,
      from /opt/OpenFOAM-2.1.1/src/meshTools/lnInclude/coordinateSystem.H:134,
      from porousReactingZone.H:63,
      from porousReactingZone.C:26:
/opt/OpenFOAM-2.1.1/src/OpenFOAM/lnInclude/Field.C: In member function 'void Foam::Field<Type>::operator=(const Foam::VectorSpace<Form, Cmpt, nCmpt>&)':
/opt/OpenFOAM-2.1.1/src/OpenFOAM/lnInclude/Field.C:680:42: warning: typedef 'VSType' locally defined but not used [-Wunused-local-typedefs]
    typedef VectorSpace<Form, Cmpt, nCmpt> VSType;
      ^
'/opt/OpenFOAM-2.1.1/biomassGasificationMedia/lib/libporousReactingZone.so' is up to date.
+ wmake libso solid
wmakeInInclude: linking include files to ./lnInclude
Making dependency list for source file rhoType/const/constRho.C
Making dependency list for source file rhoType/heterogeneousConst/heterogeneousConstRho.C
make: *** Sem regra para processar o alvo 'reaction/Reactions/solidHeterogeneousReaction/solidHeterogeneousReaction.dep', necessário por 'Make/linux64GccDPopt/dependencie
s'. Pare.
+ wmake libso chemistryModel
wmakeInInclude: linking include files to ./lnInclude
Making dependency list for source file chemistryModel/basicChemistryModel/basicChemistryModel.C
Making dependency list for source file chemistryModel/psiChemistryModel/psiChemistryModel.C
Making dependency list for source file chemistryModel/psiChemistryModel/psiChemistryModelNew.C
Making dependency list for source file chemistryModel/psiChemistryModel/psiChemistryModels.C
Making dependency list for source file chemistryModel/rhoChemistryModel/rhoChemistryModel.C
Making dependency list for source file chemistryModel/rhoChemistryModel/rhoChemistryModelNew.C
Making dependency list for source file chemistryModel/rhoChemistryModel/rhoChemistryModels.C
Making dependency list for source file chemistrySolver/chemistrySolver/makeChemistrySolvers.C
```

Figura 37 – Erro ao compilar o conjunto de bibliotecas *solid*, dos modelos termofísicos

Esse erro afeta a compilação subsequente de outras bibliotecas. O problema se encontra na letra “R” maiúscula do sub-diretório *Reactions*, indicado com uma seta na [Figura 37](#). O diretório real é escrito com letra “r” minúscula. Para resolver o problema, deve-se alterar o arquivo *files*,

presente no diretório `/biomassGasificationMedia/thermophysicalModels/solid/Make`. Altere a terceira linha, trocando o “R” maiúsculo do sub-diretório *Reactions* para “r” minúsculo, como mostrado na [Figura 38](#).

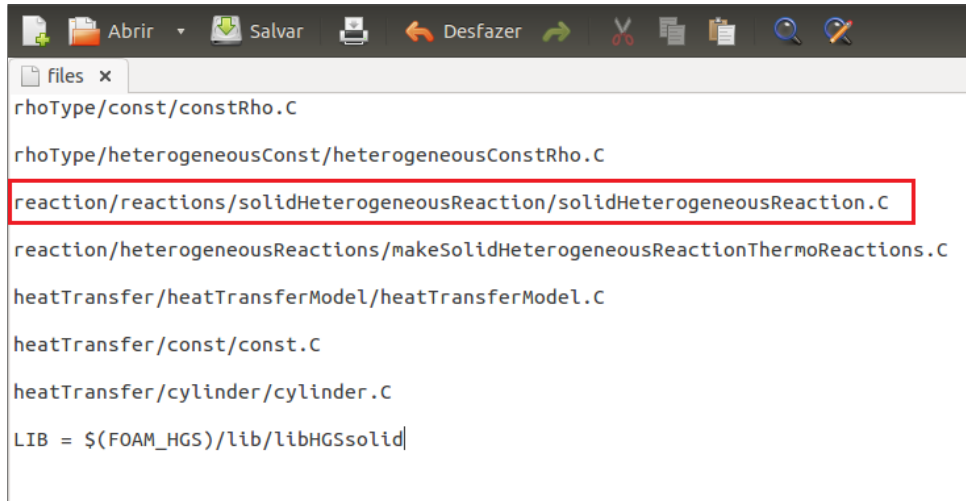


Figura 38 – Arquivo *files* alterado

Finalmente, após esses procedimentos descritos acima, o usuário conseguirá, supostamente, realizar a execução do *script install* e a instalação do `biomassGasificationFoam` sem mais problemas. Após concluído, para testar se a instalação funcionou basta executar o seguinte comando em um terminal:

```
biomassGasificationFoam
```

Se o procedimento obteve sucesso, irá aparecer na tela o cabeçalho do programa.

4.3 Campo de Porosidade Dinâmico

Conforme comentado na [seção 2.4](#), os *solvers* fornecidos na distribuição padrão do OpenFOAM para a modelagem da pirólise, escoamento com reações químicas e gaseificação em leito fluidizado (`fireFoam`, `reactingFoam` e `coalChemistryFoam`, respectivamente) não são adequados para a simulação de processos de conversão termoquímica de sólidos em leito fixo. Eles não consideram a mútua influência entre o fluxo reativo e o sólido, e as reações volumétricas que ocorrem no material poroso.

A abordagem do Campo de Porosidade Dinâmico foi então proposta por [Kwiatkowski et al. \(2013\)](#) para permitir a implementação de processos térmicos volumétricos e a modelagem da interação gás-sólido, introduzindo o meio poroso no domínio computacional como um campo volumétrico de porosidade. A modelagem do escoamento nessa abordagem segue o chamado Método do Domínio Fictício ([KHADRA et al., 2000](#)), que é uma versão especial do Método da Fronteira Imersa ([MITTAL; IACCARINO, 2005](#)). Na implementação do `biomassGasificationFoam`, as equações dentro e fora do meio poroso são as mesmas e a fronteira não possui nenhuma dinâmica própria. A fronteira do material poroso é a superfície onde os parâmetros físicos presentes nessas

equações são descontínuos. Um exemplo é a equação do momento, onde o termo da Lei de Darcy e o termo viscoso de um fluido Newtoniano estão formalmente presentes ambos os lados da fronteira, mas os coeficientes associados apresentam um salto na interface.

O meio poroso é definido por dois campos: um escalar (porosidade) e um tensorial (resistência viscosa). A fronteira do material poroso não possui uma dinâmica intrínseca. Assim, não é necessário detectar o seu formato preciso, como é o caso, por exemplo, com fronteiras elásticas ou com tensões superficiais onde a curvatura local é criticamente importante. Dessa forma, a implementação numérica se torna muito simplificada, podendo-se assumir que cada célula da malha se encontra completamente dentro ou fora do meio poroso.

Muitos códigos CFD tratam o meio poroso como uma zona separada no próprio domínio computacional, e uma vez definido no estágio de pré-processamento, não pode ser facilmente modificado sem mudar o domínio completo. A abordagem implementada no `biomassGasificationFoam` define o meio poroso como campos dentro do domínio (campos volumétricos de porosidade não nula e de resistência viscosa), que são acessíveis e facilmente modificados, se necessário, durante os cálculos.

O novo conjunto de bibliotecas implementadas no `biomassGasificationFoam` introduzem as propriedades térmicas, químicas e radiativas do campo poroso, e determinam as suas interações com a fase gasosa escoando através e em torno desse campo. As bibliotecas tratam o sólido (idealizado para biomassa) como um conjunto de campos escalares e tensoriais adicionais dentro do domínio computacional, por exemplo, a fração mássica de seus componentes, como celulose, a condutividade térmica e a densidade absoluta.

A introdução do Campo de Porosidade Dinâmico permite que os processos térmicos sejam considerados como fenômenos volumétricos que podem ocorrer dentro de todo o material poroso, dependendo das condições locais. Isso torna possível simular efeitos como a frente de reações dentro do sólido e os processos térmicos e químicos no meio poroso, que envolvem a interação mútua entre as fases sólida e gasosa. Kwiatkowski et al. (2013) acreditam que aplicações com essa abordagem excedem as simulações atuais de pirólise, combustão e gaseificação. E também que ela pode ser potencialmente aplicada na modelagem de qualquer processo envolvendo um meio poroso reativo, como catálise ou filtragem.

O modelo de porosidade é implementado na biblioteca `porousReactingMedia`, que pertence ao conjunto de bibliotecas `biomassGasificationMedia` (Figura 32). O meio poroso é introduzido no domínio computacional principal por dois campos volumétricos: um campo escalar (`volScalarField`) chamado *porosityF*; e um campo tensorial (`volTensorField`) chamado *Df*.

O campo tensorial *Df* representa o tensor de resistência viscosa \mathbf{K} (ver Equação 4.4). Para um material isotrópico, ele é um tensor cujos valores da diagonal são o inverso da permeabilidade e os outros valores são nulos, mas também é possível defini-lo para um material anisotrópico.

O campo escalar *porosityF* representa a fração de sólido, ou seja, um menos a fração de vazio. Alguns comentários são importantes em relação à esse campo e ao termo porosidade. É comum na literatura que o termo porosidade se refira à fração em volume de vazio (CUNHA,

2010):

$$\epsilon = \frac{V_g}{V_T} \quad (4.1)$$

Onde V_g é o volume de vazios que pode ser ocupado pelo gás na estrutura porosa e V_T é o volume total (volume de sólido + volume de vazios). Porém, Kwiatkowski et al. (2013), em seu trabalho, tratam a porosidade como a fração de sólido, ou seja, o volume ocupado pelo sólido na estrutura porosa dividido pelo volume total (V_s/V_T). A fração de vazio é referida por eles como *void fraction*, representado pela letra grega γ . Com o intuito de evitar confusão e oferecer maior clareza ao leitor, no restante deste trabalho serão priorizados os termos fração de sólido e fração de vazio, seguindo as definições dadas por Kwiatkowski et al. (2013), ou seja, a fração de sólido é a porosidade e a fração de vazio é o *void fraction* (γ), dado por:

$$\gamma = \frac{V_T - V_s}{V_T} = \frac{V_g}{V_T} \quad (4.2)$$

Assim sendo, $\gamma = 0$ quando apenas a fase sólida está presente, e $\gamma = 1$ quando não há sólido. Com essas definições, é importante ressaltar que o campo *porosityF* refere-se à fração de sólido ($1 - \gamma$).

As características iniciais do meio poroso são definidas com o utilitário `setPorosity`, que cria os dois campos volumétricos: *porosityF* e *Df*. Esses campos são criados usando restrições de coordenadas dos centros de cada célula computacional da malha do domínio, criada previamente. Kwiatkowski et al. (2013) comentam que, pelas suas experiências, o método mais vantajoso e eficiente para criar um meio poroso específico para o usuário é alterando o arquivo *medium.H* do utilitário `setPorosity` e recompilando-o pelo método padrão de compilação do OpenFOAM.

Detalhando melhor o procedimento para a criação do meio poroso, o usuário deverá, primeiramente, alterar o arquivo *medium.H*, pertencente ao utilitário `setPorosity`, de maneira que melhor represente o seu problema, inserindo valores para a porosidade e resistência viscosa. Em seguida, deve compilar o utilitário, abrindo um terminal em seu diretório (*biomassGasificationFoam_installPack_1.0/setPorosity*) e executando o comando:

```
wmake
```

Com isso, o utilitário `setPorosity` estará programado para criar um meio poroso específico ao ser aplicado para o caso a ser simulado. O próximo passo então é ir no diretório raiz do caso a ser simulado e executar o utilitário em um terminal, com o comando:

```
setPorosity
```

Assim, no diretório “0” do caso estarão criados os campos *porosityF* e *Df*. O usuário poderá notar também, com os exemplos fornecidos com o `biomassGasificationFoam`, que no diretório “0” também deve estar presente um campo chamado *porosityF0*. Esse campo não é criado com o `setPorosity`, cabendo ao usuário essa tarefa. Ele representa o valor inicial do campo de

porosidade, não sofrendo alterações durante o processo de cálculo. Logo, seu conteúdo pode ser copiado do arquivo *porosityF* criado. O observado pelo autor deste trabalho é que o campo *porosityF0* só é utilizado para o cálculo da evolução da área de troca térmica (na transferência de calor), como comentado na [subseção 4.8.3](#), onde ele representa a porosidade inicial na [Equação 4.35](#).

Atualmente, a biblioteca `porousReactingMedia` inclui apenas a Lei de Darcy para um meio poroso, mas pode ser estendida para outras fórmulas, como Forchheimer.

A biblioteca `heterogeneousPyrolysisModel`, inclusa no conjunto `biomassGasificationMedia` ([Figura 32](#)), é uma classe que cria os campos de propriedades do sólido e integra os submodelos físicos para a fase sólida, como radiação, cinética química e condução, por exemplo. Ela será comentada mais detalhadamente na [seção 4.9](#). Uma das equações que são resolvidas por essa biblioteca é a variação da porosidade (campo *porosityF*), pela função *evolvePorosity()* da classe derivada *volPyrolysis*, que invoca também a função *RRpor(volScalarField T)* da classe *ODESolidHeterogeneousChemistryModel*:

$$\frac{d(\text{porosity}F)}{dt} = \sum_k \frac{\sum_r R_{k,r}}{\tilde{\rho}_k} \quad (4.3)$$

Onde $\tilde{\rho}_k$ é a massa específica absoluta (massa específica sem fração de vazio) do componente sólido k , e $R_{k,r}$ é a geração ou consumo de massa do sólido k pela reação r . A fração de vazio evolui conforme a fase sólida vai desaparecendo pelo consumo nas reações químicas. Em qualquer célula computacional, quando a porosidade (fração de sólido) for suficientemente perto de zero, isto é, porosidade $< 10^{-4}$, ela é fixa em zero. É considerado que tal célula computacional contém apenas a fase gasosa e não participa na futura evolução da fase sólida. Essas informações diferem um pouco das fornecidas no artigo de [Kwiatkowski et al. \(2013\)](#). Porém, estão em conformidade com o que está implementado nos códigos. A [Equação 4.3](#) também está em conformidade com [Cunha \(2010\)](#).

4.4 Hipóteses da Modelagem

O `biomassGasificationFoam` apresenta um modelo matemático tridimensional e transiente para a conversão termoquímica de sólidos, baseado em equações de conservação para ambas as fases, sólida e gasosa. As hipóteses para esse modelo são:

- O sólido é um meio poroso isotrópico ou anisotrópico.
- O movimento ou colapso do meio poroso é negligenciado. Apenas é calculada a variação do campo de porosidade.
- A umidade está incorporada dentro da estrutura porosa. Umidade liberada a partir do sólido evapora-se imediatamente.

- A possível fase líquida (água e hidrocarbonetos líquidos) é considerada vapor.
- O gás escoar dentro do meio poroso conforme a Lei de Darcy. Outras fórmulas podem ser implementadas. O escoamento pode ser turbulento ou laminar.
- É possível considerar a transferência de calor por radiação a partir das paredes do domínio para a superfície do sólido. Porém, considera-se que a superfície do sólido apenas absorve radiação, e não emite.
- Os processos térmicos e as reações químicas podem levar ao não-equilíbrio térmico, ou seja, em um mesmo local, a temperatura do sólido pode ser diferente da temperatura do gás. Isso leva à duas equações de conservação da energia, uma para a fase sólida e outra para a fase gasosa.
- O meio poroso perde massa por meio de processos homogêneos e heterogêneos.

4.5 Equações de Conservação

As equações governantes implementadas para a conversão termoquímica de sólidos são as equações de conservação do momento, da continuidade, conservação das espécies e da energia para a fase gasosa, e conservação das espécies e da energia para a fase sólida. Todas essas equações são resolvidas para todo o domínio computacional. No entanto, onde o campo de porosidade (fração de sólido) for igual a zero, os parâmetros para a fase sólida, como massa específica, condutividade térmica e calor específico, são considerados zero.

Essas equações são apresentadas de modo que todos os termos do lado esquerdo da equação são termos padrões para a modelagem do escoamento de gases e da troca térmica em sólidos. No lado direito estão os termos fontes referentes à estrutura porosa do sólido, às reações químicas e às conversões térmicas.

4.5.1 Conservação do Momento

$$\frac{\partial \gamma \rho^G \mathbf{u}}{\partial t} + \nabla \cdot (\rho^G \mathbf{u} \mathbf{u}) + \nabla p - \nabla \cdot (\mu \nabla \mathbf{u}) = (1 - \gamma) \mathbf{K} \mu \mathbf{u} \quad (4.4)$$

Onde γ é a fração de vazio, ρ^G é massa específica do gás, \mathbf{u} é a velocidade do gás, μ é a viscosidade dinâmica do gás e \mathbf{K} é o tensor de resistência viscosa (representado pelo campo tensorial Df). O termo do lado direito está relacionado com a Lei de Darcy para meio poroso.

Essa equação está implementada no arquivo *UEqn.H* do *biomassGasificationFoam*, fazendo uso também da biblioteca *porousReactingMedia*.

4.5.2 Continuidade

$$\frac{\partial \gamma \rho^G}{\partial t} + \nabla \cdot (\rho^G \mathbf{u}) = \sum_i (R_i^{G^{evap}} + R_i^{G^{pir}} + R_i^{G^{gaseif}}) \quad (4.5)$$

Os termos do lado direito são termos fontes relacionados à taxa de reação R , em $\text{kg}/\text{m}^3/\text{s}$, para a espécie gasosa i , provenientes da decomposição do sólido. Esse termo é a taxa de consumo ou formação das espécies gasosas devido às reações do sólido. Os sobrescritos *evap*, *pir* e *gaseif* correspondem aos processos de evaporação, pirólise e gaseificação, respectivamente.

Essa equação está implementada no arquivo *rhoEqn.H* do `biomassGasificationFoam`, onde a variável *Srho* corresponde à somatória do lado direito, e foi criada no arquivo *chemistry.H*, que invoca a função *Srho()* da classe *volPyrolysis*. As taxas de reação, R_i^G , são calculadas pela classe *ODESolidHeterogeneousChemistryModel* e guardadas na variável *RRg_*.

4.5.3 Conservação das Espécies para a Fase Sólida

$$\frac{\partial \rho^S Y_k^S}{\partial t} = R_k^{S^{pir}} + R_k^{S^{gaseif}} + R_k^{S^{comb}} \quad (4.6)$$

Onde ρ^S é a massa específica aparente do sólido e Y_k^S é a fração mássica do componente k do sólido. Os termos do lado direito são termos fontes relacionados à taxa de reação R , em $\text{kg}/\text{m}^3/\text{s}$, para a espécie sólida k . Esse termo é a taxa de consumo ou formação das espécies sólidas devido às reações do sólido. Os sobrescritos *pir*, *gaseif* e *comb* correspondem aos processos de pirólise, gaseificação e combustão, respectivamente.

Essa equação é calculada na classe *volPyrolysis*, pela função *solveSpeciesMass()*. As taxas de reação, R_k^S , são calculadas pela classe *ODESolidHeterogeneousChemistryModel* e guardadas na variável *RRs_*.

4.5.4 Conservação das Espécies para a Fase Gasosa

$$\frac{\partial \gamma \rho^G Y_i^G}{\partial t} + \nabla \cdot (\rho^G \mathbf{u} Y_i^G) - \nabla \cdot (\rho^G D^* \nabla Y_i^G) = \omega_i^G + R_i^{G^{evap}} + R_i^{G^{pir}} + R_i^{G^{gaseif}} \quad (4.7)$$

Onde Y_i^G é a fração mássica da espécie gasosa i e D^* é a difusividade mássica efetiva da fase gasosa. Os termos do lado direito representam as fontes de consumo ou produção da espécie i devido às reações químicas. As taxas de reação R_i^G , comentadas anteriormente, são devido às reações do sólido. O termo ω_i^G é a taxa de reação referente às reações homogêneas da fase gasosa.

A [Equação 4.7](#) está escrita conforme consta em [Kwiatkowski et al. \(2013\)](#). A sua implementação, no arquivo *YEqn.H* do `biomassGasificationFoam`, apresenta uma simplificação que a diferencia da [Equação 4.7](#). É considerado que o Número de Schmidt é igual a um, ou seja, que

a difusividade de momento e a difusividade mássica são iguais. O Número de Schmidt é definido por:

$$Sc = \frac{\nu}{D} \quad (4.8)$$

Onde ν é a difusividade de momento (viscosidade cinemática) e D é a difusividade mássica. A hipótese de que o Número de Schmidt é igual a um implica que:

$$Sc = \frac{\nu}{D} = 1 \rightarrow D = \nu = \frac{\mu}{\rho} \rightarrow \rho D = \mu \quad (4.9)$$

Assim sendo, a implementação da conservação das espécies para a fase gasosa no arquivo *YEqn.H* substitui o termo $\rho^G D^*$ (Equação 4.7) pela viscosidade dinâmica μ . A equação implementada, com essa simplificação, é:

$$\frac{\partial \gamma \rho^G Y_i^G}{\partial t} + \nabla \cdot (\rho^G \mathbf{u} Y_i^G) - \nabla \cdot (\mu \nabla Y_i^G) = \omega_i^G + R_i^{G^{evap}} + R_i^{G^{pir}} + R_i^{G^{gaseif}} \quad (4.10)$$

Essa hipótese simplificadora é proveniente do fato do próprio OpenFOAM não possuir uma biblioteca para tratar a difusividade mássica. Uma sugestão para trabalhos futuros é a implementação de uma biblioteca para tal tarefa, e a sua utilização no `biomassGasificationFoam`. O trabalho de [Novaresio et al. \(2012\)](#) pode servir como base, junto com a sua biblioteca `multiSpeciesTransportModels`, para a modelagem da difusão de múltiplas espécies. Na página da internet do fórum *cf-d-online.com*, existe um tópico onde é possível encontrar mais informações sobre essa biblioteca, com versões para download e também o trabalho de [Novaresio \(2012\)](#). A página é “<http://www.cfd-online.com/Forums/openfoam/103227-multi-species-mass-transport-library-update.html>”.

4.5.5 Conservação da Energia para a Fase Sólida

$$\frac{\partial \rho^S C_p^S T^S}{\partial t} - \nabla \cdot (\gamma k^S \nabla T^S) = \alpha \Sigma (T^G - T^S) + H_r + S^{S^{rad}} \quad (4.11)$$

Onde T^S , C_p^S e k^S são a temperatura, o calor específico e a condutividade térmica do sólido, respectivamente. Os termos do lado direito representam os termos fontes de energia provenientes dos processos térmicos. O primeiro corresponde à transferência de calor entre o sólido e o gás, onde α é o coeficiente de transferência de calor e Σ é a área superficial específica, ou área superficial dos poros por volume de sólido. Esse termo é devido à hipótese de não-equilíbrio térmico local, apresentando uma diferença entre as temperaturas do sólido e do gás. O segundo termo, H_r , é a energia total devido às reações químicas da fase sólida. O terceiro termo, $S^{S^{rad}}$, é a energia proveniente da radiação para a fase sólida.

A equação de conservação da energia para a fase sólida é resolvida na classe *volPyrolysis*, pela função *solveEnergy()*.

4.5.6 Conservação da Energia para a Fase Gasosa

$$\frac{\partial \gamma \rho^G C_p^G T^G}{\partial t} + \mathbf{u} \cdot (\nabla \rho^G C_p^G T^G) - \nabla \cdot ((1 - \gamma) k^G \nabla T^G) = \sum_i \omega_i^G h_{f_i}^G - \alpha \Sigma (T^G - T^S) - \Gamma + S^{G^{rad}} \quad (4.12)$$

Onde T^G , C_p^G e k^G são a temperatura, o calor específico e a condutividade térmica do gás, respectivamente. Os termos do lado direito representam os termos de fonte de energia para a fase gasosa. O primeiro é a energia total proveniente das reações homogêneas na fase gasosa, onde ω_i^G é a taxa de consumo ou produção para o gás i , e $h_{f_i}^G$ é a sua entalpia de formação. O segundo termo é referente à transferência de calor entre o sólido e o gás. O terceiro termo, Γ , representa a energia necessária para aquecer ou resfriar os gases produzidos nas reações do sólido para a temperatura dos gases. O último termo, $S^{G^{rad}}$, é a energia proveniente da radiação para a fase gasosa.

A [Equação 4.12](#) está escrita conforme consta em [Kwiatkowski et al. \(2013\)](#). Porém, a equação para a conservação da energia da fase gasosa implementada no `biomassGasificationFoam` é diferente. Ela considera a conservação da entalpia sensível (h_s) do gás, e está implementada no arquivo `hsEqn.H`. Essa implementação para a entalpia sensível foi herdada do `solver reactingFoam` e ajustada para um meio poroso, conforme a [Equação 4.13](#):

$$\frac{\partial \gamma \rho^G h_s}{\partial t} + \nabla \cdot (\rho^G \mathbf{u} h_s) - \nabla \cdot (\alpha_{ef} \nabla h_s) = \frac{\partial p}{\partial t} - \frac{\partial \gamma \rho^G K}{\partial t} - \nabla \cdot (\rho^G \mathbf{u} K) + H_r^G - \alpha \Sigma (T^G - T^S) - \Gamma + S^{G^{rad}} \quad (4.13)$$

Onde h_s é a entalpia sensível, α_{ef} é a difusividade térmica efetiva e K é a energia cinética específica ($K = |\mathbf{u}|^2/2$, ver arquivos `createFields.H` e `pEqn.H`). O termo H_r^G é a energia total proveniente das reações homogêneas na fase gasosa, dada pela [Equação 4.29](#).

4.6 Implementação do Solver `biomassGasificationFoam`

O `biomassGasificationFoam` é baseado no algoritmo utilizado pelo `solver reactingFoam`, para escoamentos transientes, laminares ou turbulentos, onde ocorrem reações químicas em fase gasosa. Os modelos de turbulência são os fornecidos pela distribuição padrão do OpenFOAM. O `reactingFoam` trata do escoamento de misturas gasosas, portanto resolve apenas reações para a fase gasosa, sem os termos fontes relacionados ao sólido poroso e às interações entre o sólido e o gás, vistos na seção anterior.

Para adequar o *solver* à modelagem da conversão termoquímica de sólidos em leito fixo, as principais modificações implementados no `biomassGasificationFoam` foram (KWIATKOWSKI et al., 2013):

- Introdução de um meio poroso reativo;
- Introdução das equações de conservação para a fase sólida: conservação dos componentes sólidos (Equação 4.6) e conservação da energia (Equação 4.11);
- A equação da continuidade (Equação 4.5) e da conservação das espécies para a fase gasosa (Equação 4.10) possuem termos fontes devido aos gases gerados nas reações da fase sólida;
- A equação da conservação da energia para a fase gasosa (Equação 4.13) possui termos fontes adicionais devidos às trocas térmicas entre o sólido e o gás;
- A absorção de radiação pela superfície do sólido é possível de ser considerada.

Como comentado na seção 4.1, o código do `biomassGasificationFoam` é fornecido contendo quatro componentes, sendo que os dois utilitários (de pré e pós-processamento) foram abordados em seções anteriores. Os outros dois são os responsáveis pelo processo de simulação do problema. Um é o *solver* `biomassGasificationFoam` propriamente dito, que apresenta um algoritmo para sequenciar a solução e integrar as demais bibliotecas, e o outro é o conjunto de bibliotecas `biomassGasificationMedia`, utilizado pelo *solver*. Conforme visto em seu diretório, o *solver* é composto por vários arquivos: um arquivo fonte principal, de extensão “.C”; e um conjunto de arquivos *header*, de extensão “.H”, que são adicionados na compilação do executável.

O `biomassGasificationFoam` utiliza o algoritmo PIMPLE (PISO-SIMPLE), assim como o `reactingFoam`. Uma ilustração esquemática do *loop* do *solver* `biomassGasificationFoam` é apresentada na Figura 39. Os textos em negrito representam os novos desenvolvimentos implementados no *solver*. O *loop* principal contém duas correções internas para solucionar o acoplamento pressão-velocidade influenciado pelas distribuições de energia e espécies. As equações de conservação de massa e energia para o sólido são solucionadas e os termos fontes calculados apenas uma vez por *loop* principal.

A estrutura mostrada na Figura 39 pode ser identificada ao se analisar os códigos do *solver* `biomassGasificationFoam`, comentados anteriormente, principalmente o código fonte `biomassGasificationFoam.C`. Nos próximos parágrafos, os passos indicados na Figura 39 serão identificados em trechos de código contidos no arquivo fonte `biomassGasificationFoam.C`, incluso no Anexo A.

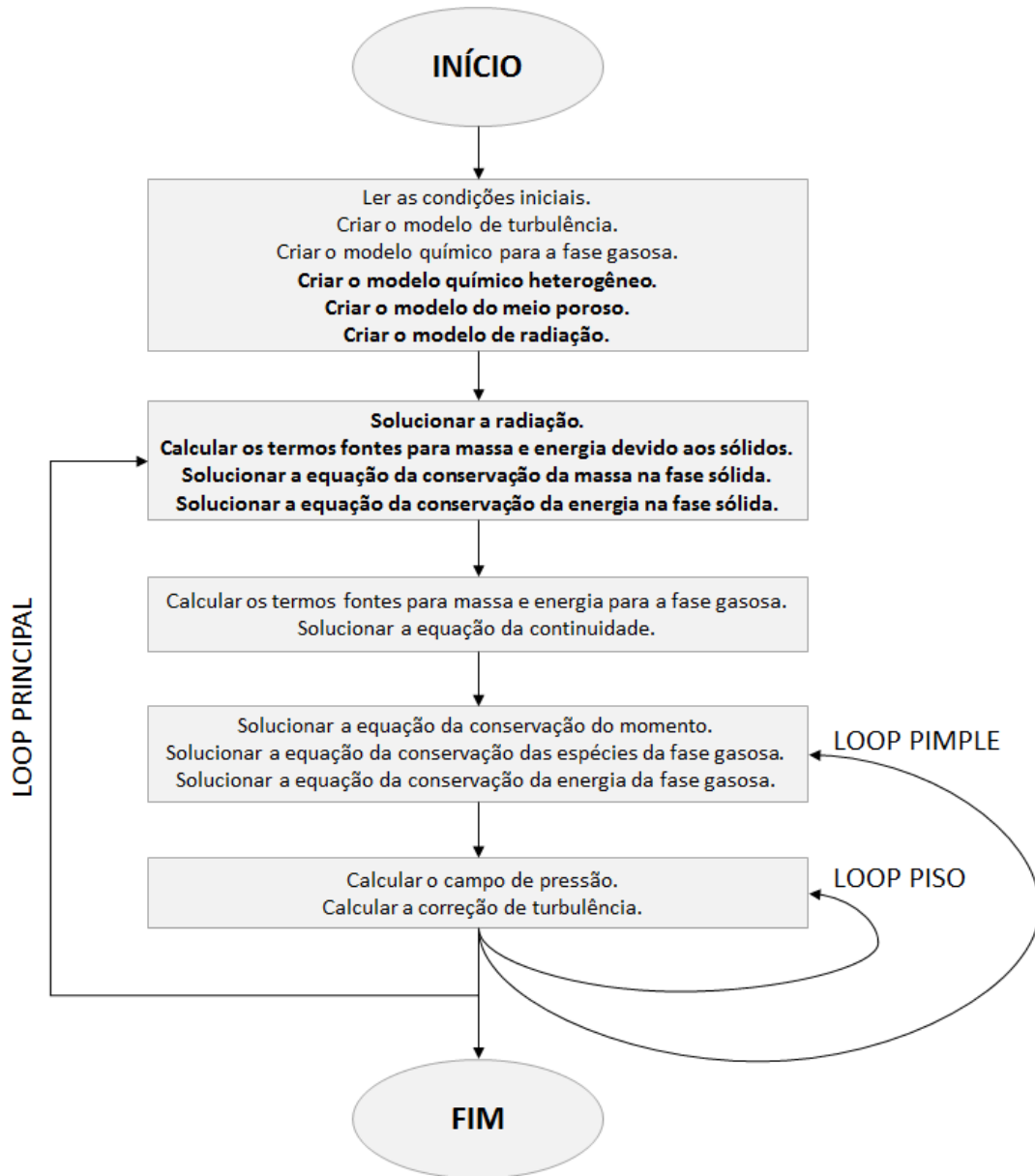


Figura 39 – Esquema do *loop* do *solver* *biomassGasificationFoam*

Fonte: Adaptado de Kwiatkowski et al. (2013)

O primeiro quadro da [Figura 39](#) contém a leitura das condições iniciais e a criação dos modelos de turbulência, de radiação, do meio poroso e os modelos químicos. Esses passos são realizados nos dois conjuntos de comandos iniciais que incluem arquivos *header*, mostrados abaixo. Eles também são responsáveis por realizar a preparação do ambiente do OpenFOAM e das ferramentas necessárias para o cálculo, como a aplicação do método do volumes finitos e a criação da malha.

```

1 /*-----*/
2
3 #include "fvCFD.H"
4 #include "turbulenceModel.H"

```

```

5 #include "psiChemistryModel.H"
6 #include "chemistrySolver.H"
7 #include "porousReactingZone.H"
8 #include "multivariateScheme.H"
9 #include "pimpleControl.H"
10 #include "basicHGSSolidThermo.H"
11 #include "solidChemistryModel.H"
12 #include "mapDistribute.H"
13 #include "heterogeneousPyrolysisModel.H"
14 #include "heterogeneousRadiationModel.H"
15
16 // * * * * *
17
18 int main(int argc, char *argv[])
19 {
20     #include "setRootCase.H"
21     #include "createTime.H"
22     #include "createMesh.H"
23     #include "readChemistryProperties.H"
24     #include "readGravitationalAcceleration.H"
25     #include "createFields.H"
26     #include "initContinuityErrs.H"
27     #include "readTimeControls.H"
28     #include "compressibleCourantNo.H"
29     #include "setInitialDeltaT.H"
30     #include "createPyrolysisModel.H"
31     #include "readPyrolysisTimeControls.H"
32     #include "createPorosity.H"
33     #include "createHeterogeneousRadiationModel.H"
34
35     pimpleControl pimple(mesh);
36
37     // * * * * *

```

O *loop* principal começa em:

```

1 Info << "\nStarting time loop\n" << endl;
2
3 while (runTime.run())
4 {

```

Os passos presentes no segundo quadro da [Figura 39](#), ou seja, a solução da radiação, o cálculo dos termos fontes de massa e energia devido aos sólidos, e a solução das equações de conservação de massa e energia para a fase sólida, são executados pelas linhas:

```

1     #include "radiation.H"
2
3     pyrolysisZone.evolve();

```

Nesse código, *pyrolysisZone* é um objeto da classe *heterogeneousPyrolysisModel* (comentada na [seção 4.9](#)), que foi criado pelo arquivo *createPyrolysisModel.H*. A função *evolve()* invoca todos os cálculos relacionados à fase sólida.

O terceiro quadro da [Figura 39](#) corresponde às linhas:

```

1      #include "chemistry.H"
2      #include "rhoEqn.H"

```

Onde o arquivo *chemistry.H* é responsável por invocar as bibliotecas e organizar os cálculos para as reações químicas da fase gasosa, obtendo os termos fontes de massa e energia devido à essas reações. Ele também trata de reações turbulentas. As bibliotecas utilizadas para modelar essas reações químicas homogêneas da fase gasosa são as padrões usadas pelo *solver reactingFoam*. O arquivo *rhoEqn.H* faz a solução da equação da continuidade.

O quarto quadro da [Figura 39](#) apresenta o início do *loop* PIMPLE e a solução das equações de conservação do momento, das espécies para a fase gasosa e da energia para a fase gasosa. Esses passos estão representados nas linhas:

```

1      while (pimple.loop())
2      {
3          #include "UEqn.H"
4          #include "YEqn.H"
5          #include "hSEqn.H"

```

Os arquivos *UEqn.H*, *YEqn.H* e *hSEqn.H* realizam a solução das equações de conservação do momento, das espécies e da energia para a fase gasosa, respectivamente. Eles são baseados nos arquivos presentes no *solver reactingFoam*, porém adaptados para incluírem o sólido poroso e as interações gás-sólido.

O quinto e último quadro da [Figura 39](#) inicia o *loop* PISO e realiza a solução do campo de pressão e a correção devido à turbulência. Esses passos correspondem à:

```

1      // — PISO loop
2      while (pimple.correct())
3      {
4          #include "pEqn.H"
5      }
6      if (pimple.turbCorr())
7      {
8          turbulence->correct();
9      }

```

4.7 biomassGasificationMedia: Bibliotecas Auxiliares

A *biomassGasificationMedia* é um conjunto de bibliotecas auxiliares utilizado pelo *solver biomassGasificationFoam* para a modelagem da conversão termoquímica de sólidos em leito fixo. Ela é composta de três elementos, *porousReactingMedia*, *thermophysicalModels* e *pyrolysisModels*, como mostrado na [Figura 32](#).

A *porousReactingMedia* é uma biblioteca para a definição do meio poroso e das suas propriedades mecânicas, em conjunto com o utilitário *setPorosity*. Ambos foram comentados na [seção 4.3](#).

thermophysicalModels é um conjunto de bibliotecas que implementam os modelos termofísicos para a fase sólida e também os modelos termoquímicos para as fases sólida e gasosa. Elas serão melhor comentadas na [seção 4.8](#). Suas bibliotecas são divididas em cinco grupos:

- *basicSolidThermo* - Bibliotecas para calcular as propriedades do sólido como um mistura de componentes.
- *chemistryModel* - Bibliotecas para o modelo químico da fase gasosa, incluindo os métodos de solução da parte química, como solucionadores de EDOs.
- *radiationModels* - Bibliotecas para os modelos de radiação.
- *solid* - Conjunto de bibliotecas para o cálculo das propriedades de cada componente do sólido, como massa específica, condutividade térmica, difusividade térmica, calor específico, entalpias total, sensível e de formação, e propriedades de radiação. Também possui bibliotecas para a construção das reações químicas envolvendo o sólido e cálculo das taxas dessas reações. Por fim, possui bibliotecas para tratar os parâmetros usados na transferência de calor entre sólido e gás.
- *solidChemistryModel* - Bibliotecas para o modelo químico da fase sólida.

O último elemento da *biomassGasificationMedia* é a biblioteca *pyrolysisModels*, composta pela classe base *heterogeneousPyrolysisModel* e pela classe derivada *volPyrolysis*. Ela é responsável por criar os campos de propriedades do sólido, como os campos de frações mássicas dos componentes sólidos, e por integrar os submodelos físicos da fase sólida, como cinética, radiação e transferência de calor. Ela calcula os termos fontes devidos ao sólido para as equações de conservação, e soluciona as equações de conservação para a fase sólida (conservação das espécies e da energia) e a variação da porosidade. Informações mais detalhadas sobre a biblioteca *pyrolysisModels* serão fornecidas na [seção 4.9](#).

4.8 Modelos Termofísicos e Termoquímicos

Nesta seção serão tratados os modelos termofísicos e termoquímicos utilizados pelo *biomassGasificationFoam*. Uma boa parte desses modelos estão implementados nas bibliotecas pertencentes ao *thermophysicalModels*, da *biomassGasificationMedia*. Outros modelos são aproveitados da distribuição padrão do OpenFOAM, contidos no diretório `/src` do local de instalação. Se ele foi instalado no diretório `/opt`, estarão em `/opt/OpenFOAM-2.1.1/src`, que é representado pela variável de ambiente `$LIB_SRC`, daí a necessidade de se definir essa variável, como comentado na [seção 4.2](#).

4.8.1 Cinética Química e Estequiometria

4.8.1.1 Fase Sólida

As reações químicas envolvendo a fase sólida podem ser tanto reações de decomposição do sólido, onde há apenas um sólido como substrato se decompondo em gases e sólidos, quanto reações heterogêneas, onde um sólido reage com um gás resultando em diversos produtos. O modo como essas reações na fase sólida são definidas, sua estequiometria, sua cinética química e seus termos fontes de massa e energia são implementados em várias bibliotecas da `biomassGasificationMedia`.

A taxa de reação, k , é muitas vezes implementada como uma Equação de Arrhenius (subseção 2.2.2.1, Equação 2.17). No `biomassGasificationFoam` existem dois modelos possíveis para essa taxa de reação:

- *irreversibleSolidArrheniusHeterogeneousReaction* - Taxa de reação modelada como uma Equação de Arrhenius:

$$k = \begin{cases} A \exp\left(-\frac{T_a}{T}\right), & \text{se } T \geq T_c \\ 0, & \text{se } T < T_c \end{cases} \quad (4.14)$$

Onde A é o fator pré-exponencial, T_c é a temperatura crítica, abaixo da qual não ocorre reação, e T_a é a chamada temperatura de ativação, definida pela razão entre a energia de ativação, E , e a constante universal dos gases, R :

$$T_a = \frac{E}{R} \quad (4.15)$$

- *irreversibleSolidEvaporationHeterogeneousReaction* - Taxa de reação para a evaporação da água:

$$k = \begin{cases} A |T - T_c|^{2/3} \exp\left(-\frac{T_a}{T}\right), & \text{se } T \geq T_c \\ 0, & \text{se } T < T_c \end{cases} \quad (4.16)$$

Tipicamente, o modelo *irreversibleSolidArrheniusHeterogeneousReaction* é usado para reações de pirólise, combustão e gaseificação. Para o processo de secagem podem ser usados tanto o *irreversibleSolidArrheniusHeterogeneousReaction* quanto o *irreversibleSolidEvaporationHeterogeneousReaction*. Outras formas possíveis para a taxa de reação podem ser implementadas, como modelos mais realistas de secagem.

Esses dois modelos estão implementados nas classes *solidArrheniusReactionRate* e *solidEvaporationRate*, contidos no diretório `/biomassGasificationMedia/thermophysicalModels/solid/reaction/reactionRate`. Nessas classes, a Equação 4.14 e a Equação 4.16 são definidas fazendo um *overload* do operador parêntesis. Por exemplo, do arquivo *solidArrheniusReactionRateI.H*:

```

1 inline Foam::scalar Foam::solidArrheniusReactionRate::operator()
2 (
3     const scalar T,
4     const scalar p,
5     const scalarField& c
6 ) const
7 {
8     scalar ak = A_;
9
10    if (T < Tcrit_)
11    {
12        ak *= 0.0;
13    }
14    else
15    {
16        ak *= exp(-Ta_/T);
17    }
18
19    return ak;
20 }

```

Esse operador parêntesis, definido em *solidArrheniusReactionRateI.H* para o cálculo da taxa de reação k , é invocado pela classe *IrreversibleSolidHeterogeneousReaction*, pelo parâmetro de retorno da função $kf(const\ scalar\ T, const\ scalar\ p, const\ scalarField\&\ c)$, como mostrado neste trecho de código do arquivo *IrreversibleSolidHeterogeneousReaction.C*:

```

1 template<class ReactionRate>
2 Foam::scalar Foam::IrreversibleSolidHeterogeneousReaction<ReactionRate>::kf
3 (
4     const scalar T,
5     const scalar p,
6     const scalarField& c
7 ) const
8 {
9     return k_(T, p, c);
10 }

```

Se o usuário tiver interesse em implementar novos modelos para a taxa de reação k pode utilizar isso como exemplo e adaptar o seu modelo.

A construção das equações envolvendo a fase sólida é feita pela classe *solidHeterogeneousReaction*. A classe *IrreversibleSolidHeterogeneousReaction*, derivada da anterior, é responsável pela obtenção dos parâmetros cinéticos de cada reação, como a taxa k , a ordem da reação em relação ao sólido e o calor de reação, se este for definido pelo usuário. Ambas as classes estão presentes no diretório */biomassGasificationMedia/thermophysicalModels/solid/reaction/reactions*.

A cinética química das reações e seus termos fontes de massa e energia são implementados na classe *ODESolidHeterogeneousChemistryModel*, que estende o modelo químico básico adicionando um pacote termodinâmico e funções de EDOs. Essa classe foi criada baseada na *ODESolidChemistryModel*, fornecida originalmente com o OpenFOAM. Devido ao interesse nos assuntos abordados por ela, a classe *ODESolidHeterogeneousChemistryModel* será uma das mais

estudadas neste trabalho.

A taxa de reação em massa por unidade de volume para a reação r envolvendo o sólido s como substrato é calculada como:

$$\Omega_r = \rho^S \left(Y_s^S\right)^{n_r} \left(Y^G\right)^{\delta^G} k_r \quad (4.17)$$

Onde ρ^S é a massa específica do sólido, Y_s^S é a fração mássica do sólido s , n_r é a ordem da reação em relação ao sólido, Y^G é a fração mássica do gás envolvido em uma reação heterogênea, e k_r é a taxa de reação k para a reação r . Do jeito como está implementado o `biomassGasificationFoam`, o parâmetro δ^G é igual a um se existe um substrato gasoso, ou igual a zero se não existe. Isso implica que não é possível considerar na cinética química uma ordem da reação em relação ao gás. Essa ordem é sempre unitária quando o gás está presente na reação.

Os termos fontes de massa, R , em $\text{kg m}^{-3} \text{s}^{-1}$, para os produtos e reagentes de uma reação r são calculados baseados na taxa de reação Ω_r . Para os sólidos reagentes s , ela é:

$$R_{s,\text{reagente}}^S = -\Omega_r \frac{\nu_s}{\sum_s \nu_{s,\text{reagente}}} \quad (4.18)$$

Onde ν_s é o coeficiente estequiométrico do sólido reagente s , e o termo $\sum_s \nu_{s,\text{reagente}}$ é a soma dos coeficientes estequiométricos dos sólidos reagentes, ou seja, que estejam como substrato na reação. Como as reações químicas nos processos de conversão termoquímica apresentam apenas um sólido como substrato, temos que $\sum_s \nu_{s,\text{reagente}} = \nu_s$. Portanto, a taxa de reação R se torna:

$$R_{s,\text{reagente}}^S = -\Omega_r \quad (4.19)$$

Para um possível produto sólido k , ela é:

$$R_{k,\text{produto}}^S = \Omega_r \frac{\nu_k}{\sum_s \nu_{s,\text{reagente}}} \quad (4.20)$$

Se a reação tiver apenas um produto que esteja no estado sólido, seu coeficiente estequiométrico corresponderá à soma dos coeficientes estequiométricos de todos os produtos sólidos, isto é, $\sum_k \nu_{k,\text{produto}} = \nu_k$. Para esse caso específico, a taxa de reação pode ser entendida como:

$$R_{k,\text{produto}}^S = \xi_r \Omega_r \quad (4.21)$$

Onde ξ_r é a razão mássica entre produtos e substratos sólidos, definido por:

$$\xi_r = \frac{\sum_k \nu_{k,produto}}{\sum_s \nu_{s,reagente}} \quad (4.22)$$

Considerando que, para uma reação r , os gases i sejam produtos e o gás j seja um substrato, a taxa de reação para os produtos i será:

$$R_{i,produto}^G = \Omega_r (1 - \xi_r) \frac{\nu_i W_i}{\sum_i \nu_i W_i - \nu_j W_j} \quad (4.23)$$

Onde W é o peso molecular do gás.

A taxa de reação para o substrato gasoso j será:

$$R_{j,reagente}^G = -\Omega_r (1 - \xi_r) \frac{\nu_j W_j}{\sum_i \nu_i W_i - \nu_j W_j} \quad (4.24)$$

A classe *ODESolidHeterogeneousChemistryModel* possui duas funções de nome *omega*, que possuem parâmetros de entrada diferentes. A primeira delas que aparece no código, *omega(c, T, p, updateC0)*, realiza o cálculo das taxas de reação R , descritas nas Equações 4.18, 4.20, 4.23 e 4.24.

A segunda dessas funções, *omega(R, c, T, p, pf, cf, lRef, pr, cr, rRef)*, é responsável pelo cálculo da taxa de reação Ω_r (Equação 4.17).

Em resumo, o conjunto composto pelo coeficiente para a velocidade de reação k (Equação 4.14 ou Equação 4.16), pela taxa de reação Ω_r (Equação 4.17) e pelas taxas de consumo ou produção das espécies i , R_i (Equações 4.18, 4.20, 4.23 e 4.24) descreve o modelo cinético implementado no *biomassGasificationFoam*. Na literatura, existem diferentes modelos para esses parâmetros, e a possibilidade de o usuário necessitar de um desses modelos diferentes do implementado é bem real. Um exemplo seria se a taxa de reação fosse dependente da pressão parcial do gás reagente, ou se ela fosse descrita por um modelo cinético de Langmuir-Hinshelwood (citados na subseção 2.2.3), ou se a velocidade de reação k fosse modelada por uma Equação de Arrhenius Modificada Equação 4.25. Nesses casos, o usuário deverá alterar a implementação do código nessas funções e classes citadas acima.

Para o usuário introduzir reações envolvendo a fase sólida, deve declará-las no sub-dicionário *reactions* do dicionário *chemistryProperties*, presente no diretório *constant* do caso estudado. Como primeiro exemplo, tem-se a reação de pirólise da hemicelulose:

```
irreversibleSolidArrheniusHeterogeneousReaction
hem = 1.57 CH4 + 8.75 H2 + 5.37 CO + 9.72 CO2 + 0.37903 char
(5.8e13 1.935e4 400 1 1)
```

A primeira linha determina o modelo escolhido para a taxa de reação k . A segunda linha descreve uma reação de pirólise da hemicelulose, onde 1 kg de hemicelulose se decompõe em 0,37903 kg de carvão (*char*) e gases voláteis (CH_4 , H_2 , CO e CO_2). A estequiometria dos gases foi calculada experimentalmente e deve ser dada em mols de gás por quilograma de sólido decomposto. A terceira linha contém os parâmetros da reação, na seguinte ordem: fator pré-exponencial (A), temperatura de ativação (T_a), temperatura crítica (T_c), ordem da reação com relação ao sólido (n_r) e calor da reação (h_r). O último parâmetro, calor de reação h_r , é opcional e será comentado na [subseção 4.8.2](#).

Um segundo exemplo ilustra como deve ser introduzida uma reação heterogênea de gaseificação do carvão:

```
irreversibleSolidArrheniusHeterogeneousReaction
char + CO2 = 2CO + 0.1 ash
(1.7e7 324 800 2 1)
```

Essa reação é interpretada como uma reação heterogênea na qual 1 kg de carvão (*char*) reage com CO_2 gasoso, produzindo 0,1 kg de cinza (*ash*) e dois moles de CO por mol de CO_2 consumido como substrato. A quantidade líquida de massa liberada para a fase gasosa é igual à perda de massa da fase sólida, o que torna possível calcular os termos fontes para todas as espécies gasosas envolvidas na reação.

O modo como a taxa de consumo ou produção das espécies sólidas e gasosas, ou seja, a taxa de reação R (Equações 4.18, 4.20, 4.23 e 4.24) foi implementada no `biomassGasificationFoam`, na classe `ODESolidHeterogeneousChemistryModel`, determina que, na declaração das reações, os coeficientes estequiométricos dos sólidos sejam dados em quilograma e os dos gases em moles. Isso é válido para reações com substrato gasoso ou para reações com apenas um sólido como substrato, mas com mais de um gás como produto. A única exceção a essa regra ocorre para reações sem substrato gasoso e com apenas um gás como produto, como por exemplo, a reação de descarbonatação do carbonato de cálcio, produzindo óxido de cálcio e CO_2 :

```
CaCO3 = 0.56 CaO + 0.44 CO2
```

Tanto o carbonato de cálcio quanto o óxido de cálcio são sólidos. Para essa reação, no cálculo da taxa de formação do gás produzido $R_{\text{CO}_2}^G$ (Equação 4.23), o coeficiente estequiométrico do CO_2 se torna irrelevante. Essa reação indica que, para 1 kg de CaCO_3 consumido, formam-se 0,56 kg de CaO e, conseqüentemente, 0,44 kg de CO_2 .

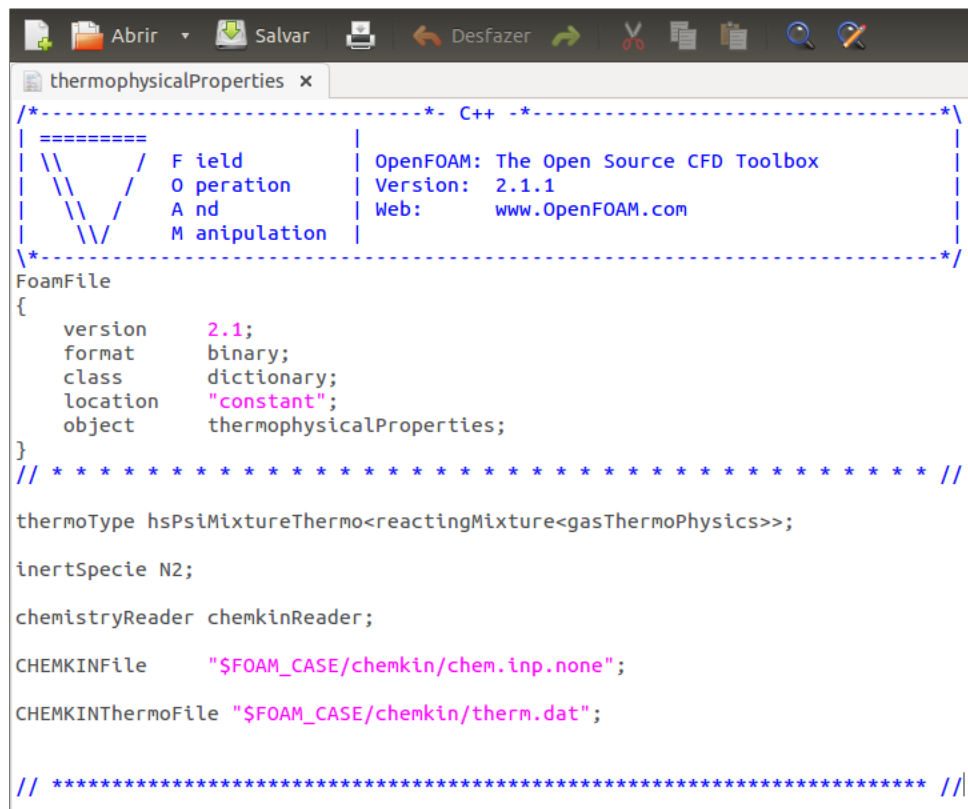
Esses detalhes em relação ao modo de entrada das reações devem ser levados em consideração pelo usuário, de forma a realizá-la corretamente.

4.8.1.2 Fase Gasosa

A implementação das reações homogêneas em fase gasosa, suas cinéticas, termos fontes e modo de defini-las foi herdada do *solver reactingFoam*. Existem duas maneiras de fazer a leitura dessas reações e das propriedades físicas das espécies gasosas, que envolve a escolha entre duas classes que fazem a leitura dessas informações. Essas classes são a *chemkinReader* e a *foamChemistryReader*, derivadas da classe base *chemistryReader*. Elas são encontradas na biblioteca padrão do OpenFOAM, no diretório `$WM_PROJECT_DIR/src/thermophysicalModels/reactionThermo/chemistryReaders`.

O usuário deve escolher qual classe quer utilizar no sub-dicionário *chemistryReader* do dicionário *thermophysicalProperties*, presente no diretório *constant* do caso estudado. Com a escolha, devem ser incluídos dois dicionários que a determinada classe usará para fazer a leitura das informações. Um define as reações químicas e o outro fornece as propriedades físicas dos gases.

Para o caso no qual seja utilizado o *chemkinReader*, as entradas do dicionário *thermophysicalProperties* devem ser conforme o exemplo da Figura 40, que pertence ao caso teste *TGA_test*, fornecido com o *biomassGasificationFoam*.



```

/*----- C++ -----*/
=====
Field      | OpenFOAM: The Open Source CFD Toolbox
Operation  | Version:  2.1.1
And        | Web:      www.OpenFOAM.com
Manipulation
/*-----*/
FoamFile
{
  version      2.1;
  format       binary;
  class        dictionary;
  location     "constant";
  object       thermophysicalProperties;
}
// *****

thermoType hsPsiMixtureThermo<reactingMixture<gasThermoPhysics>>;

inertSpecie N2;

chemistryReader chemkinReader;

CHEMKINFile     "$FOAM_CASE/chemkin/chem.inp.none";

CHEMKINThermoFile "$FOAM_CASE/chemkin/therm.dat";

// *****

```

Figura 40 – Arquivo *thermophysicalProperties* do caso *TGA_test*

Fonte: Caso *TGA_test* do *biomassGasificationFoam*

No exemplo, ambos os dicionários estão em um diretório chamado *chemkin*, dentro do diretório raiz do caso. O dicionário que faz a declaração das reações químicas é o *chem.inp.none*,

definido no sub-dicionário *CHEMKINFile*. O dicionário que fornece as propriedades físicas das espécies gasosas é o *therm.dat*, definido no sub-dicionário *CHEMKINThermoFile*.

A classe *chemkinReader* utiliza o formato CHEMKIN para os dados de reações e propriedades físicas. O arquivo descrevendo as reações químicas deve conter a declaração das reações com os seus parâmetros cinéticos, e também os elementos e espécies químicas envolvidas. A Figura 41 mostra um exemplo para o arquivo *chem.inp.frag*, fornecido com o caso teste *oneCellTest*.

```

ELEMENTS
H O C N
END
SPECIE
N2 O2 CH4 CH3 CH3O CH2O OH H2O2 H2O CO CO2 HCO H2O2
END
REACTIONS
CH4 + O2 = CH3 + H2O 7.900E+13 0.00 56000. !154
CH4 + OH = CH3 + H2O 1.600E+06 2.10 2460. !156
CH3 + H2O2 = CH3O + OH 4.300E+13 0.00 0. !75
CH3O + OH = CH2O + H2O 1.000E+13 0.00 0. !71
CH3O + O2 = CH2O + H2O2 1.200E+11 0.00 2600. !73
CH2O + O2 = HCO + H2O2 6.200E+13 0.00 39000. !139
CH2O + OH = HCO + H2O 2.430E+10 1.18 -447. !142
H2O2 + H2O2 = H2O2 + O2 2.000E+12 0.00 0. !122
H2O2 + OH = H2O + H2O2 1.000E+13 0.00 1800. !125
HCO + O2 = H2O + CO 3.300E+13 -0.40 0. !151
HCO + HCO = CH2O + CO 3.010E+13 0.00 0. !146
HCO + OH = H2O + CO 1.000E+14 0.00 0. !147
CH4 + H2O2 = CH3 + H2O2 1.000E+13 0.00 18700. !158
CH3 + CH2O = CH4 + HCO 5.500E+03 2.80 6000. !160
CH3 + HCO = CH4 + CO 1.200E+14 0.00 0. !161
CH2O + H2O2 = HCO + H2O2 3.000E+12 0.00 8000. !143
CH3O + CO = CH3 + CO2 1.570E+14 0.00 11800. !69
END

```

Figura 41 – Arquivo *chem.inp.frag* do caso *oneCellTest*

Fonte: Caso *oneCellTest* do *biomassGasificationFoam*

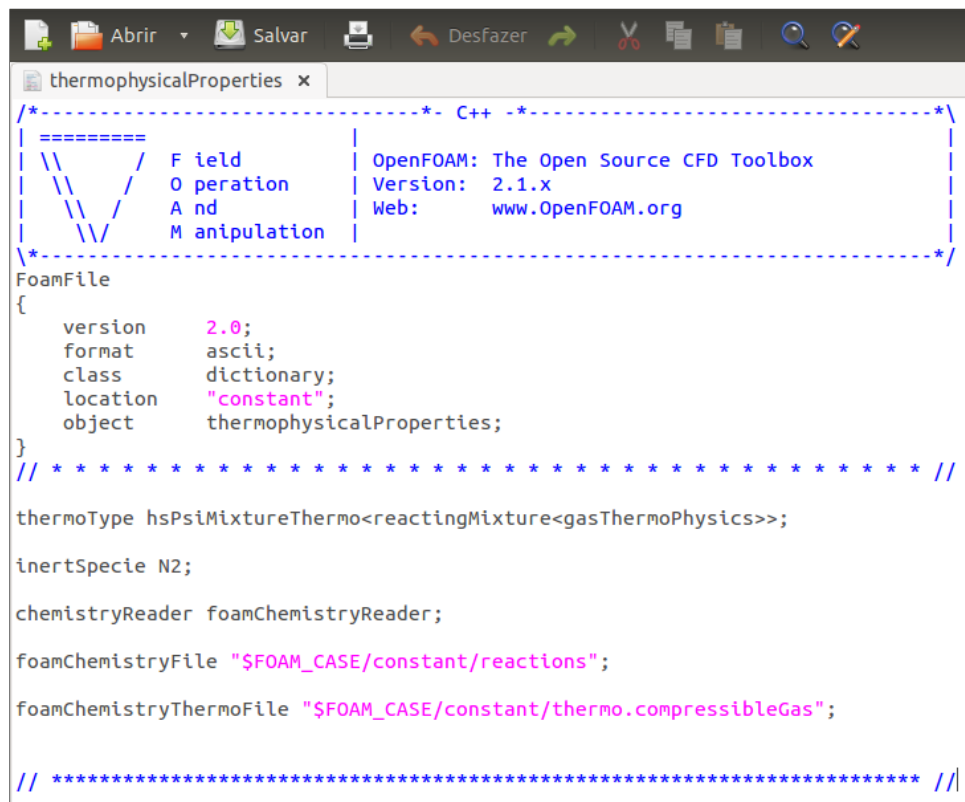
O CHEMKIN utiliza a Equação de Arrhenius Modificada para a taxa de reação k . Para uma determinada reação i , ela será:

$$k_i = A_i T^{\beta_i} \exp\left(-\frac{E_i}{RT}\right) \quad (4.25)$$

A primeira coluna dos parâmetros cinéticos da Figura 41, destacada em verde, representa o coeficiente pré-exponencial A . A segunda coluna, em azul, é o expoente β , e a terceira coluna, em vermelho, é a energia de ativação E .

Para o caso no qual seja utilizado a classe *foamChemistryReader* para a leitura das reações químicas e propriedades físicas das espécies gasosas, as entradas do dicionário *thermophysicalProperties* devem ser conforme o exemplo da Figura 42, que pertence ao caso tutorial do *reactingFoam*.

Nesse exemplo, o dois dicionários estão no diretório *constant* do caso. O dicionário que faz a declaração das reações químicas é o *reactions*, definido no sub-dicionário *foamChemistryFile*. O



```

/*-----* C++ *-----*/
|=====|
| \ \ \ \ | F i e l d | OpenFOAM: The Open Source CFD Toolbox
| \ \ \ \ | O p e r a t i o n | Version: 2.1.x
| \ \ \ \ | A n d | Web: www.OpenFOAM.org
| \ \ \ \ | M a n i p u l a t i o n |
|=====|
FoamFile
{
  version      2.0;
  format       ascii;
  class        dictionary;
  location     "constant";
  object       thermophysicalProperties;
}
// *****

thermoType hsPsiMixtureThermo<reactingMixture<gasThermoPhysics>>;

inertSpecie N2;

chemistryReader foamChemistryReader;

foamChemistryFile "$FOAM_CASE/constant/reactions";

foamChemistryThermoFile "$FOAM_CASE/constant/thermo.compressibleGas";

// *****

```

Figura 42 – Arquivo *thermophysicalProperties* do caso tutorial do *reactingFoam*

Fonte: Caso tutorial do *reactingFoam*

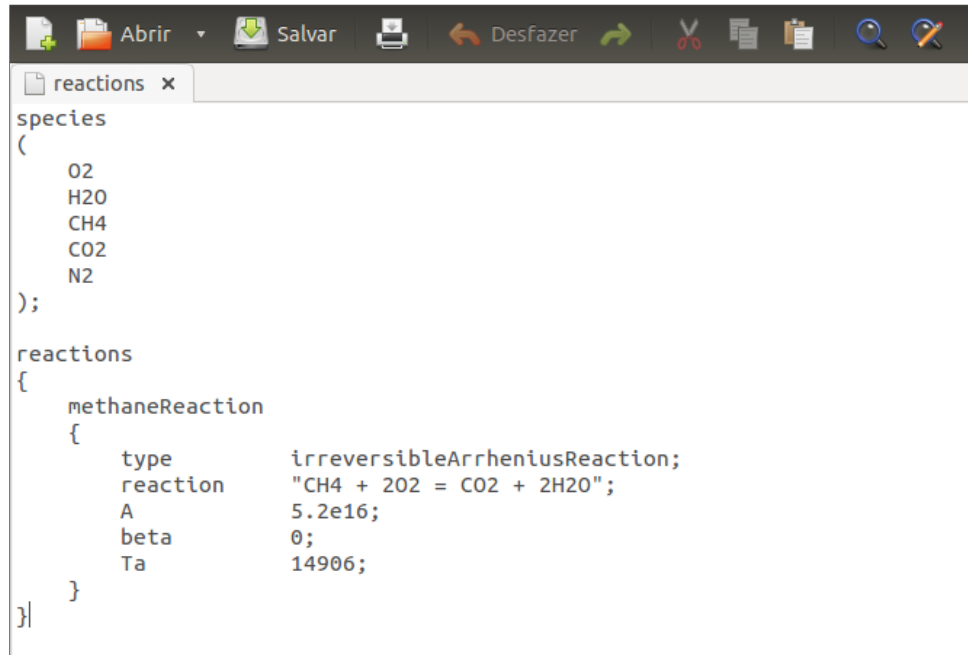
dicionário que fornece as propriedades físicas das espécies gasosas é o *thermo.compressibleGas*, definido no sub-dicionário *foamChemistryThermoFile*.

O arquivo descrevendo as reações químicas deve conter a declaração das reações com os seus parâmetros cinéticos, e também as espécies químicas envolvidas. A [Figura 43](#) mostra um exemplo para o arquivo *reactions*, fornecido com o caso tutorial do *reactingFoam*.

No sub-dicionário *reactions*, juntamente com a reação química, deve ser definido o tipo do modelo para a taxa de reação k , e seus respectivos parâmetros cinéticos. A distribuição padrão do OpenFOAM fornece alguns modelos para essa taxa de reação, que podem ser encontrados no diretório $\$WM_PROJECT_DIR/src/thermophysicalModels/specie/reaction/reactionRate$.

Por fim, mais informações sobre o formato CHEMKIN podem ser obtidas no trabalho de [Lundström \(2008\)](#).

Para as reações homogêneas em fase gasosa, a cinética química, como taxa de reação e taxa de consumo e produção de cada espécie gasosa, ω_i^G , é implementada na classe *ODEChemistryModel*, presente no diretório */biomassGasificationMedia/thermophysicalModels/chemistryModel/chemistryModel/*.



```

species
(
  O2
  H2O
  CH4
  CO2
  N2
);

reactions
{
  methaneReaction
  {
    type      irreversibleArrheniusReaction;
    reaction  "CH4 + 2O2 = CO2 + 2H2O";
    A         5.2e16;
    beta      0;
    Ta        14906;
  }
}

```

Figura 43 – Arquivo *reactions* do caso tutorial do *reactingFoam*Fonte: Caso tutorial do *reactingFoam*

4.8.2 Calor de Reação

No *biomassGasificationFoam* existem duas maneiras de se obter o calor de reação para os processos envolvendo a fase sólida: baseado nas entalpias de formação dos produtos e reagentes; ou fornecendo independentemente o valor para o calor de cada reação.

A primeira abordagem, pelas entalpias de formação, é típica para reações em fase gasosa e já está implementada no *solver reactingFoam*. Nessa abordagem, a entalpia de formação, h_f , para cada reagente e cada produto deve ser definida. O calor de reação será dado por:

$$H_r = \sum_s R_s^{tot} h_{f_s} + \sum_p R_p^{tot} h_{f_p} \quad (4.26)$$

Onde o índice s refere-se aos substratos da reação (reagentes), o índice p refere-se aos produtos, e R^{tot} é a taxa de consumo ou geração de massa total para cada espécie devido à todas as reações químicas r :

$$R_i^{tot} = \sum_r R_{i,r} \quad (4.27)$$

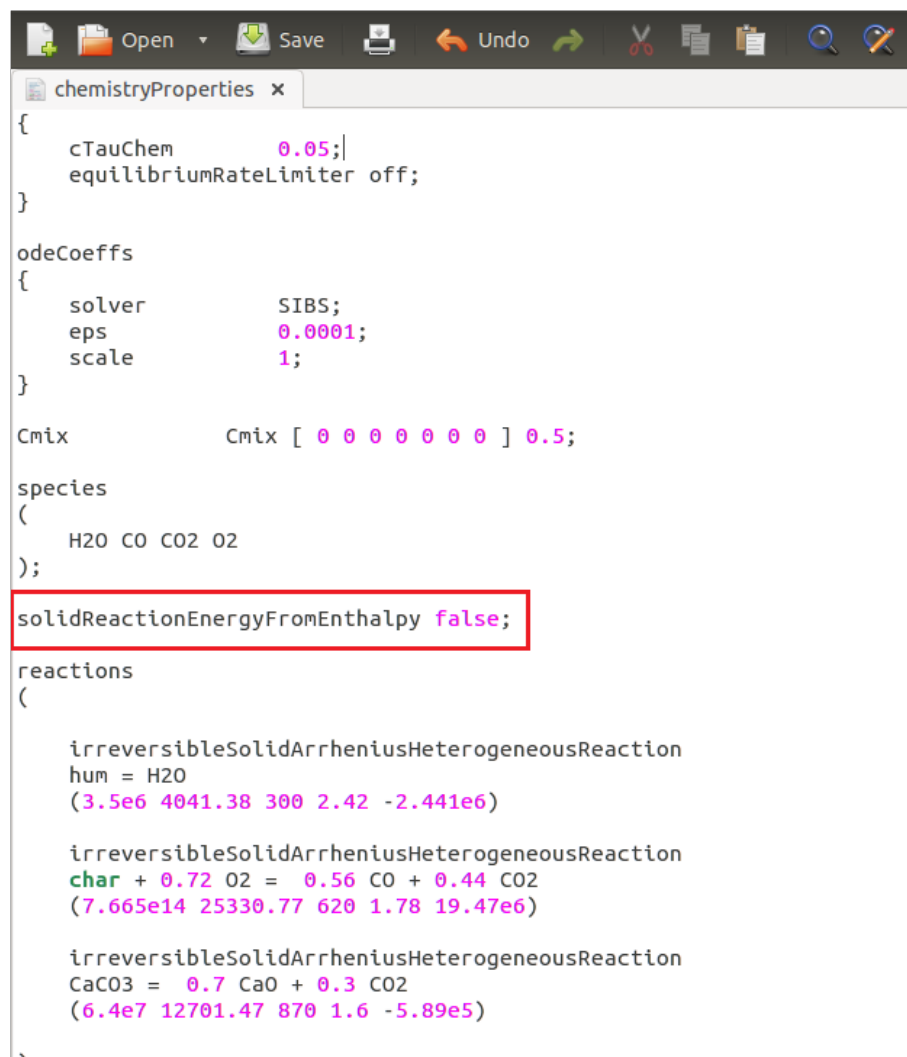
Na segunda abordagem, o usuário deve fornecer independentemente o valor para o calor de cada reação. A implementação dessa abordagem é motivada pelo fato de que as entalpias de formação dos componentes sólidos, como a lignina, são difíceis de serem determinadas, porém o

calor de reação envolvendo esses componentes pode ser obtido experimentalmente. Considerando para cada reação r , cuja taxa de reação é Ω_r e o calor de reação é h_r , o efeito energético total é dado por:

$$H_r = \sum_r h_r \Omega_r \quad (4.28)$$

As duas abordagens descritas para o calor de reação são implementadas na classe *ODESolidHeterogeneousChemistryModel*, onde o cálculo de H_r é feito pela função *Sh()*.

A escolha entre as duas abordagens é feita com o uso do sub-dicionário *solidReactionEnergyFromEnthalpy*, do dicionário *chemistryProperties*, como mostra o exemplo da [Figura 44](#). Se o usuário inserir o valor **true**, será utilizada a primeira abordagem, pela entalpias de formação. Se ele inserir o valor **false**, será utilizada a segunda abordagem, onde ele terá que fornecer o valor do calor de reação.



```
chemistryProperties {
  cTauChem 0.05;
  equilibriumRateLimiter off;
}

odeCoeffs {
  solver SIBS;
  eps 0.0001;
  scale 1;
}

Cmix Cmix [ 0 0 0 0 0 0 ] 0.5;

species (
  H2O CO CO2 O2
);

solidReactionEnergyFromEnthalpy false;

reactions (
  irreversibleSolidArrheniusHeterogeneousReaction
  hum = H2O
  (3.5e6 4041.38 300 2.42 -2.441e6)

  irreversibleSolidArrheniusHeterogeneousReaction
  char + 0.72 O2 = 0.56 CO + 0.44 CO2
  (7.665e14 25330.77 620 1.78 19.47e6)

  irreversibleSolidArrheniusHeterogeneousReaction
  CaCO3 = 0.7 CaO + 0.3 CO2
  (6.4e7 12701.47 870 1.6 -5.89e5)
)
```

Figura 44 – Trecho do arquivo *chemistryProperties*, mostrando a variável *solidReactionEnergyFromEnthalpy*

Na classe *ODESolidHeterogeneousChemistryModel*, o valor padrão definido para a variável *solidReactionEnergyFromEnthalpy* é **true**, ou seja, se o usuário não alterar esse valor, a abordagem com as entalpias de formação será utilizada. Isso é o que acontece para os dois casos testes fornecidos com o *biomassGasificationFoam*, que servem como exemplo. Eles não apresentam o sub-dicionário *solidReactionEnergyFromEnthalpy* no arquivo *chemistryProperties*, e portanto, usam a abordagem das entalpias de formação. Esse detalhe pode confundir o usuário, criando dúvidas sobre como utilizar a segunda abordagem, do calor de reação.

Como comentado na [subseção 4.8.1.1](#), a declaração de uma reação química envolvendo a fase sólida é feita no dicionário *chemistryProperties*. Um exemplo é:

```
irreversibleSolidArrheniusHeterogeneousReaction
hem = 1.57 CH4 + 8.75 H2 + 5.37 CO + 9.72 CO2 + 0.37903 char
(5.8e13 1.935e4 400 1 1)
```

A última entrada da terceira linha é o calor de reação h_r . Ele deve ser inserido se o usuário pretende utilizar a segunda abordagem para o calor de reação. Um detalhe importante é que o calor de reação h_r é definido por unidade de massa de voláteis, isto é, por unidade de massa perdida pela fase sólida. Isso se torna relevante se existe algum produto sólido na reação.

O calor de reação para as reações homogêneas em fase gasosa é implementado na classe *ODEChemistryModel*, e segue a abordagem das entalpias de formação:

$$H_r^G = - \sum_i \omega_i^G h_{f_i}^G \quad (4.29)$$

Onde ω_i^G é a taxa de consumo ou produção da espécie i devido às reações homogêneas, e $h_{f_i}^G$ é a sua entalpia de formação. O calor de reação dado pela [Equação 4.29](#) entra como termo fonte na equação de conservação da energia para a fase gasosa ([Equação 4.13](#)).

4.8.3 Transferência de Calor

O modelo para a conservação da energia implementado no *biomassGasificationFoam* assume a hipótese de não-equilíbrio térmico local, ou seja, que as temperaturas da fase sólida e da fase gasosa em um determinado ponto não são iguais. Essa diferença de temperatura requer que a transferência de calor entre o sólido e o gás seja levada em consideração.

Assume-se que a transferência de calor é proporcional à diferença entre a temperatura da fase gasosa, T^G , e a temperatura da fase sólida, T^S , conforme o termo:

$$\alpha \Sigma (T^G - T^S) \quad (4.30)$$

Onde α é o coeficiente de transferência de calor, em $\text{W}\cdot\text{m}^{-2}\cdot\text{K}^{-1}$, e Σ é a área superficial específica, ou área superficial dos poros por volume de sólido, em m^2/m^3 . Esse termo para a transferência de calor entra como termo fonte nas equações de conservação da energia para a fase sólida (Equação 4.11) e conservação da energia para a fase gasosa (Equação 4.13).

A transferência de calor no interior da amostra de sólido é diferente da transferência de calor em sua fronteira. Para representar essas duas situações, são introduzidos dois parâmetros para o coeficiente de transferência de calor. O parâmetro *HTC* (*Heat Transfer Coefficient*) denota α para as células computacionais no interior da amostra de sólido. Já o parâmetro *borderHTC* define α para as células computacionais na fronteira da amostra de sólido.

Existem duas opções para se calcular a área superficial específica dos poros, Σ . A primeira é com o modelo *constHTC*, que assume uma área superficial específica constante, $\Sigma = \text{const}$.

A segunda opção é com o modelo *cylinderHTC*, que considera a evolução do campo de porosidade. Ele assume que o meio poroso com fração de vazio inicial γ_0 é composto por uma rede regular de tubos paralelos com raio inicial a_0 . Baseado nessa hipótese, o número de tubos paralelos N por unidade de volume pode ser calculado como (KWIATKOWSKI et al., 2013):

$$N = \frac{\gamma_0 V}{\pi a_0^2 V^{1/3}} \quad (4.31)$$

Considerando N constante, os poros crescem devido apenas ao aumento do raio do tubo a . A variação do raio a , a partir de um valor inicial a_0 , pode ser calculada da razão entre o volume ocupado pelo gás em um momento qualquer, V_g , e o volume ocupado pelo gás no momento inicial, $V_{g,0}$.

$$\frac{V_g}{V_{g,0}} = \frac{\gamma V_T}{\gamma_0 V_T} = \frac{N \pi a^2 L}{N \pi a_0^2 L} \rightarrow \frac{\gamma}{\gamma_0} = \frac{a^2}{a_0^2} \rightarrow a = a_0 \sqrt{\frac{\gamma}{\gamma_0}} \quad (4.32)$$

Onde N é o número de tubos paralelos e L é o comprimento do tubo. A área superficial dos poros por unidade de volume pode ser dada por (KWIATKOWSKI et al., 2013):

$$\Sigma = \frac{N 2 \pi a V^{1/3}}{V} \quad (4.33)$$

Substituindo a Equação 4.31 na Equação 4.33, e considerando a variação do raio a pela Equação 4.32, temos:

$$\Sigma = 2 \frac{a}{a_0^2} \gamma_0 = \frac{2}{a_0^2} a_0 \left(\frac{\gamma}{\gamma_0} \right)^{1/2} \gamma_0 \quad (4.34)$$

Conclui-se então que a área superficial específica pode ser calculada por:

$$\Sigma = \frac{2}{a_0} (\gamma)^{1/2} (\gamma_0)^{1/2} \quad (4.35)$$

A equação para a área superficial específica acima (Equação 4.35) está em conformidade com a que seria obtida seguindo o modelo exposto por Cunha (2010). No entanto, ela difere ligeiramente da apresentada no trabalho de Kwiatkowski et al. (2013). Apesar disso, no código do `biomassGasificationFoam` ela foi implementada corretamente, conforme a Equação 4.35, na classe `cylinder`.

Como comentado na seção 4.3, a fração de vazio inicial γ_0 presente na Equação 4.35 é calculada a partir do campo `porosityF0`, que deve estar presente no diretório “0” do caso em estudo. Esse arquivo fornece o valor do campo de porosidade inicial, e não é alterado durante a simulação.

A implementação dos dois modelos de transferência de calor é feita nas classes `const` e `cylinder`, derivadas da classe base `heatTransferModel`. Elas podem ser encontradas no diretório `/biomassGasificationMedia/thermophysicalModels/solid/heatTransfer`. Nessas classes são lidos os dados como coeficientes de transferência de calor e raio inicial. Também é calculado o produto entre o coeficiente α (tanto para o `HTC` quanto para o `borderHTC`) e a área superficial específica Σ .

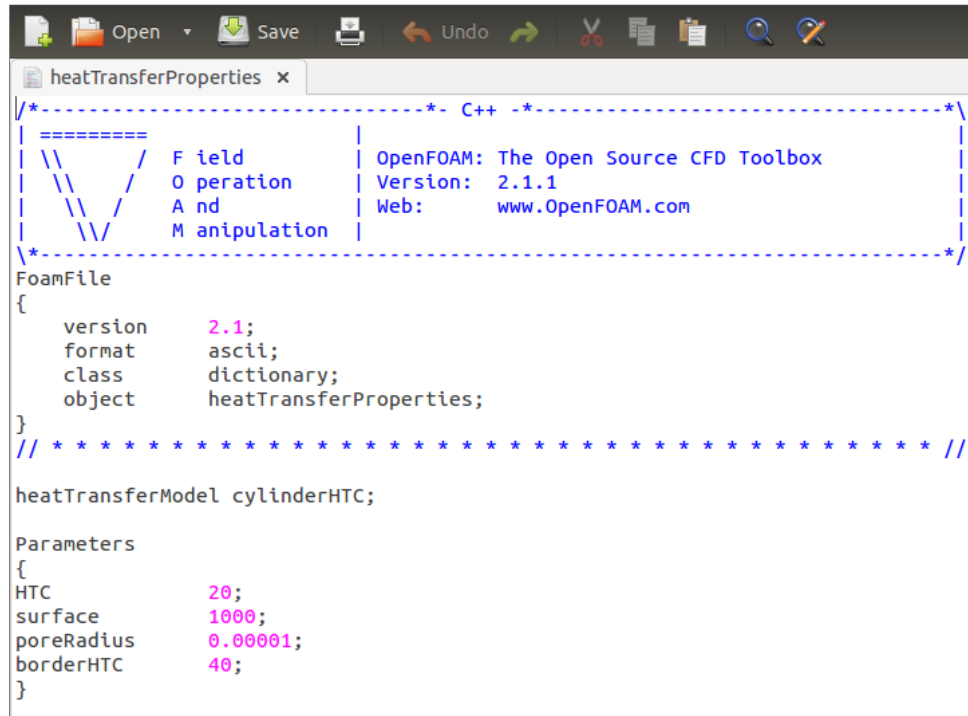
O cálculo do termo fonte devido à transferência de calor (Equação 4.30) é feito pela classe `volPyrolysis`, na função `heatTransfer()`.

No diretório `constant` do caso deve estar presente um dicionário chamado `heatTransferProperties`, onde deve-se fornecer os parâmetros para o modelo de transferência de calor. Um exemplo é mostrado na Figura 45. No primeiro sub-dicionário é onde se escolhe o modelo, `cylinderHTC` ou `constHTC`. O parâmetro `surface` é a área superficial constante Σ para o modelo `constHTC`. O parâmetro `poreRadius` é o raio inicial a_0 para o modelo `cylinderHTC`.

O `biomassGasificationFoam` assume que, durante a conversão térmica, os substratos sólidos estão à temperatura do meio poroso, enquanto os produtos gasosos estão à temperatura do gás dentro do meio poroso. Assim, para as reações heterogêneas com substrato sólido, uma energia Γ é necessária para aquecer ou resfriar os produtos gasosos para a temperatura do gás.

$$\Gamma = \left(T^G C_{p(T^G)}^G - T^S C_{p(T^S)}^G \right) \sum_i R_i^G \quad (4.36)$$

Onde $C_{p(T^G)}^G$ é o calor específico do gás na temperatura do gás e $C_{p(T^S)}^G$ é o calor específico do gás na temperatura do sólido. A energia Γ entra como termo fonte na equação de conservação da energia do gás (Equação 4.13). Esse efeito é independente do efeito de transferência de calor entre as fases sólida e gasosa, citado anteriormente.



```

/*-----* C++ *-----*/
|=====|
| \ \ \ \ | F i e l d | | OpenFOAM: The Open Source CFD Toolbox
| \ \ \ \ | O p e r a t i o n | | Version: 2.1.1
| \ \ \ \ | A n d | | Web: www.OpenFOAM.com
| \ \ \ \ | M a n i p u l a t i o n | |
|-----*-----*/
FoamFile
{
    version      2.1;
    format       ascii;
    class        dictionary;
    object       heatTransferProperties;
}
// *****

heatTransferModel cylinderHTC;

Parameters
{
    HTC          20;
    surface      1000;
    poreRadius   0.00001;
    borderHTC    40;
}

```

Figura 45 – Arquivo *heatTransferProperties* do caso *TGA_test*Fonte: Caso *TGA_test* do *biomassGasificationFoam*

4.8.4 Radiação

No *biomassGasificationFoam*, a influência da radiação no sólido é modelada de modo simplificado. Assume-se que a radiação, proveniente das paredes do domínio, é apenas absorvida pela superfície do sólido, e não é emitida por ele. Existem três modelos implementados, derivados da classe base *heterogeneousRadiationModel*, e que se encontram no diretório */biomassGasificationMedia/thermophysicalModels/radiationModels/radiationModel*. O modelo que se deseje utilizar deve ser escolhido no dicionário *radiationProperties*, localizado no diretório *constant* do caso, com uma das seguintes entradas:

- *none* - Escolhe o modelo *heterogeneousNoRadiation*;
- *heterogeneousMeanTemp* - Escolhe o modelo *heterogeneousMeanTemp*;
- *heterogeneousP1* - Escolhe o modelo *heterogeneousP1*.

O primeiro modelo, *heterogeneousNoRadiation*, não considera radiação na simulação. O modelo *heterogeneousMeanTemp* assume que a espessura ótica dos gases dentro do reator é pequena e que os gases não interferem na radiação. Nesse caso, a energia é transmitida diretamente das paredes aquecidas para a superfície do sólido. Para se determinar a área da superfície do meio poroso, são adotadas algumas hipóteses. A primeira é que, em média, toda célula é cúbica, e sua superfície é dada por $V^{\frac{2}{3}}$, onde V é o volume da célula. Outra hipótese é que um sexto da sua superfície está exposta para lado externo. Assim, a superfície exposta à radiação por célula computacional localizada na fronteira da amostra de sólido será:

$$A = \frac{1}{6} V^{\frac{2}{3}} \quad (4.37)$$

O valor líquido de energia absorvida por unidade de volume será:

$$S^{rad} = \max \left(0, \frac{1}{6} \frac{V^{\frac{2}{3}}}{V} \delta \sigma \left(T_{par}^4 - \epsilon \left(T_{sup}^S \right)^4 \right) \right) \quad (4.38)$$

Onde T_{par} é a temperatura média das paredes, T_{sup}^S é a temperatura da superfície da partícula, σ é a constante de Boltzmann, δ é o coeficiente de absorção ótica da superfície da fase sólida, e ϵ é a razão entre absorção e emissão. Nessa implementação, os dois últimos parâmetros, que descrevem as propriedades óticas da superfície do sólido, são considerados constantes.

No modelo *heterogeneousP1*, a radiação é determinada pelo modelo de radiação P1, disponível no OpenFOAM. Na implementação do `biomassGasificationFoam` não é incluída a emissão de radiação do sólido, o que implica que a massa de sólido não tem influência na densidade de radiação. A densidade de radiação calculada na fase gasosa é absorvida nas fronteiras da fase sólida. O valor líquido de energia absorvida por unidade de volume é calculado por:

$$S_{rad} = \max \left(0, \frac{1}{6} \frac{V^{\frac{2}{3}}}{V} \delta \left(G - \epsilon \sigma \left(T_{sup}^S \right)^4 \right) \right) \quad (4.39)$$

Onde G é o fluxo de energia radiativa.

O trabalho de [Vdovin \(2009\)](#) apresenta informações sobre os modelos de radiação implementados na distribuição padrão do OpenFOAM.

4.8.5 Propriedades Físicas da Fase Sólida

Para completar o modelo de conversão termoquímica, é necessário uma definição apropriada das propriedades termofísicas e químicas das fases sólida e gasosa. Primeiramente, a porosidade inicial do meio deve ser definida. Para incluir uma heterogeneidade do sólido, é possível definir um campo de porosidade não uniforme. Junto com a porosidade, a distribuição do tensor de resistência viscosa deve ser definido. O utilitário `setPorosity` é utilizado para essa tarefa, como comentado na [seção 4.3](#).

As propriedades da fase sólida devem ser fornecidas para cada componente do sólido individualmente, no dicionário `solidThermophysicalProperties`, que deve estar presente no diretório `constant` do caso. Essas propriedades físicas e químicas incluem a massa específica, a condutividade térmica, o calor específico, a entalpia de formação e as propriedades de radiação, como ilustrado na [Figura 46](#), que fornece um trecho do arquivo `solidThermophysicalProperties` com as propriedades para o componente do sólido umidade (*hum*). As propriedades da fase sólida são então uma

```

/*-----* C++ *-----*/
|=====|
| \ \ \ \ | F i e l d | | OpenFOAM: The Open Source CFD Toolbox
| \ \ \ \ | O p e r a t i o n | | Version: 2.1.1
| \ \ \ \ | A n d | | Web: www.OpenFOAM.com
| \ \ \ \ | M a n i p u l a t i o n |
|-----*-----*/
FoamFile
{
    version      2.1;
    format       ascii;
    class        dictionary;
    object       solidThermophysicalProperties;
}
// *****

thermoType solidMixtureThermo<constHeterogeneous>;

solidComponents
(
    char hum ash cel hem lig
);

humCoeffs
{
    transport
    {
        K0      0.0;
        A       0.0;
        K       0.58;
    }
    radiation
    {
        sigmaS   0.0;
        kappa    0.0; //opaque
        emissivity 0.17;
    }
    thermodynamics
    {
        Cp       4200;
        Hf       -15.86e6;
    }
    density
    {
        rho      1000;
    }
};

```

Figura 46 – Trecho do arquivo *solidThermophysicalProperties* do caso *TGA_test*

Fonte: Caso *TGA_test* do *biomassGasificationFoam*

função da sua composição. Conforme ocorre o processo de conversão, as frações mássicas dos seus componentes se alteram e as propriedades do material são ajustadas adequadamente.

Existem três modelos para o cálculo das propriedades massa específica, condutividade térmica e calor específico. O modelo a ser utilizado deve ser selecionado no sub-dicionário *thermoType* do dicionário *solidThermophysicalProperties*:

- *solidMixtureThermo<constHeterogeneous>* - massa específica constante, calor específico constante e condutividade térmica constante;
- *solidMixtureThermo<expoHeterogeneous>* - massa específica constante, calor específico

exponencial e condutividade térmica exponencial;

- *solidMixtureThermo<linearHeterogeneous>* - massa específica constante, calor específico constante e condutividade térmica linear.

Como visto, a massa específica dos componentes sólidos é sempre considerada constante. O calor específico pode ser considerado constante ou exponencial, da forma:

$$C_p(T) = C_0 \left(\frac{T}{T_{ref}} \right)^{n_0} \quad (4.40)$$

A condutividade térmica pode ser considerada constante, ou linear, da forma:

$$k(T) = K_0 + AT \quad (4.41)$$

Ou também exponencial, da forma:

$$k(T) = K_0 \left(\frac{T}{T_{ref}} \right)^{n_0} \quad (4.42)$$

Porém, deve-se ressaltar que as combinações das três propriedades estão definidas nos três modelos citados acima. Por exemplo, se o usuário tivesse a intenção de considerar a massa específica constante, o calor específico exponencial e a condutividade térmica linear, teria que implementar um novo modelo, adicionalmente aos três fornecidos.

A leitura e o cálculo das propriedades para cada componente do sólido é feito por bibliotecas presentes no diretório */biomassGasificationMedia/thermophysicalModels/solid*:

- *rhoType* - Leitura da massa específica.
- *thermo* - Leitura dos parâmetros e cálculo do calor específico, entalpia de formação, entalpia total e entalpia sensível.
- *transport* - Leitura dos parâmetros e cálculo da condutividade térmica e da difusividade térmica.
- *radiation* - Leitura das propriedades de radiação.

A massa específica que deve ser fornecida no dicionário *solidThermophysicalProperties* é a massa específica absoluta $\tilde{\rho}^S$, ou seja, a massa específica do material considerando não haver porosidade e que o volume completo é composto pelo sólido. A partir dela, é possível calcular a massa específica aparente como:

$$\rho^S = (1 - \gamma) \tilde{\rho}^S \quad (4.43)$$

Como dito anteriormente, as propriedades do material sólido são uma função da sua composição. A massa específica, a condutividade térmica e o calor específico da fase sólida são calculados por:

$$\tilde{\rho}^S = \left(\sum_i \frac{Y_i^S}{\tilde{\rho}_i^S} \right)^{-1} \quad (4.44)$$

$$k^S = \left(\sum_i k_i^S \frac{Y_i^S}{\tilde{\rho}_i^S} \right) / \left(\sum_i \frac{Y_i^S}{\tilde{\rho}_i^S} \right) \quad (4.45)$$

$$C_p^S = \sum_i Y_i^S C_{p,i}^S \quad (4.46)$$

Esse cálculo das propriedades termofísicas em função da composição está implementado no conjunto de bibliotecas *basicSolidThermo*.

4.8.6 Propriedades Físicas da Fase Gasosa

A modelagem das propriedades físicas da fase gasosa é herdada do *solver reactingFoam* e são definidas em um complexo conjunto de bibliotecas da distribuição padrão do OpenFOAM. O *biomassGasificationFoam* utiliza o modelo *gasThermoPhysics*, que é composto pelas classes *sutherlandTransport*, *specieThermo*, *janafThermo* e *perfectGas*.

A classe *sutherlandTransport* utiliza o modelo de Sutherland para a propriedade de transporte viscosidade dinâmica:

$$\mu(T) = \frac{A_s \sqrt{T}}{1 + T_s/T} \quad (4.47)$$

Onde A_s e T_s são coeficientes que devem ser fornecidos.

A classe *janafThermo* utiliza as tabelas termodinâmicas JANAF para calcular as propriedades calor específico, entalpia e entropia, em base molar, com os polinômios:

$$\frac{C_p(T)}{R} = a_1 + a_2 T + a_3 T^2 + a_4 T^3 + a_5 T^4 \quad (4.48)$$

$$\frac{h(T)}{R} = a_1 T + \frac{a_2}{2} T^2 + \frac{a_3}{3} T^3 + \frac{a_4}{4} T^4 + \frac{a_5}{5} T^5 + a_6 \quad (4.49)$$

$$\frac{s(T)}{R} = a_1 \ln(T) + a_2 T + \frac{a_3}{2} T^2 + \frac{a_4}{3} T^3 + \frac{a_5}{4} T^4 + a_7 \quad (4.50)$$

Onde R é a constante universal dos gases e a_i são os coeficientes JANAF. A classe *janafThermo* também calcula a entalpia de formação e a entalpia sensível das espécies gasosas. Para a entalpia de formação é utilizado o mesmo polinômio da [Equação 4.49](#) considerando a temperatura padrão de 298,15 K.

A classe *specieThermo* calcula essas propriedades termodinâmicas em base mássica, por meio da divisão pela massa molecular da espécie gasosa, e mais algumas outras, como energia interna e energia livre de Gibbs.

A classe *perfectGas* faz uso da consideração de gás ideal para calcular a massa específica:

$$\rho = \frac{p}{RT} \quad (4.51)$$

Como comentado na [subseção 4.8.1.2](#), existem duas maneiras de fazer a leitura das propriedades físicas e dos parâmetros necessários para calculá-las, que são a utilização da classe *chemkinReader* ou da classe *foamChemistryReader*. Cada classe necessita de dois arquivos adicionais, um que define as reações químicas e outro que fornece as propriedades do gás. Na seção citada acima foram dadas informações sobre essas classes e sobre o arquivo de reações químicas. Esta seção irá se ater ao arquivo de propriedades físicas.

Se o usuário escolheu a *chemkinReader*, o dicionário que fornece as propriedades é o *therm.dat*, presente no diretório *chemkin* do caso. Essa classe utiliza o formato CHEMKIN para os dados termodinâmicos. Esse formato é bem definido, o que significa que o modo como os números e palavras são inseridos é importante para a análise correta do arquivo. O modo mais simples de contornar isso é usar algum arquivo como exemplo, como o fornecido no caso teste *TGA_test*, do *biomassGasificationFoam*, que é um arquivo base com um número enorme de espécies. A [Figura 47](#) mostra um trecho do arquivo *therm.dat* fornecido no caso *TGA_test*, destacando os parâmetros para o cálculo das propriedades das espécies CO₂ e CO.

Para a definição dos parâmetros para cada espécie são necessárias quatro linhas numeradas de 1 a 4. Na primeira dessas linhas devem estar contidos o nome da espécie, sua composição elementar e três valores de temperatura que definem os dois intervalos de temperatura para os quais são válidos os coeficientes. A primeira temperatura é o limite inferior, T_{low} . A segunda é o limite superior, T_{high} . A terceira é a temperatura que divide os dois intervalos, T_{common} . O intervalo inferior de temperaturas vai de T_{low} a T_{common} , e o intervalo superior vai de T_{common} a T_{high} . A nomenclatura aqui adotada teve como objetivo ser equivalente à presente nas bibliotecas e dicionários do OpenFOAM e assim evitar possíveis confusões. As outras três linhas contém os coeficientes JANAF, a_i , para os polinômios das [Equações 4.48](#), [4.49](#) e [4.50](#). A segunda linha contém os coeficientes a_1 até a_5 para o intervalo superior de temperatura. A terceira contém os coeficientes a_6 e a_7 para o intervalo superior, e também a_1 , a_2 e a_3 para o intervalo inferior. A quarta linha contém os coeficientes a_4 até a_7 para o intervalo inferior de temperatura.

CNN	121286C	1N	2	G	0200.00	5000.00	1000.00	1
	0.04785930E+02	0.02559554E-01	0.01003133E-04	0.01807149E-08	0.01227383E-12			2
	0.06870411E+06	0.02953957E+01	0.03524436E+02	0.07271923E-01	0.08272698E-04			3
	0.05628705E-07	0.01641576E-10	0.06899647E+06	0.05932445E+02				4
CNO	103190C	1N	10	1	G	0200.00	4000.00	1500.00
	0.06328598E+02	0.07390401E-02	0.01110761E-05	0.01846498E-09	0.04400816E-13			2
	0.04683387E+06	0.09091839E+02	0.03819863E+02	0.06416255E-01	0.05303312E-04			3
	0.02308211E-07	0.04256414E-11	0.04775979E+06	0.04507300E+02				4
CO	121286C	10	1	G	0200.00	5000.00	1000.00	1
	0.03025078E+02	0.01442689E-01	0.05630828E-05	0.01018581E-08	0.06910952E-13			2
	-0.01426835E+06	0.06108218E+02	0.03262452E+02	0.01511941E-01	0.03881755E-04			3
	0.05581944E-07	0.02474951E-10	0.01431054E+06	0.04848897E+02				4
CO2	121286C	10	2	G	0200.00	5000.00	1000.00	1
	0.04453623E+02	0.03140169E-01	0.01278411E-04	0.02393997E-08	0.01669033E-12			2
	-0.04896696E+06	0.09553959E+01	0.02275725E+02	0.09922072E-01	0.01040911E-03			3
	0.06866687E-07	0.02117280E-10	0.04837314E+06	0.01018849E+03				4
CO2-	121286C	10	2E	1	G	0200.00	5000.00	1000.00
	0.04610574E+02	0.02532962E-01	0.01070165E-04	0.02026771E-08	0.01424958E-12			2
	-0.05479882E+06	0.01449630E+02	0.02637077E+02	0.07803230E-01	0.08196187E-04			3
	0.06537897E-07	0.02520220E-10	0.05416773E+06	0.01188955E+03				4
COS	121286C	10	1S	1	G	0200.00	5000.00	1000.00
	0.05191925E+02	0.02506123E-01	0.01024396E-04	0.01943914E-08	0.01370800E-12			2
	-0.01846210E+06	0.02825755E+02	0.02858531E+02	0.09515458E-01	0.08884915E-04			3
	0.04220994E-07	0.08557340E-11	0.01785145E+06	0.09081989E+02				4

Figura 47 – Trecho do arquivo *therm.dat* do caso *TGA_test*, destacando os parâmetros para o CO₂ e para o CO

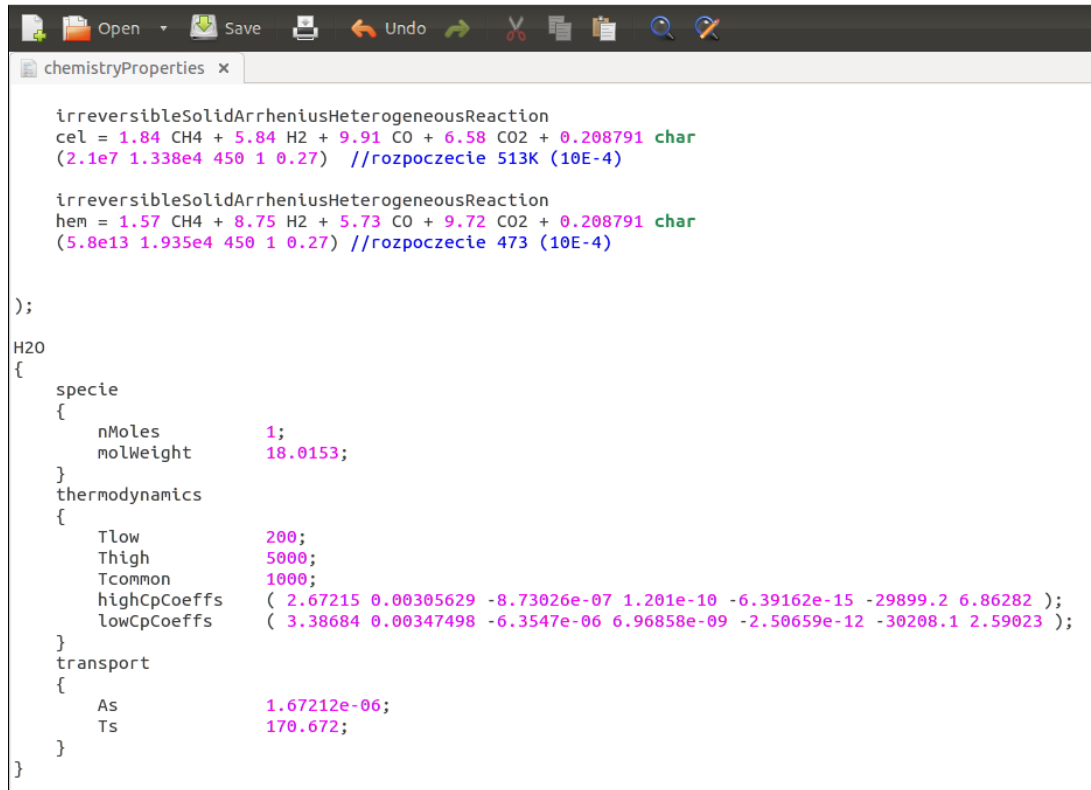
Fonte: Caso *TGA_test* do *biomassGasificationFoam*

No caso do *chemkinReader*, os coeficientes A_s e T_s para o modelo de transporte Sutherland (Equação 4.47) são considerados constantes: $A_s = 1,67212e^{-6}$ e $T_s = 170,672$. Isso significa que todas as espécies gasosas terão os mesmos coeficientes e, conseqüentemente, a mesma viscosidade dinâmica. Esses coeficientes são codificados diretamente (*hard coded*) no arquivo *chemkinLexer.L*, presente no diretório $\$WM_PROJECT_DIR/src/thermophysicalModels/reactionThermo/chemistryReaders/chemkinReader$.

Uma informação importante é que, apesar da classe *chemkinReader* utilizar o formato CHEMKIN com os dados fornecidos em dois arquivos no diretório *chemkin*, o *biomassGasificationFoam* também exige que os parâmetros e propriedades das espécies gasosas também sejam fornecidos no dicionário *chemistryProperties*, presente no diretório *constant* do caso. A Figura 48 mostra como exemplo, um trecho do dicionário *chemistryProperties* do caso *TGA_test* com os parâmetros para a espécie gasosa H₂O. Pode-se identificar os parâmetros A_s e T_s , a massa molar (*molWeight*), as três temperaturas que definem os dois intervalos para os coeficientes JANAF, T_{low} , T_{high} e T_{common} , e os coeficientes a_i para o intervalo superior (*highCpCoeffs*) e para o intervalo inferior (*lowCpCoeffs*).

Uma exceção para a necessidade de declarar a espécie e os parâmetros no dicionário *chemistryProperties* é a espécie inerte, geralmente o N₂. Porém, ela deve ser declarada no dicionário de reações químicas (*chem.inp*, por exemplo), e os parâmetros para as suas propriedades devem ser fornecidos no dicionário *therm.dat*.

Considerando agora a segunda maneira de se fazer a leitura das propriedades físicas das espécies gasosas, se o usuário escolheu a classe *foamChemistryReader*, o dicionário que fornece as propriedades é o *thermo.compressibleGas*, presente no diretório *constant* do caso. O conteúdo



```

chemistryProperties
{
  irreversibleSolidArrheniusHeterogeneousReaction
  cel = 1.84 CH4 + 5.84 H2 + 9.91 CO + 6.58 CO2 + 0.208791 char
  (2.1e7 1.338e4 450 1 0.27) //rozpoczecie 513K (10E-4)

  irreversibleSolidArrheniusHeterogeneousReaction
  hem = 1.57 CH4 + 8.75 H2 + 5.73 CO + 9.72 CO2 + 0.208791 char
  (5.8e13 1.935e4 450 1 0.27) //rozpoczecie 473 (10E-4)
);

H2O
{
  specie
  {
    nMoles      1;
    molWeight   18.0153;
  }
  thermodynamics
  {
    Tlow        200;
    Thigh       5000;
    Tcommon     1000;
    highCpCoeffs ( 2.67215 0.00305629 -8.73026e-07 1.201e-10 -6.39162e-15 -29899.2 6.86282 );
    lowCpCoeffs  ( 3.38684 0.00347498 -6.3547e-06 6.96858e-09 -2.50659e-12 -30208.1 2.59023 );
  }
  transport
  {
    As          1.67212e-06;
    Ts          170.672;
  }
}

```

Figura 48 – Trecho do arquivo *chemistryProperties* do caso *TGA_test*, demonstrando os parâmetros para o H₂O

Fonte: Caso *TGA_test* do *biomassGasificationFoam*

desse dicionário deve conter os parâmetros para as propriedades físicas de forma idêntica à fornecida no dicionário *chemistryProperties* e exemplificada na Figura 48 para o H₂O.

Novamente, apesar de fornecermos esses parâmetros no dicionário *thermo.compressibleGas*, o *biomassGasificationFoam* também exige que os parâmetros e propriedades das espécies gasosas também sejam fornecidos no dicionário *chemistryProperties*, com exceção da espécie inerte, que deve estar declarada no dicionário *reactions* e seus parâmetros devem estar presentes no *thermo.compressibleGas*.

4.9 heterogeneousPyrolysisModel

O último componente da *biomassGasificationMedia* é a biblioteca *pyrolysisModels*, composta pela classe base *heterogeneousPyrolysisModel* e pela classe derivada *volPyrolysis*, cuja função é realizar a integração de toda a modelagem referente à fase sólida. Ela é responsável por criar os campos de propriedades do sólido necessários, como os campos de frações mássicas dos componentes sólidos, e por integrar os submodelos físicos da fase sólida, como cinética, radiação e transferência de calor. Ela também calcula os termos fontes devidos ao sólido, soluciona as equações de conservação para a fase sólida e calcula a evolução do campo de porosidade.

O termo utilizado por Kwiatkowski et al. (2013) para essas bibliotecas, *pyrolysisModels*,

seria traduzido como modelos de pirólise. Esse termo, traduzido literalmente, leva a crer que essas bibliotecas tratam apenas da modelagem do processo de pirólise, ou seja, da decomposição do sólido pela ação de calor. Na realidade, como comentado acima, essas bibliotecas fazem a modelagem de todos os processos envolvendo a fase sólida.

O modelo específico a ser utilizado deve ser escolhido no dicionário *pyrolysisProperties*, presente no diretório *constant* do caso. Porém, o único modelo originalmente implementado no *biomassGasificationFoam* é o *volPyrolysis*, para processos volumétricos. Ele também pode ser definido como ativo ou inativo. Se inativo, nenhum cálculo é realizado para os campos da fase sólida.

A classe *volPyrolysis* faz a integração de todas as outras bibliotecas para os modelos físicos da fase sólida, como as presentes em */biomassGasificationMedia/thermophysicalModels*. Assim, uma de suas atribuições é definir campos volumétricos para as propriedades necessárias, como porosidade, frações mássicas, condutividade térmica, massa específica, entre outros. Ela também integra os processos térmicos e termoquímicos, como reações químicas da fase sólida (cinética e calor de reação), transferência de calor entre sólido e gás, e radiação.

Uma das funções da classe *volPyrolysis* é, portanto, calcular os termos fontes devidos à fase sólida para as equações de conservação. Eles incluem: o termo de resfriamento ou aquecimento dos produtos gasosos das reações do sólido até a temperatura do gás, Γ , presente na equação de conservação da energia da fase gasosa (Equação 4.13); o termo de radiação para a fase sólida, $S^{S^{rad}}$, presente na equação de conservação da energia da fase sólida (Equação 4.11); e o termo de transferência de calor entre sólido e gás, $\alpha\Sigma(T^G - T^S)$, presente em ambas as equações de conservação da energia.

Ela também integra os termos fontes calculados em outras bibliotecas específicas, como: os termos de consumo ou formação das espécies gasosas provenientes das reações químicas do sólido, R_i^G , que aparecem na equação da continuidade (Equação 4.5) e na equação de conservação das espécies para a fase gasosa (Equação 4.7); os termos de consumo ou formação das espécies sólidas, R_k^S , presente na equação de conservação das espécies para a fase sólida (Equação 4.6); e o termo fonte de energia devido às reações químicas da fase sólida, H_r , pertencente à equação de conservação da energia da fase sólida (Equação 4.11).

Por fim, a classe *volPyrolysis* soluciona as equações de conservação para a fase sólida e calcula a evolução do campo de porosidade. A equação de conservação das espécies para a fase sólida (Equação 4.6) é solucionada na função *solveSpeciesMass()*. A equação de conservação da energia da fase sólida (Equação 4.11) é solucionada na função *solveEnergy()*. E a evolução do campo de porosidade (Equação 4.3) é solucionada na função *evolvePorosity()*.

5 O Transporte de Calor e Massa em Meio Poroso Reativo

Neste capítulo foram realizados testes com o intuito de avaliar e validar alguns dos modelos implementados no `biomassGasificationFoam`. Primeiramente foi analisado o transporte de espécies da fase gasosa, tanto difusivo quanto advectivo. Em seguida a transferência de calor entre as fases sólida e gasosa e a sua influência na temperatura do leito. Por fim, foi analisado o modelo termoquímico das reações em fase sólida, a implementação do calor de reação e da cinética química.

5.1 Transporte de Espécies Gasosas

A implementação do transporte de espécies gasosas foi herdada do `solver reactingFoam` e adaptada para os processos de conversão termoquímica de sólidos, introduzindo o campo de porosidade e as taxas de consumo ou formação de espécies devido às reações na fase sólida (Equação 4.10).

Nesta seção foram avaliados os transportes difusivo e advectivo das espécies no meio poroso. Para tanto, o meio foi considerado inerte, não havendo reações químicas heterogêneas ou homogêneas. Isso implica que os termos de consumo ou formação de espécies devido às reações na fase sólida, R_i^G , são nulos e que a porosidade é constante. Também foi considerado que as temperaturas do gás e do sólido são constantes, iguais entre si e também iguais por todo o leito, não havendo troca de calor entre as fases e nem com o meio externo.

5.1.1 Difusão

Para a validação da difusão das espécies, além das considerações citadas acima, a velocidade do gás foi considerada nula, de forma a cancelar o termo advectivo. Da equação de conservação das espécies para a fase gasosa (Equação 4.7) sobram apenas o termo temporal e o termo difusivo. Para uma espécie i essa equação se torna:

$$\frac{\partial \gamma \rho^G Y_i^G}{\partial t} - \nabla \cdot (\rho^G D^* \nabla Y_i^G) = 0 \quad (5.1)$$

Como comentado na [subseção 4.5.4](#), o `biomassGasificationFoam` faz a consideração de Número de Schmidt igual a um, ou seja, que a difusividade de momento e a difusividade mássica são iguais. Assim, na equação de conservação das espécies para a fase gasosa ele substitui o termo $\rho^G D^*$ pela viscosidade dinâmica μ (ver [Equação 4.10](#)). A [Equação 5.1](#) com essa simplificação se torna:

$$\frac{\partial \gamma \rho^G Y_i^G}{\partial t} - \nabla \cdot (\mu \nabla Y_i^G) = 0 \quad (5.2)$$

Com o intuito de avaliar a influência dessa simplificação adotada, foram realizadas simulações para a difusão de CO₂ em um leito preenchido com N₂ puro, para duas temperaturas diferentes (300 K e 1000 K), comparando os resultados obtidos pela [Equação 5.1](#) e pela [Equação 5.2](#).

A implementação da equação de conservação das espécies para fase gasosa é feita no arquivo *YEqn.H*, presente no diretório */biomassGasificationFoam*, conforme a [Equação 4.10](#). Algumas alterações nesse arquivo foram necessárias para que possa ser utilizado o coeficiente de difusão mássica efetivo D^* . Nessas modificações ele foi considerado constante. A [Figura 49](#) mostra o trecho do arquivo *YEqn.H* modificado.

Primeiramente deve-se criar um campo escalar volumétrico para o coeficiente D^* . No exemplo foi chamado de DiffG e atribuído valor constante de $1,6576 \times 10^{-5}$. O termo convectivo foi retirado, os termos fontes serão nulos e o termo difusivo, com o laplaciano, foi modificado de forma a representar a [Equação 5.1](#).

Segundo [Incropera et al. \(2014\)](#) e [Çengel \(2009\)](#), para uma mistura de dois gases ideais, a teoria cinética dos gases indica que o coeficiente binário de difusão mássica D_{AB} é proporcional à temperatura e pressão da seguinte forma:

$$D_{AB} \propto \frac{T^{3/2}}{p} \quad (5.3)$$

O que leva a:

$$\frac{D_{AB,1}}{D_{AB,2}} = \frac{p_2}{p_1} \left(\frac{T_1}{T_2} \right)^{3/2} \quad (5.4)$$

Essa relação é muito útil na determinação do coeficiente de difusão dos gases em diferentes pressões e temperaturas a partir do conhecimento do coeficiente de difusão em uma determinada temperatura e pressão. Considerando que a simulação ocorre à pressão atmosférica, é possível utilizar um valor para uma temperatura presente em tabelas como valor de referência para a obtenção do coeficiente de difusão à uma temperatura T :

$$D_{AB,T} = D_{AB,T_{Ref}} \times \left(\frac{T}{T_{Ref}} \right)^{3/2} \quad (5.5)$$

Da Tabela A.8 de [Incropera et al. \(2014\)](#), o valor do coeficiente de difusão binária para o CO₂ em N₂ a uma atmosfera é $D_{AB} = 0,16 \times 10^{-4}$, para uma temperatura de 293 K. Esses valores

```

YEqn.H x
tmp<fv::convectionScheme<scalar> > mvConvection
(
    fv::convectionScheme<scalar>::New
    (
        mesh,
        fields,
        phi,
        mesh.divScheme("div(phi,Yi_h)")
    )
);
/*-----*/
volScalarField DiffG
(
    IOobject
    (
        "DiffG",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::NO_WRITE
    ),
    mesh,
    dimensionedScalar("zero",dimLength*dimLength/dimTime,1.6576e-5)
);
/*-----*/
{
    label inertIndex = -1;
    volScalarField Yt = 0.0*Y[0];

    forAll(Y, i)
    {
        if (Y[i].name() != inertSpecie)
        {
            volScalarField& Yi = Y[i];

            fvScalarMatrix YiEqn
            (
                fvm::ddt(oneMinusPorosityF*rho, Yi)
            // + mvConvection->fvmDiv(phi, Yi)
            // - fvm::laplacian(turbulence->muEff(), Yi)
            - fvm::laplacian(rho*DiffG, Yi)
            ==
                kappa*chemistry.RR(i)
                + pyrolysisZone.Srho(i)()
            );

            YiEqn.relax();
            YiEqn.solve(mesh.solver("Yi"));

            Yi.max(0.0);
        }
    }
}

```

Figura 49 – Trecho do arquivo *YEqn.H* modificado

foram utilizados como referência para calcular os coeficientes de difusão para as temperaturas de 300 K e 1000 K.

Na literatura existem alguns modelos para calcular o coeficiente de difusão mássica efetivo para um meio poroso, D^* . Neste trabalho foi utilizado o modelo presente em Webb (2006), que diz que o coeficiente de difusão para um meio poroso pode ser dado por:

$$D^* = \beta D_{AB} \quad (5.6)$$

O termo β é definido como:

$$\beta = \gamma S_g \tau \quad (5.7)$$

Onde γ é a fração de vazio, S_g é a saturação do gás (igual a 1 para a condição de apenas gases no fluido), e τ é a tortuosidade. A inclusão do termo β leva em conta a área efetiva para o escoamento do gás nos poros (γS_g) e a tortuosidade do meio poroso (τ) (WEBB, 2006).

Existem vários modelos para o fator de tortuosidade. Uma das correlações mais utilizadas é a de Millington e Quirk (1961), que é dada por:

$$\tau = \gamma^{1/3} S_g^{7/3} \quad (5.8)$$

Para uma condição de apenas gás no fluido, onde $S_g = 1$, temos:

$$\tau_{(S_g=1)} = \gamma^{1/3} \quad (5.9)$$

Substituindo as Equações 5.9 e 5.7 na Equação 5.6, e identificando que a simulação apresenta uma condição de apenas gases no fluido, chega-se à seguinte equação para o coeficiente efetivo de difusão mássica no meio poroso:

$$D^* = \gamma^{4/3} D_{AB} \quad (5.10)$$

Com auxílio das Equações 5.5 e 5.10, é possível calcular o coeficiente de difusão mássica D_{AB} e o coeficiente efetivo D^* para as temperaturas de 300 K e 1000 K, considerando a fração de vazio $\gamma = 0,47$. Os dados utilizados nas simulações estão apresentados na Tabela 5.

A primeira simulação foi para uma temperatura de 300 K. A Figura 50 apresenta o gráfico com os resultados para os tempos iguais a 0,5 e 1 segundo, fazendo uma comparação dos resultados obtidos sem a simplificação de Schmidt igual a um (Equação 5.1) e com a simplificação (Equação 5.2). É possível notar uma clara diferença entre os resultados. A utilização da viscosidade dinâmica (simplificação) resulta em um processo de difusão consideravelmente mais rápido que a utilização do coeficiente de difusão mássica efetivo. Portanto, para esse caso, a hipótese de Schmidt igual a um causa uma superestimação do fenômeno de difusão.

A segunda simulação foi para uma temperatura de 1000 K. Como essa temperatura é mais alta, o processo de difusão é mais rápido e os dados das simulações foram para os tempos iguais a 0,05 e 0,1 segundo. A Figura 51 apresenta o gráfico com os resultados, novamente fazendo uma comparação dos resultados obtidos sem a simplificação de Schmidt igual a um e com a simplificação. Ele mostra que a diferença entre os resultados é ainda maior para 1000 K. O

Tabela 5 – Parâmetros para a simulação da difusão do CO₂

Parâmetro	Valor	Unidade
Tamanho do leito	0,01	m
Subdivisões do domínio	100	-
Tempo final	2	s
Intervalo de tempo	0,0002	s
$Y_{CO_2} _{x=0}$	0	-
$Y_{CO_2} _{x=L}$	1	-
γ	0,47	-
D_{AB} (300 K)	$1,6576 \times 10^{-5}$	m ² /s
D^* (300 K)	$6,0575 \times 10^{-6}$	m ² /s
D_{AB} (1000 K)	$1,0088 \times 10^{-4}$	m ² /s
D^* (1000 K)	$3,6864 \times 10^{-5}$	m ² /s

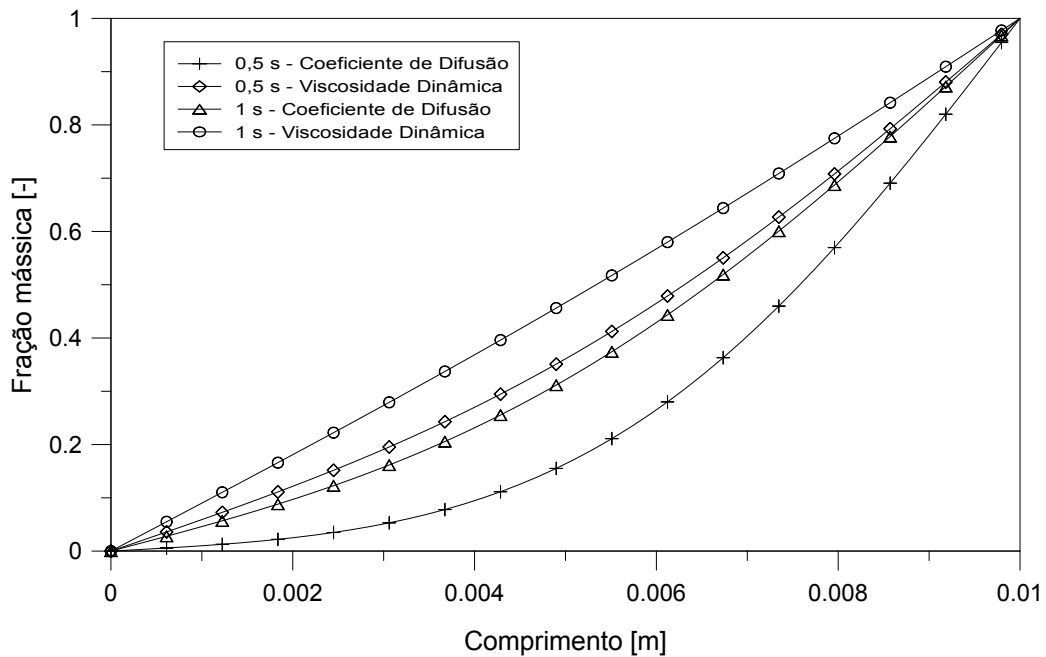


Figura 50 – Comparação dos resultados para a difusão obtidos com o uso do coeficiente de difusão efetivo D^* e com a viscosidade dinâmica μ para uma temperatura de 300 K

processo de difusão utilizando a viscosidade dinâmica (simplificação) é bem mais rápido que utilizando o coeficiente de difusão.

Os resultados dessas duas simulações mostraram que a hipótese adotada no `biomassGasificationFoam` de Número de Schmidt igual a um, que leva à [Equação 5.2](#), superestima o fenômeno de difusão tanto para temperaturas altas como para temperaturas baixas. Como dito anteriormente, a implementação da equação de conservação das espécies para a fase gasosa

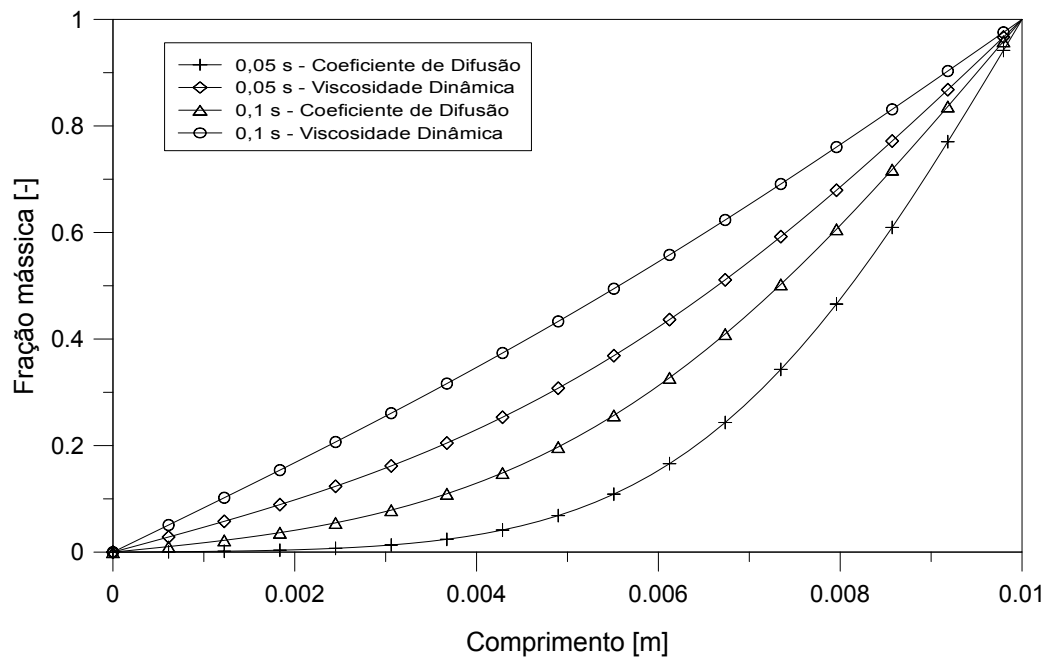


Figura 51 – Comparação dos resultados para a difusão obtidos com o uso do coeficiente de difusão efetivo D^* e com a viscosidade dinâmica μ para uma temperatura de 1000 K

foi herdada do *solver reactingFoam*, que foi designado para escoamentos reativos transientes, mas apenas com fase gasosa, sem um meio poroso. Isso levou a questionar se a simplificação implementada é mais razoável quando há apenas fase gasosa, porém se torna inadequada quando há um meio poroso.

Mais duas simulações foram então realizadas, uma para 300 K e outra para 1000 K, porém agora fazendo uso do coeficiente de difusão binária do gás, D_{AB} , ao invés de usar o coeficiente efetivo para meio poroso, D^* . A Figura 52 mostra a comparação dos resultados obtidos sem a simplificação (com o coeficiente de difusão D_{AB}) e com a simplificação (com a viscosidade dinâmica) para uma temperatura de 300 K. Nesse caso, nota-se que as curvas estão bem mais próximas. O processo de difusão com a simplificação foi até um pouco mais lento, resultado contrário ao que foi obtido anteriormente.

A Figura 53 mostra a comparação dos resultados obtidos sem a simplificação (com o coeficiente de difusão D_{AB}) e com a simplificação (com a viscosidade dinâmica) para uma temperatura de 1000 K. Novamente as curvas estão mais próximas. Entretanto, dessa vez a difusão voltou a ser mais rápida com o uso da viscosidade dinâmica.

As duas últimas simulações utilizaram o coeficiente de difusão mássica binária D_{AB} . Seus resultados confirmaram que a hipótese de Número de Schmidt igual a um é razoável para um processo onde há apenas a fase gasosa. Porém, essa simplificação não se mostrou adequada para processos onde exista um meio poroso, causando uma superestimação do fenômeno de difusão nesses casos.

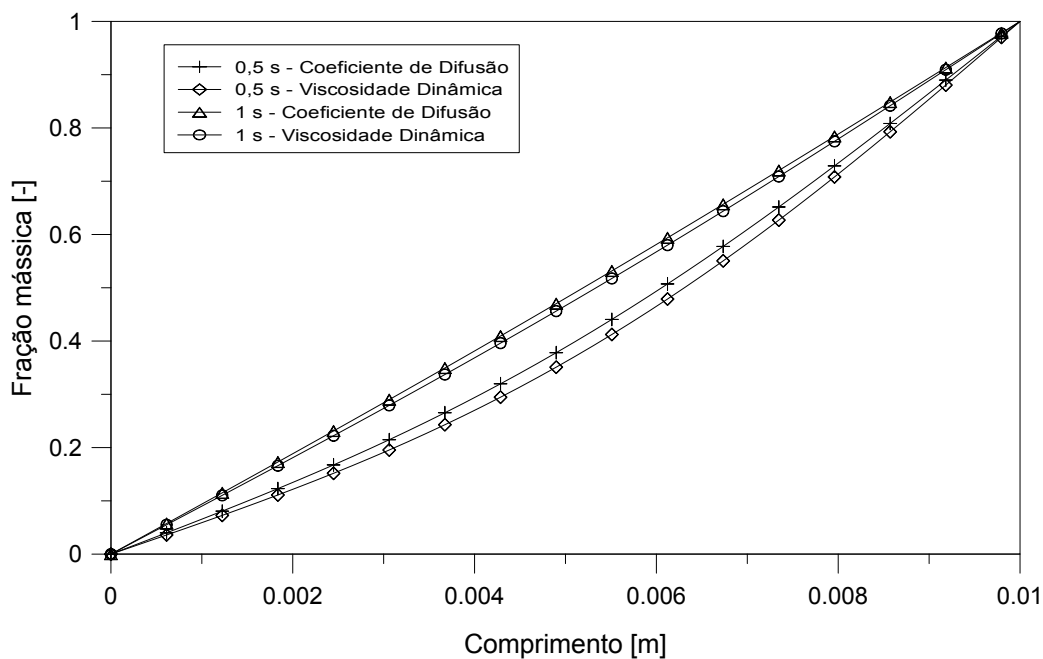


Figura 52 – Comparação dos resultados para a difusão obtidos com o uso do coeficiente de difusão D_{AB} e com a viscosidade dinâmica μ para uma temperatura de 300 K

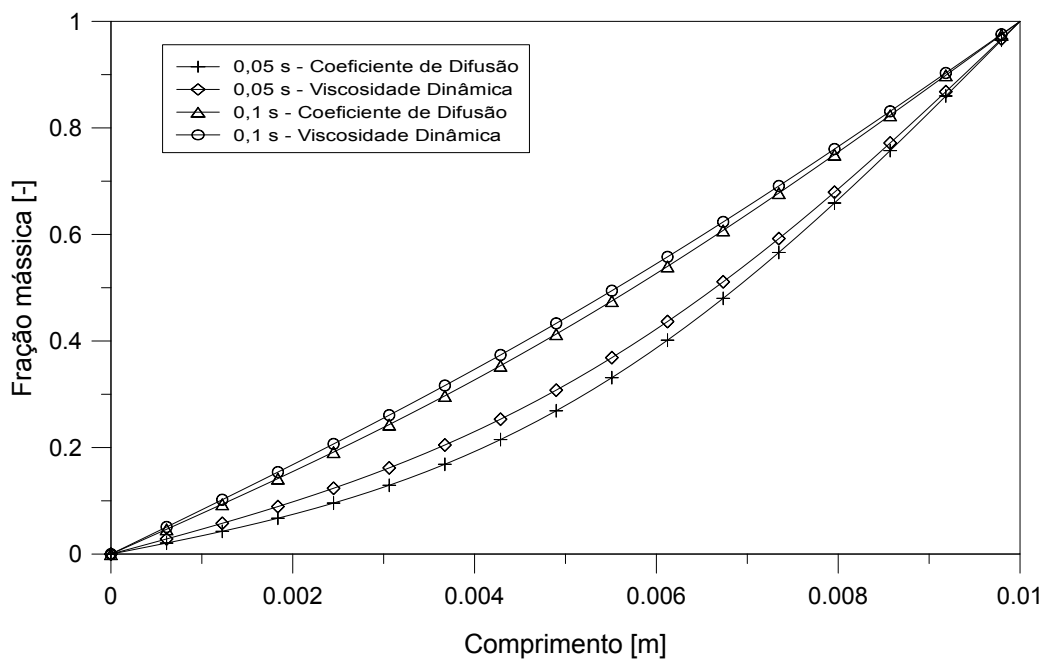


Figura 53 – Comparação dos resultados para a difusão obtidos com o uso do coeficiente de difusão D_{AB} e com a viscosidade dinâmica μ para uma temperatura de 1000 K

5.1.2 Advecção

Para a verificação da advecção das espécies gasosas, os termos fontes e o termo difusivo da equação de conservação das espécies para a fase gasosa (Equação 4.7) foram considerados nulos, sobrando apenas o termo temporal e o termo advectivo, resultando na seguinte equação para

uma espécie i :

$$\frac{\partial \gamma \rho^G Y_i^G}{\partial t} + \nabla \cdot (\rho^G \mathbf{u} Y_i^G) = 0 \quad (5.11)$$

Considerando também que a massa específica da espécie i é constante e que o problema é unidimensional, e lembrando que a porosidade é constante por ser um meio inerte, temos:

$$\frac{\partial Y_i^G}{\partial t} + \frac{u}{\gamma} \frac{\partial Y_i^G}{\partial x} = 0 \quad (5.12)$$

Os parâmetros utilizados na simulação do processo de advecção de uma espécie foram os mesmos utilizados por Lapene (2006), e estão apresentados na Tabela 6.

Tabela 6 – Parâmetros para a simulação da advecção

Parâmetro	Valor	Unidade
Tamanho do leito	0,01	m
Tempo final	20	s
Intervalo de tempo	0,0005	s
$Y_i _{x=0}$	1	-
$Y_i _{x=L}$	0	-
γ	0,407	-
u	$1,7 \times 10^{-4}$	m/s

A Figura 54 apresenta os resultados obtidos com a simulação numérica juntamente com a solução analítica da Equação 5.12 dada por Lapene (2006), para tempos iguais a 5, 10, 15 e 20 segundos. Para essa simulação, a malha foi dividida em 100 partes iguais. Os resultados numérico e analítico apresentam uma certa diferença. A solução numérica lembra a solução de uma equação de convecção-difusão. Isso ocorre devido ao fenômeno de difusão numérica característico do esquema de discretização *upwind* do método dos volumes finitos.

Uma opção para diminuir o efeito de difusão numérica é o refino da malha. Assim, foi realizada uma simulação para a advecção de uma espécie gasosa com os mesmos parâmetros da simulação anterior, apenas aumentando o número de divisões da malha para 1000 partes iguais. A Figura 55 mostra os resultados da simulação numérica juntamente com a solução analítica. Pode-se notar que o efeito de difusão numérica foi menor, mostrando a influência da malha.

5.2 Transferência de Calor Entre Gás e Sólido

A hipótese de não-equilíbrio térmico local implica que as temperaturas do sólido e do gás são diferentes, havendo duas equações de conservação da energia, uma para a fase sólida e outra

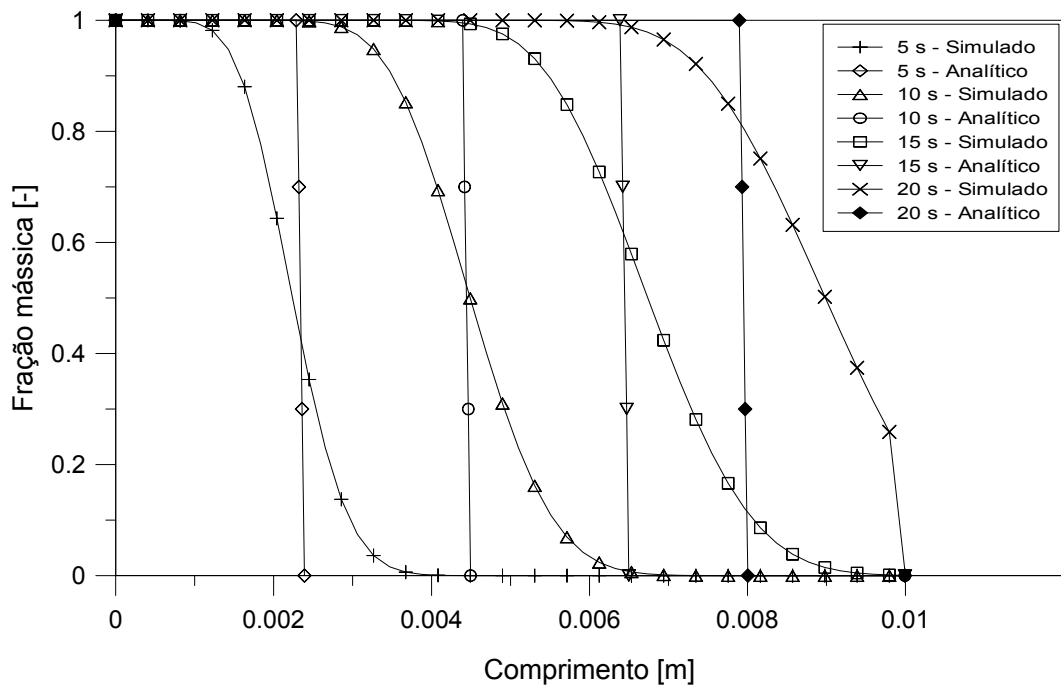


Figura 54 – Frações mássicas para a simulação da advecção para uma malha com 100 divisões

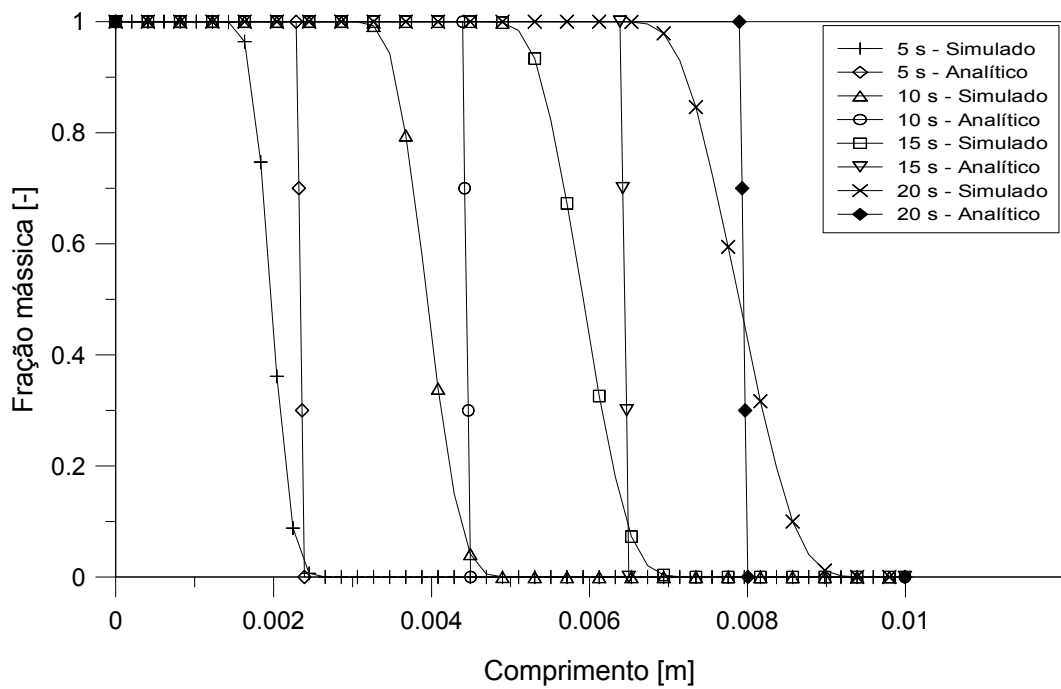


Figura 55 – Frações mássicas para a simulação da advecção para uma malha com 1000 divisões

para a fase gasosa. Considerar essa hipótese se torna mais relevante para a simulação de reatores completos, como de gaseificação ou de combustão. Um exemplo da diferença de temperatura entre as fases ocorre para os gases frios que entram em uma célula de combustão co-corrente e encontram a fase sólida quente, resultante do processo de combustão.

A diferença de temperatura entre a fase sólida e a gasosa induz uma transferência de calor entre elas. Essa transferência de calor apresenta grande importância na evolução do processo de conversão termoquímica do sólido. Assim, algumas simulações foram realizadas para avaliar a implementação da transferência de calor no `biomassGasificationFoam`, comparando os resultados com os obtidos no experimento realizado por Martins (2008).

O experimento de Martins (2008) observou a evolução da temperatura durante o resfriamento de um leito poroso, que consiste em uma célula de combustão cilíndrica com 300 mm de altura e 91 mm de diâmetro, como ilustrado na Figura 56. O leito está inicialmente a uma temperatura de 42 °C. No topo da célula há um escoamento de ar entrando com uma vazão de 16,5 l/min e temperatura de 15,8 °C.

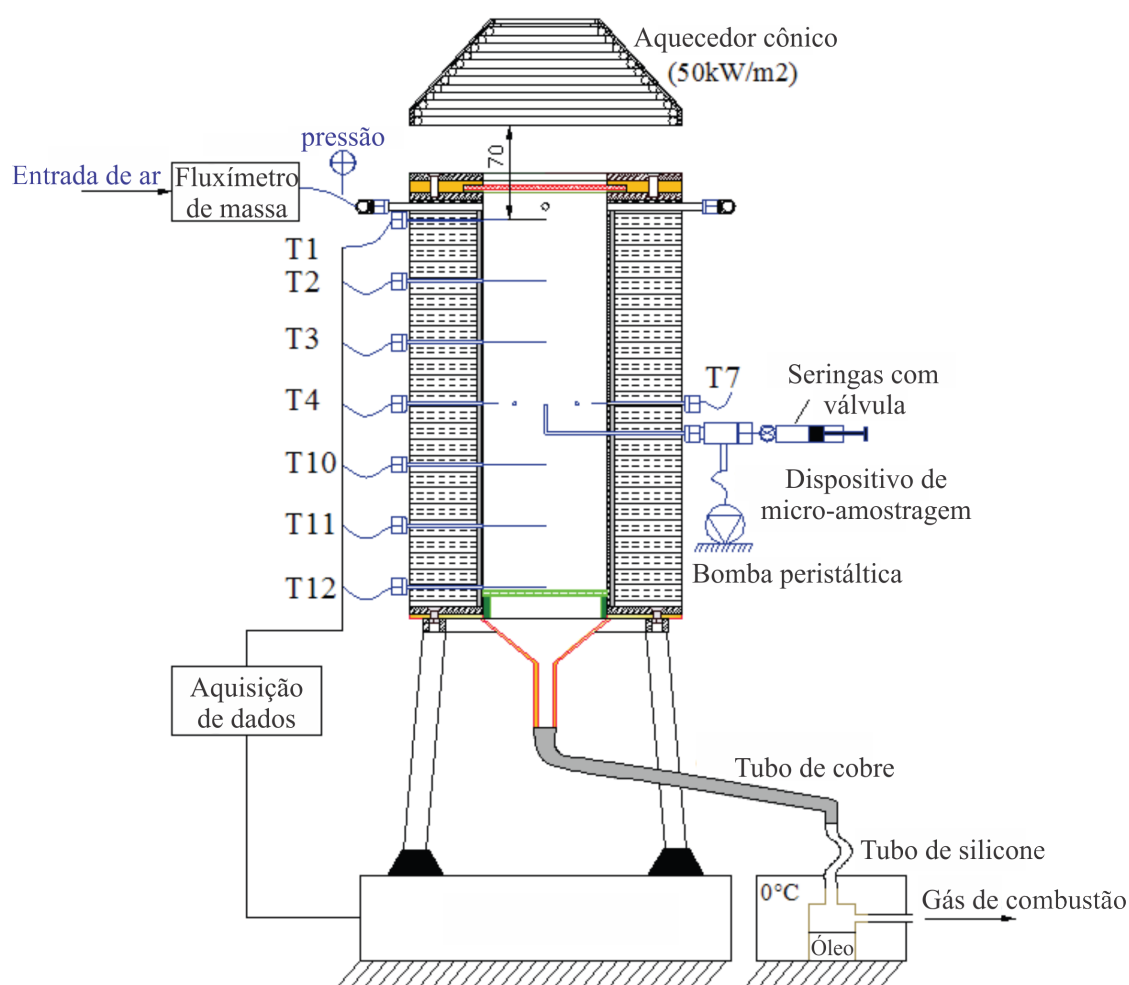


Figura 56 – Croqui da célula de combustão de Martins (2008)

Fonte: Adaptado de Martins (2008)

Como o fenômeno simulado foi apenas o resfriamento do leito, os termos fontes de energia relacionados às reações químicas são nulos. O ar de entrada foi considerado como composto por 76,7% de nitrogênio e 23,3% de oxigênio em base mássica. Os dados utilizados nas simulações estão apresentados na Tabela 7.

Tabela 7 – Parâmetros para a simulação da transferência de calor

Parâmetro	Valor	Unidade
Tempo final	5600	s
Intervalo de tempo	0,002	s
$T_{t=0}$	42	°C
T_e	15,8	°C
γ	0,47	-
Permeabilidade	$1,8 \times 10^{-10}$	m^2
u	0,04228	m s^{-1}
c_p^S	830	$\text{J kg}^{-1} \text{K}^{-1}$
ρ^S	1168	kg m^{-3}
k^S	0,395	$\text{W m}^{-1} \text{K}^{-1}$
Σ	2321	$\text{m}^2 \text{m}^{-3}$

O modelo de transferência de calor do `biomassGasificationFoam` utilizado foi o `constHTC`, comentado na [subseção 4.8.3](#). Portanto, a área superficial específica dos poros foi considerada constante com valor $\Sigma = 2321 \text{ m}^2 \text{ m}^{-3}$, dada por [Martins \(2008\)](#). Nas simulações, o coeficiente de transferência de calor α foi alterado em busca de se obter uma curva de resfriamento compatível com a obtida no experimento de [Martins \(2008\)](#). O perfil de temperatura foi avaliado a uma distância de 45 mm abaixo do topo da célula, ponto referido como termopar T2 no trabalho de [Martins \(2008\)](#).

A simulação que mostrou um melhor resultado foi com $\alpha = 1,3 \text{ W m}^{-2} \text{ K}^{-1}$, que resulta no termo $\alpha\Sigma = 3017,3 \text{ W m}^{-3} \text{ K}^{-1}$. A [Figura 57](#) mostra a comparação entre o perfil de temperatura experimental e os perfis de temperatura do sólido (T_s) e do gás (T_g) obtidos com a simulação. Eles apresentam um comportamento semelhante, indicando que o modelo implementado é satisfatório.

Para ilustrar a influência do coeficiente de transferência de calor α nos perfis de temperatura do sólido e do gás, serão mostrados alguns resultados obtidos com outros valores de α . A [Figura 58](#) mostra os resultados para $\alpha = 0,1 \text{ W m}^{-2} \text{ K}^{-1}$, que resulta no termo $\alpha\Sigma = 232,1 \text{ W m}^{-3} \text{ K}^{-1}$. Como visto no gráfico, o baixo coeficiente de transferência de calor causa uma grande diferença entre os perfis de temperatura do sólido e do gás, com temperaturas bem distantes das encontradas experimentalmente. O gás frio que entra na célula possui uma menor capacidade de resfriar o leito.

A [Figura 59](#) mostra os resultados para $\alpha = 10 \text{ W m}^{-2} \text{ K}^{-1}$, que resulta no termo $\alpha\Sigma = 23210 \text{ W m}^{-3} \text{ K}^{-1}$. O alto coeficiente de transferência de calor faz com que os perfis de temperatura do sólido e do gás fiquem bem próximos. No entanto, as temperaturas são menores que as obtidas experimentalmente.

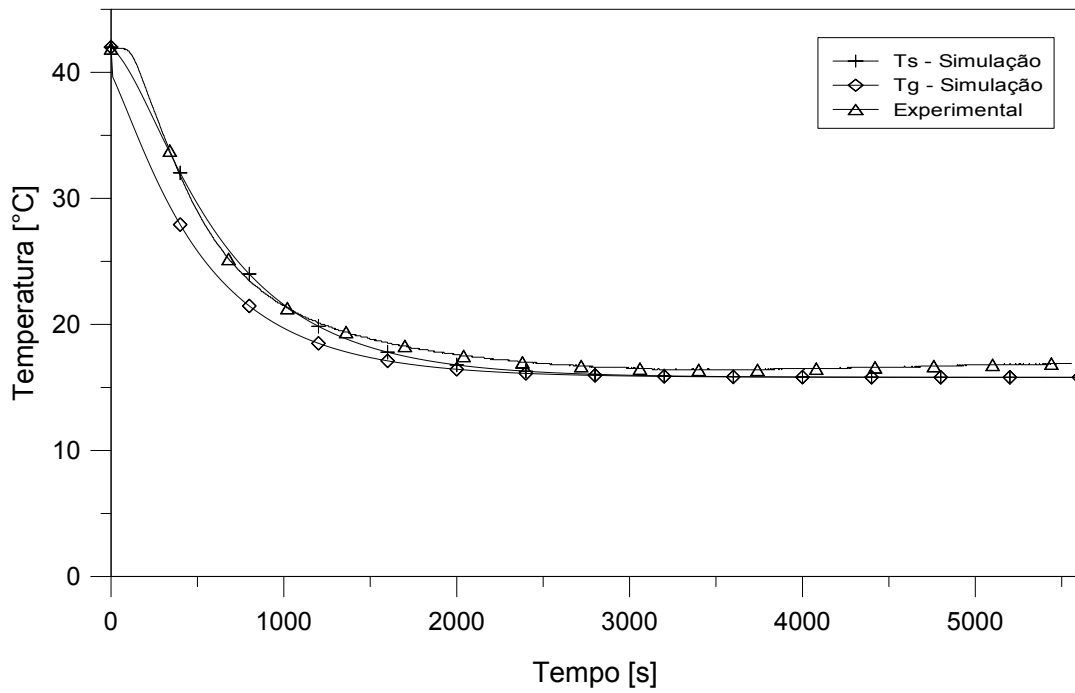


Figura 57 – Perfis de temperatura para $\alpha = 1,3 \text{ W m}^{-2} \text{ K}^{-1}$

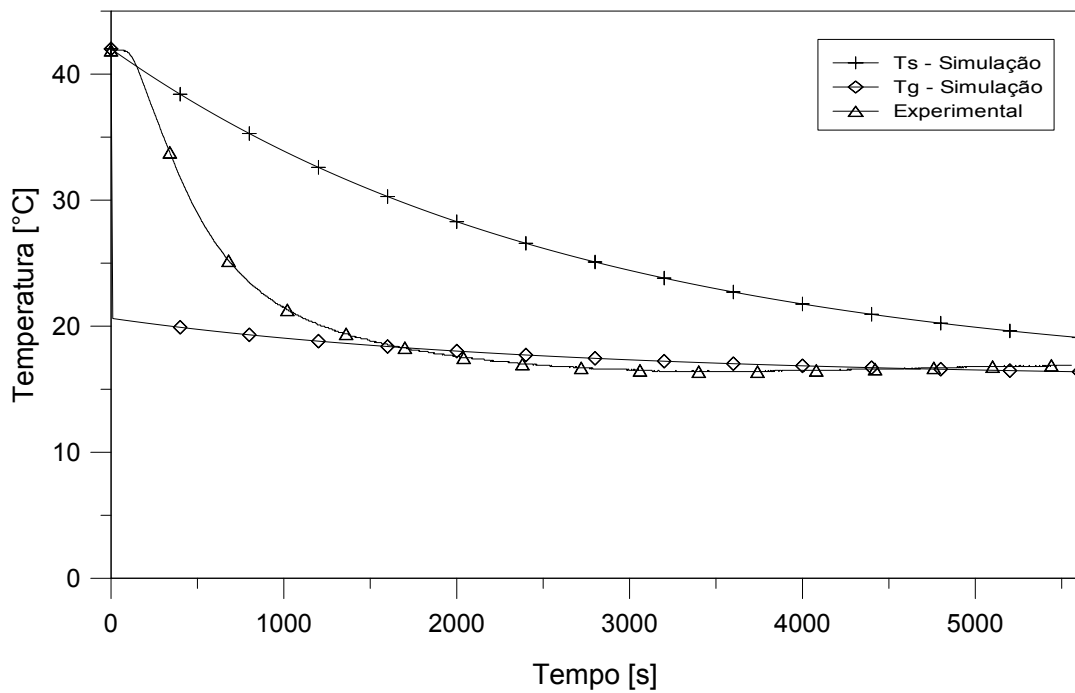


Figura 58 – Perfis de temperatura para $\alpha = 0,1 \text{ W m}^{-2} \text{ K}^{-1}$

5.3 Modelo Termoquímico das Reações na Fase Sólida

Nesta seção foi avaliada a implementação do modelo termoquímico das reações na fase sólida. A principal classe desse modelo é a *ODESolidHeterogeneousChemistryModel*, que estende o modelo químico básico adicionando um pacote termodinâmico e funções de EDOs. Ela integra

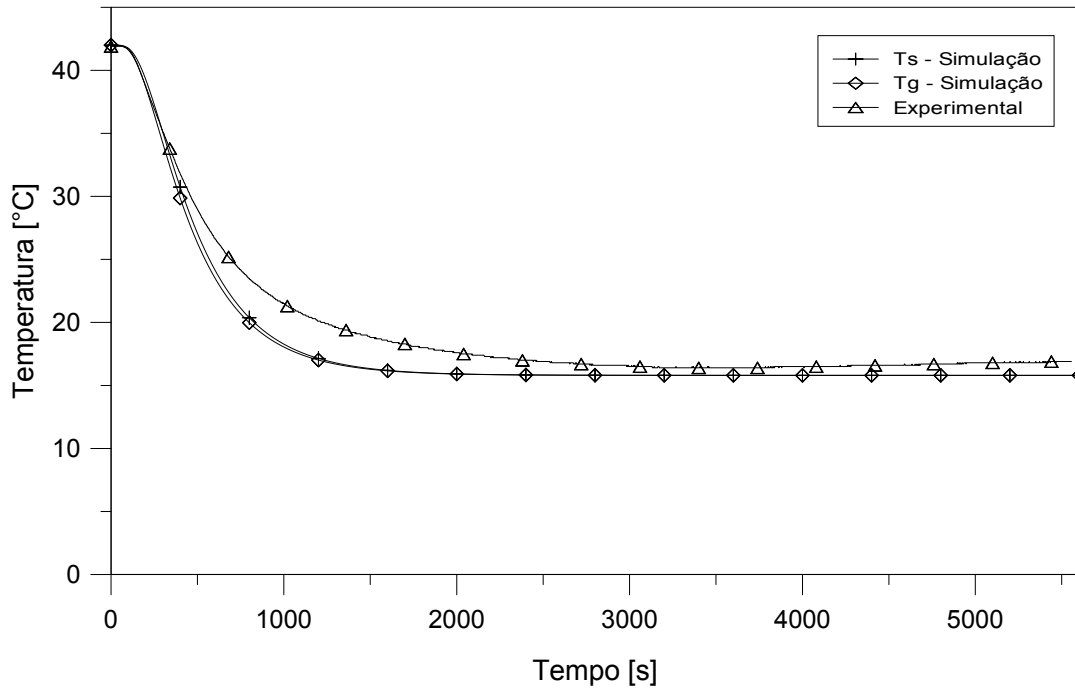


Figura 59 – Perfis de temperatura para $\alpha = 10 \text{ W m}^{-2} \text{ K}^{-1}$

as outras bibliotecas de reações químicas e taxa de reação para calcular a cinética química e os termos fontes de massa e energia devidos às reações na fase sólida. Essa classe foi criada baseada na *ODESolidChemistryModel*, fornecida originalmente com o OpenFOAM. Informações sobre os modelos implementados nessa classe foram dadas na [subseção 4.8.1.1](#) e na [subseção 4.8.2](#).

Uma descrição básica das principais funções da classe *ODESolidHeterogeneousChemistryModel* será dada abaixo, para possibilitar referenciá-las posteriormente:

- *omega(c, T, p, updateC0)* - Realiza o cálculo das taxas de reação totais para cada espécie i , R_i^{tot} , somando as taxas de reação R_i para todas as reações, descritas nas Equações 4.18, 4.20, 4.23 e 4.24. As taxas de reação R^{tot} são armazenadas em um vetor chamado `om[]`, nas posições correspondentes à cada espécie. A diferença é que a unidade dessas taxas de reação é em kg s^{-1} , e não em $\text{kg m}^{-3} \text{ s}^{-1}$. Isso ocorre devido ao cálculo da taxa Ω_r , comentado na próxima função. Essa taxa R^{tot} será usada nas funções relacionadas à solução do sistema de EDOs da cinética química. A taxa volumétrica, $R^{tot'''}$, que entra como termo fonte de massa, será calculada na função *solve(t0, deltaT)*. Se a energia de reação não for calculada pelas entalpias de formação e sim pelo calor de reação h_r fornecido para cada reação (quando a variável *solidReactionEnergyFromEnthalpy* for declarada **false**), a última posição do vetor `om[]` será preenchida com o somatório do efeito energético de cada reação da seguinte forma:

$$H_r = \sum_r \Omega_r (1 - \xi_r) h_r \quad (5.13)$$

Onde ξ_r é a razão mássica entre produtos e substratos sólidos. Isso demonstra a necessidade de se declarar o calor de reação h_r por unidade de massa perdida pela fase sólida, como comentado na [subseção 4.8.2](#). Aqui, também devido ao cálculo de Ω_r , a unidade de H_r será J s^{-1} . Para esse caso, a taxa de energia por unidade de volume será calculada na função `solve(t0, deltaT)`.

- `omega(R, c, T, p, pf, cf, lRef, pr, cr, rRef)` - Responsável pelo cálculo da taxa de reação Ω_r :

$$\Omega_r = \rho^S \left(Y_s^S \right)^{n_r} \left(Y^G \right)^{\delta^G} k_r V \quad (5.14)$$

A diferença para a equação [Equação 4.17](#) é que, ao calcular a taxa de reação Ω_r para cada célula computacional, ela multiplica pelo volume da célula, V , obtendo uma taxa Ω_r com unidade de kg s^{-1} , e não $\text{kg m}^{-3} \text{s}^{-1}$.

- `derivatives(time, c, dcdt)` - Função que calcula a expressão para a variação da massa no tempo para cada espécie (R^{tot}), e para a variação da temperatura e da pressão no tempo. Utilizada pelas classes responsáveis pela solução do sistema de EDOs.
- `jacobian(t, c, dcdt, dfdc)` - Calcula a matriz jacobiana para a linearização do sistema de EDOs. Utilizada pelas classes responsáveis pela solução do sistema de EDOs.
- `Sh()` - Essa função calcula o calor de reação se o método escolhido foi pelas entalpias de formação ([Equação 4.26](#)), em $\text{J m}^{-3} \text{s}^{-1}$. Se o método escolhido foi pelo calor de reação h_r , ela retorna H_r''' em $\text{J m}^{-3} \text{s}^{-1}$ calculada pela função `solve(t0, deltaT)`.
- `RRpor(T)` - Calcula o termo fonte para a equação da evolução da porosidade ([Equação 4.3](#)). Essa função é invocada pela função `evolvePorosity()` da classe `volPyrolysis`.
- `calculate()` - Calcula os termos fontes de massa, R^{tot}''' , referentes a cada espécie, em $\text{kg m}^{-3} \text{s}^{-1}$. Para cada célula computacional, essa função utiliza a taxa de reação calculada na função `omega(c, T, p, updateC0)`, em kg s^{-1} , e divide pelo volume da célula. Ela não calcula os termos fontes de massa por meio da solução do sistema de EDOs da cinética química, e sim diretamente pela [Equações 4.18, 4.20, 4.23 e 4.24](#). É importante salientar que, apesar de estar presente no código, a função `calculate()` não é utilizada em nenhum momento. Essa função é substituída pela função `solve(t0, deltaT)` para o cálculo dos termos fontes de massa, que calcula pela solução dos sistema de EDOs.
- `solve(t0, deltaT)` - Calcula os termos fontes de massa, R^{tot}''' , referentes a cada espécie, em $\text{kg m}^{-3} \text{s}^{-1}$, por meio da solução do sistema de EDOs. Para cada célula computacional, essa função calcula inicialmente uma variável chamada `c[]`, que é um vetor que guarda a massa de cada espécie para aquela célula. Ela guarda o valor inicial da variável `c[]` para o determinado intervalo de tempo na variável `c0[]`. Esses valores iniciais (`c0[]`) juntamente com o sistema de EDOs da cinética química caracterizam um problema de

valor inicial. A função $solve(t0, deltaT)$ então invoca as classes responsáveis pela solução do sistema de EDOs (problema de valor inicial), obtendo a evolução da massa de cada espécie ao final do intervalo de tempo e armazenando na variável $c[]$. O próximo passo é calcular a diferença $c[] - c0[]$ e guardar o resultado na variável $dc[]$. Desse modo, foi obtida a variação da massa de cada espécie para aquele intervalo de tempo da simulação, ou seja, o consumo ou geração das espécies, em kg, devido às reações químicas. Por fim, a taxa $R^{tot''''}$ em $\text{kg m}^{-3} \text{s}^{-1}$ é calculada para cada espécie. Para uma espécie i em uma célula $cell$, a taxa $R^{tot''''}$ é calculada por:

$$R_{i,cell}^{tot''''} = \frac{dc_{i,cell}}{\Delta t \cdot V_{cell}} \quad (5.15)$$

Onde Δt é o intervalo de tempo e V_{cell} é o volume da célula. O último termo calculado pela função $solve(t0, deltaT)$ é o efeito energético total do calor das reações químicas, H_r''' , em $\text{J m}^{-3} \text{s}^{-1}$, para o caso onde o usuário escolheu o método do calor de reação h_r .

5.3.1 Calor de Reação

Durante os diversos testes realizados com o `biomassGasificationFoam`, em alguns momentos foram notadas inconsistências em relação ao calor de reação. Por exemplo, reações que deviam ser exotérmicas estavam na verdade consumindo energia, mostrando-se endotérmicas. Em outros casos havia uma liberação exagerada de energia, que levava a um aumento exorbitante na temperatura e na consequente divergência da simulação.

Fatos como esses levaram a uma análise mais detalhada do código referente à implementação do cálculo do calor de reação, no arquivo `ODESolidHeterogeneousChemistryModel.C`.

O primeiro detalhe encontrado foi no cálculo do calor de reação pelo método das entalpias de formação, realizado na função `Sh()`. Nesse método, o calor de reação deve ser calculado por um somatório negativo do produto entre a taxa de consumo ou produção das espécies e suas entalpias de formação:

$$H_r''' = - \sum_i R_i^{tot''''} h_{f_i} \quad (5.16)$$

Onde $R_i^{tot''''}$ é a taxa de consumo ou geração de massa total para cada espécie devido à todas as reações químicas (Equação 4.27):

O somatório deve ser negativo para garantir que uma reação exotérmica, por exemplo, entre com o sinal positivo no termo fonte do lado direito da equação de conservação da energia da fase sólida (Equação 4.11) e represente uma fonte de energia. Da mesma forma, garante que uma reação endotérmica entre com o sinal negativo do lado direito da equação e represente um sumidouro de energia. Essa implementação do somatório negativo pode ser vista em [Cunha](#)

(2010) e também na função $Sh()$ da classe *ODEChemistryModel*, que calcula o calor de reação para as reações homogêneas na fase gasosa, H_r^G (Equação 4.29).

A Equação 5.16 poderia ser dividida para sólidos e gases da seguinte forma:

$$H_r''' = - \sum_S R_S^{tot'''} h_{fS} - \sum_G R_G^{tot'''} h_{fG} \quad (5.17)$$

Ao avaliar a função $Sh()$ do arquivo *ODESolidHeterogeneousChemistryModel.C*, nota-se que o segundo termo da Equação 5.17 entra com um sinal de adição, e não de subtração. Assim sendo, uma modificação proposta para essa função é a troca desse sinal de adição pelo de subtração, conforme mostrado no trecho de código abaixo:

```

1
2 template<class CompType, class SolidThermo, class GasThermo>
3 Foam::tmp<Foam::volScalarField>
4 Foam::ODESolidHeterogeneousChemistryModel<CompType, SolidThermo, GasThermo>::Sh()
5   const
6   {
7     tmp<volScalarField> tSh
8     (
9       new volScalarField
10      (
11        IObject
12        (
13          "Sh",
14          this->mesh_.time().timeName(),
15          this->mesh_,
16          IObject::NO_READ,
17          IObject::AUTO_WRITE,
18          false
19        ),
20        this->mesh_,
21        dimensionedScalar("zero", dimEnergy/dimTime/dimVolume, 0.0),
22        zeroGradientFvPatchScalarField::typeName
23      )
24    );
25    if (this->chemistry_)
26    {
27      scalarField& Sh = tSh();
28
29      if (solidReactionEnergyFromEnthalpy_)
30      {
31        forall(Ys_, i)
32        {
33          forall(Sh, cellI)
34          {
35            scalar hf = solidThermo_[i].hf();
36            Sh[cellI] -= hf*RRs_[i][cellI];
37          }
38        }
39        forall(Sh, cellI)

```

```

40     {
41         forAll(pyrolysisGases_ , i)
42         {
43             scalar Hc = gasThermo_[i].Hc();
44             Sh[cellI] -= Hc*RRg_[i][cellI]; // Sinal trocado de
45         } // += para -=
46     }
47 }
48 else
49 {
50     forAll(Sh, cellI)
51     {
52         Sh[cellI] = shReactionHeat_[cellI];
53     }
54 }
55 }
56
57 return tSh;
58 }

```

Para validar se essa troca de sinais foi correta e se o cálculo da energia de reação pelo método das entalpias de formação agora responde corretamente, realizou-se duas simulações de teste. O domínio é um meio poroso bidimensional. O sólido é composto de 5% de carvão e 95% de matéria inerte. Ar preenche o meio poroso, composto por 76,7% de nitrogênio e 23,3% de oxigênio em base mássica. Esse ar é estacionário, apresentando velocidade nula. A temperatura inicial do sólido e do gás por todo o domínio foi definida como 600 K. Suas quatro fronteiras são adiabáticas, evitando troca de calor com o ambiente. Dessa forma, a evolução da temperatura do sólido será influenciada apenas pelo calor de reação. A única reação química definida é a oxidação do carvão, que é conhecidamente exotérmica. Se a temperatura do sólido aumentar, a reação está exotérmica, se ela diminuir a reação está endotérmica.

A primeira simulação foi com o código original, ou seja, com o termo do somatório dos gases entrando com sinal de adição. A evolução da temperatura do sólido durante 0,2 segundo de simulação está apresentada na [Figura 60](#). Pode-se notar que a temperatura do sólido cai durante o processo de oxidação do carvão, indicando que essa reação, que é exotérmica, está sendo calculada como endotérmica.

A segunda simulação foi com o código com a modificação proposta, ou seja, onde o termo do somatório dos gases entra com sinal de subtração. [Figura 61](#) mostra a evolução da temperatura do sólido no tempo. Dessa vez, essa temperatura aumenta durante o processo de oxidação do carvão, indicando que essa reação está sendo calculada como exotérmica.

Esses resultados mostram que a modificação proposta surtiu o efeito desejado e o calor de reação calculado pelas entalpias de formação agora corresponde às expectativas.

Um segundo detalhe encontrado no código do arquivo *ODESolidHeterogeneousChemistry-Model.C* foi no cálculo da energia de reação pelo método do calor de reação h_r (quando a variável *solidReactionEnergyFromEnthalpy* for declarada **false**). Como comentado na seção anterior, a função *omega(c, T, p, updateC0)* armazena na última posição do vetor *om[]* o somatório do

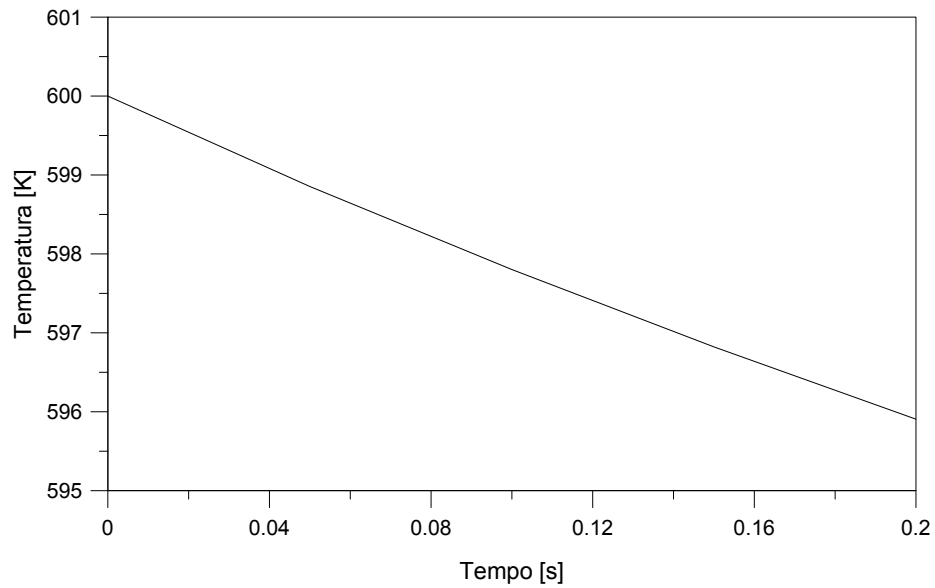


Figura 60 – Evolução da temperatura do sólido para o código original

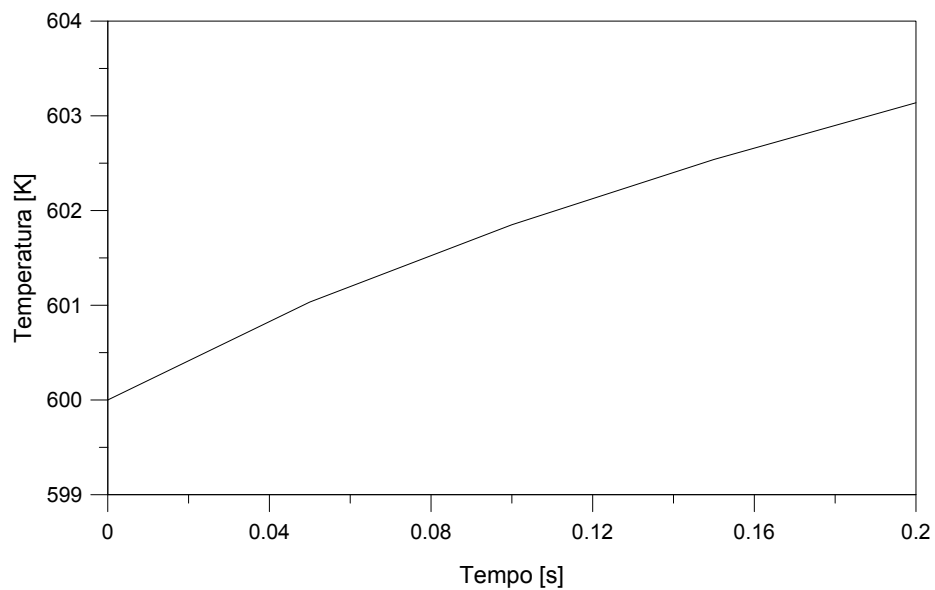


Figura 61 – Evolução da temperatura do sólido para o código com a modificação proposta

efeito energético de cada reação (Equação 5.13). A unidade de H_r é em J s^{-1} . A taxa de energia por unidade de volume, H_r''' , em $\text{J m}^{-3} \text{s}^{-1}$, é calculada na função $\text{solve}(t0, \text{delta}T)$. Por uma análise de unidades, para se obter H_r''' basta dividir H_r pelo volume. Porém, olhando o código implementado na função $\text{solve}(t0, \text{delta}T)$ é possível perceber que ele divide H_r pelo volume e também pelo intervalo de tempo. Propõe-se então uma modificação no código, retirando essa divisão pelo intervalo de tempo e deixando apenas a divisão pelo volume. O código abaixo é um trecho da função $\text{solve}(t0, \text{delta}T)$, que mostra a modificação proposta, onde a linha comentada era a implementação original. Nele, $\text{omegaPreq}[nEqns()]$ é H_r , $\text{delta}T$ é o intervalo de tempo e delta é o volume da célula.

```

1
2     deltaTMin = min(tauC, deltaTMin);
3     dc = c - c0;
4
5     forAll(RRs_, i)
6     {
7         RRs_[i][celli] = dc[i]/(deltaT*delta);
8     }
9
10    forAll(RRg_, i)
11    {
12        RRg_[i][celli] = dc[nSolids_ + i]/(deltaT*delta);
13    }
14
15    if (not solidReactionEnergyFromEnthalpy_)
16    {
17        //shReactionHeat_[celli] = omegaPreq[nEqns()]/(deltaT*delta);
18        shReactionHeat_[celli] = omegaPreq[nEqns()]/(delta);
19    }

```

Durante alguns testes com o código original, usar o calor de reação h_r fornecido na literatura para a oxidação de determinados sólidos resultou em valores exorbitantes de energia liberada a partir da reação, que causou um aumento explosivo da temperatura e a rápida divergência da simulação. Isso se deve ao fato do calor de reação h_r estar sendo desnecessariamente dividido pelo intervalo de tempo, que geralmente é um valor pequeno, muito inferior a 1.

Ao utilizar o código com a mudança proposta, os valores resultantes para a energia liberada a partir da reação foram coerentes.

5.3.2 Propostas de Alteração no Código para a Cinética Química

Nesta seção, a implementação dos códigos que lidam com a cinética química das reações na fase sólida foi analisada mais cuidadosamente. Algumas modificações foram propostas com o intuito de consertar alguns detalhes existentes e acrescentar outros novos.

Primeiramente, não foram propostas mudanças para as classes que calculam a taxa de reação k , comentadas na seção [subseção 4.8.1.1](#). Existem dois modelos possíveis:

- *irreversibleSolidArrheniusHeterogeneousReaction* - [Equação 4.14](#). Implementada na classe *solidArrheniusReactionRate*, contida no diretório */biomassGasificationMedia/thermophysicalModels/solid/reaction/reactionRate*.
- *irreversibleSolidEvaporationHeterogeneousReaction* - [Equação 4.16](#). Implementada na classe *solidEvaporationRate*, contida no diretório */biomassGasificationMedia/thermophysicalModels/solid/reaction/reactionRate*.

Essas classes demonstram implementar corretamente a taxa de reação k , e os seus modelos são de ampla aplicabilidade para os processos de conversão termoquímica.

5.3.2.1 Ordem da reação com relação ao gás

Como comentado na [seção 5.3](#), a função $\omega(R, c, T, p, pf, cf, lRef, pr, cr, rRef)$ da classe `ODESolidHeterogeneousChemistryModel` é a responsável pelo cálculo da taxa de reação Ω_r , para cada reação r envolvendo o substrato sólido s ([Equação 5.14](#)). Nessa equação, n_r é a ordem da reação em relação ao sólido, Y^G é a fração mássica do gás envolvido na reação heterogênea. Do jeito como está implementado o `biomassGasificationFoam`, o parâmetro δ^G é igual a um se existe um substrato gasoso, ou igual a zero se não existe. Isso implica que não é possível considerar na cinética química uma ordem da reação em relação ao gás. Essa ordem é sempre unitária quando o gás está presente na reação.

Essa desconsideração da ordem da reação em relação ao gás (δ^G) leva a uma limitação do modelo cinético do `biomassGasificationFoam`. Para contornar essa limitação, serão demonstradas mudanças nos códigos para implementar uma ordem da reação em relação ao gás arbitrária, que é escolhida pelo usuário, assim como a ordem em relação ao sólido (n_r).

Na [subseção 4.8.1.1](#) comentou-se que a construção das equações envolvendo a fase sólida é feita pela classe `solidHeterogeneousReaction`. A classe `IrreversibleSolidHeterogeneousReaction` é responsável pela obtenção dos parâmetros cinéticos de cada reação, como a taxa k , a ordem da reação em relação ao sólido n_r e o calor de reação h_r , se esse for definido pelo usuário. Ambas as classes estão presentes no diretório `/biomassGasificationMedia/thermophysicalModels/solid/reaction/reactions`. Assim sendo, essas são as duas classes que sofrerão mudanças para incluir a ordem de reação δ^G .

Em resumo, as mudanças implementadas na classe `IrreversibleSolidHeterogeneousReaction` foram a leitura de δ^G e a criação de uma variável para guardá-la e de uma função para acessá-la. Como essa classe é derivada da classe `solidHeterogeneousReaction`, por meio de herança, deve ser definida também uma função virtual de acesso na classe `solidHeterogeneousReaction`.

A seguir, as mudanças realizadas serão demonstradas nos trechos de código dos arquivos designados:

- `solidHeterogeneousReaction.H` - Acrescentada a declaração da função virtual de acesso a δ^G :

```

1
2 // solidHeterogeneousReactionrate coefficients
3
4 virtual scalar kf
5 (
6     const scalar T,
7     const scalar p,
8     const scalarField& c
9 ) const;
10
11 virtual scalar nReact() const;
12
13 virtual scalar nReactGas() const; // Acrescentada
14

```



```

15     virtual scalar heatReact() const;
16
17     //- Write
18     virtual void write(Ostream&) const;

```

- *solidHeterogeneousReaction.C* - Acrescentada a definição da função virtual de acesso a δ^G :

```

1
2 Foam::scalar Foam::solidHeterogeneousReaction::kf
3 (
4     const scalar T,
5     const scalar p,
6     const scalarField& c
7 ) const
8 {
9     return 0.0;
10 }
11
12
13 Foam::scalar Foam::solidHeterogeneousReaction::nReact() const
14 {
15     return 1.0;
16 }
17
18 Foam::scalar Foam::solidHeterogeneousReaction::nReactGas() const //Acrescentada
19 {
20     return 1.0;
21 }
22
23 Foam::scalar Foam::solidHeterogeneousReaction::heatReact() const
24 {
25     return 1.0;
26 }

```

- *IrreversibleSolidHeterogeneousReaction.H* - O primeiro passo é acrescentar a declaração da variável que irá guardar δ^G (*nReactGas_*):

```

1
2 template<class ReactionRate>
3 class IrreversibleSolidHeterogeneousReaction
4 :
5     public solidHeterogeneousReaction
6 {
7     // Private data
8
9     // Reaction rate
10    ReactionRate k_;
11
12    // Reaction order
13    scalar nReact_;
14
15    // Reaction order for gas component //Acrescentada

```

```

16     scalar nReactGas_;
17
18     // Reaction heat
19     scalar heatReact_;

```

O segundo passo é modificar o primeiro dos dois construtores da classe de forma a utilizar a ordem δ^G como parâmetro de entrada:

```

1
2 public:
3
4     //– Runtime type information
5     TypeName("irreversible");
6
7
8     // Constructors
9
10    //– Construct from components
11    IrreversibleSolidHeterogeneousReaction
12    (
13        const solidHeterogeneousReaction& reaction ,
14        const ReactionRate& reactionRate ,
15        const scalar nReact, //Acrescentada a virgula
16        const scalar nReactGas //Acrescentada essa linha
17    );
18
19
20    //– Construct from Istream
21    IrreversibleSolidHeterogeneousReaction
22    (
23        const PtrList<volScalarField>& gasPhaseGases ,
24        const speciesTable& components ,
25        Istream& is ,
26        const speciesTable& pyrolysisGases
27    );

```

O último passo é acrescentar a declaração da função virtual de acesso a δ^G ($nReactGas()$):

```

1
2     // Member Functions
3
4     // IrreversibleSolidHeterogeneousReaction rate coefficients
5
6     //– Forward rate constant
7     virtual scalar kf
8     (
9         const scalar T,
10        const scalar p,
11        const scalarField& c
12    ) const;
13
14
15    //– Reaction order
16    virtual scalar nReact() const;
17

```

```

18     // - Reaction order for gas component // Acrescentada
19     virtual scalar nReactGas() const;
20
21     // - Reaction order
22     virtual scalar heatReact() const;
23
24
25
26     // - Write
27     virtual void write(Ostream&) const;

```

- *IrreversibleSolidHeterogeneousReaction.C* - O primeiro passo para esse arquivo é alterar o primeiro construtor para ler δ^G como parâmetro de entrada e guardar na variável *nReactGas_*:

```

1
2 template<class ReactionRate>
3 Foam::IrreversibleSolidHeterogeneousReaction<ReactionRate>::
4     IrreversibleSolidHeterogeneousReaction
5 (
6     const solidHeterogeneousReaction& reaction ,
7     const ReactionRate& k,
8     const scalar nReact,
9     const scalar nReactGas // Acrescentada essa linha
10 )
11 :
12     solidHeterogeneousReaction(reaction),
13     k_(k),
14     nReact_(nReact),
15     nReactGas_(nReactGas), // Acrescentada essa linha
16     heatReact_(1.0)
17 {}

```

Esse primeiro construtor não é utilizado na implementação do *biomassGasificationFoam*. No entanto, a modificação já fica preparada se forem realizadas futuras implementações nesse construtor. O segundo passo é alterar o segundo construtor (que é o efetivamente utilizado) para ler δ^G e guardar na variável *nReactGas_*:

```

1
2 template<class ReactionRate>
3 Foam::IrreversibleSolidHeterogeneousReaction<ReactionRate>::
4     IrreversibleSolidHeterogeneousReaction
5 (
6     const PtrList<volScalarField>& gasPhaseGases ,
7     const speciesTable& components ,
8     Istream& is ,
9     const speciesTable& pyrolysisGases
10 )
11 :
12     solidHeterogeneousReaction(gasPhaseGases, components, is, pyrolysisGases),
13     k_(components, is),
14     nReact_(readScalar(is)),

```

```

14     nReactGas_(readScalar(is)), //Acrescentada essa linha
15     heatReact_(readScalar(is))
16 {
17     is.readEnd("solidArrheniusReactionRate(Istream&)");
18 }

```

Nesse construtor, δ^G será fornecido na declaração da reação química envolvendo a fase sólida, no dicionário *chemistryProperties*, juntamente com os outros parâmetros, comentados na [subseção 4.8.1.1](#). Dessa forma, δ^G será o quinto parâmetro da terceira linha. Como exemplo hipotético, tem-se a declaração da seguinte reação de oxidação do carvão:

```

irreversibleSolidArrheniusHeterogeneousReaction
char + 0.72 O2 = 0.56 CO + 0.44 CO2
(2.265e7 12476.55 300 1.34 1.65 19.47e6)

```

Para esse exemplo, $\delta^G = 1,65$. A terceira linha irá conter os parâmetros da reação na seguinte ordem: fator pré-exponencial (A), temperatura de ativação (T_a), temperatura crítica (T_c), ordem da reação com relação ao sólido (n_r), ordem da reação com relação ao gás (δ^G), e calor da reação (h_r).

O terceiro passo na modificação do arquivo *IrreversibleSolidHeterogeneousReaction.C* é a definição da função de acesso a δ^G (*nReactGas()*):

```

1
2 template<class ReactionRate>
3 Foam::scalar Foam::IrreversibleSolidHeterogeneousReaction<ReactionRate>::nReact()
4     const
5 {
6     return nReact_;
7 }
8 template<class ReactionRate> //Acrescentada
9 Foam::scalar Foam::IrreversibleSolidHeterogeneousReaction<ReactionRate>::
10     nReactGas() const
11 {
12     return nReactGas_;
13 }
14 template<class ReactionRate>
15 Foam::scalar Foam::IrreversibleSolidHeterogeneousReaction<ReactionRate>::
16     heatReact() const
17 {
18     return heatReact_;
19 }

```

A última modificação nesse arquivo se refere à função *write(os)*, que escreve na tela do terminal as informações sobre todas as reações químicas na fase sólida. Essa função será ampliada de forma a também escrever na tela o parâmetro δ^G :

```

1
2 template<class ReactionRate>
3 void Foam::IrreversibleSolidHeterogeneousReaction <ReactionRate >::write
4 (
5     Ostream& os
6 ) const
7 {
8     solidHeterogeneousReaction::write(os);
9     os << token::SPACE << "Reaction order for solid: " << nReact_ << nl << token
10    ::SPACE << "    Reaction order for gas: " << nReactGas_ << nl << k_;

```

As modificações mostradas acima tornam possível a definição de uma ordem da reação em relação ao gás (δ^G) introduzida pelo usuário. Esse parâmetro pode agora ser utilizado em outras classes do modelo termoquímico.

5.3.2.2 ODESolidHeterogeneousChemistryModel

Uma análise cuidadosa da classe *ODESolidHeterogeneousChemistryModel* torna possível detectar detalhes no seu código que demandam uma atenção. Assim, nesta seção, mudanças foram propostas no arquivo fonte *ODESolidHeterogeneousChemistryModel.C*. Os códigos das funções modificadas estão presentes no Apêndice A.

O primeiro detalhe é que, aos olhos do autor deste trabalho, a variável $Ys0_$ é redundante. Ela possui o mesmo papel da variável $c[]$, criada na função *solve(t0, deltaT)*. Portanto, uma primeira modificação seria a retirada dessa variável por todo o código, substituindo-a por $c[]$ onde ela for utilizada. Isso também economizaria custo computacional, porque evita os cálculos para atualizar $Ys0_$.

A função *omega(c, T, p, updateC0)*, que realiza o cálculo das taxas totais de consumo ou geração para cada espécie, R_i^{tot} , em kg s^{-1} , ficaria conforme apresentada no Apêndice A.1, onde pode-se perceber que os pontos relacionados a $Ys0_$ foram comentados do código.

A função *omega(R, c, T, p, pf, cf, lRef, pr, cr, rRef)* é responsável pelo cálculo da taxa de reação Ω_r , conforme mostrado na Equação 5.14. Porém, uma análise do seu algoritmo indica que essa função não está utilizando o expoente relacionado à ordem da reação em relação do sólido (n_r). Para uma reação heterogênea, onde há um gás como reagente, a taxa Ω_r está sendo calculada como:

$$\Omega_r = \rho^S (Y_s^S) (Y^G) k_r V \quad (5.18)$$

Para uma reação homogênea na fase sólida, ou seja, com apenas um sólido como substrato, a taxa Ω_r está sendo calculada como:

$$\Omega_r = \rho^S (Y_s^S) k_r V \quad (5.19)$$

Esse detalhe só não causa influência na simulação se a ordem da reação, n_r , for unitária. Na modificação proposta será implementada tanto a ordem da reação com relação ao sólido (n_r) quanto a ordem com relação ao gás (δ^G).

O cálculo da taxa de reação Ω_r deve ser modificado para incluir as ordens de reação citadas acima e utilizar a variável `c []` ao invés da `Ys0_`. A variável `c []` é fornecida como parâmetro de entrada para a função `omega(R, c, T, p, pf, cf, lRef, pr, cr, rRef)`, e é calculada na função `solve(t0, deltaT)` para o sólido i na célula `cell` da seguinte maneira:

$$c_{i,cell} = \rho_{cell}^S \left(Y_{i,cell}^S \right) V_{cell} \quad (5.20)$$

A ordem da reação n_r não deve ser aplicada como expoente para a variável `c []`, e sim apenas para a fração mássica do componente do sólido em questão, Y_s^S . Assim sendo, a mudança proposta no código irá calcular Ω_r como:

$$\Omega_r = c_s \left(Y_s^S \right)^{n_r-1} \left(Y^G \right)^{\delta^G} k_r \quad (5.21)$$

Assim, a taxa Ω_r corresponderá à [Equação 5.14](#).

A função `omega(R, c, T, p, pf, cf, lRef, pr, cr, rRef)` com as mudanças propostas está apresentada no Apêndice [A.2](#).

A próxima função alterada é a `solve(t0, deltaT)`, apresentada no Apêndice [A.3](#). As mudanças são poucas, e devidas à retirada da variável `Ys0_`. Uma delas é o reposicionamento da linha:

```
1 scalarField omegaPreq(omega(c0,Ti,0.0));
```

Que, no código do Apêndice [A.3](#), é trocada da linha 70 para a linha 81. A outra mudança é a retirada do trecho de código no final da função, que comanda a atualização da variável `Ys0_`. Por fim, é possível notar a mudança com relação ao cálculo da energia de reação para o método do calor de reação h_r , comentada na [subseção 5.3.1](#), retirando a divisão pelo intervalo de tempo e deixando apenas a divisão pelo volume.

A última função modificada é a `jacobian(t, c, dcdt, dfdc)`, apresentada no Apêndice [A.4](#). Ela realiza a criação da Matriz Jacobiana, utilizada para fazer a linearização do sistema não linear de EDOs da cinética química. Portanto, essa função é invocada pelas classes que solucionam o sistema de EDOS.

Conforme comentado na [seção 5.3](#), a função `omega(c, T, p, updateC0)` realiza o cálculo das taxas de conversão de massa totais para cada espécie i , R_i^{tot} (em kg s^{-1}), somando as taxas R_i de todas as reações, descritas nas Equações [4.18](#), [4.20](#), [4.23](#) e [4.24](#). O sistema cinético de EDOs pode ser escrito em forma matricial como:

$$\frac{dc}{dt} = \mathbf{R}^{tot} \quad (5.22)$$

Onde \mathbf{c} é o vetor contendo a massa de cada componente i (c_i), e \mathbf{R}^{tot} é o vetor das taxas R_i^{tot} .

De acordo com as informações expostas em [Turányi e Tomlin \(2014\)](#), o sistema cinético de EDOs juntamente com os valores iniciais ($c_{i,0}$) formam o seguinte problema de valor inicial:

$$\frac{d\mathbf{c}}{dt} = \mathbf{f}(\mathbf{c}, \mathbf{k}), \quad \mathbf{c}(t_0) = \mathbf{c}_0 \quad (5.23)$$

O termo \mathbf{f} representa o termo de geração ou consumo de massa \mathbf{R}^{tot} , que é um função das massas das espécies c_i (\mathbf{c}) e das taxas de reação k_r (\mathbf{k}). A Matriz Jacobiana para esse sistema é definida como:

$$\mathbf{J} = \frac{\partial \mathbf{f}(\mathbf{c}, \mathbf{k})}{\partial \mathbf{c}} \quad (5.24)$$

Os métodos de solução de EDOs implementados no OpenFOAM consideram também como variáveis do sistema a temperatura (T) e a pressão (p). No entanto, as modificações propostas no código serão apenas para as variações da taxa de conversão de massa das espécies em relação à massa das espécies, definida na [Equação 5.24](#), ou seja, para os termos $\partial R_i^{tot} / \partial c_j$.

Exemplos do cálculo da Matriz Jacobiana para sistemas de EDOs provenientes da cinética química (conforme a [Equação 5.24](#)) podem ser encontrados em [Turányi e Tomlin \(2014\)](#). Exemplos do cálculo da Matriz Jacobiana no OpenFOAM podem ser vistos em [Zongyuan \(2009\)](#).

Uma análise da função *jacobian*($t, c, dcdt, dfdc$) revela que os termos $\partial R_i^{tot} / \partial c_j$ precisam ser ajustados, pois as taxas R_i^{tot} não estão sendo calculadas corretamente, conforme as [Equações 5.14, 4.18, 4.20, 4.23 e 4.24](#) (para cada reação r). Várias modificações são implementadas para consertar o cálculo dos termos $\partial R_i^{tot} / \partial c_j$, como vistas no [Apêndice A.4](#).

Pode-se perceber também que a Matriz Jacobiana é preenchida de forma incompleta. Os únicos dos termos $\partial R_i^{tot} / \partial c_j$ efetivamente preenchidos são para a variação da massa dos sólidos em relação à massa dos próprios sólidos. Assim, considerando a notação do termo $\partial R_i^{tot} / \partial c_j$, os únicos preenchidos ocorrem para:

- $i = \text{sólido}; j = \text{sólido}$

Os termos não são preenchidos para:

- $i = \text{sólido}; j = \text{gás}$
- $i = \text{gás}; j = \text{sólido}$
- $i = \text{gás}; j = \text{gás}$

Em virtude do referido problema, alguns *loops* foram incluídos na função *jacobian*($t, c, dcdt, dfdc$) de forma a preencher completamente os termos $\partial R_i^{tot} / \partial c_j$.

5.3.3 Função $solve(t0, deltaT)$ e Função $calculate()$

Na seção [seção 5.3](#), foram fornecidas informações sobre a função $solve(t0, deltaT)$ e sobre a função $calculate()$, ambas da classe *ODESolidHeterogeneousChemistryModel*. O objetivo dessas funções é o mesmo: calcular os termos fontes de massa, R^{tot} , referentes a cada espécie, em $\text{kg m}^{-3} \text{s}^{-1}$. A diferença está no modo como é feito esse cálculo.

A função $calculate()$ utiliza a taxa de conversão de massa R^{tot} calculada na função $omega(c, T, p, updateC0)$, em kg s^{-1} , e divide pelo volume da célula computacional. Isso significa que, como termo fonte para a equação de conservação das espécies para a fase sólida ([Equação 4.6](#)), a função $calculate()$ insere diretamente a expressão para a variação da determinada espécie no tempo, dada pela cinética química. Para ilustrar essa afirmação, será utilizada a reação de oxidação do carbono:



Aplicando as [Equações 5.14](#) e [4.18](#), a expressão para a variação da massa de carbono (c_C) no tempo, em kg s^{-1} , será dada por:

$$\frac{dc_C}{dt} = R_C^{tot} = -\rho^S (Y_C^S)^{n_r} (Y_{O_2}^G)^{\delta_G} k_r V \quad (5.26)$$

Como comentado anteriormente, a função $calculate()$ divide o resultado pelo volume da célula (V) e insere diretamente como termo fonte na equação de conservação das espécies para a fase sólida. Assim, a equação da conservação do carbono será, a partir da [Equação 4.6](#):

$$\frac{\partial \rho^S Y_C^S}{\partial t} = -\rho^S (Y_C^S)^{n_r} (Y_{O_2}^G)^{\delta_G} k_r \quad (5.27)$$

A função $solve(t0, deltaT)$ utiliza a solução do sistema de EDOs da cinética química para calcular a variação da massa das espécies durante o intervalo de tempo. Para o exemplo fornecido da oxidação do carbono ([Equação 5.25](#)), ao se aplicar as [Equações 5.14](#), [4.18](#), [4.23](#) e [4.24](#), temos que o sistema de EDOs correspondente será:

$$\frac{dc_C}{dt} = R_C^{tot} = -\rho^S (Y_C^S)^{n_r} (Y_{O_2}^G)^{\delta_G} k_r V \quad (5.28a)$$

$$\frac{dc_{O_2}}{dt} = R_{O_2}^{tot} = -\rho^S (Y_C^S)^{n_r} (Y_{O_2}^G)^{\delta_G} k_r V \frac{W_{O_2}}{W_{CO_2} - W_{O_2}} \quad (5.28b)$$

$$\frac{dc_{CO_2}}{dt} = R_{CO_2}^{tot} = \rho^S (Y_C^S)^{n_r} (Y_{O_2}^G)^{\delta_G} k_r V \frac{W_{CO_2}}{W_{CO_2} - W_{O_2}} \quad (5.28c)$$

Primeiramente, a função $solve(t0, deltaT)$ calcula a massa de carbono inicial ($c_{C,0}$) daquele intervalo de tempo. Em seguida, invoca a solução do sistema de EDOs ([Equações 5.28](#)), obtendo

a evolução da massa de carbono, assim como das massas de O_2 e CO_2 , ao final do intervalo de tempo. Fazendo a diferença entre a massa resultante (c_C) e a massa inicial ($c_{C,0}$), obtêm-se a variação da massa para aquele intervalo de tempo. Por fim, a função `solve(t0, deltaT)` divide essa variação da massa pelo intervalo de tempo e pelo volume da célula, de forma a obter a geração ou consumo de massa em $kg\ m^{-3}\ s^{-1}$. Para esse caso, a equação da conservação do carbono será, a partir da [Equação 4.6](#):

$$\frac{\partial \rho^S Y_C^S}{\partial t} = \frac{c_C - c_{C,0}}{\Delta t \cdot V} \quad (5.29)$$

Segundo [Turányi e Tomlin \(2014\)](#), de um ponto de vista matemático, o sistema cinético de EDOs é de primeira ordem e geralmente não linear, visto que contém derivadas de primeira ordem em relação ao tempo e essa derivada é geralmente uma função não linear das concentrações (ou frações mássicas). O número de equações é igual ao número de espécies no mecanismo de reação. Essas equações são acopladas e devem ser solucionadas simultaneamente. Para um mecanismo de reação com várias reações, suas taxas podem ser bem diferentes entre si, em até várias ordens de magnitude, o que caracteriza o sistema de EDOs da cinética química como um sistema rígido.

Essas considerações implicam que, para um mecanismo de reação mais complexo, com várias reações, as expressões para a variação da massa das espécies no tempo são acopladas e formam um sistema rígido. Assim, o método utilizado pela função `calculate()` de inserir essas expressões diretamente como termo fonte nas equações de conservação das espécies pode levar a instabilidades numéricas e à consequente divergência da simulação. Nesse caso, resolver o sistema de EDOs com a função `solve(t0, deltaT)` se mostra uma melhor opção, trazendo mais estabilidade.

Porém, surge a indagação se, para mecanismos de reação simples com sistemas menos rígidos, a função `calculate()` poderia ser utilizada sem problemas de instabilidade numérica. Teoricamente, isso também reduziria o custo computacional por não realizar a solução de um sistema de EDOs.

Apesar de estar presente no código, a função `calculate()` não é utilizada pelo `biomassGasificationFoam`. Apenas a função `solve(t0, deltaT)` é utilizada para calcular os termos fontes de massa, $R^{tot''''}$. Os pontos comentados acima despertam o interesse em testar as implicações devidas ao uso da função `calculate()`. Modificações foram propostas no código para que ela possa ser utilizada, e o seu uso foi incluso nos teste de cinética química da próxima seção.

Os códigos modificados estão apresentados no Apêndice B. A primeira alteração é na própria função `calculate()`, onde são inclusos os trechos referentes ao cálculo do efeito energético total do calor da reações químicas, H_r'''' , em $J\ m^{-3}\ s^{-1}$, para o caso onde o usuário escolheu o método do calor de reação h_r . Esses trechos foram baseados no código implementado originalmente na função `solve(t0, deltaT)`. O código da função `calculate()` com as modificações propostas está apresentado no Apêndice B.1, com os trechos inclusos destacados.

A segunda alteração será na função `evolveRegion()` da classe `volPyrolysis`, que coordena a evolução da equações e termos fontes referentes à fase sólida. Uma de suas tarefas é invocar a

função $solve(t0, deltaT)$ da classe *ODESolidHeterogeneousChemistryModel* para realizar o cálculo dos termos fontes de massa, $R^{tot''''}$. Assim sendo, uma modificação é proposta para que a função invocada seja a $calculate()$. O código da função $evolveRegion()$ com as modificações propostas está apresentado no Apêndice B.2, com os trechos modificados destacados.

5.3.4 Testes para a Cinética Química

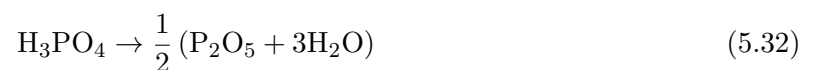
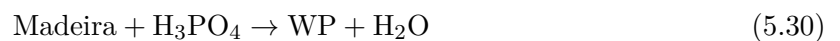
Nesta seção, foram realizados testes para avaliar a implementação da cinética química das reações na fase sólida. O primeiro grupo de testes foram para um processo de pirólise de um material sólido. O segundo grupo de testes foram para um processo de combustão de um material sólido.

Deste ponto em diante, o código original fornecido com o `biomassGasificaionFoam` será referenciado como Código Original. O código com as alterações para a cinética química propostas na subseção 5.3.2 será referenciado como Código Alterado. E por fim, o código com as alterações para a utilização da função $calculate()$ propostas na subseção 5.3.3 será referenciado como Código Calculate. Os testes aqui realizados comparam os resultados obtidos pelos três Códigos.

As simulações foram adaptadas do caso teste *oneCellTest*, fornecido com o `biomassGasificationFoam`. Segundo Kwiatkowski et al. (2013), esse caso é preparado para testar e analisar a cinética dos processos. A geometria é reduzida a uma única célula computacional. Nesse caso, o foco será nas reações química e nos mecanismos dos processos térmicos, ao invés da transferência de calor. Assim, os termos fontes envolvendo os efeitos da radiação e a transferência de calor por condução no meio poroso serão desconsiderados.

5.3.4.1 Pirólise de um material sólido

Para os testes da pirólise de um material sólido foi considerado o mecanismo de reação proposto por Hared et al. (2007), desenvolvido para a pirólise de uma madeira impregnada com ácido fosfórico para a produção de carvão ativado. Esse mecanismo é composto pelas reações:



Os parâmetros cinéticos dados por [Hared et al. \(2007\)](#) para esse mecanismo estão apresentados na [Tabela 8](#).

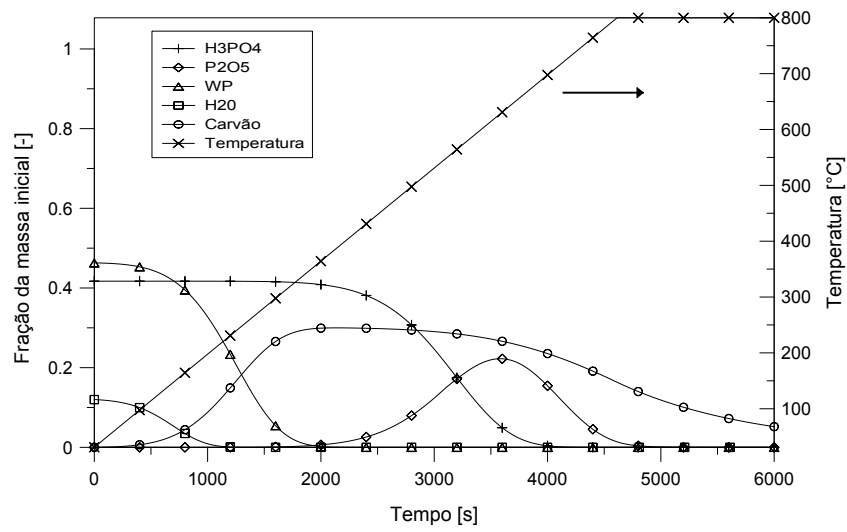
Tabela 8 – Parâmetros cinéticos para a pirólise de uma madeira impregnada com ácido fosfórico

Reação	Parâmetro	Valor
Equação (5.31)	A_2 (s^{-1})	236×10^{-1}
	E_2 ($kJ \text{ mol}^{-1}$)	30,8
Equação (5.32)	A_3 (s^{-1})	614×10^{-1}
	E_3 ($kJ \text{ mol}^{-1}$)	71,5
Equação (5.33)	A_4 (s^{-1})	6673×10^{-1}
	E_4 ($kJ \text{ mol}^{-1}$)	102,5
Equação (5.34)	A_5 (s^{-1})	553×10^{-2}
	E_5 ($kJ \text{ mol}^{-1}$)	32,8
Equação (5.35)	A_6 (s^{-1})	824×10^{-3}
	E_6 ($kJ \text{ mol}^{-1}$)	61,5
	α	0,65

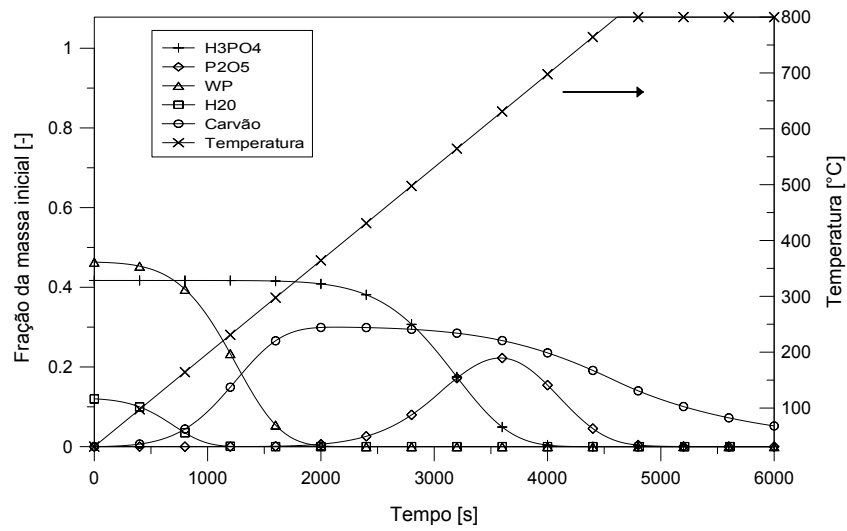
Em um dos seus experimentos, [Hared et al. \(2007\)](#) utilizou uma taxa de aquecimento de $10 \text{ }^\circ\text{C}/\text{min}$, que foi inserida como condição de contorno para a temperatura do sólido. O sólido inicia a uma temperatura de $31 \text{ }^\circ\text{C}$, aquece a uma taxa de $10 \text{ }^\circ\text{C}/\text{min}$ até atingir $800 \text{ }^\circ\text{C}$, temperatura que se mantém constante a partir de então. As frações mássicas iniciais dos componentes do sólido são: $Y_{\text{H}_3\text{PO}_4} = 0,417$; $Y_{\text{H}_2\text{O}} = 0,12$; $Y_{\text{WP}} = 0,463$.

Três simulações foram realizadas para o caso exposto, sendo uma com cada Código (Original, Alterado e Calculate), de forma a comparar seus resultados. A [Figura 62](#) mostra os gráficos obtidos para a evolução das frações mássicas pelos três Códigos diferentes. Pode-se observar que os três resultados são praticamente iguais. Isso implica que, para esse caso, as alterações no código para a cinética química não causaram diferença significativa no resultado da simulação. A diferença na fração mássica se encontra na casa de 10^{-5} .

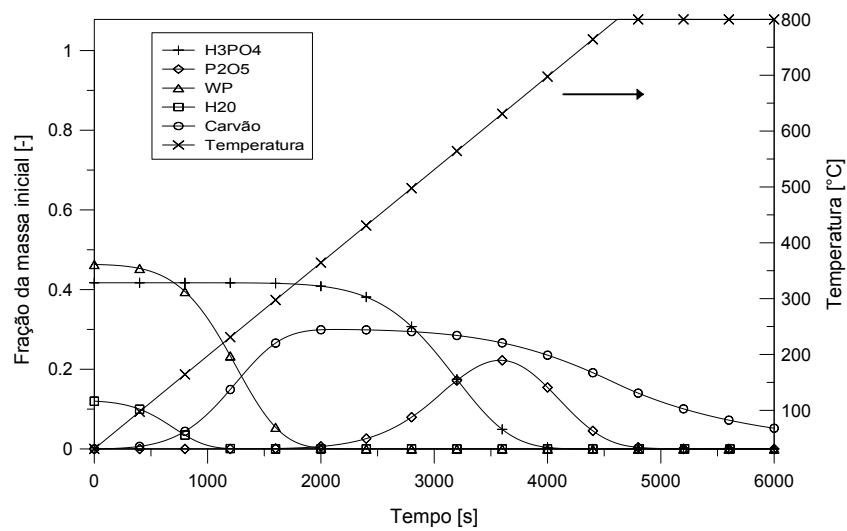
O mecanismo de reações para a pirólise de uma madeira proposto por [Hared et al. \(2007\)](#) não apresenta reações heterogêneas, apenas homogêneas na fase gasosa e na fase sólida (como desumidificação e decomposição térmica do sólido). Portanto, não possui nenhuma ordem de reação com relação ao gás (δ^G). As ordens de reação com relação ao sólido (n_r) são unitárias. Esses fatos caracterizam uma situação na qual o Código Original implementa satisfatoriamente os cálculos das taxas de reação e de conversão de massa, e as mudanças realizadas no Código Alterado com relação a esses parâmetros não surtiria efeito. Porém, era de se esperar que a alteração no cálculo da Matriz Jacobiana provocasse alguma influência na solução do sistema de EDOs, causando uma diferença nos resultados encontrados. Para esse caso, o impacto dessa modificação não foi muito significativo, e os resultados foram muito próximos.



(a) Código Original

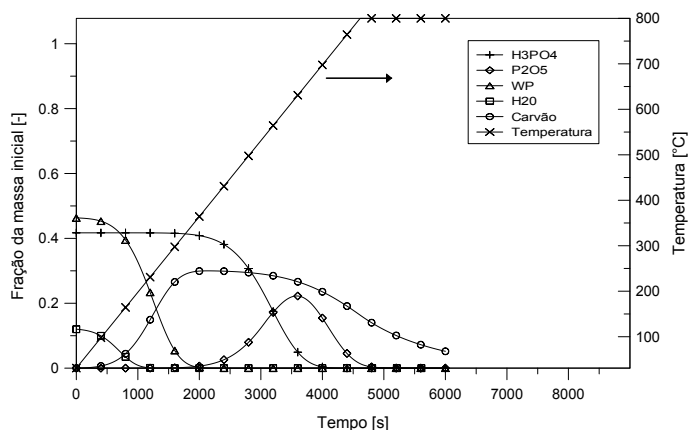


(b) Código Alterado

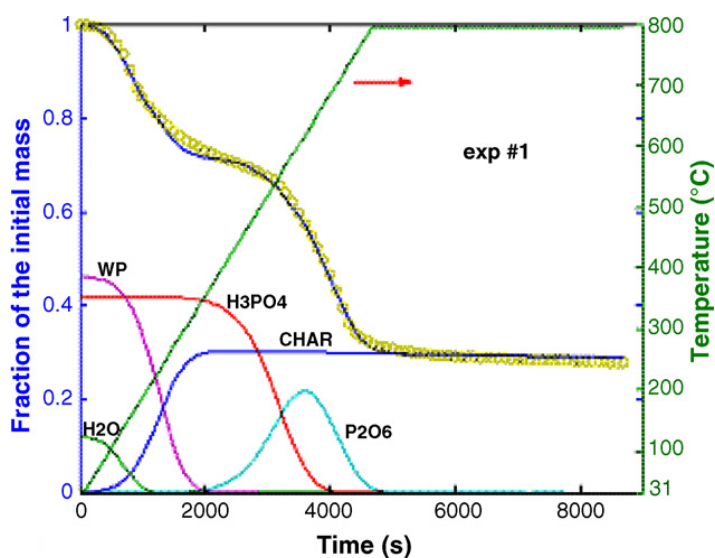


(c) Código Calculate

Figura 62 – Evolução das frações mássicas para os três Códigos diferentes



(a) Código Original



(b) Resultado de Hared et al. (2007)

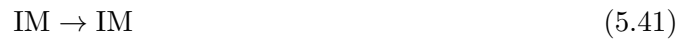
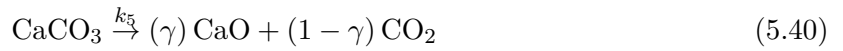
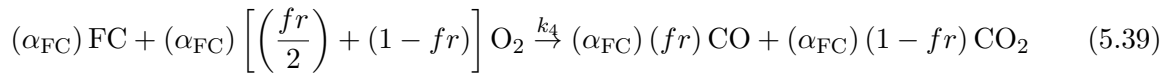
Figura 63 – Comparação da evolução das frações mássicas para o Código Original e o encontrado por Hared et al. (2007) para uma taxa de aquecimento de $10\text{ }^{\circ}\text{C}/\text{min}$

O Código Calculate apresentou resultados quase idênticos aos demais, sem instabilidades numéricas. O mecanismo de reações químicas é simples, e a função *calculate()* se mostrou viável para esse caso. Porém, não notou-se uma diferença no tempo de simulação, que era esperado.

Para confirmar que as características desse mecanismo de reações proporcionam uma situação para a qual o Código Original produz resultados satisfatórios, na Figura 63 é apresentada a comparação entre os resultados obtidos pelo Código Original e os resultados dados por Hared et al. (2007), para uma taxa de aquecimento de $10\text{ }^{\circ}\text{C}/\text{min}$. Os dois gráficos demonstram ser bem semelhantes, com exceção da fração mássica do carvão na região correspondente à reação de decomposição do carvão em gases leves (Equação 5.35). Uma possível causa da divergência de apenas esse ponto é um conjunto de parâmetros cinéticos dessa reação pouco representativos.

5.3.4.2 Combustão de um material sólido

Para os testes da combustão de um material sólido foi considerado o mecanismo de reação proposto por [Zanoni, Massard e Martins \(2012\)](#), desenvolvido para a combustão do xisto betuminoso. Esse mecanismo é composto pelas reações:



A [Equação 5.36](#) é a equação geral da decomposição do xisto betuminoso (OS). Juntamente com a equação da matéria inerte (IM) ([Equação 5.41](#)), não participam do mecanismo cinético. Assim sendo, a combustão do xisto betuminoso é um mecanismo de quatro etapas, [Equações 5.37](#) à [5.40](#). Nesse mecanismo, OM é a matéria orgânica do xisto betuminoso. Sua decomposição libera óleo (betume), CO, CO₂ e hidrocarbonetos leves (HC), deixando na matriz sólida o carbono fixo (FC).

Segundo [Zanoni, Massard e Martins \(2012\)](#), as equações para as taxas de consumo ou produção das espécies resultará no seguinte sistema de EDOs:

$$\frac{dY_{\text{H}_2\text{O}}}{dt} = -k_2 (Y_{\text{H}_2\text{O}})^{n_2} \quad (5.42)$$

$$\frac{dY_{\text{OM}}}{dt} = -k_3 (Y_{\text{OM}})^{n_3} \quad (5.43)$$

$$\frac{dY_{\text{Oil}}}{dt} = (\alpha_{\text{Oil}}) k_3 (Y_{\text{OM}})^{n_3} \quad (5.44)$$

$$\frac{dY_{\text{HC}}}{dt} = (\alpha_{\text{HC}}) k_3 (Y_{\text{OM}})^{n_3} \quad (5.45)$$

$$\frac{dY_{FC}}{dt} = (\alpha_{FC}) k_3 (Y_{OM})^{n_3} - (\alpha_{FC}) k_4 P_{O_2} (Y_{FC})^{n_4} (Y_{O_2})^{n_5} \quad (5.46)$$

$$\frac{dY_{O_2}}{dt} = -(\alpha_{FC}) \left[\left(\frac{fr}{2} \right) + (1 - fr) \right] k_4 P_{O_2} (Y_{FC})^{n_4} (Y_{O_2})^{n_5} \quad (5.47)$$

$$\frac{dY_{CO}}{dt} = (\alpha_{CO}) k_3 (Y_{OM})^{n_3} + (\alpha_{FC}) (fr) k_4 P_{O_2} (Y_{FC})^{n_4} (Y_{O_2})^{n_5} \quad (5.48)$$

$$\begin{aligned} \frac{dY_{CO_2}}{dt} = & (\alpha_{CO_2}) k_3 (Y_{OM})^{n_3} + (\alpha_{FC}) (1 - fr) k_4 P_{O_2} (Y_{FC})^{n_4} (Y_{O_2})^{n_5} \\ & + (1 - \gamma) k_5 (Y_{CaCO_3})^{n_6} \end{aligned} \quad (5.49)$$

$$\frac{dY_{CaCO_3}}{dt} = -k_5 (Y_{CaCO_3})^{n_6} \quad (5.50)$$

$$\frac{dY_{CaO}}{dt} = (\gamma) k_5 (Y_{CaCO_3})^{n_6} \quad (5.51)$$

O termo P_{O_2} refere-se à pressão parcial do oxigênio. Para a simulação com o `biomassGasificationFoam`, esse parâmetro será considerado constante e incluído no fator pré-exponencial A_4 , utilizado no cálculo da taxa de reação k_4 .

A composição inicial do xisto betuminoso e os coeficientes estequiométricos das reações químicas estão apresentados na [Tabela 9](#).

Tabela 9 – Composição inicial e coeficientes estequiométricos para a combustão do xisto betuminoso

Variável	Valor
Y_{H_2O}	1,25 %
Y_{OM}	19,7 %
Y_{CaCO_3}	34,6 %
Y_{IM}	44,45 %
α_{Oil}	0,53
α_{CO}	0,3641
α_{CO_2}	1,1475
α_{HC}	10,428
α_{FC}	0,242
fr	0,07
γ	0,53
P_{O_2}	7,5 kPa

Os parâmetros cinéticos para a combustão do xisto betuminoso foram estimados por [Zanoni, Massard e Martins \(2012\)](#) por meio de um experimento de análise termogravimétrica. Para uma taxa de aquecimento de 3 K/min, esses parâmetros estão apresentados na [Tabela 10](#).

Tabela 10 – Parâmetros cinéticos para a combustão do xisto betuminoso obtidos com uma taxa de aquecimento de 3 K/min

Reação	Parâmetro	Valor
Desumidificação Equação (5.37)	A_2 (s^{-1})	$1,09 \times 10^{10}$
	E_2 ($kJ\ mol^{-1}$)	67,8
Pirólise Equação (5.38)	n_2 (-)	2,29
	A_3 (s^{-1})	$2,64 \times 10^2$
	E_3 ($kJ\ mol^{-1}$)	65,38
Combustão Equação (5.39)	n_3 (-)	1,28
	A_4 (s^{-1})	$7,41 \times 10^6$
	E_4 ($kJ\ mol^{-1}$)	102,51
	n_4 (-)	1,32
Descarbonatação Equação (5.40)	n_5 (-)	1,96
	A_5 (s^{-1})	$2,67 \times 10^6$
	E_5 ($kJ\ mol^{-1}$)	162,41
	n_6 (-)	1,07

A primeira simulação para a cinética da combustão do xisto betuminoso utilizou o Código Original. A [Figura 64](#) mostra uma comparação entre os resultados obtidos pelo Código Original e os resultados obtidos por [Zanoni, Massard e Martins \(2012\)](#). Os resultados para as reações de decomposição da matéria orgânica e descarbonatação encontrados pelo Código Original apresentam um comportamento semelhante ao encontrado por [Zanoni, Massard e Martins \(2012\)](#), com valores parecidos para as frações mássicas finais de CaO e Óleo. O que diferencia um pouco nessas reações é que a transformação ocorre ligeiramente antes para o Código Original. A grande diferença entre os resultados obtidos ocorre para as reações de oxidação e de desumidificação. O Código Original encontra taxas extremamente elevadas para essas reações. De fato, a taxa de consumo do carbono fixo é tão rápida que a sua fração mássica não aparece no gráfico, pois tudo que é formado na decomposição da matéria orgânica é instantaneamente consumido na oxidação. A reação de oxidação é heterogênea e a reação de desumidificação possui uma ordem de reação bem elevada, e foi justamente para essas reações que o Código Original não obteve bons resultados, justificando as mudanças propostas no código na [subseção 5.3.2](#), como no cálculo da Matriz Jacobiana, da taxa de consumo e produção das espécies e na inclusão da ordem de reação em relação ao gás.

Com o intuito de verificar a influência das modificações feitas para a cinética química no Código Alterado, foi realizada uma segunda simulação com o uso desse Código, mantendo os mesmo parâmetros. A [Figura 65](#) mostra a comparação entre os resultados obtidos na simulação e

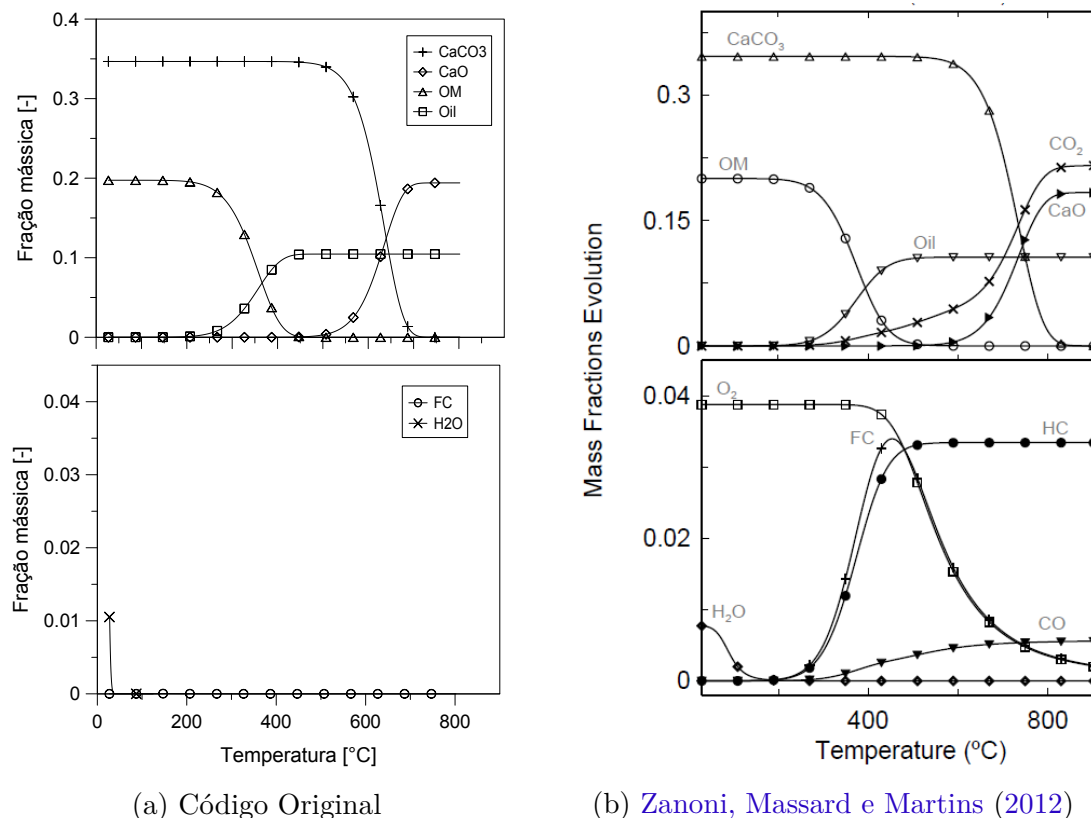


Figura 64 – Comparação da evolução das frações mássicas encontradas pelo Código Original e por Zanoni, Massard e Martins (2012)

os obtidos por Zanoni, Massard e Martins (2012). Os resultados para as reações de decomposição da matéria orgânica e descarbonatação encontrados pelo Código Alterado também apresentam um comportamento semelhante ao encontrado por Zanoni, Massard e Martins (2012). As alterações realizadas no código tiveram um efeito significativo para a reação de desumidificação, apresentando uma curva mais coerente. A taxa de consumo do carbono fixo na reação de oxidação ainda se mostrou bastante elevada.

Como visto nos gráficos apresentados, a fração mássica de carbono fixo (FC) atinge valores muito pequenos. Mais três simulações foram realizadas para comparar os resultados fornecidos pelos três Códigos (Original, Alterado e Calculate) entre si. Nessas simulações, o valor do fator pré-exponencial A_4 para a reação de oxidação (Equação 5.39) será diminuído para um valor de $A_4 = 3,488 \times 10^3$, com a intenção puramente de melhorar a visibilidade da curva de fração mássica do carbono fixo, facilitando a comparação dos resultados. A Figura 66 apresenta os gráficos obtidos pelos três Códigos diferentes. Comparando os resultados obtidos pelo Código Original e pelo Código Alterado, pode-se perceber que as alterações propostas causaram uma diferença sutil na taxa de reação para a decomposição da matéria orgânica e para a descarbonatação. Porém, uma diferença bastante significativa ocorreu para as reações de oxidação do carbono fixo e de desumidificação, onde o Código alterado obteve taxas de reação bem menores. Na terceira simulação (Figura 66c) a alteração do Código Calculate foi incluída no Código Alterado, de forma a avaliar a influência do uso da função *calculate()* no cálculo das taxas de consumo e produção das espécies. Não houve diferenças nos resultados, mostrando que, nesse caso, a influência do uso

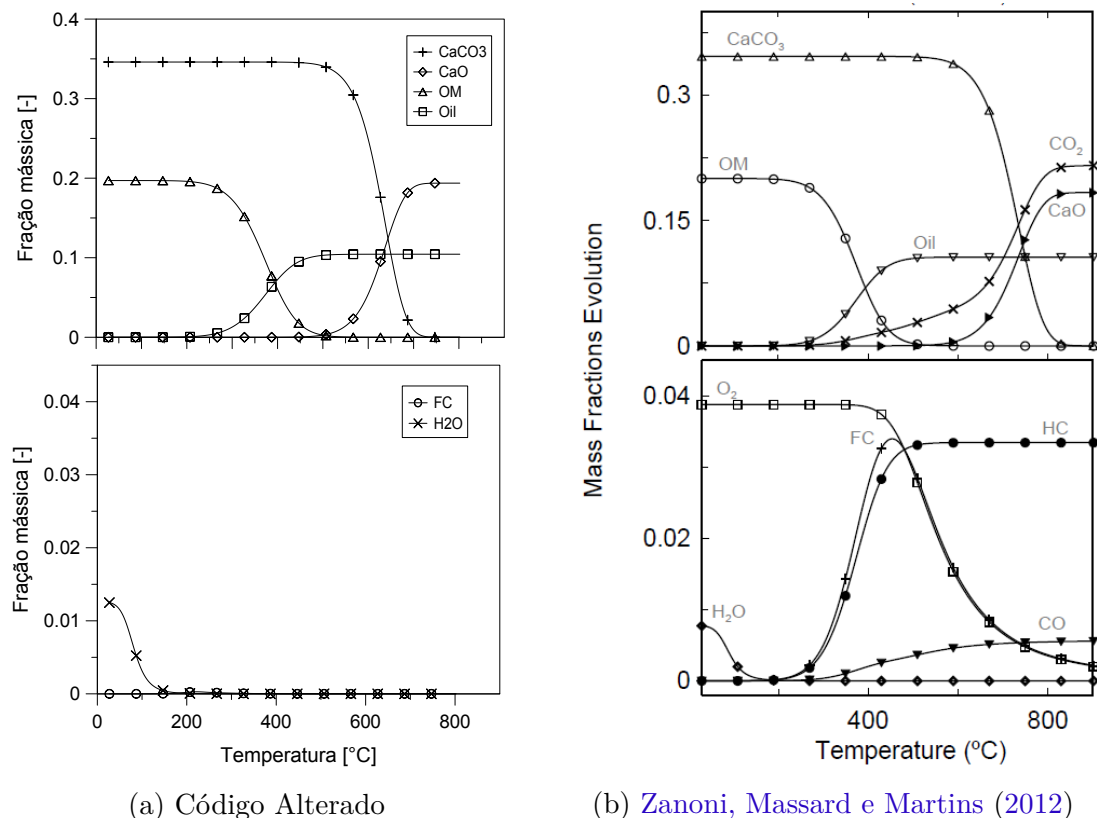
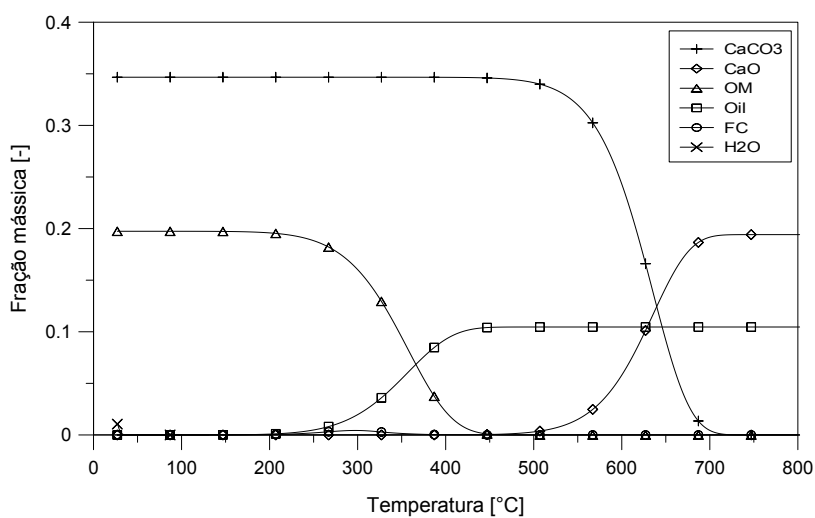


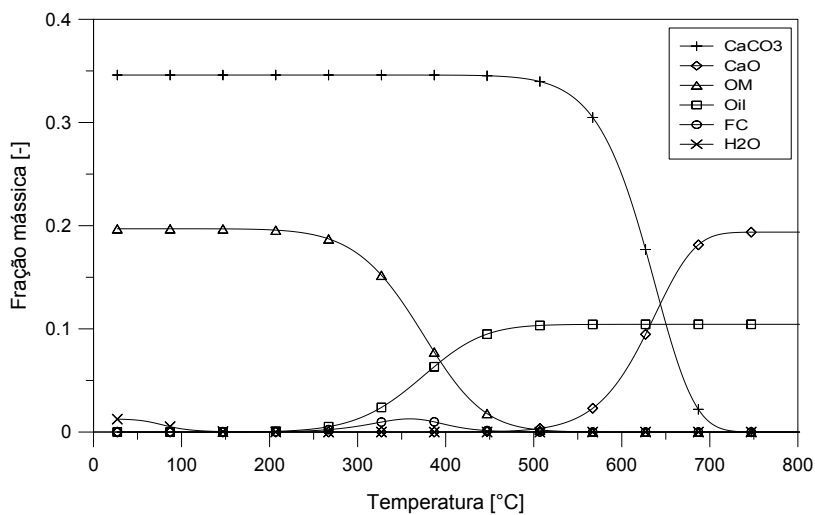
Figura 65 – Comparação da evolução das frações mássicas encontradas pelo Código Alterado e por Zanoni, Massard e Martins (2012)

da função *calculate()* foi nula.

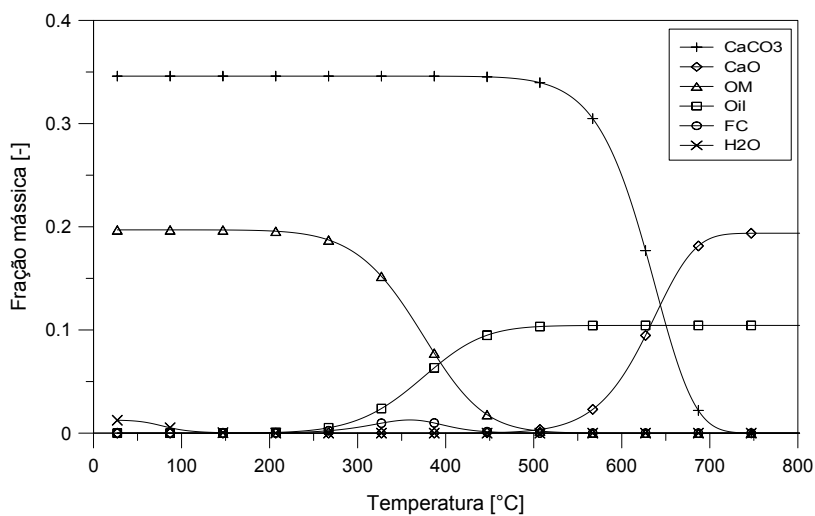
O uso da função *calculate()* no lugar da função *solve(t0, deltaT)* para o cálculo das taxas de consumo e produção das espécies não apresentou nenhuma influência para os testes realizados. Os casos testados possuem mecanismos de reação relativamente simples, não demonstrando ser estritamente necessário a solução do sistema de EDOs para a evolução da massa das espécies, como é feito pela função *solve(t0, deltaT)*. A função *calculate()* insere diretamente a expressão para a variação da determinada espécie no tempo, dada pela cinética química, como termo fonte para a equação de conservação das espécies para a fase sólida (Equação 4.6), o que não causou instabilidades numéricas para esses mecanismos simples. No entanto, não foi verificado um ganho quanto ao custo computacional para o uso da função *calculate()*, que era esperado por não realizar a solução de um sistema de EDOs. Por fim, ainda pairava uma indagação sobre a influência e aplicabilidade da função *calculate()* para mecanismos de reação mais complexos. Assim sendo, um simples teste foi realizado, adaptando o *solver chemFoam* para utilizar a função *calculate()* ao invés da função *solve(t0, deltaT)*, e aplicando-o para o tutorial *gri*, presente no diretório */tutorials/combustion/chemFoam/gri*. Esse tutorial serve para a validação da cinética química da reação de combustão em fase gasosa do metano, apresentando um mecanismo mais complexo, com 53 espécies e 325 reações. Para esse caso, a função *calculate()* não se mostrou capaz de solucionar o problema, apresentando severas instabilidades numéricas e divergência da solução, identificando de fato a necessidade de se solucionar o sistema de EDOs para obter a evolução da massa das espécies para mecanismos de reações mais complexos.



(a) Código Original



(b) Código Alterado



(c) Código Alterado + Calculate

Figura 66 – Evolução das frações mássicas para a combustão do xisto betuminoso obtidas com os três Códigos diferentes

6 Considerações finais

Neste trabalho, foi realizada a avaliação das características, da modelagem e da implementação do *solver* `biomassGasificationFoam`, desenvolvido por [Kwiatkowski et al. \(2013\)](#), constatando a sua aplicabilidade para processos diversos de conversão termoquímica de sólidos em leito fixo. No conjunto, o `biomassGasificationFoam` se mostrou uma ferramenta de grande potencial.

A análise desse *solver* identificou pontos que merecem atenção, motivando a realização de testes para a verificação e validação da modelagem do transporte das espécies gasosas, da transferência de calor entre as fases sólida e gasosa e da cinética química das reações na fase sólida.

A validação do transporte das espécies gasosas por advecção demonstrou a influência da solução numérica, constatando o efeito de difusão numérica característico do método *upwind*. Com isso em mente, os resultados apresentados foram satisfatórios comparados com a solução analítica.

Para o transporte das espécies gasosas por difusão, identificou-se que a implementação do `biomassGasificationFoam` utiliza a hipótese simplificadora de Número de Schmidt igual a um, ou seja, que a difusividade de momento e a difusividade mássica são iguais. Assim, na equação de conservação das espécies para a fase gasosa ele substitui o termo $\rho^G D^*$ pela viscosidade dinâmica μ . Os resultados obtidos para a validação do transporte por difusão mostraram que essa hipótese adotada causa uma superestimação do fenômeno de difusão para processos onde existam meios porosos, tanto para temperaturas elevadas como para temperaturas baixas.

A validação para a transferência de calor entre as fases sólida e gasosa encontrou resultados semelhantes entre a simulação e os resultados de [Martins \(2008\)](#), para um processo de resfriamento de um leito previamente aquecido pela passagem de um gás mais frio. Observou-se também a influência do coeficiente de transferência de calor nos perfis de temperatura do sólido e do gás.

Os testes realizados para a cinética química das reações na fase sólida demonstraram que o código original fornecido com o `biomassGasificationFoam` se mostrou adequado para reações homogêneas, como desumidificação e pirólise, que tenham ordem de reação unitária, como foi o caso dos testes para a pirólise de madeira impregnada com ácido fosfórico.

No entanto, o código original do `biomassGasificationFoam` não se mostrou adequado para reações homogêneas com ordem de reação diferente de um e, principalmente, para reações heterogêneas, como foi o caso da combustão do xisto betuminoso. As alterações no código propostas neste trabalho, sugerindo a mudança no cálculo das taxas de consumo e produção das espécies, no cálculo da Matriz Jacobiana e na inclusão da ordem de reação em relação ao gás, causaram diferenças significativas para esses casos, melhorando os resultados obtidos. Porém, os resultados para reações heterogêneas indicam que ainda há a necessidade de se realizar estudos mais aprofundados quanto à sua implementação.

Para os mecanismos de reação da pirólise de madeira impregnada com ácido fosfórico e da combustão do xisto betuminoso, o uso da função *calculate()* no lugar da função *solve(t0, deltaT)* para o cálculo das taxas de consumo e produção das espécies não apresentou nenhuma influência. Esses mecanismos são relativamente simples, não demonstrando ser obrigatória a solução do sistema de EDOs para a evolução da massa das espécies, como é feito pela função *solve(t0, deltaT)*, para garantir a estabilidade numérica. No entanto, para esses casos, não foi verificado um ganho em relação ao custo computacional com o uso da função *calculate()*, que era esperado por não realizar a solução de um sistema de EDOs. Por fim, foi realizado um teste com o uso da função *calculate()* para a combustão em fase gasosa do metano, cujo mecanismo de reação é mais complexo, contendo 53 espécies e 325 reações, onde constatou-se a incapacidade de solução desse problema com o uso dessa função. Isso confirma a necessidade de se solucionar o sistema de EDOs para obter a evolução da massa das espécies para mecanismos de reações mais complexos.

Devido ao número significativo de modelos cinéticos presentes na literatura, existe uma grande possibilidade de que o usuário tenha que modificar a implementação do modelo cinético de forma a adequá-lo ao seu estudo. Isso requer do usuário um conhecimento mais aprofundado da linguagem de programação C++ e de como ela é implementada no OpenFOAM.

Neste trabalho, foi gerada também uma documentação detalhada sobre o *software* OpenFOAM e, principalmente, sobre o *solver* `biomassGasificationFoam`, com o intuito de auxiliar a sua compreensão e utilização por futuros usuários interessados, visto que a complexidade dos fenômenos envolvidos e do próprio *software* implicam em uma custosa curva de aprendizado.

Algumas sugestões para trabalhos futuros são:

- Melhorar o modelo para a cinética química das reações heterogêneas.
- Incluir bibliotecas para tratar a difusão mássica das espécies gasosas em meio poroso.
- Implementar novos solucionadores de EDOs.
- Adicionar novas possibilidades para o cálculo das propriedades físicas do sólido.
- Adaptar o `biomassGasificationFoam` para versões mais recentes do OpenFOAM.

Referências

- ALVES, S. S.; FIGUEIREDO, J. L. A model for pyrolysis of wet wood. *Chemical Engineering Science*, Elsevier, v. 44, n. 12, p. 2861–2869, 1989. Citado 3 vezes nas páginas 26, 27 e 34.
- ANTAL, M. J. Biomass pyrolysis: A review of the literature Part 2 — Lignocellulose pyrolysis. In: *Advances in solar energy*. [S.l.]: Springer, 1985. p. 175–255. Citado na página 31.
- ANTAL, M. J.; GRØNLI, M. The art, science, and technology of charcoal production. *Industrial & Engineering Chemistry Research*, ACS Publications, v. 42, n. 8, p. 1619–1640, 2003. Citado na página 19.
- ANTAL, M. J.; VARHEGYI, G. Cellulose pyrolysis kinetics: the current state of knowledge. *Industrial & Engineering Chemistry Research*, ACS Publications, v. 34, n. 3, p. 703–717, 1995. Citado na página 31.
- ANTHONY, D. B. et al. Rapid devolatilization of pulverized coal. In: ELSEVIER. *Symposium (international) on combustion*. [S.l.], 1975. v. 15, n. 1, p. 1303–1317. Citado na página 35.
- ARENILLAS, A. et al. A comparison of different methods for predicting coal devolatilisation kinetics. *Journal of analytical and applied pyrolysis*, Elsevier, v. 58, p. 685–701, 2001. Citado na página 30.
- BAKER, R. R. Kinetic parameters from the non-isothermal decomposition of a multi-component solid. *Thermochimica Acta*, Elsevier, v. 23, n. 2, p. 201–212, 1978. Citado na página 31.
- BARBOSA, R. C. *Modelação Numérica de uma Caldeira Doméstica a Pellets*. Dissertação (Mestrado) — Instituto Superior de Engenharia do Porto, 2014. Citado 5 vezes nas páginas 54, 55, 56, 57 e 58.
- BARRIO, M. *Experimental investigation of small-scale gasification of woody biomass*. Tese (Doutorado) — Norwegian University of Science and Technology, 2002. Citado 4 vezes nas páginas 20, 23, 41 e 44.
- BARRIO, M. et al. Steam gasification of wood char and the effect of hydrogen inhibition on the chemical kinetics. In: _____. *Progress in Thermochemical Biomass Conversion*. [S.l.]: Blackwell Science Ltd, 2001. p. 32–46. Citado 2 vezes nas páginas 42 e 43.
- BARRIO, M.; HUSTAD, J. E. CO₂ gasification of birch char and the effect of CO inhibition on the calculation of chemical kinetics. In: _____. *Progress in Thermochemical Biomass Conversion*. [S.l.]: Blackwell Science Ltd, 2001. p. 47–60. Citado 2 vezes nas páginas 41 e 42.
- BASU, P. *Biomass Gasification, Pyrolysis, and Torrefaction: Practical Design and Theory*. 2. ed. [S.l.]: Elsevier Inc., 2013. Citado 10 vezes nas páginas 18, 19, 20, 21, 22, 23, 24, 41, 42 e 44.
- BELLAIS, M. *Modelling of the pyrolysis of large wood particles*. Tese (Doutorado) — Division of Chemical Technology, Department of Chemical Engineering and Technology, KTH - Royal Institute of Technology, 2007. Citado 2 vezes nas páginas 24 e 31.
- BERGMAN, P. C. et al. Torrefaction for biomass co-firing in existing coal-fired power stations. *Energy Centre of Netherlands, Report No. ECN-C-05-013*, 2005. Citado na página 20.
- BERGMAN, P. C.; KIEL, J. H. Torrefaction for biomass upgrading. In: *Proc. 14th European Biomass Conference, Paris, France*. [S.l.: s.n.], 2005. p. 17–21. Citado na página 20.

- BEWS, I. M. et al. The order, Arrhenius parameters, and mechanism of the reaction between gaseous oxygen and solid carbon. *Combustion and Flame*, Elsevier, v. 124, n. 1, p. 231–245, 2001. Citado na página 44.
- BRYDEN, K. M.; RAGLAND, K. W.; RUTLAND, C. J. Modeling thermally thick pyrolysis of wood. *Biomass and Bioenergy*, Elsevier, v. 22, n. 1, p. 41–53, 2002. Citado 2 vezes nas páginas 26 e 27.
- BURNHAM, A. K.; BRAUN, R. L. Global kinetic analysis of complex materials. *Energy & Fuels*, ACS Publications, v. 13, n. 1, p. 1–22, 1999. Citado 2 vezes nas páginas 31 e 37.
- CAI, J.; LIU, R. New distributed activation energy model: numerical solution and application to pyrolysis kinetics of some types of biomass. *Bioresource Technology*, Elsevier, v. 99, n. 8, p. 2795–2799, 2008. Citado na página 37.
- CHAN, W.-C. R.; KELBON, M.; KRIEGER, B. B. Modelling and experimental verification of physical and chemical processes during pyrolysis of a large biomass particle. *Fuel*, Elsevier, v. 64, n. 11, p. 1505–1513, 1985. Citado 2 vezes nas páginas 27 e 34.
- CHEN, Y. G. Extension of a coal pyrolysis model to biomass feedstocks. In: *Fuel and Energy Abstracts*. [S.l.: s.n.], 1998. v. 1, n. 39, p. 36. Citado na página 38.
- COLLAZO, J. et al. Numerical modeling of the combustion of densified wood under fixed-bed conditions. *Fuel*, Elsevier, v. 93, p. 149–159, 2012. Citado na página 26.
- CONESA, J. A. et al. Comments on the validity and utility of the different methods for kinetic analysis of thermogravimetric data. *Journal of Analytical and Applied Pyrolysis*, Elsevier, v. 58, p. 617–633, 2001. Citado 2 vezes nas páginas 30 e 31.
- CUNHA, F. A. da. *Modelo matemático para estudo de processos reativos de partículas de carvão e biomassa*. Tese (Doutorado) — Universidade de Brasília, 2010. Citado 13 vezes nas páginas 19, 31, 32, 33, 38, 39, 40, 41, 47, 95, 96, 118 e 143.
- DERNBECHER, A.; TABET, F.; ORTWEIN, A. A CFD-based approach for thermochemical conversion of straw. 2015. Citado 4 vezes nas páginas 52, 53, 54 e 55.
- DI BLASI, C. Modeling and simulation of combustion processes of charring and non-charring solid fuels. *Progress in Energy and Combustion Science*, Elsevier, v. 19, n. 1, p. 71–104, 1993. Citado 2 vezes nas páginas 31 e 34.
- DI BLASI, C. Heat, momentum and mass transport through a shrinking biomass particle exposed to thermal radiation. *Chemical engineering science*, Elsevier, v. 51, n. 7, p. 1121–1132, 1996. Citado na página 35.
- DI BLASI, C. Comparison of semi-global mechanisms for primary pyrolysis of lignocellulosic fuels. *Journal of Analytical and Applied Pyrolysis*, Elsevier, v. 47, n. 1, p. 43–64, 1998. Citado 2 vezes nas páginas 31 e 32.
- DI BLASI, C. Modeling chemical and physical processes of wood and biomass pyrolysis. *Progress in Energy and Combustion Science*, Elsevier, v. 34, n. 1, p. 47–90, 2008. Citado 4 vezes nas páginas 19, 31, 33 e 34.
- DI BLASI, C. Combustion and gasification rates of lignocellulosic chars. *Progress in energy and combustion science*, Elsevier, v. 35, n. 2, p. 121–140, 2009. Citado 6 vezes nas páginas 39, 40, 41, 42, 43 e 44.

- DI BLASI, C.; BUONANNO, F.; BRANCA, C. Reactivities of some biomass chars in air. *Carbon*, Elsevier, v. 37, n. 8, p. 1227–1238, 1999. Citado na página 44.
- DI BLASI, C.; RUSSO, G. Modeling of transport phenomena and kinetics of biomass pyrolysis. In: *Advances in thermochemical biomass conversion*. [S.l.]: Springer, 1993. p. 906–921. Citado na página 35.
- DING, L. P. et al. A heterogeneous model for gas transport in carbon molecular sieves. *Langmuir*, ACS Publications, v. 21, n. 2, p. 674–681, 2005. Citado na página 37.
- DU, Z.; SAROFIM, A. F.; LONGWELL, J. P. Activation energy distribution in temperature-programmed desorption: modeling and application to the soot oxygen system. *Energy & Fuels*, ACS Publications, v. 4, n. 3, p. 296–302, 1990. Citado na página 37.
- DUFFY, N. *Investigation of biomass combustion in grate furnaces using CFD*. Tese (Doutorado) — National University of Ireland, Galway, 2012. Citado 19 vezes nas páginas 15, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 40 e 47.
- EATON, A. M. et al. Components, formulations, solutions, evaluation, and application of comprehensive combustion models. *Progress in energy and combustion science*, Elsevier, v. 25, n. 4, p. 387–436, 1999. Citado 5 vezes nas páginas 30, 32, 37, 38 e 44.
- EL-RUB, Z. A.; BRAMER, E. A.; BREM, G. Review of catalysts for tar elimination in biomass gasification processes. *Industrial & engineering chemistry research*, ACS Publications, v. 43, n. 22, p. 6911–6919, 2004. Citado na página 34.
- ÇENGEL, Y. A. *Transferência de Calor e Massa: Uma Abordagem Prática*. 3. ed. São Paulo: McGraw-Hill, 2009. Citado na página 129.
- FELFLI, F. E. F.; LUENGO, C. A.; SOLER, P. B. Torrefação de biomassa: Características, aplicações e perspectivas. In: SCIELO BRASIL. *Proceedings of the 3. Encontro de Energia no Meio Rural*. [S.l.], 2000. Citado na página 20.
- FLETCHER, D. F. et al. A CFD based combustion model of an entrained flow biomass gasifier. *Applied mathematical modelling*, Elsevier, v. 24, n. 3, p. 165–182, 2000. Citado na página 44.
- FLETCHER, T. H. et al. Chemical percolation model for devolatilization. 2. Temperature and heating rate effects on product yields. *Energy & Fuels*, ACS Publications, v. 4, n. 1, p. 54–60, 1990. Citado na página 38.
- FLETCHER, T. H. et al. Chemical percolation model for devolatilization. 3. Direct use of carbon-13 NMR data to predict effects of coal type. *Energy & Fuels*, ACS Publications, v. 6, n. 4, p. 414–431, 1992. Citado na página 38.
- FM GLOBAL. *FireFOAM*. 2016. [Online; acessado em 02 de agosto de 2016]. Disponível em: <<https://code.google.com/archive/p/firefoam-dev/>>. Citado na página 48.
- FONT, R. et al. Kinetic law for solids decomposition. Application to thermal degradation of heterogeneous materials. *Journal of Analytical and Applied Pyrolysis*, Elsevier, v. 58, p. 703–731, 2001. Citado na página 31.
- GAO, J. et al. Experiments and numerical simulation of sawdust gasification in an air cyclone gasifier. *Chemical engineering journal*, Elsevier, v. 213, p. 97–103, 2012. Citado na página 47.
- GERUN, L. et al. Numerical investigation of the partial oxidation in a two-stage downdraft gasifier. *Fuel*, Elsevier, v. 87, n. 7, p. 1383–1393, 2008. Citado na página 46.

GRANT, D. M. et al. Chemical model of coal devolatilization using percolation lattice statistics. *Energy & Fuels*, ACS Publications, v. 3, n. 2, p. 175–186, 1989. Citado na página 38.

GRØNLI, M. G. *A theoretical and experimental study of the thermal degradation of biomass*. Tese (Doutorado) — Norwegian University of Science and Technology, 1996. Citado 5 vezes nas páginas 18, 19, 20, 23 e 28.

HALLGREN, A. *Theoretical and engineering aspects of the gasification of biomass*. Tese (Doutorado) — Lund University, Suécia, 1996. Citado na página 19.

HARED, I. A. et al. Pyrolysis of wood impregnated with phosphoric acid for the production of activated carbon: Kinetics and porosity development studies. *Journal of analytical and applied pyrolysis*, Elsevier, v. 79, n. 1, p. 101–105, 2007. Citado 4 vezes nas páginas 7, 157, 158 e 160.

HOLZINGER, G. *OpenFOAM A little User-Manual*. 2016. Disponível em: <<https://github.com/ParticulateFlow/osccar-doc>>. Citado na página 62.

HONG, J. *Modeling char oxidation as a function of pressure using an intrinsic Langmuir rate equation*. Tese (Doutorado) — Brigham Young University, 2000. Citado 4 vezes nas páginas 39, 40, 41 e 44.

HUANG, H. jun; YUAN, X. zhong. Recent progress in the direct liquefaction of typical biomass. *Progress in Energy and Combustion Science*, Elsevier, v. 49, p. 59–80, 2015. Citado na página 20.

HURT, R. H.; CALO, J. M. Semi-global intrinsic kinetics for char combustion modeling. *Combustion and flame*, Elsevier, v. 125, n. 3, p. 1138–1149, 2001. Citado 3 vezes nas páginas 41, 43 e 44.

HURT, R. H.; HAYNES, B. S. On the origin of power-law kinetics in carbon oxidation. *Proceedings of the Combustion Institute*, Elsevier, v. 30, n. 2, p. 2161–2168, 2005. Citado na página 44.

HÜTTINGER, K. J.; MERDES, W. F. The carbon-steam reaction at elevated pressure: formations of product gases and hydrogen inhibitions. *Carbon*, Elsevier, v. 30, n. 6, p. 883–894, 1992. Citado na página 42.

INCROPERA, F. P. et al. *Fundamentos de Transferência de Calor e de Massa*. 7. ed. Rio de Janeiro: LTC, 2014. Citado na página 129.

JAKOBS, T. et al. Gasification of high viscous slurry R&D on atomization and numerical simulation. *Applied Energy*, Elsevier, v. 93, p. 449–456, 2012. Citado na página 47.

JANAJREH, I.; SHRAH, M. A. Numerical and experimental investigation of downdraft gasification of wood chips. *Energy Conversion and Management*, Elsevier, v. 65, p. 783–792, 2013. Citado na página 47.

JASAK, H. *Error Analysis and Estimation for the Finite Volume Method with Applications to Fluid Flows*. Tese (Doutorado) — Imperial College, University of London, 1996. Citado na página 62.

JASAK, H. OpenFOAM: Open source CFD in research and industry. *International Journal of Naval Architecture and Ocean Engineering*, Elsevier, v. 1, n. 2, p. 89–94, 2009. Citado na página 62.

JASAK, H.; JEMCOV, A.; TUKOVIC, Z. OpenFOAM: A C++ library for complex physics simulations. In: IUC DUBROVNIK, CROATIA. *International workshop on coupled methods in numerical dynamics*. [S.l.], 2007. v. 1000, p. 1–20. Citado na página 62.

- KHADRA, K. et al. Fictitious domain approach for numerical modelling of Navier–Stokes equations. *International journal for numerical methods in fluids*, Wiley Online Library, v. 34, n. 8, p. 651–684, 2000. Citado na página 93.
- KLOSE, W.; WÖLKI, M. On the intrinsic reaction rate of biomass char gasification with carbon dioxide and steam. *Fuel*, Elsevier, v. 84, n. 7, p. 885–892, 2005. Citado na página 44.
- KOBAYASHI, H.; HOWARD, J. B.; SAROFIM, A. F. Coal devolatilization at high temperatures. *Symposium (International) on Combustion*, v. 16, n. 1, p. 411 – 425, 1977. Citado na página 32.
- KORTELAINEN, J. Meshing tools for open source CFD - A practical point of view. 2009. Citado na página 67.
- KRIEGER-BROCKETT, B.; GLAISTER, D. S. Wood devolatilization - sensitivity to feed properties and process variables. In: *Research in Thermochemical Biomass Conversion*. [S.l.]: Springer, 1988. p. 127–142. Citado na página 27.
- KWIATKOWSKI, K. et al. Biomass gasification solver based on OpenFOAM. 2013. Citado 23 vezes nas páginas 15, 48, 49, 50, 51, 52, 55, 86, 87, 88, 93, 94, 95, 96, 98, 100, 101, 102, 117, 118, 126, 157 e 167.
- KWIATKOWSKI, K. et al. Pyrolysis and gasification of single biomass particle – new openFoam solver. In: IOP PUBLISHING. *Journal of Physics: Conference Series*. [S.l.], 2014. v. 530, n. 1, p. 012015. Citado 2 vezes nas páginas 48 e 49.
- LAKSHMANAN, C. C.; WHITE, N. A new distributed activation energy model using Weibull distribution for the representation of complex kinetics. *Energy & fuels*, ACS Publications, v. 8, n. 6, p. 1158–1167, 1994. Citado na página 37.
- LAPENE, A. *Modélisation numérique de la combustion en lit fixe de combustibles solides*. Dissertação (Mestrado) — Université Paul Sabatier Toulouse, 2006. Citado na página 135.
- LAPENE, A. et al. Numerical simulation of combustion in reactive porous media. *International Review of Mechanical Engineering*, Praise Worthy Prize, v. 12, 2008. Citado na página 46.
- LAPENE, A. et al. Numerical simulation of oil shale combustion in a fixed bed: modelling and chemical aspects. In: *Eurotherm seminar*. [S.l.: s.n.], 2007. Citado na página 46.
- LATHOUWERS, D.; BELLAN, J. Modeling of dense gas–solid reactive mixtures applied to biomass pyrolysis in a fluidized bed. *International Journal of Multiphase Flow*, Elsevier, v. 27, n. 12, p. 2155–2187, 2001. Citado na página 45.
- LATHOUWERS, D.; BELLAN, J. Yield optimization and scaling of fluidized beds for tar production from biomass. *Energy & Fuels*, ACS Publications, v. 15, n. 5, p. 1247–1262, 2001. Citado na página 45.
- LAURENDEAU, N. M. Heterogeneous kinetics of coal char gasification and combustion. *Progress in energy and combustion science*, Pergamon, v. 4, n. 4, p. 221–270, 1978. Citado 3 vezes nas páginas 39, 41 e 44.
- LU, H. *Experimental and modeling investigations of biomass particle combustion*. Tese (Doutorado) — Brigham Young University, 2006. Citado na página 46.
- LUENGO, C. A.; FELFLI, F. E. F.; BEZZON, G. Capítulo X - pirólise e torrefação de biomassa. *Biomassa para Energia*, 2006. Citado 2 vezes nas páginas 20 e 21.

- LUNDSTRÖM, A. *OpenFOAM Tutorial: reactingFoam Simple Gas Phase Reaction*. 2008. Disponível em: <http://www.tfd.chalmers.se/~hani/kurser/OS_CFD_2007/AndreasLundstrom/reactingFoam.pdf>. Citado na página 113.
- LUO, Z.; WANG, S.; CEN, K. A model of wood flash pyrolysis in fluidized bed reactor. *Renewable Energy*, Elsevier, v. 30, n. 3, p. 377–392, 2005. Citado na página 45.
- MAKI, T.; TAKATSUNO, A.; MIURA, K. Analysis of pyrolysis reactions of various coals including argonne premium coals using a new distributed activation energy model. *Energy & Fuels*, ACS Publications, v. 11, n. 5, p. 972–977, 1997. Citado na página 37.
- MARTÍNEZ, J. D. et al. Waste tyre pyrolysis – a review. *Renewable and Sustainable Energy Reviews*, Elsevier, v. 23, p. 179–213, 2013. Citado na página 22.
- MARTINS, M. F. *The structure of a combustion front propagating in a fixed bed of crushed oil shale: co-current configuration*. Tese (Doutorado) — University of Toulouse, 2008. Citado 5 vezes nas páginas 7, 46, 137, 138 e 167.
- MILLINGTON, R. J.; QUIRK, J. P. Permeability of porous solids. *Transactions of the Faraday Society*, Royal Society of Chemistry, v. 57, p. 1200–1207, 1961. Citado na página 131.
- MIRANDA, M. et al. Mixtures of rubber tyre and plastic wastes pyrolysis: A kinetic study. *Energy*, Elsevier, v. 58, p. 270–282, 2013. Citado na página 22.
- MIRANDA, M. et al. Pyrolysis of rubber tyre wastes: A kinetic study. *Fuel*, Elsevier, v. 103, p. 542–552, 2013. Citado na página 22.
- MITTAL, R.; IACCARINO, G. Immersed boundary methods. *Annu. Rev. Fluid Mech.*, Annual Reviews, v. 37, p. 239–261, 2005. Citado na página 93.
- MIURA, K. A new and simple method to estimate $f(E)$ and $k_0(E)$ in the distributed activation energy model from three sets of experimental data. *Energy & Fuels*, ACS Publications, v. 9, n. 2, p. 302–307, 1995. Citado 2 vezes nas páginas 36 e 37.
- MIURA, K.; MAKI, T. A simple method for estimating $f(E)$ and $k_0(E)$ in the distributed activation energy model. *Energy & Fuels*, ACS Publications, v. 12, n. 5, p. 864–869, 1998. Citado na página 37.
- MOGHTADERI, B. The state-of-the-art in pyrolysis modelling of lignocellulosic solid fuels. *Fire and Materials*, Wiley Online Library, v. 30, n. 1, p. 1–34, 2006. Citado 2 vezes nas páginas 31 e 33.
- MORF, P.; HASLER, P.; NUSSBAUMER, T. Mechanisms and kinetics of homogeneous secondary reactions of tar from continuous pyrolysis of wood chips. *Fuel*, Elsevier, v. 81, n. 7, p. 843–853, 2002. Citado na página 34.
- NIKSA, S. Predicting the rapid devolatilization of diverse forms of biomass with bio-flashchain. *Proceedings of the Combustion Institute*, Elsevier, v. 28, n. 2, p. 2727–2733, 2000. Citado na página 38.
- NIKSA, S.; KERSTEIN, A. R. FLASHCHAIN theory for rapid coal devolatilization kinetics. 1. Formulation. *Energy & Fuels*, ACS Publications, v. 5, n. 5, p. 647–665, 1991. Citado na página 38.
- NOVARESIO, V. *Multispecies mass transport library*. 2012. Citado na página 99.

- NOVARESIO, V. et al. An open-source library for the numerical modeling of mass-transfer in solid oxide fuel cells. *Computer Physics Communications*, Elsevier, v. 183, n. 1, p. 125–146, 2012. Citado na página 99.
- OPENFOAM PROGRAMMER'S GUIDE. OpenFOAM: The Open Source CFD Toolbox Programmer's Guide Versão 2.1.1. *OpenFOAM Foundation*, 2012. Citado na página 62.
- OPENFOAM USER GUIDE. OpenFOAM: The Open Source CFD Toolbox User Guide Versão 2.1.1. *OpenFOAM Foundation*, 2012. Citado 12 vezes nas páginas 62, 63, 65, 67, 68, 72, 76, 77, 78, 80, 82 e 83.
- OUELHAZI, N.; ARNAUD, G.; FOHR, J. P. A two-dimensional study of wood plank drying. the effect of gaseous pressure below boiling point. *Transport in porous media*, Springer, v. 7, n. 1, p. 39–61, 1992. Citado na página 27.
- PEPIOT, P.; DIBBLE, C.; FOUST, T. Computational fluid dynamics modeling of biomass gasification and pyrolysis. In: *Computational Modeling in lignocellulosic biofuel production, ACS Symposium Series*. [S.l.: s.n.], 2010. v. 1052, p. 273–298. Citado na página 47.
- PETERS, B.; BRUCH, C. Drying and pyrolysis of wood particles: experiments and simulation. *Journal of Analytical and Applied Pyrolysis*, Elsevier, v. 70, n. 2, p. 233–250, 2003. Citado 3 vezes nas páginas 25, 27 e 31.
- PITT, G. J. The kinetics of the evolution of volatile products from coal. *Fuel*, Elsevier, v. 41, n. 3, p. 267–274, 1962. Citado na página 35.
- PLEASE, C. P.; MCGUINNESS, M. J.; MCELWAIN, D. L. S. Approximations to the distributed activation energy model for the pyrolysis of coal. *Combustion and Flame*, Elsevier, v. 133, n. 1, p. 107–117, 2003. Citado 2 vezes nas páginas 36 e 37.
- ROSTAMI, A.; MURTHY, J.; HAJALIGOL, M. Modeling of smoldering process in a porous biomass fuel rod. *Fuel*, Elsevier, v. 83, n. 11, p. 1527–1536, 2004. Citado na página 45.
- ROSTAMI, A. A.; HAJALIGOL, M. R.; WRENN, S. E. A biomass pyrolysis sub-model for CFD applications. *Fuel*, Elsevier, v. 83, n. 11, p. 1519–1525, 2004. Citado 3 vezes nas páginas 35, 36 e 37.
- SAIDI, M. et al. A 3D modeling of static and forward smoldering combustion in a packed bed of materials. *Applied mathematical modelling*, Elsevier, v. 31, n. 9, p. 1970–1996, 2007. Citado na página 46.
- SHAFIZADEH, F.; CHIN, P. P. S. Thermal deterioration of wood. In: *ACS Symposium Series American Chemical Society*. [S.l.: s.n.], 1977. Citado 3 vezes nas páginas 33, 34 e 35.
- SHARUDDIN, S. D. A. et al. A review on pyrolysis of plastic wastes. *Energy Conversion and Management*, Elsevier, v. 115, p. 308–326, 2016. Citado na página 21.
- SILVA, L. F. L. R. *Uma visão geral do pacote CFD OpenFOAM*. 2007. Citado 3 vezes nas páginas 60, 62 e 78.
- SILVA, L. F. L. R. *Desenvolvimento de metodologias para simulação de escoamentos polidispersos usando código livre*. Tese (Doutorado) — Universidade Federal do Rio de Janeiro, 2008. Citado 3 vezes nas páginas 59, 60 e 62.
- SMITH, I. W. The combustion rates of coal chars: a review. In: ELSEVIER. *Symposium (International) on combustion*. [S.l.], 1982. v. 19, n. 1, p. 1045–1065. Citado na página 44.

- SMITH, K. L. et al. *The structure and reaction processes of coal*. [S.l.]: Springer Science & Business Media, 1994. Citado 3 vezes nas páginas 38, 40 e 44.
- SMOOT, L. D.; SMITH, P. J. *Coal combustion and gasification*. [S.l.]: Springer Science & Business Media, 1985. Citado na página 44.
- SOLOMON, P. R. et al. General model of coal devolatilization. *Energy & Fuels*, ACS Publications, v. 2, n. 4, p. 405–422, 1988. Citado na página 37.
- SOLOMON, P. R. et al. Advanced in the FG-DVC model of coal devolatilization. *Preprints of Papers, American Chemical Society, Division of Fuel Chemistry;(USA)*, v. 35, n. CONF-900402–, 1990. Citado na página 37.
- SUTTON, D.; KELLEHER, B.; ROSS, J. R. H. Review of literature on catalysts for biomass gasification. *Fuel Processing Technology*, Elsevier, v. 73, n. 3, p. 155–173, 2001. Citado na página 34.
- SUUBERG, E. Approximate solution technique for nonisothermal, Gaussian distributed activation energy models. *Combustion and Flame*, Elsevier, v. 50, p. 243–245, 1983. Citado na página 37.
- THURNER, F.; MANN, U. Kinetic investigation of wood pyrolysis. *Industrial & Engineering Chemistry Process Design and Development*, ACS Publications, v. 20, n. 3, p. 482–488, 1981. Citado 2 vezes nas páginas 33 e 35.
- TURÁNYI, T.; TOMLIN, A. S. Reaction kinetics basics. In: _____. *Analysis of Kinetic Reaction Mechanisms*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014. p. 5–37. ISBN 978-3-662-44562-4. Disponível em: <http://dx.doi.org/10.1007/978-3-662-44562-4_2>. Citado 2 vezes nas páginas 154 e 156.
- UBHAYAKAR, S. K. et al. Rapid devolatilization of pulverized coal in hot combustion gases. *Symposium (International) on Combustion*, v. 16, n. 1, p. 427 – 436, 1977. Citado na página 32.
- VAND, V. A theory of the irreversible electrical resistance changes of metallic films evaporated in vacuum. *Proceedings of the Physical Society*, IOP Publishing, v. 55, n. 3, p. 222, 1943. Citado na página 35.
- VÁRHEGYI, G.; SZABÓ, P.; ANTAL, M. J. Kinetics of charcoal devolatilization. *Energy & fuels*, ACS Publications, v. 16, n. 3, p. 724–731, 2002. Citado na página 37.
- VDOVIN, A. *Radiation heat transfer in OpenFOAM*. 2009. Disponível em: <http://www.tfd.chalmers.se/~hani/kurser/OS_CFD_2009/AlexeyVdovin/Radiation_in_OpenFoam_final.pdf>. Citado na página 120.
- WALKER, P. L.; RUSINKO, F.; AUSTIN, L. G. Gas reactions of carbon. *Advances in catalysis*, Elsevier, v. 11, p. 133–221, 1959. Citado 2 vezes nas páginas 39 e 44.
- WANG, Y.; YAN, L. CFD studies on biomass thermochemical conversion. *International journal of molecular sciences*, Molecular Diversity Preservation International, v. 9, n. 6, p. 1108–1130, 2008. Citado 4 vezes nas páginas 15, 31, 44 e 47.
- WEBB, S. W. Gas transport mechanisms. In: _____. *Gas Transport in Porous Media*. Dordrecht: Springer Netherlands, 2006. p. 5–26. ISBN 978-1-4020-3962-1. Disponível em: <http://dx.doi.org/10.1007/1-4020-3962-X_2>. Citado 2 vezes nas páginas 130 e 131.
- WELLER, H. G. et al. A tensorial approach to computational continuum mechanics using object-oriented techniques. *Computers in physics*, AIP Publishing, v. 12, n. 6, p. 620–631, 1998. Citado na página 62.

- WILLIAMS, A. et al. Modelling coal combustion: the current position. *Fuel*, Elsevier, v. 81, n. 5, p. 605–618, 2002. Citado na página 44.
- YANG, Y. B. et al. Combustion of a single particle of biomass. *Energy & Fuels*, ACS Publications, v. 22, n. 1, p. 306–316, 2007. Citado na página 26.
- YIN, C.; ROSENDAHL, L. A.; KÆR, S. K. Grate-firing of biomass for heat and power production. *Progress in Energy and Combustion Science*, Elsevier, v. 34, n. 6, p. 725–754, 2008. Citado na página 47.
- YU, L. et al. Numerical simulation of the bubbling fluidized bed coal gasification by the kinetic theory of granular flow (KTGF). *Fuel*, Elsevier, v. 86, n. 5, p. 722–734, 2007. Citado na página 45.
- ZANONI, M. A. B.; MASSARD, H.; MARTINS, M. F. Formulating and optimizing a combustion pathways for oil shale and its semi-coke. *Combustion and Flame*, Elsevier, v. 159, n. 10, p. 3224–3234, 2012. Citado 5 vezes nas páginas 7, 161, 163, 164 e 165.
- ZHOU, H.; FLAMANT, G.; GAUTHIER, D. DEM-LES of coal combustion in a bubbling fluidized bed. Part I: gas-particle turbulent flow structure. *Chemical engineering science*, Elsevier, v. 59, n. 20, p. 4193–4203, 2004. Citado na página 45.
- ZHOU, H.; FLAMANT, G.; GAUTHIER, D. DEM-LES simulation of coal combustion in a bubbling fluidized bed Part II: coal combustion at the particle level. *Chemical Engineering Science*, Elsevier, v. 59, n. 20, p. 4205–4215, 2004. Citado na página 45.
- ZONGYUAN, G. *Introduction to ODE solvers and their application in OpenFOAM*. 2009. Disponível em: <http://www.tfd.chalmers.se/~hani/kurser/OS_CFD_2008/ZongyuanGu/reportZongyuan.pdf>. Citado na página 154.

Apêndices

APÊNDICE A – Classe

ODESolidHeterogeneousChemistryModel:

Mudanças Propostas

A.1 Função *omega(c, T, p, updateC0)*

```

1  template<class CompType, class SolidThermo, class GasThermo>
2  Foam::scalarField Foam::
3  ODESolidHeterogeneousChemistryModel<CompType, SolidThermo, GasThermo>::omega
4  (
5      const scalarField& c,
6      const scalar T,
7      const scalar p,
8      const bool updateC0
9  ) const
10 {
11     scalar pf, cf, pr, cr;
12     label lRef, rRef;
13
14     // const label cellI = cellCounter_;
15     label nEq = nEqns();
16
17     if (not solidReactionEnergyFromEnthalpy_)
18     {
19         nEq = nEqns()+1;
20     }
21
22     scalarField om(nEq,0.0);
23
24     forAll(reactions_, i)
25     {
26         const solidHeterogeneousReaction& R = reactions_[i];
27
28         scalar omegai = omega
29         (
30             R, c, T, 0.0, pf, cf, lRef, pr, cr, rRef
31         );
32
33         scalar massCoefficient = 1;
34         scalar massStoRatio = 1;
35
36         scalar substrates = 0;
37         scalar products = 0;
38
39         scalar solidSubstrates = 0;
40         scalar solidProducts = 0;
41

```



```

42     forAll(R.grhs(), g)
43     {
44         label gi = R.grhs()[g];
45         products += gasThermo_[gi].W() * R.grhsSto()[g];
46     }
47
48     forAll(R.glhs(), g)
49     {
50         label gi = pyrolysisGases_[gasPhaseGases_[R.glhs()[g]].name()];
51         substrates += gasThermo_[gi].W()*R.glhsSto()[g];
52     }
53
54     forAll(R.slhs(), s)
55     {
56         solidSubstrates += R.slhsSto()[s];
57     }
58
59     forAll(R.srhs(), s)
60     {
61         solidProducts += R.srhsSto()[s];
62     }
63
64     massCoefficient *= products/(products - substrates);
65     massStoRatio *= substrates/products;
66
67     forAll(R.slhs(), s)
68     {
69         label si = R.slhs()[s];
70         om[si] -= omegai*R.slhsSto()[s]/solidSubstrates;
71         /* if (updateC0)
72         {
73             Ys0_[si][celli] = this->solidThermo().rho()[celli] *Ys_[si][celli]
74                 * V_[celli];
75         } */
76     }
77
78     scalar sr = solidProducts/solidSubstrates;
79
80     forAll(R.srhs(), s)
81     {
82         label si = R.srhs()[s];
83         om[si] += omegai*R.srhsSto()[s]/solidSubstrates;
84
85         /* if (updateC0)
86         {
87             Ys0_[si][celli] = this->solidThermo().rho()[celli] *Ys_[si][celli]
88                 * V_[celli];
89         } */
90     }
91
92     forAll(R.grhs(), g)
93     {
94         label gi = R.grhs()[g];

```

```

96         om[gi + nSolids_] += (1.0 - sr)*omegai*massCoefficient*gasThermo_[gi
97             ].W()*R.grhsSto()[g]/products;
98     }
99     forAll(R.glhs(), g)
100     {
101         label gi = pyrolysisGases_[gasPhaseGases_[R.glhs()[g]].name()];
102         om[gi + nSolids_] -= (1.0 - sr)*omegai*massCoefficient*massStoRatio*
103             gasThermo_[gi].W()*R.glhsSto()[g]/substrates;
104     }
105     if (not solidReactionEnergyFromEnthalpy_)
106     {
107         om[nEqns()] += omegai*(1.0 - sr)*R.heatReact();
108     }
109 }
110 }
111
112 return om;
113 }

```

A.2 Função $\omega(R, c, T, p, pf, cf, IRef, pr, cr, rRef)$

```

1  template<class CompType, class SolidThermo, class GasThermo>
2  Foam::scalar
3  Foam::ODESolidHeterogeneousChemistryModel<CompType, SolidThermo, GasThermo>::
4  omega
5  (
6      const solidHeterogeneousReaction& R,
7      const scalarField& c,
8      const scalar T,
9      const scalar p,
10     scalar& pf,
11     scalar& cf,
12     label& IRef,
13     scalar& pr,
14     scalar& cr,
15     label& rRef
16 ) const
17 {
18     scalarField c1(nSpecie_, 0.0);
19     label cellI = cellCounter_;
20
21     for (label i=0; i<nSpecie_; i++)
22     {
23         c1[i] = max(0.0, c[i]);
24     }
25
26     scalar kf = R.kf(T, 0.0, c1);
27
28     scalar exponent = R.nReact(); //Adicionado
29     scalar exponentGas = R.nReactGas(); //Adicionado
30 }

```

```

31  const label NI = R.slhs().size();
32
33  if ( R.glhs().size() > 0 )
34  {
35      for (label s=0; s < NI; s++)
36      {
37          label si = R.slhs()[s];
38          forAll(R.glhs(), i)
39          {
40              if (c1[si] == 0)
41              {
42                  kf *= 0;
43              }
44              else
45              {
46                  kf *=
47                      c1[si]*Foam::pow(Ys_[si][celli], exponent - 1.0)
48                      *Foam::pow(gasPhaseGases_[R.glhs()[i]].internalField()[celli],
49                          exponentGas); //Modificado
49          }
50      }
51  }
52  }
53  else
54  {
55      for (label s=0; s<NI; s++)
56      {
57          label si = R.slhs()[s];
58          if (c1[si] == 0)
59          {
60              kf *= 0;
61          }
62          else
63          {
64              kf *=
65                  c1[si]*Foam::pow(Ys_[si][celli], exponent - 1.0); //Modificado
66          }
67      }
68  }
69  return kf;
70 }

```

A.3 Função $solve(t0, \delta T)$

```

1  template<class CompType, class SolidThermo, class GasThermo>
2  Foam::scalar
3  Foam::ODESolidHeterogeneousChemistryModel<CompType, SolidThermo, GasThermo>::
4  solve
5  (
6      const scalar t0,
7      const scalar deltaT
8  )
9  {

```

```
10  const volScalarField rho
11  (
12      IObject
13      (
14          "rho",
15          this->time().timeName(),
16          this->mesh(),
17          IObject::NO_READ,
18          IObject::NO_WRITE,
19          false
20      ),
21      this->solidThermo().rho()
22  );
23
24  if (this->mesh().changing())
25  {
26      forAll(RRs_, i)
27      {
28          RRs_[i].setSize(rho.size());
29      }
30      forAll(RRg_, i)
31      {
32          RRg_[i].setSize(rho.size());
33      }
34  }
35
36  forAll(RRs_, i)
37  {
38      RRs_[i] = 0.0;
39  }
40  forAll(RRg_, i)
41  {
42      RRg_[i] = 0.0;
43  }
44
45  forAll(shReactionHeat_, cellI)
46  {
47      shReactionHeat_[cellI] = 0.0;
48  }
49
50  if (!this->chemistry_)
51  {
52      return GREAT;
53  }
54
55  scalar deltaTMin = GREAT;
56
57  forAll(rho, cellI)
58  {
59      if (reactingCells_[cellI])
60      {
61          cellCounter_ = cellI;
62
63          scalar rhoI = rho[cellI];
64          scalar Ti = this->solidThermo().T()[cellI];
65
66          scalarField c(nSpecie_, 0.0);
```

```

67     scalarField c0(nSpecie_, 0.0);
68     scalarField dc(nSpecie_, 0.0);
69
70     //scalarField omegaPreq(omega(c0, Ti, 0.0));
71
72     scalar delta = this->mesh().V()[celli];
73
74     for (label i=0; i<nSolids_; i++)
75     {
76         c[i] = rhoi*Ys_[i][celli]*delta;
77     }
78
79     c0 = c;
80
81     scalarField omegaPreq(omega(c0, Ti, 0.0)); //Acrescentado
82
83     scalar t = t0;
84     scalar tauC = this->deltaTChem_[celli];
85     scalar dt = min(deltaT, tauC);
86     scalar timeLeft = deltaT;
87
88     // calculate the chemical source terms
89     while (timeLeft > SMALL)
90     {
91         tauC = this->solve(c, Ti, 0.0, t, dt);
92         t += dt;
93
94         // update the temperature
95         scalar cTot = 0.0;
96
97         //Total mass concentration
98         for (label i=0; i<nSolids_; i++)
99         {
100             cTot += c[i];
101         }
102
103         scalar newCp = 0.0;
104         scalar newhi = 0.0;
105         scalar invRho = 0.0;
106         scalarList dcdt = (c - c0)/dt;
107
108         if (solidReactionEnergyFromEnthalpy_)
109         {
110             for (label i=0; i<nSolids_; i++)
111             {
112                 scalar dYi = dcdt[i]/cTot;
113                 scalar Yi = c[i]/cTot;
114                 newCp += Yi*solidThermo_[i].Cp(Ti);
115                 newhi -= dYi*solidThermo_[i].hf();
116                 invRho += Yi/solidThermo_[i].rho(Ti);
117             }
118         }
119         else
120         {
121             for (label i=0; i<nSolids_; i++)
122             {
123                 scalar Yi = c[i]/cTot;

```

```

124         newCp += Yi*solidThermo_[i].Cp(Ti);
125         invRho += Yi/solidThermo_[i].rho(Ti);
126     }
127     newhi += omegaPreq[nEqns()];
128 }
129
130 scalar dTi = (newhi/newCp)*dt;
131
132 Ti += dTi;
133
134 timeLeft -= dt;
135 this->deltaTChem_[celli] = tauC;
136 dt = min(timeLeft, tauC);
137 dt = max(dt, SMALL);
138 }
139
140 deltaTMin = min(tauC, deltaTMin);
141 dc = c - c0;
142
143 forAll(RRs_, i)
144 {
145     RRs_[i][celli] = dc[i]/(deltaT*delta);
146 }
147
148 forAll(RRg_, i)
149 {
150     RRg_[i][celli] = dc[nSolids_ + i]/(deltaT*delta);
151 }
152
153 if (not solidReactionEnergyFromEnthalpy_)
154 {
155     shReactionHeat_[celli] = omegaPreq[nEqns()]/(delta);
156     //shReactionHeat_[celli] = omegaPreq[nEqns()]/(deltaT*delta);
157 }
158
159 /* // Update Ys0_
160 omegaPreq = omega(c0, Ti, 0.0, true);
161 if (solidReactionEnergyFromEnthalpy_)
162 {
163     dc = omegaPreq;
164 }
165 else
166 {
167     forAll(dc, i)
168     {
169         dc[i] = omegaPreq[i];
170     }
171 } */
172 }
173 }
174
175 // Don't allow the time-step to change more than a factor of 2
176 deltaTMin = min(deltaTMin, 2*deltaT);
177
178 return deltaTMin;
179 }

```

A.4 Função *jacobian(t, c, dcdt, dfdc)*

```

1  template<class CompType, class SolidThermo, class GasThermo>
2  void Foam::ODESolidHeterogeneousChemistryModel<CompType, SolidThermo, GasThermo
   >::jacobian
3  (
4      const scalar t,
5      const scalarField& c,
6      scalarField& dcdt,
7      scalarSquareMatrix& dfdc
8  ) const
9  {
10     label cellI = cellCounter_;
11
12     scalar T = c[nSpecie_];
13
14     scalarField c2(nSpecie_, 0.0);
15
16     for (label i=0; i<nSolids_; i++)
17     {
18         c2[i] = max(c[i], 0.0);
19     }
20
21     for (label i=0; i<nEqns(); i++)
22     {
23         for (label j=0; j<nEqns(); j++)
24         {
25             dfdc[i][j] = 0.0;
26         }
27     }
28
29     scalarField omegaPreq(omega(c2,T,0.0));
30     if (solidReactionEnergyFromEnthalpy_)
31     {
32         dcdt = omegaPreq;
33     }
34     else
35     {
36         forAll(dcdt, i)
37         {
38             dcdt[i] = omegaPreq[i];
39         }
40     }
41
42     for (label ri=0; ri<reactions_.size(); ri++)
43     {
44         const solidHeterogeneousReaction& R = reactions_[ri];
45
46         scalar kf0 = R.kf(T, 0.0, c2);
47
48     /*-----*/
49
50     scalar massCoefficient = 1;
51     scalar massStoRatio = 1;
52
53     scalar substrates = 0;

```

```

54     scalar products = 0;
55
56     scalar solidSubstrates = 0;
57     scalar solidProducts = 0;
58
59     forAll(R.grhs(), g)
60     {
61         label gi = R.grhs()[g];
62         products += gasThermo_[gi].W() * R.grhsSto()[g];
63     }
64
65
66     forAll(R.glhs(), g)
67     {
68         label gi = pyrolysisGases_[gasPhaseGases_[R.glhs()[g]].name()];
69         substrates += gasThermo_[gi].W()*R.glhsSto()[g];
70     }
71
72     forAll(R.slhs(), s)
73     {
74         solidSubstrates += R.slhsSto()[s];
75     }
76
77     forAll(R.srhs(), s)
78     {
79         solidProducts += R.srhsSto()[s];
80     }
81
82     massCoefficient *= products/(products - substrates);
83     massStoRatio *= substrates/products;
84
85     scalar sr = solidProducts/solidSubstrates;
86
87     /*-----*/
88
89     forAll(R.slhs(), j)
90     {
91         label sj = R.slhs()[j];
92         scalar kf = kf0;
93         forAll(R.slhs(), i)
94         {
95
96             label si = R.slhs()[i];
97             scalar exp = R.nReact();
98             scalar expGas = R.nReactGas(); //Adicionado
99             if (i == j)
100            {
101
102                if (c2[si] == 0.0)
103                {
104                    kf *= 0.0;
105                }
106                if (c2[si]>SMALL)
107                {
108                    kf *= c2[si]*exp*Foam::pow(Ys_[si][celli] + VSMALL, exp -
109                        2.0);

```



```

110         else
111         {
112             kf = 0.0;
113         }
114         if (R.glhs().size() > 0)
115         {
116             forAll(R.glhs(), i)
117             {
118                 kf *= Foam::pow(gasPhaseGases_[R.glhs()[i]].
119                     internalField()[celli], expGas); //Modificado
120             }
121         }
122         else
123         {
124             Info<< "Solid reactions have only elements on slhs"
125                 << endl;
126             kf = 0.0;
127         }
128     }
129
130     forAll(R.slhs(), i)
131     {
132         label si = R.slhs()[i];
133         dfdc[si][sj] -= kf*R.slhsSto()[i]/solidSubstrates; //Antes era
134                                                     -= kf apenas
135     }
136     forAll(R.srhs(), i)
137     {
138         label si = R.srhs()[i];
139         dfdc[si][sj] += kf*R.srhsSto()[i]/solidSubstrates; //Antes era
140                                                     += kf apenas
141     }
142     forAll(R.grhs(), i) //Acrescentado o loop
143     {
144         label gi = R.grhs()[i];
145         dfdc[gi + nSolids_][sj] += kf*(1.0 - sr)*massCoefficient*
146             gasThermo_[gi].W()*R.grhsSto()[i]/products;
147     }
148     forAll(R.glhs(), i) //Acrescentado o loop
149     {
150         label gi = pyrolysisGases_[gasPhaseGases_[R.glhs()[i]].name()];
151         dfdc[gi + nSolids_][sj] -= kf*(1.0 - sr)*massCoefficient*
152             massStoRatio*gasThermo_[gi].W()*R.glhsSto()[i]/substrates;
153     }
154 }
155
156 /*-----*/
157
158 if (R.glhs().size() > 0) //Acrescentado o if
159 {
160     forAll(R.glhs(), j) //Acrescentado este loop
161     {
162         label sj = pyrolysisGases_[gasPhaseGases_[R.glhs()[j]].name()];
163         scalar kf = kf0;
164         forAll(R.slhs(), i)
165         {

```

```

162
163     label si = R.slhs()[i];
164     scalar exp = R.nReact();
165     scalar expGas = R.nReactGas(); //Adicionado
166
167     if (c2[si]>SMALL)
168     {
169         if (gasPhaseGases_[R.glhs()[j]].internalField()[celli]>
170             SMALL)
171         {
172             kf *= c2[si]*expGas*Foam::pow(Ys_[si][celli], exp - 1.0)
173                 *Foam::pow(gasPhaseGases_[R.glhs()[j]].internalField
174                     ([celli], expGas - 1.0);
175         }
176         else
177         {
178             kf = 0.0;
179         }
180     }
181     else
182     {
183         kf = 0.0;
184     }
185 }
186
187 forAll(R.slhs(), i)
188 {
189     label si = R.slhs()[i];
190     dfdc[si][sj + nSolids_] -= kf*R.slhsSto()[i]/solidSubstrates;
191 }
192 forAll(R.srhs(), i)
193 {
194     label si = R.srhs()[i];
195     dfdc[si][sj + nSolids_] += kf*R.srhsSto()[i]/solidSubstrates;
196 }
197 forAll(R.grhs(), i)
198 {
199     label gi = R.grhs()[i];
200     dfdc[gi + nSolids_][sj + nSolids_] += kf*(1.0 - sr)*
201         massCoefficient*gasThermo_[gi].W()*R.grhsSto()[i]/products;
202 }
203 forAll(R.glhs(), i)
204 {
205     label gi = pyrolysisGases_[gasPhaseGases_[R.glhs()[i]].name()];
206     dfdc[gi + nSolids_][sj + nSolids_] -= kf*(1.0 - sr)*
207         massCoefficient*massStoRatio*gasThermo_[gi].W()*R.glhsSto()[i]
208         /substrates;
209 }
210 }
211 }
212
213 /*-----*/

```

```
213 // calculate the dcdT elements numerically
214 scalar delta = 1.0e-8;
215
216 scalarField dcdT0(dcdt);
217 omegaPreq = omega(c2,T - delta ,0);
218 if (solidReactionEnergyFromEnthalpy_)
219 {
220     dcdT0 = omegaPreq;
221 }
222 else
223 {
224     forAll(dcdT0,i)
225     {
226         dcdT0[i] = omegaPreq[i];
227     }
228 }
229
230
231 scalarField dcdT1(dcdt);
232 omegaPreq = omega(c2,T + delta ,0);
233 if (solidReactionEnergyFromEnthalpy_)
234 {
235     dcdT1 = omegaPreq;
236 }
237 else
238 {
239     forAll(dcdT1,i)
240     {
241         dcdT1[i] = omegaPreq[i];
242     }
243 }
244
245 for (label i=0; i<nEqns(); i++)
246 {
247     dfdc[i][nSpecie_] = 0.5*(dcdT1[i] - dcdT0[i])/delta;
248 }
249
250 }
```

APÊNDICE B – Mudanças para usar a função *calculate()*

B.1 Arquivo *ODESolidHeterogeneousChemistryModel.C* - Função *calculate()*

```

1 template<class CompType, class SolidThermo, class GasThermo>
2 void Foam::ODESolidHeterogeneousChemistryModel<CompType, SolidThermo, GasThermo
   >::calculate()
3 {
4
5     const volScalarField rho
6     (
7         IObject
8         (
9             "rho",
10            this->time().timeName(),
11            this->mesh(),
12            IObject::NO_READ,
13            IObject::NO_WRITE,
14            false
15        ),
16        this->solidThermo().rho()
17    );
18
19    if (this->mesh().changing())
20    {
21        forAll(RRs_, i)
22        {
23            RRs_[i].setSize(rho.size());
24        }
25        forAll(RRg_, i)
26        {
27            RRg_[i].setSize(rho.size());
28        }
29    }
30
31    forAll(RRs_, i)
32    {
33        RRs_[i] = 0.0;
34    }
35    forAll(RRg_, i)
36    {
37        RRg_[i] = 0.0;
38    }
39
40    /******
41    forAll(shReactionHeat_, cellI)

```

```

42     {
43         shReactionHeat_[celli] = 0.0;
44     }
45     /*****/
46
47     if (this->chemistry_)
48     {
49         forAll(rho, celli)
50         {
51             cellCounter_ = celli;
52
53             const scalar delta = this->mesh().V()[celli];
54
55             if (reactingCells_[celli])
56             {
57                 scalar rhoi = rho[celli];
58                 scalar Ti = this->solidThermo().T()[celli];
59
60                 scalarField c(nSpecie_, 0.0);
61                 scalarField c0(nSpecie_, 0.0); //Acrescentado c0
62
63                 /*****/
64                 scalarField omegaPreq(omega(c0, Ti, 0.0));
65                 /*****/
66
67                 for (label i=0; i<nSolids_; i++)
68                 {
69                     c[i] = rhoi*Ys_[i][celli]*delta;
70                 }
71                 // it is not seen from outside the class so no need to wrap it
72                 const scalarField dcdt = omega(c, Ti, 0.0, true);
73
74                 forAll(RRs_, i)
75                 {
76                     RRs_[i][celli] = dcdt[i]/delta;
77                 }
78
79                 forAll(RRg_, i)
80                 {
81                     RRg_[i][celli] = dcdt[nSolids_ + i]/delta;
82                 }
83
84                 /*****/
85                 if (not solidReactionEnergyFromEnthalpy_)
86                 {
87                     shReactionHeat_[celli] = omegaPreq[nEqns()]/(delta);
88                     //shReactionHeat_[celli] = omegaPreq[nEqns()]/(deltaT*delta);
89                 }
90                 /*****/
91             }
92         }
93     }
94 }
95 }

```

B.2 Arquivo `volPyrolysis.C` - Função `evolveRegion()`

```

1 void volPyrolysis::evolveRegion()
2 {
3     forAll(porosity_ , cellI)
4     {
5         oneMinusPorosity_[ cellI ] = 1.0 - porosity_[ cellI ];
6     }
7
8     if (equilibrium_)
9     {}
10    else
11    {
12        HTC_ = HTC();
13    }
14
15    /*    timeChem_ = solidChemistry_ -> solve
16        (
17            time_.value() - time_.deltaTValue() ,
18            time_.deltaTValue()
19        ); */
20
21    /******
22        solidChemistry_ -> calculate();
23    *****/
24
25    heatUpGas_ = heatUpGasCalc()();
26    chemistrySh_ = solidChemistry_ -> Sh()();
27    solveSpeciesMass();
28    evolvePorosity();
29    solidThermo_.correct();
30    forAll(rho_ , cellI)
31    {
32        rho_[ cellI ] = porosity_[ cellI ] * rho_[ cellI ];
33        if (whereIsNot_[ cellI ] == 1)
34        {
35            rho_[ cellI ] = 0;
36        }
37    }
38
39    if (equilibrium_)
40    {
41        solveEnergy();
42    }
43    else
44    {
45        for (int nonOrth=0; nonOrth<=nNonOrthCorr_ ; nonOrth++)
46        {
47            solveEnergy();
48        }
49    }
50
51    calculateMassTransfer();
52
53    info();
54 }

```

Anexos

ANEXO A – Arquivo

biomassGasificationFoam.C

```

1  /*-----*/
2  =====
3  \ \      F i e l d      |   OpenFOAM: The Open Source CFD Toolbox
4  \ \      O p e r a t i o n   |
5  \ \      A n d             |   Copyright held by original author
6  \ \      M a n i p u l a t i o n |
7  -----
8  License
9      This file is part of OpenFOAM.
10
11     OpenFOAM is free software: you can redistribute it and/or modify it
12     under the terms of the GNU General Public License as published by
13     the Free Software Foundation, either version 3 of the License, or
14     (at your option) any later version.
15
16     OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
17     ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
18     FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License
19     for more details.
20
21     You should have received a copy of the GNU General Public License
22     along with OpenFOAM. If not, see <http://www.gnu.org/licenses/>.
23
24 Application
25     biomassGasificationFoam
26
27 Description
28     Solver for combustion with chemical reactions and zonal porosity created by
29     Kamil Kwiatkowski & Pawel Zuk (biomassgasification.eu project)
30 \*-----*/
31
32 #include "fvCFD.H"
33 #include "turbulenceModel.H"
34 #include "psiChemistryModel.H"
35 #include "chemistrySolver.H"
36 #include "porousReactingZone.H"
37 #include "multivariateScheme.H"
38 #include "pimpleControl.H"
39 #include "basicHGSSolidThermo.H"
40 #include "solidChemistryModel.H"
41 #include "mapDistribute.H"
42 #include "heterogeneousPyrolysisModel.H"
43 #include "heterogeneousRadiationModel.H"
44
45 // * * * * *
46
47 int main(int argc, char *argv[])

```



```

48 {
49     #include "setRootCase.H"
50     #include "createTime.H"
51     #include "createMesh.H"
52     #include "readChemistryProperties.H"
53     #include "readGravitationalAcceleration.H"
54     #include "createFields.H"
55     #include "initContinuityErrs.H"
56     #include "readTimeControls.H"
57     #include "compressibleCourantNo.H"
58     #include "setInitialDeltaT.H"
59     #include "createPyrolysisModel.H"
60     #include "readPyrolysisTimeControls.H"
61     #include "createPorosity.H"
62     #include "createHeterogeneousRadiationModel.H"
63
64     pimpleControl pimple(mesh);
65
66     // * * * * *
67
68
69     Info<< "\nStarting time loop\n" << endl;
70
71     while (runTime.run())
72     {
73         #include "readTimeControls.H"
74         #include "compressibleCourantNo.H"
75         #include "setDeltaT.H"
76
77         #include "solidRegionDiffusionNo.H"
78         #include "setMultiRegionDeltaT.H"
79
80         runTime++;
81         Info<< "Time = " << runTime.timeName() << nl << endl;
82
83         #include "radiation.H"
84
85         pyrolysisZone.evolve();
86
87         #include "updateOneMinusPorosityF.H"
88
89         #include "chemistry.H"
90         #include "rhoEqn.H"
91         while (pimple.loop())
92         {
93             #include "UEqn.H"
94             #include "YEqn.H"
95             #include "hsEqn.H"
96
97             // — PISO loop
98             while (pimple.correct())
99             {
100                 #include "pEqn.H"
101             }
102             if (pimple.turbCorr())
103             {
104                 turbulence->correct();

```

```
105     }
106   }
107
108   rho = thermo.rho();
109
110   if (runTime.write())
111   {
112     chemistry.dQ().write();
113   }
114
115   runTime.write();
116
117   Info<< "ExecutionTime = " << runTime.elapsedCpuTime() << " s"
118     << "   ClockTime = " << runTime.elapsedClockTime() << " s"
119     << nl << endl;
120
121   }
122
123   Info<< "End\n" << endl;
124
125   return 0;
126 }
127
128
129 // ***** //
```