LUIZ PAULO BORGES MIRANDA

SIMULAÇÃO NUMÉRICA DE ESCOAMENTOS VISCOPLÁSTICOS COM TRANSFERÊNCIA DE CALOR ATRAVÉS DO SOFTWARE OPENFOAM

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA MECÂNICA
2019

LUIZ PAULO BORGES MIRANDA

SIMULAÇÃO NUMÉRICA DE ESCOAMENTOS VISCOPLÁSTICOS COM TRANSFERÊNCIA DE CALOR ATRAVÉS DO SOFTWARE OPENFOAM

Dissertação apresentada ao Programa de Pós-graduação em Engenharia Mecânica da Universidade Federal de Uberlândia, como parte dos requisitos para a obtenção do título de **MESTRE EM ENGENHARIA MECÂNICA**.

Área de concentração: Transferência de calor e mecânica dos fluidos.

Orientador: Prof. Dr. Daniel Dall'Onder dos Santos

Uberlândia - MG
2019

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema de Bibliotecas da UFU, MG, Brasil.

M672s 2019

Miranda, Luiz Paulo Borges, 1991-
Simulação numérica de escoamentos viscoplásticos com transferência de calor através do software OpenFOAM [recurso eletrônico] / Luiz Paulo Borges Miranda. - 2019.

Orientador: Daniel Dall'Onder dos Santos.
Dissertação (mestrado) - Universidade Federal de Uberlândia, Programa de Pós-Graduação em Engenharia Mecânica.
Modo de acesso: Internet.
Disponível em: http://dx.doi.org/10.14393/ufu.di.2019.55
Inclui bibliografia.
Inclui ilustrações.

1. Engenharia mecânica. 2. Mecânica dos fluidos. 3. Calor - Transmissão. 4. Simulação por computador. 5. Viscosidade. I. Santos, Daniel Dall'Onder dos, 1986-, (Orient.). II. Universidade Federal de Uberlândia. Programa de Pós-Graduação em Engenharia Mecânica. III. Título.

CDU: 621

Rejâne Maria da Silva – CRB6/1925

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

ATA DE DEFESA

Programa de Pós-Graduação em:	Engenharia Mecânica				
Defesa de:	Dissertação de Mestrado Acadêmico / 546 / COPEM				
Data:	doze de julho de dois mil e dezenove	Hora de início:	[09:00]	Hora de encerramento:	[11:10]
Matrícula do Discente:	11722EMC008				
Nome do Discente:	Luiz Paulo Borges Miranda				
Título do Trabalho:	Simulação Numérica de Escoamentos Viscoplásticos com Transferência de Calor através do Software OPENFOAM				
Área de concentração:	Transferência de Calor e Mecânica dos Fluidos				
Linha de pesquisa:	Dinâmica dos Fluidos e Transferência de Calor				
Projeto de Pesquisa de vinculação:					

Reuniu-se no Anfiteatro A do Bloco 5O, Campus Santa Mônica, da Universidade Federal de Uberlândia, a Banca Examinadora, designada pelo Colegiado do Programa de Pós-graduação em Engenharia Mecânica, assim composta: Professores Doutores: Francisco José de Souza - FEMEC/UFU e Daniel Dall'Onder dos Santos - FEMEC/UFU orientador(a) do(a) candidato(a). Ressalta-se que a Profa. Dra. Flávia Schwarz Franceschini Zinani - UNISINOS participou da defesa por meio de Skype desde a cidade de São Leopoldo/RS e os demais membros da banca e o aluno participaram *in loco.*

Iniciando os trabalhos o(a) presidente da mesa, Dr(a). Daniel Dall'Onder dos Santos, apresentou a Comissão Examinadora e o candidato(a), agradeceu a presença do público, e concedeu ao Discente a palavra para a exposição do seu trabalho. A duração da apresentação do Discente e o tempo de arguição e resposta foram conforme as normas do Programa.

A seguir o senhor(a) presidente concedeu a palavra, pela ordem sucessivamente, aos(às) examinadores(as), que passaram a arguir o(a) candidato(a). Ultimada a arguição, que se desenvolveu dentro dos termos regimentais, a Banca, em sessão secreta, atribuiu o resultado final, considerando o(a) candidato(a):

Aprovado.

Esta defesa faz parte dos requisitos necessários à obtenção do título de Mestre .

O competente diploma será expedido após cumprimento dos demais requisitos, conforme as normas do Programa, a legislação pertinente e a regulamentação interna da UFU.

Nada mais havendo a tratar foram encerrados os trabalhos. Foi lavrada a presente ata que após lida e achada conforme foi assinada pela Banca Examinadora.

Documento assinado eletronicamente por **Daniel Dall'Onder dos Santos**, **Professor(a) do Magistério Superior**, em 12/07/2019, às 11:13, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Francisco José de Souza**, **Professor(a) do Magistério Superior**, em 12/07/2019, às 11:18, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Flavia Zinani**, **Usuário Externo**, em 12/07/2019, às 11:19, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://www.sei.ufu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **1388798** e o código CRC **CD29A1FA**.

"Se viveram no mundo, eles deveriam ver e tocar, ouvir, amar e aprender coisas."

Sem-Nome

AGRADECIMENTOS

Primeiramente agradeço a Deus por todas as oportunidades que me proporcionou ao longo de toda minha vida, e por me guiar dando força, saúde e disposição para que eu pudesse concluir este trabalho.

Ao meu orientador Prof. Dr. Daniel Dall'Onder dos Santos, pelo auxílio, ajuda e confiança, durante todos esses anos. Expresso minha total gratidão e apreço não somente aos conhecimentos e experiências compartilhados.

Aos meus pais, Sérgio Miranda e Luziely de Fátima Borges Miranda, aos meus irmãos Ana Luísa Borges Miranda e Carlos Eduardo Borges Miranda, e aos meus amigos que sempre acreditaram em em mim, dirigiram suas boas energias, pensamentos, orações, palavras e intensões em favor da concretização deste trabalho.

Ao professor Francisco José de Souza, que me ajudou em diversos momentos dessa caminhada.

À todos os professores e funcionários do Programa de Pós-Graduação em Engenharia Mecânica da Universidade Federal de Uberlândia pela oportunidade de realizar este curso.

À CAPES pelo apoio financeiro através da bolsa de estudos.

Miranda, L. P. B., **Simulação Numérica de Escoamentos Viscoplásticos com Transferência de Calor Através do Software OpenFOAM®.** 2019. 216 f. Dissertação de Mestrado, Universidade Federal de Uberlândia, Uberlândia.

Resumo

Este trabalho tem por principal objetivo desenvolver, testar a qualidade dos resultados e aplicar uma rotina para o software livre OpenFOAM® que seja capaz de resolver escoamentos de fluidos não newtonianos segundo o modelo de fluido newtoniano generalizado. Outros *softwares* com as mesmas características costumam não estar acessíveis à grande parte da comunidade científica, seja pelo custo de obtenção da sua licença, seja por serem restritos à determinados tipos de escoamento. Dessa forma, esse trabalho busca desenvolver uma ferramenta barata e versátil para o estudo dessa classe de escoamentos. Para atingir esse objetivo, em um primeiro momento, as edições no código são apresentadas e discutidas, em especial devido ao importante obstáculo numérico imposto por alguns comportamentos não newtonianos, como o comportamento viscoplástico, por exemplo. Em seguida, a qualidade das rotinas foi testada, comparando os resultados obtidos em simulações de escoamento de convecção natural em recipientes quadrados com os resultados encontrados na literatura. Os resultados obtidos com a rotina proposta apresentaram uma alta correlação com os resultados da literatura, e assim as rotinas propostas foram utilizadas para resolver escoamentos de fluidos viscoplásticos em uma geometria de um canal planar com uma expansão seguida de uma contração. Os grupos adimensionais que definem este escoamento são o número de Reynolds, número de Pradtl, *jump number*, *plastic number*, o índice de comportamento e o patamar de viscosidade para altas taxas de cisalhamento. Dentre estes, foi analisada a influência do número de Reynolds, do *plastic number* e do índice de comportamento sobre o escoamento. Os resultados foram obtidos em termos do número de Nusselt médio, *head loss* e a eficiência de deslocamento.

Palavras Chave: viscoplástico, OpenFOAM®, fluido newtoniano generalizado, canal com expansão e contração

Miranda, L. P. B., **Numerical Simulation of Viscoplastic Flows With Heat Transfer With the Software OpenFOAM®.** 2019. 216 p. Master Thesis, Universidade Federal de Uberlândia, Uberlândia.

ABSTRACT

The main objective of this work is to develop, validate and apply an OpenFOAM® routine able to solve non-Newtonian fluid flows, adopting the generalized Newtonian model. Others software with the same characteristics are often not accessible to the majority of the scientific community, either because they are restricted to certain types of flows or because of the cost of their license. Thus, this work seeks to develop an inexpensive and versatile tool to study of this class of flows. In order to achieve this goal, the development of the code is presented and discussed, especially due to the important numerical obstacle imposed by some non-Newtonian behaviors, such as viscoplastic behavior, for example. Then, the routines were validated, comparing the results obtained in simulations of natural convection flows inside a square enclosure with the results found in the literature. The results obtained with the proposed routine presented a high correlation with the results of the literature, and thus the proposed routines were used to solve viscoplastic fluid flows inside a planar channel with an expansion followed by an abrupt contraction. The dimensionless groups that define this flow are the Reynolds number, the Pradtl number, the jump number, the plastic number, the behavior index and the viscosity plateau for high shear rates. Among these, the influence of the Reynolds number, of the plastic number, and of the behavior index were analyzed. The results were obtained in terms of the mean Nusselt number, the head loss and the displacement efficiency.

Keywords: viscoplastic behavior, OpenFOAM®, generalized Newtonian fluid, expansion and abrupt contraction

Lista de Figuras

Lista de Tabelas

Lista de símbolos, abreviações e siglas

Símbolos

a	*Theoretical order of accuracy*	$[-]$
A	Matriz com os termos da diagonal de **M**	$[-]$
A_c	Área da cavidade	$[m^2]$
$A_{c,0}$	Área aparentemente não escoada dentro da cavidade	$[m^2]$
$A_{c,sta}$	Área aparentemente não escoada estacionária dentro da cavidade	$[m^2]$
b	Vetor dos termos independeres de ϕ	$[-]$
Bn	Número de Bingham	$[-]$
c	Campo vetorial genérico	$[-]$
D	Tensor taxa de deformação	$[1/s]$
e_{ext}	Erro relativo em relação ao valor extrapolado	$[-]$
F	Fluxo da propriedade ϕ sobre uma determinada face	$[-]$
g	Vetor gravidade	$[m/s^2]$
H'	Matriz com os termos fora da diagonal de **M**	$[-]$
$H1$	Altura do canal	$[m]$
$H2$	Altura da cavidade	$[m]$
h^*_{teo}	Altura adimensional teórica de um volume de controle	$[-]$
Hl	*Head loss*	$[-]$

I_D	Primeiro invariante do tensor taxa de deformação	$[1/s]$
II_D	Segundo invariante do tensor taxa de deformação	$[1/s^2]$
III_D	Terceiro invariante do tensor taxa de deformação	$[1/s^3]$
J	Jump number	$[-]$
K	Índice de consistência	$[Pa \cdot s^n]$
L	Comprimento do lado da cavidade	$[m]$
$L1$	Comprimento da cavidade	$[m]$
$L2$	Comprimento do canal à montante e à jusante	$[m]$
L_c	Comprimento característico	$[m]$
m	Parâmetro de regularização	$[-]$
M	Matriz dos coeficientes lineares de ϕ	$[-]$
n	Índice de comportamento	$[-]$
n	Vetor normal à superfície do volume de controle	$[-]$
n_e	Número de elementos em cada direção	$[-]$
Nu	Número de Nusselt	$[-]$
p	Pressão	$[Pa]$
P	Campo de pressão	$[Pa]$
P'	Previsão do campo de pressão	$[Pa]$
P*	Correção do campo de pressão	$[Pa]$
p^*_{10}	Diferença entre a pressão adimensional medida nos pontos $(x^*, y^*) = (-10, 0)$ e $(x^*, y^*) = (10, 0)$	$[-]$
Pl	Plastic number	$[-]$
Pr	Número de Prandtl	$[-]$

q_ϕ	Termo fonte de ϕ	$[-]$
r	Razão de um comprimento representativo de duas malhas diferentes	$[-]$
r^2	Coeficiente de determinação	$[-]$
r_e	Razão de expansão	$[-]$
r_{max}	Resíduo normalizado máximo	$[-]$
r_{norm}	Resíduo normalizado	$[-]$
Ra	Número de Rayleigh	$[-]$
Re	Número de Reynolds	$[-]$
$RMSE$	Raiz do erro quadrático médio	$[-]$
s	Número de pontos utilizados na comparação dos resultados	$[-]$
S	Superfície do volume de controle	$[m^2]$
S_c	Termo constante do termo fonte	$[-]$
S_p	Coeficiente linear do termo fonte	$[-]$
S_{Nu}	Superfície adotada para o cálculo de $\overline{Nu}$	$[m^2]$
SSE	Soma dos quadrados dos resíduos	$[-]$
t	Tempo	$[s]$
T	Temperatura	$[K]$
T_C	Temperatura da parede fria	$[K]$
T_H	Temperatura da parede quente	$[K]$
T_{ref}	Temperatura de referência	$[K]$
T_{wall}	Temperatura na parede	$[K]$
u	Vetor velocidade	$[m/s]$
U	Campo de vetores velocidade	$[m/s]$

U'	Previsão do campo de vetores velocidade	$[m/s]$
U*	Correção do campo de vetores velocidade	$[m/s]$
u_x	Velocidade na direção x	$[m/s]$
V	Volume do volume de controle	$[m^3]$
V_c	Velocidade característica	$[m/s]$
x	Vetor posição	$[m]$
x	Posição horizontal	$[m]$
y	Posição vertical	$[m]$

Letras gregas

α	Difusividade térmica	$[m^2/s]$
α_ϕ	Fator de relaxação da propriedade ϕ	$[-]$
β	Termo de expansão térmico	$[1/k]$
$\dot{\gamma}$	Magnitude do tensor de taxa deformação	$[1/s]$
$\dot{\gamma}_0$	Taxa de deformação de transição entre o patamar de alta viscosidade e a região de transição	$[1/s]$
$\dot{\gamma}_1$	Taxa de deformação de transição entre a região de transição e a região *power-law*	$[1/s]$
$\dot{\gamma}_2$	Taxa de deformação de transição entre a região *power-law* e o patamar para altas taxas de deformação	$[1/s]$
$\dot{\gamma}_{ref}$	Taxa de deformação de referência	$[1/s]$
$\dot{\gamma}_{xy}$	Taxa de deformação normal ao eixo x no sentido do eixo y	$1/s$
Γ_ϕ	Difusividade de ϕ	$[-]$
$\frac{\delta_{min,cell}}{L}$	Comprimento adimensional mínimo de elemento	$[-]$
Δp^*_{max}	Diferença entre a maior e a menor pressão adimensional em todo o domínio	$[-]$

Δp_c	Diferença de pressão entre um ponto a montante e um ponto a jusante da cavidade	$[Pa]$
Δp_s	Diferença de pressão entre dois pontos de um canal simples	$[Pa]$
Δt	Termo da discretização temporal	$[s]$
ΔT	Diferença de temperatura	$[k]$
η	Viscosidade aparente	$[Pa \cdot s]$
η_0	Platô de viscosidade aparente para baixas taxas de deformação	$[Pa \cdot s]$
η_{nom}	Viscosidade nominal	$[Pa \cdot s]$
η_∞	Platô de viscosidade aparente para altas taxas de deformação	$[Pa \cdot s]$
η^*_∞	Platô de viscosidade adimensional para altas taxas de deformação	$[-]$
θ	Temperatura adimensional	$[-]$
θ_C	Temperatura adimensional da parede fria	$[-]$
θ_H	Temperatura adimensional da parede quente	$[-]$
μ	Viscosidade newtoniana	$[Pa \cdot s]$
μ_B	Viscosidade plática do modelo de Bingham	$[Pa \cdot s]$
ρ	Volume específico	kg/m^3
σ_{xx}	Tensão normal no sentido do eixo x	$[Pa]$
τ	Tensão de cisalhamento	$[Pa]$
$\boldsymbol{\tau}$	Tensor tensão extra total	$[Pa]$
τ_0	Tensão limite de escoamento	$[Pa]$
τ_{xy}	Tensão cisalhante normal ao eixo x no sentido do eixo y	$[Pa]$
ϕ	Propriedade genérica do fluido	$[-]$
$\boldsymbol{\phi}$	Campo solução da propriedade ϕ	$[-]$

ϕ_{calc}	Propriedade ϕ calculada antes da aplicação do fator de relaxação	$[-]$
ϕ_{ext}	Valor extrapolado da propriedade ϕ	$[-]$
$\widehat{\phi_i}$	Valor da propriedade ϕ no ponto *i* obtido com as rotinas propostas	$[-]$
$\phi_{lit,i}$	Valor da propriedade ϕ no ponto *i* encontrado na literatura	$[-]$
Φ_{de}	Eficiência de deslocamento	$[-]$
Φ_{sta}	Eficiência de deslocamento não estacionária	$[-]$
Ω	Volume do volume de controle	$[m^3]$

Sobrescritos e subscritos

$*$	Indica uma variável adimensional
$1, 2\ e\ 3$	Indicam a malha analisada, sendo 1 a malha mais fina e 3 a mais grossa
c	Indica um propriedade característica do escoamento
E	Indica que a propriedade é avaliada no centroide do volume de controle à direita do volume de controle analisado
e	Indica que a propriedade é avaliada na face entre o centroide P e E
er	Indica o erro relativo de um parâmetro
f	Indica que a propriedade é avaliada em cada face
fc	Indica que a propriedade é avaliada no centroide de cada face
j	Indica o tempo ou a iteração da simulação
max	Indica o valor máximo de um conjunto de valores
min	Indica o valor mínimo de um conjunto de valores
P	Indica que a propriedade é avaliada no centroide do volume de controle analisado
W	Indica que a propriedade é avaliada no centroide do volume de controle à esquerda do volume de controle analisado

w Indica que a propriedade é avaliada na face entre o centroide P e W

wf Indica que a propriedade foi avaliada no contato do fluido com a parede

Operações matemáticas

$\nabla \cdot \mathbf{C}$ Divergente da matriz **C**

$\nabla \mathbf{C}$ Gradiente da matriz **C**

$\nabla^2 \mathbf{C}$ Laplaciano da matriz **C**

$\overline{\mathbf{C}}$ Média do campo **C**

$det(\mathbf{C})$ Determinante da matriz **C**

$tr(\mathbf{C})$ Traço da matriz **C**

Abreviações e siglas

CFD Computational fluid dynamics

FD *Finite difference*

FE *Finite element*

FOAM *Field Operation and Manipulation*

FV *Finite volume*

GNF *Generalized Newtonian Fluid*

OpenFOAM *Open Field Operation and Manipulation*

PISO *Pressure-Implicit with Splitting of Operators*

SIMPLE *Semi-Implicit Method for Pressure Linked Equations*

SMD Modelo de viscosidade aparente proposto por de Souza Mendes e Dutra (2004)

SUMÁRIO

CAPÍTULO I

INTRODUÇÃO

1.1 Motivação

Diversos escoamentos presentes na indústria apresentam comportamentos não newtonianos relevantes, como diferentes graus de viscoelasticidade, viscoplasticidade, tixotropia, entre outros. A lista de fluidos não newtonianos é extensa: tintas, colas, medicamentos, fluidos lubrificantes, fluidos abrasivos, cosméticos, produtos de limpeza e produtos alimentares são apenas alguns exemplos. Esse interesse econômico motivou diversos estudos nessa área, com o objetivo de se entender melhor esses comportamentos, modelando-os e desenvolvendo as bases para a simulação numérica desses escoamentos. Essa possibilidade de se simular escoamentos não newtonianos permite aumentar a eficiência e reduzir desperdícios em aplicações, além de ser uma técnica mais eficiente e mais barata que a construção de protótipos, por exemplo.

Porém, ainda que esse campo de estudo apresente um grande potencial a ser desenvolvido, o número de trabalhos nessa área é limitado quando comparado com fluidos newtonianos, por exemplo. Essa diferença se dá por uma série de fatores, tais como:

- Dificuldade numérica: alguns comportamentos newtonianos impõem sérias dificuldades na solução do escoamento, o que pode muitas vezes inviabilizar simulações mais complexas.

- Modelagem: é possível se identificar uma série de comportamentos não newtonianos diferentes, e uma série de modelos que buscam aproximar cada um desses comporta-

mentos.

- Acesso aos *Softwares*: os *softwares* usados para fazer simulações de escoamentos com propriedades não newtonianas podem ser divididos em dois grupos principais: os *softwares* comerciais e os softwares de código aberto:

  – Os *softwares* comerciais em geral são capazes de lidar com geometrias mais complexas, o que é interessante para aplicações industriais, porém podem apresentar obstáculos para se implementar novos modelos. Outro obstáculo importante é o preço para se obter a licença desses *softwares*. Os principais exemplos de *softwares* comerciais são: FLUENT®, COMSOL®, StarCCM+® e PHOENICS®.

  – Os *softwares* de código aberto não apresentem o obstáculo do custo da licença, mas costumam ter sua aplicação mais limitada a grupos mais restritos, como laboratórios que desenvolvem seus próprios códigos, por exemplo. Além desses softwares desenvolvidos *in house*, o principal exemplo desse grupo é o OpenFOAM®.

1.2 Objetivo

O principal objetivo deste trabalho é apresentar, verificar a qualidade e aplicar uma rotina de OpenFOAM® que seja não somente uma ferramenta eficiente para se resolver escoamentos não newtonianos, mas que também seja acessível a um público que tem pouca ou nenhuma experiência com este software. É possível dividir esse objetivo na seguinte lista de objetivos específicos:

- Elaboração e compilação de funções capazes de reproduzir o modelo de bi-viscosidade e o modelo SMD;

- Elaboração de um código *solver*, capaz de resolver escoamentos não-isotérmicos em regime permanente adotando a hipótese de fluido newtoniano generalizado e a aproximação de Boussinesq. A equação de balanço de massa, da quantidade de movimento linear e da energia devem ser resolvidas;

- Configuração das etapas de pré e pós-processamento, discutindo as soluções empregadas;

- Verificação da validade das rotinas desenvolvidas, comparando os resultados obtidos com outros trabalhos da literatura;
- Aplicação dos códigos desenvolvidos.

1.3 Justificativa

Dada a demanda da comunidade científica por uma ferramenta acessível, eficiente e barata para a solução de escoamentos não newtonianos, este trabalho propõe uma solução adotando o *software* OpenFOAM®. O OpenFOAM® é um *software* livre, com rotinas bem estabelecidas e que se destaca pela sua capacidade de customização. Desta forma, este trabalho busca explorar o potencial do OpenFOAM® para atender à demanda da comunidade científica.

1.4 Organização da Dissertação

Esta dissertação é dividida da seguinte forma:

- Capítulo 1: São apresentadas as motivações e objetivos deste trabalho, assim como sua organização
- Capítulo 2: Uma revisão bibliográfica sobre a modelagem da viscosidade aparente é apresentada, como a classificação dos comportamentos não newtonianos, seus modelos e trabalhos que os adotam.
- Capítulo 3: Este capítulo é composto por uma revisão bibliográfica sobre os principais componentes de uma simulação numérica, tais como os principais métodos e aproximações adotados.
- Capítulo 4: São apresentadas as edições e comandos necessários para se compilar a rotina *solver* e as duas funções para se modelar a viscosidade aparente.
- Capítulo 5: Os parâmetros das etapas de pré-processamento e pós-processamento são discutidos.

- Capítulo 6: As rotinas propostas nos capítulos anteriores são validadas, comparando os resultados obtidos em uma simulação de escoamento de convecção natural em uma cavidade quadrada adotando dois modelos de viscosidade aparente: o modelo *power-law* e o modelo de Bingham, aproximado pelo modelo bi-viscosidade.
- Capítulo 7: As rotinas são então utilizadas para se resolver um escoamento de um canal com uma expansão seguida de uma contração, adotando o modelo SMD.

CAPÍTULO II

EQUAÇÕES CONSTITUTIVAS

Este capítulo tem por objetivo apresentar uma revisão bibliográfica sobre a modelagem da viscosidade. São apresentados os principais comportamentos da viscosidade e os diferentes modelos desenvolvidos para tentar equacionar esses comportamentos. Além de discutir as características e limitações de cada modelo, também é apresentada uma breve revisão bibliográfica dos trabalhos desenvolvidos utilizando esses modelos, a fim de ilustrar o estado da arte desse campo de estudo, além de ressaltar a importância do presente trabalho e estabelecer uma base de comparação para os resultados obtidos.

2.1 Definição de fluido newtoniano

A divisão entre fluidos newtonianos e não newtonianos é baseada em como o fluido se comporta sob a ação de uma tensão de cisalhamento. Para um fluido newtoniano incompressível em um escoamento unidirecional laminar, a tensão cisalhante é igual ao produto da taxa de deformação pela viscosidade do fluido. Ou seja:

$$\tau_{yx} = \mu \left(-\frac{du_x}{dy} \right) = \mu \dot{\gamma}_{yx} \tag{2.1}$$

onde τ é a tensão cisalhante, μ é a viscosidade do fluido, u_x é a velocidade na direção *x*, $\dot{\gamma}$ é a taxa de deformação, o primeiro índice de τ e de $\dot{\gamma}$ representa a direção normal do plano onde atua a tensão cisalhante, e o segundo índice representa a direção da tensão e do escoamento, para τ e $\dot{\gamma}$, respectivamente. O sinal negativo da expressão mostra que τ oferece uma resistência ao movimento. A Fig. 2.1 representa graficamente as tensões normais

e cisalhantes em um elemento cúbico de fluido em um escoamento.

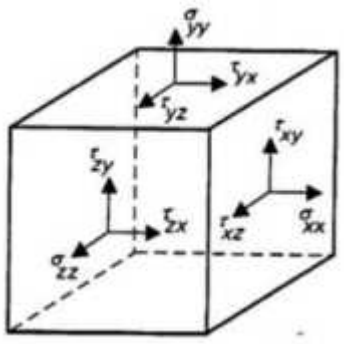


Figura 2.1 – Representação das tensões cisalhantes e normais em um elemento tridimensional de fluido. Fonte: Adaptado de Chhabra e Richardson (1999).

Para ser classificado como fluido newtoniano, o fluido precisa atender as seguintes condições simultaneamente (CHHABRA; RICHARDSON, 1999):

- A viscosidade μ é uma função apenas do fluido, da pressão e da temperatura;
- As tensões normais, representadas na Fig. 2.1 por σ_{xx}, σ_{yy} e σ_{zz}, tem seus módulos iguais.

Mesmo que um fluido qualquer atenda alguma dessas condições, mas não a outra, ele será considerado como não newtoniano. Como pode ser verificado pela Eq.(2.1), o gráfico tensão cisalhante (τ) pela taxa de deformação *($\dot{\gamma}$)* de um fluido newtoniano é uma reta, com coeficiente angular igual a μ e coeficiente linear igual a zero, ou seja: para um fluido newtoniano, o gráfico da tensão cisalhante pela taxa de deformação sempre passará pela origem. A Fig. **??** apresenta alguns exemplos de comportamento newtoniano.

2.2 Definição de fluido não newtoniano

Os fluidos não newtonianos são todos aqueles que apresentam comportamentos diferentes do exposto na Seção 2.1. Dadas as características geralmente apresentadas por escoamentos não newtonianos, é comum associar essa característica à hipótese de incompressibilidade. Para facilitar o estudo desses fluidos, eles são subdivididos em diversas categorias:

- Os fluidos independentes do tempo ou puramente viscosos: são aqueles fluidos onde o valor da taxa de deformação pode ser definido a partir da tensão cisalhante em um determinado ponto;

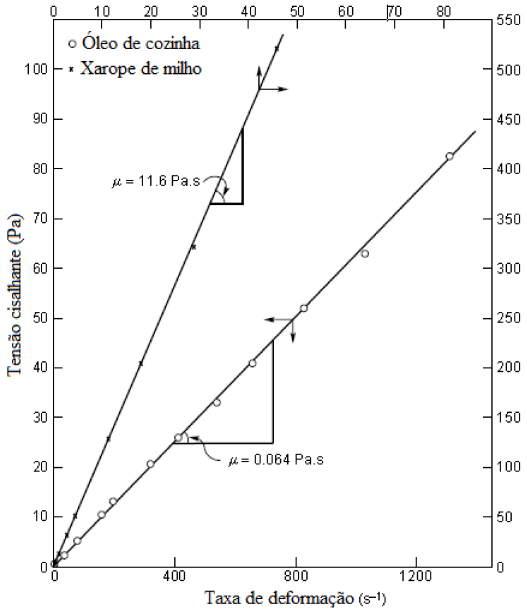


Figura 2.2 – Exemplos de curvas de tensão cisalhante por taxa de deformação para fluidos newtonianos. Fonte: Chhabra e Richardson (1999).

- Os fluidos dependentes do tempo: são aqueles fluidos onde o valor da taxa de deformação depende não somente da tensão cisalhante, mas também da duração do cisalhamento e do histórico cinemático;

- Os fluidos viscoelásticos: aqueles fluidos que apresentam tanto propriedades de um fluido ideal quanto propriedades de um sólido elástico, podendo realizar uma recuperação elástica parcial após a deformação.

Essa classificação é arbitrária e é comum que o mesmo fluido apresente características de mais de um desses grupos em função das condições do escoamento.

2.2.1 Fluidos dependentes do tempo

Os fluidos não newtonianos dependentes do tempo costumam ser divididos em dois principais grupos:

- Fluidos tixotrópicos ou afinantes: São caracterizados por possuírem uma estrutura que é quebrada em função do tempo e da taxa de deformação. Dessa forma, quando o fluido é submetido a uma tensão de cisalhamento constante, sua viscosidade aparente diminui com o passar do tempo. Uma vez que essa tensão seja retirada, as estruturas internas do fluido e sua viscosidade aparente passam por uma recuperação gradual;

- Fluido reopéticos ou espessantes: Semelhante ao comportamento tixotrópico, com a diferença que a viscosidade aparente aumenta com o tempo de cisalhamento.

Esses dois comportamentos estão representados na Fig. 2.3, onde são comparados com um fluido de viscosidade aparente independente do tempo.

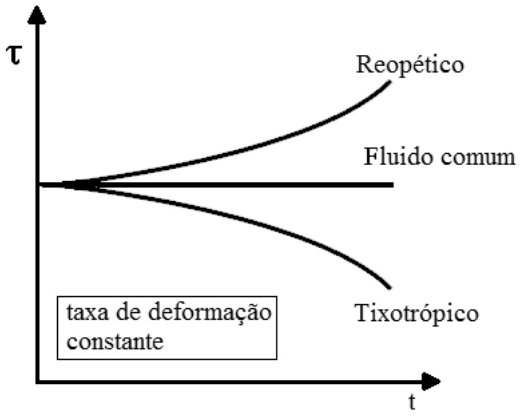


Figura 2.3 – Gráfico da tensão de cisalhamento em função do tempo. Comparação entre os comportamentos tixotrópicos, reopéticos e o comportamento independente do tempo. Fonte: Adaptado de White (2002).

2.2.2 Fluidos viscoelásticos

Os fluidos viscoelásticos apresentam um comportamento que reúne características de fluidos e de sólidos. Um sólido perfeitamente elástico sofre uma deformação finita sob ação de uma tensão, e retoma sua forma original quando essa tensão é removida. Além disso, a relação entre tensão e deformação é linear. Já um fluido ideal se deforma de forma contínua quando uma tensão é aplicada sobre ele, e uma vez que a tensão é removida, as tensões internas são dissipadas e ele não apresenta nenhuma recuperação elástica, ou seja: não apresenta nenhuma tendência a voltar a sua forma original. Os fluidos viscoelásticos apresentam uma combinação desses dois comportamentos, em função principalmente do tempo da aplicação e do tempo de relaxação do material. Esse tempo de relaxação é definido como o tempo necessário para que as tensões internas sejam dissipadas a partir do momento em que a força motriz de um escoamento é retirada. Ou seja: esse tempo de relaxação pode ser interpretado como uma medida que busca quantificar o tempo que é necessário para o fluido se “adaptar” à sua nova condição. Assim, se um fluido viscoelástico for submetido a uma deformação tal que essa operação seja realizada em um tempo muito menor que seu tempo de

relaxação, as tensões internas continuaram presentes no fluido depois do fim da operação, e dessa forma o seu comportamento irá apresentar características próximas às características de um sólido perfeitamente elástico, ou seja: o fluido apresentará uma deformação elástica o tenderá a voltar para o seu formato anterior. De maneira semelhante, um fluido viscoelástico submetido a uma deformação durante um período muito maior que o seu tempo de relaxação apresentará características de próximas de um fluido ideal. A recuperação elástica parcial que pode ser apresentada por essa classe de fluidos é definida como memória do fluido. Como em um fluido ideal essa relaxação das tensões internas é instantânea, o tempo de relaxação de um fluido ideal é igual à zero.

2.2.3 Fluidos puramente viscosos

Os fluidos independes do tempo são caracterizados por apresentarem uma equação constitutiva da forma:

$$\dot{\gamma} = f(\tau) \tag{2.2}$$

Ou seja: a taxa de deformação é função apenas da tensão cisalhante. Assim, também é possível escrever a sua forma inversa, ou seja:

$$\tau = f^{-1}(\dot{\gamma}) \tag{2.3}$$

Essa categoria de fluidos não newtonianos pode ser subdividida em:

- Fluidos pseudoplásticos: quando a viscosidade diminui com o aumento da taxa de deformação;
- Fluidos dilatantes: quando a viscosidade aumenta com o aumento da taxa de deformação;
- Fluidos viscoplásticos: são os fluidos caracterizados por apresentarem um patamar de alta viscosidade aparente até determinada tensão limite de escoamento (τ_0). Uma vez que essa tensão é atingida, a viscosidade aparente diminui e o fluido passa a escoar com maior facilidade, o que pode fazer com que um pequeno aumento na tensão cisalhante provoque uma variação de algumas ordens de grandeza na taxa de deformação.

Na Fig. 2.4 esses alguns desses comportamentos são comparados graficamente.

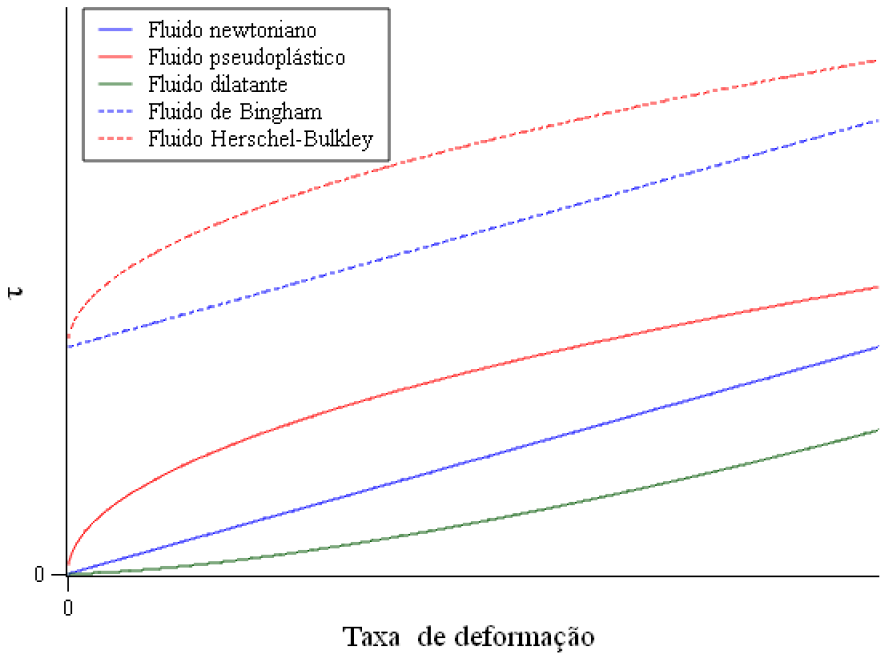


Figura 2.4 – Representação gráfica dos diferentes tipos de fluidos puramente viscosos. Fonte: Adaptado de Chhabra e Richardson (1999).

Fluidos pseudoplásticos e dilatantes

O fluidos pseudoplásticos são os mais comuns entre os comportamentos não newtonianos (BOGER, 1977). Sua principal característica é que a viscosidade aparente diminui com o aumento da taxa de deformação. A Fig. 2.5 apresenta um dos comportamentos típicos dessa classe de fluidos: um patamar de viscosidade aparente para baixas taxas de deformação (η_0), seguido por uma região onde a viscosidade aparente diminui exponencialmente em função da taxa de deformação, e por fim outro patamar onde a viscosidade aparente é relativamente constante para altas taxas de deformação (η_∞).

Da mesma forma que foi discutido para fluidos newtonianos, a inclinação da curva tensão cisalhante em função da taxa de deformação é a viscosidade aparente de um fluido não newtoniano. Nem todos os fluidos pseudoplásticos apresentam o comportamento apresentado na Fig. 2.5, e é possível que alguns apresentem apenas um desses patamares, por exemplo. Outra dificuldade é que para a elaboração dessas curvas é necessário estudar várias ordens de grandeza da taxa de deformação, e a tensão cisalhante também pode variar algumas ordens de grandeza. Na Fig. 2.5 a taxa de deformação varia de $10^{-2}s^{-1}$ até $10^{5}s^{-1}$, o que gera a necessidade de se trabalhar com escalas logarítmicas, além de impor uma série de dificuldades na elaboração dessas curvas. Outra conclusão que é possível extrair da Fig. 2.5 é que o

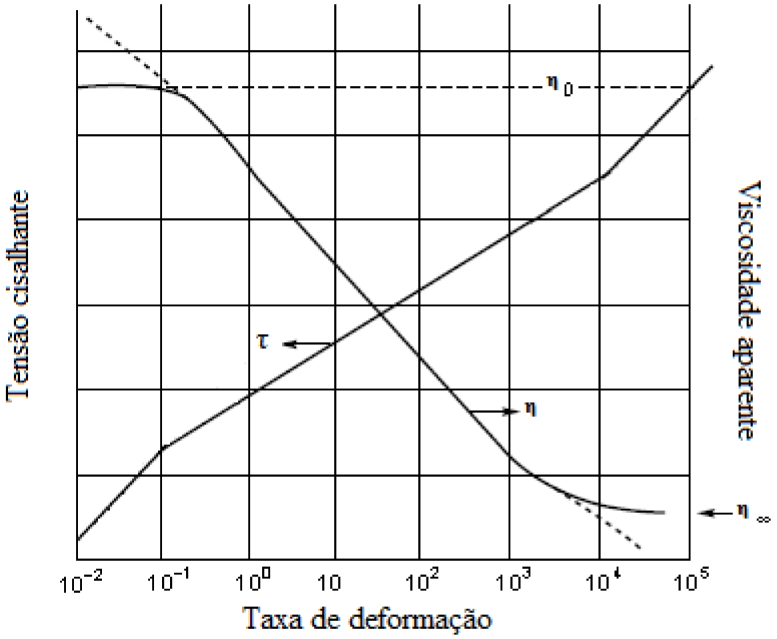


Figura 2.5 – Tensão cisalhante e viscosidade aparente em função da taxa de deformação de um fluido pseudoplástico com um patamar de viscosidade aparente para baixas taxas de deformação (η_0) e outro para altas taxas de deformação(η_∞). Fonte: Adaptado de Chhabra e Richardson (1999).

comportamento não newtoniano se torna mais relevante quando são analisadas várias ordens de grandeza da taxa de deformação. Para intervalos mais curtos de taxa de deformação, a hipótese de viscosidade constante pode conseguir representar bem o escoamento estudado.

Já a característica que define os fluidos dilatantes é que eles apresentam um comportamento oposto ao dos pseudoplásticos: a viscosidade aparente aumenta com o aumento da taxa de deformação. Esse comportamento foi primeiramente observado em suspensões concentradas, e um exemplo comum desse comportamento é a solução de amido de milho em água. Devido às propriedades desses fluidos, Junior et al. (2006) analisa o uso de fluidos dilatantes no desenvolvimento de uma armadura flexível. Neste trabalho, as armaduras contendo fluidos dilatantes apresentaram melhoras significativas quando comparadas a armaduras de tecido puro com densidade superficial equivalente. Para a comparação, foram realizados testes de resistência ao corte e à perfuração à baixas e altas velocidades.

Fluidos viscoplásticos

O comportamento viscoplástico é caracterizado por um patamar de viscosidade aparente alta até que a sua tensão limite de escoamento seja atingida, e então a viscosidade apa-

rente diminui de forma significativa, o que permite que o fluido escoe com mais facilidade. Esse comportamento induz que escoamentos dessa classe possam apresentar duas regiões distintas: uma região aparentemente não escoada, com alta viscosidade aparente e tensão cisalhante mais baixa que a tensão limite de escoamento, e uma região escoada, com viscosidade aparente significativamente menor e com tensão cisalhante mais alta que a tensão limite de escoamento. Como esse comportamento é independente do tempo ou do histórico cinemático, a viscosidade aparente varia instantaneamente, pois depende apenas da tensão cisalhante.

Quando esses fluidos foram descobertos, os reômetros não conseguiam mensurar a taxa de deformação na região de alta viscosidade, o que era interpretado como se o fluido viscoplástico apresentasse viscosidade aparente infinita nas regiões onde a tensão de cisalhamento era menor que a tensão limite de escoamento. Outra diferença em relação aos outros comportamentos é que a curva de tensão cisalhante pela taxa de deformação ficava deslocada, sem passar pela origem, já que para se atingir valores de taxa de deformação mensuráveis, era necessário aplicar uma tensão cisalhante maior que a tensão limite de escoamento do fluido. Hoje, os reômetros modernos são capazes de medir taxas de deformação mesmo nas zonas aparentemente não escoadas, ou seja, a viscosidade aparente tem um valor finito mesmo nas regiões com tensão cisalhante abaixo da tensão limite de escoamento (de Souza Mendes e Dutra (2004)).

Na Fig. 2.4 esse comportamento foi subdivido em dois: o plástico de Bingham e os fluidos *Herschel-Bulkley*. O plástico de Bingham que é um caso particular de viscoplástico: uma vez vencida a tensão inicial, o fluido passa a ter um comportamento semelhante a um fluido newtoniano. Já a curva *Herschel-Bulkley* representa um comportamento que apresenta uma tensão limite de escoamento, mas que depois que essa tensão é superada, o fluido se comporta como um fluido pseudoplástico ou como um fluido dilatante. A Fig. 2.6 mostra esquematicamente todas as principais classes de fluidos já apresentados neste trabalho.

Fluido Newtoniano Generalizado

A tensão cisalhante em fluido newtoniano incompressível pode ser escrita da seguinte maneira:

$$\boldsymbol{\tau} = 2\mu\mathbf{D}(\mathbf{u}) \tag{2.4}$$

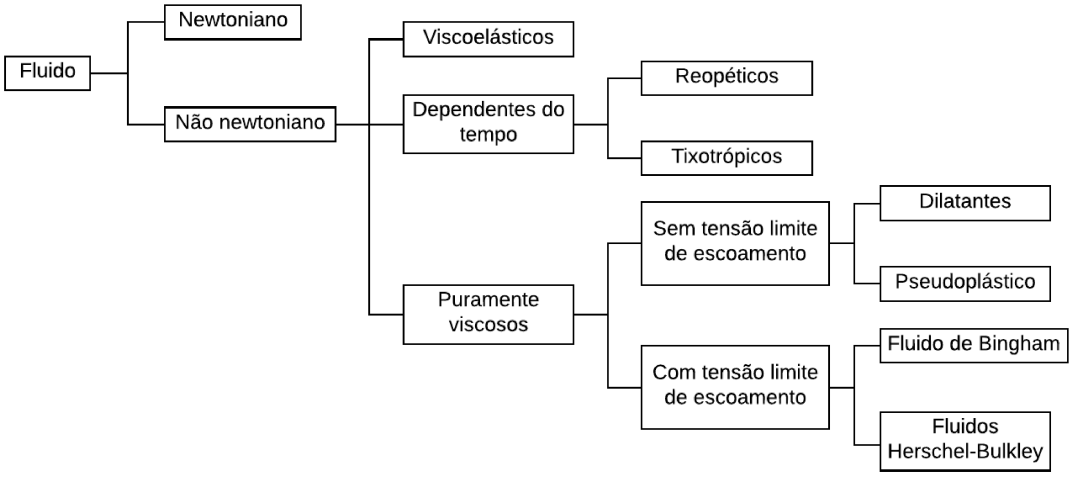


Figura 2.6 – Representação gráfica da classificação dos fluidos newtonianos e não newtonianos. Fonte: Adaptado de Bicalho (2015).

onde

$$\mathbf{D}(\mathbf{u}) = \frac{1}{2}(\nabla\mathbf{u} + \nabla\mathbf{u}^T) \tag{2.5}$$

onde **u** é o vetor velocidade, **D** é o tensor taxa de deformação e τ é o tensor de tensão extra total.

Como já discutido, μ é função do fluido, da temperatura e da pressão. Com o intuito de se criar um modelo semelhante ao modelo newtoniano, mas que consiga representar a variação da viscosidade em função da taxa de deformação, foi definido o Modelo Newtoniano Generalizado (em inglês *Generalized Newtonian Fluid*, sigla GNF), que é caracterizado pela seguinte equação constitutiva:

$$\boldsymbol{\tau} = 2\eta\mathbf{D}(\mathbf{u}) \tag{2.6}$$

onde η é a viscosidade aparente, que é função dos invariantes do tensor taxa de deformação **D**(**u**). Os invariantes são:

$$\begin{aligned} I_D &= tr(D) \\ II_D &= tr(D^2) \\ III_D &= det(D) \end{aligned} \tag{2.7}$$

Para fluidos incompressíveis:

$$I_D = 2(\nabla \cdot \mathbf{u}) = 0 \tag{2.8}$$

Em escoamentos puramente cisalhantes,

$$III_D = 0 \tag{2.9}$$

já que o elemento de fluido não tem o seu volume alterado. Assim, η é função apenas de II_D, e a Eq.(2.6) pode ser escrita da seguinte forma:

$$\boldsymbol{\tau} = 2\eta(\dot{\gamma})\mathbf{D}(\mathbf{u}) \tag{2.10}$$

onde $\dot{\gamma}$ é a magnitude do tensor taxa de deformação, e pode ser definido como:

$$\dot{\gamma} = \sqrt{2II_D} = \sqrt{2tr\mathbf{D}(\mathbf{u})^2} \tag{2.11}$$

2.3 Modelos de viscosidade não newtoniana independentes do tempo

Para modelar os comportamentos não newtonianos independentes do tempo, diversos modelos matemáticos foram propostos através de relações empíricas, ajuste de curvas e estudos teóricos. Dentre esses modelos, se destacam:

- Modelo *Power-law* ou Ostwald de Waele;
- Modelo de Bingham;
- Modelo Herschel-Bulkley;
- Modelo SMD.

2.3.1 Modelo power-law

O modelo *power-law* é caracterizado pela equação constitutiva

$$\tau = K\dot{\gamma}^n \tag{2.12}$$

ou na forma:

$$\eta = \frac{\tau}{\dot{\gamma}} = K\dot{\gamma}^{n-1} \tag{2.13}$$

onde *K* é denominado índice de consistência do fluido e *n* é o índice de comportamento. Esses dois parâmetros são definidos empiricamente, de forma que:

- *Se n<1*, o fluido apresenta um comportamento pseudoplástico;
- *Se n=1*, o fluido apresenta um comportamento newtoniano;
- *Se n>1*, o fluido apresenta um comportamento dilatante.

Esse é o modelo mais utilizado em publicações de aplicação em engenharia, ainda que apresente algumas limitações por causa da sua simplicidade. Como pode ser observado na Fig. 2.7, o modelo *power-law* é válido para uma faixa de valores de taxas de deformação, mas ele não prevê comportamentos como a existência de um patamar η_0 ou η_∞, por exemplo.

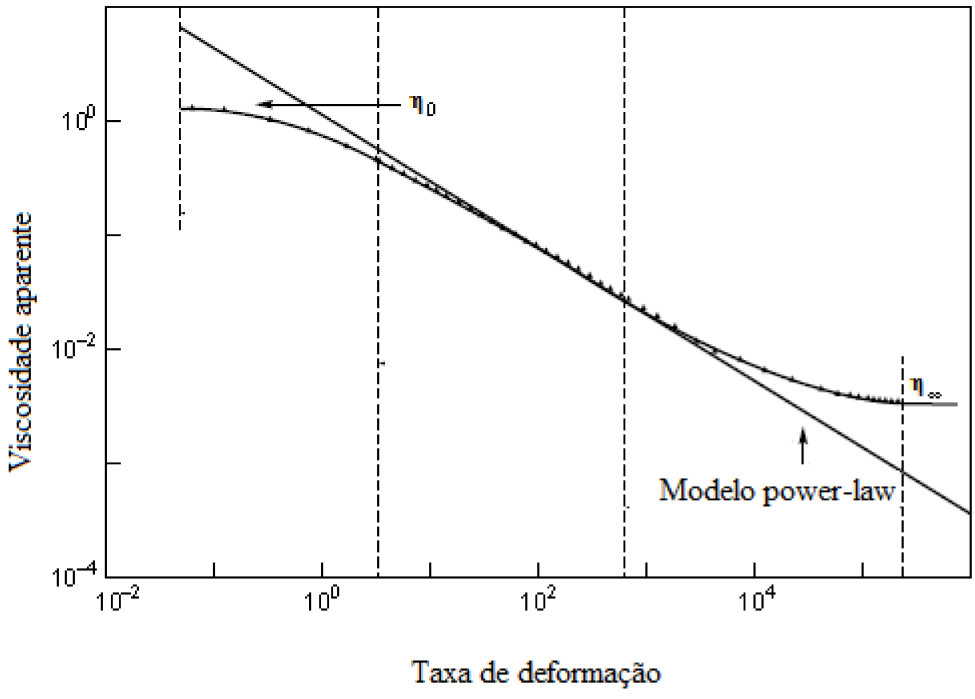


Figura 2.7 – Comparação entre a viscosidade aparente obtida com o modelo *Power-law* e a viscosidade de um fluido pseudoplásticos com patamar η_0 e η_∞. Fonte: Adaptado de Chhabra e Richardson (1999).

Turan et al. (2011) apresenta um estudo sobre a convecção natural em uma cavidade quadrada bidimensional preenchida por um fluido *power-law*, onde as paredes laterais apre-

sentam temperaturas constantes e diferentes entre si. O estudo analisa os efeitos do índice de comportamento, do número de Rayleigh e do número de Prandtl sobre o número de Nusselt médio, o perfil de velocidade vertical e o perfil de temperatura no eixo horizontal que atravessa o recipiente na metade da sua altura. O número de Nusselt médio apresenta uma dependência positiva em relação ao número de Rayleigh e uma dependência negativa em relação ao índice de comportamento. Dessa forma, o estudo afirma que fluidos pseudoplásticos (ou seja, $n < 1$) apresentam um número de Nusselt médio maior que os fluidos newtonianos ou dilatantes, para um mesmo número de Rayleigh. Segundo os autores, um *n* menor favorece o transporte convectivo. Para um valor de *n* suficientemente grande, o número de Nusselt médio tende à unidade, o que significa que a transferência de calor se dá principalmente pela condução. Com base nos resultados obtidos nas simulações, os autores propõem expressões que correlacionam o número de Nusselt médio e os parâmetros analisados.

Kumar, Dhiman e Baranyi (2015) estuda a convecção forçada em fluidos *power-law* em um canal planar com um cilindro semicircular imerso e com temperatura constante. Para realizar as simulações, os autores utilizaram o *software* Fluent®. Em um primeiro momento, o trabalho busca identificar os efeitos do índice de comportamento e do número de Reynolds sobre o coeficiente de arrasto, o comprimento da zona de recirculação e sobre o número de Nusselt médio. Os autores concluíram que o coeficiente de arrasto apresenta uma dependência positiva em relação ao índice de comportamento, e que o comprimento da zona de recirculação tende a diminuir com o aumento de *n*. Já o número de Nusselt médio aumenta com o aumento do número de Reynolds, mas diminui com o aumento de *n*. Em um segundo momento, os autores fixam o número de Reynolds e estudam também a influência da razão entre o diâmetro do cilindro e a altura do canal. Nessa segunda análise, o coeficiente de arrasto e o número de Nusselt médio apresentaram uma dependência positiva com o novo parâmetro estudado.

Bijjam, Dhiman e Gautam (2015) analisa os resultados obtidos em simulações numéricas de um escoamento bidimensional de fluidos *power-law* dilatantes ($n > 1$) em torno de um cilindro aquecido confinado entre duas superfícies adiabáticas, usando o *software* Fluent®. Foram realizadas simulações para um intervalo de valores de número de Reynolds, de número de Prandtl, de índice de comportamento e para diferentes razões de bloqueio e diferentes alturas relativas do cilindro, gerando assim geometrias assimétricas. Para essa configuração, o coefi-

ciente de arrasto apresentou uma correlação positiva com o índice de comportamento e com a razão de bloqueio, mas o aumento do número de Reynolds tende a diminuir o coeficiente de arrasto. Além disso, quanto mais o cilindro se afasta do plano central entre as duas superfícies, menor é o coeficiente de arrasto. Já o número de Nusselt médio aumentou com o aumento do número de Reynolds e da razão de bloqueio, mas diminuiu com o aumento do índice de comportamento.

2.3.2 Modelo de Bingham

O modelo de Bingham tenta aproximar o comportamento de fluidos viscoplásticos, separando a equação constitutiva em duas condições: uma para até que a tensão limite de escoamento τ_0 seja atingida e a outra para quando a tensão de cisalhamento seja maior que essa tensão limite de escoamento. A equação constitutiva do modelo de Bingham apresenta a seguinte forma:

$$\begin{aligned} \tau &= \tau_0 + \mu_B \dot{\gamma} \qquad se \quad \tau > \tau_0 \\ \dot{\gamma} &= 0 \qquad se \quad \tau < \tau_0 \end{aligned} \tag{2.14}$$

Desse modo, a viscosidade aparente do modelo de Bingham é igual a:

$$\begin{aligned} \eta &= \mu_B \qquad se \quad \tau > \tau_0 \\ \eta &= \infty \qquad se \quad \tau < \tau_0 \end{aligned} \tag{2.15}$$

onde τ_0 é a tensão limite de escoamento, μ_B é a viscosidade plástica, um parâmetro especifico do modelo de Bingham, como representado pelo subíndice B.

Uma vez ultrapassada a tensão limite de escoamento, a curva tensão cisalhante por taxa de deformação se torna uma reta com coeficiente linear constante e igual a μ_B. Como pode ser observado na Eq.(2.15), o modelo de Bingham apresenta uma descontinuidade no valor da viscosidade aparente. Para resolver esse problema, Papanastasiou (1987) propôs uma versão regularizada, na seguinte forma:

$$\tau = (1 - exp(-m\dot{\gamma}))\tau_0 + \mu_B \dot{\gamma} \tag{2.16}$$

onde m é o parâmetro de regularização.

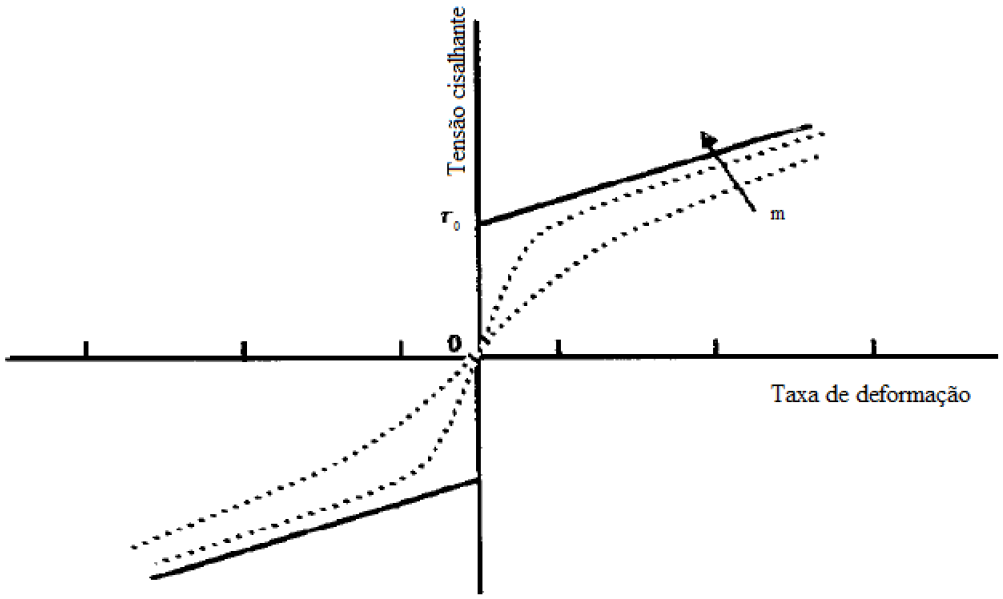


Figura 2.8 – Tensão cisalhante em função da taxa de deformação segundo a regularização de Papanastasiou para diferentes valores de m. Fonte: Papanastasiou (1987).

A versão regularizada de Papanastasiou apresenta uma viscosidade aparente contínua, mas continua prevendo viscosidade aparente infinita quando $\dot{\gamma}$ tende a zero. Como já discutido, os reômetros modernos mostram que os fluidos viscoplásticos apresentam uma viscosidade finita mesmo quando a tensão limite de escoamento ainda não foi superada. Outra característica da versão regularizada por Papanastasiou é que ela acrescenta um novo parâmetro, *m*, que pode ser representado como a velocidade de transição entre os dois regimes característicos dos fluidos viscoplásticos: quanto maior o valor de *m*, mais acentuada é a transição, e assim mais a versão regularizada se aproxima do modelo de Bingham, um valor de *m* infinito recupera o modelo de Bingham. A Fig. 2.8 mostra esse comportamento do parâmetro de regularização: quanto maior o valor de *m*, mais a regularização se aproxima do modelo de Bingham, representado na figura pela linha cheia.

Roustaei e Frigaard (2013) realizaram simulações com fluidos de Bingham em canais com paredes onduladas, em função do número de Bingham, da amplitude da onda e da razão entre o diâmetro do canal e o comprimento da onda. Foi imposta a condição de escoamento de Stokes, ou seja o fluido é considerado incompressível, as simulações ocorrem em regime permanente e o número de Reynolds é muito menor que a unidade. De acordo com a combinação desses três fatores, o estudo busca definir uma amplitude crítica, a partir da qual começam a ser formadas incrustações na parte mais exterior da ondulação. Assim, são propostas ex-

pressões em função desses três fatores para se definir as condições necessárias e suficientes para o surgimento das incrustações. Os autores destacam que as relações propostas são praticamente independentes do número de Bingham quando este é suficientemente grande. Os autores também analisam outros formatos de onda, como ondas quadradas e ondas triangulares, e analisam os formatos das zonas não escoadas, comparando-as com as estruturas equivalentes em escoamentos newtonianos, por exemplo.

Nirmalkar, Bose e Chhabra (2014) estuda a transferência de calor por convecção natural laminar em um cilindro horizontal aquecido, imerso em escoamento vertical de um fluido de Bingham. A versão regularizada de Papanastasiou foi empregada, com um parâmetro *m* suficientemente grande para se garantir um comportamento próximo ao modelo de Bingham original. As equações de balanço de quantidade de movimento e de energia foram resolvidas numericamente usando *software* Comsol Multiphysics®, usando o método dos elementos finitos. Os parâmetros analisados no trabalho foram: o número de Rayleigh, o número de Prandtl, e o número de Bingham. Os campos de velocidade e de temperatura obtidos através de várias combinações desses parâmetros são comparados, além das regiões escoadas e aparentemente não escoadas de cada um. O número de Nusselt local e médio das simulações são apresentados, e os autores propõem uma expressão para se determinar o número de Nusselt médio em função do número de Rayleigh e do número de Prandtl modificados, incorporando os efeitos viscoplásticos nas suas definições. O trabalho também apresenta uma comparação entre as regiões escoadas e aparentemente não escoadas obtidas adotando a versão regularizada de Papanastasiou e o modelo de bi-viscosidade com a viscosidade inicial suficientemente alta.

O trabalho de Nalluri, Patel e Chhabra (2015) busca estudar a transferência de calor por convecção mista em um escoamento laminar bidimensional em torno de um hemisfério imerso, de tal forma que a face arredondada do hemisfério é voltada contra o sentido do escoamento. O hemisfério é mantido a uma temperatura constante e o fluido do escoamento é um fluido de Bingham, modelado pela versão regularizada de Papanastasiou. A influência do número de Reynolds, do número de Prandtl, do número de Richardson e do número de Bingham foram analisadas. As linhas de corrente e as isotermas de escoamentos com várias combinações desses 4 parâmetros são apresentadas. A geometria do problema pode levar a zonas de recirculação na face plana do hemisfério, e o trabalho compila dados sobre como o comprimento

dessa recirculação varia em função dos parâmetros estudados. Também é discutido a influência desses parâmetros no formato das zonas não escoadas presentes nas simulações e no número de Nusselt local e médio. Os autores propõem uma correlação do número de Nusselt médio em função dos quatro parâmetros estudados, e destacam a dependência positiva do número de Nusselt médio com o número de Reynolds e o número de Prandlt, ainda que essas relações se enfraqueçam com o aumento do número de Bingham. O número de Nusselt médio também varia positivamente com o número de Richardson, ainda que sua influência seja menor. Para melhor analisar os efeitos do número de Richardson, os autores propõem um número de Nusselt normalizado, que é a razão entre o número de Nusselt médio sobre o número de Nusselt médio obtido em uma simulação com os mesmos parâmetros mas com o número de Richardson igual a zero.

2.3.3 Modelo de Herschel-Bulkley

O modelo de Herschel-Bulkley também busca modelar o comportamento dos fluidos viscoplásticos, mas propõe um comportamento semelhante ao modelo *power-law* quando a tensão cisalhante é maior que a tensão limite de escoamento. Sua equação constitutiva é definida como:

$$\begin{aligned} \tau &= \tau_0 + K\dot{\gamma}^n \qquad se \quad \tau > \tau_0 \\ \dot{\gamma} &= 0 \qquad se \quad \tau < \tau_0 \end{aligned} \tag{2.17}$$

Quando *n=1*, o modelo de Herschel-Bulkley é igual ao de Bingham. Como esse modelo é composto por 3 coeficientes diferentes, é possível modelar comportamentos mais complexos: além do efeito viscoplástico, o modelo de Herschel-Bulkley também permite modelar o comportamento pseudoplástico ou dilatante quando a tensão de cisalhamento for maior que a tensão limite de escoamento. Para tanto, o valor de *n* segue a mesma lógica do modelo *power-law*.

A versão regularizada proposta por Papanastasiou para o modelo de Bingham pode ser modificada a fim de conseguir regularizar o modelo de Herschel-Bulkley também, na seguinte forma:

$$\tau = (1 - exp(-m\dot{\gamma}))\tau_0 + K\dot{\gamma}^n \tag{2.18}$$

Devido à semelhança entre o modelo proposto por Papanastasiou e o modelo modificado de Papanastasiou, este modelo também apresenta as mesmas dificuldades com a magnitude

da viscosidade quando $\dot{\gamma}$ tende a zero e com a definição do valor de m. Além disso, esse modelo não apresenta um patamar η_∞, o que significa que se *n<1*, a viscosidade aparente tende a zero quando $\dot{\gamma}$ tende ao infinito. Da mesma forma, se *n>1* a viscosidade aparente aumenta indefinidamente com o aumento da taxa de deformação. Esses dois resultados não tem significado físico e não são coerentes com os valores obtidos em experimentos (SARAMITO, 2016).

Entre os escoamentos que podem ser modelados pela equação constitutiva de Herschel-Bulkley, destacam-se os escoamentos presentes na perfuração de um poço de petróleo. Nessa aplicação, um fluido viscoplástico escoa dentro de um canal anular transportando os resíduos do processo de perfuração do poço. Como existe um grande interesse industrial nessa aplicação, é possível encontrar diversos trabalhos na literatura sobre ela. Os trabalhos de Nouar, Desaubry e Zenaidi (1998), Kelessidis et al. (2006) e Bicalho (2015) são alguns exemplos desses trabalhos.

Nouar, Desaubry e Zenaidi (1998) analisa numericamente e experimentalmente a transferência de calor em um fluido de Herschel-Bulkley escoando dentro um canal anular. Neste trabalho, a temperatura do cilindro externo é mantida constante, e o cilindro interno apresenta uma rotação. Em um primeiro momento, o estudo considera as propriedades do fluido como constantes, e analisa os efeitos da rotação do cilindro interno sobre os perfis de velocidade e de temperatura. A rotação do cilindro interno aumenta a taxa de cisalhamento do fluido próximo a ele, o que facilita a transição de uma zona aparentemente não escoada para uma zona escoada. Dessa forma, o estudo busca definir um número de Rossby crítico, que indica que o escoamento não apresenta mais zonas não escoadas. O número de Rossby é definido como a razão entre as forças de inércia pela força de Coriolis. O número de Rossby crítico é dado em função do índice de comportamento do fluido, da razão entre os raios e do número de Herschel-Bulkley do fluxo axial. Em seguida, os autores propõem outra modelagem da viscosidade, onde o índice de consistência *K* pode variar com a temperatura. Os perfis de velocidade e de temperatura e as regiões escoadas e aparentemente não escoadas são comparadas para cada uma das duas modelagens da viscosidade. Os autores chegam a conclusão que o índice de consistência *K* aumenta sua influência sobre a transferência de calor quando o problema passa a considerar que o cilindro interno rotaciona.

No trabalho de Kelessidis et al. (2006) são discutidas as técnicas para a determinação

dos três parâmetros do modelo de Herschel-Bulkley (*n*, *K* e τ_0) a partir de dados experimentais. Os autores afirmam que a aplicação direta de técnicas de regressão não-linear para determinação dos parâmetros podem levar a valores de τ_0 negativos, resultado que não possui significado físico. Para solucionar esse problema, os autores propõem o uso do método da seção áurea para determinação de τ_0, e os demais parâmetros continuam sendo obtidos pela regressão não-linear. Os autores buscaram na literatura diversas bases de dados experimentais e aplicaram sobre elas três técnicas para a obtenção dos parâmetros do modelo, evidenciando as diferenças entre os valores obtidos. Os métodos comparados são: a regressão não-linear sem restrições, a regressão não-linear com a restrição $\tau_0 > 0$ e o método proposto pelos autores. Em seguida, os autores realizaram simulações com os parâmetros obtidos através de diferentes técnicas, para evidenciar as diferenças geradas pela escolha de um determinado método para obtenção dos parâmetros do modelo. O estudo de caso proposto para se analisar as diferenças entre os métodos foi o escoamento dentro de um canal anular concêntrico. Os autores concluíram que a escolha de método adequado tem influência significativa sobre os resultados, em especial nos cálculos sobre perda de carga, nos perfis de viscosidade aparente e em outros parâmetros operacionais.

Bicalho (2015) analisa diversas variações do escoamento em um duto anular. São consideradas configurações com dutos anulares concêntricos e excêntricos, com e sem rotação do tubo interno, e com a presença ou não de uma obstrução parcial do canal anular. Além desses fatores, também foi variada a concentração polimérica, alterando assim as propriedades do fluido. Foram realizadas simulações numéricas usando o *software* Fluent® com diferentes parâmetros operacionais e reológicos, para geometrias com diferentes excentricidades e alturas de obstrução. Os resultados para a queda de pressão dessas simulações foram comparados com os resultados experimentais. Também foram injetadas partículas nos experimentos, a fim de poder observar suas trajetórias e comparar esse resultado com as linhas de corrente obtidas nas simulações. Os resultados numéricos e experimentais apresentaram boa correlação. A rotação do eixo interno não apresentou influência na queda de pressão, enquanto o aumento da concentração polimérica e da vazão aumentaram a queda de pressão. Além disso, o aumento da excentricidade diminuiu a queda de pressão. Também foi observada uma forte interação entre os efeitos das variáveis estudadas.

2.3.4 Modelo SMD

Como já discutido, nos modelos de Bingham e de Herschel-Bulkley a viscosidade aparente apresenta um valor infinito antes da tensão limite de escoamento, resultados que não são coerentes com os valores obtidos pelos reômetros modernos. Assim, o modelo de bi-viscosidade foi proposto por O'Donovan e Tanner (1984), na seguinte forma:

$$\begin{aligned} \eta(\dot{\gamma}) &= \eta_0 & for\ \ \dot{\gamma} < \frac{\tau_0}{\eta_0} \\ \eta(\dot{\gamma}) &= \frac{\tau_0}{\dot{\gamma}} + \eta_\infty \left(1 - \frac{\tau_0}{\eta_0 \dot{\gamma}}\right) & for\ \ \dot{\gamma} \geq \frac{\tau_0}{\eta_0} \end{aligned} \tag{2.19}$$

onde η_0 é a viscosidade aparente nas regiões onde a tensão cisalhante é menor que a tensão limite de escoamento e η_∞ é a viscosidade para altas taxas de deformação.

Porém, a viscosidade nesse modelo apresenta uma descontinuidade em sua derivada quando $\tau = \tau_0$. Buscando resolver essa questão, de Souza Mendes e Dutra (2004) propuseram uma regularização do modelo de bi-viscosidade, que respeita a condição de viscosidade finita até a tensão limite de cisalhamento e que não apresenta essa descontinuidade. A tensão cisalhante segundo o modelo SMD é dada por:

$$\eta = \left[1 - exp\left(-\frac{\eta_0}{\tau_0}\dot{\gamma}\right)\right]\left(\frac{\tau_0}{\dot{\gamma}} + K\dot{\gamma}^{n-1}\right) \tag{2.20}$$

O modelo SMD apresenta 4 parâmetros (*K, n,* τ_0, η_0), o mesmo número da regularização modificada de Papanastasiou, porém os parâmetros do modelo SMD são todos parâmetros reológicos, ao contrário do parâmetro *m* da regularização modificada de Papanastasiou.

Para se modelar fluidos pseudoplásticos ($n < 1$) alguns trabalhos acrescentam um termo para evitar que a viscosidade aparente tenda a zero quando a taxa de deformação tende ao infinito, resultado que não tem significado físico. A expressão apresentada por de Souza Mendes (2007a) é a seguinte:

$$\eta = \left[1 - exp\left(-\frac{\eta_0}{\tau_0}\dot{\gamma}\right)\right]\left(\frac{\tau_0}{\dot{\gamma}} + K\dot{\gamma}^{n-1}\right) + \eta_\infty \left[1 - exp\left(-\frac{\eta_\infty}{K\dot{\gamma}^{n-1}}\right)\right] \tag{2.21}$$

Já a expressão utilizada por Santos et al. (2013) é igual a:

$$\eta = \left[1 - exp\left(-\frac{\eta_0}{\tau_0}\dot{\gamma}\right)\right]\left(\frac{\tau_0}{\dot{\gamma}} + K\dot{\gamma}^{n-1}\right) + \eta_\infty \tag{2.22}$$

As soluções propostas pela Eq.(2.21) e Eq.(2.22) são especificas para fluidos pseudo-plásticos, e não resolvem a questão dos fluidos dilatantes ($n > 1$), onde a viscosidade tende ao infinito quando a taxa de deformação tende ao infinito, resultado que também não apresenta significado físico. A Fig. 2.9 mostra diferentes curvas de viscosidade aparente adimensional em função da taxa de deformação adimensional, obtidas adotando a Eq.(2.22) para diferentes valores de n.

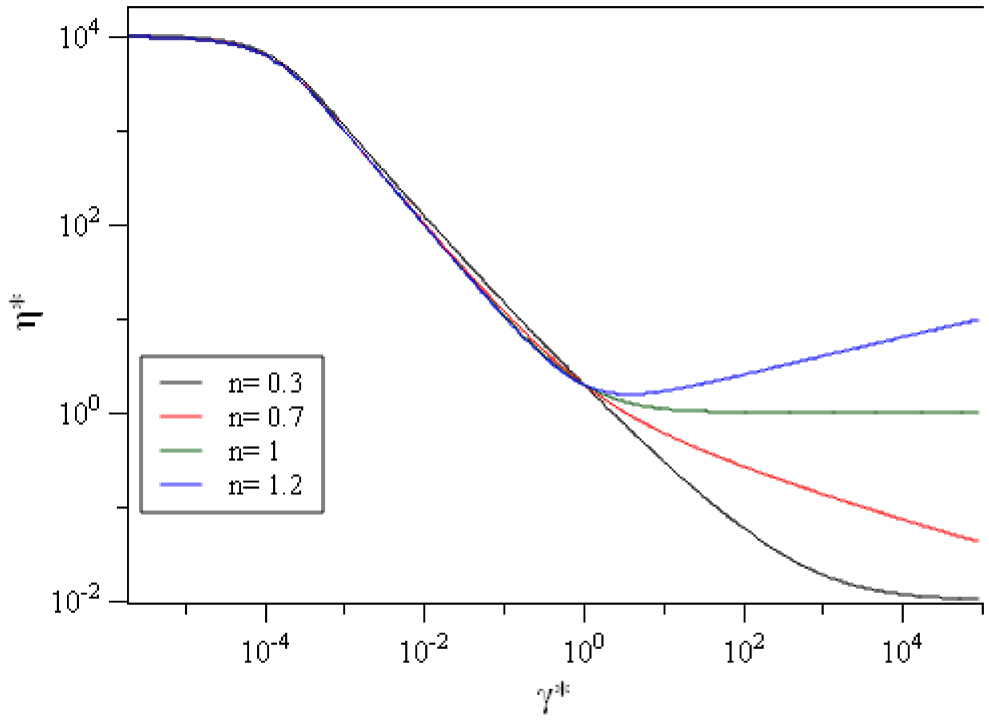


Figura 2.9 – η^* em função de $\dot{\gamma}^*$, usando o conjunto padrão de adimensionais deste trabalho variando n.

Além disso, segundo de Souza Mendes (2007a), é possível identificar três taxas de deformação de transição no comportamento viscoplástico com dois patamares de viscosidade aparente: $\dot{\gamma}_0$, que separa o patamar de alta viscosidade da região de transição, $\dot{\gamma}_1$ que separa a região de transição da região *power-law*, e por fim $\dot{\gamma}_2$ que separa a região *power-law* do platô de viscosidade aparente para altas taxas de deformação. As expressões para $\dot{\gamma}_0$, $\dot{\gamma}_1$, $\dot{\gamma}_2$, a taxa de deformação adimensional ($\dot{\gamma}^*$) e a viscosidade aparente adimensional (η^*) são:

$$\dot{\gamma}_0 = \frac{\tau_0}{\eta_0} \qquad \dot{\gamma}_1 = \left(\frac{\tau_0}{K}\right)^{\frac{1}{n}} \qquad \dot{\gamma}_2 = \left(\frac{K}{\eta_\infty}\right)^{1/(1-n)} \tag{2.23}$$

$$\dot{\gamma}^* = \frac{\dot{\gamma}}{\dot{\gamma}_1} \qquad \eta^* = \frac{\eta\dot{\gamma}_1}{\tau_0} \tag{2.24}$$

O trabalho de de Souza Mendes et al. (2007b) apresenta uma comparação entre simulações numéricas e resultados experimentais de escoamentos de fluidos viscoplásticos em um canal axissimétrico com uma expansão seguida de uma contração. Os experimentos foram realizados com soluções com diferentes concentrações de carbopol e microesferas refletoras, e um feixe de laser foi utilizado para assim poder identificar a trajetória das partículas, além das zonas escoadas e aparentemente não escoadas. O número de Reynolds foi mantido sempre abaixo de 0,1 para garantir que os efeitos de inércia são negligenciáveis, o que também, teoricamente, provoca que o escoamento seja simétrico em relação ao plano médio entre a expansão e a contração. Porém, os resultados experimentais não apresentaram essa simetria, e os autores atribuíram esse comportamento ao comportamento elástico das soluções de carbopol nas regiões onde a tensão é menor que τ_0. Os parâmetros analisados nas simulações foram o *jump number*, o índice de comportamento, a tensão adimensional na parede, a razão entre a altura do canal e da cavidade formada entre a expansão e a contração, e a razão entre a altura da cavidade e o seu comprimento. O *jump number* é um grupo adimensional proposto pelos autores que proporciona uma medida relativa da variação da taxa de deformação em torno do ponto onde $\tau = \tau_0$. O modelo SMD foi adotado para modelar a viscosidade aparente. Além das comparações dos escoamentos, também foram apresentados resultados para a eficiência de deslocamento e o *head loss*, que uma grandeza adimensional que busca computar a influência da geometria adotada sobre o escoamento. Para tanto, a queda de pressão na geometria proposta é comparada com a queda de pressão em um canal simples.

Naccache e Barbosa (2007) analisam numericamente um escoamento de fluido viscoplástico em um canal planar com uma expansão seguida de uma contração. As condições do escoamento são definidas para se garantir que os efeitos de inércia sejam negligenciáveis, e a viscosidade aparente foi modelada com a função SMD. O software utilizado para resolver o escoamento foi o Fluent®. As simulações foram realizadas com uma geometria fixa e a tensão cisalhante adimensional na parede do canal foi mantida contante, para poder analisar o efeito

do *jump number* e do índice de comportamento sobre os campos de tensão e velocidade, e o *head loss*. O *head loss* apresentou uma correlação positiva com os parâmetros analisados, e para baixos números de salto, o *head loss* apresentou valores negativos.

Santos (2010) apresenta um estudo numérico de escoamentos bidimensionais em regime permanente de fluidos viscoplásticos em uma cavidade quadrada com velocidade horizontal não nula imposta sobre a parede superior. O trabalho propõe uma metodologia para se definir as zonas escoadas e não escoadas que busca evitar erros nas regiões de transição entre essas zonas, onde as tensões de cisalhamento são próximas à tensão limite de escoamento e uma pequena variação na tensão cisalhante pode provocar uma diferença de algumas ordens de grandeza na taxa de deformação e na viscosidade aparente. As simulações foram realizadas com um *software* de elementos finitos para fluidos não newtonianos desenvolvido no Laboratório de Mecânica dos Fluidos Aplicada e Computacional (LAMAC) da UFRGS. Para se modelar a viscosidade, foi adotado o modelo SMD, e os parâmetros variados nas simulações numéricas realizadas foram: o *jump number*, o índice de comportamento, a vazão adimensional e o número de Reynolds reológico. Para se identificar a influência desses parâmetros no escoamento, os campos de velocidade, de tensão e as zonas escoadas e aparentemente não escoadas foram analisados. O aumento do *jump number*, do índice de comportamento e da vazão adimensional provocaram a diminuição das zonas aparentemente não escoadas.

Alegria (2011) analisa o problema do duto anular adotando a função SMD para modelar a viscosidade aparente. Esse trabalho estuda a influência de parâmetros geométricos, cinemáticos e reológicos do escoamento. Como o processo de perfuração de poços de óleo e gás apresentam várias pertubações externas, a geometria dos canais pode ser alterada. Assim, o autor propõe três configurações de tubos externos elípticos: um canal elíptico simples, canal elíptico com um cilindro circular concêntrico, e um canal elíptico com um cilindro circular excêntrico. O autor apresenta soluções analíticas e soluções numéricas para o problema. As simulações numéricas são realizadas com o *software* PHOENICS®, utilizando o método dos volumes finitos. Para cada geometria investigada foram obtidos parâmetros de interesse na engenharia como: perfil de velocidade, vazão volumétrica, perfil das tensões e expressão para o fator de atrito. Comparando os resultados numéricos e os analíticos, o autor identificou faixas de parâmetros geométricos e cinemáticos onde a concordância entre as duas metodologias são maiores, e que as maiores discrepâncias ocorriam para números de Bingham maiores.

Assim, o autor propõe correlações entre os resultados numéricos e analíticos, para obter uma melhor concordância entre eles.

Ramos, Soares e Thompson (2013) estuda a influência do *jump number* e do índice de comportamento sobre o tamanho e a forma das zonas de transição entre as zonas não escoadas e as zonas escoadas, ou seja: as regiões onde a tensão cisalhante é próxima à tensão limite de escoamento, e por isso uma pequena variação na tensão cisalhante pode provocar uma variação de algumas ordens de grandeza na taxa de cisalhamento. A geometria adotada foi uma contração abrupta 4:1, a viscosidade aparente foi aproximada pelo modelo SMD e o método dos elementos finitos foi empregado para discretizar as equações. No intervalo de valores de índice de comportamento analisado ($0.7 \leq n \leq 1$), este apresentou uma influência tímida nos resultados globais. Já o número de salto apresentou grande influência no tamanho das zonas de transição, ainda que não tenha apresentado influência significativa na correção de Couette, que é um parâmetro utilizado para avaliar a perda de energia devido à contração brusca.

Hermany et al. (2013) apresenta um estudo sobre os efeitos da inércia em escoamentos em um canal axissimétrico com uma expansão seguida de uma contração. O modelo adotado é a função SMD, e o índice de comportamento é sempre menor que um. Os parâmetros analisados são a densidade adimensional, a intensidade do escoamento adimensional e o índice de comportamento. Os resultados são apresentados em função do *head loss* e das zonas escoadas e aparentemente não escoadas. A intensidade do escoamento adimensional e o número de salto apresentam grande influência sobre a zonas cilhadas e aparentemente não escoadas, enquanto o índice de comportamento apresenta um efeito mais tímido. A densidade adimensional é o parâmetro que representa a inércia, e dessa forma quando a densidade adimensional é relevante, o escoamento perde a sua simetria em relação ao plano central.

CAPÍTULO III

MÉTODOS NUMÉRICOS

Neste capítulo as principais características de uma simulação numérica são analisadas, com o objetivo de embasar e justificar as escolhas feitas no decorrer deste trabalho. Os principais componentes de um método numérico são apresentados, assim como suas principais propriedades.

Ainda que os princípios para a obtenção de soluções numéricas sejam conhecidos há mais de um século, foi apenas com o desenvolvimento dos computadores modernos que os métodos numéricos puderam ser aplicados em larga escala para resolver as equações da mecânica dos fluidos, gerando assim a área de estudo conhecida como ”dinâmica dos fluidos computacional”, ou *computational fluid dynamics* (CFD), em inglês. Comparado com os experimentos físicos, a dinâmica dos fluidos computacional oferece uma solução mais rápida e barata, mas a precisão dos resultados depende dos modelos e métodos empregados. Outra vantagem do CFD é a maior facilidade para se obter resultados locais, já que muitas vezes a simples tentativa de se medir um resultado local pode acabar influenciando-o durante o experimento.

A equação de conservação de massa e a equação de conservação de quantidade de movimento linear são equações diferenciais parciais que fazem parte do conjunto de equações que governam os escoamentos. Porém, elas só apresentam solução analítica em casos particulares, e assim a aplicação de métodos numéricos se faz necessária para a obtenção de soluções de casos mais complexos. Para se obter essa solução numérica, um método de discretização é aplicado em um conjunto finito de subdomínios de espaço e/ou tempo, obtendo-se assim um sistema de equações que aproximam as equações diferenciais, e então esse sis-

tema é resolvido. Essa solução aproximada é válida para um conjunto discreto de pontos, e é necessário utilizar um método de interpolação para se obter os valores das propriedades do escoamento nos demais pontos do domínio.

A equação de conservação de uma quantidade escalar ϕ qualquer pode ser escrita da seguinte forma:

$$\underbrace{\frac{\partial \rho\phi}{\partial t}}_{\text{derivada temporal}} + \underbrace{\nabla \cdot (\rho\phi\mathbf{u})}_{\text{termo convectivo}} = \underbrace{\nabla \cdot (\Gamma_\phi \nabla(\phi))}_{\text{termo difusivo}} + \underbrace{q_\phi}_{\text{termo fonte}} \tag{3.1}$$

onde ρ é a densidade do fluido, *t* é o tempo, Γ_ϕ é a difusividade de ϕ e q_ϕ é o termo fonte de ϕ. A partir da Eq.(3.1) e aplicando a condição de fluido incompressível, é possível se obter a equação de conservação de massa, de conservação da quantidade de movimento linear e de conservação de energia, que são, respectivamente:

$$\nabla \cdot (\rho\mathbf{u}) = 0 \tag{3.2}$$

$$\rho\frac{\partial \mathbf{u}}{\partial t} + \rho\nabla \cdot (\mathbf{uu}) = \nabla \cdot (\eta\nabla(\mathbf{u})) - \nabla p \tag{3.3}$$

$$\frac{\partial T}{\partial t} + \nabla \cdot (T\mathbf{u}) = \alpha\nabla^2(T) \tag{3.4}$$

onde *p* é a pressão e α é a difusividade térmica.

3.1 Componentes de um modelo numérico

Os principais componentes de um modelo numérico são (FERZIGER; PERIC, 2002):

3.1.1 Modelo matemático

O modelo matemático envolve o equacionamento do problema, as hipóteses e os modelos adotados. As condições de contorno do sistema também são parte do modelo matemático. Por exemplo, na Eq.(3.2), a hipótese de fluido incompressível foi utilizada para simplificar a equação de conservação de massa. Dessa forma, a qualidade de um modelo matemático é função não somente da dificuldade para a sua solução, mas também de quão bem ele con-

segue representar o fenômeno físico. Como esses dois objetivos podem ser conflituosos, o processo de otimização do modelo matemático pode levar a soluções particulares para cada tipo de escoamento. Além disso, a qualidade dos modelos adotados também influencia diretamente a qualidade do modelo matemático, o que pode levar à necessidade de validação dos modelos com experimentos físicos.

3.1.2 Modelo de discretização

O modelo de discretização é o método para se obter um sistema de equações que aproximem as equações diferenciais em um conjunto discreto de pontos no tempo e no espaço. Os principais métodos de discretização são o método das diferenças finitas, dos volumes finitos e dos elementos finitos, ou *finite difference* (FD), *finite volume* (FV) e *finite element* (FE), em inglês, respectivamente. Além desses métodos, é possível encontrar métodos híbridos e métodos cuja aplicação costuma ficar restrita a determinados casos. Ainda que alguns métodos sejam mais adequados para certos tipos de problema, em uma malha suficientemente fina e com os parâmetros adequados, todos os métodos convergem para o mesmo resultado.

Método das Diferenças finitas

O método das diferenças finitas é o método mais antigo para a solução numérica de equações diferenciais parciais, e também pode ser considerado o método mais fácil de ser aplicado em geometrias simples. O método se baseia nas equações de conservação na forma diferencial, ou seja, na forma da Eq.(3.1).

No método das diferenças finitas, a equação diferencial é aproximada em cada nó da malha, substituindo as derivadas parciais por aproximações definidas em função dos valores dos nós vizinhos. Assim, se obtém uma equação por nó de grade, relacionando os valores de cada nó com os valores dos nós vizinhos.

Para se obter as aproximações das derivadas de primeira e segunda ordem, podem ser usados o método da expansão em série de Taylor ou técnicas de ajuste de curvas. Essas técnicas também podem ser aplicadas como métodos de interpolação, para se obter valores da propriedade estudada em posições diferentes dos nós da malha.

Como desvantagem, a simples aplicação do método das diferenças finitas não garante a conservação das variáveis, e esse método costuma ser aplicado exclusivamente em geome-

trias mais simples.

Método dos Volumes finitos

No método dos volumes finitos, a malha é discretizada em um conjunto finito de volumes de controle, e as equações de conservação são integradas sobre esses volumes. Integrando a Eq.(3.1) sobre cada volume de controle, e utilizando o teorema de Gauss (Eq.(3.5)) nos termos convectivo (Eq.(3.6)) e difusivo (Eq.(3.7)) da Eq.(3.1), é possível se obter a equação de conservação na forma integral para uma gradeza ϕ (Eq.(3.8)) :

$$\int_{\Omega} \nabla \cdot (\mathbf{c}) d\Omega = \int_{S} \mathbf{n} dS \cdot \mathbf{c} \tag{3.5}$$

$$\int_{\Omega} \nabla \cdot (\rho\phi\mathbf{u}) d\Omega = \int_{S} \mathbf{n} dS \cdot \rho\phi\mathbf{u} \tag{3.6}$$

$$\int_{\Omega} \nabla \cdot (\rho\Gamma_{\phi}\nabla(\phi)) d\Omega = \int_{S} \mathbf{n} dS \cdot \rho\Gamma_{\phi}\nabla(\phi) \tag{3.7}$$

$$\frac{\partial}{\partial t}\int_{\Omega} \rho\phi d\Omega + \int_{S} \mathbf{n} dS \cdot \rho\phi\mathbf{u} = \int_{S} \mathbf{n} dS \cdot \rho\Gamma_{\phi}\nabla(\phi) + \int_{\Omega} q_{\phi} d\Omega \tag{3.8}$$

onde ***c*** é um campo vetorial qualquer, Ω é o volume do volume de controle, *S* a sua superfície e ***n*** é o vetor unitário normal à cada superfície, sempre com sentido voltado para fora do volume de controle.

Assim, os valores são calculados para o centroide de cada volume de controle, e os valores das superfícies de cada volume são obtidos por interpolação usando os valores dos centroides vizinhos a aquela superfície. Então, as integrais e derivadas são aproximadas para se obter o sistema de equações a ser resolvido, em função dos valores da propriedade nos centroides dos volumes de controle.

O método dos volumes finitos apresenta duas principais vantagens sobre o método das diferenças finitas: o método dos volumes finitos pode ser aplicado em qualquer tipo de malha, o que permite a aplicação desse método em escoamentos com geometrias mais complexas. Outra vantagem é que ele garante a conservação de suas propriedades, já que a conservação é garantida em cada um dos volumes de controle, e assim, ao se somar todos os volumes de

controle que compõe o domínio geométrico, a conservação passa a ser garantida na malha como um todo (PATANKAR, 1980).

Em compensação, o método de volumes finitos apresenta maiores dificuldades para se desenvolver métodos de aproximação de ordem superior à segunda ordem, já que este método requer três níveis de aproximação: interpolação, derivação e integração.

Método dos elementos finitos

O método dos elementos finitos também utiliza um conjunto de volumes discretos, tal como o método dos volumes finitos. Porém, no método dos elementos finitos, o valor das propriedades são calculados nos nós da malha, e uma função de aproximação é usada para se obter os valores no interior do elemento. Para se obter os coeficientes dessa função de aproximação, uma função-peso é utilizada.

O método dos elementos finitos consegue lidar bem com geometrias complexas, e permite que sua malha seja facilmente refinada, por exemplo. Porém, malhas muito complexas podem gerar matrizes que não são tão adequadas para a solução, o que pode fazer a eficiência do método diminuir.

3.1.3 Sistema de coordenadas

A escolha do sistema de coordenadas mais adequado depende de cada problema analisado, e o modelo matemático e a malha sofrem influência da escolha do sistema de coordenadas.

3.1.4 Malha

A malha é uma representação discreta do domínio geométrico do problema. A malha divide o domínio do escoamento em um conjunto finito de subdomínios, como volumes de controle, superfícies ou pontos, por exemplo. A Fig. 3.1 mostra a diferença entre essas diferentes formas de se discretizar o domínio. Como descrito no item anterior, cada modelo de discretização aplica as equações de conservação sobre a malha de uma maneira diferente. No métodos das diferenças finitas, as equações de conservação são aplicadas sobre os nós da malha, como representado na Fig. 3.1(c). Já no método dos volumes finitos as equações de conservação são aplicadas em cada volume de controle, e assim são calculados os valores

para os centroides dos volumes, como na figura Fig. 3.1(a). Como pode ser observado na Eq.(3.5), o método dos volumes finitos também utiliza os valores na superfície de cada volume, e métodos de interpolação são utilizados para se aproximar uma malha no formato da Fig. 3.1(b). As condições de contorno definidas no método matemático são aplicadas sempre sobre uma superfície, como pode ser observado na Fig. 3.1.

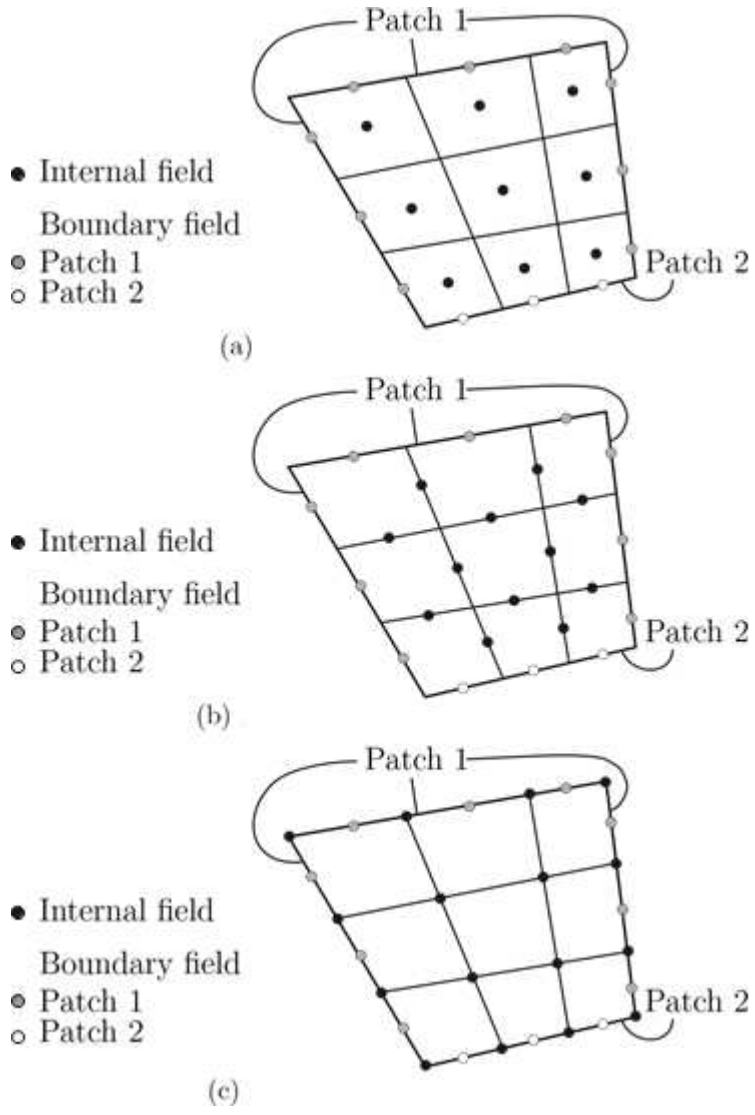


Figura 3.1 – Diferentes representações de malha discretizando um domínio com duas condições de contorno ("*Patch* 1" e "*Patch* 2") em (a) um conjunto de volumes, (b) um conjunto de superfícies, e (c) um conjunto de pontos. Fonte: Adaptado de Greenshields (2015).

Uma malha é considerada ortogonal quando as suas linhas sempre formam 90° quando se cruzam, e uma malha é dita estruturada se suas linhas podem ser divididas em conjuntos de tal forma que as linhas de um mesmo conjunto não se cruzam e duas linhas de conjuntos diferentes se interceptam no máximo uma única vez (FERZIGER; PERIC, 2002). A malha ser

estruturada facilita a identificação dos elementos vizinhos, já que essa propriedade garante, por exemplo, que todo volume de controle interno de uma malha bidimensional vai ter 4 volumes de controle vizinhos. Como pode ser observado na Eq.(3.5), a ortogonalidade da malha pode facilitar a resolução das equações de conservação. Dessa forma, uma das maneiras de se avaliar a qualidade de uma malha é justamente analisar o quanto aquela malha se distancia de uma malha ortogonal, avaliando o ângulo formado entre a normal de uma face e a linha que une os centroides dos dois volumes que ela separa. Quanto maior esse ângulo, menos ortogonal a malha é, e maior será a dificuldade numérica imposta pela malha. Uma malha pode ser estruturada sem ser ortogonal, como malhas com padrões radiais ou como na Fig. 3.2. Também é possível que uma malha seja estruturada por blocos.

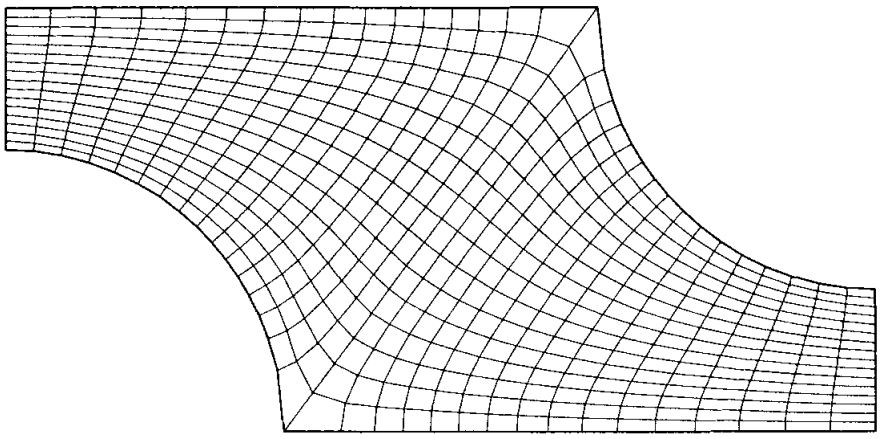


Figura 3.2 – Exemplo de malha estruturada. Fonte: Ferziger e Peric (2002).

Por fim, uma malha pode ser não-estruturada, quando a malha não segue os padrões descritos acima. As malhas não-estruturadas tem maior flexibilidade para discretizar meios complexos, e permite que os volumes de controle tenham diferentes formas, inclusive permitindo elementos de formatos diferentes dentro de uma mesma malha. Porém, mesmo com toda essa flexibilidade, uma malha não estruturada não será interessante se ela apresentar altos índices de “não-ortogonalidade” ou se a dificuldade de se definir os elementos vizinhos levar a obtenção de uma matriz que não seja adequada para a sua solução. A Fig. 3.3 mostra um exemplo de malha não estruturada.

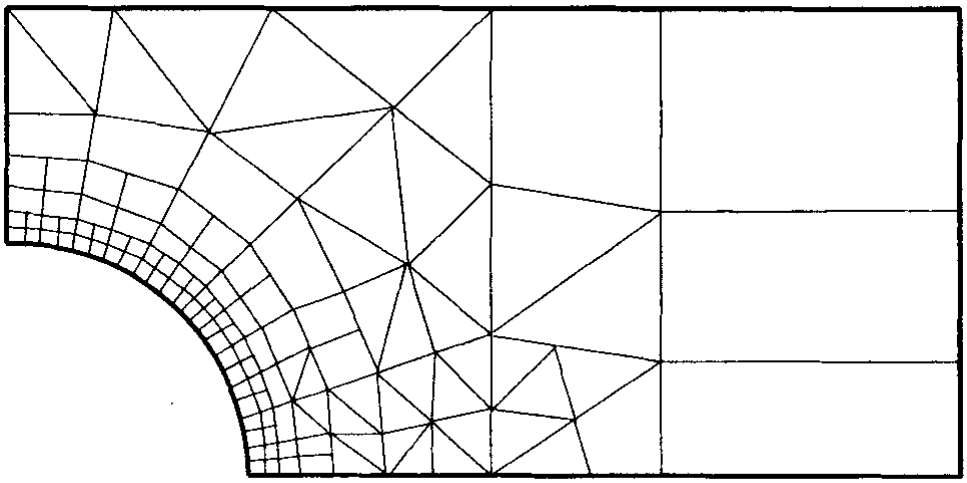


Figura 3.3 – Exemplo de malha não-estruturada. Fonte: Ferziger e Peric (2002).

3.1.5 Métodos de aproximação

Para se obter as aproximações das equações diferenciais parciais obtidas através do método de volumes finitos, esquemas numéricos devem ser adotados para aproximar as integrais e derivadas presentes no modelo matemático. No método de elementos finitos, a função de forma e a função peso devem ser definidas. A escolha desses esquemas numéricos influencia a precisão do modelo, a velocidade do código e o uso de memória para a solução do sistema.

Os métodos de aproximação são classificados de acordo com a menor ordem do erro gerado por aquela aproximação. Um método de interpolação que adota uma função de segundo grau não consegue modelar funções de graus maiores que o segundo, logo o erro provocado por essa interpolação é de pelo menos terceira ordem. Quanto maior a ordem do método, maior sua complexidade e mais rápido seu erro tende a diminuir.

Por exemplo, para se discretizar a Eq.(3.8) é necessário aplicar uma discretização temporal e uma discretização espacial. Para a discretização espacial, o seguinte procedimento pode ser adotado:

$$\int_S \mathbf{n} dS \cdot \rho\phi\mathbf{u} = \sum_f \int_f \mathbf{n} dS \cdot \rho\phi\mathbf{u} \tag{3.9}$$

onde o subíndice *f* indica que a superfície *S* do volume de controle está sendo avaliada em cada face individualmente. Utilizando o teorema do ponto central para aproximar a integral, tem-se:

$$\sum_f \int_f \mathbf{n} dS \cdot \rho\phi\mathbf{u} \approx \sum_f \mathbf{S}_f \cdot (\overline{\rho\phi\mathbf{u}})_f = \sum_f \mathbf{S}_f \cdot (\rho\phi\mathbf{u})_{fc} \tag{3.10}$$

onde $\mathbf{S}_f = S_f\mathbf{n}$ e o subíndice *fc* indica que os termos são avaliados no centroide da face. O mesmo procedimento pode ser aplicado ao termo difusivo da Eq.(3.8):

$$\int_S \mathbf{n} dS \cdot \rho\Gamma_\phi \nabla(\phi) = \sum_f \int_f \mathbf{n} dS \cdot (\rho\Gamma_\phi \nabla(\phi))_f \approx \sum_f \mathbf{S}_f \cdot (\overline{\rho\Gamma_\phi \nabla(\phi)})_f = \sum_f \mathbf{S}_f \cdot (\rho\Gamma_\phi \nabla(\phi))_{fc} \tag{3.11}$$

O termo fonte pode ser linearizado da seguinte maneira:

$$\int_\Omega q_\phi d\Omega = S_c\Omega_P + S_p\Omega_P\phi_P \tag{3.12}$$

onde o Ω_P é o volume do volume de controle cujo centroide é o ponto P, S_c e S_p representam a parte constante e o coeficiente linear do termo fonte, respectivamente. ϕ_P é a propriedade ϕ avaliada no centroide P. Substituindo as expressões das Eq.(3.10), Eq.(3.11) e Eq.(3.12) na Eq.(3.8):

$$\frac{\partial}{\partial t}\int_\Omega \rho\phi d\Omega + \sum_f \mathbf{S}_f \cdot (\rho\phi\mathbf{u})_{fc} = \sum_f \mathbf{S}_f \cdot (\rho\Gamma_\phi \nabla(\phi))_{fc} + S_c\Omega_P + S_p\Omega_P\phi_P \tag{3.13}$$

Por fim, para a discretização temporal, a Eq.(3.13) é integrada entre o tempo *t* e $t + \Delta t$, assumindo a seguinte forma:

$$\int_t^{t+\Delta t} \left[\left(\frac{\partial \rho\phi}{\partial t}\right)_P \Omega_P + \sum_f \mathbf{S}_f \cdot (\rho\phi\mathbf{u})_{fc} - \sum_f \mathbf{S}_f \cdot (\rho\Gamma_\phi \nabla(\phi))_{fc} \right] dt = \int_t^{t+\Delta t} (S_c\Omega_P + S_p\Omega_P\phi_P)\, dt \tag{3.14}$$

E então um método de discretização temporal é utilizado para se aproximar a derivada temporal. Na Eq.(3.14) é possível observar que são utilizados tanto valores obtidos no centroide *P* quanto nos centroides das faces do volume de controle. Dessa forma, além de todas as aproximações já aplicadas no modelo matemático, é necessário definir um método de interpolação para que seja possível escrever todas as gradezas necessárias para a solução da

equação em função dos valores dos centroides. Dois métodos de interpolação comumente adotados em CFD são o método das diferenças centradas e o método *Upwind*. Além desses dois métodos, é possível identificar uma série de outros, e também é possível se obter métodos de ordens superiores utilizando esquemas mais complexos, que usam valores de mais pontos da malha. Em malhas não ortogonais, é possível adicionar um termo de correção na discretização a fim de mitigar os erros que podem ser gerados pela escolha da malha.

Método das diferenças centradas

O método das diferenças centradas propõe uma interpolação linear entre o centroide *P* e os centroides vizinhos. Dessa forma, se a malha for ortogonal e os dois volumes tiverem exatamente as mesmas dimensões, o valor na interface entre os dois volumes será a média do valor do dois centroides. Para volumes com dimensões diferentes, o valor da face será uma média ponderada entre os valores dos centroides. Como o método das diferenças centradas utiliza uma função linear, este método é de segunda ordem. A Fig. 3.4 apresenta um exemplo de uma interpolação pelo método das diferenças centradas, onde o valor das faces ϕ_e e ϕ_w são calculados usando os valores dos centroides ϕ_P, ϕ_W, ϕ_E

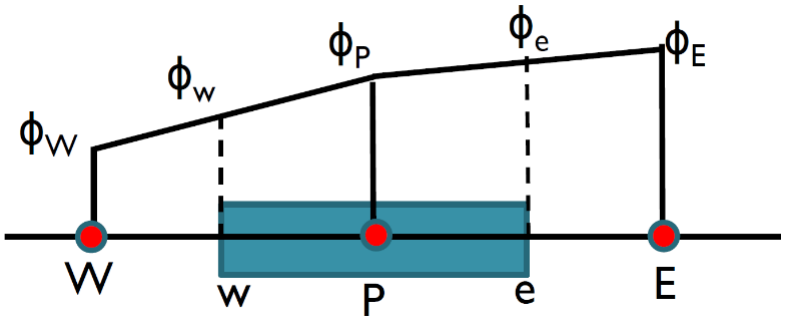


Figura 3.4 – Exemplo de uma interpolação pelo método das diferenças centradas. Fonte: Guerrero (2018).

Método Upwind

O método *Upwind* interpola o valor na face em função do fluxo sobre essa face, da seguinte maneira:

$$\phi_e = \begin{cases} \phi_P & \text{se} \quad F_e > 0 \\ \phi_E & \text{se} \quad F_e < 0 \end{cases} \tag{3.15}$$

onde o subíndice *e* indica que aquela propriedade foi avaliada na superfície *e*, e ϕ_P e ϕ_E são os valores das propriedades avaliados nos centroides *P* e *E*, respectivamente. Como o método upwind usa uma função constante para realizar a interpolação, ele é considerado de primeira ordem. *F* é fluxo de ϕ sobre uma determinada face, e é definido como:

$$F = \mathbf{S}_f \cdot (\rho\phi\mathbf{u})_{fc} \tag{3.16}$$

A Fig. 3.5 apresenta um exemplo de interpolação utilizando o método *Upwind*.

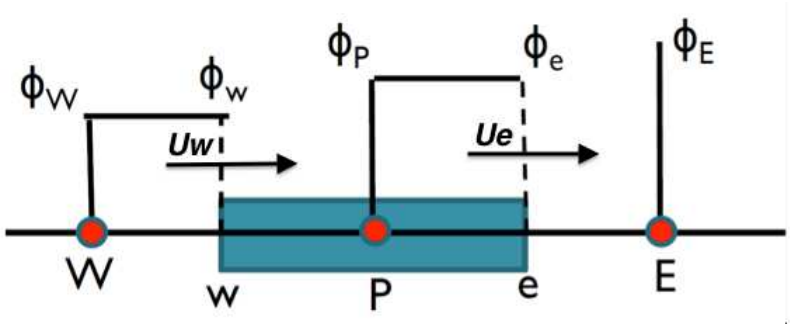


Figura 3.5 – Exemplo de interpolação utilizando o método *Upwind*. Fonte: Guerrero (2018).

3.1.6 Métodos de solução

Uma vez obtidas as equações aproximadas, é necessário definir o método com que elas serão resolvidas. A escolha do método depende da natureza do problema, por exemplo: no caso de problemas de escoamento permanente são usados esquemas iterativos onde sucessivas linearizações são realizadas a fim de se obter e resolver um sistema linear a cada iteração. Ou seja, uma vez devidamente discretizada, a Eq.(3.14) pode ser linearizada na seguinte forma:

$$\mathbf{M}\phi^j = \mathbf{b} \tag{3.17}$$

onde ϕ^j é o vetor que contem os valores da propriedade ϕ em cada um dos centroides dos volumes de controle da malha, na iteração *j*. Em simulações em regime transiente, *j* representa o tempo da qual aquela equação faz parte. ***M*** contém os coeficientes associados aos valores de ϕ e ***b*** é um vetor com os valores independentes de ϕ. ***M*** é uma matriz quadrada, e cada uma das linhas de ϕ^j, ***M*** e ***b*** representam a equação Eq.(3.14) aplicada em um centroide de um volume de controle. É interessante que a matriz ***M*** seja diagonalmente dominante, uma

vez que para se obter o vetor ϕ^j é necessário inverte-la. Essa é uma das razões do porque a ortogonalidade de uma malha pode facilitar a solução do sistema.

Existem diversas formas de se classificar os métodos de solução:

- Métodos explícitos: Nos métodos explícitos ϕ^j é calculado usando unicamente os valores da iteração anterior, já conhecida. Dessa forma, a equação para cada volume pode ser resolvida individualmente. Métodos explícitos podem não ter sua estabilidade garantida, são mais simples de se programar e exigem um número maior de iterações, o que normalmente significa um tempo maior de simulação. A Fig. 3.5 ilustra um método explicito, onde cada uma das variáveis da iteração *j* só depende dos valores da iteração *j-1*.

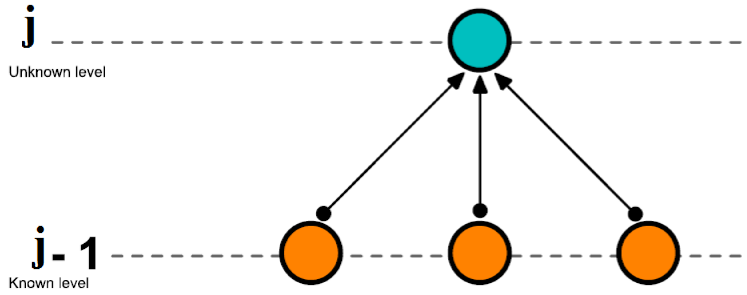


Figura 3.6 – Representação esquemática de um método explícito. Fonte: adaptado de Guerrero (2018).

- Métodos implícitos: Nos métodos implícitos, as equações são resolvidas usando não somente valores da iteração anterior, mas da própria iteração também. Dessa forma, as equações do sistema devem ser resolvidas de maneira simultânea. Métodos implícitos são incondicionalmente estáveis e exigem menos iterações para se chegar na solução, mas demandam mais memória para serem resolvidos. A Fig. 3.5 representa um método implícito, onde as variáveis da iteração *j* são calculadas usando outros valores da iteração *j* e os valores da iteração *j-1*.

- Métodos segregados: Como o modelo matemático pode conter diversas equações diferentes, os métodos segregados resolvem cada uma das equações por vez, resolvendo primeiro a conservação da quantidade de movimento linear e em seguida resolvendo a equação de conservação de massa, por exemplo. Dessa forma, o sistema de equações a ser resolvido de cada vez é menor e as matrizes são mais fáceis de se inverter. Em

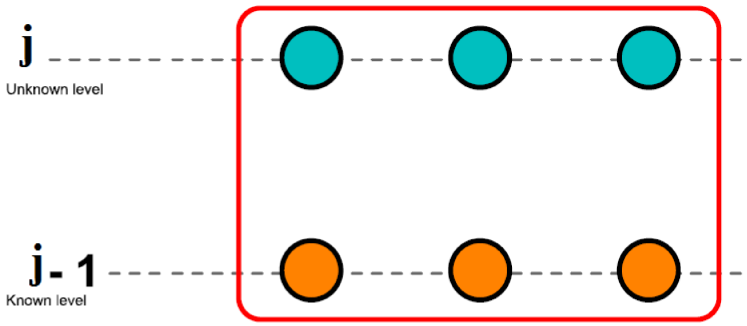


Figura 3.7 – Representação esquemática de um método implícito. Fonte: adaptado de Guerrero (2018).

compensação, os métodos segregados demandam um número de iterações maior para se obter a solução.

- Métodos acoplados: Ao contrário dos métodos segregados, os métodos acoplados resolvem todas as equações do modelo matemático de maneira simultânea, o que exige um número menor de iterações, mas aumenta a dificuldade de se resolver o sistema linear obtido, devido ao tamanho das matrizes envolvidas.

Entre os métodos segregados, uma das estratégias para se resolver as equações de conservação é escrever os campos de velocidade e pressão da seguinte forma:

$$\mathbf{U} = \mathbf{U}^* + \mathbf{U}' \tag{3.18}$$

$$\mathbf{P} = \mathbf{P}^* + \mathbf{P}' \tag{3.19}$$

onde $\mathbf{U}^*$ e $\mathbf{P}^*$ são previsões dos campos de velocidade e pressão, e $\mathbf{U}'$ e $\mathbf{p}'$ são as correções desses campos. Os métodos "*Pressure-correction*", por exemplo, buscam a estimativa do campo de velocidade resolvendo a equação da conservação da quantidade de movimento linear, usando o campo de pressão da iteração anterior. A seguir, essa estimativa é utilizada para se resolver a equação de Poisson, e assim a correção do campo de pressão é obtida, e os campos de pressão e velocidade são corrigidos. Dependendo do método, essa etapa de correção pode ser aplicada diversas vezes. Uma vez terminada a correção, os campos de pressão e de velocidade são avaliados pelo critério de convergência. Estes métodos são intrinsecamente implícitos.

Equação de Poisson

Os métodos "*Pressure-correction*" utilizam a equação de conservação da quantidade de movimento linear para se estimar a velocidade, e o termo de correção da pressão deve garantir que equação de conservação de massa seja resolvida. Para se evidenciar a influência do campo de pressão na solução da equação de conservação de massa, é necessário alguma manipulação algébrica. Uma solução é buscar a equação de Poisson para a pressão, da seguinte maneira: a equação da conservação da quantidade de movimento linear, quando escrita na forma da Eq.(3.17) apresenta os seguintes termos:

$$\mathbf{MU} = -\nabla \mathbf{P} \tag{3.20}$$

O lado direito da Eq.3.20 pode ser reescrito como:

$$\mathbf{MU} = \mathbf{AU} - \mathbf{H} \tag{3.21}$$

$$\mathbf{H} = -\mathbf{H'U} \tag{3.22}$$

onde ***A*** é a diagonal da matriz ***M***, e ***H'*** é a matriz com os termos restantes de ***M***. Substituindo a Eq.(3.21) na Eq.(3.20) e isolando ***U***, tem-se:

$$\mathbf{U} = -\mathbf{A}^{-1} \cdot \nabla p + \mathbf{A}^{-1} \cdot \mathbf{H} \tag{3.23}$$

Por fim, aplicando o operador divergente dos dois lados da Eq.(3.23):

$$\nabla \cdot \mathbf{U} = -\nabla \cdot (\mathbf{A}^{-1} \cdot \nabla p) + \nabla \cdot (\mathbf{A}^{-1} \cdot \mathbf{H}) \tag{3.24}$$

Assim, para que a equação da continuidade (Eq.(3.2)) seja verdadeira, basta igualar o lado direito da Eq.(3.24) a zero. Logo:

$$\nabla \cdot (\mathbf{A}^{-1} \cdot \nabla p) = \nabla \cdot (\mathbf{A}^{-1} \cdot \mathbf{H}) \tag{3.25}$$

Método SIMPLE

O método SIMPLE é um método "*Pressure-correction*" proposto por Patankar e Spalding (1972) onde é realizada apenas uma correção a cada iteração. SIMPLE é um acrônimo para *"Semi-Implicit Method for Pressure Linked Equations"*. Existem várias modificações desse método, como o SIMPLEC (DOORMAAL; RAITHBY, 1984) e o SIMPLER (PATANKAR, 1980), que buscam otimizar o processo de solução. O fluxograma do método SIMPLE está representado na Fig. 3.8.

Método PISO

O método PISO também é um método "*Pressure-correction*", proposto por Issa (1985), onde é realizada mais de uma correção a cada iteração. Este método foi originalmente desenvolvido para problemas transientes. PISO é um acrônimo para *"Pressure-Implicit with Splitting of Operators"*. O fluxograma do método PISO está representado na Fig. 3.9. O método PIMPLE é uma combinação do SIMPLE e do PISO.

escoadaxação

Principalmente em simulações de escoamentos em regime permanente, é possível usar fatores de relaxação para aumentar a estabilidade do método. Existem duas técnicas principais de relaxação: a relaxação de campo e a relaxação na matriz. Nos dois casos, quanto menor o fator de relaxação, mais estável é o método, mas maior o número necessário de iterações para o resultado convergir.

A relaxação de campo age diretamente sobre o campo de uma propriedade, como na equação Eq.(3.26). Na solução das equações de conservação, essa técnica é normalmente utilizada sobre o campo de pressão.

$$\phi^j = \phi^{j-1} + \alpha_\phi(\phi^j_{calc} - \phi^{j-1}) \tag{3.26}$$

onde ϕ^j e ϕ^{j-1} são os valores da propriedade ϕ na iteração *j* e *j-1*, respectivamente. α_ϕ é o fator de relaxação da propriedade ϕ e ϕ^j_{calc} é o valor da propriedade ϕ calculado na iteração *j*. Ou seja: o valor de ϕ^j é diferente do valor de ϕ^j_{calc}. Se α_ϕ<1, a relaxação de campo diminui o quanto a propriedade ϕ varia a cada iteração, o que dificulta que ϕ divirja, mas ao mesmo

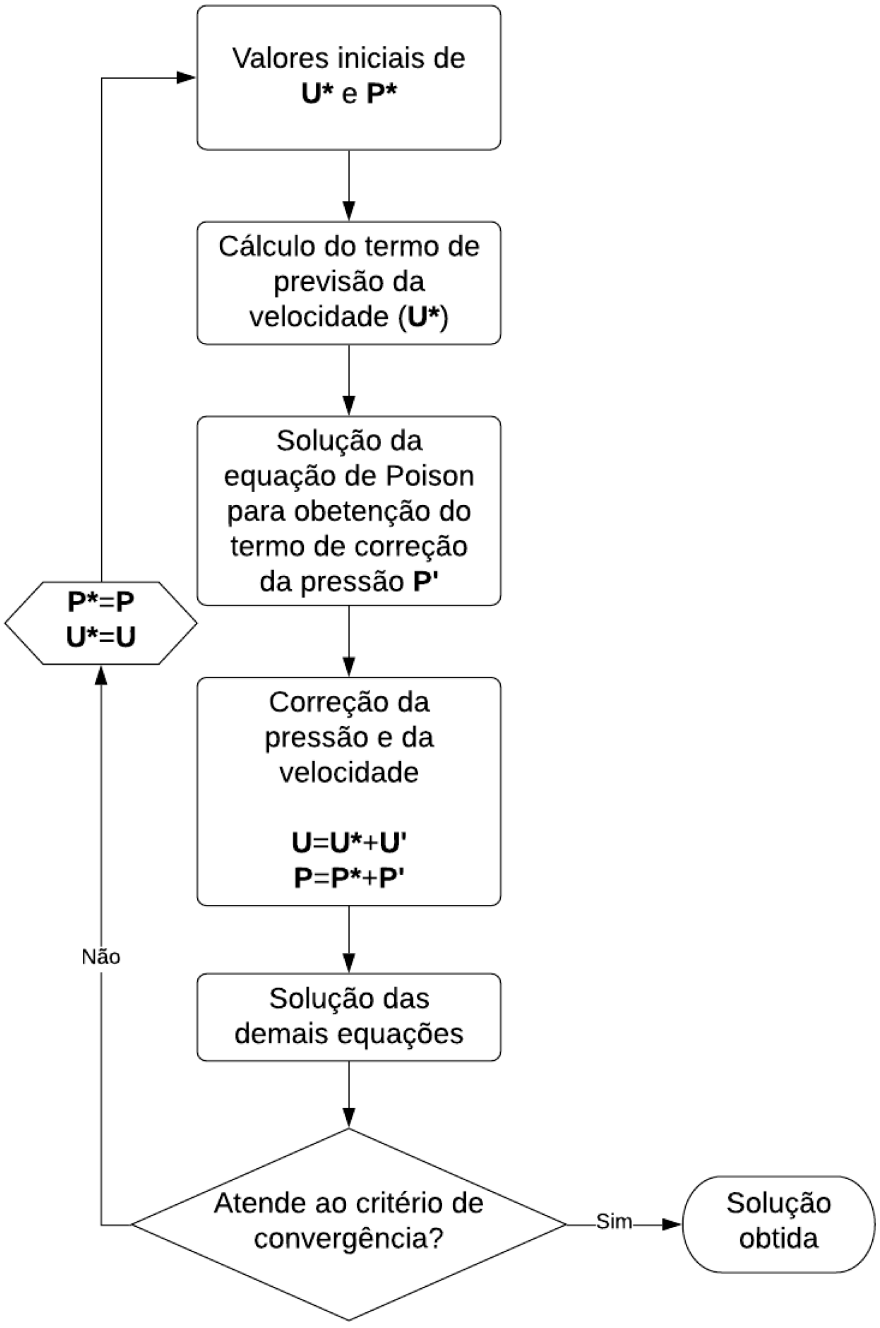


Figura 3.8 – Fluxograma do método SIMPLE. Fonte: adaptado de Guerrero (2018).

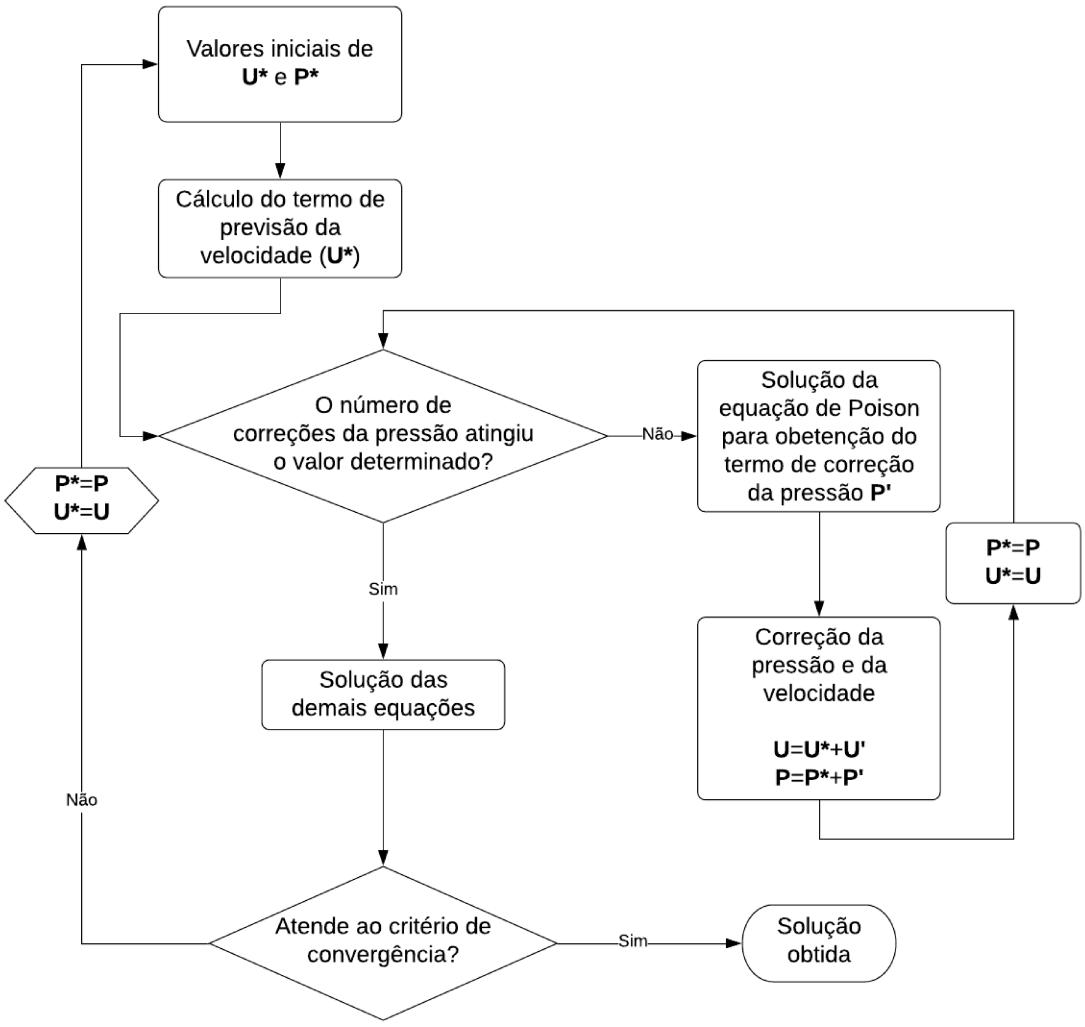


Figura 3.9 – Fluxograma do método PISO. Fonte: adaptado de Guerrero (2018).

tempo diminui a velocidade com que o método se aproxima da solução. Alguns sistemas permitem o uso de $\alpha_\phi > 1$. Essa é uma estratégia utilizada para se aumentar a eficiência de cálculo, uma vez que se $\alpha_\phi > 1$, a propriedade ϕ pode se aproximar mais da solução em um número menor de iterações.

Já a relaxação de matriz age nas matrizes a serem resolvidas pelo sistema, tornando-as mais adequadas para a solução dos sistemas lineares. Para se resolver a Eq.(3.21), por exemplo, é importante que **M** seja uma matriz diagonalmente dominante, ou seja:

- O valor do módulo do elemento da diagonal em cada coluna deve ser ao menos igual ao valor da soma dos módulos dos elementos fora da diagonal
- Ao menos uma coluna tem o valor do módulo do elemento da diagonal maior que o valor da soma dos módulos dos elementos fora da diagonal

Assim, a relaxação de matriz divide os elementos da diagonal pelo fator de relaxação, o que aumenta o valor dos elementos na diagonal se o fator de relaxação for menor que a unidade. Para garantir a consistência do método, os termos independentes também devem ser modificados, o que aumenta o caráter explicito do método e assim aumenta o número de iterações necessárias para se chegar à solução (HOLZMANN, 2016).

3.1.7 Critério de convergência

Como são métodos não exatos, é necessário estabelecer um critério de convergência para os métodos empregados. Normalmente, existem dois níveis de iteração: um nível de iteração para se resolver os sistemas lineares e outro nível mais abrangente, que cobre as não-linearidades das equações e o acoplamento entre elas. O critério de convergência tem forte influência na precisão dos resultados e na eficiência do código.

Os dois principais critérios de convergência são o monitoramento de propriedades de interesse e o resíduo normalizado máximo. O monitoramento de propriedades de interesse avalia a variação das propriedades em um intervalo de iterações, caso a variação seja menor que o critério definido, o sistema é dado como convergido. Já o resíduo normalizado r_{norm} de uma equação na forma da Eq.(3.17) pode ser definido pela expressão:

$$r_{norm} = \frac{1}{n_r} \sum (|\mathbf{b} - \mathbf{M}\phi^j|) \tag{3.27}$$

$$n_r = \sum (|\mathbf{M}\phi^j - \mathbf{M}\overline{\phi^j}| + |\mathbf{b} - \mathbf{M}\overline{\phi^j}|) \tag{3.28}$$

onde $\overline{\phi^j}$ é o valor médio de ϕ^j.

Graças à normalização, r_{norm} varia entre zero e um, e quanto mais próximo de zero, mais precisa é a solução. Dessa forma, quando os resíduos de todas as equações de uma determinada iteração ficam abaixo do critério estabelecido, o sistema é considerado como convergido.

3.2 Propriedades de um modelo numérico

Para se escolher um método adequado para uma determinada aplicação, certas características de seus componentes devem ser analisadas. Cada componente influencia as propriedades do método de uma maneira diferente, e a escolha de um componente que não esteja alinhado com os outros pode funcionar como um gargalo para aquela propriedade. As principais propriedades de um método numérico são (FERZIGER; PERIC, 2002):

- Consistência: a diferença entre as equações discretizadas e a solução exata é chamada de erro de truncamento. Um método é consistente quando o erro de truncamento tende a zero quando os subdomínios no tempo e no espaço tendem a zero também.
- Estabilidade: Um método é estável quando os erros tendem a diminuir a cada iteração.
- Convergência: um método numérico é convergente quando a solução das equações discretizadas tendem à solução das equações diferenciais parciais.
- Conservação: Como as equações a serem resolvidas são equações de conservação, os esquemas numéricos empregados também devem ter essa propriedade. O uso de esquemas não conservativos não impede a obtenção de soluções corretas, mas dificulta a avaliação dos erros.
- Realizabilidade: Ainda que algumas soluções sejam matematicamente possíveis, elas não tem significado físico. Uma densidade negativa ou igual a zero normalmente não

tem significado físico em problemas tratados por CFD, por exemplo. Assim, os modelos adotados devem ser escolhidos de forma a garantir resultados realísticos. Métodos que não respeitam essa característica podem ter mais problemas de convergência e estabilidade.

- Precisão: As soluções numéricas geram resultados aproximados, e as diversas etapas desse processo podem introduzir erros de diversas naturezas. Os principais tipos de erros são:
  - Erros de modelagem: São as diferenças entre o escoamento real e a solução exata do modelo matemático proposto.
  - Erros de discretização: É a diferença entre a solução exata das equações de conservação e a solução exata do sistema linear obtido.
  - Erros de iteração: É a diferença entre a solução exata do sistema linear obtido e a solução obtida em uma determinada iteração.

CAPÍTULO IV

ELABORAÇÃO DOS CÓDIGOS EM OpenFOAM®

Este capítulo tem por objetivo descrever todas as edições necessárias para se obter as rotinas do software OpenFOAM® para se resolver escoamentos internos em regime permanente e laminar de fluidos não newtonianos com transferência de calor. Para modelar a convecção natural, será adotada a hipótese de Boussinesq (BOUSSINESQ, 1903). Em seguida são expostas duas rotinas para se modelar o comportamento viscoplástico: uma para o modelo de bi-viscosidade e outra para o modelo SMD. Os códigos em seu formato integral estão disponíveis nos Apêndices 10.1 e 10.2.

4.1 OpenFOAM®

Em 1989, Henry Weller desenvolveu o *software* FOAM, um acrônimo para *Field Operation and Manipulation*, capaz de realizar operações com campos tensoriais. Em 2004 o FOAM teve seu código liberado e passou a se chamar OpenFOAM (*Open Field Operation and Manipulation*). Hoje, o OpenFOAM® é um conjunto de módulos escritos em linguagem C++, utilizado para realizar resolver problemas de engenharia, como a simulação de escoamentos. Como o OpenFOAM® segue uma programação orientada a objetos e é um *software* livre, ele se destaca por ser gratuito e pela sua capacidade de customização e flexibilidade.

O pacote de rotinas padrão do OpenFOAM® apresenta uma série de *solvers* que usam o método dos volume finitos para a solução de diversos problemas de engenharia. Estes *solvers* adotam o método SIMPLE, o método PISO e o método PIMPLE para resolver as equações do sistema. Ainda que o método de solução seja próprio do *solver*, o OpenFOAM® define uma sé-

rie de parâmetros numéricos na configuração da simulação, como os métodos de aproximação e interpolação e os *solvers* lineares, como será discutido na Seção 5.

A função mais básica para se gerar malhas no OpenFOAM® é a função *blockmesh*, que gera malhas ao menos estruturadas por partes. Porém, é possível trabalhar com malha não estruturadas no OpenFOAM®, gerando a malha com outro *software* e usando as funções de conversão de malha que o OpenFOAM® oferece.

4.2 nonNewtonianSimpleFoam

4.2.1 Objetivos

O objetivo desta seção é apresentar uma rotina de OpenFOAM® que seja não somente uma ferramenta eficiente para se resolver escoamentos não newtonianos, mas que também seja acessível a um público que tem pouco ou nenhuma experiência com este *software*. Dessa forma, as soluções apresentadas nesse capítulo buscam balancear essas duas características: busca-se explorar ao máximo as ferramentas que o OpenFOAM® oferece, mas sempre utilizando arranjos simples e de fácil reprodução.

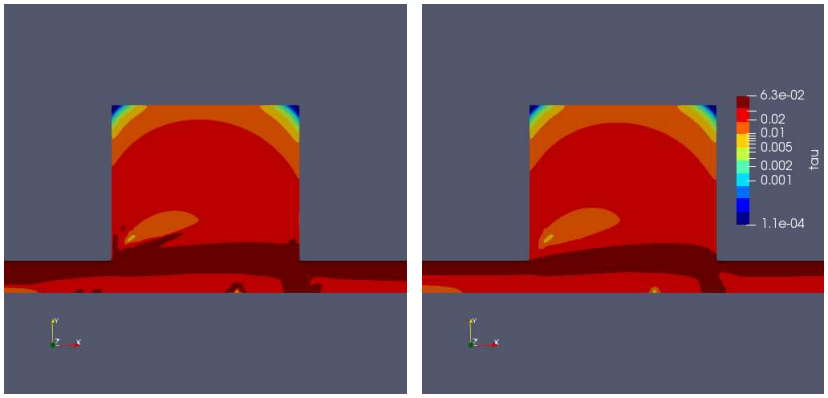
Uma das características do OpenFOAM® que mais chama a atenção de quem inicia a utilização desse *software* é que suas rotinas em geral são curtas e suas funções muitas vezes apresentam grande semelhança com equações literais. Isso acontece porque o OpenFOAM® utiliza uma estrutura de códigos onde um pequeno arquivo pode inicializar uma série de funções, que por sua vez possuem subfunções, e assim sucessivamente. Cada nível de cálculo desses tem suas características: certas funções podem não estar definidas em alguns desses níveis, por exemplo. Alterar os níveis mais básicos dessa estrutura pode não ser interessante, tanto do ponto de vista de eficiência do código, quanto da dificuldade de se editar todas funções associadas à ele. Dessa forma, os códigos aqui apresentados buscam sempre manter e usar a estrutura que o OpenFOAM® já oferece. Por exemplo, ainda que em todo o presente trabalho a viscosidade aparente seja representada pela letra “eta”, nos códigos o nome da variável que representa a viscosidade aparente é “nu”, que é o modo como as funções do OpenFOAM® chamam a viscosidade aparente por padrão. Alterar esse nome não seria necessariamente trabalhoso, mas envolveria a edição de vários códigos do pacote padrão do OpenFOAM®, o que abriria mais espaço para erros e pode exigir um conhecimento mais profundo das rotinas

envolvidas.

Além disso, certos comportamentos não newtonianos podem acrescentar um caráter não-linear importante na solução dos sistemas quando comparados com escoamentos newtonianos. É necessário ter essa característica sempre em mente ao se trabalhar com esses códigos: além da dificuldade numérica para se resolver os sistemas, essa característica impõe desafios importantes sobre a etapa de pós-processamento, chegando ao ponto onde alguns métodos numéricos possam ter sua validade comprometida.

A Fig. 4.1 trás um exemplo dessa dificuldade. Nela está representado o campo de tensão cisalhante de um escoamento de um fluido viscoplástico. As duas imagens foram obtidas usando o software $paraFoam$, que é a versão do software livre $ParaView$ do pacote OpenFOAM®. Os dois campos foram calculados usando a relação $\tau = \eta\dot{\gamma}$, a partir dos mesmos campos de viscosidade aparente e $\dot{\gamma}$. A diferença entre as imagens é que um campo foi calculado pelo próprio OpenFOAM®, durante a solução do escoamento, e a outra foi calculada diretamente pelo $paraFoam$. *Plug-zones* são zonas aparentemente não escoadas móveis que surgem próximas às linhas de simetria de um canais devido às baixas tensões de cisalhamento dessa região. Na Fig. 4.1 é possível observar uma pequena *plug-zone* que começa a ser formada logo abaixo da cavidade, como mostra o ponto de menor tensão próximo da linha de simetria. Movendo-se do interior para o exterior dessa *plug-zone*, a Fig. 4.1(a) indica um pico de tensão cisalhante nos limites da *plug-zone*. Esse resultado não apresenta significado físico: o $paraFoam$ interpola os resultados do campo de viscosidade aparente e de taxa de deformação dos centroides para os nós da malha antes de calcular o campo de tensão. Como a taxa de deformação e a viscosidade são propriedades com forte comportamento exponencial na região onde a tensão cisalhante é ligeiramente maior que τ_0, ao tratar os dados no $paraFoam$, os erros da interpolação se tornam grandes o suficiente para comprometer os resultados obtidos. Por outro lado, ao calcular o campo de tensão no próprio OpenFOAM®, todos os campos são calculados para os centroides, o que evita esse problema.

Dessa forma, para tentar manter a qualidade dos resultados, os códigos que são apresentados neste trabalho buscam agregar o máximo possível do pós-processamento no próprio OpenFOAM®, a fim de assegurar que os dados estão sendo tratados da maneira mais eficiente possível. Uma das vantagens dessa estratégia é a possibilidade de se acompanhar a evolução dos resultados durante a simulação. Essa propriedade pode ser uma ferramenta importante



(a) paraFoam (b) OpenFOAM®

Figura 4.1 – Exemplo da diferença provocada por diferentes métodos de pós-processamento. τ calculado na geometria de canal com expansão-contração (a) pelo OpenFOAM® e (b) pelo software paraFoam. Ambas as figuras compartilham a mesma escala.

para se otimizar procedimentos, ajudando a entender como cada propriedade se comporta em função do número de iterações, por exemplo, e ajudando a verificar a convergência dos resultados. Porém, o pós-processamento também particulariza o código, podendo chegar ao extremo de ser necessário elaborar um *solver* novo para cada aplicação. Além disso, a adição de novas rotinas no código pode fazer ele perder eficiência, de acordo com a frequência que essas rotinas são ativadas e da sua complexidade.

Para se manter a fluidez e clareza do texto, sempre que for necessário referenciar uma linha específica do código, somente serão utilizados os códigos fonte: os códigos padrão do pacote do OpenFOAM® e os códigos apresentados nos apêndices deste trabalho. Ou seja, o número da linha citado neste trabalho pode ser diferente do número da linha do código em edição. Dessa forma, padroniza-se o processo de edição do código e permite que o programador organize o seu código da maneira que lhe for mais conveniente. Aos que desejarem apenas utilizar os códigos apresentados nesta seção, todos eles estão compilados nos Apêndices 10.1 e 10.2 deste trabalho. A numeração das linhas dificulta o simples “copiar-colar”, então recomenda-se usar comandos um pouco mais elaborados, como manter a tecla *alt* pressionada ao se selecionar o texto, o que em alguns softwares permite selecionar apenas uma área do documento, excluindo assim a numeração.

Por fim, as boas práticas de programação também são recomendadas, tais como sempre manter um *backup*, manter os comentários e descrições do códigos atualizados, indentar o código e trabalhar em um ambiente adequado, com permissão para criar e editar documentos,

por exemplo. Devido ao formato deste trabalho, nem sempre foi possível manter a indentação correta dos códigos aqui apresentados. Nos códigos do OpenFOAM®, o símbolo utilizado para indicar um comentário é "//".

As rotinas aqui apresentadas foram elaboradas para OpenFOAM® v5, e testadas em versões do *software* tanto para Ubuntu quanto para Windows®. Não é necessária nenhuma adaptação para que as rotinas sejam utilizadas com a versão mais recente do OpenFOAM®, a v6.

4.2.2 Modelo Mecânico

O código *nonNewtonianSimpleFoam* é o *solver* desenvolvido neste trabalho. Nesta rotina, as equações são definidas e resolvidas segundo o método SIMPLE. As hipóteses de fluido incompressível e de fluido newtoniano generalizado são adotadas. O escoamento será considerado em regime permanente e a dissipação viscosa não será considerada no balanço de energia. Por fim, o termo da aproximação de Boussinesq foi adicionado na equação de balanço de quantidade de movimento linear. Assim, a partir dessas considerações, as equações de balanço de massa, de quantidade de movimento linear e de energia são iguais a:

$$\nabla \cdot \mathbf{u} = 0 \tag{4.1}$$

$$\rho(\nabla \mathbf{u})\mathbf{u} = -\nabla p + \nabla \cdot \boldsymbol{\tau} + \rho \mathbf{g} \beta (T - T_c) \tag{4.2}$$

$$(\nabla T)\mathbf{u} = \alpha \nabla^2 T \tag{4.3}$$

onde **g** é o vetor gravidade, β é o termo de expansão térmico, T é a temperatura local e T_c é a temperatura de referência.

4.2.3 Solvers similares

Entre os solvers padrão do OpenFOAM®, os dois mais próximos ao *solver* desenvolvido neste trabalho são o *simpleFoam* e o *nonNewtonianIcoFoam*. Ambos pode ser encontrados na pasta *applications**solvers**incompressible*, dentro do diretório do OpenFOAM®.

- *nonNewtonianIcoFoam*: é um *solver* para escoamentos incompressíveis e em regime transiente, e suas equações levam em consideração a característica não newtoniana

do escoamento. Dentro da pasta deste código, apenas duas rotinas estão presentes: *createFields.H* e *nonNewtonianIcoFoam.C*. A rotina *createFields.H* declara as principais variáveis do código, e *nonNewtonianIcoFoam.C* é o *solver* em si, onde todas as equações estão presentes. A equação de balanço de energia não é resolvida. A pasta *Make* contém arquivos que dão suporte para a compilação da rotina.

- *simpleFoam*: é um *solver* para escoamentos incompressíveis em regime permanente, e que permite a modelização de escoamentos newtonianos turbulentos. Além dos arquivos semelhantes aos descritos anteriormente, a pasta desse *solver* contém duas subpastas com variações deste código, e cada equação do modelo mecânico é descrita em um arquivo separado, tal como *UEqn.H* e *pEqn.H* para a velocidade e a pressão, por exemplo. Novamente, a equação de balanço de energia não é resolvida.

Dadas as características do *solver* a ser desenvolvido, o código do *simpleFoam* será editado para resolver as equações que levam em consideração a hipótese do fluido newtoniano generalizado, acrescentando a equação do balanço de energia e o termo da aproximação de Boussinesq.

4.2.4 Códigos

O bloco de comandos a seguir tem por objetivo fazer uma cópia da pasta do *simpleFoam* e preparar os arquivos para serem editados. Esses comandos devem ser executados em uma janela de comando:

```
run
cp -r $FOAM_APP/solvers/incompressible/simpleFoam nonNewtonianSimpleFoam
cd nonNewtonianSimpleFoam/
wclean
rm -r SRFSimpleFoam/
rm -r porousSimpleFoam/
mv simpleFoam.C nonNewtonianSimpleFoam.C
cp UEqn.H TEqn.H
```

O comando *run* é um atalho para a pasta de mesmo nome, que é o ambiente onde é recomendado salvar os arquivos do OpenFOAM® elaborados pelo usuário, como funções e simulações. O segundo comando copia a pasta do *simpleFoam* para a pasta *run*, alterando seu nome para *nonNewtonianSimpleFoam*. *$FOAM_APP* é um atalho para a pasta *applications*, e

vários outros atalhos para pastas do OpenFOAM® seguem este formato. O comando seguinte é apenas para acessar a pasta recém criada. O comando *wclean* é uma boa prática, uma vez que ele remove os arquivos desnecessários para a compilação, em especial quando outra compilação anterior estiver causando problemas. Os dois comandos seguintes são apenas para apagar as outras versões do solver, já que é o código original do *simpleFoam* que será usado para a edição. Por fim, o arquivo *simpleFoam.C* tem seu nome alterado para o nome do novo *solver*, *nonNewtonianSimpleFoam.C*, e o arquivo que vai definir a equação de balanço de energia é criado, como uma cópia do arquivo *UEqn.H*.

TEqn.H

O conteúdo deste arquivo deve ser completamente substituído pelo seguinte:

```
fvScalarMatrix TEqn
        (
        fvm::div(phi, T)
        == fvm::laplacian(alpha_, T) );

TEqn.relax();
TEqn.solve();
```

As terceira e quarta linhas são a equação de balando de energia. É notável a semelhança entre o código e a equação diferencial literal:

```
fvm::div(phi, T)  == fvm::laplacian(alpha_, T)
```

$$(\nabla T)\mathbf{u} = \alpha \nabla^2 T \tag{4.4}$$

Para diferenciar as variáveis do código das *inputs* de cada problema, é comum adicionar o símbolo *underline* ao declarar as variáveis do código, tal como em *alpha _*. A variável *phi* representa o fluxo volumétrico para escoamentos incompressíveis e o fluxo mássico para escoamentos compressíveis. Por fim, *fvm* significa *finite volume method*, e indica um grupo de funções que calculam as derivadas implícitas e geram uma matriz de coeficientes. Em contrapartida ao *fvm*, existe o *fvc*, que significa *finite volume calculus* e calcula derivadas explicitas, retornando um campo geométrico (GREENSHIELDS, 2015).

As duas últimas linhas do código tem por função aplicar os fatores de relação e resolver o sistema montado nas linhas anteriores. Como discutido na seção 3.1.6, o fator de relaxação é aplicado sobre as matrizes do sistema a ser resolvido.

UEqn.H

O comentário na primeira linha do arquivo *UEqn.H* identifica essa equação como sendo o "*momentum predictor*", através da qual o termo de previsão da velocidade é obtido, segundo o método *SIMPLE* (Fig. 3.8). A primeira linha de código, "*MRF.correctBoundaryVelocity(U);*", é uma função para se adicionar um termo fonte para simular escoamentos rotativos, como turbinas e ventiladores. *MRF* significa *Multi reference frame*. Como este não é o objetivo deste trabalho, essa linha de código pode ser suprimida. Em seguida, a maior parte da equação de balanço de quantidade de movimento linear é definida, faltando apenas acrescentar o termo do gradiente de pressão, que é inserido na linha 21. *fvOptions* é uma estrutura que permite acrescentar termos fonte com mais facilidade, sem alterar o código.

Como já discutido, existe um *solver* com uma equação de balanço de quantidade de movimento linear adaptada para resolver escoamentos não newtonianos. Essa equação está definida entre as linhas 63 e 69 do arquivo *nonNewtonianIcoFoam.C* e representa a seguinte expressão, já com o termo do gradiente de pressão adicionado:

```
fvm::ddt(U)
+ fvm::div(phi, U)
- fvm::laplacian(fluid.nu(), U)
- (fvc::grad(U) & fvc::grad(fluid.nu()))
```

$$\frac{\partial \mathbf{u}}{\partial t} + \rho(\nabla \mathbf{u})\mathbf{u} - \eta\nabla^2\mathbf{u} - \nabla\mathbf{u}\cdot\nabla\eta = -\nabla p \tag{4.5}$$

Nesta equação, o termo difusivo da equação de balanço de quantidade de movimento linear foi divido em dois termos, utilizando a propriedade da derivada do produto:

$$\nabla\cdot\boldsymbol{\tau} = \nabla\cdot(\eta\nabla(\mathbf{u})) = \eta\nabla^2\mathbf{u} + \nabla\mathbf{u}\cdot\nabla\eta \tag{4.6}$$

Como o objetivo deste trabalho é obter uma rotina para escoamentos em regime permanente, a derivada temporal pode ser suprimida. Além disso, também é necessário acrescentar o termo da aproximação de Boussinesq: $\rho\mathbf{g}\beta(T - T_c)$. Para simplificar o código, será adotado que $T_c = 0$, porém outras abordagens também são válidas, como definir T_c como uma variável de entrada, por exemplo. Realizando estas edições, as linhas de 7 a 11 do código *UEqn.H* devem ser substituídas por:

```
rho_.value()*fvm::div(phi, U)
```

```
- fvm::laplacian(fluid.nu(), U)
- (fvc::grad(U) & fvc::grad(fluid.nu()))
- rho_.value()*g*beta_*T
```

Por padrão, quando simulando escoamentos incompressíveis, o OpenFOAM® divide todas as equações pela massa específica, já que ela é constante, e assim esse termo não aparece nas equações originais. Dessa forma, *phi* representa o fluxo mássico para escoamentos compressíveis e passa a representar o fluxo volumétrico para escoamentos incompressíveis. Como ρ pode ser uma variável importante em alguns processos de adimensionalização, esse termo foi reintroduzido. Porém, como todos os termos das equações mantêm suas unidades originais, o termo da massa específica é inserido na forma *rho_.value()*, que representa que apenas o valor de ρ, sem considerar sua unidade. Dessa forma, evita-se problemas no balanço das unidades.

pEqn.H

Essa é a etapa da obtenção dos termos de correção dos campos, como apresentado na Seção 3.1.6. Ainda que seja um pouco mais complicado de se entender o código, o objetivo dele é fazer exatamente a mesma manipulação algébrica apresentada na Seção 3.1.6 para se obter a equação de Poisson, resolvê-la e a seguir corrigir o campo de velocidades. As linhas 10 a 16 do arquivo *pEqn.H* são um laço que transformam o método *SIMPLE* no *SIMPLEC* quando essa opção estiver ativada. Outra diferença desta equação para as outras é que dessa vez o fator de relaxação é aplicado sobre o campo de pressão diretamente, e não sobre sua equação. A única edição possível nesse arquivo é a retirada da linha 5, que novamente trata de funções do grupo *MRF*. Mesmo se essa linha for mantida, isso não deve gerar problemas para o código.

createFields.H

O arquivo *createFields.H* é o arquivo que inicializa a maioria das variáveis, tanto variáveis internas do código quanto aquelas que serão registradas, como os resultados. No *simpleFoam*, o primeiro campo a ser declarado é o da pressão, definido como um *volScalarField*, ou seja, um campo uma grandeza escalar representado pelos volumes de controle da malha, tal como representado na Fig. 3.1. Além do *volScalarField*, existem os *volVectorField*, que são os campos de grandezas vetoriais, e também os campos de superfícies. O campo *p* também

foi definido como um *IOobject*, que são as variáveis que podem ser registradas ao longo do processo de solução do sistema. Os argumentos apresentados nas linhas seguintes definem a forma com que essa variável sera armazenada e lida: o primeiro argumento (*"p"*) é o nome do arquivo onde ela será salva; a seguir é definido onde ela será salva, no caso, na pasta da iteração/tempo atual (*runTime.timeName()*); o terceiro argumento associa esse campo a um objeto, no caso, a malha. Os dois argumentos seguintes definem quando os arquivos devem ser lidos e quando devem ser escritos, respectivamente. O argumento seguinte é o valor inicial do campo *p*.

No *simpleFoam*, apenas dois *IOobject* são declarados: *p* e **u**. Como o campo de temperatura foi adicionado, ele deve ser declarado, acrescentando após a definição do campo *U*:

```
Info<< "Reading field T\n" << endl;
volScalarField T
(
    IOobject
    (
        "T",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);
```

A seguir, o dicionário "*scalarDict*" será declarado. A função desse dicionário é armazenar informações de interesse durante a simulação. Neste exemplo, esse dicionário será utilizado para armazenar o tempo de simulação, para facilitar a comparação entre diferentes rotinas ou configurações numéricas. Exitem diversos outros usos para esse dicionário, e outras formas de se obter o tempo de simulação. A estrutura aqui proposta é um exemplo de como essa ferramenta pode ser aplicada. O código é o seguinte:

```
IOdictionary scalarDict
(
    IOobject
    (
        "scalarDict",
        runTime.timeName(),
```

```
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    )
);
```

Como já discutido, esse código tem por objetivo realizar o máximo possível do pós-processamento. Para isso, quatro outros campos serão definidos: um campo de taxa de deformação (sr), um de tensão cisalhante (*tau*), um campo para definir as zonas escoadas e aparentemente não escoadas (*yieldGama*), e por fim um campo que servirá de suporte para o cálculo da eficiência de deslocamento (*yieldGama2*), que irá computar apenas as zonas aparentemente não escoadas de uma região do escoamento. Não é necessário declarar um campo de viscosidade, pois ele já é declarado por padrão em uma rotina interna do OpenFOAM®. Os campo *tau* e *sr* poderiam ser declarados de maneira semelhante à forma que o campo de viscosidade aparente *nu* é definido, mas as edições necessárias para isso são mais complexas, envolvendo mais rotinas, por exemplo. Dessa forma, uma solução de mais fácil implementação é apresentada.

```
volScalarField    sr
(
    IOobject
    (
        "sr",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);

volScalarField    tau
(
    IOobject
    (
        "tau",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
```

```
);

volScalarField    yieldGama
(
    IOobject
    (
        "yieldGama",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);

volScalarField    yieldGama2
(
    IOobject
    (
        "yieldGama2",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);
```

Uma vez declaradas as variáveis que serão armazenadas, o código original começa a declarar as variáveis internas e algumas variáveis padronizadas, como *phi*, através da função *createPhi.H*. Também é declarado um valor de referência para a pressão, uma vez que nas equações de balanço a pressão sempre aparece na forma de um gradiente. Após o bloco de funções sobre a pressão, uma variável chamada "*laminarTransport*" é declarada, da classe "*singlePhaseTransportModel*". Essa variável será responsável por armazenar uma série de propriedades do fluido, inclusive o campo de viscosidade. As linhas seguintes do arquivo original são próprias para a simulação de ecoamentos turbulentos, o que foge do escopo deste trabalho. No código aqui proposto, a variável *laminarTransport* teve seu nome trocado por *fluid*. Assim, as seguintes linhas de código devem substituir os códigos sobre a turbulência:

```
singlePhaseTransportModel fluid(U, phi);

dimensionedVector g("g",dimAcceleration,vector(0,-1,0));

dimensionedScalar alpha_(fluid.lookup("alpha"));
```

```
dimensionedScalar rho_(fluid.lookup("rho"));
dimensionedScalar beta_(fluid.lookup("beta"));

scalar ti(readScalar(scalarDict.lookup("tempo")));
scalar ratio_(readScalar(fluid.lookup("ratio")));
dimensionedScalar gama_(fluid.lookup("gamaRef"));
```

Nessas linhas, a variável **g** é declarada, com dimensões de aceleração e igual ao vetor (0,-1,0). Por comodidade, o vetor gravidade foi definido como um vetor unitário, orientado no sentido oposto ao do eixo *y*. α, ρ e β também são declaradas, todas baseadas na variável *fluid*. Além dessas variáveis, é criada uma variável *ti*, que irá receber o tempo registrado no dicionário *scalarDict* quando a rotina *createFields.H* for executada, ou seja, quando a simulação começar ou for retomada. Este recurso permite que a simulação seja pausada e retomada sem que o registro do tempo de duração simulação seja prejudicado. A variável *ratio_* será a relação entre a área analisada para a eficiência de deslocamento e a área total da simulação. Por fim, *gama_* será a taxa de deformação usada como critério para se classificar as zonas escoadas e as zonas aparentemente não escoadas. Outras propriedades do fluido, tal como a tensão cisalhante ou a viscosidade, também podem ser utilizadas como critério.

nonNewtonianSimpleFoam.C

O arquivo *nonNewtonianSimpleFoam.C* é o *solver* em si, e todas as outras funções poderiam ser agrupadas nesse código, tal como foi feito no código *nonNewtonianIcoFoam.C*. Uma peculiaridade do *solver* subdividido em funções é que, caso ele seja compilado com sucesso, é necessário que o arquivo *.C* dele seja editado para que o OpenFOAM® identifique as alterações e assim possa compilar o programa novamente. Ainda que as outras funções tenham sido editadas, ao tentar re-compilar uma função sem que seu arquivo principal tenha sido alterado, o OpenFOAM® não identificará as edições, irá considerar que a rotina já foi compilada e terminará o comando sem executar a compilação. Ou seja, o executável continuará sem as edições realizadas. Este é outro exemplo onde a função *wclean* pode evitar contratempos.

As edições necessárias neste arquivo são apenas para implementar as funções que já foram escritas e realizar algumas das operações do pós-processamento. As linhas 63 a 72 do arquivo original devem ser substituídas por:

```
// --- Pressure-velocity SIMPLE corrector
```

```
{
#include "UEqn.H"
#include "pEqn.H"
#include "TEqn.H"
}
fluid.correct();
scalarDict.set("tempo", ti+runTime.elapsedCpuTime());
sr=max(
       1.41421356237*mag(symm(fvc::grad(U))),
       dimensionedScalar ("VSMALL", dimless/dimTime, VSMALL));
tau=sr*fluid.nu();

yieldGama=pos(-sr+gama_);
yieldGama2=pos(-sr+gama_)
           *pos(U.mesh().C().component(1)
                -dimensionedScalar("one", dimLength, 1)
                )/ratio_;
```

A função *TEqn.H* foi adicionada, após a rotina *pEqn.H*, seguindo o fluxograma da Fig. 3.8. As outras linhas que foram adicionadas em seguida são para atualizar o valor do tempo no dicionário *scalarDict*, para calcular o campo de taxa da deformação e o campo da tensão cisalhante, nesta ordem. A seguir, a função *pos*, que retorna 1 quando o valor for positivo e o valor zero quando não for, foi utilizada para identificar as regiões onde a taxa de deformação é maior que o critério estabelecido. A definição de *yieldGama2* acrescenta um novo termo, que verifica também a posição do centroide de cada volume de controle. Por padrão, as listas do OpenFOAM® começam do zero, logo *component(0)* é a componente *x* da posição do centroide, *component(1)* é *y* e *component(2)* é a componente *z*, quando trabalhando em um sistema de coordenadas cartesiano. No caso da geometria do canal com uma expansão-contração (Seção 7.2), a cavidade é definida como toda a região onde o *component(1)*, ou seja *y*, da posição do centroide é maior do que 1.

Como já discutido, o OpenFOAM® estabelece uma estrutura com diferentes níveis de cálculo. Existe uma função especifica para o cálculo da taxa de deformação, e ela será utilizada na implementação dos modelos de viscosidade. Porém, ela não está definida no nível da rotina *nonNewtonianSimpleFoam.C*, e definir esta função nesse nível de cálculo não é necessariamente uma tarefa trivial. Dessa forma, a solução mais simples é recalcular a taxa de deformação, usando uma expressão quase idêntica à expressão da na linha 60 do arquivo *viscosityModel.C*, que por sua vez pode ser encontrado na pasta com seu próprio nome, na pasta *src\transportModels\incompressible\viscosityModels*, que é onde todas as funções pa-

drão de viscosidade aparente estão definidas também. As diferenças entre os dois códigos é que na expressão da rotina *nonNewtonianSimpleFoam.C*, o valor da raiz quadrada de dois já foi substituído, e que um novo arranjo é adicionado para evitar que o valor da taxa de deformação seja igual a zero: neste caso, a função *max* retorna o maior valor entre o valor da taxa de deformação calculado e o valor *VSMALL*, que significa "*very small*" e seu valor é de 10^{-300} quando trabalhando com precisão dupla, como é o padrão do OpenFOAM®. Dessa maneira, evita-se que a taxa de deformação seja igual a zero, o que poderia gerar problemas ao se calcular a viscosidade aparente, por exemplo, que pode ser definida como: $\eta = \frac{\tau}{\dot{\gamma}}$. Além desse valor, outras variáveis com valores de referência são definidas por padrão no OpenFOAM®, tais como a *SMALL* (10^{-15}) e a *GREAT*(10^{15}), por exemplo.

Porém, a variável *VSMALL* é uma grandeza escalar, e a taxa de deformação é uma grandeza escalar com dimensões, no caso $[1/s]$. Para evitar problemas, o trecho do código *dimensionedScalar ("VSMALL", dimless/dimTime, VSMALL)* declara uma variável escalar com dimensões, com valor igual ao da variável *VSMALL* e unidade igual a *dimless/dimTime*. A mesma estratégia é utilizada para se definir o campo *yieldGama2*, onde uma variável foi declarada com valor igual a um e dimensão igual a *dimLength*, que é a mesma unidade do termo *U.mesh().C().component(1)*, que é uma medida de comprimento.

Pasta Make

Por fim, os arquivos $files$ e $options$ devem ser editados. Estes arquivos auxiliam no processo de compilação do código, definindo quais arquivos devem ser compilados e quais as rotinas e bibliotecas que serão associadas a eles. O conteúdo do arquivo $files$ deve ser substituído por:

```
nonNewtonianSimpleFoam.C

EXE = $(FOAM_USER_APPBIN)/nonNewtonianSimpleFoam
```

Ou seja, a primeira linha do arquivo *files* define qual arquivo que será compilado, no caso, o *nonNewtonianSimpleFoam.C*, e a segunda linha define qual o executável que será criado, o *nonNewtonianSimpleFoam*, que será salvo na pasta *FOAM_USER_APPBIN*. O uso da pasta *FOAM_USER_APPBIN* indica que esta é uma rotina elaborada pelo usuário, ao contrário dos aplicativos padrão do OpenFOAM®, que são salvos como *FOAM_APPBIN*.

Já o arquivo *options* define as rotinas e bibliotecas que o executável precisa ter acesso. O conteúdo desse arquivo deve ser igual a:

```
EXE_INC = \
    -I$(LIB_SRC)/transportModels \
    -I$(LIB_SRC)/transportModels/incompressible\
/singlePhaseTransportModel \
    -I$(LIB_SRC)/transportModels/incompressible\
/viscosityModels/viscosityModel \
    -I$(LIB_SRC)/finiteVolume/lnInclude \
    -I$(LIB_SRC)/meshTools/lnInclude \
    -I$(LIB_SRC)/sampling/lnInclude \
    -L$(FOAM_USER_LIBBIN)

EXE_LIBS = \
    -lincompressibleTransportModels \
    -lfiniteVolume \
    -lmeshTools \
    -lfvOptions \
    -lsampling
```

A linha de comando *-L$(FOAM_USER_LIBBIN)* acrescenta a biblioteca do usuário às bibliotecas que o programa tem acesso. Isso facilita, por exemplo, o uso de funções de viscosidade aparente desenvolvidas pelos usuários nesta rotina. Porém, mesmo que o solver não seja compilado com acesso à essa biblioteca, é possível incluí-la através do arquivo *controlDict*, que será apresentado nas próximas seções. A forma com que esses dois arquivos devem ser redigidos é especialmente delicada, e até mesmo o uso da tecla *tab* para organizar a indentação pode gerar erros, por exemplo. Novamente, o comando *wclean* é uma ferramenta que pode ajudar a resolver diversos problemas, já que é possível que uma compilação mal sucedida possa causar erros nas compilações seguintes, ainda que o erro tenha sido corrigido.

wmake

Para compilar o programa, basta executar o comando *wmake* em uma janela de comando aberta na pasta *nonNewtonianSimpleFoam*. Se a última mensagem na janela de comando for algo como "*Error occurred with cv2pdb, have stripped binary as a workaround.*", o programa foi compilado com sucesso. Uma vez compilado o programa, ele só será recompilado após o comando *wclean* ou se o seu arquivo *.C* for editado.

4.3 Novas funções de viscosidade aparente

Nesta seção dois novos códigos serão redigidos e compilados: uma função para implementar o modelo SMD de viscosidade aparente, e outra função para o modelo de bi-viscosidade. Novamente, os seguintes comandos devem ser inseridos em uma janela de comando:

```
run
mkdir userViscosityModels
cd userViscosityModels
cp -r $FOAM_SRC/transportModels/incompressible\
/viscosityModels/powerLaw biviscosity
cd biviscosity/
mv powerLaw.H biviscosity.H
mv powerLaw.C biviscosity.C
sed -i s/powerLaw/biviscosity/g biviscosity.C
sed -i s/powerLaw/biviscosity/g biviscosity.H
cd ..
cp -r biviscosity SMD
cd SMD
mv biviscosity.H SMD.H
mv biviscosity.C SMD.C
sed -i s/biviscosity/SMD/g SMD.C
sed -i s/biviscosity/SMD/g SMD.H
```

Essa sequência de comandos tem por objetivo criar a pasta *userViscosityModels*, onde os modelos serão salvos. Dentro dessa pasta, uma pasta para cada modelo é criada, e dentro de cada pasta, uma cópia do modelo *powerLaw* é salva. Então, o nome "*powerLaw*" é substituído pelo nome do novo modelo.

biviscosity.C e biviscosity.H

As linhas 52 a 70 do arquivo original contém o equacionamento do modelo *powerLaw*. Por definição, todos os modelos retornam o valor da viscosidade aparente, e não o valor da tensão cisalhante, por exemplo. A expressão adotada pelo OpenFOAM® para o modelo *power-law* contém um valor máximo e um mínimo para a viscosidade aparente, o que justifica o tamanho do código. Para obter a expressão do modelo de bi-viscosidade expressa pela Eq.(2.19), a seguinte expressão deve substituir o conteúdo das linhas 52 a 70 do arquivo *biviscosity.C*:

```
{
```

```
tmp<volScalarField> sr(strainRate());

return
(
    min
    (
        eta0_,
        (tau0_*(1-etaP_/eta0_) + etaP_*sr())
        /(max
             (
             sr(),
             dimensionedScalar ("VSMALL", dimless/dimTime, VSMALL)
             )
         )
    )
);
}
```

Novamente, a função *max* é utilizada para se evitar um taxa de deformação nula. As edições seguintes são apenas para declarar as novas variáveis e para se obter o valor delas no arquivo *transportProperties*, que será discutido nos próximos capítulos. As linhas 83 a 88 do arquivo original devem ser substituídas por:

```
viscosityModel(name, viscosityProperties, U, phi),
biViscosityCoeffs_
(viscosityProperties.optionalSubDict(typeName + "Coeffs")),
eta0_("eta0", dimViscosity, biViscosityCoeffs_),
tau0_("tau0", dimViscosity/dimTime, biViscosityCoeffs_),
etaP_("etaP", dimViscosity, biViscosityCoeffs_),
```

E as linhas 115 a 118 também devem ser substituídas:

```
biViscosityCoeffs_.lookup("eta0") >> eta0_;
biViscosityCoeffs_.lookup("tau0") >> tau0_;
biViscosityCoeffs_.lookup("etaP") >> etaP_;
```

No arquivo *biviscosity.H*, apenas um trecho deve ser editado, substituindo o conteúdo das linhas 61 a 64 por:

```
dimensionedScalar eta0_;
dimensionedScalar tau0_;
dimensionedScalar etaP_;
```

SMD.C e SMD.H

O modificações para se obter a função *SMD* são semelhantes às edições que foram realizadas para se obter a função bi-viscosidade. O conteúdo das linhas 52 a 70 do arquivo *SMD.C* deve ser substituído pela equação Eq.(2.22), com o seguinte código:

```
dimensionedScalar etaInf=max
                         (
                          etaI_.value(),
                          VSMALL
                          )
                          *dimensionedScalar("one", dimViscosity, 1.0);

tmp<volScalarField> sr(
                 max
                 (
                  strainRate(),
                  dimensionedScalar ("VSMALL", dimless/dimTime, VSMALL)
                  )
                  *dimensionedScalar("one", dimTime, 1.0));

return (
       (1-exp(-eta0_.value()*sr()/tau0_.value()))
       *( tau0_/sr()*dimensionedScalar ("one", dimTime, 1)
          +k_*pow(sr(),n_.value()-1)
        )
       +etaInf
       );
```

Neste código, a variável *etaInf* representa η_∞, e a primeira linha do código evita que essa variável seja igual a zero. Sem esse artifício, a função pode apresentar problemas ao ser utilizada adotando $\eta_\infty = 0$. Como o valor de *VSMALL* é muito pequeno, essa modificação não compromete a validade do código. A função SMD utiliza uma função exponencial, e o OpenFOAM® aceita apenas variáveis da classe "*scalar*" como argumento para essa função. Mesmo que uma variável da classe "*dimensionedScalar*" tenha todas as suas unidades nulas, o OpenFOAM® não a reconhece como um argumento válido para a função *exp*. Dessa forma, sempre que uma grandeza escalar com dimensões é usada como expoente de algum termo, ela deve ser apresentada com a extensão *.value()*

Uma vez definidas as equações, basta atualizar os comandos para ler os arquivos adequados e declarar as novas variáveis. Ainda no arquivo *SMD.C*, as linhas 83 a 88 devem ser substituídas, lembrando que o número dessas linhas faz referência ao arquivo original, e não

ao que está no processo de edição:

```
SMDCoeffs_(viscosityProperties.optionalSubDict(typeName + "Coeffs")),
k_("k", dimViscosity, SMDCoeffs_),
n_("n", dimless, SMDCoeffs_),
eta0_("eta0", dimViscosity, SMDCoeffs_),
tau0_("tau0", dimViscosity/dimTime, SMDCoeffs_),
etaI_("etaI", dimViscosity, SMDCoeffs_),
```

E as linhas 115 a 118 também devem ser substituídas:

```
SMDCoeffs_.lookup("k") >> k_;
SMDCoeffs_.lookup("n") >> n_;
SMDCoeffs_.lookup("tau0") >> tau0_;
SMDCoeffs_.lookup("eta0") >> eta0_;
SMDCoeffs_.lookup("etaI") >> etaI_;
```

E novamente no arquivo *SMD.H*, conteúdo das linhas 61 a 64 deve ser substituído por:

```
dimensionedScalar k_;
dimensionedScalar n_;
dimensionedScalar eta0_;
dimensionedScalar tau0_;
dimensionedScalar etaI_;
```

Make

Para se compilar as duas funções, também é necessário uma pasta *Make*, com dois arquivos: o *files* e o *options*. Por praticidade, a pasta *Make* do *solver* da seção anterior pode ser copiada e colada na pasta *userViscosityModel*, junto com a pasta *biviscosity* e *SMD*. O arquivo *files* deve ter seu conteúdo original substituído por:

```
SMD/SMD.C
biViscosity/biViscosity.C

LIB = $(FOAM_USER_LIBBIN)/libUserViscosity
```

Dessa vez, as duas rotinas serão compiladas com o mesmo comando, criando assim a biblioteca "*libUserViscosity*". O arquivo *options* deve ter o seguinte conteúdo:

```
EXE_INC = \
-I$(LIB_SRC)/transportModels/incompressible/lnInclude/ \
```

```
-I$(LIB_SRC)/finiteVolume/lnInclude

LIB_LIBS = -lfiniteVolume -lincompressibleTransportModels
```

Uma vez que todos os arquivos estejam devidamente editados, basta abrir uma janela de comando na pasta *userViscosityModel* e executar o comando *wmake libso*. O comando é diferente do comando usado para se compilar o *solver* pois agora as rotinas devem compilar uma biblioteca, e não um executável.

CAPÍTULO V

PRÉ-PROCESSAMENTO E PÓS-PROCESSAMENTO

Neste capítulo as rotinas de pré e pós-processamento utilizadas neste trabalho são apresentadas e discutidas. Na fase de pré-processamento, as condições de contorno são impostas, a malha é gerada e as propriedades do escoamento são definidas. Além disso, os esquemas numéricos são escolhidos, assim como os fatores de relaxamento e os critérios de convergência. Os parâmetros de controle da simulação também são definidos, como o tempo inicial e final da simulação, frequência de registro do resultados, entre outros. Este capítulo busca não somente apresentar os parâmetros adotados, como também discutir a escolha de cada um deles, e como estes parâmetros podem ser otimizados. Mais informações sobre os arquivos de pré-processamento podem ser obtidas em Greenshields (2018), Guerrero (2018), Holzinger (2018) e Holzmann (2016).

Além do pré-processamento, neste trabalho busca-se realizar o maior número possível de etapas do pós-processamento no próprio OpenFOAM®. Alguns desses processos foram inclusive incorporados ao *solver*, permitindo assim a analise da convergência dos resultados. Cada tipo de analise pode demandar uma técnica de pós-processamento diferente, e é possível que existam diversas formas de se tratar os resultados de uma simulação. Assim, se faz necessário discutir as limitações e potenciais de cada método.

5.1 Pré-processamento

Na pasta base do OpenFOAM®é possível encontrar a pasta *tutorials*, onde uma série de exemplos de aplicações podem ser encontrados, organizados em função dos seus respectivos

solvers. Nessas aplicações, uma série de funções do OpenFOAM® são apresentadas, fornecendo assim um importante material de referência para quem começa a utilizar este *software* ou para quem busca entender como funciona uma função nova. Os arquivos necessários para cada simulação variam de *solver* para *solver* e assim, uma série de edições são necessárias para poder aplicar as rotinas desenvolvidas na Seção 4 em qualquer um desses tutoriais. Todos os arquivos necessários para uma simulação usando o *nonNewtonianSimpleFoam* estão presentes no Apêndice 10.3. Ainda assim, o tutorial *pitzDaily* do *solver simpleFoam*, que pode ser encontrado dentro da pasta *tutorials\incompressible*, foi usado como uma referência para se discutir a configuração de uma simulação no OpenFOAM®. Esta pasta deve ser copiada para um outro ambiente, como a pasta *run* apresentada na Seção 4. Dentro da pasta *pitzDaily* existem 3 outras pastas:

- *system*: é a pasta que contém todos os arquivos que definem os esquemas e parâmetros numéricos e de controle da simulação. Também contém os arquivos que definem as funções associadas à simulação, como as funções de pós-processamento e algumas funções que geram e manipulam a malha.
- *constant*: Nesta pasta são definidas as propriedades dos fluidos envolvidos na simulação. Além disso, quando a malha é gerada, ela é salva nesta pasta.
- 0: é a pasta de representa o tempo ou iteração zero, que será utilizada como ponto inicial da simulação. Ela contém os campos das propriedades analisadas na simulação, definindo assim as condições de contorno, por exemplo.

Diferentemente dos códigos apresentados na Seção 4, os arquivos aqui apresentados frequentemente são específicos de cada simulação, e os parâmetros adotados muitas vezes são escolhas de quem realiza a simulação. Assim, esta seção busca principalmente discutir as escolhas realizadas, ao invés de tentar definir os parâmetros “ótimos”. Além disso, diversas vezes uma condição de contorno pode ser definida de mais de uma forma, ou mais de uma função pode ser usada para gerar a malha, por exemplo.

5.1.1Pasta system

No tutorial *pitzDaily*, a pasta *system* contém originalmente 5 arquivos:

- *blockmeshDict*: é um dicionário com as instruções para a função *blockMesh* gerar a malha. Esta é uma das formas mais simples de se gerar uma malha no OpenFOAM®, e todas as malhas deste trabalho foram geradas com essa função.
- *fvSchemes*: é onde os esquemas numéricos são definidos, como os métodos de interpolação e de aproximação.
- *fvSolution*: é onde os *solvers* dos sistemas lineares e os parâmetros de convergência e de relaxação são definidos.
- *controlDict*: Este arquivo define os parâmetros de controle da simulação, como tempo inicial e final, frequência e formato que os resultados serão salvos, número de pastas de tempo que devem ser mantidos, entre outros parâmetros. Aqui também podem ser incluídas as novas bibliotecas e as funções de pós-processamento que serão realizadas durante a simulação.
- *streamlines*: O arquivo *streamlines* é uma função de pós-processamento. No caso, esta função calcula as linhas de fluxo de uma determinada propriedade.

Destes cinco arquivos, três são indispensáveis para a simulação: o *fvSchemes*, o *fvSolution* e o *controlDict*. É possível realizar uma simulação sem o *blockmeshDict*, caso a malha seja gerada por outra função, como as funções de conversão de malha, por exemplo, ou caso a malha já tenha sido gerada e copiada para o local apropriado. Porém, todas as malhas usadas neste trabalho foram geradas usando a função *blockMesh*, então o arquivo *blockmeshDict* também será discutido. Por fim, a função *streamlines* não foi usada em nenhuma simulação deste trabalho, e este arquivo pode ser suprimido.

blockmeshDict

O arquivo *blockMeshDict* contém as instruções para a malha ser gerada através do comando *blockMesh*. Os arquivos *blockMeshDict* utilizados neste trabalho começam com o comando *convertToMeters*. Este comando aplica um fator de escala nas dimensões da geometria, o que facilita a conversão entre diferentes unidades de medida, por isso seu nome. O fator de escala foi definido igual a 1, já que as dimensões das geometrias já serão inseridas com as devidas unidades.

Em seguida, é possível se definir variáveis para facilitar a edição da malha. Estas variáveis não são variáveis do *solver*, apenas da função *blockMesh*. Para acessar o valor da variável, é necessário acrescentar um "$" antes dela sempre que for utilizada. Além de definir variáveis, também é possível realizar cálculos, usando um código na forma:

```
a #calc "$b - $c";
```

onde a variável *a* é definida como a diferença de duas outras variáveis, *b* e *c*, por exemplo. Assim, é possível gerar malhas complexas em função de um conjunto de variáveis.

A seguir são definidos os vértices da geometria. Por definição, a malha é definida em três dimensões. Para facilitar as etapas seguintes, é uma boa prática indicar após cada vértice a sua posição na lista em um comentário. O primeiro vértice ocupa a posição zero da lista.

Uma vez definidos os vértices, o comando *blocks* é então usado para gerar hexaedros (*hex*). O primeiro argumento desta função é um conjunto oito vértices que formam o hexaedro. Cada vértice é representado pela sua posição na lista de vértices. A sequência em que os vértices aparecem nesta função é importante: para cada hexaedro, um sistema local de coordenadas é formado, onde o eixo *z* é definido como a direção do primeiro para o quinto vértices definidos neste primeiro argumento da função. Para um observador na direção oposta a esse eixo, os vértices devem ser ordenados no sentido anti-horário para cada um dos dois planos normais ao eixo *z*. Após definir como os vértices estão organizados, o número de elementos em cada direção é definido. No caso de uma geometria bidimensional com 100 elementos em cada direção, o vetor com o número de volumes é igual a (100 100 1), por exemplo. Para obter outras formas além do hexaedro, é possível colapsar arestas usando o mesmo vértice mais de uma vez ao declarar os vértices nesta função.

Em seguida, a função *simpleGrading* define como esses elementos serão organizados. O vetor (1 1 1) significa que todos os elementos possuem o mesmo tamanho. Já o vetor (2 1 1) significa que o último elemento do eixo *x* tem o dobro do tamanho do primeiro. Para malhas mais complexas, o fator de expansão de cada direção pode ser subdividido em intervalos, onde cada intervalo é caracterizado por três parâmetros, nesta ordem: a porcentagem do comprimento total que aquele intervalo representa, a porcentagem de elementos presentes no intervalo, e por fim a razão entre o comprimento do último elemento do intervalo divido pelo comprimento do primeiro elemento do intervalo.

Por definição, os vértices são unidos por linhas retas, mas é possível impor geometrias diferentes usando o comando *edges*. Cada formato tem suas próprias características, e no presente trabalho todas as geometrias foram feitas sem utilizar deste recurso.

Por fim, as faces externas são agrupadas para se definir as condições de contorno: para cada conjunto é definido um nome, um tipo base e as faces que o compõe. Para definir as faces, seus quatro vértices devem ser declarados, com a condição que eles sejam vizinhos, mas a ordem de declaração não importa. O tipo base pode definir diretamente qual a condição de contorno é aplicada, ou apenas restringir quais condições podem ser aplicadas. O tipo base mais genérico é o *patch*, que não acrescenta nenhuma característica, e o tipo *empty* é utilizado para transformar geometrias tridimensionais em bidimensionais, por exemplo. O *defaultPatch* agrupa todas as faces externas da geometria que não foram definidas em nenhum grupo, o que pode facilitar a configuração de geometrias onde várias faces pertencem a um mesmo grupo.

fvSchemes

O arquivo *fvSchemes* define os métodos numéricos de aproximação e interpolação adotados na simulação. É possível se determinar um método específico para cada termo, como por exemplo um método exclusivo para o termo *grad(p)*, ou um método *default*, que será empregado para todos os termos que não forem explicitamente definidos. Os esquemas numéricos adotados exercem grande influência na eficiência e a convergência da simulação. Além disso, eles apresentam uma forte interação entre eles e com outros fatores numéricos, como os fatores de relaxamento. Dessa forma, a configuração ótima varia de acordo com as propriedades do fluido, a geometria e malhas adotadas, entre outros. O processo de otimização dos parâmetros numéricos é longo e trabalhoso, em especial porque um parâmetro pode influenciar a eficiência dos outros, e a simples mudança de um determinado parâmetro pode levar a uma configuração ótima completamente diferente. A configuração aqui apresentada é capaz de resolver todas as simulações usadas neste trabalho, mas existe muito espaço para melhorias.

Uma forma de se descobrir quais métodos o OpenFOAM® disponibiliza é inserir propositalmente um erro no nome de algum método. Quando o *solver* for inicializado, o processo será abortado e uma mensagem com os métodos que estão disponíveis para aquela função será exibida. O OpenFOAM® tem por padrão uma série de métodos numéricos implementados

para cada função, o que aumenta ainda mais as possibilidades de otimização.

Métodos como "*Gauss linear*" são na realidade a combinação do uso do teorema de Gauss, como descrito na seção 3.1.2, com um esquema de interpolação linear. Ou seja, o método *Gauss linear* é o método das diferenças centradas apresentado na seção 3.1.5. Outra característica importante é que cada função tem seu próprio método de interpolação, e o campo *interpolationSchemes* define o método de interpolação usado em outras funções, como no cálculo de *phi*, por exemplo.

Todas as funções utilizadas neste trabalho são de segunda ordem, com a exceção do *interpolationSchemes*, que usa um esquema de interpolação cúbica. Quanto maior a ordem do método, mais eficiente se torna o sistema, ainda que aumente a chance do sistema divergir. Além disso, toda interpolação pode gerar erros, em especial quando a propriedade interpolada apresenta um comportamento de ordem muito superior à ordem do método de interpolação. Além disso, esse erro também sofre influência do tamanho da malha: quanto mais fina a malha, menor será o erro de interpolação. Como já discutido, a viscosidade em fluidos viscoplásticos apresenta um forte comportamento não linear. Isso aumenta o custo computacional e aumenta os erros gerados por interpolações na zona de transição entre regiões escoadas e regiões aparentemente não escoadas. Na Fig. 5.1, o erro gerado por essa diferença de ordem pode ser observado: como a viscosidade varia várias ordens de grandeza em uma pequena distância, os métodos de interpolação lineares apresentam erros significativos, que geram pertubações no campo de pressão, por exemplo. Como os parâmetros numéricos e a malha são adequadas, essa pertubação é rapidamente absorvida pelo sistema. Em malhas mais grosseiras, essas pertubações podem chegar se propagar e interagir entre si, comprometendo o resultado. Uma das consequências disso é que os testes de qualidade de malha devem ser realizados com malhas semelhantes, respeitando as características do método escolhido para se avaliar a independência. Ao se comparar malhas muito diferentes, é possível que o erro provocado por essa pertubação seja tão grande que comprometa qualquer resultado obtido com a malha mais grosseira.

Para se resolver esse problema, duas abordagens podem ser aplicadas: aumentar a ordem dos métodos de interpolação e refinar a malha, em especial nas regiões de transição entre zonas escoadas e zonas aparentemente não escoadas. O refinamento de malha aumenta o custo computacional de maneira significativa. O aumentar da ordem dos método de

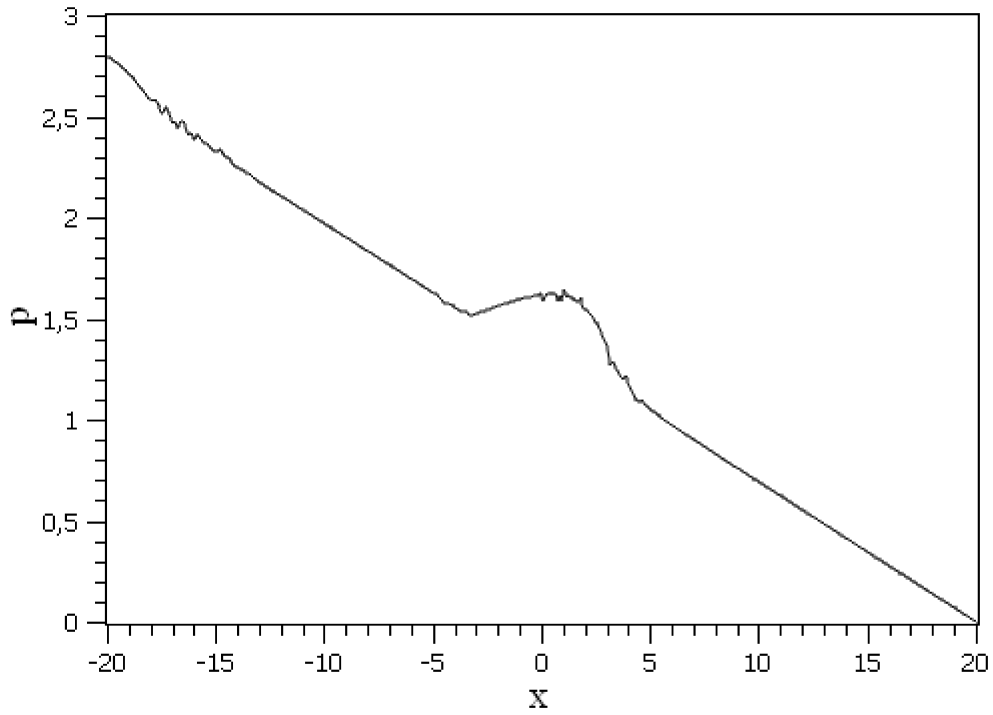


Figura 5.1 – Oscilações no campo de pressão provocadas pelos erros de interpolação. Valores de pressão obtidas sobre a linha de simetria do canal com expansão e contração.

interpolação pode deixar o sistema mais instável. Assim, a solução encontrada neste trabalho foi aumentar apenas a ordem do *interpolationSchemes*. Apenas essa mudança já foi suficiente para conseguir a convergência e precisão desejadas. A simples alteração dos outros métodos de interpolação para ordens superiores geralmente comprometem a estabilidade das simulações apresentadas neste trabalho. Para se obter configurações com outros métodos de interpolação de maior ordem é necessário otimizar o sistema como um todo, alterando diversos parâmetros da simulação.

fvSolution

Neste arquivo são definidos os parâmetros dos métodos de solução, tanto dos sistemas lineares quando do sistema como um todo. Para cada equação, um *solver* linear deve ser definido. Os *solvers* lineares que o OpenFOAM® oferece por padrão são descritos em Greenshields (2018). Além do *solver*, é necessário definir seus critérios de convergência. Um critério muito rígido consome muito tempo para ser atingido, e um critério muito fraco pode demandar muitas iterações para que o sistema como um todo convirja. Existem três possíveis

critérios:

- *tolerance* : O solver linear considera que o sistema convergiu quando os resíduos normalizados daquela equação são mais baixos que o valor da *tolerance*.
- *relTol*: Este critério considera que o sistema convergiu quando a razão entre o resíduo atual e o resíduo inicial for mais baixa que o valor definido pelo *relTol*.
- *maxIter*: esse critério considera que o sistema convergiu quando o número de iterações atinge um determinado número. Este parâmetro é opcional, e por padrão é igual a 1000.

De um modo geral, o sistema linear que mais consome tempo é a solução do campo da pressão. Porém, dadas as condições de escoamentos de fluidos viscoplásticos, é possível definir um critério mais frouxo para este campo, uma vez que é necessário um número elevado de iterações devido ao caráter não linear que este comportamento acrescenta ao sistema. Assim, é possível economizar tempo na solução do campo de pressão.

Por fim, também é neste arquivo que os fatores de relaxação são definidos. Os fatores de relaxação para simulações de fluidos viscoplásticos podem ser sensivelmente diferentes dos utilizados para simular fluidos newtonianos. Devido ao importante caráter não linear desses fluidos, os fatores de relaxação tendem a ser particularmente baixos para manter a convergência do sistema, em especial o parâmetro associado à equação do balanço de quantidade de movimento linear. Porém, as condições do escoamento podem permitir que o campo de pressão apresente fatores de relaxação maiores, inclusive com valores maiores que a unidade para algumas configurações. Essa característica pode aumentar substancialmente a eficiência do código, ainda que os fatores de relaxação ótimos variem de simulação para simulação. Os parâmetros apresentados neste trabalho são parâmetros médios, e eles devem ser alterados para conseguir reproduzir todos os resultados aqui apresentados.

controlDict

O *controlDict* define uma série de parâmetros de controle da simulação. Como já discutido, em simulações de escoamentos em regime permanente, o tempo passa a representar o número de interações, e por isso o parâmetro *deltaT* deve ser definido igual a um. Pelo mesmo motivo, o *endTime* deve ser suficientemente grande para que o sistema convirja. O parâmetro *startFrom* define de qual tempo a simulação vai começar: *latestTime* faz com que

o OpenFOAM® verifique qual a pasta de maior tempo disponível, e continue a simulação a partir dela. Isso permite que a simulação seja pausada e retomada. Além disso, o *purgeWrite* define quantas pastas de tempo devem ser mantidas pela simulação. Simulações de fluidos viscoplásticos podem ser especialmente longas, e manter muitos resultados pode ocupar um espaço considerável de armazenamento de disco. Por fim, *runTimeModifiable* define se é possível alterar parâmetros da simulação enquanto ela está ocorrendo. Essa propriedade pode ser especialmente interessante em simulações com dificuldade para convergir, permitindo que parâmetros como os fatores de relaxação sejam ajustados.

Também é neste arquivo que as funções de pós-processamento são incluídas. Assim é possível monitorar os resultados, como será discutido nas próximas seções. A ordem que as funções são declaradas é a mesma ordem que estas funções serão executadas. Além das funções, é possível adicionar bibliotecas, no formato: "`libs ( "libUserViscosity" );` ". Dessa forma, é possível utilizar bibliotecas criadas pelo usuário em *solvers* padrão do OpenFOAM®.

5.1.2 Pasta 0

A pasta 0 contém os campos do tempo de simulação zero, ou na iteração zero, se for uma simulação de regime permanente. Dessa forma, essa pasta é uma pasta de tempo, ainda que o seu significado seja ligeiramente diferente para simulações de regime permanente. Para iniciar a simulação, é necessário que a presença de ao menos uma pasta de tempo dentro da pasta de simulação, não necessariamente a pasta 0, de acordo com o parâmetro *startFrom* do *controlDict*.

A pasta 0 contém as condições inicias da simulação e as condições de contorno. Ao iniciar um *solver*, ele irá verificar e ler todas os campos que estão definidos no seu arquivo *createFields.H*. Logo, para realizar uma simulação usando o *nonNewtoninanSimpleFoam*, além dos arquivos *U, T* e *p*, é necessário que os arquivos *sr, tau, yieldGama, yieldGama2* e o dicionário *scalarDict* estejam presentes nesta pasta. No decorrer da simulação, outros campos serão acrescentados nas pastas de tempo, mas eles não são necessários para iniciar uma simulação. Cada um desses arquivos representa um campo de uma variável, e nele devem estar definidas as condições iniciais do interior da malha e a condições de contorno. As condições de contorno para todos os grupos definidos no *blockMeshDict* devem ser especificadas para cada um dos campos. Como citado na Seção 5.1.1, alguns tipo base de contorno já são

condição suficiente para o OpenFOAM® reconhecer as condições de contorno, e assim não é necessário definir essas condições novamente, como ocorre no grupo de superfícies de tipo base *empty* e *symmetryPlane*, por exemplo.

Cada campo é definido pela suas dimensões, pelo campo interno da malha e pelas condições de contorno. As dimensões são definidas por um vetor, onde cada coluna representa uma dimensão, como tempo, massa, comprimento, entre outros. A sequência das dimensões pode ser encontrada em Greenshields (2018). O comando *internalField* define o valor inicial do campo interno da malha.

Cada tipo base de contorno da malha permite que determinadas condições de contorno sejam impostas. Para os campos **u**, *T* e *p*, definidos nos arquivos *U, T* e *p*, respectivamente, as duas principais condições de contorno impostas são a *fixedValue* para valores impostos e a *zeroGradient* para gradiente igual a zero daquela variável. A condição *fixedValue* precisa que o valor a ser fixado também seja definido. Outras condições equivalentes a essas também poderiam ser impostas, como a condição *noSlip*, que é uma condição própria do tipo base *wall*, e impõe que a velocidade seja nula no contorno.

Por fim, é necessário definir os campos *sr, tau, yieldGama* e *yieldGama2*. Todas as suas condições de contorno receberam a condição *calculated*. Esta condição de contorno interpola os valores internos para as faces externas. Normalmente, essa condição é aplicada em campos gerados pelo próprio OpenFOAM®, e só está sendo utilizada aqui pois estes campos são usados apenas como pós-processamento, e não fazem parte da solução do sistema.

mapFields

Como alternativa à estratégia de se iniciar as simulações com um campo constante, duas opções são possíveis: usar os resultados de uma configuração com características próximas ou realizar uma simulação em uma malha mais grosseira para estimar as propriedades na malha adequada. Para a primeira solução, basta substituir os resultados nas pastas adequadas — é necessário que as malhas sejam idênticas. Para a segunda opção, o comando "*mapFields*" deve ser empregado, como explicado em Greenshields (2018). O uso desse comando pode ser particularmente útil para simulações de configuração próxima em diferentes malhas, como nos testes de qualidade de malha. Porém, como já discutido, alguns de método de interpolação apresentam uma eficiência limitada em simulações de fluidos viscoplásticos.

5.1.3 Pasta constant

No tutorial *pitzDaily*, a pasta *constant* originalmente contém dois arquivos: *transportProperties* e o *turbulenceProperties*. Em *turbulenceProperties* é definido o modelo de turbulência adotado; em *transportProperties* o modelo de viscosidade é escolhido. No caso do tutorial *pitzDaily*, o modelo de viscosidade adotado é o newtoniano com a viscosidade igual a 10^{-5}. Como o presente trabalho visa realizar simulações em regime laminar, o arquivo *turbulenceProperties* não é necessário.

Da mesma forma que os campos declarados na rotina *createFields.H* são definidos na pasta *0*, diversas constantes declaradas nesta rotina devem ser definidas no arquivo *transportProperties*, são elas: *alpha*, *beta*, *rho*, *ratio* e o critério para se definir as zonas escoadas e aparentemente não escoadas (*gamaRef*). As outras duas constantes definidas na rotina *createFields.H* são a gravidade, que foi imposta no código do *solver*, e o tempo inicial da simulação *ti*, que é definido em outro dicionário, o *scalarDict*. Além das variáveis definidas no *createFields.H*, existem aquelas definidas na função da viscosidade aparente. Cada função tem seus parâmetros de entrada, e devem ser definidos em uma subdivisão do dicionário *transportProperties*. Para correlacionar o modelo com a subdivisão correta do dicionário, o nome da subdivisão deve ser composto pelo nome do modelo mais a partícula "*Coeffs*", como pode ser observado no arquivo correspondente do Apêndice 10.3.

Entre os parâmetros a serem definidos diretamente neste dicionário, apenas o *ratio* pertece à classe "*scalar*", e os outros parâmetros são grandezas escalares com dimensões ("*dimensionedScalar*"'). Definir essas duas classes é um processo semelhante ao utilizado para declarar essas mesmas classes no código do *solver*: para o escalar, é necessário apenas escrever o nome do parâmetro seguido do seu valor. Já as grandezas escalares com dimensões são definidas na mesma ordem que a gravidade foi definida na linha 116 do código *createFields.H* do Apêndicie 10.1.5: o primeiro argumento é a variável seguida do nome associado a ela, que normalmente é a mesma palavra, as dimensões do parâmetro e, por fim, seu valor. Os parâmetros do modelo de viscosidade não precisam ter seu nome repetido, mas é necessário definir suas unidades e o seu valor.

5.1.4 allrun

Para otimizar o processo de simulação, é comum definir em um arquivo a ordem de comandos que deve ser executada. Dessa forma, ao invés de chamar uma função por vez, é possível executar toda a cadeia com um comando só. Por padrão, esse arquivo é chamado de *allrun*. Vários tutoriais tem seus próprios *allruns*, normalmente com comandos mais elaborados que aqueles que são apresentados aqui. Para se obter um exemplo de *allrun*, basta salvar o seguinte código em um arquivo de texto:

```
blockMesh
nonNewtonianSimpleFoam
postProcess -func "graph"
```

Depois de salvo, basta executar esse arquivo em uma janela de comando aberta na pasta da simulação que as funções ali inseridas serão executadas nessa ordem. No caso, esse *allrun* gera a malha com o comando *blockMesh*, faz a simulação com o *solver nonNewtonianSimpleFoam*, e executa a função *graph* de pós-processamento, que será discutida na próxima seção. O uso de um arquivo *allrun* é altamente recomendado, não apenas por otimizar a simulação, mas também por padronizar as funções executadas e permitir que uma sequência de simulações sejam realizadas de maneira sistemática.

5.2 Pós-processamento

Existem duas formas principais de se realizar o pós-processamento no OpenFOAM® :

1. Usando um comando específico na janela de comando em uma pasta de simulação. A função selecionada será executada usando todas as pastas de tempo disponíveis. O comando normalmente segue o seguinte formato:

   postProcess -func "graph"

2. Incluindo a função na solução do escoamento. A função será executada na frequência especificada junto com os outros parâmetros da função. Para incluir a função na solução, basta acrescentar a função no arquivo *controlDict*, segundo o seguinte modelo:

```
functions
    {
    #includeFunc  residuals
    }
```

A principal diferença entre esses dois métodos é a frequência com que a função é executada: no primeiro, a função é executada de forma pontual, enquanto no segundo método ela é executada em uma frequência pré-determinada, o que permite ao usuário acompanhar a evolução dos parâmetros estudados. Em ambos os casos, a função será executada adotando as especificações determinadas em um arquivo com o nome da própria função localizado dentro da pasta *system*, ou usando as configurações padrão daquela função, quando possível.

As funções de pós-processamento em geral podem retornar três tipos de resposta:

1. Quando a função retorna um escalar ou um vetor, um arquivo é criado com uma lista de valores obtidos para cada uma das pastas de tempo disponíveis. Este arquivo é salvo na pasta *postProcessing*, que é organizada por função e por um tempo de referência: se a função for executada diretamente na janela de comando, o tempo de referência é a última pasta de tempo; se a função for incluída no arquivo *controlDict*, é o tempo que a função foi inicializada.

2. Quando a função retorna um gráfico ou um plano, uma pasta para cada tempo disponível é criada dentro da pasta *postProcessing*. Incluir funções desse tipo na simulação pode gerar um grande número de pastas.

3. Quando a função retorna um campo, este campo é salvo nas pastas de tempo da simulação junto com os outros campos definidos pelo *solver*.

Quando a função retorna um valor escalar ou vetor, pode ser interessante acompanhar a evolução desse parâmetro com o desenvolvimento da simulação. A forma mais fácil de se fazer isso é usando a função *foamMonitor*, que é basicamente um conjunto de comandos para o *software gnuplot*. Este arquivo pode ser encontrado na pasta *bin* do OpenFOAM®. O comando para usar essa função é o seguinte:

```
foamMonitor -l postProcessing/residuals/0/residuals.dat
```

Neste caso, a função *foamMonitor* foi utilizada para plotar gráficos da função *residuals*, que registra o resíduo normalizado da simulação. Neste exemplo, a função *foamMonitor* foi executada de uma janela de comando aberta na pasta da simulação, daí a necessidade de se especificar o caminho até o arquivo *residuals.dat*. Neste formato, sempre que a simulação for retomada, é necessário corrigir a pasta de tempo, que neste caso é a pasta 0. Para evitar isso, é possível executar a função *foamMonitor* em uma janela de comando aberta diretamente na pasta de tempo dentro da pasta *postProcessing*, o que dispensa a necessidade de se especificar o caminho até o arquivo *residuals.dat*. Essa função pode ser executada em paralelo com a simulação, mas não é recomendável colocar a simulação em segundo plano para poder executar a função *foamMonitor* na mesma janela de comando. Sugere-se abrir uma segunda janela de comando para poder executar as duas funções.

Dentro da pasta *etc\caseDicts\postProcessing* é possível encontrar alguns *templates* de arquivos com os parâmetros de cada função de pós-processamento. Além dos *templates*, também é possível identificar as configurações padrão de cada função, e até modificá-las, se for o caso. As funções utilizadas neste trabalho são:

- *grad*: a função *grad* retorna um campo com o gradiente de uma propriedade. Neste trabalho, essa função calculará o gradiente de temperatura que será utilizado para calcular o valor do número de Nusselt.
- *residuals*: como já citado, a função *residuals* retorna o valor do resíduo normalizado. Como a função retorna uma lista de valores em função do número de iterações, esta função combinada com a *foamMonitor* facilita a análise da convergência da simulação.
- *patchIntegrate*: esta função retorna a integral em uma superfície. Neste trabalho, ela será usada para calcular o valor do número de Nusselt médio.
- *cellAvarege*: retorna o valor médio de um campo. Essa função será usada junto com o campo *yieldGama2* para calcular a eficiência de deslocamento. Como a eficiência de deslocamento é definida apenas para uma região do escoamento, a constante *ratio* converte o valor médio em todo o domínio para o valor médio apenas na região de interesse.
- *graph*: Esta função gera perfis dos parâmetros determinados. De todas as funções aqui apresentadas, é a única função usada através de um comando próprio e não incluída no arquivo *controlDict*.

Existem uma série de outras funções de pós-processamento, e cada função tem seus parâmetros próprios. A função *graph*, por exemplo, tem vários esquemas de interpolação, vários formatos de resultados e várias formas de se determinar os pontos que vão formar o perfil. Além disso, também é possível definir a frequência com que cada função será executada. Como o número de iterações em simulações de fluidos viscoplásticos pode ser elevado, normalmente as funções de pós-processamento deste trabalho são executadas apenas uma vez a cada mil iterações.

CAPÍTULO VI

VERIFICAÇÃO

Este capítulo busca verificar se as rotinas elaboradas na Seção 4 são capazes de reproduzir alguns resultados encontrados na literatura. A verificação do *solver nonNewtonianSimpleFoam* foi realizada comparando os resultados desta rotina com os resultados de dois trabalhos: O primeiro trabalho usado como referência foi Turan et al. (2011), que analisa um escoamento bidimensional de convecção natural de um fluido *power-law* provocado pela diferença de temperatura imposta sobre as duas paredes laterais de um recipiente quadrado de lado L. Este trabalho discute o efeito do número de Rayleigh, do número de Prandtl e do índice de comportamento n sobre o número de Nusselt médio, o perfil de velocidade vertical e de temperatura avaliados na metade da altura da geometria.

O segundo trabalho adotado como referência foi Turan, Chakraborty e Poole (2010), que adota a mesma geometria do trabalho anterior, mas modela o fluido como um fluido de Bingham. Devido a mudança da natureza do fluido, o número de Bingham substitui o índice de comportamento como uma das propriedades que são avaliadas no estudo. Para simular o comportamento do fluido de Bingham, o modelo de bi-viscosidade é usado impondo $\eta_0/\eta_\infty = 10^4$.

Dessa forma, o modelo mecânico adotado neste capítulo é composto pelas equações de balanço apresentadas na Seção 4.2.2 e as equações de viscosidade aparente, que para o caso de escoamentos de fluidos *power-law* é a Eq.(2.13) , e para fluidos de Bingham é a Eq.(2.19).

Uma metodologia parecida foi adotada por Favero et al. (2010), Habla et al. (2011), Higuera, Lara e Losada (2013): cada um desses trabalhos busca validar e verificar rotinas do

OpenFOAM® propostas pelos autores, comparando seus resultados com resultados numéricos, experimentais e analíticos encontrados na literatura. Favero et al. (2010) buscam validar dois *solvers* desenvolvidos pelos autores: um *solver* específico para resolver escoamentos internos e outro para escoamentos com superfície livre de fluidos viscoelásticos. Habla et al. (2011) também apresentam o processo de verificação de um *solver* capaz de resolver escoamentos viscoelásticos multifásicos. Já Higuera, Lara e Losada (2013) buscam validar uma condição de contorno para modelar a geração de ondas e absorção ativa em OpenFOAM®.

6.1 Grupos adimensionais de interesse

Para facilitar a comparação dos resultados, as definições dos grupos adimensionais adotadas são exatamente as mesmas adotadas por Turan et al. (2011) e Turan, Chakraborty e Poole (2010). Dessa forma, as expressões para o número de Rayleigh e para o número de Prandtl são:

$$Ra = \frac{\rho|\mathbf{g}|\beta\Delta T L_c^3}{\eta_{nom}\alpha} \tag{6.1}$$

$$Pr = \frac{\eta_{nom}}{\alpha\rho} \tag{6.2}$$

onde L_c é o comprimento característico, que nesta geometria é o comprimento do lado da cavidade (*L*), ΔT é a diferença de temperatura imposta entre as paredes laterais do recipiente, e η_{nom} é a viscosidade nominal, que deve ser definida para cada tipo de fluido. Para fluidos de Bingham, $\eta_{nom} = \eta_\infty$, e para fluidos *power-law*, η_{nom} é igual a:

$$\eta_{nom} = K\left(\frac{\alpha}{L_c^2}\right)^{n-1} \tag{6.3}$$

O número de Nusselt (*Nu*) é igual a:

$$Nu = \frac{L_c}{|T_{wall} - T_{ref}|}\left.\frac{\partial T}{\partial x}\right|_{wf} \tag{6.4}$$

onde o subíndice *wf* indica que aquela expressão deve ser avaliada no contato do fluido com a parede. T_{wall} é a temperatura da parede, e T_{ref} é a temperatura de referência: quando o número de Nusselt é avaliado sobre a parede mais quente, a temperatura de referência deve

ser a temperatura da parede mais fria, por exemplo.

O número de Nusselt médio $\overline{Nu}$ é definido por:

$$\overline{Nu} = \frac{1}{S_{Nu}} \int_{S_{Nu}} Nu \, dS \tag{6.5}$$

onde S_{Nu} é a superfície analisada. Nesta seção, S_{Nu} é a parede lateral de maior temperatura.

O número de Bingham *Bn* é definido pela expressão:

$$Bn = \frac{\tau_0}{\eta_{nom}} \sqrt{\frac{L_c}{g\beta\Delta T}} \tag{6.6}$$

A altura adimensional (y^*), o comprimento adimensional (**x***), a velocidade vertical adimensional (u_y^*) e a temperatura adimensional (θ) são definidas como:

$$\mathbf{x}^* = \frac{\mathbf{x}}{L_c} \qquad \mathbf{u}^* = \frac{\mathbf{u} L_c}{\alpha} \qquad \theta = \frac{T - T_C}{T_H - T_C} \tag{6.7}$$

onde u_y é a velocidade vertical, e T_H e T_C são as temperaturas impostas nas paredes laterais, de tal forma que $T_H > T_C$. T_H é a temperatura da parede quente e T_C é a temperatura da parede fria.

6.2 Geometria e condições de contorno

A Fig. 6.1 ilustra esquematicamente a geometria adotada para as simulações de verificação. As paredes verticais tem a temperatura imposta, criando assim um gradiente de temperatura. As temperaturas impostas são $\theta_H = 1$ e $\theta_C = 0$. Já as paredes horizontais são consideradas adiabáticas. Para o campo de velocidades, todas as paredes apresentam a condição de não deslizamento.

A malha adotada contém 100 elementos em cada direção, e um total de 10^4 elementos. A malha é mais refinada perto das paredes, e mais grosseira no interior da geometria. Todo elemento é retangular e o comprimento adimensional mínimo de elemento é igual a $\frac{\delta_{min,cell}}{L} = 3,6960 \times 10^{-3}$, como indicado na Tab. 6.1. A razão entre dois comprimentos vizinhos é igual a $r_e = 1,0361$, no sentido dos contornos para o interior da geometria. Isso significa que o comprimento em uma dada direção (horizontal ou vertical) de um elemento qualquer é igual a 1,0361 o tamanho do seu vizinho mais próximo das paredes naquela direção, e é igual a

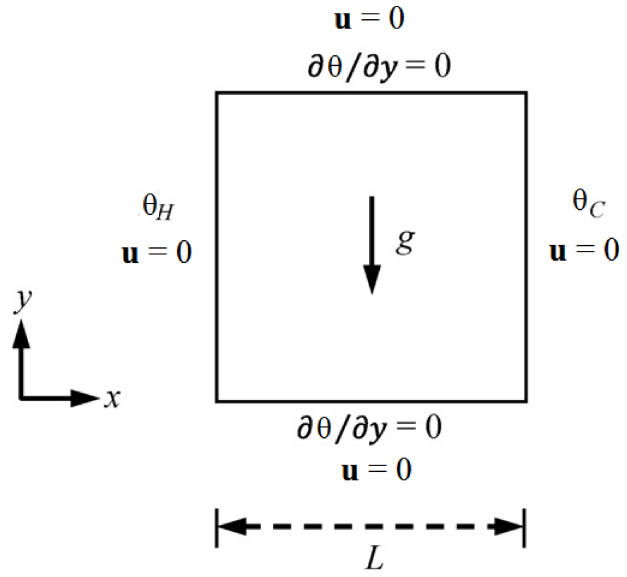


Figura 6.1 – Representação esquemática da geometria adotada para a verificação das rotinas. Fonte: adaptado de Turan, Chakraborty e Poole (2010).

$1,0361^{-1}$ o tamanho do seu vizinho voltado para o interior da geometria.

Fig. 6.2 dá um exemplo de como essa lógica de refinamento funciona, com uma malha com 20x20 elementos e com r_e igual 1,5137.

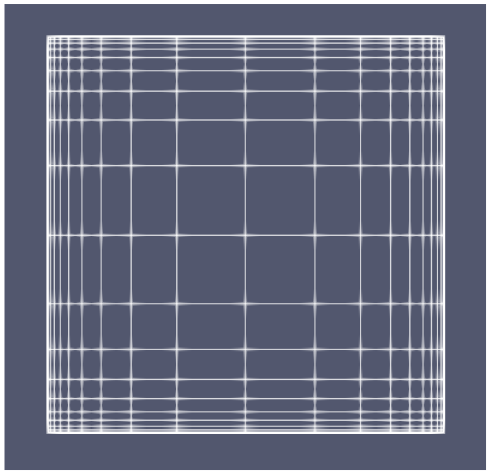


Figura 6.2 – Malha com 20x20 elementos para ilustrar a lógica de refinamento.

O processo de análise da qualidade de malha foi realizado de acordo com o procedimento proposto por Celik (2008), adotando um *theoretical order of accuracy* $a = 2$. Este método é baseado na teoria de extrapolação de resultados de Richardson. O valor extrapolado ϕ_{ext} é definido por

$$\phi_{ext} = \frac{r_{21}^{a} * \phi_1 - \phi_2}{r_{21}^{a} - 1} \tag{6.8}$$

onde os subíndices indicam qual malha foi utilizada para se obter aquele valor, sendo 1 a malha mais refinada, 2 a malha intermediária e 3 a malha mais grosseira. ϕ é o parâmetro analisado, r é a razão de um comprimento representativo de duas malhas diferentes e o erro em relação ao valor extrapolado é igual a

$$e_{ext,i} = \left| \frac{\phi_{ext} - \phi_i}{\phi_{ext}} \right| \tag{6.9}$$

As incertezas numéricas foram estimadas usando os valores extrapolados e os valores obtidos em cada uma das malhas. Para fluidos *power-law*, três condições foram analisadas: $n = 1$; $n = 0,6$ e $n = 1,8$, com valores de número de Rayleigh e número de Prandtl fixados em $Ra = 10^6$ e $Pr = 100$. Já para os fluidos de Bingham, duas condições foram analisadas: $Bn = 0$ e $Bn = 0,5$, com $Ra = 10^4$ e $Pr = 7$. Os critérios adotados para avaliar as malhas foram a velocidade vertical adimensional máxima para $y^* = 0,5$ e o número de Nusselt médio. As malhas utilizadas na análise da qualidade de malha estão definidas na Tab. 6.1. Os resultados da análise para fluidos *power-law* são mostrados na Tab. 6.2, e os resultados para fluidos de Bingham estão na Tab. 6.3. A malha escolhida apresenta um erro máximo de $e_{ext} = 4,45\%$ em relação ao valor extrapolado, e sete das dez avaliações realizadas apresentam $e_{ext} < 1\%$, oferecendo assim uma boa relação entre eficiência e acurácia. O parâmetro CGI_{fine}^{21} é calculado através da seguinte expressão:

$$CGI_{fine}^{21} = \frac{1,25 e_{21}}{r_{21}^{a} - 1} \tag{6.10}$$

onde

$$e_{21} = \left| \frac{\phi_1 - \phi_2}{\phi_1} \right| \tag{6.11}$$

Tabela 6.1 – Parâmetros das malhas adotadas na análise da qualidade de malha para o processo de verificação.

Malha	3	2	1
Número de elementos	50 × 50	75 × 75	100 × 100
$\frac{\delta_{min,cell}}{L}$	$7,39 \times 10^{-3}$	5,54$\times 10^{-3}$	3,67$\times 10^{-3}$
r_e	1,0747	1,0446	1,0361

Tabela 6.2 – Análise da qualidade de malha para fluidos *power-law*.

		$\overline{Nu}$			$u^*_{y,max}$		
	malha	3	2	1	3	2	1
$n = 0,6$	ϕ	26,3	25,6	26,8	1496	1597	1575
	ϕ_{ext}	27,82			1549		
	$e_{ext}(\%)$	5,57	8,12	4,45	3,42	3,07	1,68
	$CGI^{21}_{fine}(\%)$	5,82			$2,06$		
$n = 1$	ϕ	7,60	7,62	7,59	166	166	168
	ϕ_{ext}	7,55			169		
	$e_{ext}(\%)$	0,68	0,99	0,54	2,06	1,79	0,98
	$CGI^{21}_{fine}(\%)$	0,67			$1,24$		
$n = 1,8$	ϕ	2,22	2,22	2,22	20,7	20,7	20,8
	ϕ_{ext}	2,22			20,8		
	$e_{ext}(\%)$	0,04	0,10	0,05	0,43	0,27	0,15
	$CGI^{21}_{fine}(\%)$	0,67			$0,19$		

Tabela 6.3 – Análise da qualidade de malha para fluidos de Bingham.

		$\overline{Nu}$			$u^*_{y,max}$		
	malha	3	2	1	3	2	1
$Bn = 0$	ϕ	4,72	4,74	4,72	73,6	73,2	73,6
	ϕ_{ext}	4,71			$73,9$		
	$e_{ext}(\%)$	0,33	0,61	0,33	0,53	0,96	0,53
	$CGI^{21}_{fine}(\%)$	0,66			$0,41$		
$Bn = 0,5$	ϕ	3,85	3,85	3,87	45,6	44,4	46,2
	ϕ_{ext}	3,90			48,3		
	$e_{ext}(\%)$	1,04	1,25	0,69	5,45	7,98	4,37
	$CGI^{21}_{fine}(\%)$	0,07			$5,46$		

Comparando a malha adotada neste trabalho com a malha adotada por Turan, Chakraborty e Poole (2010) e Turan et al. (2011), é possível observar que todas as malhas seguem a mesma lógica de refinamento, mas a malha adotada por Turan et al. (2011) contém apenas 80x80 elementos, enquanto a malha adotada por Turan, Chakraborty e Poole (2010) e no presente trabalho contém 100x100 elementos. Outra diferença é que Turan et al. (2011) adota condições de contorno semelhantes às adotadas por Turan, Chakraborty e Poole (2010), mas

a posição das paredes à alta e à baixa temperaturas são invertidas. Essa modificação na condição de contorno não compromete os resultados obtidos, uma vez que uma geometria é a imagem espelhada da outra.

6.3 Pré-processamento

6.3.1 Malha

O arquivo *blockMeshDict* usado para gerar a malha descrita nessa seção é o arquivo apresentado no Apêndice 10.3.4. Para facilitar a definição dos vértices, uma variável foi declarada com o valor do lado da cavidade. A maior dificuldade para se gerar essa malha é reproduzir a lógica de refinamento: primeiro porque existem duas tendências em cada direção, e segundo porque o OpenFOAM® usa uma lógica diferente para a razão de expansão, adotando a relação entre o primeiro e o último elemento, e não a relação entre elementos vizinhos. Para resolver o segundo problema, basta calcular $r_e^{n_e/2-1}$ onde n_e é o número de elementos em cada direção. O valor obtido é a razão de expansão quando os elementos estão crescendo. Para quando eles estão diminuindo, basta inverter esse número.

Para separar a malha em intervalos, é necessário empregar a lógica apresentada na Seção 5.1.1. Como a divisão dos elementos é simétrica, a razão de expansão tanto na direção *x* quanto na direção *y* deve ser substituída por ((0.5 0.5 5.6737) (0.5 0.5 0.17625)), que significa que 50% do comprimento naquela direção contém 50% dos elementos desta direção e a razão entre o primeiro e o último elemento neste intervalo é de 5,6737. O segundo intervalo contem a mesma proporção do comprimento e dos elementos, mas é o intervalo onde os elementos diminuem de tamanho ao avançar na direção do eixo analisado, por isso que a razão de expansão é menor que um.

Após editar o arquivo *blockMeshDict*, é necessário executar o comanto *blockMesh* em uma janela de comando aberta no pasta da simulação. Este comando usará as instruções do arquivo *blockMeshDict* e irá gerar uma malha na pasta *constant* da simulação.

6.3.2 Condições iniciais e de contorno

As condições iniciais e de contorno são definidas nos arquivos da pasta *0*. A condição de contorno para a velocidade em todas as paredes é a condição de velocidade imposta

(*fixedValue*) e igual a um vetor nulo. Para a pressão, todas as paredes tem gradiente nulo (*zeroGradient*). Para o campo de temperatura, as paredes horizontais também apresentam a condição de gradiente nulo, e as paredes verticais apresentam a condição de valor fixo, igual a zero ou um, como mostra a Fig. 6.1.

Essas simulações podem ter dificuldade de convergência ao iniciar com todos os campos internos nulos, ainda que isso não comprometa o resultado final. Para se resolver esse problema, basta inserir um valor não nulo em alguma propriedade no interior da malha, como uma componente da velocidade, por exemplo.

6.3.3 Propriedades do fluido

O arquivo *transportProperties* deve conter uma série de propriedades do fluido, em especial qual o modelo de viscosidade aparente adotado e quais suas propriedades. Nestas simulações, dois modelos diferentes foram aplicados: O modelo *power-law* e o modelo de bi-viscosidade. O modelo *power-law* é uma das funções padrão do pacote OpenFOAM®, e o modelo de bi-viscosidade é a rotina apresentada na Seção 4.3. Cada uma dessas duas funções tem suas *inputs* características: o modelo biviscosidade é definido com três parâmetros (τ_0, η_0 e η_∞) e o modelo *power-law* é definido por quatro parâmetros ($K, n, \eta_{min}, \eta_{max}$). Como nestas simulações não é intenção trabalhar com patamares de viscosidade aparente na função *power-law*, é necessário definir um η_{min} suficientemente pequeno e um η_{max} suficientemente grande para que eles não influenciem na simulação.

6.4 Pós-processamento

Os resultados utilizados para realizar a verificação dos códigos foram o número de Nusselt médio, o perfil de temperatura e de velocidade vertical. Para os dois perfis, a função de pós-processamento *graph* foi utilizada. Os pontos que definem esse perfil são os pontos onde a linha $y^* = 0,5$ corta as faces dos elementos da malha. Assim, ao se definir os parâmetros desse perfil, foi definido "*type face;*". Como o método de volume finitos calcula sempre o valor das propriedades do escoamento no centroide, foi escolhido um esquema de interpolação (*interpolationScheme*) igual a "*cellPoint*", o que significa que o OpenFOAM® adotou um esquema de interpolação linear para obter os valores das propriedades fora do centroide da geometria.

Outros esquemas de interpolação poderiam ser adotados, como considerar o valor da propriedade constante no interior de cada elemento e com valor igual ao valor da propriedade calculada no centroide, por exemplo.

A função *graph* pode ser usada para obter perfis de várias propriedades de uma só vez, tanto de propriedades escalares quanto de propriedades vetoriais, por exemplo. Por fim, é recomendável que a função *graph* seja executada através de um comando próprio quando a simulação já estiver concluída, pois esta função gera uma pasta de arquivos cada vez que é executada, o que pode gerar um número expressivo de arquivos quando ela é incluída durante a simulação.

Para o número de Nusselt médio, duas funções foram utilizadas: a primeira foi a função *grad* para se obter o campo do gradiente da temperatura. Em seguida, a função *patchIntegrate* foi utilizada para se realizar a integral sobre a superfície definida no campo *name*, dentro do dicionário dessa função. Como a função *grad* gera um campo na pasta do tempo atual, basta definir "*fields (grad(T));*" para se calcular a integral do gradiente da velocidade sobre a superfície analisada. Ao incluir essas duas funções na simulação, é possível monitorar o valor do número de Nusselt médio no decorrer da simulação, usando a função *foamMonitor* com o devido endereço dos arquivos.

6.5 Critérios de comparação

Para realizar a verificação das rotinas, é importante estabelecer os critérios de comparação entre os resultados obtidos e os resultados adotados como referência. Como já citado, Turan, Chakraborty e Poole (2010) e Turan et al. (2011) expõem uma série de resultados em forma de gráficos, como os perfis de temperatura e de velocidade vertical, por exemplo. Para se realizar comparações quantitativas, um software de análise de imagem foi utilizado para converter os gráficos em valores numéricos. Em seguida, três parâmetros estatísticos foram utilizados para se avaliar quão próximos são os resultados obtidos com a rotina elaborada neste trabalho com os resultados da literatura. Os três parâmetros adotados foram: a soma dos quadrados dos resíduos (*SSE*), a raiz do erro quadrático médio (*RSME*) e o coeficiente de determinação (r^2). O *SSE* é uma medida de discrepância entre o modelo analisado, que neste caso é a rotina a ser validada, e os valores de referência, que neste caso são os valores obtidos na literatura. Quanto menor o *SSE*, melhor o modelo consegue aproximar os valores

de referência. O *RMSE* é um erro médio, o que auxilia a interpretação dos resultados ao inserir o conceito de quantos pontos estão sendo analisados. Ainda assim, tanto o *SSE* quanto o *RMSE* não são normalizados, o que limita a sua interpretação para comparar curvas diferentes. Já o r^2 é uma medida de quão bem o modelo consegue replicar os valores de referência, sendo que $r^2 = 1$ significa o ajuste perfeito da curva. As expressões que definem esses três parâmetros são:

$$SSE = \sum_{i=1}^{s} (\phi_{lit,i} - \widehat{\phi}_i)^2 \tag{6.12}$$

$$r^2 = 1 - \frac{SSE}{\sum_{i=1}^{s} (\phi_{lit,i} - \overline{\phi_{lit,i}})^2} \tag{6.13}$$

$$RMSE = \sqrt{\frac{SSE}{s}} \tag{6.14}$$

onde *s* é o número de pontos utilizados na comparação dos resultados, $\phi_{lit,i}$ é o valor da propriedade ϕ no ponto i encontrado na literatura, $\widehat{\phi}_i$ é o valor obtido pela rotina proposta no ponto i e $\overline{\phi_{lit,i}}$ é o valor médio dos valores usados como referência. Neste trabalho, ao avaliar os perfis de temperatura e de velocidade vertical, todos os vértices da malha sobre o eixo analisado foram utilizados na comparação.

6.6 Resultados

6.6.1 Fluidos power-law

Essa seção tem por objetivo comparar os resultados obtidos com a rotina proposta neste trabalho com os resultados apresentados por Turan et al. (2011), usando a função power-law para modelar a viscosidade. Para analisar os resultados dos perfis de velocidade e de temperatura, 27 simulações foram realizadas: todas as combinações possíveis entre três índices de comportamento (*n*=0,6; *n*=1 e *n*=1,8), três números de Rayleigh (*Ra*=10^4; *Ra*=10^5 e *Ra*=10^6) e três números de Pradtl (*Pr*=10^2; *Pr*=10^3 e *Pr*=10^4). A Fig. 6.3 compara qualitativamente algumas das curvas avaliadas. A Tab. 6.4 apresenta os valores máximos de *RMSD* e *SSE* e os valores mínimos de r^2 obtidos nessas 27 simulações.

Como pode ser observado na Fig. 6.3, vários perfis de velocidade vertical se sobrepõem,

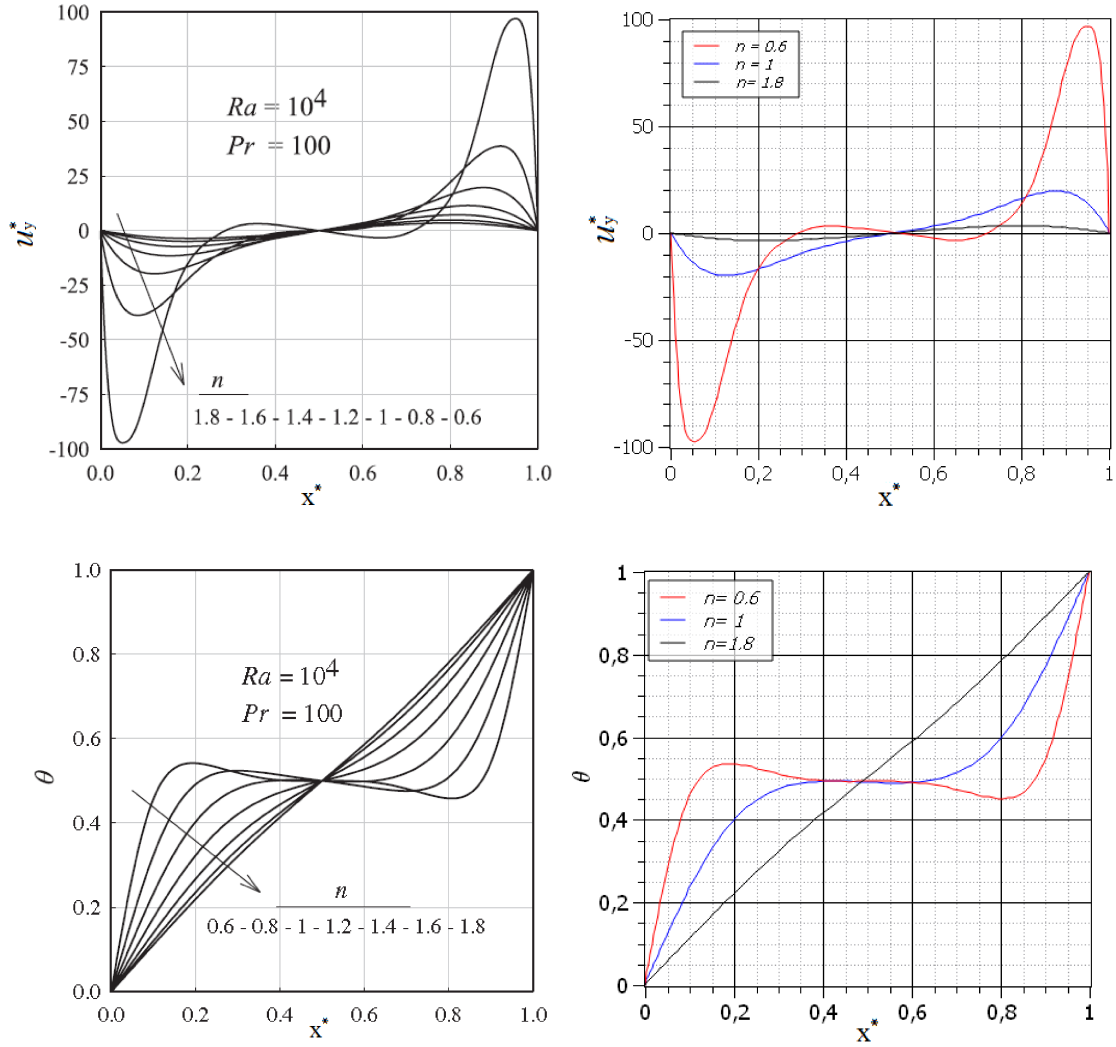


Figura 6.3 – Perfis de temperatura adimensional e de velocidade vertical adimensional para $y^* = 0,5$, *Pr*= 100; *Ra*=10^4 e diferentes valores de *n*. A coluna da esquerda são gráficos obtidos por Turan et al. (2011), e a coluna da direita são os resultados obtidos neste trabalho com o software OpenFOAM®.

Tabela 6.4 – Valor mínimo de r^2 e valores máximos de *RMSD* e de *SSE* para os perfis de temperatura adimensional e de velocidade vertical adimensional, usando os resultados de Turan et al. (2011) como referência.

	u_y^*	θ
r^2_{min}	0,9526	0,9603
$RMSD_{max}$	167,8212	0,0223
SSE_{max}	$2,8445 \times 10^6$	0,0503

e a magnitude de cada perfil pode variar alguma ordens de grandeza. Essas duas características diminuem a precisão do software de análise de imagem, introduzindo erros na comparação. Ainda assim, o r^2_{min} continua ligeiramente abaixo da unidade, e a maioria dos r^2 obtidos estão acima de 0,99, o que indica que a rotina do OpenFOAM® consegue reproduzir os resul-

tados com precisão. Como a magnitude de cada perfil varia consideravelmente, a comparação de *RMSD* e *SSE* fica comprometida, já que essas grandezas não são normalizadas e assim um erro de menor relevância em uma curva de maior magnitude pode gerar *RMSD* e *SSE* mais altos que um erro mais severo em uma curva de magnitude menor.

Os perfis de temperatura apresentam um r^2_{min} ainda maior. Como a temperatura adimensional varia sempre entre zero e um, o software de análise de imagem consegue capturar as curvas com maior precisão, embora os perfis continuem se sobrepondo. Também por causa da magnitude constante dos perfis, os baixos valores de $RMSD_{max}$ e SSE_{max} corroboram com a conclusão que os resultados obtidos com o OpenFOAM® reproduzem com fidelidade os resultados obtidos por Turan et al. (2011).

A Fig. 6.4 apresenta a comparação para os resultados dos valores do número de Nusselt médio. Turan et al. (2011) propôs uma relação para se obter o valor do número de Nusselt médio em função de *Pr*, *Ra* e *n*. A Tab. 6.5 mostra a comparação entre os valores obtidos usando as rotinas de OpenFOAM® e essa correlação. Foram realizadas 108 simulações para se obter esses resultados.

Tabela 6.5 – r^2, *RMSD* e *SSE* para os valores do número de Nusselt médio, comparados com a correlação proposta por Turan et al. (2011).

	Número de Nusselt médio
r^2	0,9978
$RMSD$	0,3586
SSE	13,8877

Os valores de r^2, *RMSD* e *SSE* confirmam que as duas metodologias apresentam resultados com boa concordância. A correlação proposta por Turan et al. (2011) tem alguns coeficientes determinados por um processo de ajuste de curvas, e assim a correlação é válida para os seguintes intervalos: $10^4 \leq Ra \leq 10^6; 10 \leq Pr \leq 10^5$ e $0,6 \leq n \leq 1,8$. Dessa forma, era esperado que os valores simulados apresentassem diferenças em relação aos valores da correlação, em especial para configurações próximas aos limites da validade da correlação.

6.6.2 Fluidos de Bingham

Nesta seção, os resultados obtidos neste trabalho com as rotinas do OpenFOAM® foram comparados com os resultados de Turan, Chakraborty e Poole (2010). A metodologia aplicada é praticamente a mesma da Seção 6.6.1: a única diferença é que não foi realizada a compa-

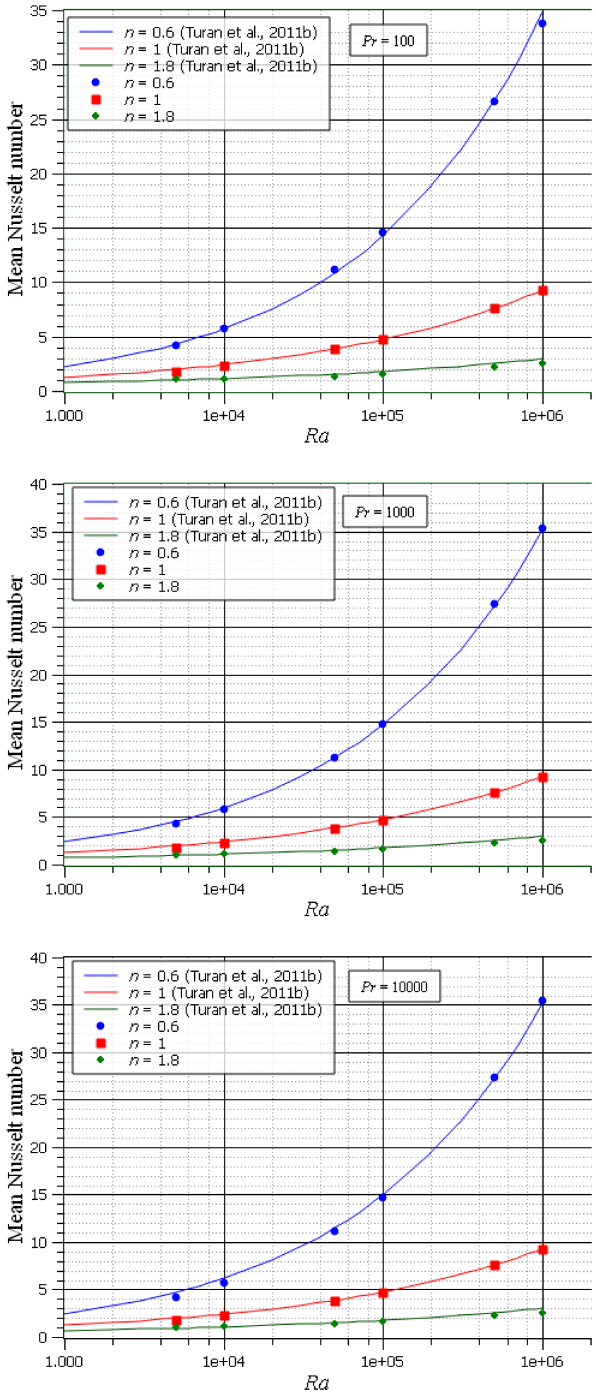


Figura 6.4 – Número de Nusselt médio em função de *Ra*, para *Pr* = 100; *Pr* = 1000; *Pr* = 10000. As linhas contínuas representam a correlação proposta por Turan et al. (2011), e os valores pontuais são os resultados obtidos com as rotinas do OpenFOAM®.

ração entre a correlação proposta por Turan, Chakraborty e Poole (2010) e os valores obtidos de número de Nusselt médio. A correlação proposta por Turan, Chakraborty e Poole (2010) contém três coeficientes que não tiveram seus valores definidos pelos autores. Para obter esses valores, um processo de ajuste de curva deveria ser feito. Porém, tal metodologia poderia

induzir um bom ajuste de resultados, o que comprometeria a qualidade do processo de verificação. Dessa forma, apenas foram analisados os resultados para os perfis de temperatura e de velocidade vertical.

Novamente, o software de análise de imagem apresentou dificuldades ao converter em valores numéricos perfis que se sobrepunham e perfis com magnitudes muito diferentes no mesmo gráfico. Como pode ser observado na Fig. 6.5, essas características se tornam críticas em gráficos de perfis de velocidade vertical com diferentes *Bn*, chegando ao ponto onde o erro introduzido pelo software de análise de imagem tornar a comparação impraticável.

A Fig. 6.5 mostra os perfis de velocidade vertical e de temperatura para *Bn*=0,5, *Pr*=7 e um intervalo de *Ra* e a Fig. 6.6 apresenta os perfis de temperatura e velocidade vertical para *Ra*=10000, *Pr*=7 e um intervalo de *Bn*. Por fim, a Tab. 6.6 mostra o valor mínimo de r^2 e os valores máximos de *RMSD* e *SSE* para essas simulações. Oito simulações foram realizadas para se obter esses resultados.

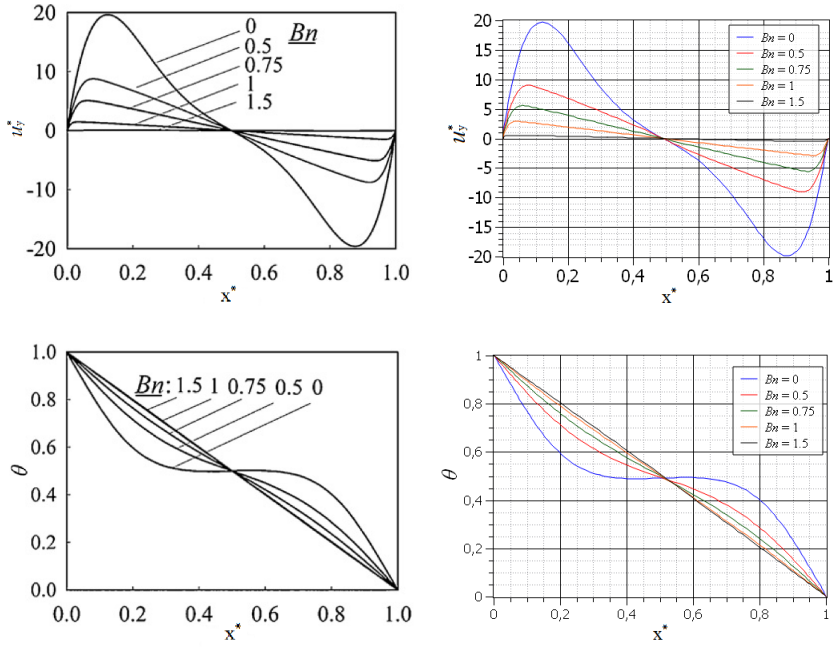


Figura 6.5 – Perfis de temperatura adimensional e velocidade vertical adimensional em $y^* = 0,5$ para *Ra*=10000, *Pr*= 7 e um intervalo de *Bn*. Os gráficos da coluna da esquerda foram obtidos por Turan, Chakraborty e Poole (2010), e os gráficos da coluna da direta foram obtidos com a rotina *nonNewtonianSimpleFoam*.

Todos os parâmetros apresentados na Tab. 6.6 indicam boa correlação entre os resultados obtidos usando o OpenFOAM® e os resutados apresentados por Turan, Chakraborty e Poole (2010). Os valores de r^2 são particularmente altos para os perfis de temperatura adi-

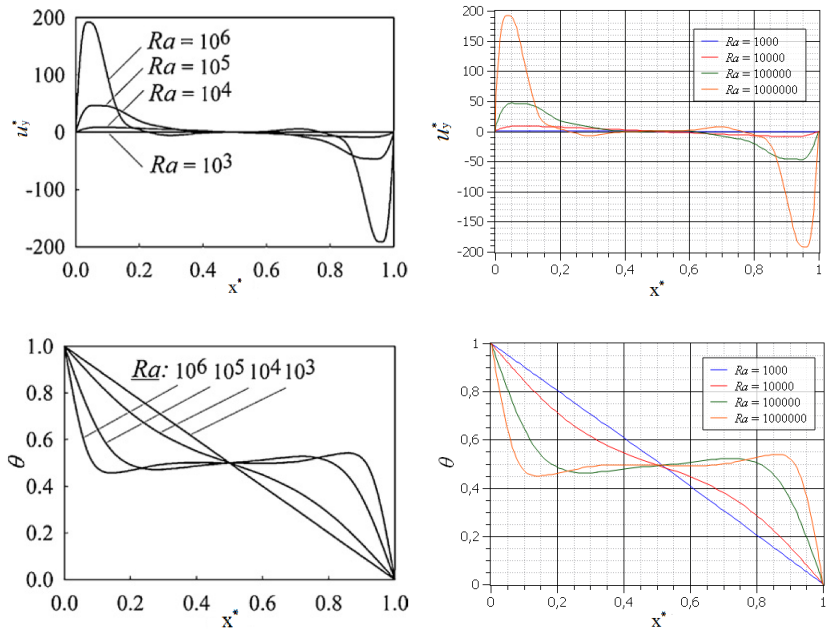


Figura 6.6 – Perfis de temperatura adimensional e velocidade vertical adimensional em $y^* = 0,5$ para *Bn*=0,5, *Pr*= 7 e um intervalo de *Ra*. Os gráficos da coluna da esquerda foram obtidos por Turan, Chakraborty e Poole (2010), e os gráficos da coluna da direta foram obtidos com a rotina *nonNewtonianSimpleFoam*.

Tabela 6.6 – Valor de r^2 mínimo e *RMSD* e *SSE* máximos obtidos para na comparação dos perfis de temperatura adimensional e velocidade vertical adimensional para fluidos de Bingham, usandos os resultados de Turan, Chakraborty e Poole (2010) como referência.

	u_y^*	θ
r^2_{min}	0,9605	0,9978
$RMSD_{max}$	5,3469	0,0077
SSE_{max}	$2,8875 \times 10^3$	0,006

mensional, o que pode ser explicado pelo fato de que os perfis para diferentes valores de Bn se sobrepõem menos que os outros gráficos. Assim, o software de análise de imagem consegue capturar os valores com maior precisão, e o erro introduzido por ele é minimizado.

6.7 Conclusão do processo de verificação

Nesta seção, várias simulações foram realizadas usando as rotinas apresentadas na Seção 4, e seus resultados foram comparados com os resultados de Turan, Chakraborty e Poole (2010) e Turan et al. (2011). Tanto comparações numéricas quanto comparações qualitativas foram realizadas para temperatura adimensional e velocidade vertical adimensional, tanto para fluidos *power-law* e quanto para fluidos de Bingham. Nas simulações de fluidos *power-*

law também foi comparado o valor do número de Nusselt médio. Todas as comparações mostraram uma alta correlação entre os valores obtidos, mesmo em comparações onde já era esperado uma diferença entre os resultados, seja pelo erro inserido pelo uso do software de análise de imagem, seja por usar como referência valores obtidos com uma correlação oriunda de um processo de ajuste de curvas.

Os parâmetros estatísticos adotados para a comparação numérica (r^2, *RMSD* e *SSE*) reafirmam a alta correlação entre os resultados, em especial o r^2, que ficou sempre acima de 0,95, mesmo onde o software de análise de imagem não conseguia capturar os resultados com grande precisão. Os resultados obtidos para esses parâmetros mostraram forte influência da qualidade da gráfico usado como referência: sempre que o gráfico apresentava menos áreas de sobreposição de curvas e as magnitudes das curvas nele representadas eram adequadas à sua escala, o parâmetro r^2 apresentava uma melhora sensível, o que indica que boa parte do erro identificado tem sua origem no software de análise de imagem, e não na rotina avaliada.

CAPÍTULO VII

APLICAÇÃO

Nesta seção, a rotina *nonNewtonianSimpleFoam* foi empregada para resolver um escoamento em um canal planar com uma expansão seguida de uma contração. A expansão-contração cria uma cavidade acima do canal, e um gradiente de temperatura foi imposto entre as paredes dessa cavidade e a temperatura inicial do fluido. A viscosidade do fluido foi modelada segundo a função SMD, que junto com as equações apresentadas na Seção 4.2.2 compõem o modelo mecânico do sistema. Para este estudo de caso, a convecção natural não será analisada, e o coeficiente de expansão β foi definido como zero para todas as simulações.

Foram analisados os efeitos do número de Reynolds, do *plastic number* e do índice de comportamento sobre diversos parâmetros do escoamento, tais como o número de Nusselt médio, a eficiência de deslocamento e o *head loss*. A influência de cada parâmetro foi analisada individualmente, partindo de uma configuração padrão de grupos adimensionais.

7.1 Grupos adimensionais de interesse

Os grupos adimensionais que caracterizam o escoamento estudado nesta seção são: o número de Reynolds (*Re*), o *plastic number* (*Pl*), o índice de comportamento (n), o *jump number* (*J*), o número de Prandtl (*Pr*) e o platô de viscosidade adimensional para altas taxas de deformação (η^*_∞). Além destes grupos adimensionais, o número de Nusselt médio ($\overline{Nu}$), a eficiência de deslocamento (Φ_{de}) e o *head loss* (Hl) são os parâmetros adimensionais adotados para se avaliar a influência do *Re*, *Pl* e n.

As expressões adotadas para *Re*, *Pr* e *Pl* nestes capítulo são as expressões propostas

por Thompson e Soares (2016), na seguinte forma:

$$Re = \frac{\rho {V_c}^2}{\tau_0 + K(\frac{V_c}{L_c})^n + \eta_\infty(\frac{V_c}{L_c})} \tag{7.1}$$

$$\frac{1}{Pr} = \frac{K(\frac{V_c}{L_c})^n + \eta_\infty(\frac{V_c}{L_c})}{\tau_0 + K(\frac{V_c}{L_c})^n + \eta_\infty(\frac{V_c}{L_c})} \frac{\rho\alpha}{K(\frac{V_c}{L_c})^{n-1} + \eta_\infty} \tag{7.2}$$

$$Pl = \frac{\tau_0}{\tau_0 + K(\frac{V_c}{L_c})^n + \eta_\infty(\frac{V_c}{L_c})} \tag{7.3}$$

onde V_c é a velocidade característica e L_c é o comprimento característico. Neste trabalho, V_c é a velocidade de entrada do fluido, e L_c é igual à altura do canal (*H1*).

O grupo adimensional *J* foi proposto por de Souza Mendes et al. (2007b), e ele proporciona uma medida relativa da variação da taxa de deformação em torno de $\tau = \tau_0$. A expressão que define *J* é:

$$J = \frac{\eta_0\dot{\gamma}_1}{\tau_0} - 1 \tag{7.4}$$

onde

$$\dot{\gamma}_1 = \left(\frac{\tau_0}{K}\right)^{\frac{1}{n}} \tag{7.5}$$

O grupo adimensional η^*_∞ evita que a viscosidade aparente tenda a zero para altas taxas de deformação quando $n < 1$. Ele é definido da seguinte forma:

$$\eta^*_\infty = \frac{\eta_\infty\dot{\gamma}_1}{\tau_0} \tag{7.6}$$

O parâmetro *head loss* (Hl) é uma medida adimensional da influência da cavidade sobre a queda de pressão do escoamento, definido como:

$$Hl = \frac{\Delta p_s - \Delta p_c}{\rho(\dot{\gamma}_1 L_c)^2} \tag{7.7}$$

onde Δp_c é a diferença de pressão entre um ponto a montante e um ponto a jusante da cavidade, onde o escoamento está completamente desenvolvido. Já Δp_s é a queda de pressão entre dois pontos de um canal simples, de forma que a distância entre esses pontos seja a mesma que a distância entre os pontos utilizados para determinar Δp_c. Para se calcular Δp_s, um método numérico foi aplicado para resolver a equação de balanço de quantidade de movimento linear com as condições de escoamento desenvolvido: gradiente de pressão

constante em qualquer direção, e gradiente de qualquer outra propriedade igual a zero na direção do escoamento.

A eficiência de deslocamento Φ_{de} é a razão entre a área escoada dentro da cavidade e a área total da cavidade, que pode ser escrita da seguinte maneira, em função da área aparentemente não escoada:

$$\Phi_{de} = 1 - \frac{A_{c,0}}{A_c} \tag{7.8}$$

onde $A_{c,0}$ é a área aparentemente não escoada dentro da cavidade, e A_c é a superfície total da cavidade. Para alguns tipos de escoamentos, como escoamentos de produtos da indústria alimentícia, por exemplo, a taxa de renovação do fluido é um parâmetro importante, e assim as zonas aparentemente não escoadas estacionárias podem ser particularmente danosas, criando zonas de incrustação que comprometem essa taxa de renovação. Dessa forma, também é interessante definir uma adaptação do parâmetro de eficiência de deslocamento que compute apenas zonas aparentemente não escoadas estacionárias, assim o parâmetro eficiência de deslocamento não estacionária Φ_{sta} é definido:

$$\Phi_{sta} = 1 - \frac{A_{c,sta}}{A_c} \tag{7.9}$$

onde $A_{c,sta}$ é a área aparentemente não escoada estacionária dentro da cavidade. Neste trabalho, todas as zonas aparentemente não escoadas ligadas à uma parede foram consideradas como estacionárias.

O número de Nusselt médio ($\overline{Nu}$) foi calculado pelo mesmo método apresentado na Seção 6.1 e na Seção 6.4, considerando a superfície S_{Nu} como sendo toda a parede interna da expansão-contração.

O vetor posição adimensional (**x***), a velocidade adimensional (**u***), a pressão adimensional (p^*), a temperatura adimensional (θ), a taxa de deformação adimensional ($\dot{\gamma}^*$), a tensão cisalhante adimensional (τ^*) e a viscosidade aparente adimensional (η^*) são definidas pelas seguintes expressões:

$$\begin{aligned} &\mathbf{x}^* = \frac{\mathbf{x}}{L_c} \qquad \mathbf{u}^* = \frac{\mathbf{u}}{\dot{\gamma}_1 L_c} \qquad p^* = \frac{p}{\tau_0} \\ &\theta = \frac{T - T_C}{T_H - T_C} \qquad \dot{\gamma}^* = \frac{\dot{\gamma}}{\dot{\gamma}_1} \qquad \tau^* = \frac{\tau}{\tau_0} \qquad \eta^* = \frac{\eta \dot{\gamma}_1}{\tau_0} \end{aligned} \tag{7.10}$$

O parâmetro para identificar zonas aparentemente não escoadas será $\dot{\gamma}_{ref}$, que pode ser obtido resolvendo a seguinte expressão:

$$\dot{\gamma}_{ref}\eta(\dot{\gamma}_{ref}) = \tau_0 \tag{7.11}$$

Como já citado, cada um dos três parâmetros analisados foi variado individualmente, o que gera a necessidade de se estabelecer um conjunto padrão de adimensionais que é empregado como referência para a análises. Esse conjunto padrão é o seguinte: *Re=25; Pl=0,4; Pr=14,02; J=*10^4*; n=0,5;* $\eta^*_{\rm inf}$*=0,01*. O único caso onde mais de um grupo adimensional é alterado é quando $n \geq 1$, onde o parâmetro $\eta^*_{\rm inf}$ assume o valor nulo. Essa mudança ocorre pois neste intervalo de valores de n, $\eta^*_{\rm inf}$ perde seu significado físico, como pode ser observado na Fig. 2.9.

7.2 Geometria e condições de contorno

A geometria adotada é um canal planar com uma expansão seguida de uma contração, representada na Fig. 7.1. A razão entre a altura da cavidade H_2 e a altura do canal H_1 é igual a 6,3. O comprimento da cavidade (L_2) é igual à altura da cavidade (H_2); a razão entre o comprimento do canal à montante e à jusante L_1 e a altura do canal H_1 é igual a 16,85. Dessa forma, o comprimento total do canal é igual a 40 vezes a sua altura. Essa geometria foi inspirada em uma das geometrias adotadas por de Souza Mendes et al. (2007b), sendo que a principal diferença entre elas é que a geometria de de Souza Mendes et al. (2007b) é um canal axissimétrico e a geometria do presente trabalho é um canal planar.

Para reduzir o custo computacional, apenas metade da geometria foi simulada, adotando a linha de simetria em $y^* = 0$. As condições de contorno para a velocidade são não-deslizamento no contato com as paredes e um perfil constante de velocidade na região de entrada. A condição imposta na saída é o perfil de pressão constante. As paredes do canal são consideradas adiabáticas e a temperatura adimensional nas paredes da cavidade é constante e igual a um. A temperatura adimensional do fluido que entra no canal é igual a zero.

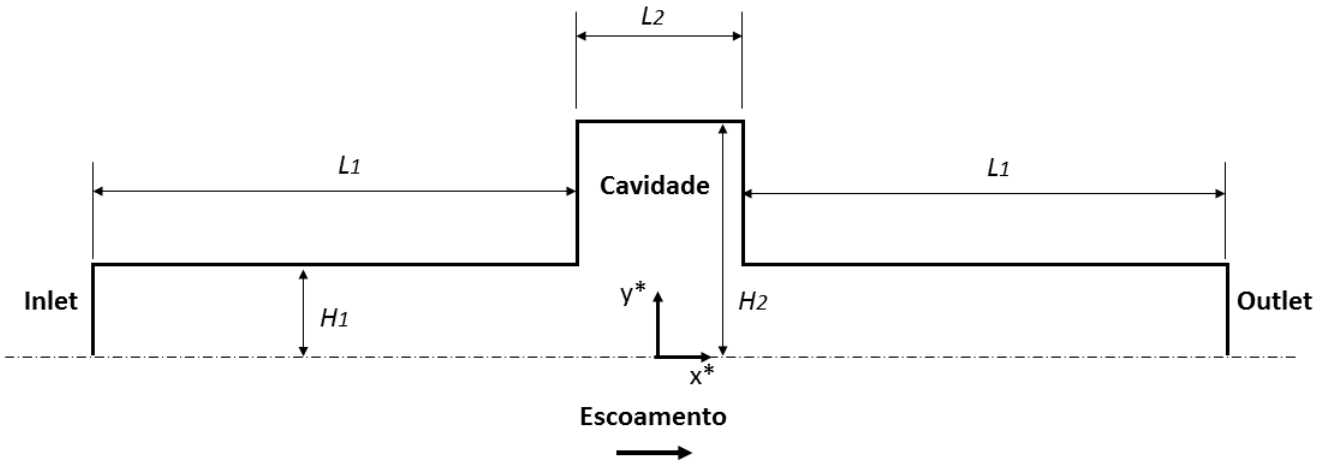


Figura 7.1 – Geometria do canal planar com a cavidade.

7.3 Malha

A malha adotada é completamente estruturada, e como vários parâmetros do escoamentos foram estudados, não é adequado estabelecer um padrão de refinamento, já que as características do escoamento podem mudar significativamente, e um padrão de refinamento poderia não contemplar todas as diferentes configurações. Dessa forma, os volumes de controle da malha podem ser divididos em quatro grupos principais, e as dimensões dos volumes de controle de cada um desses grupos são constantes:

- O canal à montante é composto por volumes de controle retangulares com a proporção entre altura e comprimento igual a 1:2.
- O interior da cavidade é formado por volumes de controle quadrados.
- O canal logo abaixo da cavidade é composto por volumes de controle quadrados.
- O canal à jusante é formado por volumes de controle retangulares com a proporção entre altura e comprimento igual a 1:2.

A proporção exata de cada região pode variar de acordo com a geometria da região. Como a malha é completamente estruturada, seguindo as proporções propostas acima, a altura de todos os volumes de controle do canal é constante, e a diferença entre a altura dos volumes de controle do canal e do interior da cavidade se dá apenas por causa das dimensões dessas regiões. Além disso, os elementos do interior da cavidade e do canal logo abaixo dela

apresentam o mesmo comprimento. Dessa maneira, basta definir a altura de um elemento no canal que a malha como um todo pode ser obtida.

A análise da qualidade de malha seguiu o mesmo procedimento descrito por Celik (2008). Como vários parâmetros foram analisados, o procedimento foi realizado diversas vezes, tentando contemplar todas as condições críticas. A Tab. 7.1 apresenta as malhas usadas no processo de validação, onde h^*_{teo} é o valor teórico da altura adimensional de um volume de controle na região de entrada do fluido. Como discutido na Seção 5.1.1, o valor de r é mantido relativamente baixo, para se evitar que os erros numéricos comprometam a validação. Neste caso, r_{12} e r_{23} são iguais a 1,5.

Tabela 7.1 – Parâmetros das malhas adotadas na análise da qualidade de malha.

Malha	3	2	1
Número de elementos	2044	4542	10304
h^*_{teo}	0,1687	0,1125	0.075

Três critérios foram adotados para avaliar a qualidade de malha: o $\overline{Nu}$, o perfil de velocidade e Δp^*_{10}. Δp^*_{10} é a diferença entre a pressão medida nos pontos $(x^*, y^*) = (-10, 0)$ e $(x^*, y^*) = (10, 0)$. Dadas as características do escoamento, $|\mathbf{u}|_{max}$ ou qualquer uma de suas componentes não são critérios adequados para se avaliar a qualidade de malha, uma vez que $|\mathbf{u}|_{max}$ sempre é localizado sobre a linha de simetria, por exemplo. Da mesma maneira, impor que a velocidade será avaliada sobre um ponto arbitrariamente escolhido também não é uma metodologia adequada, uma vez que diversos parâmetros do escoamento são analisados, e existe a possibilidade que este ponto escolhido esteja situado em uma região de transição entre uma zona escoada e uma zona aparentemente não escoada. Como já discutido, essa região apresenta erros numéricos maiores que nas outras regiões, em especial para malhas mais grosseiras, o que pode comprometer a qualidade da análise. Para se evitar esses problemas, foi escolhido um novo critério: todo o perfil de $|\mathbf{u}^*|$ na direção do eixo y^* em $x^* = 0$ foi avaliado, usando mil pontos igualmente espaçados como pontos de observação. O erro apresentado aqui é a média ponderada do erro encontrado, usando o perfil de $|\mathbf{u}^*|_{y^*=0}|$ da malha mais fina como função peso. Essa metodologia foi adotada para dar prioridade aos pontos de maior velocidade, e que consequentemente influenciam mais o escoamento. Assim, filtra-se os erros em pontos onde a velocidade é muito baixa, como nas zonas aparentemente não escoadas estacionárias.

A Tab. 7.2 apresenta os resultados da análise da qualidade de malha para as condições de

Re mínimo ($Re = 0,1$) e Re máximo ($Re = 50$). De maneira semelhante, a Tab. 7.3 apresenta os resultados para as condições de Pl mínimo ($Pl = 0,1$) e Pl máximo ($Pl = 0,8$), e a Tab. 7.4 expõe os resultados das condições de n mínimo ($n = 0,3$) e n máximo ($n = 1,2$). Todo os erros em relação ao valor extrapolado são menores do que 3%, tanto para propriedades locais, como o perfil de velocidade e Δp^*_{10}, quanto para propriedades globais. O parâmetro que sofre maior influência da malha é o $\overline{Nu}$, devido ao fato que para o cálculo desse parâmetro, é necessário calcular o gradiente de temperatua, extrapolar o seu valor para as faces do volume de controle, e então integrá-lo sobre a superfície. Dessa forma, é necessário o uso de esquemas numéricos de três naturezas diferentes, e o tamanho da malha tem influência direta nos erros gerados por cada um deles.

Tabela 7.2 – Análise da qualidade de malha para Re mínimo ($Re = 0,1$) e Re máximo ($Re = 50$).

		$\overline{Nu}$			$\lvert\mathbf{u}^*\rvert_{y^*=0}$			Δp^*_{10}		
	malha	3	2	1	3	2	1	3	2	1
$Re = 0,1$	ϕ	1,10	1,13	1,15	-	-	-	79,1	75,1	75,1
	ϕ_{ext}	1,16			-			75,2		
	$e_{ext}(\%)$	5,17	2,84	0,83	1,95	2,83	1,25	5,22	0,13	0,05
	$CGI^{21}_{fine}(\%)$	1,05			0,59			0,07		
$Re = 50$	ϕ	10,5	11,2	11,9	-	-	-	84,0	83,1	83,0
	ϕ_{ext}	12,2			-			82,9		
	$e_{ext}(\%)$	13,7	7,68	2,24	5,58	2,67	1,17	1,23	0,26	0,11
	$CGI^{21}_{fine}(\%)$	0,07			0,41			0,14		

7.4 Critério de convergência

Esta seção busca definir os critérios que devem garantir que a solução do sistema convergiu. Uma simulação com o conjunto padrão de adimensionais foi realizada e diversos parâmetros de interesse foram monitorados, como $\overline{Nu}$, Φ_{de} e a pressão máxima (Δp^*_{max}) calculada sobre toda a malha. Hl é um parâmetro de difícil monitoramento, devido aos diversos fatores que o compõem, e dessa forma foi substituído por Δp^*_{max}. Δp^*_{max} não é um parâmetro que será analisado como resultado, mas ainda assim ele reflete com certa precisão a qualidade da solução do campo de pressão: por definição, o ponto de maior pressão ocorre sempre na quina

Tabela 7.3 – Análise da qualidade de malha para Pl mínimo ($Pl = 0,1$) e Pl máximo ($Pl = 0,8$).

		$\overline{Nu}$			$\lvert\mathbf{u}^*\rvert_{y^*=0}$			Δp^*_{10}		
	malha	3	2	1	3	2	1	3	2	1
$Pl = 0,1$	ϕ	10,0	10,6	11,1	-	-	-	355	357	357
	ϕ_{ext}	11,3			-			357		
	$e_{ext}(\%)$	11,4	5,70	1,66	2,97	1,64	0,72	0,57	$8,15 \times 10^{-3}$	$3,59 \times 10^{-3}$
	$CGI^{21}_{fine}(\%)$	2,12			0,53			$4,49 \times 10^{-3}$		
$Pl = 0,8$	ϕ	5,99	6,39	6,76	-	-	-	34,7	34,0	33,5
	ϕ_{ext}	6,91			-			33,1		
	$e_{ext}(\%)$	13,3	7,60	2,22	10,0	5,03	2,21	4,64	2,48	1,09
	$CGI^{21}_{fine}(\%)$	2,84			0,8			1,35		

Tabela 7.4 – Análise da qualidade de malha para n mínimo ($n = 0,3$) e n máximo ($n = 1,2$).

		$\overline{Nu}$			$\lvert\mathbf{u}^*\rvert_{y^*=0}$			Δp^*_{10}		
	malha	3	2	1	3	2	1	3	2	1
$n = 0,3$	ϕ	6,62	7,14	7,70	-	-	-	73,0	71,2	70,9
	ϕ_{ext}	7,93			-			70,6		
	$e_{ext}(\%)$	16,5	9,94	2,91	5,05	2,42	1,06	3,43	0,93	0,41
	$CGI^{21}_{fine}(\%)$	3,75			0,57			0,51		
$n = 1,2$	ϕ	8,85	9,60	10,2	-	-	-	117,0	116	117
	ϕ_{ext}	10,4			-			117		
	$e_{ext}(\%)$	15,2	8,09	2,36	4,82	1,06	0,46	0,06	0,99	0,43
	$CGI^{21}_{fine}(\%)$	3,03			0,32			0,55		

superior da região de entrada do fluido, devido às condições de contorno impostas. Como na região de saída do fluido a pressão foi definida como zero, Δp^*_{max} pode ser interpretado como um gradiente de pressão que engloba toda a geometria do problema, e assim, qualquer erro ou mudança no campo de pressão tende a ter reflexos no valor de Δp^*_{max}.

Para facilitar a análise, o erro relativo de $\overline{Nu}$, Φ_{de} e Δp^*_{max} foi definido da seguinte maneira:

$$\overline{Nu}_{er} = \frac{|\overline{Nu} - \overline{Nu}_f|}{\overline{Nu}_f} \tag{7.12}$$

$$\Phi_{de,er} = \frac{|\Phi_{de} - \Phi_{de,f}|}{\Phi_{de,f}} \tag{7.13}$$

$$\Delta p^*_{max,er} = \frac{|\Delta p^*_{max,er} - p^*_{max,f}|}{p^*_{max,f}} \tag{7.14}$$

onde o subíndice f indica que este parâmetro foi avaliado com condição de resíduo normalizado máximo igual a 10^{-12}, que foi o critério adotado arbitrariamente como sendo a condição de solução convergida.

A Fig. 7.2 e apresenta os resultados de $\overline{Nu}_{er}$ e $\Delta p^*_{max,er}$ em função do resíduo normalizado máximo, e a Fig. 7.3 apresenta $\Phi_{de,er}$ também em função do resíduo máximo. A eficiência de deslocamento não é um parâmetro contínuo: um volume da malha pode estar ou não em uma zona aparentemente não escoada. Assim, ao diminuir o resíduo normalizado, é possível se obter o valor exato de $\Phi_{de,f}$, daí a necessidade de se usar uma escala linear. Como pode ser observado, o sistema começa a apresentar características de solução convergida quando o resíduo normalizado está na ordem de 10^{-8}. Para garantir a convergência nos diversos casos estudados, foi adotado o critério de resíduo normalizado máximo igual a 10^{-9}.

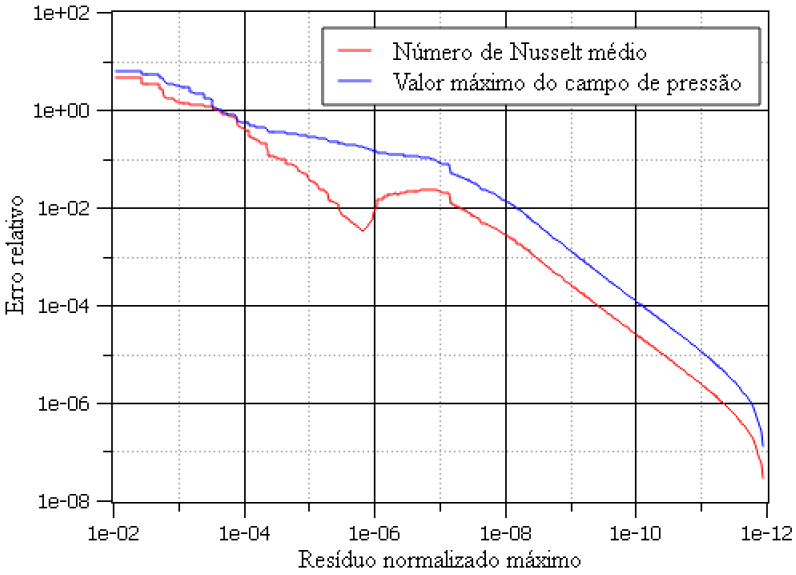


Figura 7.2 – Erro relativo $\overline{Nu}_{er}$ e $\Delta p_{max,er}$ em função do resíduo normalizado máximo para o conjunto padrão de adimensionais.

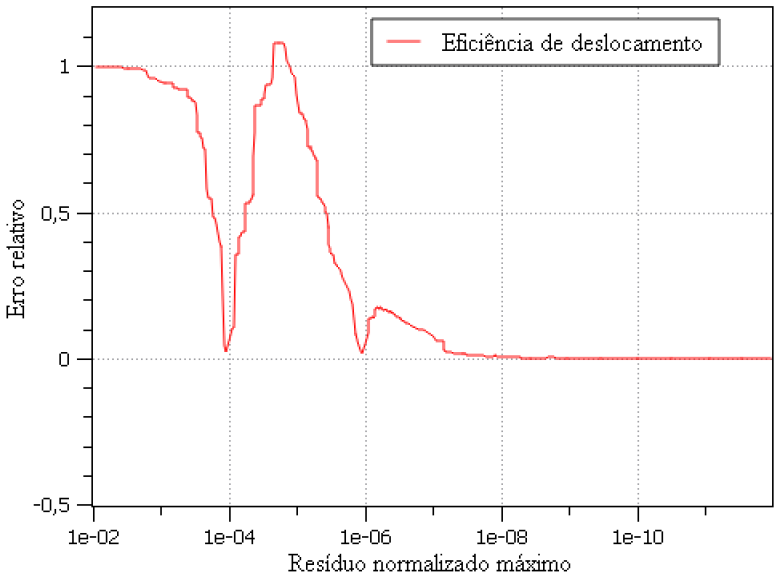


Figura 7.3 – Erro relativo $\Phi_{de,er}$ em função do resíduo normalizado máximo para o conjunto padrão de adimensionais.

7.5 Análise numérica

Esta seção tem por objetivo apresentar uma breve análise numérica do processo de solução dos escoamentos propostos. Para se identificar como cada comportamento do fluido influencia o número de iterações necessário para se obter uma taxa de resíduo normalizado máxima menor que 10^{-9}, foram realizados os seguintes testes: uma simulação adotando o conjunto padrão de adimensionais (*Re=25; Pl=0,4; Pr=14,02; J=*10^4*; n=0,5;* η^*_{inf}*=0,01*), uma simulação adotando um fluido pseudoplástico, uma simulação com fluido de Bingham e uma simulação considerando o fluido como sendo newtoniano. A função SMD foi utilizada em todas essas simulações, impondo $n = 1$ no fluido de Bingham e no fluido newtoniano, e um Pl suficientemente baixo ($Pl < 10^{-6}$) em simulações de fluidos pseudoplásticos e newtonianos. Os outros grupos adimensionais foram mantidos contantes e iguais ao conjunto padrão de adimensionais, quando possível. O número de iterações necessário para resolver cada um desses escoamentos é apresentado na Tab. 7.5. Todos os parâmetros numéricos foram mantidos constantes, e o critério de convergência foi o resíduo normalizado máximo igual a 10^{-9}. Como pode ser observado, a introdução do comportamento viscoplástico aumenta drasticamente o número necessário de iterações para o sistema convergir. A influência do comportamento pseudoplástico sobre o número e iterações é sensivelmente menor.

A Fig. 7.4 e a Tab. 7.6 apresentam o número de iterações em função do resíduo norma-

Tabela 7.5 – Comparação entre os diferentes tipos de fluido.

Condição	número de iterações
Fluido newtoniano ($n = 1; Pl < 10^{-6}; \eta^*_\infty = 0$)	46793
Fluido pseudoplástico ($Pl < 10^{-6}$)	46861
Fluido de Bingham ($n = 1; \eta^*_\infty = 0$)	3659670
Conjunto padrão de adimensionais	2936640

lizado máximo, que é o critério de convergência adotado neste trabalho, para uma simulação considerando o conjunto padrão de adimensionais. Dessa forma, é possível se analisar como um critério de convergência mais rígido influência no número de iterações da simulação, e consequentemente no tempo para se obter a solução. Já a Fig. 7.5 apresenta o resíduo normalizado de cada um dos sistemas lineares resolvidos na simulação em função do número de iterações. Na Fig. 7.4 é possível observar que o resíduo máximo não decai de maneira constante em relação ao número de interações: é possível observar degraus em determinados momentos, onde o número de iterações aumenta expressivamente, mas o resíduo normalizado máximo decai de maneira discreta. Além disso, a Fig. 7.5 ilustra instabilidades em todas as curvas, mas mais acentuadas no resíduo normalizado do campo da pressão. Uma vez mais, esses dois comportamentos são provocados pelo comportamento viscoplástico: a velocidade no interior de uma *plug-zone*, por exemplo, é praticamente constante, devido à baixa taxa de deformação. Durante a solução do escoamento, qualquer alteração no tamanho da *plug-zone* gera a necessidade de se alterar a velocidade em toda a *plug-zone*, para respeitar a equação de conservação de massa. Além disso, a brusca alteração na viscosidade também gera dificuldades na solução da equação de conservação de quantidade de movimento. Essa transição é refletida em todos os campos, como pode ser comparando as Fig. 7.3 e Fig. 7.5: quando a eficiência de deslocamento se estabiliza, todas as outras propriedades passam a convergir de maneira mais constante, mesmo aqueles parâmetros associados ao escoamento nos canais, como Δp^*_{max}. Esse resultado leva à conclusão que todas as zonas aparentemente não escoadas do escoamento se estabilizam em torno de um valor do resíduo máximo normalizado, e não apenas as zonas aparentemente não escoadas da cavidade. Outra característica imposta por essa condição é que é possível que uma solução não apresente estabilidade suficiente para superar essas zonas de instabilidade, o que impede a convergência da solução. Para se solucionar esse problema, uma possível alternativa é diminuir o valor do fator de relaxação da equação de conservação de quantidade de movimento linear, aumentando assim a estabilidade da solução.

Tabela 7.6 – Número de iterações em função do resíduo normalizado máximo (r_{max}) para o conjunto padrão de adimensionais.

r_{max}	número de iterações	$\overline{Nu}_{er}$	$\Delta p^*_{max,er}$	$\Phi_{de,er}$
10^{-7}	291000	$1{,}77\times10^{-2}$	$7{,}86\times10^{-2}$	$7{,}34\times10^{-2}$
10^{-8}	1505000	$2{,}07\times10^{-3}$	$1{,}31\times10^{-2}$	$2{,}96\times10^{-3}$
10^{-9}	2741000	$2{,}23\times10^{-4}$	$1{,}61\times10^{-3}$	$1{,}48\times10^{-3}$
10^{-10}	4078000	$2{,}25\times10^{-5}$	$1{,}25\times10^{-4}$	0
10^{-11}	5365000	$1{,}98\times10^{-6}$	$8{,}40\times10^{-6}$	0

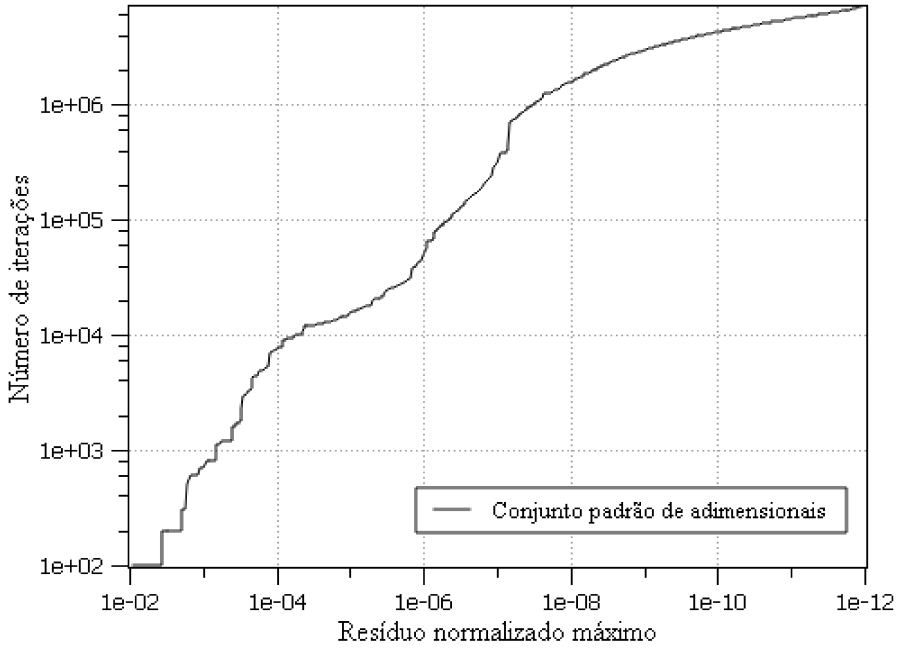


Figura 7.4 – Número de iterações em função do resíduo normalizado máximo para o conjunto padrão de adimensionais.

Comparando a Tab. 7.5 e a Tab. 7.6, é possível perceber que o número de iterações para a simulação com o conjunto padrão de adimensionais e $r_{max} = 10^{-9}$ apresenta uma variação. Essa variação é provocada pela alteração dos parâmetros numéricos. Como já discutido, uma estratégia para aumentar a eficiência do código é afrouxar os parâmetros numéricos da solução com campo de pressão, e essa estratégia foi adotada ao realizar a simulação da Tab. 7.5. Dessa forma, mesmo com o aumento do número de iterações, o tempo necessário para a realização de cada iteração diminuiu, e assim o tempo total da simulação também diminuiu. Na simulação da Tab. 7.6 uma configuração numérica mais conservadora foi adotada para se garantir que $r_{max} = 10^{-12}$ seria atingido. Todas as simulações apresentadas neste trabalho foram realizadas em um computador de uso pessoal, e várias vezes mais de uma simulação foi realizada simultaneamente. Essas características comprometeram as comparações do tempo

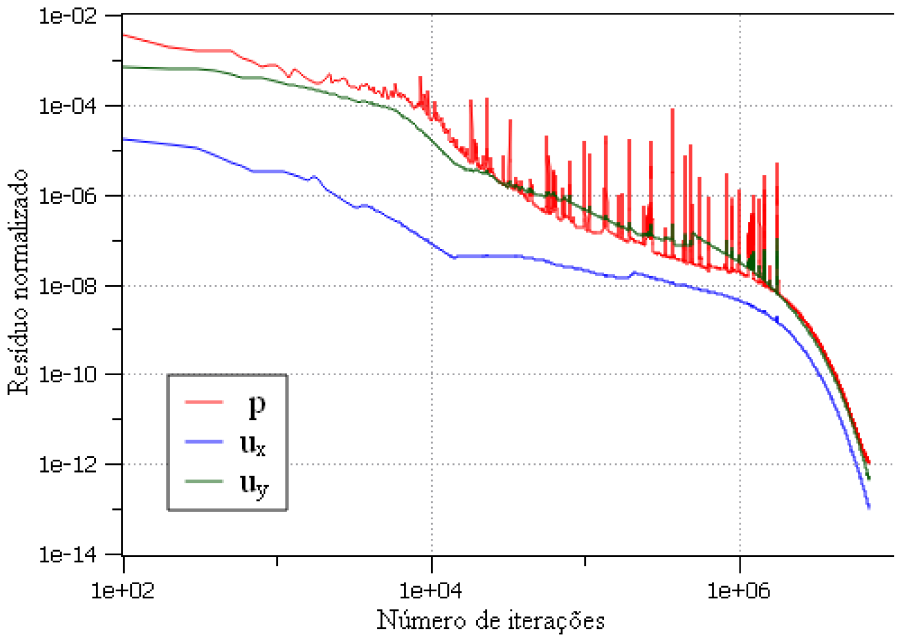


Figura 7.5 – Resíduo normalizado de cada uma das variáveis do sistema em função do número de iterações, adotando o conjunto padrão de adimensionais.

de duração de cada simulação, e por isso todos os resultados de eficiência de código aqui apresentadas estão em função do número de iterações.

7.6 Pré-processamento

7.6.1 blockmesh

A malha é formada por quatro regiões diferentes: os dois canais, a cavidade e o canal abaixo dela. Como a malha é uniforme, não é necessário se definir um coeficiente de expansão como foi feito na Seção 6.3.1. Porém, é necessário tomar cuidado para que as linhas das malhas dos blocos se conectem, impondo assim a condição de que blocos vizinhos devam ter o mesmo número de elementos na face em comum. Para evitar essa condição, é possível usar a função *mergePatchPairs*. Além disso, para malhas mais complexas é mais interessante gerar a malha com outros *softwares* como o *gmsh*, e então usar as funções do OpenFOAM® para convertê-las. Neste trabalho, esses recursos não foram utilizados, e as malhas de cada bloco foram escolhidas de forma a garantir que toda a malha seja estruturada.

7.6.2 Pasta 0

Para o campo de velocidades, a condição imposta sobre todas as paredes foi de velocidade imposta (*fixedValue*) igual a um vetor nulo. Na região de entrada (*inlet*), a condição também é a de velocidade imposta, porém o vetor contém uma componente não nula na direção x^*. Na região de saída (*outlet*), a condição imposta foi a de gradiente nulo(*zeroGradient*). Não é necessário impor uma condição sobre o plano de simetria, o tipo base dessa condição de contorno já é condição suficiente para o OpenFOAM®.

Para a pressão, todas as paredes e a região de entrada apresentam a condição de gradiente nulo. A única condição de contorno diferente é a da região de saída, onde a condição imposta é a condição de valor fixo, no caso, igual a zero.

Para o campo de temperatura, as paredes dos canais tem condição de gradiente zero, para modelar a condição de parede adiabática. A região de saída apresenta a mesma condição de gradiente nulo. As paredes da cavidade tem temperatura imposta igual a um. A região de entrada tem a temperatura imposta igual a zero.

O arquivo *scalarDict* é exatamente igual ao usado na Seção 6. Já os arquivos *tau, sr, yieldGama* e *yieldGama2* continuam com a condição "*calculated*", mas é necessário que os nomes de cada conjunto de contornos esteja de acordo com o definido no arquivo *blockmesh*.

7.7 Pós-processamento

Para calcular o valor do $\overline{Nu}$, o mesmo procedimento que foi utilizado durante a validação será aplicado, a única diferença é o nome da superfície onde a integral foi realizada. Para calcular a eficiência de deslocamento, a função *cellAvarege* foi aplicada sobre o campo *yieldGama2*. A função *cellAvarege* calcula a média em todo o domínio, e para obter o valor apenas da média dentro da cavidade, as outras zonas aparentemente não escoadas foram filtradas na definição de *yieldGama2* e o campo resultante foi dividido pela razão da área da cavidade sobre a área do domínio. Assim, quando a média foi realizada pela função *cellAvarege*, o valor obtido corresponde à porcentagem de zonas escoadas dentro da cavidade. A opção "*operation volAverage;*" é usada para definir que a média deve ser calculada como uma média ponderada usando o volume de cada célula como função peso.

O parâmetro *head loss* é de difícil determinação exclusivamente pelo OpenFOAM®: a

metodologia mais simples é usar a função *graph* para se obter o valor da pressão nos pontos desejados, e então usar outro software, como o Scilab, MATLAB® ou mesmo uma planilha de OpenOffice, por exemplo, para realizar os outros cálculos. Dessa forma, também é possível automatizar muito do pós-processamento, organizando os resultados do OpenFOAM®. Além da função *graph*, a função *probes* ou a função *boundaryCloud* também podem ser usadas para monitorar o valor da pressão em um ponto específico, tomando cuidado uma vez que a função *probes* retorna o valor do centroide mais próximo da posição selecionada, e não o valor interpolado daquela posição.

O cálculo de Φ_{sta} envolve verificar se a zona aparentemente não escoada é estacionária ou não. À primeira vista, a maneira mais fácil de se calcular isso seria acrescentar mais um filtro, usando uma velocidade de referência, por exemplo. Porém, essa metodologia apresenta uma série de limitações, como a necessidade de se definir essa velocidade de referência, e o fato de que o interior da zona aparentemente não escoada móvel no centro da recirculação também apresenta velocidades muito baixas compromete a qualidade dos resultados. Como a velocidade no interior de uma zona aparentemente não escoada não rotativas é relativamente constante e a condição de contorno aplicada sobre o campo de velocidade nas paredes da cavidade é a condição de velocidade nula, é razoável considerar que todas as zonas aparentemente não escoadas conectadas às paredes são também estacionárias. Assim, neste trabalho, Φ_{sta} foi calculado em uma rotina fora do OpenFOAM®que verificava se cada célula pertencia a uma zona aparentemente não escoada e também se aquela célula estava ligada, direta ou indiretamente, a uma parede da cavidade.

7.8 Resultados

7.8.1 Influência do número de Reynolds

Para analisar os efeitos do número de Reynolds sobre o escoamento, foram realizadas simulações com este parâmetro variando-o de 0,1 até 50. Os outros grupos adimensionais ($Pr, J, Pl, \eta^*_{\inf}$ e n) foram mantidos constantes e iguais ao conjunto padrão de adimensionais. A Fig. 7.6 mostra os perfis de magnitude da velocidade na direção do eixo y^* em $x^* = 0$. Para $Re > 20$, os perfis de velocidade apresentem um aumento considerável após y^*=2, indicando a presença de uma zona de recirculação dentro da cavidade formada pela geometria do canal.

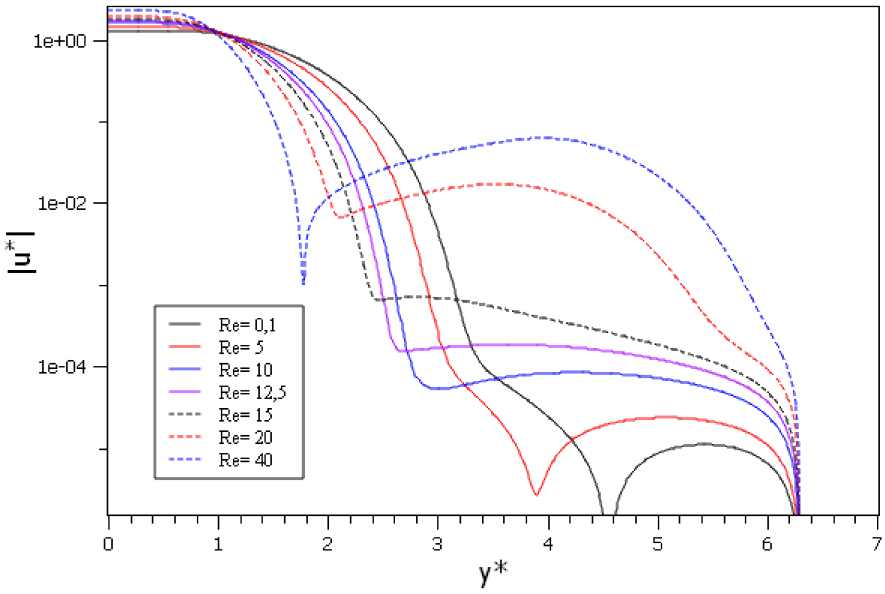


Figura 7.6 – Magnitude da velocidade na direção do eixo y^* em $x^* = 0$ para $0,1 \leq Re \leq 40$.

Teoricamente, todo escoamento apresenta pelo menos uma zona de recirculação dentro da cavidade, e possivelmente até mais de uma. Porém, o platô de alta viscosidade aparente para baixas taxas de deformação característico do comportamento viscoplástico impede que algumas dessas zonas de recirculação escoem de fato. Isso pode ser observado na Fig. 7.7, que mostra a sobreposição de trajetórias de partículas sobre as zonas escoadas e aparentemente não escoadas de uma simulação onde $Pl = 0,8$, e os outros grupos adimensionais foram mantidos iguais ao conjunto padrão de adimensionais. Na Fig. 7.7, é possível identificar uma zona de recirculação no interior de uma zona aparentemente não escoada estacionária. Isso significa que a velocidade da zona de recirculação é baixa o suficiente para não influenciar o escoamento. Outro exemplo desse comportamento pode ser observado na Fig. 7.6, onde mesmo para *Re* < 10, o perfil de velocidade apresenta um aumento da sua magnitude dentro da cavidade, mas ainda assim a magnitude da velocidade continua algumas ordens de grandeza abaixo da velocidade na linha de simetria, por exemplo.

Com o aumento do número de Reynolds, as zonas de recirculação passam a ultrapassar a tensão limite de escoamento τ_0, como pode ser visto na Fig. 7.8 e na Fig. 7.9. Aumentando o número de Reynolds ainda mais, a zona de recirculação continua crescendo, até um limite imposto pela geometria da cavidade. A Fig. 7.8(c) mostra claramente que existe uma zona de recirculação dentro da cavidade, ainda que devido ao seu tamanho e posição do seu centro, o perfil de velocidade para $Re = 15$ da Fig. 7.6 quase não apresente evidências da presença

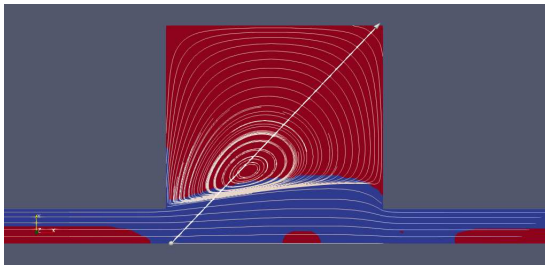
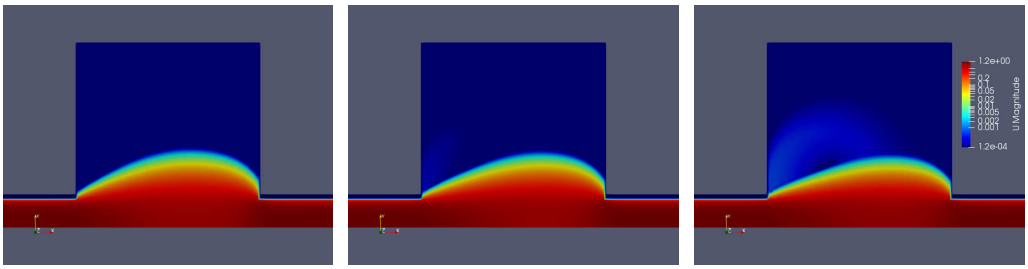
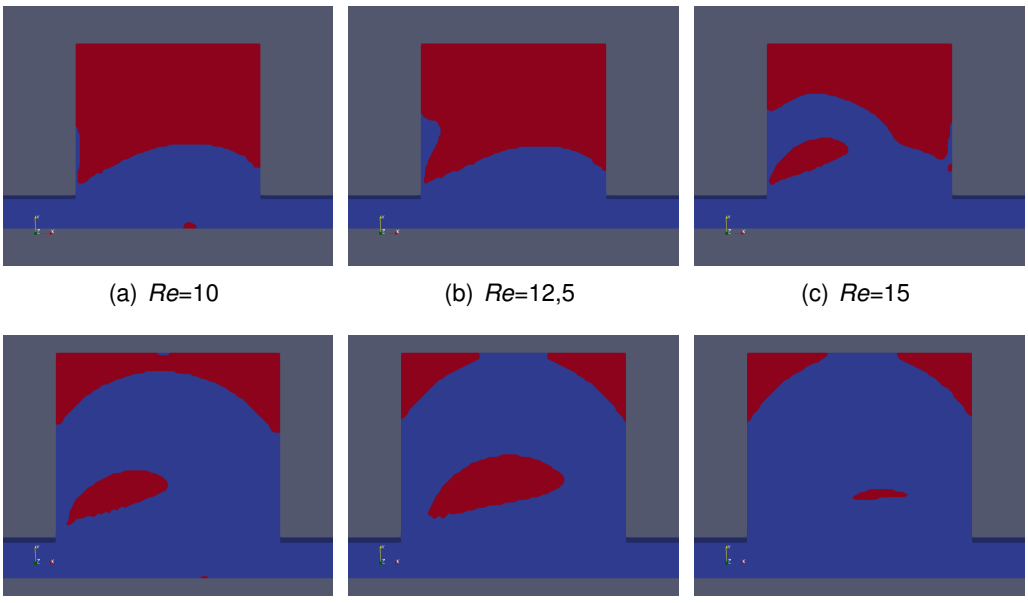


Figura 7.7 – Linhas de fluxo e zonas aparentemente não escoadas (em vermelho) e aparentemente escoadas (em azul) para *Re=25; Pl=0,8; Pr=14,02; J*=10^4*; n=0,5;* η^*_{inf}*=0,01*.



(a) *Re*=10 (b) *Re*=12,5 (c) *Re*=15

Figura 7.8 – Magnitude da velocidade dentro da cavidade para (a) *Re*=10; (b) *Re*=12,5; (c) *Re*=15. Todas a figuras compartilham a mesma escala.



(a) *Re*=10 (b) *Re*=12,5 (c) *Re*=15

(d) *Re*=20 (e) *Re*=30 (f) *Re*=50

Figura 7.9 – Zonas aparentemente não escoadas (em vermelho) e aparentemente escoadas (em azul) para (a) *Re*=10; (b) *Re*=12,5; (c) *Re*=15; (d) *Re*=20; (e) *Re*=30; (f) *Re*=50.

dessa zona de recirculação.

A Fig. 7.9(c) indica que uma nova zona aparentemente não escoada foi formada no interior

da zona de recirculação. Com essa nova zona de recirculação, é possível identificar três tipos de zonas aparentemente não escoadas no escoamento

- a *plug zone*: uma zona aparentemente não escoada móvel que ocorre principalmente no canal, próxima à linha de simetria;
- o centro da zona de recirculação: uma zona aparentemente não escoada móvel que gira em torno de um ponto;
- a zona aparentemente não escoada estacionária: que ocorre principalmente no interior da cavidade próxima às paredes e quinas.

Esses dois últimos tipos de de zonas aparentemente não escoadas podem estar presentes dentro da cavidade, mas apenas um deles é estacionário. A Fig. 7.10 mostra a eficiência de deslocamento Φ_{de} e Φ_{sta}. Para $Re < 10$, não existe uma zona de recirculação dentro da cavidade, e as duas eficiências de deslocamento apresentam os mesmos valores. Para $Re = 0,1$, as forças de inércia são negligenciáveis e o escoamento apresenta uma simetria em relação ao eixo y^*. Ao aumentar *Re*, a simetria é quebrada, e a região aparentemente não escoada se inclina na direção do canal a jusante, como pode ser visto na Fig. 7.8(a). Esse deslocamento aumenta esta região dentro da cavidade, diminuindo assim a eficiência de deslocamento. Esse comportamento é o oposto da tendência global, onde a eficiência de deslocamento tende a aumentar com o aumento de *Re*, em especial por causa da zona de recirculação. O platô de Φ_{sta} entre *Re*=30 e *Re*=50 indica que a zona de recirculação atingiu seu tamanho máximo. Já Φ_{de} não apresenta o mesmo comportamento, indicando que a zona aparentemente não escoada do centro da recirculação ainda está se modificando neste intervalo de *Re*. Além disso, a Fig. 7.9 também mostra como se comporta a zona aparentemente não escoada móvel presente no centro da zona de recirculação. É possível identificar duas tendência diferentes: o deslocamento do centro da zona de recirculação em direção ao centro da cavidade entre *Re*=20 e *Re*=30 favorece o crescimento da zona aparentemente não escoada móvel nesse intervalo, mas o aumento de *Re* também acelera a zona de recirculação, o que provoca a diminuição da zona aparentemente não escoada móvel entre *Re*=30 e *Re*=50.

A Fig. 7.10 também apresenta o número de Nusselt médio em função do número de Reynolds. *Re* baixos dificultam a transferência de calor por convecção, logo era esperado $\overline{Nu}$ menores. Além disso, o surgimento da zona de recirculação favorece a transferência de

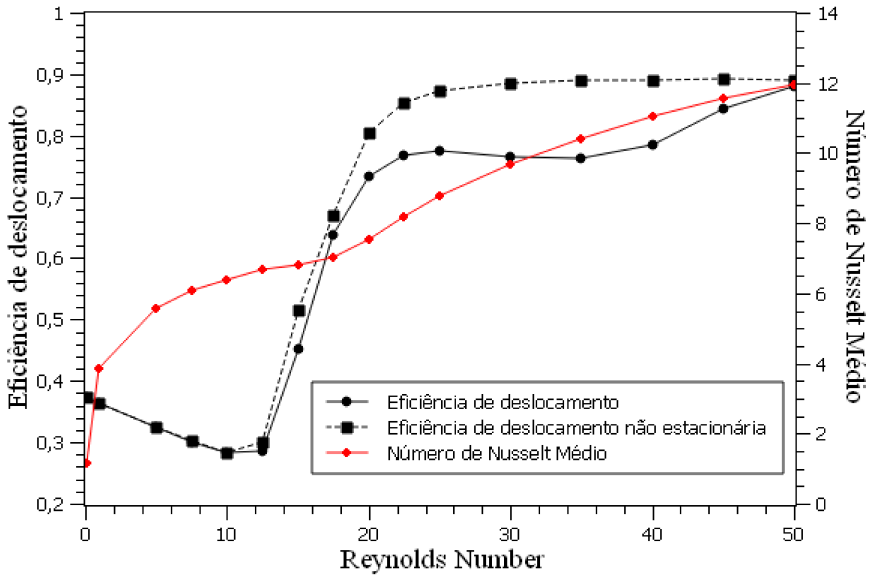


Figura 7.10 – Número de Nusselt médio, eficiência de deslocamento e a eficiência de deslocamento não estacionária para $0,1 \leq Re \leq 50$.

calor por convecção. Assim, é possível identificar duas principais tendências: $\overline{Nu}$ aumenta naturalmente com o aumento de *Re*, e essa tendência é intensificada pela presença da zona de recirculação.

A Fig. 7.11 apresenta os perfis de temperatura na direção do eixo y^* em $x^* = 0$. A presença da zona de recirculação cria um platô de temperatura dentro da cavidade. Como a Fig. 7.8(c) mostra, a zona de recirculação emerge para $Re > 15$, mas os perfis de temperatura só apresentam um platô de forma clara para $Re > 25$, como pode ser visto na Fig. 7.11 . Essa diferença pode ser explicada não somente pelo tamanho e posição da zona de recirculação, mas também pela intesidade do fluxo nessa região: quanto maior o fluxo na zona de recirculação, maior a transferência de calor, e assim, para que o platô de viscosidade seja observado no perfil de temperatura, é necessário que a zona de recirculação apresente uma fluxo suficientemente grande para influenciar a transferência de calor. Embora $Re = 0,1$ apresente o menor $\overline{Nu}$, este é o perfil com as maiores temperaturas. Isso ocorre devido à transferência de calor por condução que é mais relevante neste caso.

Os resultados para Hl são exibidos pela Fig. 7.12. Quando as forças de inércia são baixas, Hl é alto e positivo, o que significa que o canal com a cavidade apresenta uma queda de pressão menor que o canal simples. Isso ocorre pois o canal simples sofre mais influência das forças viscosas, uma vez que a cavidade é um canal de seção maior, e que assim oferece

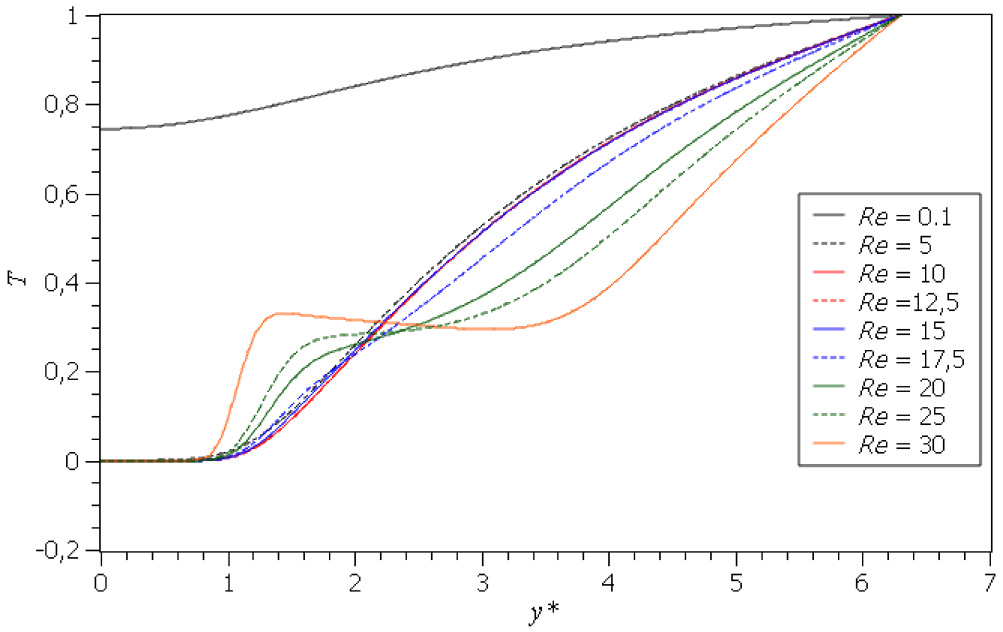


Figura 7.11 – Perfis de temperatura na direção do eixo y^* em $x^* = 0$ para 0,1$\leq$ *Re*$\leq$30.

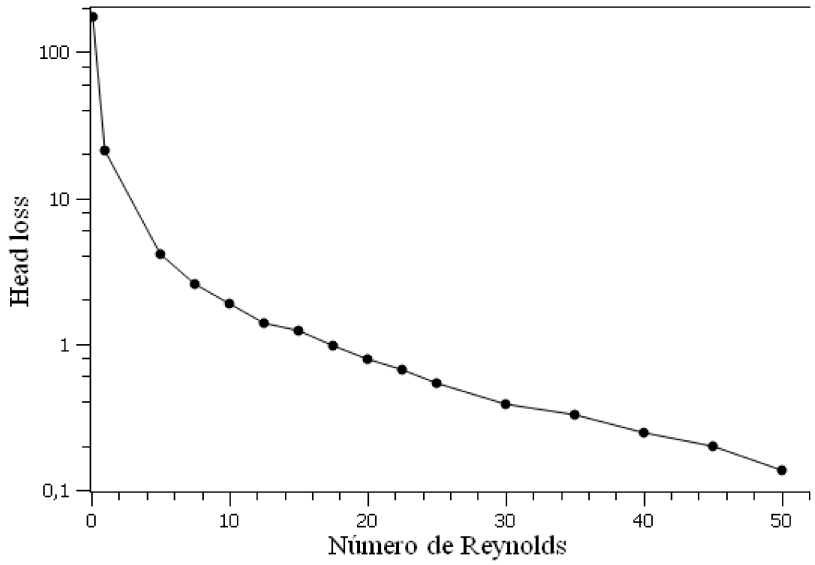


Figura 7.12 – Hl para 0,1$\leq$ *Re*$\leq$50.

menos resistência ao escoamento. Com o aumento das forças de inércia, a diferença entre as duas geometrias diminui, ainda que Hl continue positivo, ou seja, o canal com a cavidade continua com uma queda de pressão menor que o canal simples. Além disso, a Fig. 7.12 não apresenta nenhum sinal claro de influência da zona de recirculação sobre Hl, o que já era esperado, uma vez que a queda de pressão Δp_c é medida em pontos onde o escoamento é considerado como desenvolvido, ou seja, são pontos afastados da cavidade.

Comparando os resultados da Fig. 7.10 e da Fig. 7.12, é possível concluir que não existe um valor de *Re* que provoque uma situação ideal, com Hl e $\overline{Nu}$ altos. Porém, como a zona de recirculação não apresenta indícios de influência sobre Hl, uma maneira de se otimizar o sistema seria provocar a zona de recirculação para *Re* menores, aumentando $\overline{Nu}$ sem que Hl diminua significativamente.

7.8.2 Influência do plastic number

Nesta seção, os resultados das simulações variando *Pl* são discutidos. Os outros parâmentros adimensionais (*Re, Pr, J,* η^*_{inf} *e* n) são definidos pelo conjunto padrão de adimensionais. Um intervalo de $0,1 \leq Pl \leq 0,8$ foi analisado. Um *Pl* igual a zero significa que o fluido não apresenta nenhum comportamento viscoplástico, e um *Pl* igual a um significa indica plasticidade máxima (Thompson e Soares (2016)). A Fig. 7.13 apresenta os resultados para $\overline{Nu}$, Φ_{de} e Φ_{sta}.

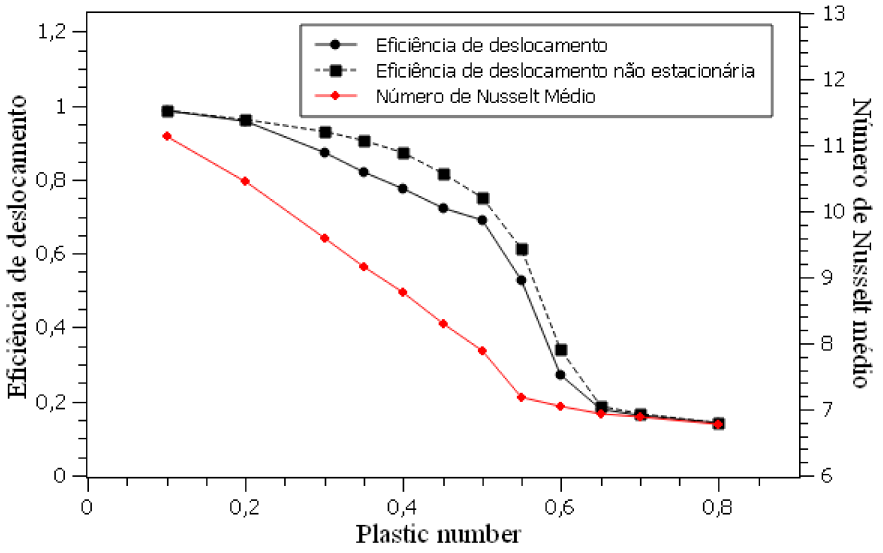


Figura 7.13 – Número de Nusselt médio, a eficiência de deslocamento e a eficiência de deslocamento não estacionária para $0,1 \leq$ *Pl* $\leq 0,8$.

Devido à sua natureza, a influência de *Pl* apresenta tendências opostas à influência de *Re*: $\overline{Nu}$ e a eficiência de deslocamento são maiores para valores mais baixos de *Pl*. O aumento do *plastic number* aumenta as zonas aparentemente não escoadas, o que por definição diminui a eficiência de deslocamento. Além disso, essas zonas aparentemente não escoadas dificultam a troca de calor por convecção, diminuindo assim $\overline{Nu}$. Para $0,1 < Pl < 0,6$, é

possível identificar uma zona de recirculação dentro da cavidade, e para valores de *Pl* maiores que 0,6 a zona de recirculação não consegue superar a tensão limite de cisalhamento. A presença de zonas de recirculação também pode ser identificada na Fig. 7.14, que mostra os perfis de temperatura adimensional na direção do eixo y^* em $x^* = 0$. Além disso, a Fig. 7.14 também permite identificar a influência do aumento de *Pl* sobre o perfil de velocidade: ao dificultar a transferência de calor por convecção, a temperatura do platô formado na região de recirculação diminui com o aumento de *Pl*.

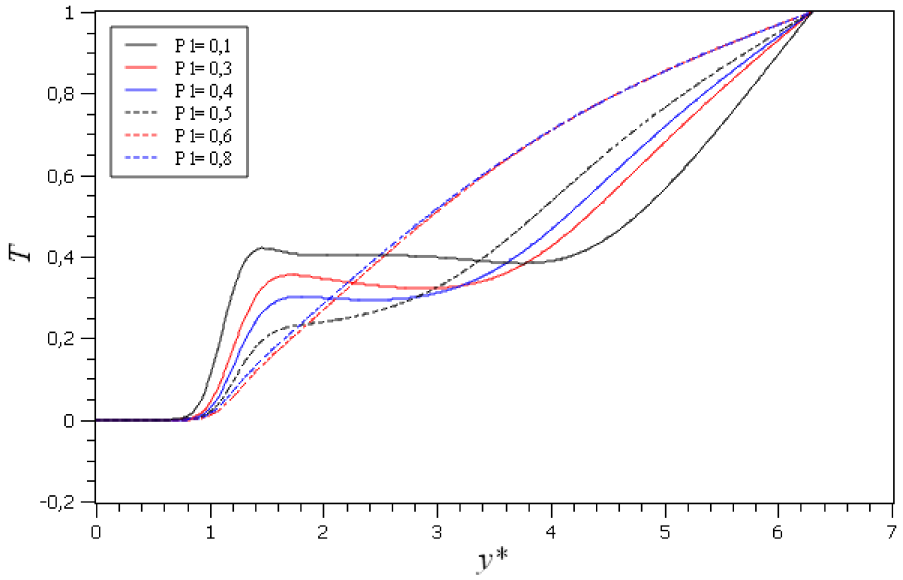
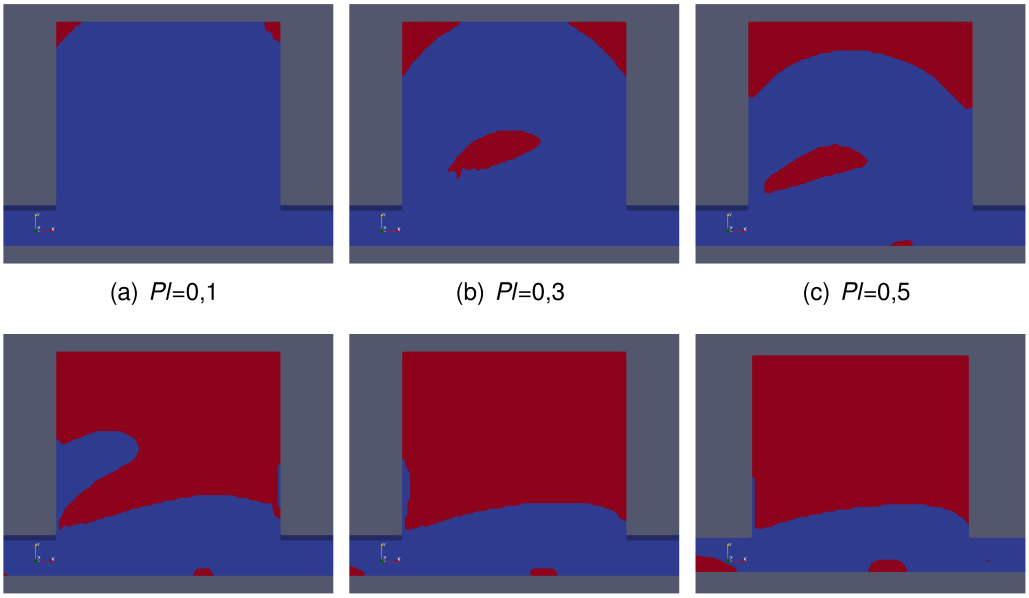


Figura 7.14 – Perfis de temperatura na direção do eixo y^* em $x^* = 0$ para 0,1≤ *Pl*≤0,8.

As zonas aparentemente não escoadas para diferentes valores de *Pl* estão representadas na Fig. 7.15. É possível identificar que a posição da zona aparentemente não escoada no centro da cavidade sofre menos influência de *Pl* do que de *Re*.

A Fig. 7.16 apresenta os resultados para Hl. Ao contrário de $\overline{Nu}$, Φ_{de} e Φ_{sta}, Hl mantém a tendência apresentada na análise de *Re*: Hl é positivo e alto para valores baixos de *Pl*. Com o aumento de *Pl*, Hl diminui, ainda que continue positivo. A queda de pressão é um fenômeno associado principalmente à interação entre as superfícies e o fluido. *Pl* é uma medida de quão forte é o comportamento viscoplástico, mas em geral na região próxima à parede do canal a taxa de deformação é alta o suficiente para superar τ_0. Isso pode ser visto na Fig. 7.15(f), que mostra as zonas aparentemente não escoadas para $Pl = 0,8$. Na figura, a eficiência de deslocamento está abaixo de 0,15, mas ainda assim uma parte considerável do canal é composta por uma zona escoada. Analisando a Eq.(7.3), é possível concluir que ao



(a) *Pl*=0,1 (b) *Pl*=0,3 (c) *Pl*=0,5

(d) *Pl*=0,6 (e) *Pl*=0,7 (f) *Pl*=0,8

Figura 7.15 – Zonas aparentemente não escoadas (em vermelho) e aparentemente escoadas (em azul) para (a) *Pl*=0,1; (b) *Pl*=0,3; (c) *Pl*=0,5; (d) *Pl*=0,6; (e) *Pl*=0,7; (f) *Pl*=0,8.

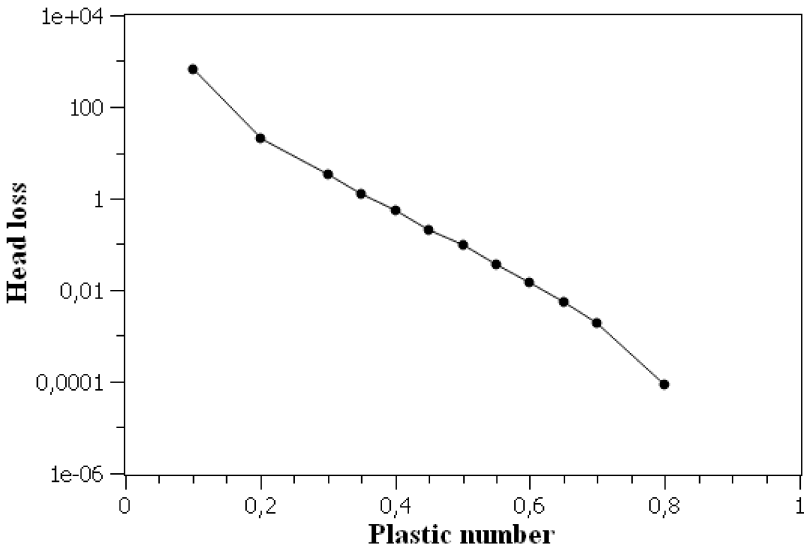


Figura 7.16 – Hl para $0,1 \leq Pl \leq 0,8$.

aumentar *Pl*, outros parâmetros do fluido tem sua relevância diminuída, como *K*, *n* e η_∞. Esses parâmetros governam a viscosidade na região próxima às paredes, o que significa que tanto Δp_s quanto Δp_c diminuem, o que também diminui a influência da cavidade, e assim Hl diminui. Além disso, o valor de $\dot{\gamma}_1$ também deve ser analisado. Como mostra a Eq.(7.5), $\dot{\gamma}_1$ aumenta quando τ_0 aumenta, e τ_0 é função direta de *Pl*. Dessa forma, esses dois mecanismos fazem

com o aumento de *Pl* faça com que Hl tenda a unidade. Esse resultado destaca que mesmo que *Pl* seja associado às zonas de alta viscosidade aparente do escoamento, na realidade ele diminui as forças viscosas nas paredes do canal.

7.8.3 Influência do índice de comportamento

Nesta seção, a influência do índice de comportamento n será discutida. O intervalo $0,3 \leq n \leq 1,2$ será adotado, enquanto os outros grupos adimensionais serão mantidos constantes e iguais ao conjunto padrão de adimensionais. Quando $n \geq 1$, η^*_∞ foi definido como nulo. A Fig. 2.9 mostra como este parâmetro influência a viscosidade adimensional.

Como já citado, a zona de recirculação fica evidente quando o escoamento consegue superar τ_0, que é um parâmetro principalmente associado ao número de Reynolds e ao *plastic number*. Como a Fig. 2.9 mostra, a principal influência de n ocorre na região onde $\dot{\gamma}$>$\dot{\gamma}_1$. Dessa forma, a variação de n tem apenas uma pequena influencia no surgimento da zona de recirculação. Devido aos valores de *Re* e *Pl* adotados, todas as simulações apresentam zonas de recirculação.

A Fig 7.17 apresenta os resultados para $\overline{Nu}$, Φ_{de} e Φ_{sta}. Como todas as simulações apresentam zonas de recirculação, os valores de Φ_{de} e Φ_{sta} apresentam uma variação menor do que nas simulações de *Pl* e *Re*. A Fig. 7.18 mostra o campo de tensão cisalhante adimensional para $n = 0,3$ e $n = 1,2$, de forma que as zonas escuras representam as regiões onde $\tau < \tau_0$, ou seja, são zonas aparentemente não escoadas. A Fig. 7.19 apresenta o campo de taxa de deformação adimensional para $n = 0,3$ e $n = 1,2$. Comparando as figuras apresentadas, é possível perceber que mesmo que uma parte considerável da cavidade apresente taxas de deformação acima de $\dot{\gamma}_{ref}$, em apenas uma pequena região próxima ao canal a taxa de deformação é maior que $\dot{\gamma}_1$. Quando este intervalo de $\dot{\gamma}_{ref}$ até $\dot{\gamma}_1$ é analisado na Fig. 2.9, é possível perceber que a influência de n nesta região ainda é pequena, o que explica a pequena variação de Φ_{de} e Φ_{sta}.

A Fig 7.17 também mostra o único resultado onde a eficiência de deslocamento e $\overline{Nu}$ apresentam tendências opostas. $\overline{Nu}$ aumenta com n, o que a primeira vista contraria a interpretação de que um n maior aumenta as forças viscosas, o que deveria dificultar a transferência de calor por convecção. Analisando a Fig. 7.20, é possível identificar a razão do aumento de $\overline{Nu}$: a velocidade da zona de recirculação é sensivelmente maior para valores de n maiores.

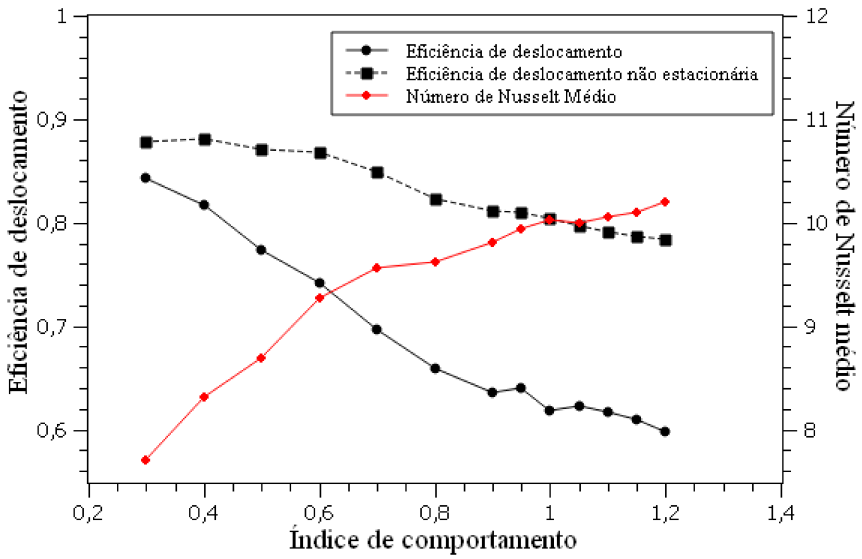
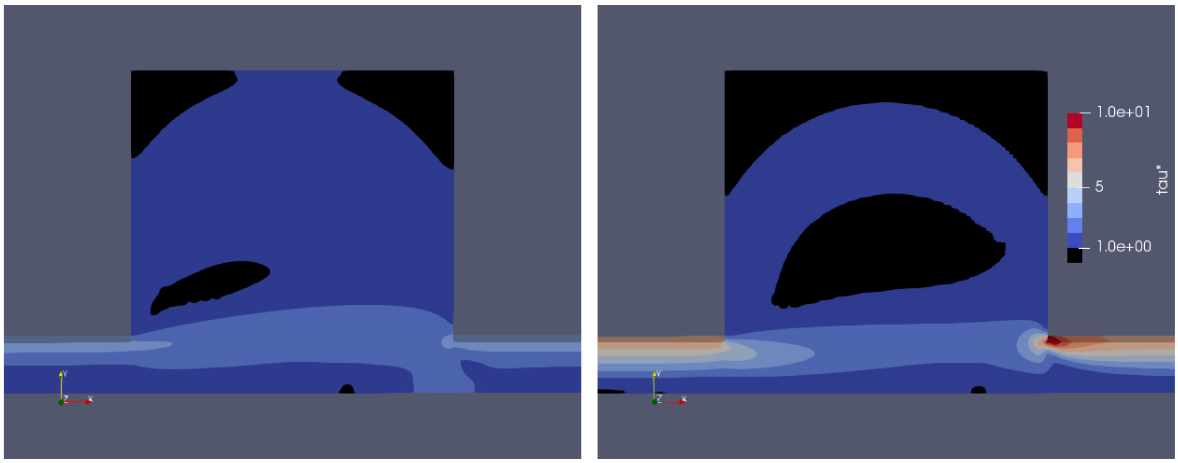


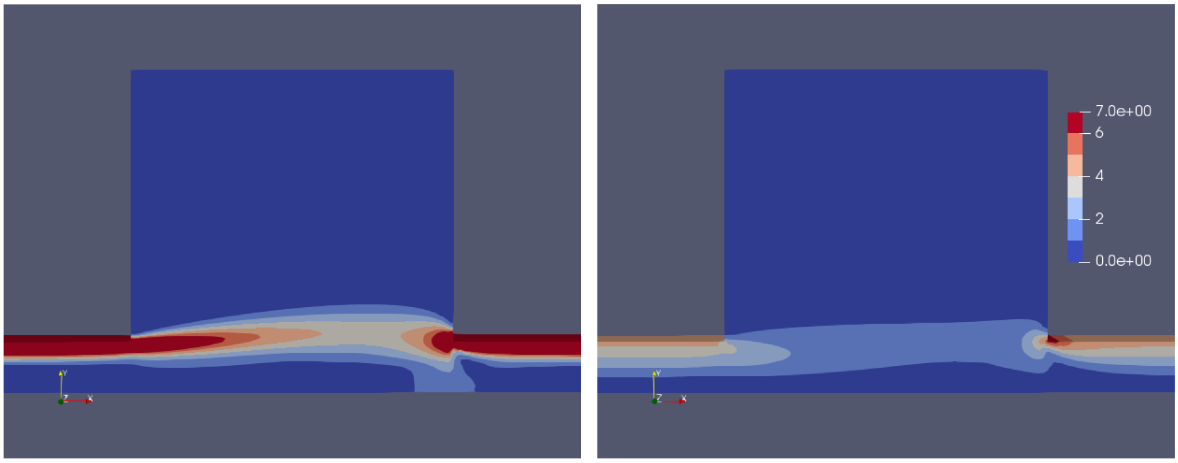
Figura 7.17 – Número de Nusselt médio, a eficiência de deslocamento e a eficiência de deslocamento não estacionária para 0,3≤ *n*≤1,2.



(a) *n*=0,3 (b) *n*=1,2

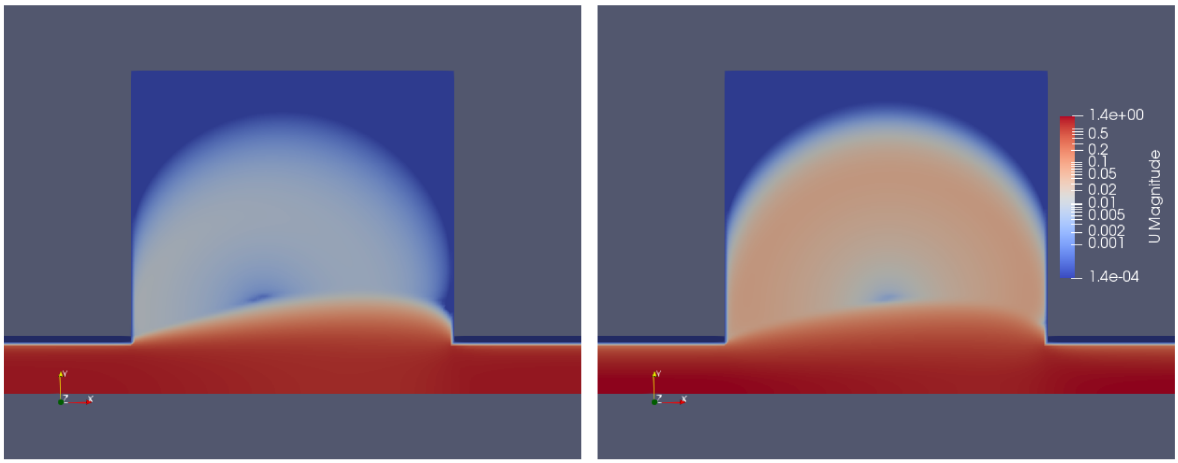
Figura 7.18 – Campo de tensão cisalhante adimensional, para (a) *n*=0,3 e (b) *n*=1,2. As duas figuras compartilham a mesma escala. Nas regiões escuras, $\tau < \tau_0$.

Essa intensificação do escoamento favorece a troca de calor por convecção, o que explica o aumento de $\overline{Nu}$. Para as regiões onde a taxa de deformação é maior que $\dot{\gamma}_1$, quando n é menor que a unidade, quanto maior a taxa de deformação, menor será a viscosidade, o que permite ao fluido escoar com mais facilidade, e assim favorece o aumento da taxa de deformação, formando assim um ciclo. Dessa forma, a taxa de deformação tende a ter valores maiores para n menores, como pode ser observado na Fig. 7.19. Quando n se aproxima da unidade, esse mecanismo perde força, e os valores da taxa de deformação tendem a variar menos. Como pode ser observado na Fig. 7.18, quando $n = 0,3$, a tensão cisalhante entre o escoamento



(a) *n*=0,3 (b) *n*=1,2

Figura 7.19 – Campo de taxa de deformação adimensional, para (a) *n*=0,3 e (b) *n*=1,2. As duas figuras compartilham a mesma escala.



(a) *n*=0,3 (b) *n*=1,2

Figura 7.20 – Magnitude da velocidade dentro da cavidade, para (a) *n*=0,3 e (b) *n*=1,2. As duas figuras compartilham a mesma escala.

e a parede do canal é dissipada logo depois que o escoamento entra na cavidade. Já para $n = 1,2$, a tensão cisalhante continua alta durante uma parte considerável do comprimento da cavidade. Esse comportamento indica que o fluido com n maior exerce uma tensão cisalhante maior sobre o fluido dentro da cavidade, acelerando a zona de recirculação. E como a taxa de deformação na cavidade é sempre menor que $\dot{\gamma}_1$, as forças viscosas dentro da cavidade são relativamente constantes em relação a n. Isso significa que quanto maior o valor de n, maior a quantidade de movimento linear que é transferida para a cavidade, mas a taxa de dissipação de movimento linear no interior da cavidade continua relativamente constante. Dessa forma, mesmo com o aumento das forças viscosas, a velocidade da zona de recirculação aumenta,

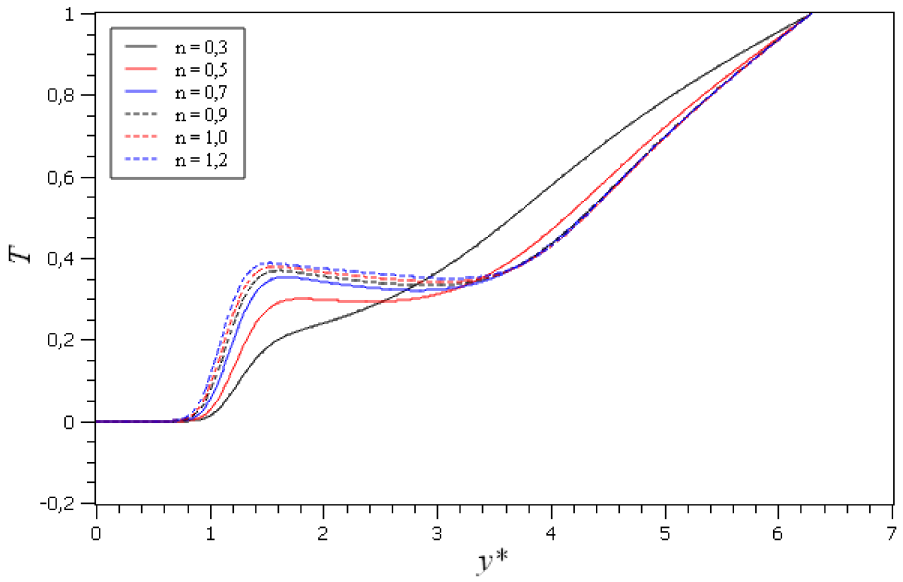


Figura 7.21 – Perfis de temperatura na direção do eixo y^* em $x^* = 0$ para 0,3$\leq$ *n*$\leq$1,2.

e com ela o valor de $\overline{Nu}$ também aumenta. A Fig. 7.21 confirma esse mecanismo: os perfis de temperatura no centro da cavidade mostram que existem zonas de recirculação em todos os valores de n estudados, mas também é possível identificar que quanto menor o n, menos evidente é o platô de temperatura provocado pela zona de recirculação, mesmo mantendo *Re* e *Pl* constantes.

Os resultados para Hl são apresentados na Fig. 7.22. Ao contrário do que acontece nos outros parâmetros estudados, Hl não apresenta uma tendência única. Dos três parâmetros analisados, apenas o n é um parâmetro do fluido, e não um parâmetro do escoamento, de acordo com as definição adotadas de Re e Pl. Outra característica que o diferencia dos demais é que ele influencia todos os parâmetros do fluido, normalmente na forma de um expoente de algum termo. Além disso, Hl é o único parâmetro de resultado onde a adimensionalização é feita com propriedades do fluido (ρ e $\dot{\gamma}_1$), e não com propriedades particulares do escoamento, como a área da cavidade, um gradiente de temperatura imposto ou um comprimento característico. Dessa forma, a influência de n sobre os parâmetros do fluido e a dependência de Hl em relação à esses mesmos parâmetros induzem que Hl apresente um comportamento mais complexo em relação à variação de n, em especial na metodologia adotada neste trabalho, onde os outros grupos adimensionais são mantidos constantes mesmo eles sendo função do índice de comportamento.

Como já discutido, a variação de n não altera apenas a relação do fluido com as super-

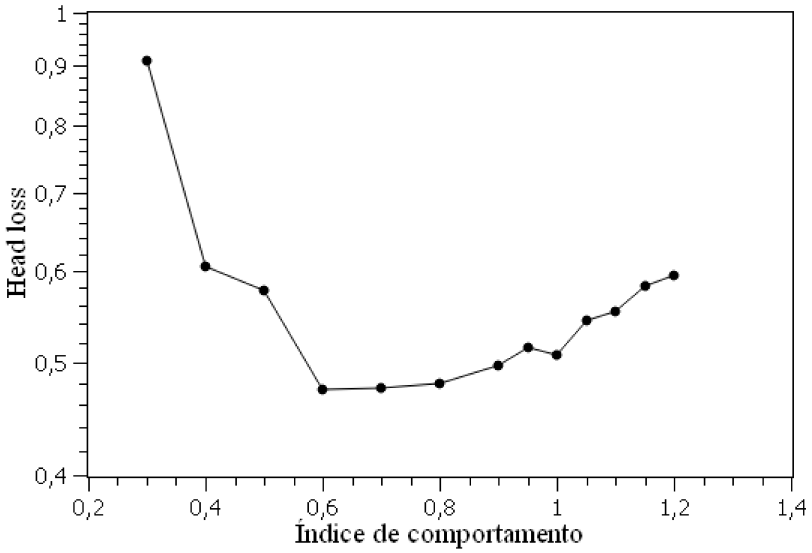


Figura 7.22 – Hl para 0.3$\leq$ *n*$\leq$1.2.

fícies, mas também altera como a cavidade exerce sua resistência ao escoamento. Dessa forma, mesmo que o aumento de n aumente as forças viscosas nas paredes, ele também aumenta a resistência ao escoamento exercida pela cavidade, e assim a diferença entre Δp_s e Δp_c não aumenta de forma tão intensa com o aumento das forças viscosas, como aconteceu na análise do outros parâmetros. Assim, com o aumento de n, a diferença entre Δp_s e Δp_c aumenta, mas de maneira mais comedida.

Comparando as curvas para $n = 0,3$ e $n = 0,7$ da Fig. 2.9, é possível perceber que o platô η_∞ é alcançado com taxas de deformação menores quando n é menor, o que significa que $\dot{\gamma}_2$ é relativamente menor para n menores quando os outros grupos adimensionais que definem as propriedades do fluido são mantidos constantes. Na realidade, todas as três taxas de deformação de transição apresentam uma correlação positiva em relação a n, porém esse comportamento é mais difícil de ser reconhecido na Fig. 2.9 devido à adimensionalização escolhida.

Assim, Hl pode ser interpretado como a soma de dois mecanismos diferentes: a diferença entre Δp_s e Δp_c aumenta com o aumento de n, mas em um ritmo menor do que o observado em outros parâmetros. Além disso, $\dot{\gamma}_1$ também aumenta com o aumento de n, mas com um comportamento exponencial: quando n é pequeno, $\dot{\gamma}_1$ também apresenta valores mais baixos e uma grande taxa de crescimento. Com o aumento de n, $\dot{\gamma}_1$ diminui sua taxa de crescimento. Assim, os altos valores de Hl para valores mais baixos de n são explicados pelos valores mais

baixos de $\dot{\gamma}_1$. Com o aumento de n, em um primeiro momento a taxa de crescimento de $\dot{\gamma}_1$ provoca uma queda nos valores de Hl, só que esta taxa de crescimento também diminui com o aumento de n, até o momento onde o crescimento da diferença entre Δp_s e Δp_c se torna mais relevante que o crescimento de $\dot{\gamma}_1$, e assim Hl volta a crescer.

Para tentar contornar essa dificuldade imposta pela adimensionalização de Hl, um novo parâmetro é apresentado na Fig. 7.23: $\Delta p_c/\Delta p_s$. Além da diferente metodologia de adimensionalização, outra característica desse parâmetro é que o seu valor numérico é função da distância entre os pontos usados para o cálculo de Δp_s e Δp_c: quanto maior a distância, mais $\Delta p_c/\Delta p_s$ se aproxima da unidade. Dessa forma, esse parâmetro é utilizado apenas como uma referência de comparação para o comportamento da queda de pressão. A Fig. 7.23 mostra que para todo o intervalo analisado, o aumento de n aumenta a diferença entre Δp_s e Δp_c.

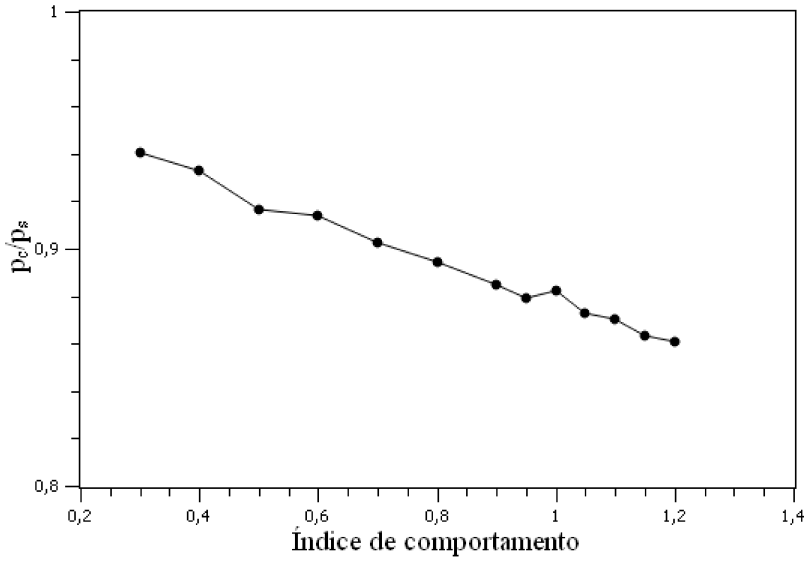


Figura 7.23 – $\Delta p_c/\Delta p_s$ para $0,3 \leq n \leq 1,2$.

CAPÍTULO VIII

CONCLUSÃO

Este trabalho apresentou o processo de desenvolvimento, validação e aplicação de uma rotina no software OpenFOAM® capaz de resolver escoamentos laminares de não-isotérmicos de fluidos não newtonianos incompressíveis em regime permanente. Foi adotada a hipótese de fluido newtoniano generalizado e a aproximação de Boussinesq. O modelo mecânico desse *solver* é composto pelas equações de balanço de massa, de quantidade de movimento linear e de energia, além de uma equação constitutiva característica do modelo de viscosidade aparente adotado.

Em um primeiro momento, este trabalho buscou apresentar os princípios teóricos que sustentam os códigos. Uma revisão bibliográfica sobre os principais modelos de viscosidade independente do tempo foi apresentada, e trabalhos de simulação numérica que adotam essa modelagem são usados para se apresentar o estado da arte desse campo de estudo. Em seguida, uma revisão bibliográfica sobre os principais métodos numéricos, seus componentes e propriedades é apresentada.

Uma vez estabelecidas as bases teóricas, este trabalho apresentou o procedimento de elaboração e compilação das funções no OpenFOAM®. Três funções são apresentadas: uma rotina *solver* e duas funções para modelar a viscosidade aparente, acrescentando o modelo SMD e o biviscosidade ao pacote de funções do OpenFOAM®. Em seguida, as etapas de pré-processamento e pós-processamento são discutidas, com o objetivo de justificar as escolhas realizadas, facilitando assim a reprodução e ampliação dos resultados apresentados neste trabalho.

Após implementar as funções, este trabalho buscou veridicar a validade delas, compa-

rando os resultados obtidos com os resultados apresentados por Turan, Chakraborty e Poole (2010) e Turan et al. (2011). Para realizar as comparações numéricas, um *software* de análise de imagem foi utilizado para converter os perfis apresentados nos trabalhos adotados como referência em valores numéricos. Foram comparados resultados locais e globais, adotando o modelo *power-law* e o modelo de Bingham. Mesmo com o erro introduzido pelo *software* de análise de imagem, o coeficiente de determinação (r^2) ficou sempre acima de 0,95, e grande maioria dos perfis analisados apresentaram $r^2 > 0,99$. Além disso, r^2 apresentou uma melhora sensível em comparações onde as condições para o *software* de análise de imagem eram mais favoráveis, o que indica que esta era a principal fonte de erros na comparação entre os resultados obtidos e o resultados adotados como referência.

Em seguida, este trabalho apresentou um estudo de caso onde eles foram aplicados. A geometria adotada foi o canal planar com uma expansão seguida de uma contração, formando assim uma cavidade. O modelo de viscosidade aparente adotado foi o SMD. O escoamento foi definido por seis grupos adimensionais: o número de Reynolds, o número de Pradlt, o *plastic number*, o *Jump number*, o índice de comportamento e o platô de viscosidade adimensional para altas taxas de deformação. Destes, foi analisada a influência de três sobre o escoamento: o número de Reynolds, o *plastic number* e o índice de comportamento. Para se avaliar influência deles, os resultados foram apresentados em termos do número de Nusselt médio, da eficiência de deslocamento e do *head loss*.

Cada um dos parâmetros foi analisado individualmente partindo de um conjunto padrão de adimensionais. Em diversas simulações, foi possível observar uma zona de recirculação dentro da cavidade. Essa zona de recirculação apresentou influência significativa sobre o número de Nusselt médio e a eficiência de deslocamento, mas o parâmetro *head loss* não apresentou qualquer indício de influência da zona de recirculação, o que pode ser explicado pelo fato que os parâmetros do escoamento usados no cálculo do *head loss* são medidos distante da cavidade.

Na análise da influência do número de Reynolds, foi observada uma correlação positiva do número de Nusselt médio e da eficiência de deslocamento em relação a esse parâmetro. O *head loss* apresenta uma correlação negativa: quanto maior o número de Reynolds, menor o valor do *head loss*. Já a análise do *plastic number* apresentou uma correlação negativa para os três parâmetros avaliados. Além disso, a maior variação do *head loss* foi observada na aná-

lise do intervalo proposto do *plastic number*. Por fim, a análise de índice de comportamento apresentou tendências opostas para o número de Nusselt médio e a eficiência de deslocamento: o número de Nusselt médio aumentou com o aumento do índice de comportamento, mas a eficiência de deslocamento diminuiu. O aumento do número de Nusselt médio pode ser explicado pela intensificação do fluxo da zona de recirculação dentro da cavidade, provocada pela maior tensão cisalhante na interface entre o escoamento na cavidade e o escoamento no canal. O *head loss* na análise do índice de comportamento foi a única curva que apresentou mais de uma tendência, e esse comportamento mais complexo pode ser explicado pela influência que o índice de comportamento exerce sobre os parâmetros do fluido, inclusive sobre os parâmetros usados para adimensionalizar o *head loss*.

8.1 Trabalhos futuros

Como um dos objetivos desse trabalho é oferecer as bases para outros trabalhos de simulação numérica de escoamentos não newtonianos usando o OpenFOAM®, é possível identificar uma série de frentes de trabalho para trabalhos futuros, como:

- A implementação de outros modelos de viscosidade aparente, inclusive modelando comportamentos como viscoelasticidade e a viscosidade aparente sendo uma função do tempo e da temperatura.
- A implementação de *solvers* para outros tipos de escoamento que não somente os escoamentos internos, como os escoamentos com superfície livre e os escoamentos bifásicos, por exemplo.
- A implementação e otimização dos parâmetros numéricos, como o uso de esquemas de interpolação de ordens superiores ou a substituição do método SIMPLE por outro mais adequado para escoamentos não newtonianos.
- A adição de outros fenômenos nas simulações, como a convecção natural no escoamento do canal planar com expansão-contração, e a sua influência nas zonas aparentemente não cisalhadas estacionárias em função da posição do duto em relação à gravidade, por exemplo.

- O aumento da complexidade da geometria adotada, inclusive usando outros *softwares* para a geração da malha. Um exemplo é a solução de escoamentos tridimensionais, como o canal axissimétrico apresentado na Fig. 8.1.

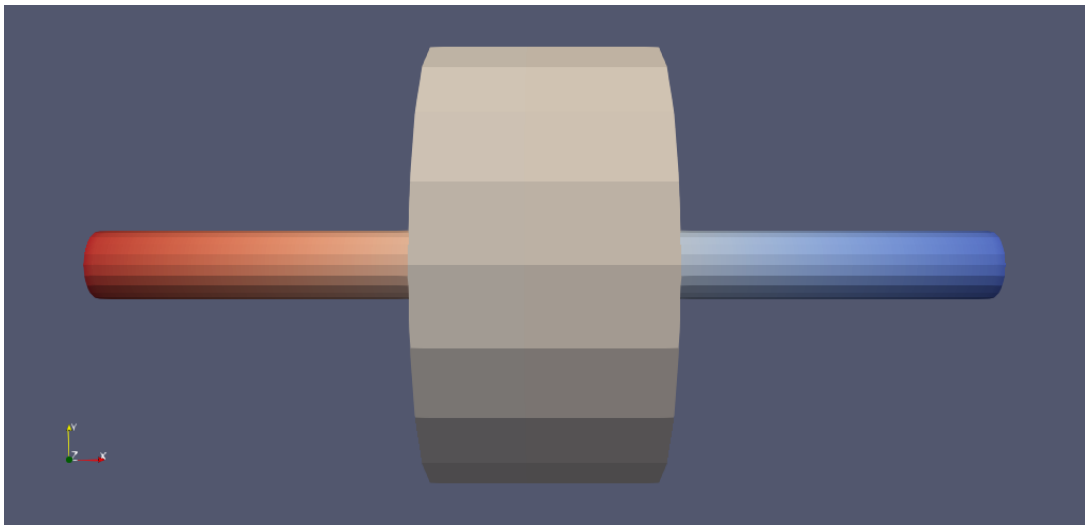


Figura 8.1 – Canal axissimétrico tridimensional com uma expansão seguida de uma contração.

- Desenvolvimento de correlações entre os grupos adimensionais, aplicando técnicas de planejamento de experimentos. Por exemplo: uma correlação que relaciona como o número de Nusselt médio se comporta em função do número de Reynolds, do *plastic number* e do índice de comportamento.

CAPÍTULO IX

REFERÊNCIA BIBLIOGRÁFICA

ALEGRIA, Luis Miguel Casanova. **Soluções analíticas e numéricas para o escoamento laminar desenvolvido de fluído viscoplástico em dutos e anulares elípticos**. Dissertação (Mestrado) — Universidade Tecnológica Federal do Paraná, 2011.

BICALHO, Isabele Cristina. **Estudo experimental e de simulação por CFD de escoamentos em seções anulares com excentricidade variável e obstrução parcial da coluna**. Tese (Doutorado) — Universidade Federal de Uberlândia, 2015.

BIJJAM, Sudheer; DHIMAN, Amit; GAUTAM, Vandana. Laminar momentum and heat transfer phenomena of power-law dilatant fluids around an asymmetrically confined cylinder. **International Journal of Thermal Sciences**, Elsevier, v. 88, p. 110–127, 2015. Disponível em: <https://doi.org/10.1016/j.ijthermalsci.2014.09.015>.

BOGER, David V. Demonstration of upper and lower newtonian fluid behaviour in a pseudoplastic fluid. **Nature**, Nature Publishing Group, v. 265, n. 5590, p. 126, 1977. Disponível em: <https://doi.org/10.1038/265126a0>.

BOUSSINESQ, Joseph. **Théorie analytique de la chaleur**. [S.l.]: Gauthier-Villars, 1903.

CELIK, Ismail B. Procedure for estimation and reporting of discretization error in cfd applications. **Journal of Fluids Engineering**, v. 130, n. 7, p. 078001, 2008. Disponível em: <https://doi.org/10.1115/1.2960953>.

CHHABRA, RP; RICHARDSON, John Francis. **Non-Newtonian Flow in the Process Industries: Fundamentals and Engineering Applications**. [S.l.]: Butterworth-Heinemann, 1999.

de Souza Mendes, Paulo Roberto. Dimensionless non-Newtonian fluid mechanics. **Journal of Non-Newtonian Fluid Mechanics**, v. 147, p. 109–116, 2007a. Disponível em: <https://doi.org/10.1016/j.jnnfm.2006.09.007>.

de Souza Mendes, Paulo Roberto; DUTRA, Eduardo S. S. Viscosity function for yield-stress liquids. **Applied Rheology**, v. 14, p. 296–302, 2004. Disponível em: <https://doi.org/10.1515/arh-2004-0016>.

de Souza Mendes, P. Roberto et al. Flow of viscoplastic liquids through axisymmetric expansions-contractions. **Journal of Non-Newtonian Fluid Mechanics**, v. 142, p. 207–217, 2007b. Disponível em: <https://doi.org/10.1016/j.jnnfm.2007.07.010>.

DOORMAAL, JP Van; RAITHBY, GD. Enhancements of the simple method for predicting incompressible fluid flows. **Numerical heat transfer**, Taylor & Francis, v. 7, n. 2, p. 147–163, 1984. Disponível em: <https://doi.org/10.1080/01495728408961817>.

FAVERO, J L et al. Viscoelastic fluid analysis in internal and in free surface flows using the software openfoam. **Computers & chemical engineering**, Elsevier, v. 34, n. 12, p. 1984–1993, 2010. Disponível em: <https://doi.org/10.1016/j.compchemeng.2010.07.010>.

FERZIGER, Joel H; PERIC, Milovan. **Computational methods for fluid dynamics**. Springer Science & Business Media, 2002. Disponível em: <https://doi.org/10.1007/978-3-642-56026-2>.

GREENSHIELDS, Christopher J. Openfoam programmer's guide. **OpenFOAM Foundation**, 2015.

GREENSHIELDS, Christopher J. Openfoam user guide. **OpenFOAM Foundation**, 2018.

GUERRERO, Joel. 2018 summer session - unige introductory openfoam training. **University of Genoa. DICCA.**, July 2018.

HABLA, Florian et al. Numerical simulation of viscoelastic two-phase flows using openfoam®. **Chemical engineering science**, Elsevier, v. 66, n. 22, p. 5487–5496, 2011. Disponível em: <https://doi.org/10.1016/j.ces.2011.06.076>.

HERMANY, Lober et al. Flow of yield-stress liquids through an axisymmetric abrupt expansion-contraction. **Journal of Non-Newtonian Fluid Mechanics**, Elsevier, v. 201, p. 1–9, 2013. Disponível em: <https://doi.org/10.1016/j.jnnfm.2013.07.002>.

HIGUERA, Pablo; LARA, Javier L; LOSADA, Inigo J. Simulating coastal engineering processes with openfoam®. **Coastal Engineering**, Elsevier, v. 71, p. 119–134, 2013. Disponível em: <https://doi.org/10.1016/j.coastaleng.2012.06.002>.

HOLZINGER, Gerhard. Openfoam–a little user-manual. **CD-Laboratory-Particulate Flow Modelling Johannes Keplper University, Linz, Austria**, 2018.

HOLZMANN, Tobias. **Mathematics, Numerics, Derivations and OpenFOAM®**. [S.l.: s.n.], 2016. 145 p.

ISSA, Raad I. Solution of the implicitly discretised fluid flow equations by operator-splitting. **Journal of computational physics**, Elsevier, v. 62, n. 1, p. 40–65, 1985. Disponível em: <https://doi.org/10.1016/0021-9991(86)90099-9>.

JUNIOR, R G Egres et al. Stab resistance of shear thickening fluid (stf)–kevlar composites for body armor applications. In: **Transformational Science And Technology For The Current And Future Force: (With CD-ROM)**. [S.l.]: World Scientific, 2006. p. 264–271.

KELESSIDIS, VC et al. Optimal determination of rheological parameters for herschel–bulkley drilling fluids and impact on pressure drop, velocity profiles and penetration rates during drilling. **Journal of Petroleum Science and Engineering**, Elsevier, v. 53, n. 3-4, p. 203–224, 2006. Disponível em: <https://doi.org/10.1016/j.petrol.2006.06.004>.

KUMAR, Anuj; DHIMAN, Amit; BARANYI, László. Cfd analysis of power-law fluid flow and heat transfer around a confined semi-circular cylinder. **International Journal of Heat and Mass Transfer**, Elsevier, v. 82, p. 159–169, 2015. Disponível em: <https://doi.org/10.1016/j.ijheatmasstransfer.2014.11.046>.

NACCACHE, Mônica F; BARBOSA, Rafael S. Creeping flow of viscoplastic materials through a planar expansion followed by a contraction. **Mechanics Research Communications**, Elsevier, v. 34, n. 5-6, p. 423–431, 2007. Disponível em: <https://doi.org/10.1016/j.mechrescom.2007.06.003>.

NALLURI, SV; PATEL, SA; CHHABRA, RP. Mixed convection from a hemisphere in bingham plastic fluids. **International Journal of Heat and Mass Transfer**, Elsevier, v. 84, p. 304–318, 2015. Disponível em: <https://doi.org/10.1016/j.ijheatmasstransfer.2014.12.059>.

NIRMALKAR, N; BOSE, A; CHHABRA, RP. Free convection from a heated circular cylinder in bingham plastic fluids. **International Journal of Thermal Sciences**, Elsevier, v. 83, p. 33–44, 2014. Disponível em: <https://doi.org/10.1016/j.ijthermalsci.2014.04.004>.

NOUAR, C; DESAUBRY, C; ZENAIDI, H. Numerical and experimental investigation of thermal convection for a thermodependent herschel-bulkley fluid in an annular duct with rotating inner cylinder. **European Journal of Mechanics-B/Fluids**, Elsevier, v. 17, n. 6, p. 875–900, 1998. Disponível em: <https://doi.org/10.1016/S0997-7546(99)80018-1>.

O'DONOVAN, EJ; TANNER, RI. Numerical study of the bingham squeeze film problem. **Journal of Non-Newtonian Fluid Mechanics**, Elsevier, v. 15, n. 1, p. 75–83, 1984. Disponível em: <https://doi.org/10.1016/0377-0257(84)80029-4>.

PAPANASTASIOU, Tasos C. Flow of materials with yield. **Journal of Rheology**, v. 31, p. 385–404, 1987. Disponível em: <https://doi.org/10.1122/1.549926>.

PATANKAR, Suhas. **Numerical heat transfer and fluid flow**. [S.l.]: CRC press, 1980.

PATANKAR, S V; SPALDING, D B. A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. **International Journal Heat Mass Transfer**, v. 15, p. 1787–1806, 1972. Disponível em: <https://doi.org/10.1016/0017-9310(72)90054-3>.

RAMOS, GG; SOARES, EJ; THOMPSON, RL. The yield zone concept and its application on a 4: 1 abrupt contraction for an apparent-yield-stress fluid. **Latin American Applied Research**, v. 43, p. 363–367, 2013.

ROUSTAEI, A; FRIGAARD, IA. The occurrence of fouling layers in the flow of a yield stress fluid along a wavy-walled channel. **Journal of Non-Newtonian Fluid Mechanics**, Elsevier, v. 198, p. 109–124, 2013. Disponível em: <https://doi.org/10.1016/j.jnnfm.2013.03.005>.

SANTOS, D. D. et al. Numerical investigation of elastic and viscous effects on inertial viscoplastic fluid flows. In: **Proceedings of the 22nd International Congress of Mechanical Engineering - COBEM 2013**. Ribeirão Preto, Brazil: [s.n.], 2013.

SANTOS, Daniel Dall'Onder dos. **Modelagem mecânica e investigação numérica de escoamentos de fluidos SMD empregando um método multi-campos de galerkin mínimos-quadrados**. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Sul, 2010.

SARAMITO, Pierre. **Complex fluids**. Springer, 2016. 255 p. Disponível em: <https://doi.org/10.1007/978-3-319-44362-1>.

THOMPSON, Roney Leon; SOARES, Edson José. Viscoplastic dimensionless numbers. **Journal of Non-Newtonian Fluid Mechanics**, v. 238, p. 57–64, 2016. Disponível em: <https://doi.org/10.1016/j.jnnfm.2016.05.001>.

TURAN, O.; CHAKRABORTY, N.; POOLE, N. R. J. Laminar natural convection of Bingham fluids in a square enclosure with differentially heated side walls. **Journal of Non-Newtonian Fluid Mechanics**, v. 165, p. 901–913, 2010. Disponível em: <https://doi.org/10.1016/j.jnnfm.2010.04.013>.

TURAN, Osman et al. Laminar natural convection of power-law fluids in a square enclosure with differentially heated side walls subjected to constant temperatures. **Journal of Non-Newtonian Fluid Mechanics**, Elsevier, v. 166, n. 17-18, p. 1049–1063, 2011. Disponível em: <https://doi.org/10.1016/j.jnnfm.2011.06.003>.

WHITE, Frank M. **Mecânica dos fluidos 4ª Edição**. [S.l.]: McGraw Hill Brasil, 2002.

CAPÍTULO X

APÊNDICES E ANEXOS

10.1 APÊNDICE I - nonNewtonianSimpleFoam

10.1.1 nonNewtonianSimpleFoam.C

```
/*---------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2016 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------------
Application
    nonNewtonianSimpleFoam

Description
    Steady-state solver for incompressible flow of non Newtonian
    fluids, using the SIMPLE algorithm.

\*---------------------------------------------------------------------------*/

#include "fvCFD.H"
#include "singlePhaseTransportModel.H"
#include "viscosityModel.H"
#include "simpleControl.H"
#include "fvOptions.H"

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

int main(int argc, char *argv[])
{
```

```
27      #include "postProcess.H"
28      #include "setRootCase.H"
29      #include "createTime.H"
30      #include "createMesh.H"
31      #include "createControl.H"
32      #include "createFields.H"
33      #include "createFvOptions.H"
34      #include "initContinuityErrs.H"
35
36      // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
37
38      Info<< "\nStarting time loop\n" << endl;
39
40      while (simple.loop())
41      {
42          Info<< "Time = " << runTime.timeName() << nl << endl;
43
44          // --- Pressure-velocity SIMPLE corrector
45          {
46              #include "UEqn.H"
47              #include "pEqn.H"
48              #include "TEqn.H"
49                      fluid.correct();
50           }
51           scalarDict.set("tempo", ti+runTime.elapsedCpuTime());
52           singlePhaseTransportModel fluid(U, phi);
53
54          runTime.write();
55
56          Info<< "ExecutionTime = " << runTime.elapsedCpuTime() << " s"
57              << "  ClockTime = " << runTime.elapsedClockTime() << " s"
58              << nl << endl;
59      }
60
61      Info<< "End\n" << endl;
62
63      return 0;
64  }
65
66  // ************************************************************************* //
```

10.1.2 TEqn.H

```
fvScalarMatrix TEqn
        (
        fvm::div(phi, T)
        == fvm::laplacian(alpha_, T) );

        TEqn.relax();
        TEqn.solve();
```

10.1.3 UEqn.H

```
// Momentum predictor


tmp<fvVectorMatrix> tUEqn
    (
           rho_.value()*fvm::div(phi, U)
          - fvm::laplacian(fluid.nu(), U)
          - (fvc::grad(U) & fvc::grad(fluid.nu()))
          - rho_.value()*g*beta_*T

    );
    fvVectorMatrix& UEqn = tUEqn.ref();

    UEqn.relax();

    fvOptions.constrain(UEqn);



    if (simple.momentumPredictor())
    {
        solve(UEqn == -fvc::grad(p));

        fvOptions.correct(U);
    }
```

10.1.4 pEqn.H

```
{
    volScalarField rAU(1.0/UEqn.A());
    volVectorField HbyA(constrainHbyA(rAU*UEqn.H(), U, p));
    surfaceScalarField phiHbyA("phiHbyA", fvc::flux(HbyA));
    adjustPhi(phiHbyA, U, p);

    tmp<volScalarField> rAtU(rAU);

    if (simple.consistent())
    {
        rAtU = 1.0/(1.0/rAU - UEqn.H1());
        phiHbyA +=
            fvc::interpolate(rAtU() - rAU)*fvc::snGrad(p)*mesh.magSf();
        HbyA -= (rAU - rAtU())*fvc::grad(p);
    }
    tUEqn.clear();

    // Update the pressure BCs to ensure flux consistency
    constrainPressure(p, U, phiHbyA, rAtU());

    // Non-orthogonal pressure corrector loop
    while (simple.correctNonOrthogonal())
    {
        fvScalarMatrix pEqn
        (
            fvm::laplacian(rAtU(), p) == fvc::div(phiHbyA)
        );
        pEqn.setReference(pRefCell, pRefValue);

        pEqn.solve();

        if (simple.finalNonOrthogonalIter())
        {
            phi = phiHbyA - pEqn.flux();
        }
    }
    #include "continuityErrs.H"

    // Explicitly relax pressure for momentum corrector
    p.relax();

    // Momentum corrector
    U = HbyA - rAtU()*fvc::grad(p);
    U.correctBoundaryConditions();
    fvOptions.correct(U);
}
```

10.1.5 createFields.H

```
Info<< "Reading field p\n" << endl;
volScalarField p
(
    IOobject
    (
        "p",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);

Info<< "Reading field U\n" << endl;
volVectorField U
(
    IOobject
    (
        "U",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);
Info<< "Reading field T\n" << endl;
volScalarField T
(
    IOobject
    (
        "T",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);

IOdictionary scalarDict
(
    IOobject
    (
        "scalarDict",
        runTime.timeName(),
```

```
48          mesh,
49          IOobject::MUST_READ,
50          IOobject::AUTO_WRITE
51      )
52  );
53
54  volScalarField    sr
55  (
56      IOobject
57      (
58          "sr",
59          runTime.timeName(),
60          mesh,
61          IOobject::NO_READ,
62          IOobject::AUTO_WRITE
63      ),
64      mesh
65  );
66
67  volScalarField    tau
68  (
69      IOobject
70      (
71          "tau",
72          runTime.timeName(),
73          mesh,
74          IOobject::NO_READ,
75          IOobject::AUTO_WRITE
76      ),
77      mesh
78  );
79
80  volScalarField    yieldGama
81  (
82      IOobject
83      (
84          "yieldGama",
85          runTime.timeName(),
86          mesh,
87          IOobject::NO_READ,
88          IOobject::AUTO_WRITE
89      ),
90      mesh
91  );
92
93  volScalarField    yieldGama2
94  (
95      IOobject
96      (
97          "yieldGama2",
```

```
98          runTime.timeName(),
99          mesh,
100         IOobject::NO_READ,
101         IOobject::AUTO_WRITE
102     ),
103     mesh
104 );
105 
106 #include "createPhi.H"
107 
108 
109 label pRefCell = 0;
110 scalar pRefValue = 0.0;
111 setRefCell(p, simple.dict(), pRefCell, pRefValue);
112 mesh.setFluxRequired(p.name());
113 
114 
115 singlePhaseTransportModel fluid(U, phi);
116 dimensionedVector g("g",dimAcceleration,vector(0,-1,0));
117 
118 dimensionedScalar alpha_(fluid.lookup("alpha"));
119 dimensionedScalar beta_(fluid.lookup("beta"));
120 
121 dimensionedScalar rho_(fluid.lookup("rho"));
122 
123 scalar ti(readScalar(scalarDict.lookup("tempo")));
124 scalar ratio_(readScalar(fluid.lookup("ratio")));
125 dimensionedScalar gama_(fluid.lookup("gamaRef"));
```

10.1.6 files

```
nonNewtonianSimpleFoam.C

EXE = $(FOAM_USER_APPBIN)/nonNewtonianSimpleFoam
```

10.1.7 options

```
EXE_INC = \
    -I$(LIB_SRC)/transportModels \
    -I$(LIB_SRC)/transportModels/incompressible\
/singlePhaseTransportModel \
    -I$(LIB_SRC)/transportModels/incompressible\
/viscosityModels/viscosityModel \
    -I$(LIB_SRC)/finiteVolume/lnInclude \
    -I$(LIB_SRC)/meshTools/lnInclude \
    -I$(LIB_SRC)/sampling/lnInclude \
    -L$(FOAM_USER_LIBBIN)

EXE_LIBS = \
    -lincompressibleTransportModels \
    -lfiniteVolume \
    -lmeshTools \
    -lfvOptions \
    -lsampling
```

10.2 APÊNDICE II - Funções biViscosity e SMD

10.2.1 biViscosity.C

```
/*---------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2016 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------------


#include "biViscosity.H"
#include "addToRunTimeSelectionTable.H"
#include "surfaceFields.H"

// * * * * * * * * * * * * * * * Static Data Members * * * * * * * * * * * //

namespace Foam
{
namespace viscosityModels
{
    defineTypeNameAndDebug(biViscosity, 0);

    addToRunTimeSelectionTable
    (
        viscosityModel,
        biViscosity,
        dictionary
    );
}
}


// * * * * * * * * * * * * Private Member Functions  * * * * * * * * * * * //

Foam::tmp<Foam::volScalarField>
Foam::viscosityModels::biViscosity::calcNu() const
{
tmp<volScalarField> sr(strainRate());

return
(
    min
    (
        eta0_,
        (tau0_*(1-etaP_/eta0_) + etaP_*sr())
        /(max
```

```
46              (
47              sr(),
48              dimensionedScalar ("VSMALL", dimless/dimTime, VSMALL)
49              )
50          )
51      )
52  );
53  }
54
55
56  // * * * * * * * * * * * * * * Constructors  * * * * * * * * * * * * * * //
57
58  Foam::viscosityModels::biViscosity::biViscosity
59  (
60      const word& name,
61      const dictionary& viscosityProperties,
62      const volVectorField& U,
63      const surfaceScalarField& phi
64  )
65  :
66      viscosityModel(name, viscosityProperties, U, phi),
67      biViscosityCoeffs_
68      (viscosityProperties.optionalSubDict(typeName + "Coeffs")),
69      eta0_("eta0", dimViscosity, biViscosityCoeffs_),
70      tau0_("tau0", dimViscosity/dimTime, biViscosityCoeffs_),
71      etaP_("etaP", dimViscosity, biViscosityCoeffs_),
72      nu_
73      (
74          IOobject
75          (
76              name,
77              U_.time().timeName(),
78              U_.db(),
79              IOobject::NO_READ,
80              IOobject::AUTO_WRITE
81          ),
82          calcNu()
83      )
84  {}
85
86
87  // * * * * * * * * * * * * * * Member Functions  * * * * * * * * * * * * * * //
88
89  bool Foam::viscosityModels::biViscosity::read
90  (
91      const dictionary& viscosityProperties
92  )
93  {
94      viscosityModel::read(viscosityProperties);
95
```

```
96      biViscosityCoeffs_ =
97          viscosityProperties.optionalSubDict(typeName + "Coeffs");
98
99      biViscosityCoeffs_.lookup("eta0") >> eta0_;
100
101     biViscosityCoeffs_.lookup("tau0") >> tau0_;
102     biViscosityCoeffs_.lookup("etaP") >> etaP_;
103
104     return true;
105 }
106
107
108 // ************************************************************************* //
```

10.2.2 biViscosity.H

```
/*---------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2016 OpenFOAM Foundation
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/

Class
    Foam::viscosityModels::biViscosity

Description
     biViscosity non-Newtonian viscosity model.

SourceFiles
    biViscosity.C

\*---------------------------------------------------------------------------*/

#ifndef biViscosity_H
#define biViscosity_H

#include "viscosityModel.H"
#include "dimensionedScalar.H"
#include "volFields.H"

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

namespace Foam
{
namespace viscosityModels
{

/*---------------------------------------------------------------------------*\
                          Class biViscosity Declaration
\*---------------------------------------------------------------------------*/

class biViscosity
:
    public viscosityModel
{
    // Private data

        dictionary biViscosityCoeffs_;

        dimensionedScalar eta0_;
        dimensionedScalar tau0_;
```

```
48          dimensionedScalar etaP_;
49
50          volScalarField nu_;
51
52
53      // Private Member Functions
54
55          //- Calculate and return the laminar viscosity
56          tmp<volScalarField> calcNu() const;
57
58
59  public:
60
61      //- Runtime type information
62      TypeName("biViscosity");
63
64
65      // Constructors
66
67          //- Construct from components
68          biViscosity
69          (
70              const word& name,
71              const dictionary& viscosityProperties,
72              const volVectorField& U,
73              const surfaceScalarField& phi
74          );
75
76
77      //- Destructor
78      virtual ~biViscosity()
79      {}
80
81
82      // Member Functions
83
84          //- Return the laminar viscosity
85          virtual tmp<volScalarField> nu() const
86          {
87              return nu_;
88          }
89
90          //- Return the laminar viscosity for patch
91          virtual tmp<scalarField> nu(const label patchi) const
92          {
93              return nu_.boundaryField()[patchi];
94          }
95
96          //- Correct the laminar viscosity
97          virtual void correct()
```

```
98          {
99              nu_ = calcNu();
100         }
101
102         //- Read transportProperties dictionary
103         virtual bool read(const dictionary& viscosityProperties);
104 };
105
106
107 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
108
109 } // End namespace viscosityModels
110 } // End namespace Foam
111
112 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
113
114 #endif
115
116 // ************************************************************************* //
```

10.2.3 SMD.C

```
/*---------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2016 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------------

#include "SMD.H"
#include "addToRunTimeSelectionTable.H"
#include "surfaceFields.H"


// * * * * * * * * * * * * * Static Data Members * * * * * * * * * * * * * //

namespace Foam
{
namespace viscosityModels
{
    defineTypeNameAndDebug(SMD, 0);

    addToRunTimeSelectionTable
    (
        viscosityModel,
        SMD,
        dictionary
    );
}
}


// * * * * * * * * * * * * Private Member Functions  * * * * * * * * * * * * //

Foam::tmp<Foam::volScalarField>
Foam::viscosityModels::SMD::calcNu() const
{
dimensionedScalar etaInf=max
                         (
                          etaI_.value(),
                          VSMALL
                          )
                          *dimensionedScalar("one", dimViscosity, 1.0);

tmp<volScalarField> sr(
                  max
                  (
                   strainRate(),
```

```
48                    dimensionedScalar ("VSMALL", dimless/dimTime, VSMALL)
49                    )
50                    *dimensionedScalar("one", dimTime, 1.0));
51
52  return (
53         (1-exp(-eta0_.value()*sr()/tau0_.value()))
54         *( tau0_/sr()*dimensionedScalar ("one", dimTime, 1)
55            +k_*pow(sr(),n_.value()-1)
56           )
57         +etaInf
58         );
59  }
60
61
62  // * * * * * * * * * * * * * * Constructors  * * * * * * * * * * * * * //
63
64  Foam::viscosityModels::SMD::SMD
65  (
66      const word& name,
67      const dictionary& viscosityProperties,
68      const volVectorField& U,
69      const surfaceScalarField& phi
70  )
71  :
72      viscosityModel(name, viscosityProperties, U, phi),
73      SMDCoeffs_(viscosityProperties.optionalSubDict(typeName + "Coeffs")),
74      k_("k", dimViscosity, SMDCoeffs_),
75      n_("n", dimless, SMDCoeffs_),
76      eta0_("eta0", dimViscosity, SMDCoeffs_),
77      tau0_("tau0", dimViscosity/dimTime, SMDCoeffs_),
78      etaI_("etaI", dimViscosity, SMDCoeffs_),
79      nu_
80      (
81          IOobject
82          (
83              name,
84              U_.time().timeName(),
85              U_.db(),
86              IOobject::NO_READ,
87              IOobject::AUTO_WRITE
88          ),
89          calcNu()
90      )
91
92  {}
93
94
95  // * * * * * * * * * * * * * * Member Functions  * * * * * * * * * * * * * //
96
97  bool Foam::viscosityModels::SMD::read
```

```
98  (
99      const dictionary& viscosityProperties
100 )
101 {
102     viscosityModel::read(viscosityProperties);
103
104     SMDCoeffs_ = viscosityProperties.optionalSubDict(typeName + "Coeffs");
105
106     SMDCoeffs_.lookup("k") >> k_;
107     SMDCoeffs_.lookup("n") >> n_;
108     SMDCoeffs_.lookup("tau0") >> tau0_;
109     SMDCoeffs_.lookup("eta0") >> eta0_;
110     SMDCoeffs_.lookup("etaI") >> etaI_;
111
112
113     return true;
114 }
115
116
117 // ************************************************************************* //
```

10.2.4 SMD.H

```
/*---------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2016 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------------

Class
    Foam::viscosityModels::SMD

Description
     SMD non-Newtonian viscosity model.

SourceFiles
    SMD.C

\*---------------------------------------------------------------------------*/

#ifndef SMD_H
#define SMD_H

#include "viscosityModel.H"
#include "dimensionedScalar.H"
#include "volFields.H"

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

namespace Foam
{
namespace viscosityModels
{

/*---------------------------------------------------------------------------*\
                           Class SMD Declaration
\*---------------------------------------------------------------------------*/

class SMD
:
    public viscosityModel
{
    // Private data

        dictionary SMDCoeffs_;

        dimensionedScalar k_;
        dimensionedScalar n_;
```

```
48          dimensionedScalar eta0_;
49          dimensionedScalar tau0_;
50          dimensionedScalar etaI_;
51
52          volScalarField nu_;
53
54
55
56      // Private Member Functions
57
58          //- Calculate and return the laminar viscosity
59          tmp<volScalarField> calcNu() const;
60
61
62  public:
63
64      //- Runtime type information
65      TypeName("SMD");
66
67
68      // Constructors
69
70          //- Construct from components
71          SMD
72          (
73              const word& name,
74              const dictionary& viscosityProperties,
75              const volVectorField& U,
76              const surfaceScalarField& phi
77          );
78
79
80      //- Destructor
81      virtual ~SMD()
82      {}
83
84
85      // Member Functions
86
87          //- Return the laminar viscosity
88          virtual tmp<volScalarField> nu() const
89          {
90              return nu_;
91          }
92
93          //- Return the laminar viscosity for patch
94          virtual tmp<scalarField> nu(const label patchi) const
95          {
96              return nu_.boundaryField()[patchi];
97          }
```

```
98
99          //- Correct the laminar viscosity
100         virtual void correct()
101         {
102             nu_ = calcNu();
103         }
104
105         //- Read transportProperties dictionary
106         virtual bool read(const dictionary& viscosityProperties);
107 };
108
109
110 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
111
112 } // End namespace viscosityModels
113 } // End namespace Foam
114
115 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
116
117 #endif
118
119 // ************************************************************************* //
```

10.2.5 files

```
SMD/SMD.C
biViscosity/biViscosity.C

LIB = $(FOAM_USER_LIBBIN)/libuserViscosity
```

10.2.6 options

```
EXE_INC = \
-I$(LIB_SRC)/transportModels/incompressible/lnInclude/ \
-I$(LIB_SRC)/finiteVolume/lnInclude

LIB_LIBS = -lfiniteVolume -lincompressibleTransportModels
```

10.3 APÊNDICE III - Pré-processamento

10.3.1 controlDict

```
/*--------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2016 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     nonNewtonianSimpleFoam;

startFrom       latestTime;

startTime       0;

stopAt          endTime;

endTime         30000000;

deltaT          1;

writeControl    runTime;

writeInterval   1000;

purgeWrite      2;

writeFormat     ascii;

writePrecision  8;

writeCompression off;

timeFormat      general;

timePrecision   7;

```

```
46 runTimeModifiable true;
47
48 functions
49     {
50     #includeFunc  residuals
51     #includeFunc  grad
52     #includeFunc  patchIntegrate
53     }
54
55 // ************************************************************************* //
```

10.3.2 fvSchemes

```
/*--------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2016 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default         steadyState;
}
gradSchemes
{
    default         Gauss linear;
    grad(p)         leastSquares;
}
divSchemes
{
    default         none;
    div(phi,U)      Gauss linear;
    div(phi,T)      Gauss linear;
}
laplacianSchemes
{
    default         Gauss linear corrected;
}
interpolationSchemes
{
    default         cubic;
}
snGradSchemes
{
    default         corrected;
}

// ************************************************************************* //
```

10.3.3 fvSolution

```
/*--------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2016 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    p
    {
        solver          GAMG;
        smoother        GaussSeidel;
        relTol 1e-3;
        tolerance 1e-12;
    }

    U
    {
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance 1e-12;
        relTol          1e-3;

    }

    T
    {
        solver PBiCG;
        preconditioner DILU;
        tolerance 0;
        relTol 1e-5;
    }
}

SIMPLE
{
```

```
48      nNonOrthogonalCorrectors 0;
49      pRefCell        0;
50      pRefValue       0;
51
52      residualControl
53      {
54          p               1e-9;
55          U               1e-9;
56          T               1e-9;
57      }
58  }
59
60  relaxationFactors
61  {
62      fields
63      {
64          p               1;
65      }
66      equations
67      {
68          U               0.3;
69          ".*"            0.7;
70      }
71  }
72  // ************************************************************************* //
```

10.3.4 blockMeshDict - Validação

```
/*--------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2016 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------
FoamFile
{
version     2.0;
format      ascii;
class       dictionary;
object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

convertToMeters 1;

l 1;

vertices
(
  (  0     0    0) //0
  (  $l    0    0) //1
  (  $l    $l   0) //2
  (  0     $l   0) //3

  (  0     0    1) //4
  (  $l    0    1) //5
  (  $l    $l   1) //6
  (   0    $l   1) //7

);

blocks
(
  hex ( 0 1 2 3 4 5 6 7) (100 100 1 )
    simpleGrading ( ( (0.5 0.5 5.6737) (0.5 0.5 0.17625) )
    ( (0.5 0.5 5.6737) (0.5 0.5 0.17625) ) 1)
);

edges
(
);

defaultPatch
{
```

```
48    name walls;
49    type wall;
50  }
51
52  boundary
53  (
54    hot
55    {
56      type wall;
57      faces
58      (
59        (1 2 6 5)
60      );
61    }
62    cold
63    {
64      type wall;
65      faces
66      (
67        (0 3 7 4)
68      );
69    }
70
71    back
72    {
73      type empty;
74      faces
75      (
76        (0 1 2 3)
77        (4 5 6 7)
78      );
79    }
80  );
81
82  // ************************************************************************* //
```

10.3.5 blockMeshDict - Aplicação

```
/*--------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2016 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------
FoamFile
{
version     2.0;
format      ascii;
class       dictionary;
object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

convertToMeters 1;

vertices
(
  (-20    0   0) //0
  ( -3.15 0   0) //1
  (  3.15 0   0) //2
  ( 20    0   0) //3
  (-20    1   0) //4
  ( -3.15 1   0) //5
  (  3.15 1   0) //6
  ( 20    1   0) //7
  ( -3.15 6.3 0) //8
  (  3.15 6.3 0) //9

  (-20    0    1) //10
  ( -3.15 0    1) //11
  (  3.15 0    1) //12
  ( 20    0    1) //13
  (-20    1    1) //14
  ( -3.15 1    1) //15
  (  3.15 1    1) //16
  ( 20    1    1) //17
  ( -3.15 6.3  1) //18
  (  3.15 6.3  1) //19
);

blocks
(

  hex (0 1 5 4 10 11 15 14) ( 113 14 1 ) simpleGrading (1 1 1)
```

```
48    hex (1 2 6 5 11 12 16 15) ( 84 14 1 ) simpleGrading (1 1 1)
49    hex (2 3 7 6 12 13 17 16) ( 113 14 1 ) simpleGrading (1 1 1)
50    hex (5 6 9 8 15 16 19 18) ( 84 71 1 ) simpleGrading (1 1 1)
51
52  );
53
54  edges
55  (
56  );
57
58  boundary
59  (
60    inlet
61    {
62      type patch;
63      faces
64      (
65        (0 4 14 10)
66      );
67    }
68
69    walls
70    {
71      type wall;
72      faces
73      (
74        (4 5 15 14)
75        (6 7 17 16)
76      );
77    }
78
79    wallsCavity
80    {
81      type wall;
82      faces
83      (
84        (5 8 18 15)
85        (8 9 19 18)
86        (6 9 19 16)
87      );
88    }
89
90    outlet
91    {
92      type patch;
93      faces
94      (
95        (3 7 17 13)
96      );
97    }
```

```
98
99    centreline
100   {
101     type symmetryPlane;
102     faces
103     (
104       (0 1 11 10)
105       (1 2 12 11)
106       (2 3 13 12)
107     );
108   }
109
110   frontAndBack
111   {
112     type empty;
113     faces
114     (
115       (0 1 5 4)
116       (1 2 6 5)
117       (2 3 7 6)
118       (5 6 9 8)
119       (10 11 15 14)
120       (11 12 16 15)
121       (12 13 17 16)
122       (15 16 19 18)
123     );
124   }
125 );
126
127 mergePatchPairs
128 (
129 );
130
131 // ************************************************************************* //
```

10.3.6 transportProperties - Validação

```
/*--------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2016 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------
FoamFile
{
version     2.0;
format      ascii;
class       dictionary;
location    "constant";
object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

transportModel  powerLaw;


alpha alpha [0 2 -1 0 0 0 0]  0.00014142;

rho rho [1 -3 0 0 0 0 0]  1;

beta beta [0 0 0 -1 0 0 0]  1;

gamaRef gamaRef [0 0 -1 0 0 0 0] 1;

ratio 1;

powerLawCoeffs
{
nuMax    [ 0 2 -1 0 0 0 0 ] 1e10;
nuMin    [ 0 2 -1 0 0 0 0 ] 1e-10;
k        [ 0 2 -1 0 0 0 0 ] 16.9865;
n        [ 0 0  0 0 0 0 0 ]    1.8;
}

biViscosityCoeffs
{
eta0        [ 0 2 -1 0 0 0 0 ]   10000;
etaP        [ 0 2 -1 0 0 0 0 ] 1;
tau0        [ 0 2 -2 0 0 0 0 ] 59.7614;
}

// ************************************************************************* //
```

10.3.7 transportProperties - Aplicação

```
/*--------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2016 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------
FoamFile
{
version     2.0;
format      ascii;
class       dictionary;
location    "constant";
object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

transportModel  SMD;

alpha alpha [0 2 -1 0 0 0 0]   0.0028538;

rho rho [1 -3 0 0 0 0 0]  1;

beta beta [0 0 0 -1 0 0 0]  0;

gamaRef gamaRef [0 0 -1 0 0 0 0] 0.00018027;

ratio 0.4550;

SMDCoeffs
{
eta0            [0 2 -1 0 0 0 0]   349.6237;
tau0            [0 2 -2 0 0 0 0]   0.016;
n       [0 0 0 0 0 0 0]   0.5;
k         [0 2 -1 0 0 0 0]   0.02365;
etaI       [0 2 -1 0 0 0 0]   0.00034959;
}

// ************************************************************************* //
```

10.3.8 Pasta 0 - Validação

scalarDict

```
/*--------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2016 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------------
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "0";
    object      scalarDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

tempo    0;


// ************************************************************************* //

```

U

```
/*--------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2016 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
    cold
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }

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

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

// ************************************************************************* //

```

p

```
/*--------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2016 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    hot
    {
        type            zeroGradient;
    }

   cold
    {
        type            zeroGradient;
    }

    walls
    {
        type            zeroGradient;
    }
}

// ************************************************************************* //

```

T

```
/*--------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2016 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      T;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 1 0 0 0];

internalField   uniform 0;

boundaryField
{
    cold
    {
        type            fixedValue;
        value           uniform 0;
    }

    hot
    {
        type            fixedValue;
        value           uniform 1;
    }

    walls
    {
        type            zeroGradient;
    }
}

// ************************************************************************* //

```

tau

```
/*--------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2016 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      sr;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    hot
    {
        type            calculated;
        value           uniform 0;
    }

   cold
    {
        type            calculated;
        value           uniform 0;
    }

    walls
    {
        type            calculated;
        value           uniform 0;
    }
    #includeEtc "caseDicts/setConstraintTypes"
}

// ************************************************************************* //

```

sr

```
/*--------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2016 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      sr;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 -1 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    hot
    {
        type            calculated;
        value           uniform 0;
    }

   cold
    {
        type            calculated;
        value           uniform 0;
    }

    walls
    {
        type            calculated;
        value           uniform 0;
    }
    #includeEtc "caseDicts/setConstraintTypes"
}

// ************************************************************************* //

```

yieldGama

```
/*--------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2016 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      yieldGama;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    hot
    {
        type            calculated;
        value           uniform 0;
    }

   cold
    {
        type            calculated;
        value           uniform 0;
    }

    walls
    {
        type            calculated;
        value           uniform 0;
    }
    #includeEtc "caseDicts/setConstraintTypes"
}

// ************************************************************************* //

```

yieldGama2

```
/*--------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2016 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      yieldGama2;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    hot
    {
        type            calculated;
        value           uniform 0;
    }

   cold
    {
        type            calculated;
        value           uniform 0;
    }

    walls
    {
        type            calculated;
        value           uniform 0;
    }
    #includeEtc "caseDicts/setConstraintTypes"
}

// ************************************************************************* //

```

10.3.9 Pasta 0 - Aplicação

U

```
/*--------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2016 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------------
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (1 0 0);

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform (1 0 0);
    }
    outlet
    {
        type            zeroGradient;
    }
    walls
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
    wallsCavity
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
    #includeEtc "caseDicts/setConstraintTypes"
}
// ************************************************************************* //

```

p

```
/*--------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2016 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    inlet
    {
        type            zeroGradient;
    }

    outlet
    {
        type            fixedValue;
        value           uniform 0;
    }

    walls
    {
        type            zeroGradient;
    }

    wallsCavity
    {
        type            zeroGradient;
    }
    #includeEtc "caseDicts/setConstraintTypes"
}

// ************************************************************************* //

```

T

```
/*--------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2016 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      T;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 1 0 0 0];

internalField   uniform 0;

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform 0;
    }

    outlet
    {
        type            zeroGradient;
    }

    walls
    {
        type            zeroGradient;
    }

    wallsCavity
    {
        type            fixedValue;
        value           uniform 1;
    }
    #includeEtc "caseDicts/setConstraintTypes"
}
// ************************************************************************* //
```

tau

```
/*--------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2016 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    inlet
    {
        type            calculated;
        value           uniform 0;
    }
    outlet
    {
        type            calculated;
        value           uniform 0;
    }
    walls
    {
        type            calculated;
        value           uniform 0;
    }
    wallsCavity
    {
        type            calculated;
        value           uniform 0;
    }
    #includeEtc "caseDicts/setConstraintTypes"
}
// ************************************************************************* //
```

sr

```
/*--------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2016 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      sr;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 -1 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    inlet
    {
        type            calculated;
        value           uniform 0;
    }
    outlet
    {
        type            calculated;
        value           uniform 0;
    }
    walls
    {
        type            calculated;
        value           uniform 0;
    }
    wallsCavity
    {
        type            calculated;
        value           uniform 0;
    }
    #includeEtc "caseDicts/setConstraintTypes"
}
// ************************************************************************* //

```

yieldGama

```
/*--------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2016 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      yieldGama;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    inlet
    {
        type            calculated;
        value           uniform 0;
    }
    outlet
    {
        type            calculated;
        value           uniform 0;
    }
    walls
    {
        type            calculated;
        value           uniform 0;
    }
    wallsCavity
    {
        type            calculated;
        value           uniform 0;
    }
    #includeEtc "caseDicts/setConstraintTypes"
}
// ************************************************************************* //
```

yieldGama2

```
/*--------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2016 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      yieldGama2;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    inlet
    {
        type            calculated;
        value           uniform 0;
    }
    outlet
    {
        type            calculated;
        value           uniform 0;
    }
    walls
    {
        type            calculated;
        value           uniform 0;
    }
    wallsCavity
    {
        type            calculated;
        value           uniform 0;
    }
    #includeEtc "caseDicts/setConstraintTypes"
}
// ************************************************************************* //
```

10.4 APÊNDICE IV - Pós-processamento

10.4.1 residuals

```
/*--------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2016 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------------
Description
    For specified fields, writes out the initial residuals for the first
    solution of each time step; for non-scalar fields (e.g. vectors), writes
    the largest of the residuals for each component (e.g. x, y, z).

\*--------------------------------------------------------------------------*/

#includeEtc "caseDicts/postProcessing/numerical/residuals.cfg"

fields (p U T);

writeControl    timeStep;
writeInterval   100;
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

10.4.2 grad

```
/*--------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2016 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------------
Description
    Calculates the gradient of a field.

\*--------------------------------------------------------------------------*/

type            grad;
libs            ("libfieldFunctionObjects.so");

field           T;

executeControl  writeTime;
writeControl    writeTime;

// ************************************************************************* //
```

10.4.3 patchIntegrate

```
/*--------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2016 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------------
Description
    Calculates the surface integral of one or more fields on a patch.

\*--------------------------------------------------------------------------*/

name   hot;
//name   cavityWalls;

fields (grad(T));

operation areaNormalIntegrate;
#includeEtc "caseDicts/postProcessing/surfaceFieldValue/patch.cfg"

writeControl    timeStep;
writeInterval   1000;

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

10.4.4 cellAvarege

```
/*--------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2016 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------------
Description
    Writes out the minimum cell value for one or more fields.

\*--------------------------------------------------------------------------*/

#includeEtc "caseDicts/postProcessing/minMax/cellAverage.cfg"

fields (yieldGama2);

writeControl    timeStep;
writeInterval   1000;
operation       volAverage;

// ************************************************************************* //
```

10.4.5 graph - Validação

```
/*--------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2016 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      ;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

graph
{
    type            sets;
    libs ("libsampling.so");
    writeControl   writeTime;

    interpolationScheme cellPoint;

    setFormat       csv;

    sets
    (
        cavidade1
        {
            type    face;
            axis    x;
            start   ( 0  .5 .5);
            end     ( 1  .5 .5);
        }


    );

    fields      ( U T);
}

// ************************************************************************* //
```

10.4.6 graph - Aplicação

```
/*--------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | Copyright (C) 2011-2016 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      ;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

graph
{
    type            sets;
    libs ("libsampling.so");
    writeControl   writeTime;

    interpolationScheme cellPoint;

    setFormat       raw;

    sets
    (
       midPlane
        {
            type    uniform;
            nPoints  1000;
            axis    y;
            start   ( 0  0 .5);
            end     ( 0  6.3 .5);
        }

       symPlane
        {
            type    uniform;
            nPoints  1000;
            axis    x;
            start   ( -10  0 .5);
            end     (  10  0 .5);
        }


```

```
48
49      );
50
51      fields      (T p);
52  }
53
54  // ************************************************************************* //
```