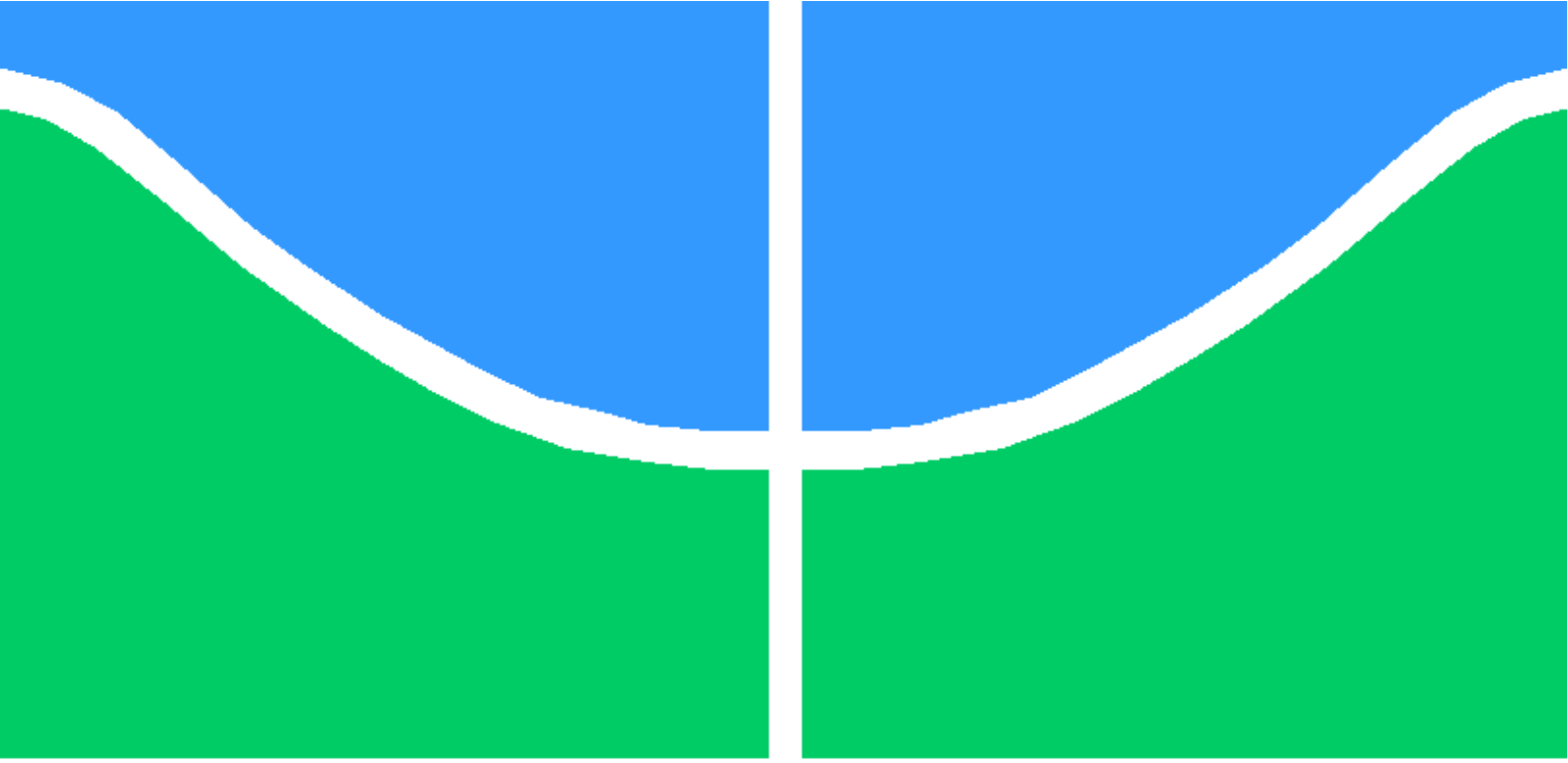

MODELO COMPUTACIONAL PARA UM PARAFUSO DE MÁQUINA EXTRUSORA DE POLÍMEROS

Matheus Rocha Ferrari

PROJETO DE GRADUAÇÃO

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA MECÂNICA

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia
Departamento de Engenharia Mecânica

PROJETO DE GRADUAÇÃO

MODELO COMPUTACIONAL PARA UM PARAFUSO DE MÁQUINA EXTRUSORA DE POLÍMEROS

Matheus Rocha Ferrari

Relatório submetido como requisito para obtenção do grau de Engenheiro Mecânico.

Banca Examinadora

Dianne Viana Magalhães, UnB/ENM _____
Adriano Possebon Rosa, UnB/ENM _____
Fabio Gomes de Castro, UnB/ENM _____
Taygoara Felamino de Oliveira, UnB/ENM _____

Brasília, 21 de Novembro de 2020

"Tatakae, tatakae."
-JAEGER, Eren

RESUMO

O processo mais adequado para a condição de operação de uma máquina extrusora, em questão de produtividade, envolve alguns fatores como o polímero utilizado, a vazão, incrementos de mistura, controle de temperatura e outros. Para se determinar a velocidade do escoamento no interior da máquina e garantir a homogeneidade de produção do mesmo, deve-se estudar o comportamento do escoamento ao longo do parafuso. Esse trabalho é um primeiro passo na elaboração de um modelo computacional capaz de simular um fluido não Newtoniano movido pelo parafuso. O código se baseia nas equações de Navier-Stokes, discretizadas através do método diferenças finitas, em um espaço bidimensional. Após a solução numérica das equações de Navier-Stokes, a pressão e a velocidade do escoamento e a força exercida na parede superior são obtidas, nos permitindo prever de forma simplificada a pressão exercida na parede, e assim, a força à qual as parede estão sujeitas.

ABSTRACT

The most suitable process for an operating extruder machine, in terms of productivity, involves some factors such as the polymer used, the flow rate, mixing increments, temperature control and others. In order to determine the flow velocity inside the machine and guarantee its homogeneous production, the behavior of the flow along the screw must be studied. This work is a first step in the preparation of a computational model capable of simulating a non-Newtonian fluid moved by the screw. The code is based on the Navier-Stokes equation, discretized through the finite element method, in a two-dimensional space. After the numerical solution of the Navier-Stokes equations, the pressure, the velocity and the force exerted on the upper wall are determined, allowing us to predict in a simplified way the pressure exercised on the wall, and thus, the force in the wall.

Sumário

Lista de Figuras

Lista de Símbolos

u - Campo de velocidades em x $[\frac{m}{s}]$;

v - Campo de velocidades em y $[\frac{m}{s}]$;

x - Distância em x $[m]$;

y - Distância em y $[m]$;

z - Distância em z $[m]$;

t - Tempo $[s]$;

μ - Viscosidade $[\frac{m^2}{s}]$;

μ_0 - Consistência do escoamento $[\frac{m^2}{s}]$;

$\dot{\gamma}$ - Taxa de cisalhamento $[\ \frac{1}{s}]$;

n - Índice da Lei de Potência;

p - Campo de pressão $[Pa]$;

H - Altura da cavidade $[m]$;

x^* - Distância em x;

y^* - Distância em y;

p^* - Campo de pressão;

t^* - Tempo;

u^* - Velocidade em x;

v^* - Velocidade em v;

Re - Número de Reynolds;

D - Viscosidade;

ν - Viscosidade cinemática $[\frac{m^2}{s}]$

ρ - Massa específica $[\frac{Kg}{m^3}]$;

u_0 - Velocidade da tampa movida;

γ_1 - Limite inferior da taxa de cisalhamento;

γ_2 - Limite superior da taxa de cisalhamento;

ψ - Linha de corrente;

i - Índice utilizado na direção x;

j - Índice utilizado na direção y;

k - Índice contador de iterações no tempo;

nx - Número de divisões do domínio, em x;

ny - Número de divisões do domínio, em y.

CAPITULO 1

INTRODUÇÃO

Soluções numéricas para as equações de Navier-Stokes são amplamente estudadas na área de dinâmica do fluidos (CFD), impulsionadas pelo grande número de aplicações na física e na engenharia. É uma ferramenta importante para estudar o comportamento de fluidos Newtonianos e não Newtonianos, e muitos dos materiais presentes na indústria pertencem a esta última categoria citada.

Este trabalho busca simular o polímero dentro de uma máquina extrusora, elaborando um modelo capaz de obter a solução para as equações de Navier-Stokes aplicadas para um fluido não Newtoniano, em um problema da cavidade com dependência no tempo. Existem muitos modelos que conseguem descrever, com boa precisão, o comportamento físico desse escoamento. Devido às características físicas do canal pelo qual o polímero escoa, é possível simplificar o domínio do nosso problema para um domínio bidimensional, com sistema de coordenadas cartesianas, quando normalmente o problema é tratado como cilíndrico e tridimensional.

Uma dificuldade encontrada na solução desse problema é desacoplar a velocidade da pressão, uma relação imposta pela restrição de incompressibilidade. O obstáculo é contornado utilizando o método de projeção de Chorin [7], amplamente estudado na literatura e comprovadamente um modelo numérico eficiente. Para avaliar a acurácia e a eficiência do método numérico, pode ser efetuada a comparação com resultados de trabalhos bem estabelecidos, uma vez que o problema da cavidade é amplamente estudado. Baseando-se na relação existente entre a taxa de cisalhamento e a viscosidade, o fluido pode ser descrito como Newtoniano ou não Newtoniano.

No presente trabalho, quando lidando com um fluido não Newtoniano, equações constitutivas muito mais complexas são utilizadas no modelo, se comparado com as equações utilizadas no modelo Newtoniano. A viscosidade de um fluido não Newtoniano pode assumir, dentro da lei de potência utilizada para descrevê-la, duas características, a de se tornar mais ou menos viscosa com o aumento da taxa de cisalhamento. Quando ocorre o aumento da taxa de cisalhamento e a viscosidade cai, o fluido é chamado de pseudoplástico; quando a viscosidade sobe, o fluido é chamado de dilatante.

O melhor design para um parafuso extrusor é particular para cada condição de operação, como o tipo de polímero escolhido, a vazão da máquina, dispositivos de mistura, ventilação e bocal. Um parafuso bem projetado é crítico para a otimização de produção, controle da extrusora e tipos de polímeros que podem ser utilizados. Um parafuso mal projetado pode levar à instabilidade de alta frequência, causado por variações de temperatura e pressão dentro da extrusora [1]. De maneira geral, parafusos projetados para tratar polietileno ou polipropileno de alta densidade não funcionam de maneira adequada quando utilizado policarbonatos, náilon e politereftalato de etileno. Não existe um parafuso com propósito de processamento universal, se o objetivo final for alta eficiência.

1.1 Objetivos

O objetivo desta etapa do trabalho é dar um primeiro passo na elaboração de um código capaz conseguir retirar dados necessários e suficientes para determinar a geometria de um parafuso extrusor de alta produtividade. Neste trabalho devemos ser capazes de retirar os campos de velocidade, pressão e taxa de cisalhamento, assim como a força exercida na tampa da cavidade (ideia elaborada no Capítulo 3). O foco deste trabalho é observar o comportamento do fluido não Newtoniano dentro de uma cavidade, entender a influência de n nos campos de velocidade e pressão e na força necessária para manter o escoamento.

O capítulo 2 deste trabalho contém uma introdução à máquina extrusora - e o processo de transformação do polímero que ocorre no seu interior - de maneira simplificada; Introduz conceitos propostos no trabalho de Gaspar-Cunha [2], que caracteriza o fenômeno de mudança de fase em diversos estágios. O capítulo 3 contém a descrição do modelo físico utilizado para a solução do problema - baseado no trabalho de Richard [4] - mostrando o sistema de coordenadas adotado, o sistema referencial e as equações governantes. O capítulo 4 contém a descrição da metodologia utilizada na resolução das equações governantes - método da projeção, de Chorin [7] - e a implementação numérica do método. O capítulo 5 contém as informações pertinentes ao código criado e seu funcionamento. O capítulo 6 contém a comparação entre os dados obtidos e literatura - para diversos casos de velocidade e características viscosas - por meio das linhas de corrente e pontos específicos de velocidades. Os autores utilizados para o comparativo foram Ghia [5] e Li et al. [6]. No capítulo 7 temos as interpretações feitas sobre os dados obtidos, as curvas de distribuição de forças e a curva que relaciona os carregamentos gerados por diferentes valores para o número de Reynolds e n (variável da lei de potência).

CAPITULO 2

CARACTERÍSTICAS DA MÁQUINA EXTRUSORA

O polímero a ser processado na máquina extrusora, na maior parte dos casos, vem em grãos do material. Os grãos despejados em um funil de alimentação adentram a máquina por uma abertura no tubo que envolve o parafuso extrusor. O polímero, até então no estado sólido, derrete à medida que avança pela extensão do parafuso. As sessões a seguir mostram os padrões observados na fusão do material e como são divididas as etapas de processamento do mesmo [2].

2.1 Máquina Extrusora

A extrusora é um equipamento empregado na indústria do plástico na fabricação de produtos contínuos como filmes, perfis, tubos, monofilamentos, entre outros. Além da produção de material granulado novo ou reciclado. A Figura 1 mostra uma máquina extrusora de parafuso único com suas principais componentes destacadas.

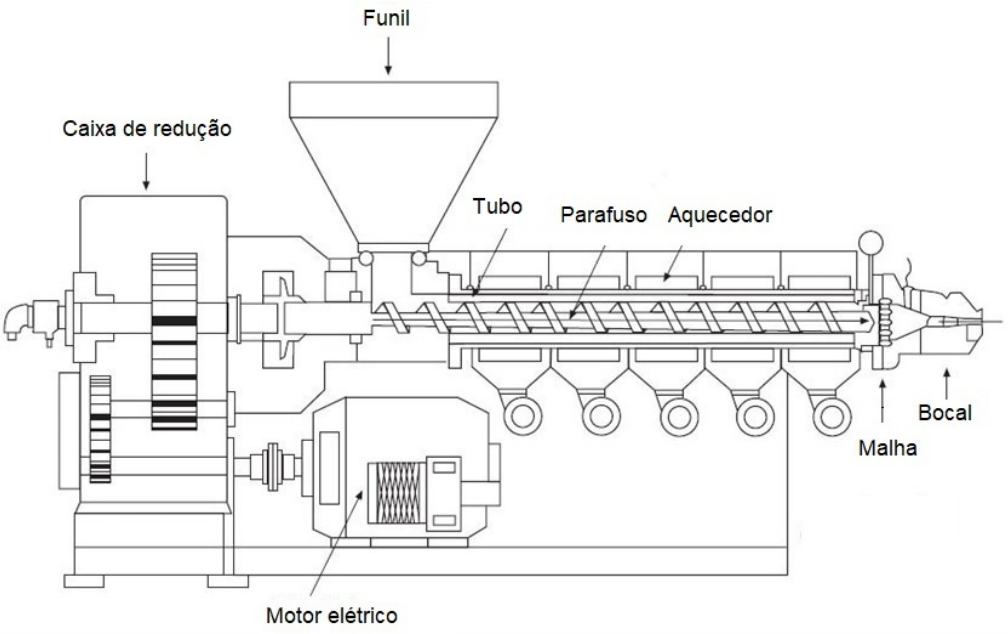

Figura 1: Máquina extrusora e componentes, adaptada de Giles et al. [1].

O polímero entra pelo funil de alimentação da máquina e é movido para dentro do tubo, ou *barrel*, com auxílio do parafuso extrusor. À medida que entra na máquina, o polímero recebe

calor através de condução pelas paredes do tubo e através do calor gerado por efeitos de atrito. O calor que o tubo conduz ao plástico em processamento é oriundo dos aquecedores dispostos ao longo de todo o seu comprimento. Esses aquecedores costumam ser controlados por sensores *PID*, evitando o superaquecimento do sistema e, consequentemente, que o material mude sua estrutura original. Em alguns casos, se faz necessário a presença de um sistema de ventilação para auxiliar o controle de temperatura, uma vez que o controle padrão só permite elevar a temperatura. A malha presente no final do percurso do parafuso busca melhorar a mistura do processo e orienta a estrutura do material de forma linear, antes com aspecto torcido [1]. A Figura 2 permite uma visualização detalhada da malha.

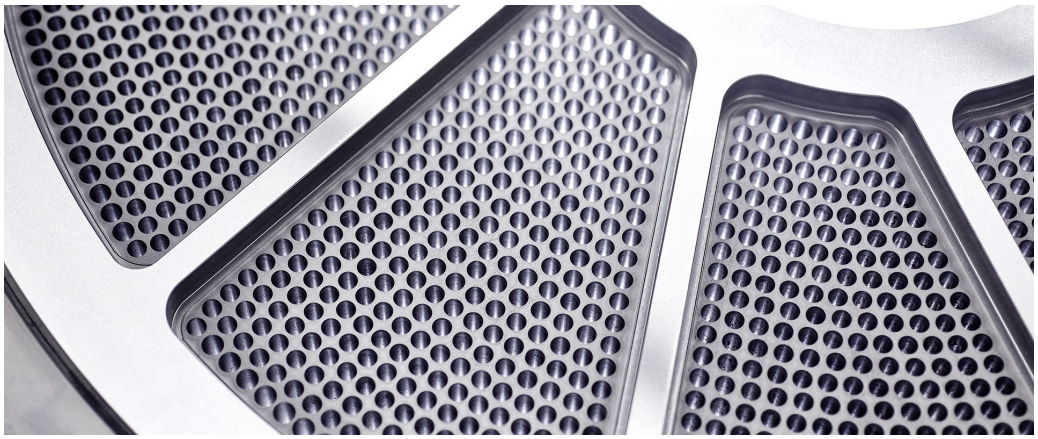

Figura 2: Malha de máquina extrusora. *Gneuss Kunststofftechnik website*

2.2 Parafuso Extrusor

Usualmente, um parafuso extrusor tem três zonas com diferentes geometrias, denominadas zonas de alimentação, de compressão e de dosagem. Cada uma das zonas tem uma profundidade de canal específica, assim como um comprimento. A Figura 3 ilustra a divisão em zonas citada acima. A forma como o fluido começa a fundir pode ser observada na Figura 4.

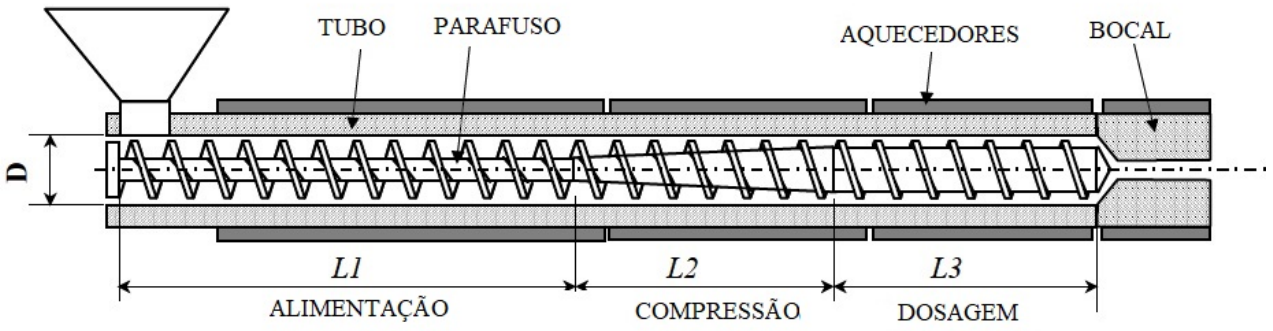

Figura 3: Seção em corte da extrusora, para ilustração dos parâmetros. Imagem adaptada de Gaspar-Cunha [2].

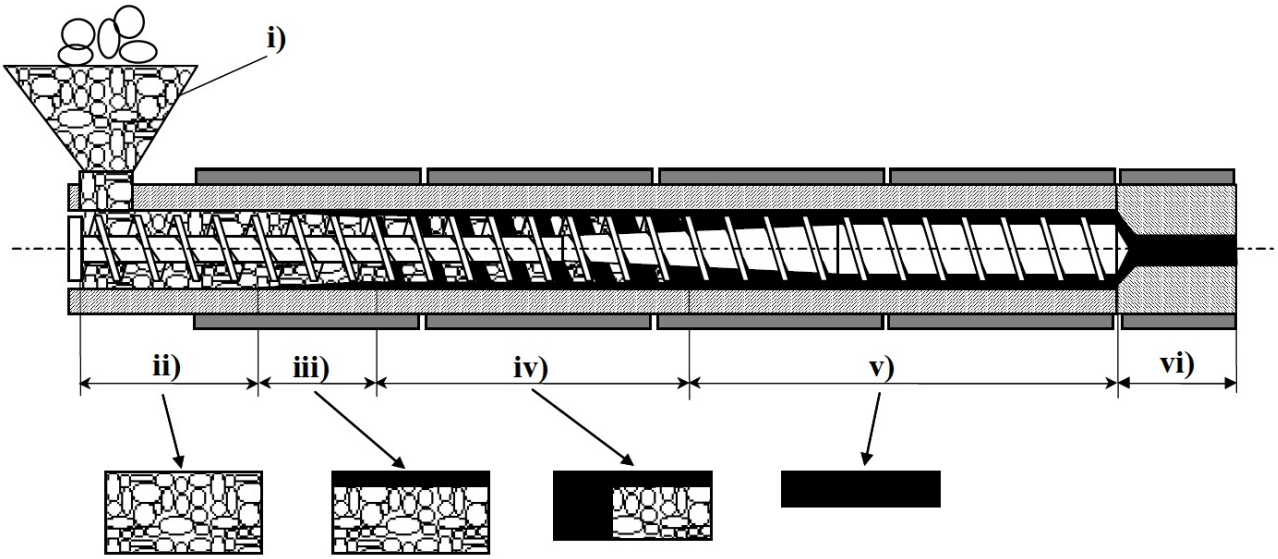

Figura 4: Detalhes do material e comportamento. Imagem adaptada de Gaspar-Cunha [2].

No começo da zona de alimentação, a resina é forçada contra a parede interna do tubo, gerando fricção. Combinada a essa fricção, o calor provindo dos aquecedores por condução criam um filme de resina fundida na superfície interna do tubo. O tratamento desta etapa é mais simples, uma vez que, dada a baixa razão entre os tamanhos da película e do parafuso, o escoamento pode ser tratado como de Couette para um fluido com características não-Newtonianas, em um plano bidimensional.

Segundo Giles et al. [1], na zona de compressão, ocorre o incremento de pressão e compressão, fazendo com que o polímero seja dividido em três áreas distintas: uma cama de sólido compactado, um filme de resina derretida na parede do tubo e uma poça de fusão entre elas.

Na zona de dosagem, a profundidade de canal continua decrescendo. O derretimento da cama sólida deve acontecer por completo, evitando a distorção do escoamento causado por grãos ainda sólidos.

2.2.1 Implemento de Mistura

Existem muitos tipos de implementos para mistura, patenteados por diferentes companhias, para atender a diferentes demandas por mistura (mistura dispersiva, distributiva ou os dois). A geometria adequada depende do tipo de processamento, a velocidade do parafuso e dos materiais a serem misturados. A Figura 5 ilustra o design desses dispositivos.

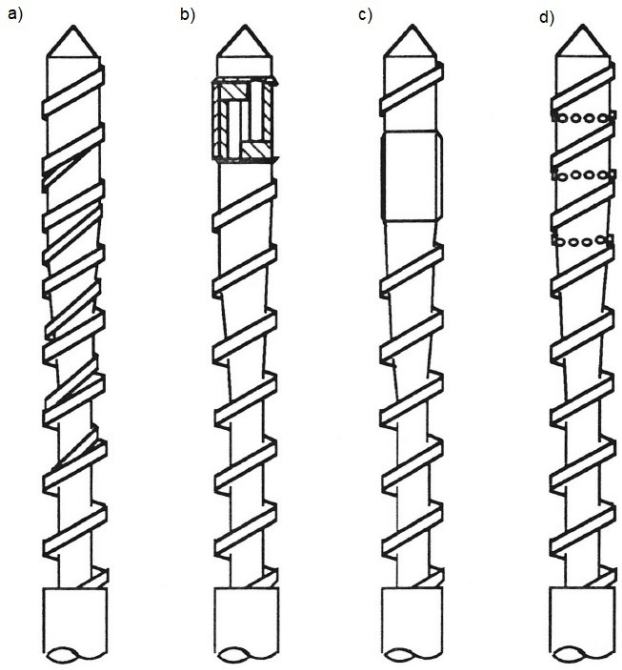

Figura 5: a) Parafuso com barreira; b) *Maddock mixing head*;c) *Shearing torpedo*; d) Parafuso com pinos. De Thibault et al. [3].

A adição de pinos ao parafuso foi uma das primeiras modificações feitas, visando a melhora da mistura, devido à facilidade com que os pinos podem ser inseridos radialmente ao parafuso. Tem a funcionalidade de interromper o fluxo laminar da resina e, consequentemente, dividir e recombinar linhas de fluxo [3]. A Figura 6 evidencia as linhas de fluxo distorcidas pela presença dos pinos de mistura.

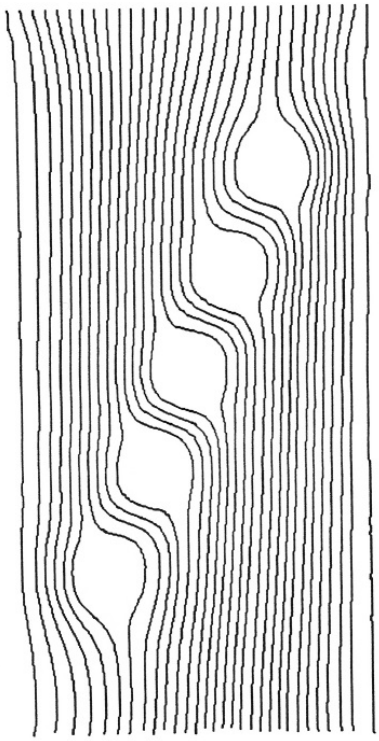

Figura 6: Linhas de corrente para parafuso com pinos de mistura [3].

2.2.1.1 Parafuso com barreira

O *design* do parafuso proposto na Figura 7 funciona de maneira eficiente para a maioria dos polímeros. Suas limitações vêm quando se trata de polímero de difícil fusão [1].

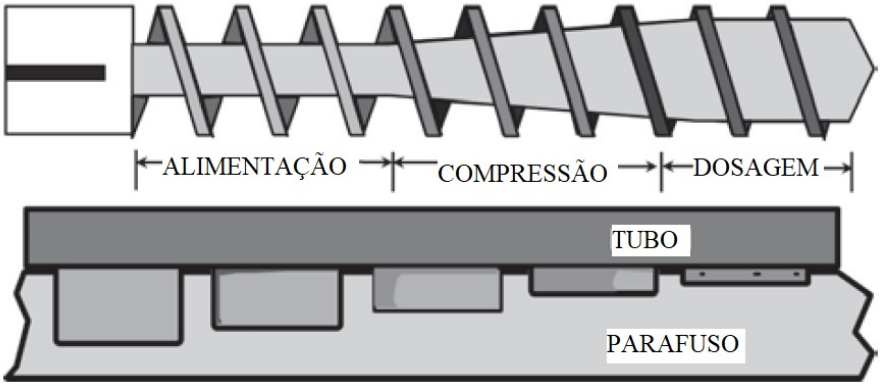

Figura 7: Padrão de derretimento de um parafuso extrusor padrão. De Giles et al. [1].

A capacidade de derretimento pode ser incrementada com a utilização de barreiras. A barreira de um parafuso extrusor consiste em uma segunda rosca de passo diferente. Existe defasagem entre as alturas das roscas principal e de barreira. A Figura 8 ilustra um parafuso com barreira e seu padrão de derretimento. A Figura 9 mostra uma patente de parafuso de barreira.

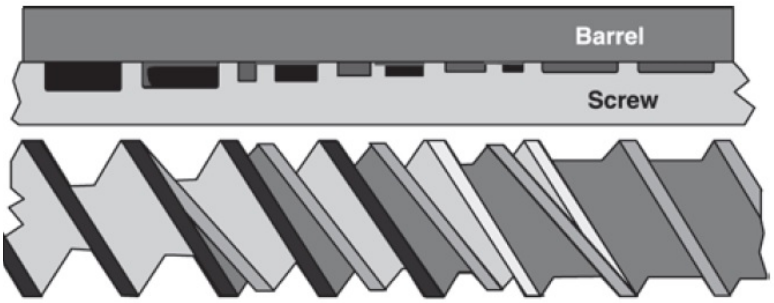

Figura 8: Padrão de derretimento de um parafuso com barreira [1].

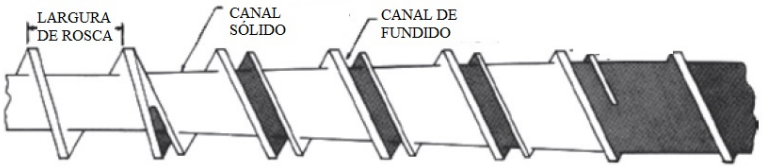

Figura 9: Parafuso *New Castles Industries* ^{TM} . Imagem adaptada [1].

A barreira, ou rosca secundária, tem o diâmetro ligeiramente menor que o diâmetro da rosca principal, de forma que o polímero escoe para dentro do canal da barreira, coletando e separando o material derretido do remanescente sólido. Enquanto isso ocorre, os efeitos viscosos e de condução térmica continuam a derreter o material em estado sólido. Não existe a possibilidade de grãos sólidos chegarem à zona de vazão da máquina, uma vez que toda a resina deve fluir por uma área estreita entre os canais principal e de barreira [1].

No próximo capítulo será feita uma análise das etapas de processamento propostas na Figura 7, dentro do sistema baseado no trabalho de Richard [4].

CAPITULO 3

MODELAGEM DO PROBLEMA

3.1 Modelo Físico

A modelagem física buscar simular, de forma simplificada, o fenômeno que ocorre no interior de uma máquina extrusora. Com esse propósito, o fluido foi tomado como não-Newtoniado com viscosidade regida pela lei de potência e a geometria do parafuso foi simplificada.

A lei de potência, também conhecida como lei de *Ostwald-de Waele*, é muito utilizada devido à sua simplicidade, para aproximar o comportamento de um fluido não Newtoniano. Se o índice da lei de potência for menor do que uma unidade, a viscosidade decai com o aumento da taxa de cisalhamento, indefinidamente, sendo necessário um fluido com viscosidade infinita no repouso e nenhuma viscosidade para uma alta taxa de cisalhamento. Porém, um fluido real tem tanto o limite superior quanto inferior para a viscosidade. A lei de potência descreve de maneira satisfatória o comportamento do fluido dentro da sua faixa de variação de viscosidade particular.

Um modelo desenvolvido para um fluido Newtoniano começa a apresentar erros severos para fluidos com $n < 0.8$, o que impossibilita a simulação da maior parte dos polímeros presentes no mercado (PE, PP, PVC, Poliéster, etc). Estes têm o valor de n abaixo de 0.5. Uma profundidade de canal otimizada para um fluido Newtoniano ($n = 1$) é cerca de 40 porcento mais largo do que a profundidade otimizada típica de um parafuso para polímeros ($n = 0.4$) [8]. Existem outros modelos que descrevem de forma mais aproximada o comportamento desses fluidos, mas o fazem ao custo da simplicidade e alta carga de processamento. A maior parte das máquinas extrusoras operam em uma rotação abaixo de 100 rotações por minuto, com tendência a reduzir a velocidade para polímeros com características pseudoplásticas.

No interior de uma máquina extrusora, o polímero percorre um caminho formado pelo espaço presente entre o parafuso e o tubo que envolve o mesmo, segundo a Figura 10.

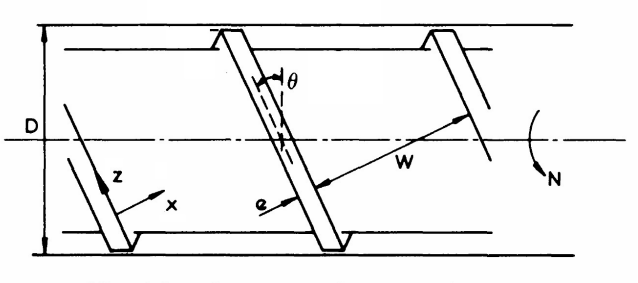

Figura 10: Geometria de um parafuso extrusor, de Richard [4].

Esse canal com formato helicoidal tem uma largura w, uma altura H e um ângulo de hélice θ. O canal é relativamente raso, em comparação com as medidas de diâmetro do parafuso e do tubo que o envolve. Desta forma, é razoável assumir que o canal pode ser desenrolado e tratado como retilinear, de acordo com o trabalho de e R.T Fenner [9]. O sistema de coordenadas cartesianas é adotado para esse problema, fixo a um parafuso estacionário. Desta forma, a parede que se desloca é a do tubo que envolve o parafuso. O canal helicoidal é desenrolado em um canal retilíneo, em que a parede superior se move diagonalmente, com componentes de velocidade nas direções x e z (Figura 11). O entendimento dessa aproximação é essencial para o entendimento do trabalho efetuado a seguir.

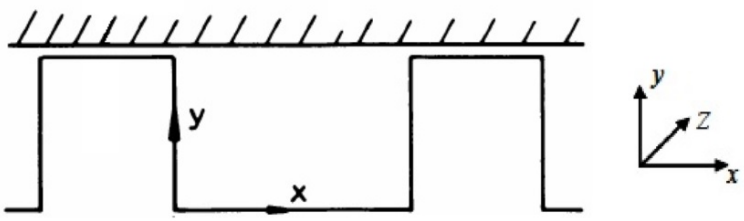

Figura 11: Vista do canal do parafuso. De Richard [4], com adaptações.

Com a simplificação proposta, chegamos a um problema semelhante ao da cavidade cisalhante, com vasta literatura para comparação. A tampa cisalhante se movimentaria de maneira diagonal sobre a cavidade, com uma componente em x e uma componente em z. Como o foco do trabalho está em obter os esforços aos quais o parafuso e o tubo estão sujeitos, a componente da velocidade em z - que possui magnitude consideravelmente inferior à componente de velocidade em x - é desconsiderada. O escoamento do fluido não Newtoniano pode ser descrito de forma satisfatória como bidimensional e não isotérmico, como proposto no algoritmo de Tadmor [10]. A Figura 12 ilustra as componentes de velocidades citadas em relação ao parafuso.

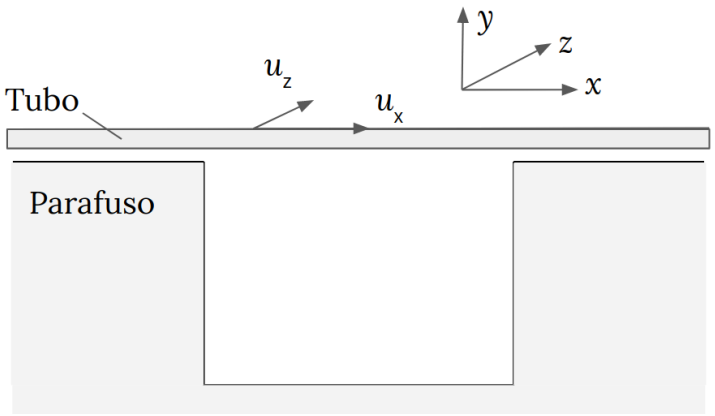

Figura 12: Velocidades da tampa da cavidade.

3.2 Modelo Matemático

Para estudar o comportamento do polímero, vamos considerar um escoamento bidimensional, em que o vetor velocidade é dada por:

$$\boldsymbol{u} = u\hat{\mathbf{e}}_x + v\hat{\mathbf{e}}_y. \tag{1}$$

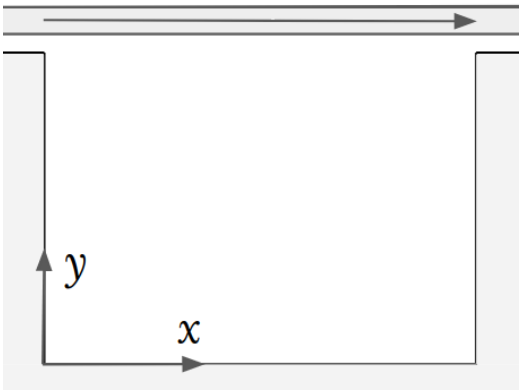

Figura 13: Cavidade e sistema de coordenadas.

As equações governantes são dadas por:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0. \tag{2}$$

$$\rho\frac{\partial u}{\partial t} + \rho u\frac{\partial u}{\partial x} + \rho v\frac{\partial u}{\partial y} = -\frac{\partial p}{\partial x} + \frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{xy}}{\partial y} \tag{3}$$

$$\rho\frac{\partial v}{\partial t} + \rho u\frac{\partial v}{\partial x} + \rho v\frac{\partial v}{\partial y} = -\frac{\partial p}{\partial x} + \frac{\partial \tau_{yx}}{\partial x} + \frac{\partial \tau_{yy}}{\partial y} \tag{4}$$

em que $\boldsymbol{\tau}$ é o tensor de tensões viscosas:

$$\boldsymbol{\tau} = \begin{bmatrix} \tau_{xx} & \tau_{xy} \\ \tau_{yx} & \tau_{yy} \end{bmatrix} \tag{5}$$

As componentes do tensor são:

$$\boldsymbol{\tau} = \eta(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T) \tag{6}$$

$$\tau_{xx} = 2\eta\frac{\partial u}{\partial x} \tag{7}$$

$$\tau_{yy} = 2\eta\frac{\partial v}{\partial y} \tag{8}$$

$$\tau_{xy} = \tau_{yx} = \eta\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) \tag{9}$$

em que η é a viscosidade do fluidos.

Na modelagem da viscosidade através de uma lei de potência, η é uma função da taxa de cisalhamento, e é dada por

$$\eta = K|\dot{\gamma}|^{(n-1)}, \tag{10}$$

em que K é o índice de consistência do escoamento, $\dot{\gamma}$ é a segunda invariante do tensor da taxa de deformação e n o índice de comportamento do fluido. As unidades de medida, no Sistema Internacional, de η, $\dot{\gamma}$ e de K são:

$$[\eta] = Pa \cdot s \tag{11}$$

$$[\dot{\gamma}] = s^{-1} \tag{12}$$

$$[K] = Pa \cdot s \cdot s^{n-1} = \frac{kg \cdot s^{2-n}}{m} \tag{13}$$

Substituindo as componentes do tensor τ nas equações 3 e 4, obtemos:

$$\rho\frac{\partial u}{\partial t} + \rho u\frac{\partial u}{\partial x} + \rho v\frac{\partial u}{\partial y} = -\frac{\partial p}{\partial x} + \frac{\partial}{\partial x}\left(2\eta\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(\eta\frac{\partial u}{\partial y} + \eta\frac{\partial v}{\partial x}\right) \tag{14}$$

$$\rho\frac{\partial v}{\partial t} + \rho u\frac{\partial v}{\partial x} + \rho v\frac{\partial v}{\partial y} = -\frac{\partial p}{\partial y} + \frac{\partial}{\partial x}\left(\eta\frac{\partial u}{\partial y} + \eta\frac{\partial v}{\partial x}\right) + \frac{\partial}{\partial y}\left(2\eta\frac{\partial v}{\partial y}\right) \tag{15}$$

O comportamento de um fluido não Newtoniano pode ser simulado de uma forma muito aproximada usando a equação 10. O índice de comportamento n é responsável por definir a natureza da viscosidade. Caso o índice n assuma um valor no intervalo $0 < n < 1$, seu comportamento é pseudoplástico, caso o valor de n esteja no intervalo $n > 1$, o comportamento é dilatante. É comum encontrar os últimos termos citados escritos em inglês, estes são *shear-thinning* e *shear-thickening*, respectivamente. A maior parte dos polímeros utilizados na indústria tem o valor de seu índice de comportamento próximo de 0.4.

Para um escoamento bidimensional, a taxa de cisalhamento é dada por:

$$|\dot{\gamma}| = \sqrt{2\left(\frac{\partial u}{\partial x}\right)^2 + 2\left(\frac{\partial v}{\partial y}\right)^2 + \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)^2}. \tag{16}$$

Para adimensionalizar as equações 2, 14 e 15, as variáveis foram adimensionalizadas como proposto:

$$x^* = \frac{x}{W}; \quad y^* = \frac{y}{H}; \quad p^* = \frac{p}{\rho U^2}; \quad t^* = \frac{tU}{L}; \quad u^* = \frac{u}{U}; \quad v^* = \frac{v}{U}; \tag{17}$$

$$\dot{\gamma}^* = \frac{\dot{\gamma}L}{U}; \quad Re = \frac{\rho U^{2-n} L^n}{K}; \tag{18}$$

em que W é a largura da cavidade, H a altura da cavidade e U a velocidade característica da tampa. Após a adimensionalização, a o termo da viscosidade passa a ser representado por D, expresso por:

$$D = \mu_0 \dot{\gamma}^{n-1} \tag{19}$$

em que μ_0 é a viscosidade característica.

Usando as variáveis descritas acima, retirando a notação utilizada para indicar a forma adimensional (*), a forma adimensional das equações governantes podem ser escritas como

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0, \tag{20}$$

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{\partial p}{\partial x} + \frac{1}{Re}\frac{\partial}{\partial x}\left(2D\frac{\partial u}{\partial x}\right) + \frac{1}{Re}\frac{\partial}{\partial y}\left(D\frac{\partial u}{\partial y}\right) + \frac{1}{Re}\frac{\partial}{\partial y}\left(D\frac{\partial v}{\partial x}\right), \tag{21}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{\partial p}{\partial y} + \frac{1}{Re}\frac{\partial}{\partial x}\left(D\frac{\partial v}{\partial x}\right) + \frac{1}{Re}\frac{\partial}{\partial y}\left(2D\frac{\partial v}{\partial y}\right) + \frac{1}{Re}\frac{\partial}{\partial x}\left(D\frac{\partial u}{\partial y}\right), \tag{22}$$

Seguindo o modelo de lei de potência proposto no trabalho de Molla and Yao [11], devem ser estabelecidos limites para os valores da viscosidade. Isso é necessário pois quando a viscosidade assume valores muito próximos a zero ou tendendo ao infinito, podem ocorrer inconsistências na resolução numérica. A forma como os limites são aplicados é dada por:

$$D = \begin{cases} 1 & \text{se} \quad |\dot{\gamma}| < \gamma_1, \\ |\dot{\gamma}|^{n-1} & \text{se} \quad \gamma_1 \leq |\dot{\gamma}| \leq \gamma_2, \\ \gamma_2^{n-1} & \text{se} \quad |\dot{\gamma}| > \gamma_2. \end{cases} \tag{23}$$

As constantes γ_1 e γ_2 são os limites do intervalo proposto para a taxa de cisalhamento. Os seus valores são $\gamma_1 = 10^{-2}$ e $\gamma_2 = 10^5$. A equação 23 mostra que D deve estar compreendido no intervalo entre os limites apresentados para a taxa de cisalhamento.

Com os dados das velocidades, é possível solucionar as equações da linha de corrente. Estas equações são

$$u = \frac{\partial \psi}{\partial y}, \tag{24}$$

$$v = \frac{\partial \psi}{\partial x} \tag{25}$$

e

$$\nabla^2 \psi = -(\boldsymbol{\nabla} \times \mathbf{u}) \cdot \hat{\mathbf{e}}_{\boldsymbol{z}}. \tag{26}$$

Portanto, neste trabalho serão utilizadas as equações 20, 21 e 22 na resolução do problema do escoamento do fluido na cavidade. Os detalhes com relação à implementação numérica dessa solução são apresentados no próximo capítulo.

CAPITULO 4

METODOLOGIA

O método de projeção é um meio efetivo para a resolução numérica do escoamento de fluidos incompressíveis em regime transiente. Foi inicialmente introduzida por Chorin [7] em 1967. A vantagem do método está em solucionar os campos de velocidade e pressão de forma desacoplada.

4.1 Método de Projeção de Chorin

O algoritmo do método de projeção é baseado na decomposição de Helmholtz (ou Helmholtz-Hodge) de qualquer campo vetorial em uma parte solenoidal e uma parte irrotacional. Usualmente aplicado em dois estágios. No primeiro estágio, a velocidade intermediária u^* - que não satisfaz a condição de incompressibilidade - é calculada para cada passo de tempo. Na segunda etapa, a pressão é utilizada para projetar a velocidade intermediária em um campo de velocidades livre do divergente, para obter a próxima atualização de velocidade e pressão. Os primeiros passos da aplicação do algoritmo, nas equações governantes, pode ser observado nas Equações 27, 31 e 32.

$$\frac{u^* - u^k}{dt} + u^k \cdot \nabla u^k = \frac{1}{Re}\frac{\partial}{\partial x}\left(2D\frac{\partial u}{\partial x}\right)^k + \frac{1}{Re}\frac{\partial}{\partial y}\left(D\frac{\partial u}{\partial y}\right)^k + \frac{1}{Re}\frac{\partial}{\partial y}\left(D\frac{\partial v}{\partial x}\right)^k \tag{27}$$

O índice k indica o "passo"no tempo. Esta equação permite o cálculo da velocidade $\mathbf{u}^*$. Temos também que:

$$\mathbf{u}^* = \mathbf{u_b} \tag{28}$$

em $\partial\Omega$. Isto é, a velocidade nas fronteiras da cavidade é igual a velocidade da fronteira.

As equações a seguir são utilizadas para calcular o termo da pressão utilizando a velocidade intermediária u^*.

$$\mathbf{u}^* = \mathbf{u}^{k+1} + \Delta t \nabla p^{k+1} \tag{29}$$

Para garantir que o vetor $\mathbf{u}^{k+1}$ seja solenoidal, a relação a seguir deve ser atendida.

$$\nabla \cdot \mathbf{u}^{k+1} = 0 \tag{30}$$

Aplicando o divergente na Equação 29, e utilizando a Equação 30 para zerar o termo que em a velocidade está um passo à frente no tempo, chegamos a:

$$\nabla^2 p^{k+1} = \frac{1}{\Delta t} \nabla \cdot \mathbf{u}^* \tag{31}$$

Nessas duas primeiras etapas da aplicação do método, é possível obter a velocidade intermediária $\mathbf{u}^*$ e o campo de pressão para utilizar nos passos seguintes. O termo da velocidade que é relacionado a um "passo"a mais no tempo, $\mathbf{u}^{k+1}$, é calculado então através da Equação 32.

$$u^{k+1} = u^* - \Delta t \nabla p^{k+1}. \tag{32}$$

A forma de aplicação do método proposto acima, dentro do modelo criado, pode ser observado no código em anexo.

4.2 Condições de Contorno

As condições de contorno são de extrema importância, influenciando a acurácia e estabilidade do método numérico. Para a velocidade, a condição de não deslizamento é considerada, e pode ser expressa por

$$v(x = 0, y) = 0, \quad v(x = 1, y) = 0, \quad u(x, y = 0) = 0, \quad u(x, y = 1) = u_0. \tag{33}$$

A condição de contorno de Neumann é utilizada, de forma que a derivada normal da pressão é nula em todas as paredes, i.e.

$$\frac{\partial p}{\partial \mathbf{n}} = 0 \tag{34}$$

em que $\mathbf{n}$ é o vetor normal às superfície.

Uma representação da cavidade e a origem do sistema de coordenadas adotado podem ser visualizados na Figura 14.

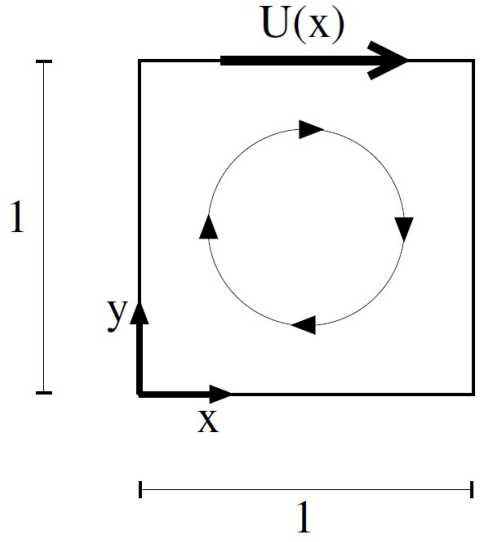

Figura 14: Escoamento do polímero em uma cavidade movida.

Para as linhas de corrente, as condições de contorno são

$$\psi(0,y) = \psi(1,y) = \psi(x,0) = \psi(x,1) = 0. \tag{35}$$

4.3 Implementação Numérica

A implementação numérica foi feita usando o método da projeção em conjunto com uma malha defasada. Na malha defasada, as velocidades nos eixos x e y, u e v respectivamente, e a pressão, p, são posicionados em diferentes lugares, seguindo a Figura 15.

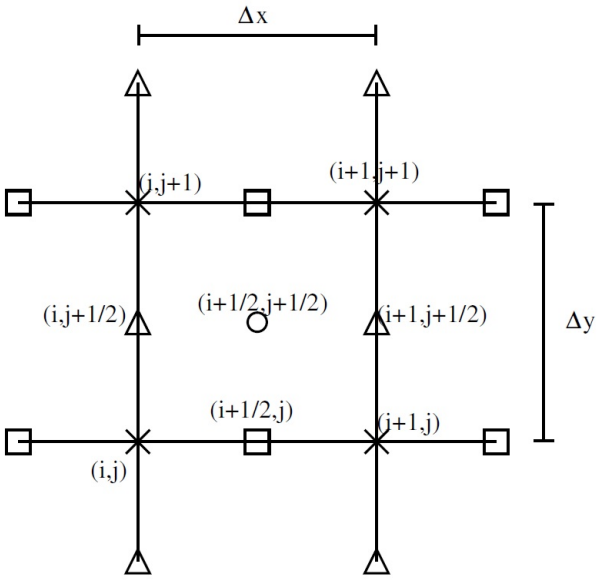

Figura 15: Célula da malha defasada. Triângulo: u. Quadrado: v. Círculo: p.

O modelo é baseado nas três principais matrizes, u, v and p. As dimensões de cada matriz são: $u[0:nx, -1:ny]$, $v[-1:nx, 0:ny]$, $p[-1:nx, -1:ny]$. A posição física de u, está no ponto $u(x,y) = u(i\Delta x, j\Delta y + \frac{1}{2}\Delta y)$, e, a posição análoga na matriz u seria $u[i,j]$. Desta forma, a relação com as outras variáveis podem ser expressas seguindo o padrão indicado abaixo.

$$u[i,j] \leftarrow u^k_{i,j+\frac{1}{2}} \tag{36}$$

$$v[i,j] \leftarrow v^k_{i+\frac{1}{2},j} \tag{37}$$

$$u_{star}[i,j] \leftarrow u^*_{i,j+\frac{1}{2}} \tag{38}$$

$$v_{star}[i,j] \leftarrow v^*_{i+\frac{1}{2},j} \tag{39}$$

$$u_{new}[i,j] \leftarrow u^{k+1}_{i,j+\frac{1}{2}} \tag{40}$$

$$v_{new}[i,j] \leftarrow v^{k+1}_{i+\frac{1}{2},j} \tag{41}$$

$$p[i,j] \leftarrow p^k_{i+\frac{1}{2},j+\frac{1}{2}} \tag{42}$$

$$D[i,j] \leftarrow D^k_{i+\frac{1}{2},j+\frac{1}{2}} \tag{43}$$

A forma adimensional da viscosidade, D, ocupa o mesmo local da pressão na malha. Apesar de ocupar o mesmo local da pressão, os valores de D foram interpolados durante o cálculo das derivadas (linha 153 da sessão de funções do programa A.2.2). As dimensões de cada célula podem ser definidas por $dx = \frac{1}{nx}$ e $dy = \frac{1}{ny}$, as constantes nx e ny representam o número de partições da malha, no eixo x e y, respectivamente.

É possível notar que existem valores de u, v e p fora do domínio físico. Esses pontos são denominados pontos fantasmas, ou *ghostpoints*, e são usados para impor as condições de contorno.

As equações a seguir mostram o cálculo para as paredes superior e inferior, para u, e das paredes direita e esquerda, para v, usando os pontos fantasma. Para os outros casos, não existem pontos fantasma, os outros contornos podem ser calculados de forma direta, dada pelos componentes que já existem sobre a parede.

$$\frac{u[i,ny+1] - u[i,ny]}{2} = u_0 \tag{44}$$

$$\frac{u[i,0] - u[i,-1]}{2} = 0 \tag{45}$$

$$\frac{v[-1,j] - v[0,j]}{2} = 0 \tag{46}$$

$$\frac{v[nx+1,j] - v[nx,j]}{2} = 0 \tag{47}$$

Os valores de pressão nos contornos é calculado usando os pontos fantasmas, como exemplificado abaixo, para a parede esquerda.

$$\frac{p[-1,j] - p[0,j]}{dx} = 0. \tag{48}$$

A matriz da linha de corrente é calculada a partir da equação 26 e é iterada entre $1 \leq i \leq nx-1$ e $1 \leq j \leq ny-1$. As condições de contorno adotadas podem ser vistas na Sessão 4.2 do trabalho.

4.4 Modelo Computacional

O código criado foi comparado com outros trabalhos presentes na literatura, com a finalidade de observar a confiabilidade da ferramenta no estudo do comportamento do fluido. A geometria da cavidade, a velocidade com que a tampa se move e a natureza da viscosidade são dados de entrada do código. Após o desenvolvimento do escoamento por determinado tempo, o programa retorna os dados relativos aos campos de velocidade, pressão e taxa de cisalhamento. O programa foi feito em *Python 3*, com auxilio da ferramenta *Numba*.

O método de projeção de primeira ordem é usado neste papel, tem um erro de primeira ordem no tempo e de segunda ordem no espaço. A maior parte dos resultados apresentados neste trabalho foram calculados para um tempo de desenvolvimento de 60 em uma malha de

129 por 129 pontos. A ferramenta *streamplot* do *Python* apresenta as linhas pretas com setas, já as cores são calculadas de forma direta, através das equações de linhas de corrente.

Para ilustrar a lógica seguida dentro do código, a Figura 16 foi elaborada.

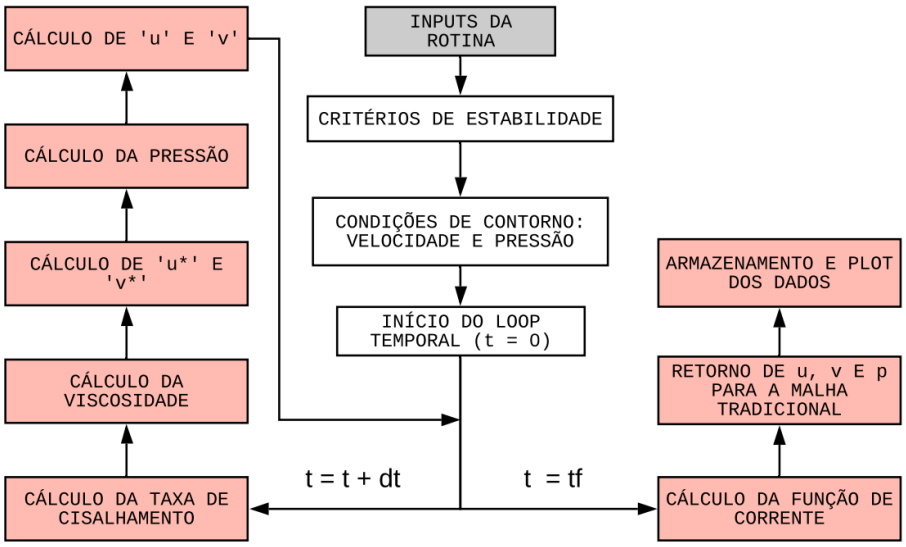

Figura 16: Fluxograma do código.

Na primeira etapa, deve-se estabelecer valores para as dimensões da cavidade nos eixos x e y, assim como o seu fracionamento. Deve-se também escolher a velocidade do escoamento, ditada pelo número de Reynolds (Re). O intervalo de tempo equivalente ao desenvolvimento do escoamento é compreendido entre o tempo inicial (t) e o tempo final (t_f) escolhidos. Na etapa seguinte, restrições relacionadas à estabilidade do código (Eq. 23,) e as condições de contorno (Eq. 33) são aplicadas. As condições de estabilidade são fundamentais para poder utilizar o código para uma ampla faixa de valores para Re, nx e ny.

Após essas duas etapas preliminares, é iniciado um *loop* temporal que irá seguir a lógica proposta no fluxograma presente na Figura 16. Quando o tempo final é atingido, as linhas de corrente são calculadas e as velocidades e pressão devem retornar à malha tradicional.

CAPITULO 5

RESULTADOS

Neste trabalho, o escoamento de um fluido não Newtoniano incompressível em uma cavidade com a tampa se movendo é solucionado com o uso do método da projeção. O comportamento do escoamento é fortemente afetado pelo número de Reynolds e n (lei de potência). Os padrões de recirculação são uma forma qualitativa de comparação entre os dados obtidos e a literatura. Este capítulo terá duas sessões principais, uma com o comparativo entre os resultados obtidos para um fluido Newtoniano e literatura; outra com o comparativo entre os resultados obtidos para um fluido não Newtoniano e literatura.

5.1 Validação

A primeira etapa de validação foi feita adotando características Newtonianas para o fluido ($n = 1$). A simulação foi feita para um tempo final de 60 e o tamanho da malha foi de $129x129$. Após a comparação qualitativa, foi feita uma comparação quantitativa, utilizando os dados das velocidades u e v, fornecidos por Ghia [5], para as linhas médias vertical e horizontal, respectivamente. As linhas de corrente obtidas para um número de Reynolds igual a 100 são mostradas na Figura 17. A Figura 18 mostra o resultado de Ghia [5] para o mesmo valor de Reynolds.

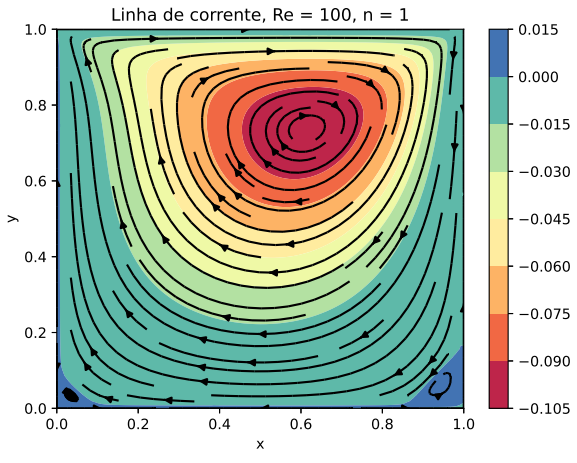

Figura 17: Simulação numérica, modelagem para fluido Newtoniano. $Re = 100$.

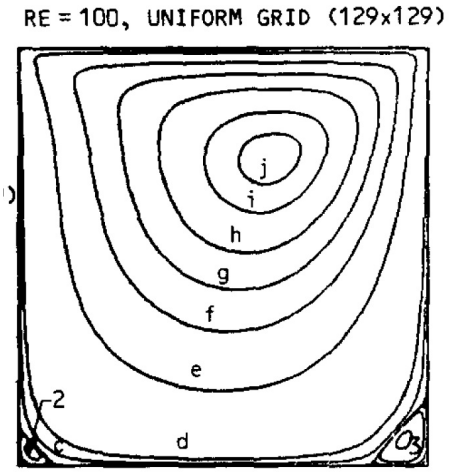

Figura 18: Cavidade movida, fluido Newtoniano, $Re = 100$, imagem de Ghia [5].

É possível ver o surgimento de uma zona de recirculação nos cantos inferiores da caixa, assim como uma curvatura das linhas de corrente se acentuando à medida que se aproxima da tampa, nas proximidades da parede esquerda.

A comparação quantitativa é feita utilizando as velocidades de u ao longo da linha vertical que passa pelo centro da cavidade e as velocidades de v ao longo da linha horizontal que passa pelo centro da cavidade. As Figuras 19 e 20 mostram as curvas geradas pelos dados em comparação e o erro entre elas, para velocidades na direção x. As figuras 21 e 22 fazem o mesmo, para velocidades na direção y

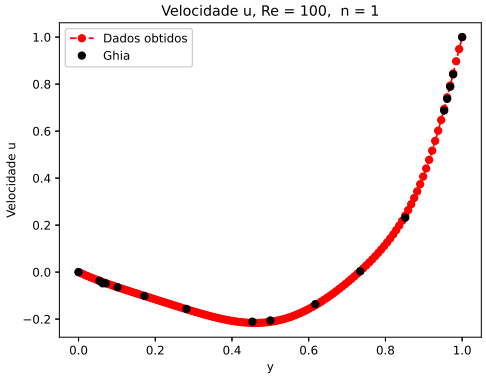

Figura 19: Velocidade u na linha vertical centrada. $Re = 100$ e $n = 1.0$.

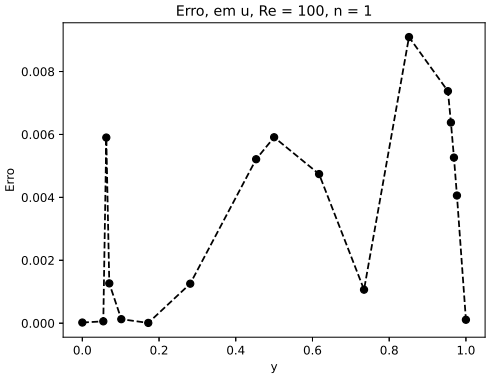

Figura 20: Diferença entre as curvas apresentadas (ou erro).

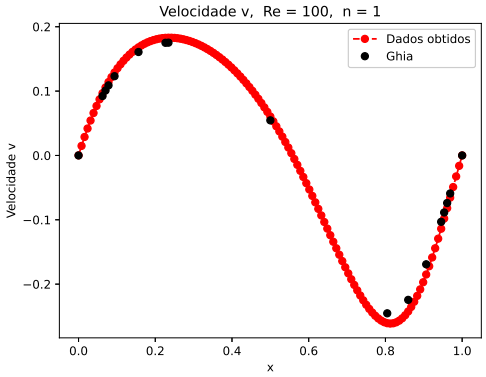

Figura 21: Velocidade v na linha horizontal centrada. $Re = 100$ e $n = 1.0$.

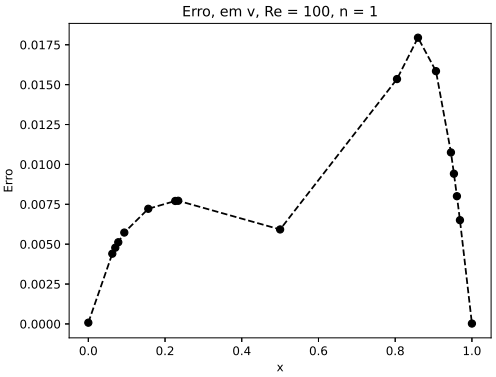

Figura 22: Diferença entre as curvas apresentadas (ou erro).

O mesmo procedimento foi feito para o valor de Reynolds igual à 400. As Figuras 23 e 24 mostram as linhas de corrente obtidas e da literatura, respectivamente, para uma comparação qualitativa.

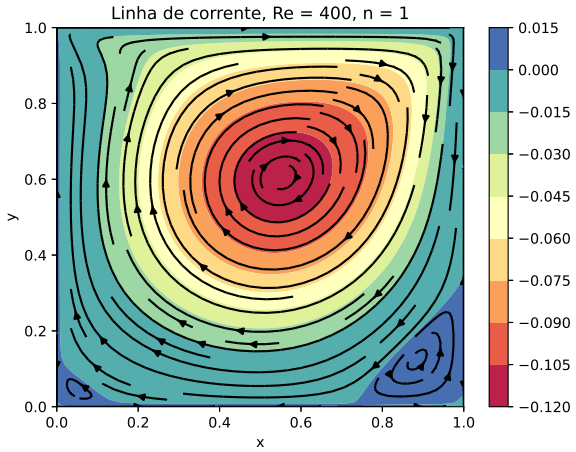

Figura 23: Simulação numérica, modelagem para fluido Newtoniano. $Re = 400$.

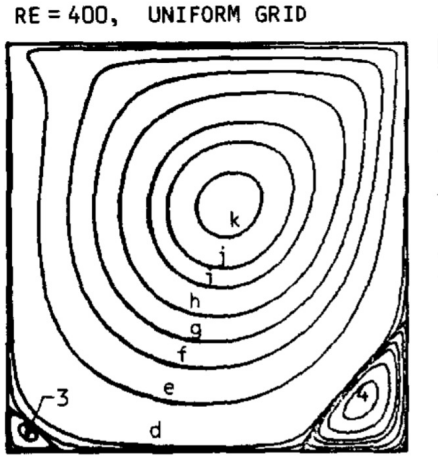

Figura 24: Cavidade movida, fluido Newtoniano, $Re = 400$, imagem de Ghia [5].

As Figuras 25 e 26 mostram o comparativo entre as velocidades obtida e da literatura, na direção x. As Figuras 27 e 28 mostram o comparativo na direção y.

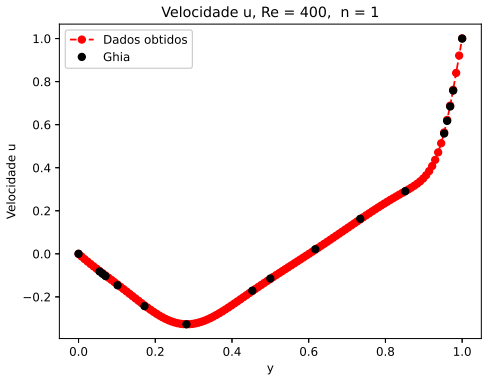

Figura 25: Velocidade u na linha vertical centrada. $Re = 400$ e $n = 1.0$.

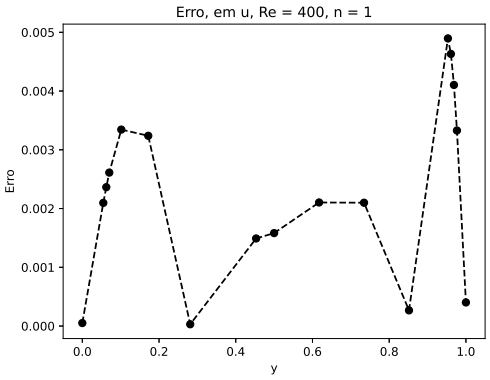

Figura 26: Diferença entre as curvas apresentadas (ou erro).

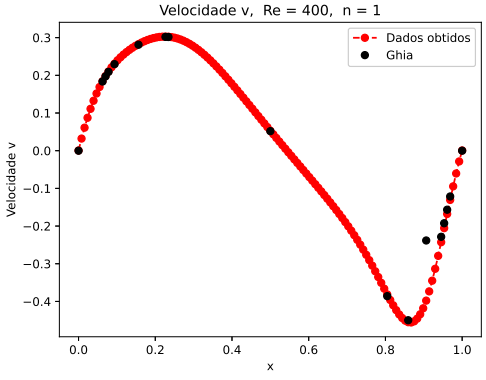

Figura 27: Velocidade v na linha horizontal centrada. $Re = 400$ e $n = 1.0$.

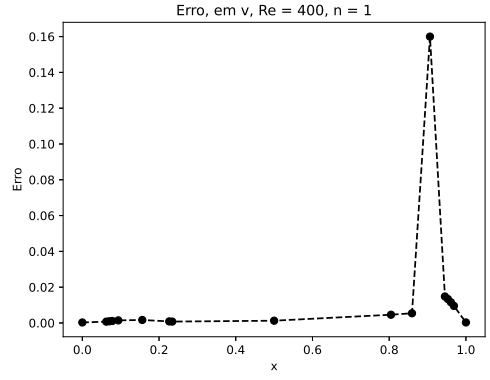

Figura 28: Diferença entre as curvas apresentadas (ou erro).

As Figuras 29 e 30 mostram a comparação qualitativa entre as linhas de corrente para o valor de Reynolds igual à 1000.

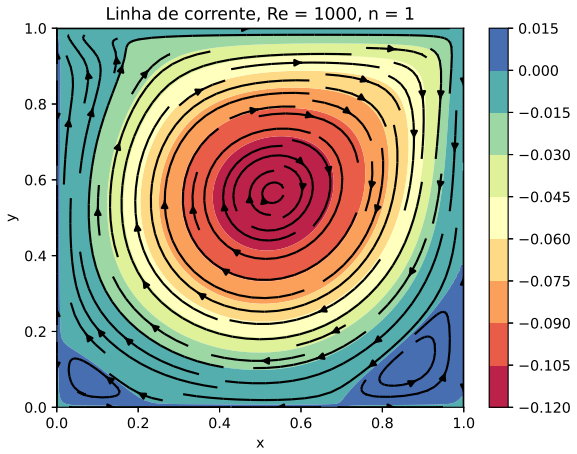

Figura 29: Simulação numérica, modelagem para fluido Newtoniano. $Re = 1000$.

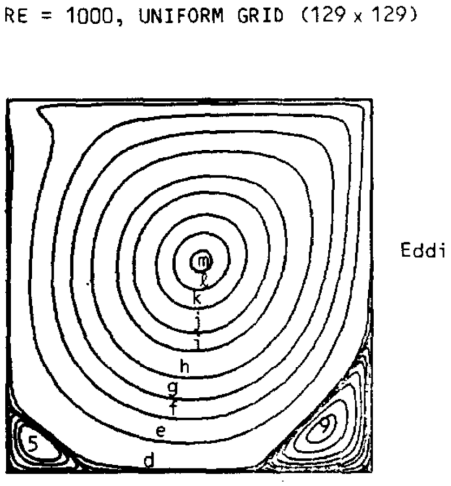

Figura 30: Cavidade movida, fluido Newtoniano, $Re = 1000$, imagem de Ghia [5].

As Figuras 31 e 32 mostram o comparativo entre as velocidades obtida e da literatura, na direção x. As Figuras 33 e 34 mostram o comparativo na direção y.

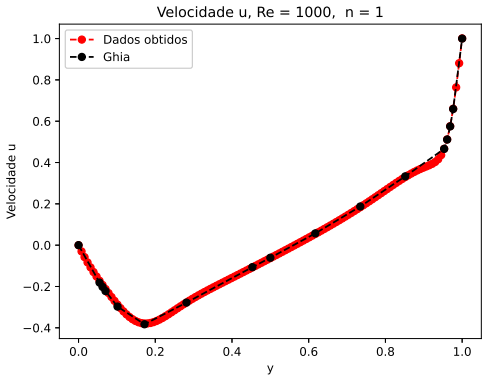

Figura 31: Velocidade u na linha vertical centrada. $Re = 1000$ e $n = 1.0$.

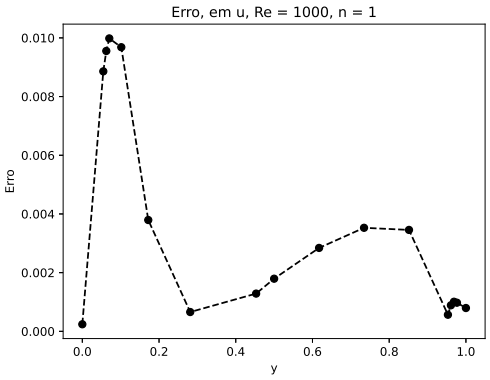

Figura 32: Diferença entre as curvas apresentadas (ou erro).

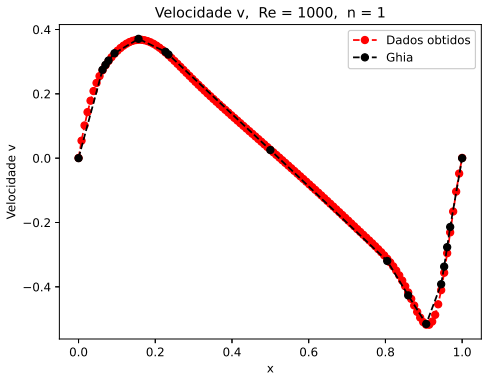

Figura 33: Velocidade v na linha horizontal centrada. $Re = 1000$ e $n = 1.0$.

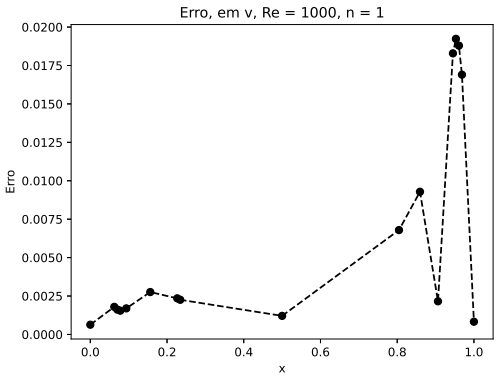

Figura 34: Diferença entre as curvas apresentadas (ou erro).

O código apresentou bons resultados, com resultados satisfatoriamente próximos dos apresentados em literatura. A Figura 27 apresenta um ponto deslocado - na curva de dados fornecidos por Ghia [5] - possivelmente um erro durante a apresentação dos resultados. Desta forma, o programa está validado e apto a ser utilizado na simulação de fluidos não Newtonianos.

5.2 Fluido não Newtoniano

A variável que rege o comportamento do fluido como dilatante ou pseudoplástico, dentro da lei de *Ostwald-de Waele*, é representada por n. Nessa seção do trabalho podemos observar o comportamento do fluido para diferentes número de Reynolds e valores de n e um comparativo com literatura é feito.

5.2.1 Re = 100

As simulações apresentadas nessa sessão foram feitas para um número de Reynolds igual a 100 e o valor de n assume os valores de 0.5 e 1.5. A Figura 35 mostra o resultado encontrado para o valor de Reynolds igual a 100 e $n = 0.5$. A Figura 36 mostra o resultado obtido por Li et al. [6], para os mesmos valores de Re e n.

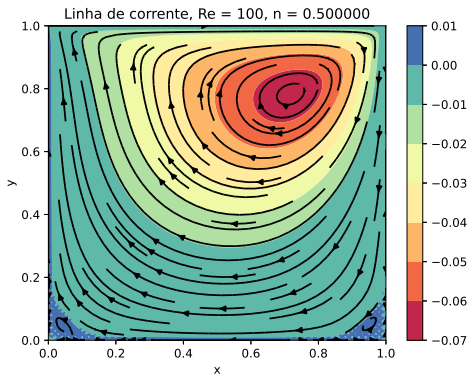

Figura 35: Linhas de corrente, $Re = 100$, $n = 0.5$.

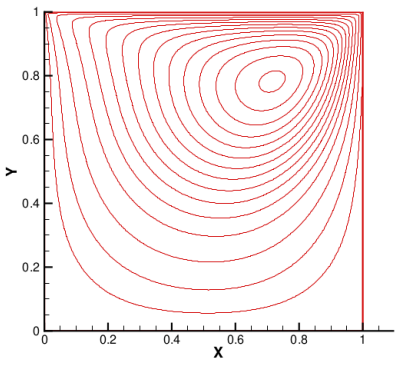

Figura 36: Linhas de corrente, $Re = 100$ $n = 0.5$. [6]

A partir das linhas de corrente apresentadas, é possível ver um deslocamento da recirculação principal da cavidade, em relação aos resultados obtidos para um fluido não Newtoniano e mesmo número de Reynolds (Fig. 17). É possível observar o deslocamento na direção do canto superior direito. A recirculação diminui, no entanto, se observada a magnitude dos valores apresentados.

A comparação entre as velocidades u e v nas linhas vertical e horizontal que passam pelo centro, é feita com o trabalho de Li et al. [6]. A Figura 37 e 38 mostram os resultados para u e v respectivamente.

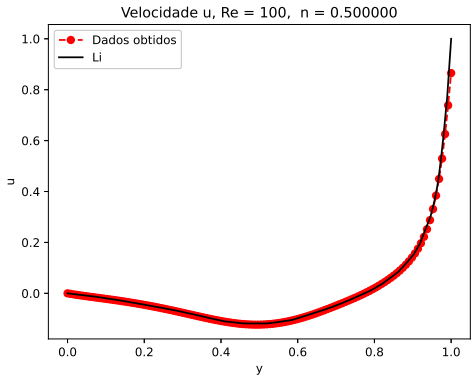

Figura 37: Velocidade u na linha vertical centrada. $Re = 100$ e $n = 0.5$. [6]

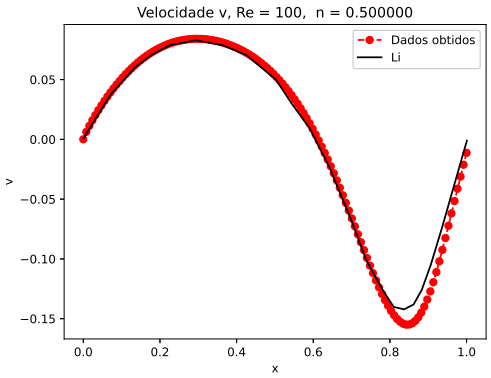

Figura 38: Velocidade v na linha horizontal centrada. $Re = 100$ e $n = 0.5$. [6]

A diferença observada entre os resultados obtidos e os resultados fornecidos em literatura decai com o aumento do número de Reynolds. As curvas com dados referentes à literatura foram retiradas das imagens presentes no trabalho de Li et al. [6] - com auxílio da ferramenta *plotdigitizer* do *Python 3* - de forma que há um erro associado à obtenção dos pontos.

A comparação entre as linhas de corrente obtidas, para $n = 1.5$, e da literatura, pode ser feita com auxílio das Figuras 39 e 40.

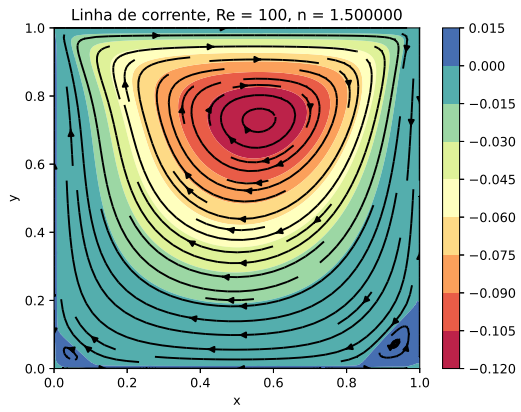

Figura 39: Linhas de corrente, $Re = 100$ $n = 1.5$.

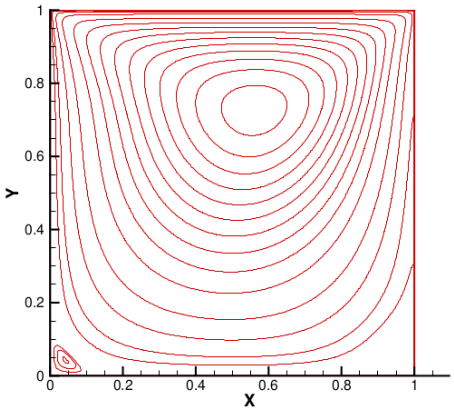

Figura 40: Linhas de corrente, $Re = 100$ $n = 0.5$. [6]

Com a característica de fluido dilatante, a recirculação se mostrou mais próxima do centro se comparada com os resultados obtidos para um fluido Newtoniano (17). Como o fluido se torna mais viscoso em zonas com alta taxa de cisalhamento, é esperado que a recirculação não

se aproxime da zona no canto superior direito. Essa região concentra grandes valores para a taxa de cisalhamento, apresentados no Anexo A.

A comparação entre as velocidades u e v nas linhas médias da cavidade são comparadas nas Figuras 41 e 42.

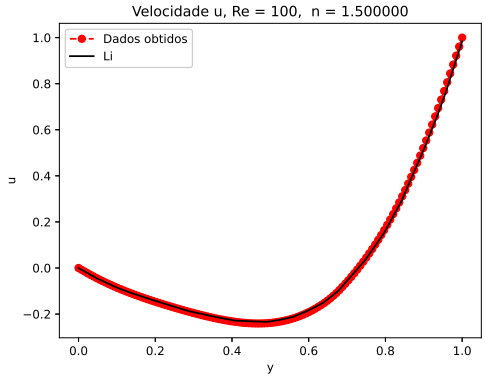

Figura 41: Velocidade u na linha vertical centrada. $Re = 100$ e $n = 1.5$. [6]

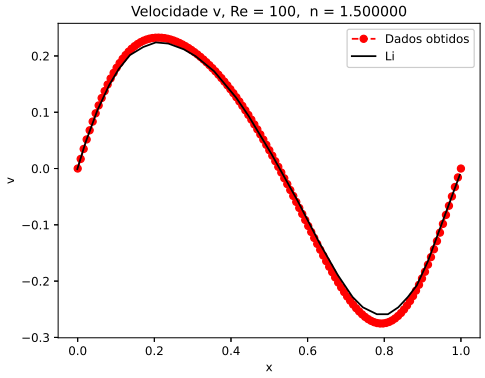

Figura 42: Velocidade v na linha horizontal centrada. $Re = 100$ e $n = 1.5$. [6]

5.2.2 Re = 500

A Figura 43 mostra o resultado encontrado para o valor de $Re = 500$ e $n = 0.5$. A Figura 44 mostra o resultado obtido por Li et al. [6], para os mesmos valores de Re e n.

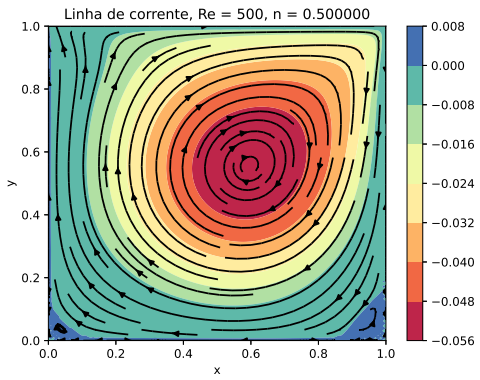

Figura 43: Linhas de corrente, $Re = 500$, $n = 0.5$.

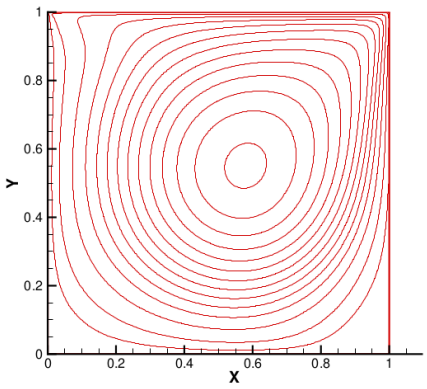

Figura 44: Linhas de corrente, $Re = 500$ $n = 0.5$. [6]

Nas Figuras 45 e 46 é possível ver o comparativo entre os dados obtidos e o da literatura.

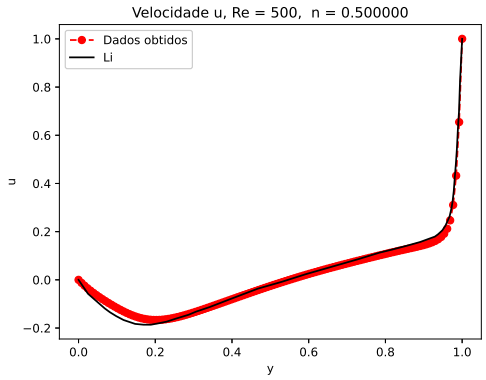

Figura 45: Velocidade u na linha vertical centrada. $Re = 500$ e $n = 0.5$. [6]

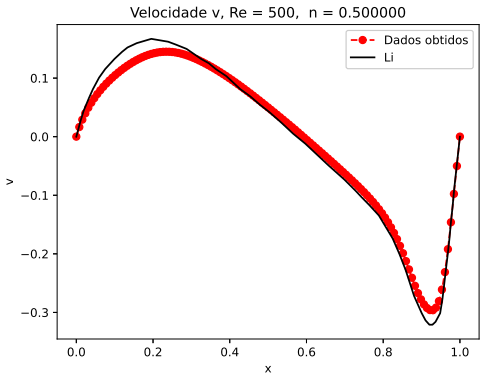

Figura 46: Velocidade v na linha horizontal centrada. $Re = 500$ e $n = 0.5$. [6]

A comparação entre os dados obtidos e o trabalho de Li et al. [6] pode ser observado a seguir, para o valor de $n = 1.5$ e $Re = 500$.

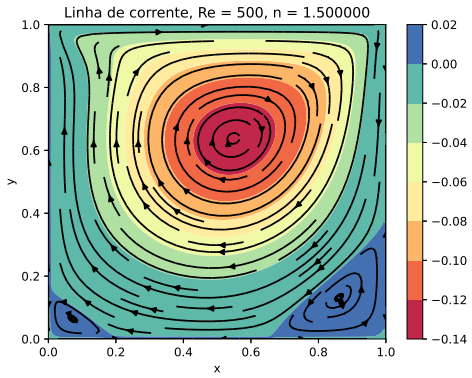

Figura 47: Linhas de corrente. $Re = 500$ e $n = 1.5$. [6]

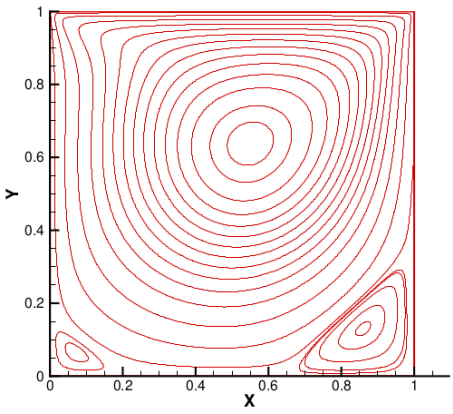

Figura 48: Linhas de corrente. $Re = 500$ e $n = 1.5$. [6]

Quando comparamos as linhas de corrente das Figuras 43 e 47, observamos padrões de recirculação bem diferentes apesar do mesmo número de Reynolds. O fluido com característica dilatante mostrou uma tendência a gerar recirculações maiores nos cantos inferiores da cavidade.

A comparação entre dados é feita, para $n = 1.5$, nas Figuras 49 e 50, em u e v, respectivamente.

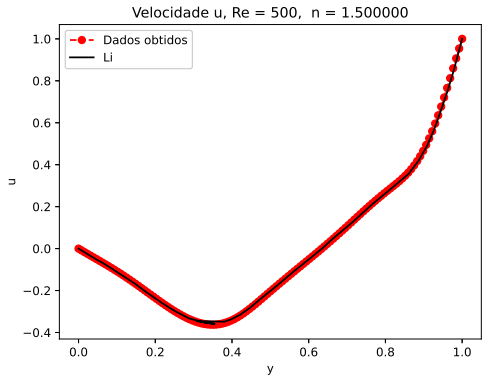

Figura 49: Velocidade u na linha vertical centrada. $Re = 500$ e $n = 1.5$. [6]

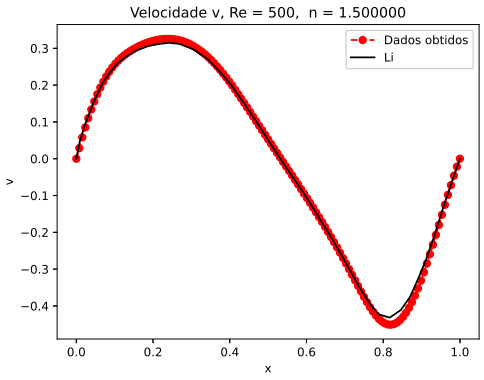

Figura 50: Velocidade v na linha horizontal centrada. $Re = 500$ e $n = 1.5$. [6]

5.2.3 Re = 1000

Nesta sessão, os resultados obtidos para um número de Reynolds de valor igual a 1000 podem ser observados. A Figura 51 mostra as linhas de corrente obtidas para o valor de $n = t0.5$. A Figura 53 mostra as linhas de corrente obtidas para um valor de $n = 1.5$. O trabalho usado para comparação não fornece resultados para este número de Reynolds e $n = 1.5$.

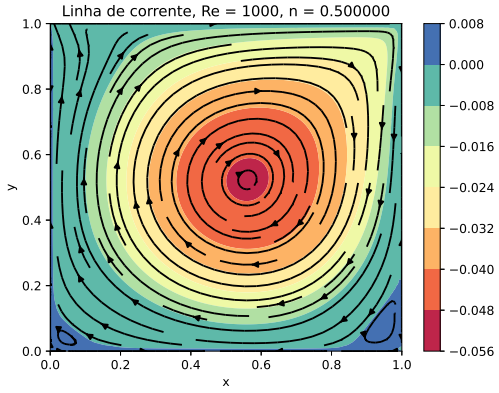

Figura 51: Linhas de corrente, $Re = 1000$, $n = 0.5$.

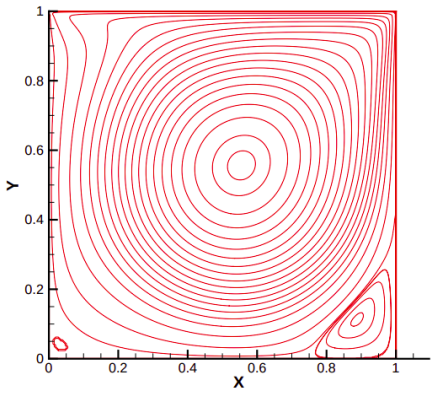

Figura 52: Linhas de corrente, $Re = 1000$, $n = 0.5$. [6]

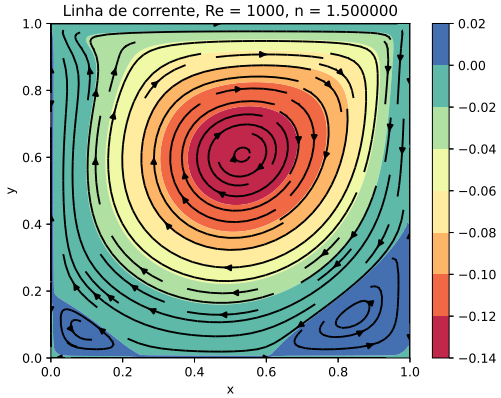

Figura 53: Linhas de corrente, $Re = 1000$, $n = 1.5$.

O comparativo entre as linhas de correntes apresentadas nesta sessão e as linhas de corrente apresentadas na Figura 29 nos permite observar uma tendência relativa às recirculações nos cantos inferiores. Com o aumento do valor índice de comportamento (n), essas recirculações tendem a aumentar em tamanho. Nos casos em que o fluido se torna mais viscoso com o aumento da taxa de cisalhamento ($n > 1$), ocorre a formação zonas de alta pressão nas arestas inferiores, onde as recirculações secundárias aparecem com maior intensidade. Acredito que o comportamento se deva ao fato de os termos viscosos terem maior influência na velocidade, em relação aos termos de inércia, uma vez que a taxa de cisalhamento pode crescer de maneira exponencial.

Nos gráficos de tensão de cisalhamento, grande parte da região da cavidade apresenta valores muito próximos e não chegam a ser destacados. Desta forma, as Figuras 54 e 56 mostram os resultados obtidos. As Figuras 55 e 57 apresentam uma aproximação na imagem, a área compreendida entre 0.95 e 1 nos dois eixos.

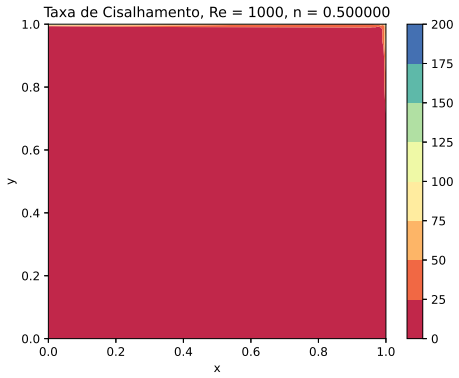

Figura 54: Taxa de Cisalhamento, $Re = 1000$, $n = 0.5$.

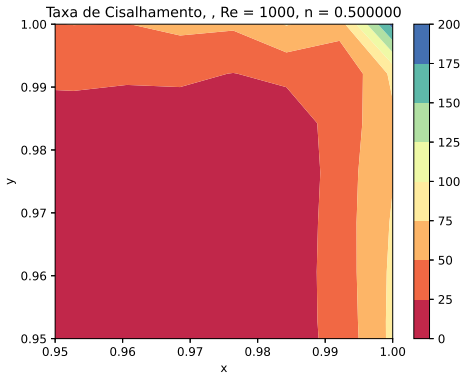

Figura 55: Taxa de Cisalhamento, $Re = 1000$, $n = 0.5$. (*zoom in*)

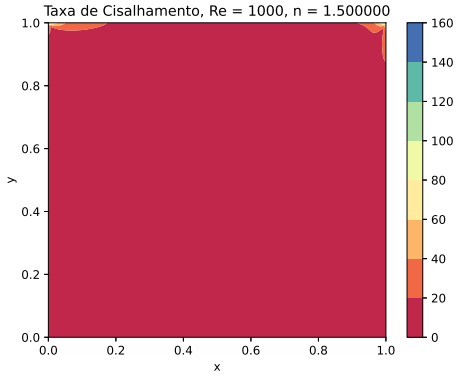

Figura 56: Taxa de Cisalhamento, $Re = 1000$, $n = 1.5$.

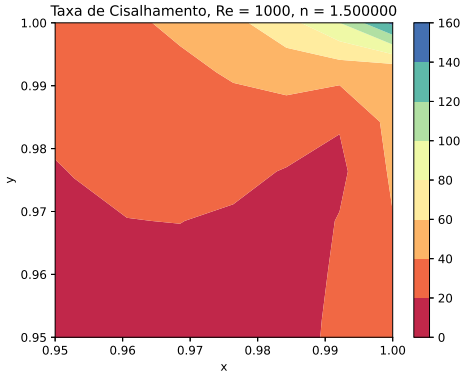

Figura 57: Taxa de Cisalhamento, $Re = 1000$, $n = 1.5$. (*zoom in*)

O canto superior direito, mostrado em destaque, concentra a maior parte da informação da imagem. As Figuras 58 e 59 mostram essa região para os valores de n iguais a 0.75 e 1.25, respectivamente.

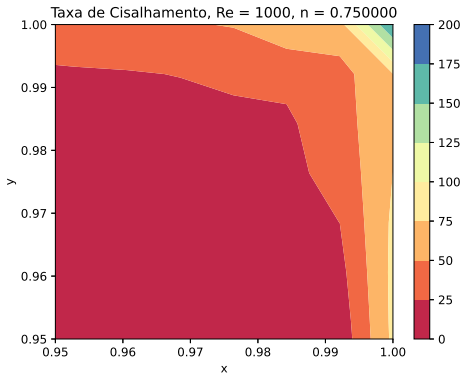

Figura 58: Taxa de Cisalhamento, $Re = 1000$, $n = 0.75 (zoom\ in)$

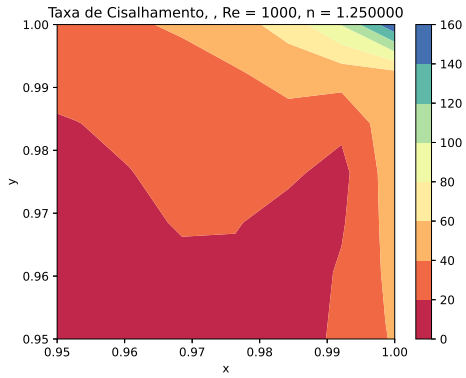

Figura 59: Taxa de Cisalhamento, $Re = 1000$, $n = 1.25$ $(zoom\ in)$.

Nas regiões com valores próximos a 50 (coloração alaranjada) o fluido sofre maior efeito da lei de Ostwald-de Weale. Isto é, para valores de n menores do que 1, o fluido se desloca com mais liberdade nessa região; para valores de n maiores do que 1, o fluido encontra maior resistência nessa região. Na seguinte sessão, serão introduzidos os resultados respectivos a distribuição de forças na parede superior, para diversos valores de n.

5.3 Distribuição de Forças na Parede Superior

Os resultados obtidos para os campos de velocidade se mostraram consideravelmente próximos aos propostos em literatura. Desta forma, a ferramenta pode ser utilizada para calcular a distribuição de forças na tampa com certa precisão. Este é um primeiro passo necessário no projeto do tubo que envolve o parafuso extrusor. As Figuras 60, 61 e 62 mostram os resultados encontrados para a distribuição de força sobre a tampa, para uma força adimensionalizada. A Equação 49 foi utilizada no cálculo da força.

$$F^* = (\nabla \mathbf{u})\, m^* \tag{49}$$

em que $F*$ é a força e $m*$ a massa, em suas formas adimensionais. Como o cálculo é para a tampa da cavidade, a direção da aceleração utilizada foi em y.

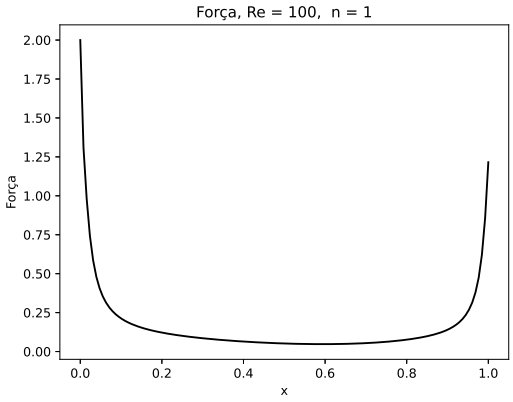

Figura 60: Distribuição de forças na tampa, $Re = 100$.

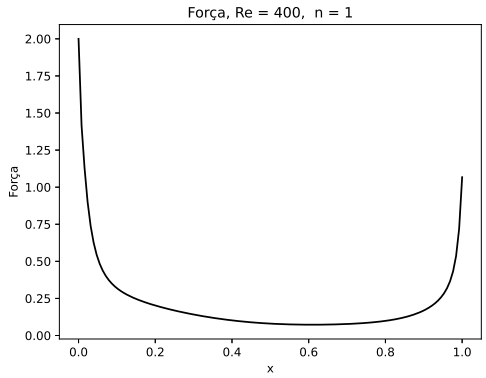

Figura 61: Distribuição de forças na tampa, $Re = 400$.

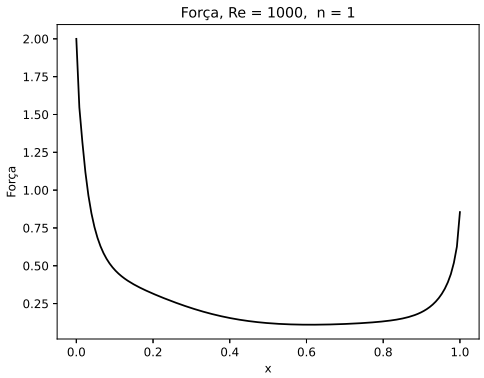

Figura 62: Distribuição de forças na tampa, $Re = 1000$.

A distribuição de força normal sobre a tampa, para um fluido não Newtoniano pode ser vista nas Figuras 63, 64 e 65. É possível observar uma mudança de comportamento na distribuição de força exercida na tampa da cavidade. Com o aumento do número de Reynolds, o incremento da força na tampa aumenta de forma mais drástica em um fluido não Newtoniano com $n = 0.5$, se comparado com os resultados obtidos para um fluido Newtoniano.

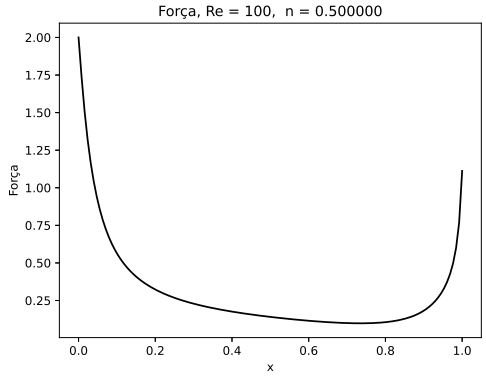

Figura 63: Distribuição de forças na tampa, $Re = 100$, $n = 0.5$.

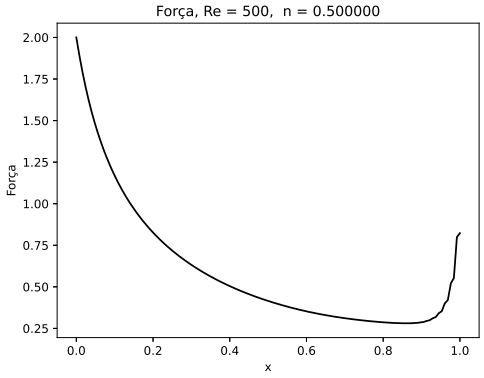

Figura 64: Distribuição de forças na tampa, $Re = 500$, $n = 0.5$.

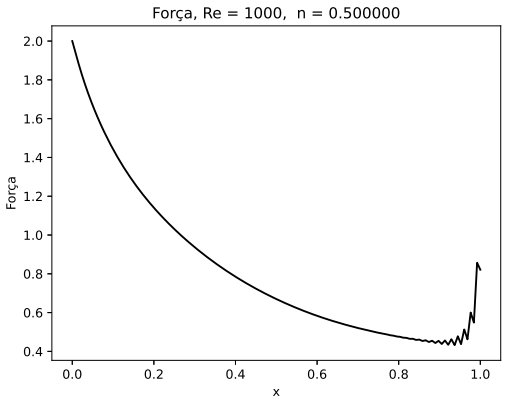

Figura 65: Distribuição de forças na tampa, $Re = 1000$, $n = 0.5$.

As curvas a seguir mostram a distribuição de força quando $n = 1.5$.

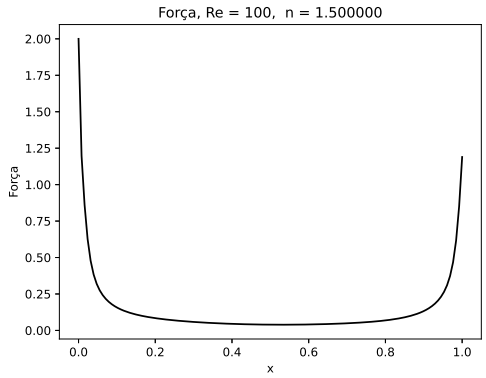

Figura 66: Distribuição de forças na tampa, $Re = 100$, $n = 1.5$.

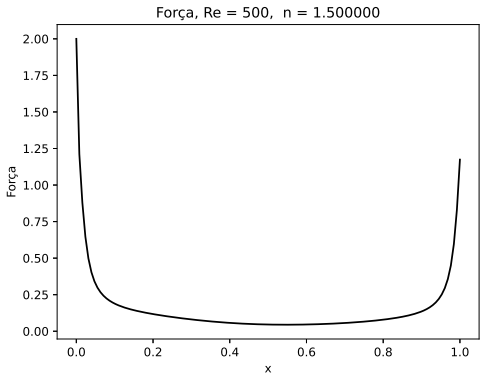

Figura 67: Distribuição de forças na tampa, $Re = 500$, $n = 1.5$.

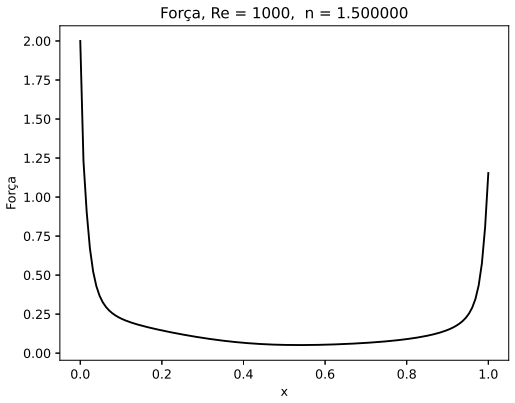

Figura 68: Distribuição de forças na tampa, $Re = 1000$, $n = 1.5$.

É possível observar pela distribuição de forças que, para um valor de $n < 1$, temos a situação que mais exerce carga sobre o contorno da cavidade. O gradiente das velocidades próximas à tampa pode ser utilizado no cálculo da força normal sobre a tampa e da força tangente necessária para se manter o escoamento ilustrado. Como a força é calculada através do gradiente de velocidades, é possível assumir que, para um fluido pseudoplástico, a força (normal) exercida no tubo que envolve o parafuso tende a ser maior do que para características Newtonianas ou dilatantes. As características de um fluido pseudoplástico acarretam em esforços mais intensos pois permitem maior aceleração - uma vez que a viscosidade tende a cair - e o fluido se move mais livremente. A Figura 69 mostra um comparativo entre as forças resultantes obtidas para diferentes Reynolds ($100, 500$ e 1000) e n ($0.5, 1.0$ e 1.5).

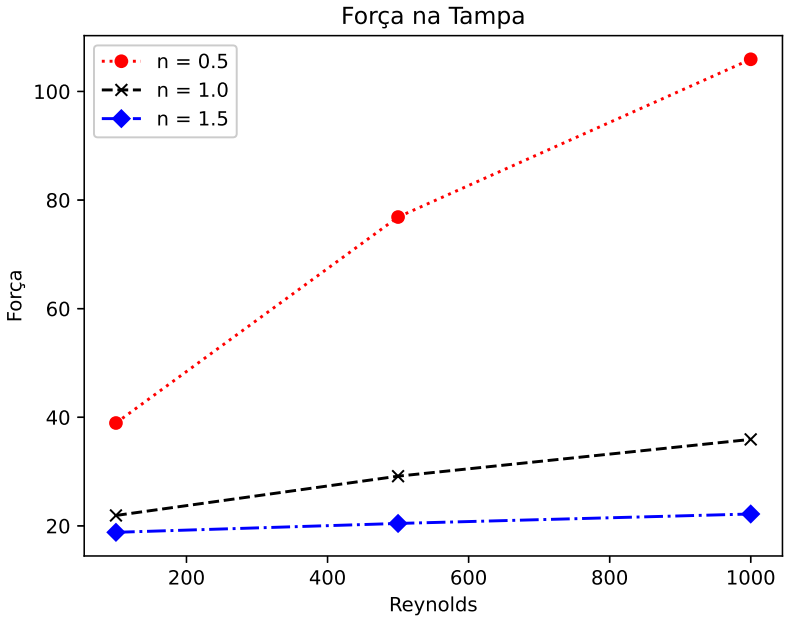

Figura 69: Resultados - Forças *versus* Reynolds.

A relação entre o índice de comportamento n e a força resultante pode ser observado de forma mais detalhada na figura 70. É possível observar que o decrescimento da força resultante - com o aumento do índice n - segue um padrão exponencial.

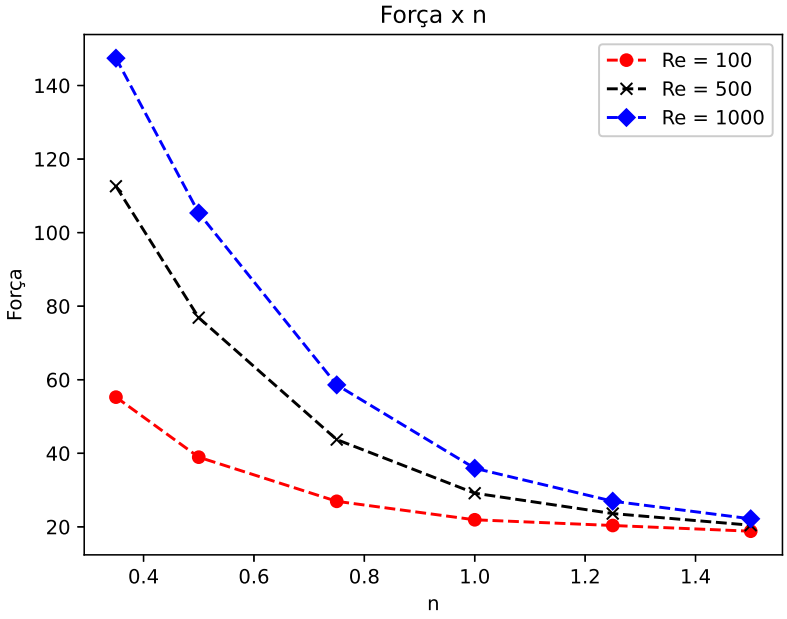

Figura 70: Resultados - Forças *versus* n.

Os valores de força resultante - encontrados para cada valor de n - foram utilizados para aproximar o comportamento do fluido com uma relação exponencial. Essa curva é apresentada na Figura 71.

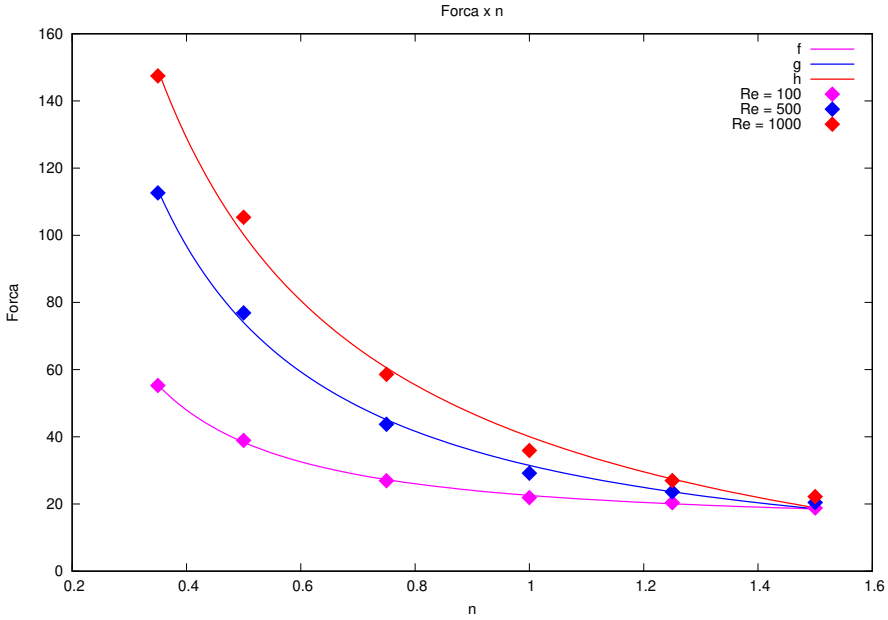

Figura 71: Ajuste da curva de força resultante *versus* n.

em que $h(n)$, $f(n)$ e $g(n)$ são:

$$f(n) = 13.62 - 8.94n^{(-1.47)} \tag{50}$$

$$g(n) = -2.82 + 34.31n^{(-1.16)} \tag{51}$$

$$h(n) = -28.90 + 68.95n^{(-0.9)} \tag{52}$$

respectivamente. Os pontos de $f(n)$, $g(n)$ e $h(n)$ são respectivos aos valores de Reynolds iguais a 100, 500 e 1000, respectivamente.

Para calcular um Reynolds característico para um problema e obter o valor das forças em estado dimensional, deve-se conhecer a geometria do parafuso. Essa geometria pode variar de acordo com o polímero a ser extrudado e a necessidade de vazão [1].

CAPITULO 6

CONCLUSÃO

O código feito em *Python* 3 *Anaconda*, e se mostrou capaz de prever as velocidades, pressão e taxa de cisalhamento, e assim, prever as forças exercidas nas paredes da cavidade. Esta é uma primeira etapa desse trabalho, com foco na validação da ferramenta e no estudo do comportamento do fluido para diferentes números de Reynolds e n. Avaliando os resultados obtidos foi possível entender o comportamento das linhas de corrente, do campo de pressão e taxa de viscosidade, para diferentes entradas no código. Foi necessário o recorte para ampliação, em algumas imagens, para observar pontos críticos, como no caso da taxa de cisalhamento. Algumas questões foram levantadas em relação à mudanças do centro da recirculação principal, assim como o tamanho das recirculações secundárias e o índice de comportamento do fluido.

Com as simulações , foi possível observar que o resultado mais crítico para a máquina extrusora acontece quando o fluido tem características pseudoplásticas, isto é, decresce em viscosidade com o aumento da velocidade. Para uma avaliação estrutural detalhada, a força das paredes deve ser obtida de maneira análoga à forma utilizada para a tampa. Os dados obtidos para força resultante na tampa da cavidade foram utilizados para determinar uma relação que descreva o comportamento do fluido de maneira aproximada. Com as relações obtidas para determinados números de Reynolds, para se determinar a força verdadeira deve-se determinar a velocidade e dimensões características das paredes da cavidade.

Uma próxima etapa deste trabalho deve conter o implemento da temperatura na malha do código e na lei de potência, assim como a equação da velocidade em $\hat{e}_y$ deve conter o termo de convecção natural. A influência da temperatura é de suma importância no processo de extrusão através de uma máquina extrusora de parafuso único, uma vez que a maior parte da energia utilizada na liquefação dos polímeros vem por condução térmica, e não efeitos viscosos. Deve conter também um mecanismo que mude a geometria da cavidade - de forma a representar uma cama sólida de polímero ainda não derretido - de acordo com a temperatura e tempo decorrido.

Referências Bibliográficas

[1] Harold F. Giles, John R. Wagner, and Eldridge M. Mount. *Extrusion: the definitive processing guide and handbook*. PDL handbook series. William Andrew, 2005. ISBN 978-0-8155-1473-2.

[2] J. A. Gaspar-Cunha, António e Covas. *Optimization in polymer processing*. Nova Science Publishers, 2011. ISBN 978-1-61122-818-2.

[3] F. Thibault, P. A. Tanguy, D. Blouin, and P. Hurez. Modeling of single-screw extruders with mixing pins: The case of PVC. *Journal of Vinyl and Additive Technology*, 1(3):127–136, 1995-09. ISSN 10835601. doi: 10.1002/vnl.730010304.n. URL `http://doi.wiley.com/10.1002/vnl.730010304.n`.

[4] J. R. A. Richard, S. M. e Pearson. *Computational Analysis of Polymer Processing*. Springer Nether-lands, 1983.

[5] C.T. Ghia, U. K.N.and Shin. High-Re solutions for incompressible flow using the navier-stokes equations and a multigrid method. *Journal of Computational Physics*, 1982.

[6] Qiuxiang Li, Ning Hong, Baochang Shi, and Zhenhua Chai. Simulation of power-law fluid flows in two-dimensional square cavity using multi-relaxation-time lattice boltzmann method. *Communications in Computational Physics*, 15(1):265–284, 2014. doi: 10.4208/cicp.160212.210513a.

[7] A. J. Chorin. Numerical solution of the navier-stokes equations. In *Mathematics of Computation*, 1968.

[8] Wolfgang Roland, Christian Marschik, Michael Krieger, Bernhard Löw-Baselli, and Jürgen Miethlinger. Symbolic regression models for predicting viscous dissipation of three-dimensional non-newtonian flows in single-screw extruders. *Journal of Non-Newtonian Fluid Mechanics*, 268:12–29, 2019. ISSN 0377-0257. doi: https://doi.org/10.1016/j.jnnfm.2019.04.006. URL `https://www.sciencedirect.com/science/article/pii/S0377025719300886`.

[9] I.R Edmondson e R.T Fenner. Melting of thermoplastics in single screw extruders. *Polymer*, 16(1):49 – 56, 1975. ISSN 0032-3861. doi: https://doi.org/10.1016/0032-3861(75)90095-6. URL `http://www.sciencedirect.com/science/article/pii/0032386175900956`.

[10] I. Tadmor, Z. e Klein. *Engineering Principles of Plasticating Extrusion*. Polymer science and engineering series. Krieger, 1978. ISBN 9780882756981. URL `https://books.google.com.br/books?id=z2XZAAAAMAAJ`.

[11] M. M. Molla and .L.S. Yao. Non-newtonian natural convection along a vertical heated wavy surface using a modified power-law viscosity model. *Journal of Heat Transfer*, 2009.

Anexos

CAPITULO A

ANEXO

A.1 Figuras do Programa

A.1.1 Fluido Newtoniano

A.1.1.1 Re 100

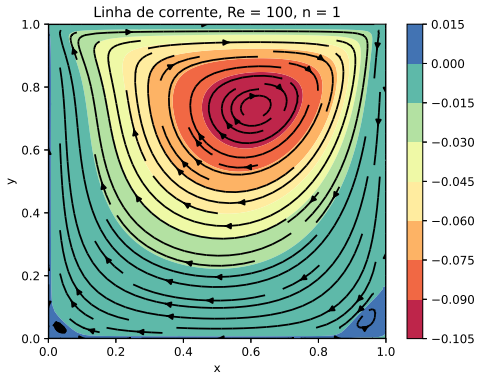

Figura 72: Linhas de corrente, $Re = 100$, $n = 1.0$.

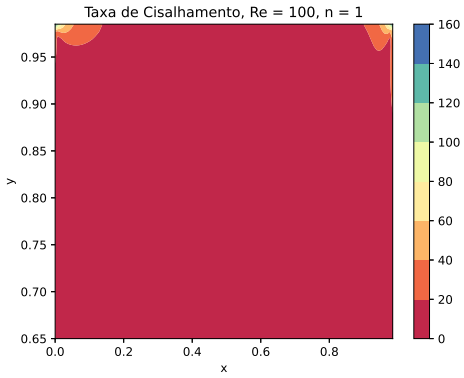

Figura 73: Taxa de Cisalhamento, $Re = 100$, $n = 1.0$.

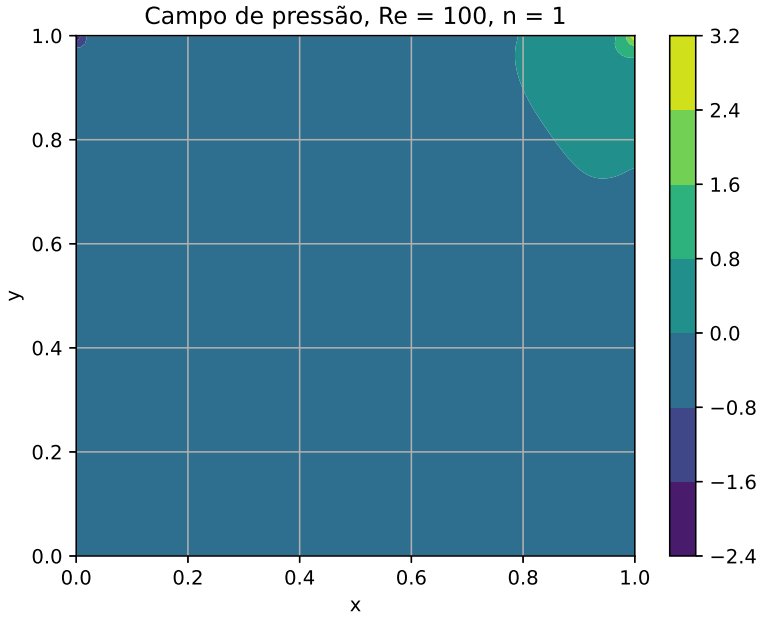

Figura 74: Campo de Pressão, $Re = 100$, $n = 1.0$.

A.1.1.2 Re 400

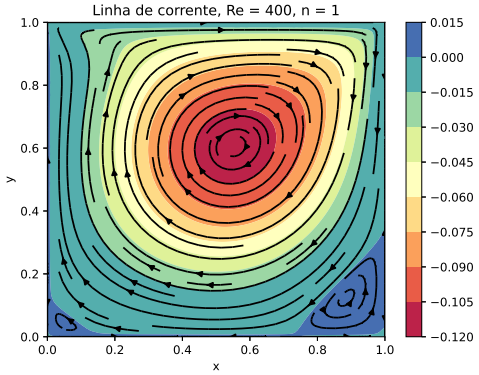

Figura 75: Linhas de corrente, $Re = 400$, $n = 1.0$.

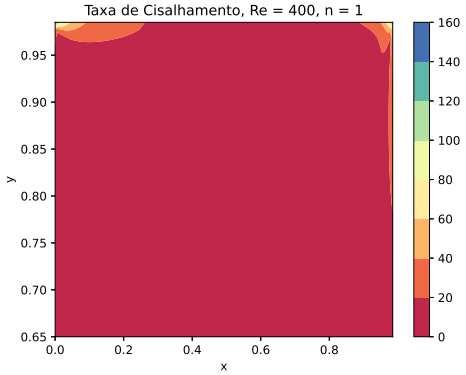

Figura 76: axa de Cisalhamento, $Re = 400$, $n = 1.0$.

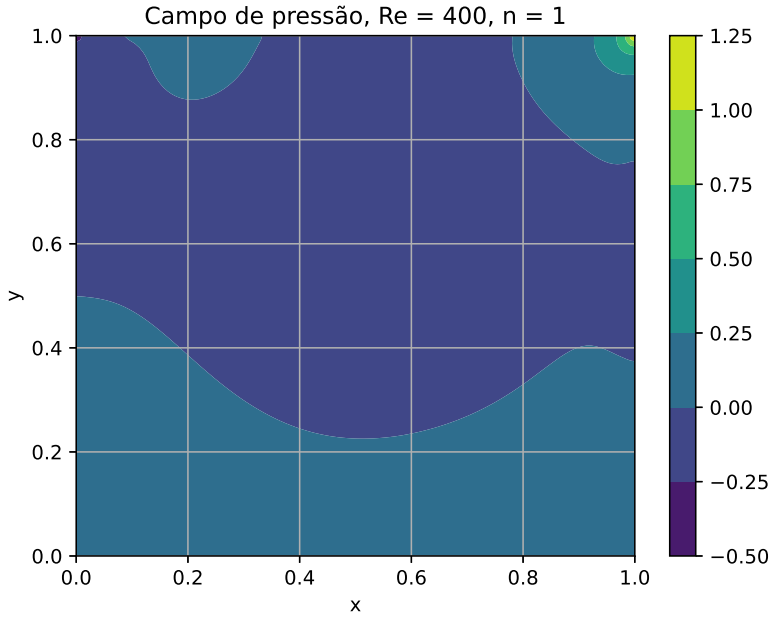

Figura 77: Campo de Pressão, $Re = 400$, $n = 1.0$.

A.1.1.3 Re 1000

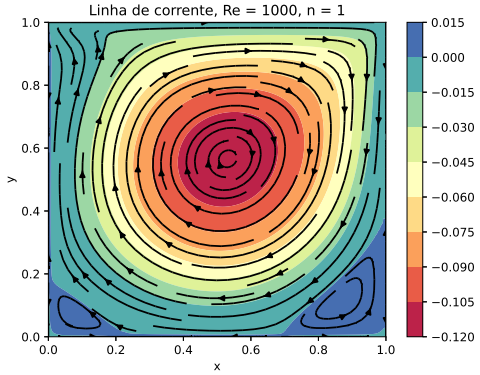

Figura 78: Linhas de corrente, $Re = 1000$, $n = 1.0$.

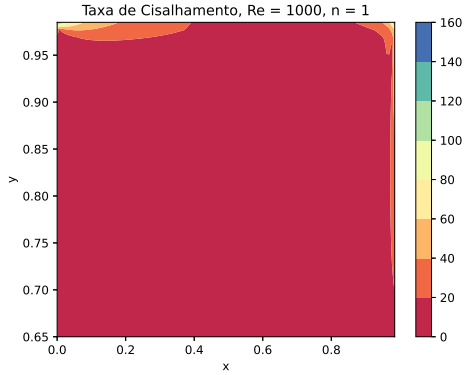

Figura 79: Taxa de Cisalhamento, $Re = 1000$, $n = 1.0$.

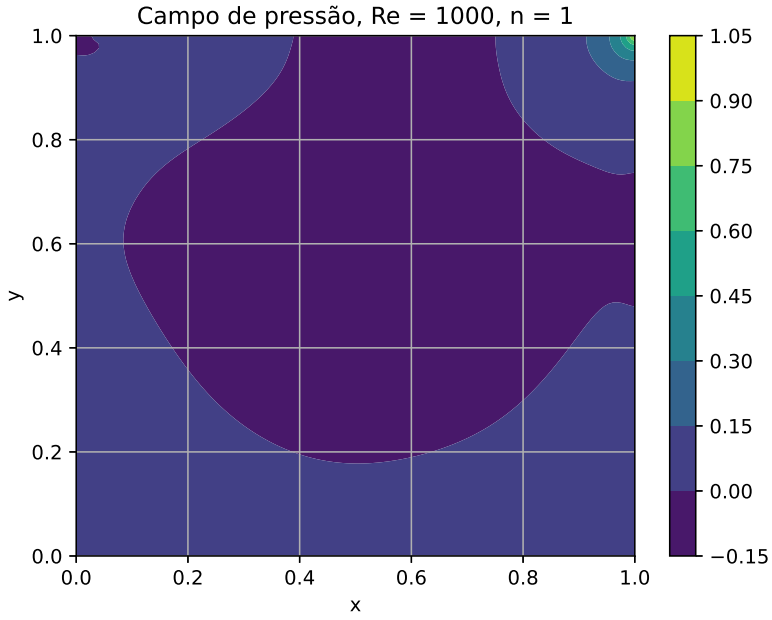

Figura 80: Campo Pressão, $Re = 1000$, $n = 1.0$.

A.1.2 Fluido não-Newtoniano

A.1.2.1 Re 100, n = 0.5

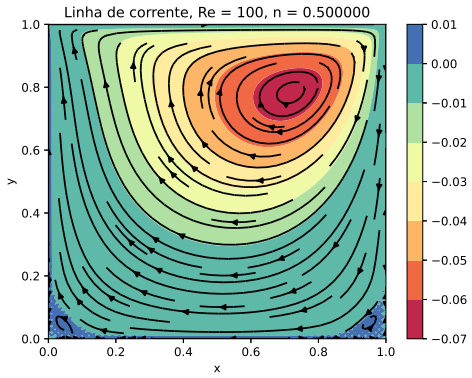

Figura 81: Linhas de corrente, $Re = 100$, $n = 0.5$.

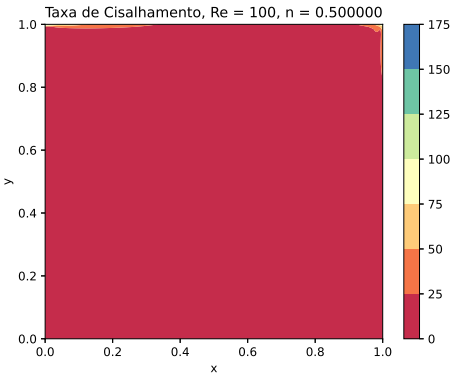

Figura 82: Taxa de Cisalhamento, $Re = 100$, $n = 0.5$.

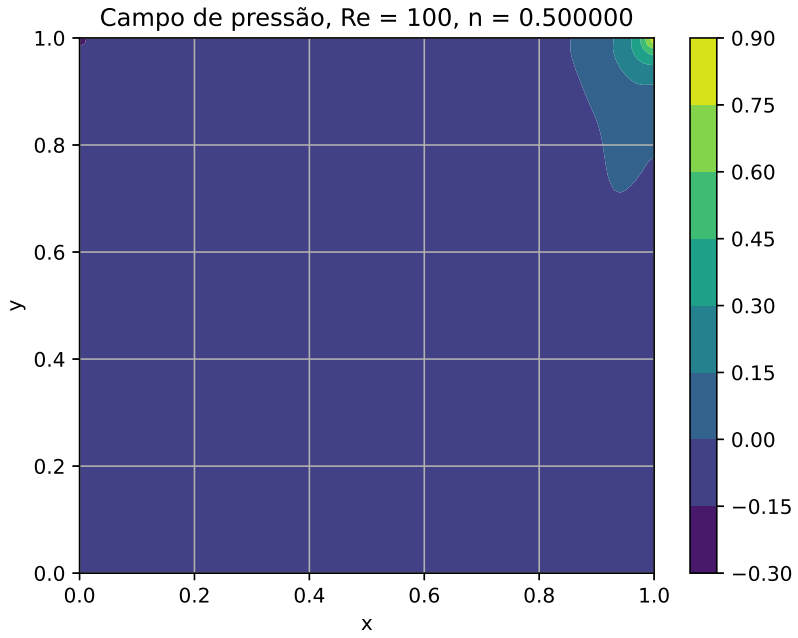

Figura 83: Campo de Pressão, $Re = 100$, $n = 0.5$.

A.1.2.2 Re 500, n = 0.5

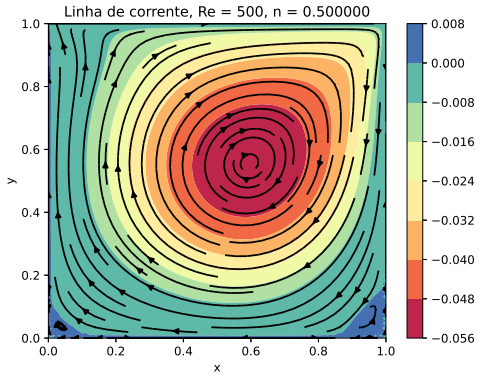

Figura 84: Linhas de corrente, $Re = 500$, $n = 0.5$.

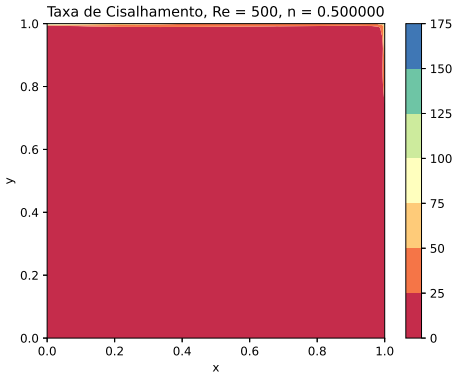

Figura 85: Taxa de Cisalhamento, $Re = 500$, $n = 0.5$.

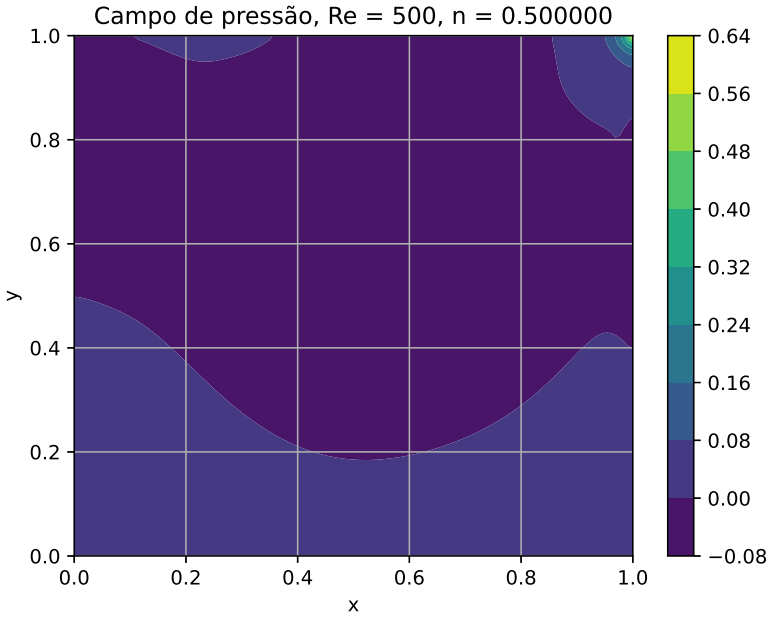

Figura 86: Campo de Pressão, $Re = 500$, $n = 0.5$.

A.1.2.3 Re 1000, n = 0.5

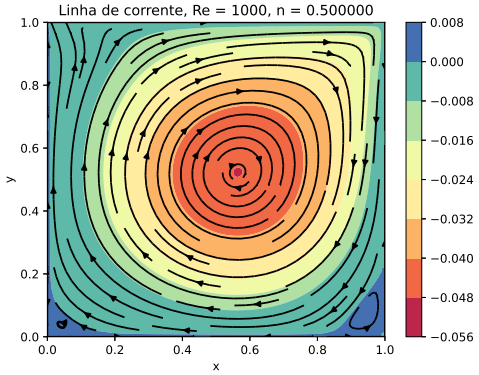

Figura 87: Linhas de corrente, $Re = 1000$, $n = 0.5$.

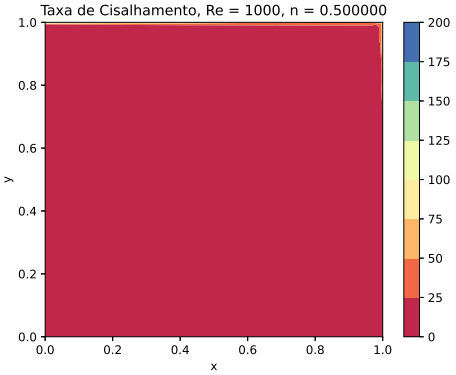

Figura 88: Taxa de Cisalhamento, $Re = 1000$, $n = 0.5$.

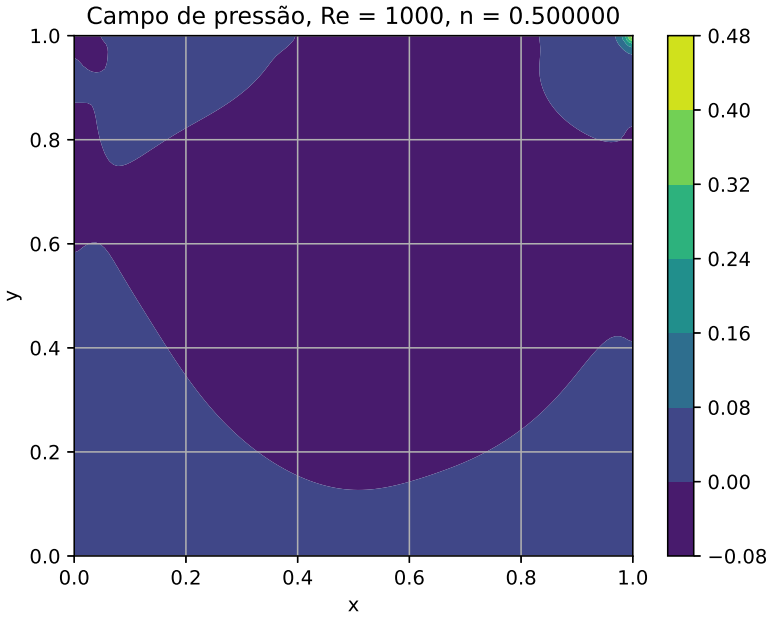

Figura 89: Campo de Pressão, $Re = 1000$, $n = 0.5$.

A.1.2.4 Re 100, n = 1.5

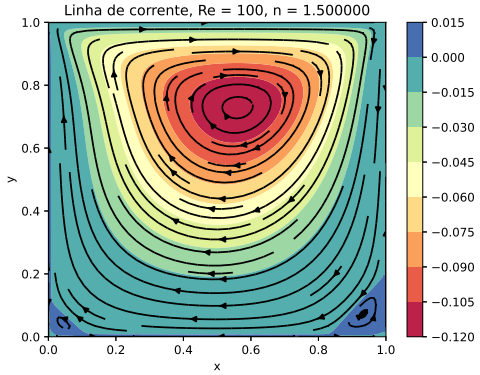

Figura 90: Linhas de corrente, $Re = 100$, $n = 1.5$.

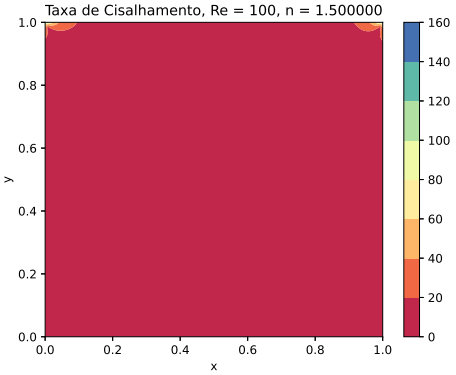

Figura 91: Taxa de Cisalhamento, $Re = 100$, $n = 1.5$.

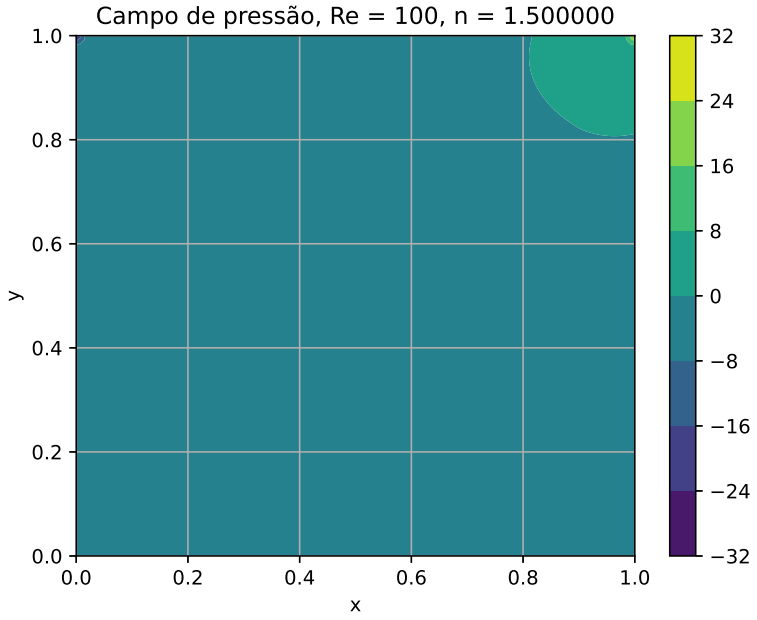

Figura 92: Campo de Pressão, $Re = 100$, $n = 1.5$.

A.1.2.5 Re 500, n = 1.5

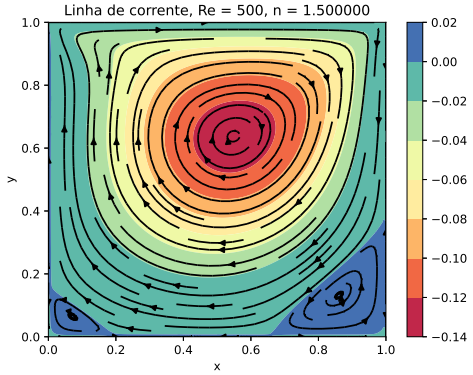

Figura 93: Linhas de corrente, $Re = 500$, $n = 1.5$.

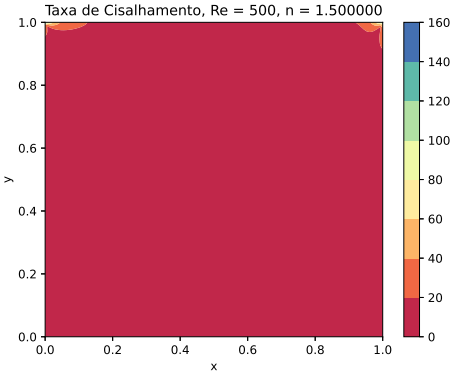

Figura 94: Taxa de Cisalhamento, $Re = 500$, $n = 1.5$.

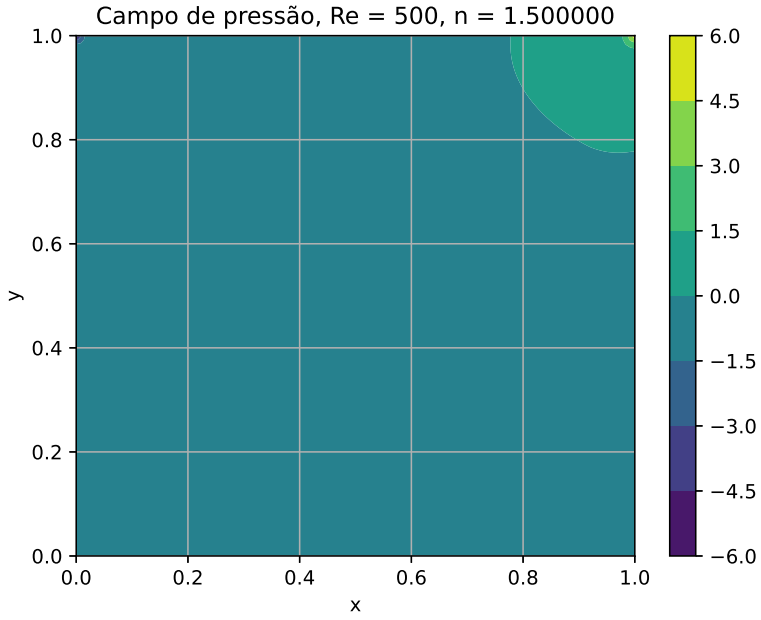

Figura 95: Campo de Pressão, $Re = 500$, $n = 1.5$.

A.1.2.6 Re 1000, n = 1.5

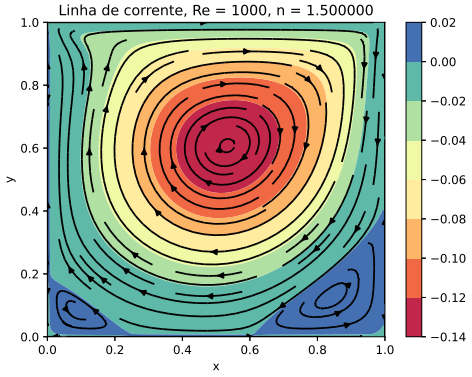

Figura 96: Linhas de corrente, $Re = 1000$, $n = 1.5$.

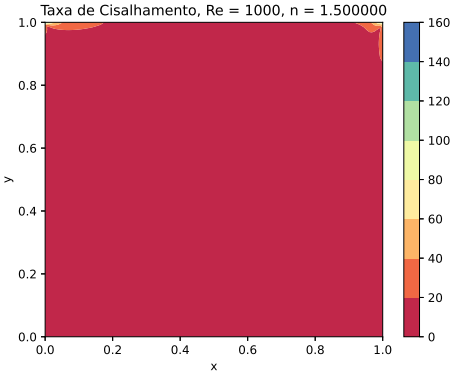

Figura 97: Taxa de Cisalhamento, $Re = 1000$, $n = 1.5$.

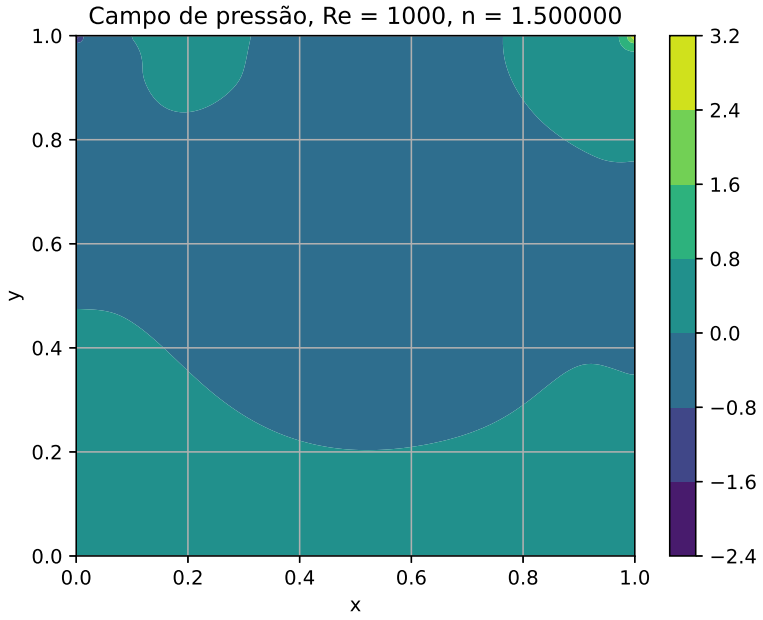

Figura 98: Campo de Pressão, $Re = 1000$, $n = 1.5$.

A.2 Código Python

A.2.1 Programa Principal

```
#LID DRIVEN CAVITY/ POWER LAW MODEL/ MATHEUS FERRARI
from PG2_FUNCTIONS import *
import numpy as np
import pandas
import timeit



# Condições Iniciais.
#####################################################
Re = 100 #Reynolds Number
n =  1.0 # Power Law Index


nx = 128 # malha = nx+1 (129 para igualar a malha do Ghia)
ny = nx

lx = 1.0 # x length
ly = 1.0# y length

x = np.linspace(0, lx, nx+1)  # Vetor x para plot
y = np.linspace(0, ly, ny+1)  # Vetor y para plot
#t_p = np.linspace(0,1,nx+2)

dx = lx/nx # Incremento em x
```

```
26 dy = ly/ny # Incremento em y
27
28 t_0 = 0.0 # Tempo inicial
29 t_f = 30.0 # Tempo final
30 t = 0.0
31
32 tol = 10.0**(-8) # Tolerância (Valor máximo do div (campo de velocidade))
33
34
35 #Definindo as matrizes de u, v, psi e p.
36
37 sx = (nx+1, ny+2) # Tamanho do campo u
38 sy = (nx+2, ny+1) # Tamanho do campo v
39 sp = (nx+2, ny+2) # Tamanho do campo p
40 spsi = (nx+1, ny+1)# Tamanho do campo 'psi'
41 sg = (nx,ny) # Tamanho de gamma_d
42
43
44
45 # Declarando os vetores de velocidade, pressão, linha de corrente, taxa de
   ↪  cisalhamento e divergente
46 u_star = np.zeros(sx, float)
47 v_star = np.zeros(sy, float)
48 gamma_d = np.ones(sg,float)
49 u_comp = np.zeros(nx+1, float)
50 v_comp = np.zeros(ny+1, float)
51 u = np.zeros(sx, float)
52 v = np.zeros(sy, float)
53 p = np.zeros(sp, float)
54 psi = np.zeros(spsi, float)
55 Ui = np.ones(nx+1, float)
56
57 div = np.zeros((nx,ny),float)
58 ####################
59
60
61
62
63 ###    CRITÉRIOS DE ESTABILIDADE
64 if dx >= 0.25/(np.sqrt(Re)):
65     print('INSTÁVEL')
66     exit()
67
68
69 dt = 0.1*Re*dx*dy # Se o programa parar, diminuir dt
70
71
72
73 ########################
74 # C.c.    de u, v e p
75
```

```
for i in range (nx+1):
    u[i,-1] = -u[i,0]
    u[i,ny] = 2.0*Ui[i] - u[i, ny-1]
for j in range (ny):
    u[0,j] = 0
    u[nx,j] = 0
u_star = np.copy(u)
for i in range (nx):
    v[i,0] = 0
    v[i,ny] = 0

for j in range (ny+1):
    v[-1,j] = -v[0,j]
    v[nx,j] = -v[nx-1,j]
v_star = np.copy(v)



for i in range (nx):
    p[i,-1] = p[i,0]
    p[i,ny] = p[i,ny-1]
for j in range (ny):
    p[-1,j] = p[0,j]
    p[nx,j] = p[nx-1,j]

p[-1,-1] = p[0,0]
p[-1,ny] = p[0,ny-1]
p[nx,-1] = p[nx-1,0]
p[nx,ny] = p[nx-1,ny-1]


# Time evolution. This is the main loop.
#####################################################

k = 0

while t < t_f - 0.001*dt:

    calculate_gamma_d(u,v, dx, dy, nx, ny, gamma_d)


    calculate_u_star(u_star, u, v, Re, nx, ny, dx, dy, dt, gamma_d, n)
    if np.any(u_star)>1:
        quit()
        print('explodiu')
    update_bc_u_star(u_star, Ui, nx, ny)

    calculate_v_star(v_star, u, v, Re, nx, ny, dx, dy, dt, n, gamma_d)

    update_bc_v_star(v_star, nx, ny)

```

```
127
128     calculate_pressure(p, u_star, v_star, nx, ny, dx, dy, dt, tol)
129
130     calculate_u_new(u_star, nx, ny, dt, p, u, v, dx)
131     calculate_v_new(v_star, nx, ny, dt, p, u, v, dy)
132
133     if (k % 100 == 0):
134       check_for_incompressibility(u, v, n, dx, dy, div)
135
136     u = np.copy(u)
137     v = np.copy(v)
138
139     t = t + dt
140     k = k + 1
141     print(t, np.amax(np.abs(div)))
142
143     force = np.zeros(nx, float)
144     for i in range(nx):
145         force[i] = force[i] + (u[i,ny]-u[i,ny-1])
146
147
148 calculate_stream_function(psi, nx, ny, v, u, dx, dy, tol)
149
150
151 u_plot = np.zeros((nx+1, ny+1), float)
152 v_plot = np.zeros((nx+1, ny+1), float)
153 p_plot = np.zeros((nx+1, ny+1), float)
154 #t_plot = np.zeros((nx+1, ny+1), float)
155
156 for i in range (0,nx+1):
157     for j in range (0, ny+1):
158
159         u_plot[i,j] = (0.5)*(u[i,j] + u[i,j-1])
160         v_plot[i,j] = (0.5)*(v[i,j] + v[i-1,j])
161         if i == 0 and j == 0:
162             p_plot[i,j] = (1/3)*(p[i,j] + p[i, j-1] + p[i-1,j])
163         elif i == 0 and j == ny:
164             p_plot[i,j] = (1/3)*(p[i,j] + p[i, j-1] + p[i-1,j-1])
165         elif i == nx and j == 0:
166             p_plot[i,j] = (1/3)*(p[i,j] + p[i-1,j] + p[i-1,j-1])
167         elif i == nx and j == ny:
168             p_plot[i,j] = (1/3)*(p[i, j-1] + p[i-1,j] + p[i-1,j-1])
169         else:
170             p_plot[i,j] = (1/4)*(p[i,j] + p[i, j-1] + p[i-1,j] + p[i-1,j-1])
171
172
173
174
175
176 # Criando os vetores horizontal e vertical de v e u
177 u_comp = np.zeros(nx, float)
```

```
178 v_comp = np.zeros(nx, float)
179 comparativo_lit(Re, n, u_comp, v_comp, nx, ny, u_plot, v_plot)
180
181
182
183
184
185
186
187
188 #Definindo os dados oferecidos por Ghia.
189
190 if Re == 100:
191
192     u_ghia =
        ↪  [0.0,-0.03717,-0.04775,-0.04775,-0.06434,-0.10150,-0.15662,-0.21090,-0.20581,
193     v_ghia = [0.0, 0.09233, 0.10091, 0.10890, 0.12317, 0.16077, 0.17507,
        ↪  0.17527, 0.05454, -0.24533, -0.22445, -0.16914, -0.10313, -0.08864,
        ↪  -0.07391, -0.05906, 0.0]
194
195
196 if Re == 400:
197
198     u_ghia = [0.000, -0.08186, -0.09266, -0.10338,
        ↪  -0.14612,-0.24299,-0.32726,-0.17119,-0.11477,0.02135,0.16256,0.29093,0.55892,
199     v_ghia = [0.000, 0.18360, 0.19713, 0.20920, 0.22965, 0.28124, 0.30203,
        ↪  0.30174, 0.05186, -0.38598, -0.44993, -0.23827, -0.22847, -0.19254,
        ↪  -0.15663, -0.12146, 0.000]
200
201
202 if Re == 1000:
203
204     u_ghia =
        ↪  [0.0,-0.18109,-0.20196,-0.22220,-0.29730,-0.38289,-0.27805,-0.10648,-0.06080,
205     v_ghia = [0.000, 0.27485, 0.29012, 0.30353, 0.32627, 0.37095, 0.33075,
        ↪  0.32235, 0.02526, -0.31966, -0.42665, -0.51550, -0.39188, -0.33714,
        ↪  -0.27669, -0.21388, 0.000]
206
207
208 y_ghia = (0.0, 0.0547, 0.0625, 0.0703, 0.1016, 0.1719, 0.2813, 0.4531,
    ↪  0.5000, 0.6172, 0.7344, 0.8516, 0.9531, 0.9609, 0.9688, 0.9766, 1.000)
209 x_ghia = (0.0, 0.0625, 0.0703, 0.0781, 0.0938, 0.1563, 0.2266, 0.2344,
    ↪  0.5000, 0.8047, 0.8594, 0.9063, 0.9453, 0.9531, 0.9609, 0.9688, 1.000)
210
211
212
213 #Definindo os pontos do não newtoniano (Li)
214 ###Re 100  n = 0.5
215
216 if Re == 100 and n == 0.5:
217 #u:
```

```
218     x_u = (0.00000000000, -0.0134408602150538, -0.0268817204301075,
        ↪  -0.0403225806451613, -0.0564516129032258, -0.0672043010752688,
        ↪  -0.0833333333333333, -0.0994623655913978, -0.110215053763441,
        ↪  -0.115591397849462, -0.118279569892473, -0.118279569892473,
        ↪  -0.118279569892473, -0.118279569892473, -0.112903225806452,
        ↪  -0.104838709677419, -0.0940860215053763, -0.0779569892473118,
        ↪  -0.0591397849462366, -0.0403225806451613, -0.0188172043010753,
        ↪  0.010752688172043, 0.0376344086021505, 0.0833333333333333,
        ↪  0.120967741935484, 0.153225806451613, 0.193548387096774,
        ↪  0.252688172043011, 0.301075268817204, 0.362903225806452,
        ↪  0.422043010752688, 0.475806451612903, 0.540322580645161,
        ↪  0.610215053763441, 0.771505376344086, 1.00000000000000)
219     y_u =  (0.00, 0.0743589743589744, 0.133333333333333, 0.184615384615385,
        ↪  0.243589743589744, 0.282051282051282, 0.330769230769231,
        ↪  0.376923076923077, 0.41025641025641, 0.443589743589744,
        ↪  0.458974358974359, 0.47948717948718, 0.497435897435897,
        ↪  0.520512820512821, 0.548717948717949, 0.584615384615385,
        ↪  0.607692307692308, 0.641025641025641, 0.67948717948718,
        ↪  0.712820512820513, 0.746153846153846, 0.78974358974359,
        ↪  0.82051282051282, 0.861538461538462, 0.884615384615385,
        ↪  0.902564102564103, 0.917948717948718, 0.933333333333333,
        ↪  0.946153846153846, 0.956410256410256, 0.964102564102564,
        ↪  0.969230769230769, 0.974358974358974, 0.979487179487179,
        ↪  0.98974358974359, 1.00000000)
220 #v:
221     x_v = (0.00096585505275244, 0.0723455001502989, 0.13598352133012,
        ↪  0.178397370508897, 0.226586638676581, 0.294033873916493,
        ↪  0.3633881801024, 0.425025501530092, 0.502047514155197,
        ↪  0.544387445855447, 0.590580942102903, 0.642525218179758,
        ↪  0.70022027408601, 0.738677074247643, 0.78100715028409,
        ↪  0.809874389564823, 0.836844413563365, 0.861902438784008,
        ↪  0.883126610785053, 0.906297276387308, 0.935272927969881,
        ↪  1.00096092722085)
222     y_v = (0.00102301790281328, 0.037851662404092, 0.0603580562659847,
        ↪  0.0705882352941177, 0.078772378516624, 0.0828644501278772,
        ↪  0.078772378516624, 0.0705882352941177, 0.0501278772378517,
        ↪  0.0296675191815857, 0.00920716112531972, -0.0235294117647059,
        ↪  -0.0685421994884911, -0.101278772378517, -0.125831202046036,
        ↪  -0.140153452685422, -0.142199488491049, -0.138107416879795,
        ↪  -0.125831202046036, -0.10537084398977, -0.0746803069053709,
        ↪  -0.00102301790281334)
223
224
225
226 ###Re 100 n = 1.5
227
228 if Re == 100 and n == 1.5:
229 #u
230
231     x_u = (0.0000, -0.0456989247311828, -0.0833333333333333,
        ↪  -0.104838709677419, -0.131720430107527, -0.161290322580645,
        ↪  -0.188172043010753, -0.212365591397849, -0.228494623655914,
        ↪  -0.231182795698925, -0.233870967741935, -0.225806451612903,
        ↪  -0.212365591397849, -0.185483870967742, -0.150537634408602,
```

```
232     y_u = (0.000000, 0.0487179487179487, 0.0974358974358974,
        ↪  0.130769230769231, 0.179487179487179, 0.235897435897436,
        ↪  0.28974358974359, 0.356410256410256, 0.407692307692308,
        ↪  0.446153846153846, 0.492307692307692, 0.523076923076923,
        ↪  0.558974358974359, 0.597435897435897, 0.638461538461538,
        ↪  0.676923076923077, 0.746153846153846, 0.794871794871795,
        ↪  0.841025641025641, 0.902564102564103, 0.933333333333333,
        ↪  0.956410256410256, 1.00000)
233 #v
234     x_v = (-0.00096585505275244, 0.0203420900906228, 0.0513085857615225,
        ↪  0.0822455144410114, 0.107362673644477, 0.136323541731344,
        ↪  0.171040117479513, 0.203815127458372, 0.236565498277723,
        ↪  0.277003286863878, 0.307802236250117, 0.355907731275471,
        ↪  0.413597859349822, 0.446274312690646, 0.505841944719582,
        ↪  0.573082211019618, 0.630703349447344, 0.67873492699417,
        ↪  0.717176943660098, 0.744107545003425, 0.780686841210473,
        ↪  0.809588575314519, 0.836593094136373, 0.861685614180329,
        ↪  0.884856279782584, 0.913861498356568, 0.997082723514136)
235     y_v = (-0.00102301790281334, 0.0460358056265985, 0.103324808184143,
        ↪  0.148337595907928, 0.176982097186701, 0.20153452685422,
        ↪  0.215856777493606, 0.224040920716113, 0.221994884910486,
        ↪  0.211764705882353, 0.199488491048593, 0.172890025575448,
        ↪  0.125831202046036, 0.0930946291560102, 0.0255754475703325,
        ↪  -0.0562659846547314, -0.131969309462916, -0.18925831202046,
        ↪  -0.228132992327366, -0.246547314578005, -0.258823529411765,
        ↪  -0.258823529411765, -0.246547314578005, -0.228132992327366,
        ↪  -0.2076726342711, -0.164705882352941, -0.0112531969309463)
236
237
238
239
240 ##################   PLOT DA PRESSÃO
241 plt.figure()
242 plt.contourf(x,y,np.transpose(p_plot))
243 plt.colorbar()
244 plt.xlabel('x')
245 plt.ylabel('y')
246 plt.axis((0, lx, 0, ly))
247 plt.title('Campo de pressão, Re = %i, n = %f' %(Re, n))
248 plt.grid()
249 plt.show()
250
251 ###################PLOT DA LINHA DE CORRENTE
252 plt.figure()
253 plt.contourf(x,y,np.transpose(psi), cmap = 'Spectral')
254 plt.colorbar()
255 plt.streamplot(x,y,np.transpose(u_plot), np.transpose(v_plot), color = 'k')
256 plt.xlabel('x')
257 plt.ylabel('y')
258 plt.axis((0, lx, 0, ly))
259 plt.title('Linha de corrente, Re = %i, n = %f' %(Re, n))
```

```
260 plt.show()
261
262 ################PLOT DA VELOCIDADES (QUIVER)
263 plt.figure()
264 plt.quiver(x,y,np.transpose(u_plot), np.transpose(v_plot))
265 plt.xlabel('x')
266 plt.ylabel('y')
267 plt.axis((0, lx, 0, ly))
268 # plt.colorbar()
269 plt.title('Campo de velocidades, Re = %i, n = %f' %(Re, n))
270 plt.show()
271
272
273
274 # ======================== ERRO COMPARAÇÃO GHIA
    ↪  ================================================
275 # #PLOT DO ERRO EM U
276 # plt.figure()
277 # plt.plot(y_ghia,(np.abs(u_ghia-u_vec)),marker='o',color = 'k',
    ↪  linestyle='dashed')
278 # plt.xlabel('y')
279 # plt.ylabel('Erro')
280 # plt.title('Erro, em u, Re = %i, n = %i' %(Re, n))
281 # plt.show()
282 #
283 # #PLOT DO ERRO EM V
284 # plt.figure()
285 # plt.plot(x_ghia,(np.abs(v_ghia-v_vec)),marker='o',color = 'k',
    ↪  linestyle='dashed')
286 # plt.xlabel('x')
287 # plt.ylabel('Erro')
288 # plt.title('Erro, em v, Re = %i, n = %i' %(Re, n))
289 # plt.show()
290 #
    ↪  ================================================================================
291
292 # =========== COMPARATIVO NEWTONIANO
    ↪  ================================================================
293 #PLOT DA LINHA MEDIA EM U (VERTICAL)
294 plt.figure()
295 plt.plot(y,u_comp,marker='o', linestyle='dashed', color='r')
296 plt.plot(y_ghia,u_ghia,marker='o', linestyle='dashed', color =
    ↪  'k')#,markersize=12)
297 plt.legend(["Dados obtidos", "Ghia"])
298 plt.xlabel('y')
299 plt.ylabel('Velocidade u')
300 plt.title('Velocidade u, Re = %i,  n = %i' %(Re, n))
301 plt.show()
302
303 #PLOT DA LINHA MÉDIA EM V (HORIZONTAL)
304
```

```
plt.figure()
plt.plot(x,v_comp, marker='o', linestyle='dashed', color = 'r')
plt.plot(x_ghia,v_ghia,marker='o', linestyle='dashed', color = 'k')
plt.legend(["Dados obtidos", "Ghia"])
plt.xlabel('x')
plt.ylabel('Velocidade v')
plt.title('Velocidade v,  Re = %i,  n = %i' %(Re, n))
plt.show()




# ========================= COMPARATIVO N-NEWTONIANO

# ================================================================================
# plt.figure()
# plt.plot(y,u_comp,marker='o', linestyle='dashed', color='r')
# plt.plot(y_u, x_u, color = 'k')
# plt.legend(["Dados obtidos", "Li"])
# plt.xlabel('y')
# plt.ylabel('u')
# plt.title('Força, Re = %i,  n = %f' %(Re, n))
# plt.show()
#
#
# plt.figure()
# plt.plot(x,v_comp,marker='o', linestyle='dashed', color='r')
# plt.plot(x_v, y_v, color = 'k')
# plt.legend(["Dados obtidos", "Li"])
# plt.xlabel('x')
# plt.ylabel('v')
# plt.title('Força, Re = %i,  n = %f' %(Re, n))
# plt.show()
# ================================================================================

# ================================================================================




x = np.linspace(0, lx, nx)
y = np.linspace(0,ly,ny)

plt.figure()
plt.contourf(x,y,np.transpose(gamma_d), cmap = 'Spectral')
plt.colorbar()
```

```
plt.xlabel('x')
plt.ylabel('y')
plt.axis((0.0, 1.0, 0.0, 1.0))
plt.title('Taxa de Cisalhamento, Re = %i, n = %f' %(Re, n))
plt.show()



plt.figure()
plt.plot(x,force, color = 'k')
plt.xlabel('x')
plt.ylabel('Força')
plt.title('Força, Re = %i,  n = %f' %(Re, n))
plt.show()
```

A.2.2 Funções do Programa

```
import numpy as np

from mpl_toolkits import mplot3d

import matplotlib.pyplot as plt

from numba import jit, njit

#################################################


#################################################

@njit
def calculate_gamma_d(u,v, dx, dy, nx, ny, gamma_d):

    for i in range(0, nx):
        for j in range(0, ny):


            a1 = 2.0*(((u[i+1,j]-u[i,j])/dx)**2.0)
            a2 = 2.0*(((v[i,j+1]-v[i,j])/dy)**2.0)
            a3 = (u[i,j+1] + u[i+1,j+1] - u[i,j-1] - u[i+1,j-1])/(4.0*dy)
            a4 = (v[i+1,j] + v[i+1,j+1] - v[i-1,j] - v[i-1,j+1])/(4.0*dx)
            a5 = (a3+a4)**2.0


            gamma_d[i,j] = np.sqrt(a1 + a2 + a5)


    return gamma_d

```

```
33
34  ################################################
35
36
37  ################################################
38  @njit
39  def calculate_u_star(u_star, u, v, Re, nx, ny, dx, dy, dt, gamma_d, n):
40
41
42
43
44      for i in range (1,nx):
45          for j in range (0,ny):
46
47
48              C1 = (1.0/4.0)*(v[i,j+1] + v[i-1,j+1] + v[i,j] + v[i-1,j])
49
50
51
52
53              gr = gamma_d[i,j] #gamma right
54              gl = gamma_d[i-1,j] #gamma left
55
56
57
58
59              if j == ny-1:
60
61                  gu = (3.0/4.0)*(gamma_d[i-1, j] + gamma_d[i,j]) -
                    ↪  (1.0/4.0)*(gamma_d[i-1,j-1]+ gamma_d[i,j-1]) #gamma up
62                  gd =
                    ↪  (gamma_d[i,j]+gamma_d[i-1,j]+gamma_d[i,j-1]+gamma_d[i-1,j-1])/4.0
                    ↪  # gamma down
63
64              elif j == 0:
65
66                  gu =
                    ↪  (gamma_d[i,j]+gamma_d[i-1,j]+gamma_d[i,j+1]+gamma_d[i-1,j+1])/4.0
67                  gd = (3.0/4.0)*(gamma_d[i-1, j] + gamma_d[i,j]) -
                    ↪  (1.0/4.0)*(gamma_d[i-1,j+1]+ gamma_d[i,j+1])
68
69              else:
70                  gd =
                    ↪  (gamma_d[i,j]+gamma_d[i-1,j]+gamma_d[i,j-1]+gamma_d[i-1,j-1])/4.0
71                  gu =
                    ↪  (gamma_d[i,j]+gamma_d[i-1,j]+gamma_d[i,j+1]+gamma_d[i-1,j+1])/4.0
72
73
74
75              if gu <= 10.0**(-2):
76                  gu_n = 1.0
```

```
77              elif gu >= 10.0**(5):
78                  gu_n= 10.0**(5)
79              else:
80                  gu_n = gu**(n-1)
81
82              if gd <= 10.0**(-2):
83                  gd_n = 1.0
84              elif gd >= 10.0**(5):
85                  gd_n= 10.0**(5)
86              else:
87                  gd_n = gd**(n-1)
88
89              if gl <= 10.0**(-2):
90                  gl_n = 1.0
91              elif gl >= 10.0**(5):
92                  gl_n= 10.0**(5)
93              else:
94                  gl_n = gl**(n-1)
95
96              if gr <= 10.0**(-2):
97                  gr_n = 1.0
98              elif gr >= 10.0**(5):
99                  gr_n= 10.0**(5)
100             else:
101                 gr_n = gr**(n-1)
102
103
104
105
106             aux_u1 = (u[i,j]*((u[i+1,j]-u[i-1,j])/(2*dx))  +  C1*((u[i,j+1] -
                ↪  u[i,j-1])/(2*dy)))*Re
107             aux_u2 = 2*(gr_n*(u[i+1,j] - u[i,j]) - gl_n*(u[i,j] -
                ↪  u[i-1,j]))/(dx**2)
108             aux_u3 = (gu_n*(u[i,j+1] - u[i,j]) - gd_n*(u[i,j] -
                ↪  u[i,j-1]))/(dy**2)
109             aux_u4 = (gu_n*(v[i,j+1]-v[i-1,j+1]) - gd_n*(v[i,j]-
                ↪  v[i-1,j]))/(dx*dy)
110
111
112 #
    ↪  ==========================================================================
113 #             aux_u1 = (u[i,j]*((u[i+1,j]-u[i-1,j])/(2*dx))  +
    ↪  C1*((u[i,j+1] - u[i,j-1])/(2*dy)))*Re   # Termo de inércia
114 #
115 #             aux_u2 = 2*((u[i+1,j] - u[i,j]) - (u[i,j] -
    ↪  u[i-1,j]))/(dx**2)
116 #             aux_u3 = ((u[i,j+1] - u[i,j]) - (u[i,j] - u[i,j-1]))/(dy**2)
117 #             aux_u4 = ((v[i,j+1]-v[i-1,j+1]) - (v[i,j]- v[i-1,j]))/(dx*dy)
118 #
    ↪  ==========================================================================
119
```

```
            u_star[i,j] = u[i,j] + (dt/Re)*(-aux_u1 + aux_u2 + aux_u3 +
            ↪  aux_u4)


    return u_star
#####################################################

#####################################################
@njit
def update_bc_u_star(u_star, Ui, nx, ny):

    for j in range (0,ny):
        u_star[0,j] = 0
        u_star[nx,j] = 0

    for i in range (0,nx+1):
        u_star[i,-1] = -u_star[i,0]
        u_star[i,ny] = 2*Ui[i] - u_star[i, ny-1]

    return u_star

#####################################################



#####################################################
@njit
def calculate_v_star(v_star, u, v, Re, nx, ny, dx, dy, dt, n, gamma_d):


    for i in range (0,nx):
        for j in range (1,ny):

            C2 = (0.25)*(u[i+1,j] + u[i,j] + u[i+1,j-1] + u[i,j-1])

            gu = gamma_d[i,j]
            gd = gamma_d[i,j-1]

            if i == nx-1:

                gl = (gamma_d[i,j] + gamma_d[i,j-1] + gamma_d[i-1,j]+
                ↪  gamma_d[i-1,j-1])*(1.0/4.0)
                gr = (gamma_d[i,j-1] + gamma_d[i,j])*(3.0/4.0) -
                ↪  (gamma_d[i-1,j-1] - gamma_d[i-1,j])*(1.0/4.0)

            elif i == 0 :

                gl = (gamma_d[i,j-1] + gamma_d[i,j])*(3.0/4.0) -
                ↪  (gamma_d[i+1,j-1] - gamma_d[i+1,j])*(1.0/4.0)
                gr = (gamma_d[i,j] + gamma_d[i,j-1] + gamma_d[i+1,j]+
                ↪  gamma_d[i+1,j-1])*(1.0/4.0)
```

```
166
167             else:
168
169                 gl = (gamma_d[i,j] + gamma_d[i,j-1] + gamma_d[i-1,j]+
                    ↪  gamma_d[i-1,j-1])*(1.0/4.0)
170                 gr = (gamma_d[i,j] + gamma_d[i,j-1] + gamma_d[i+1,j]+
                    ↪  gamma_d[i+1,j-1])*(1.0/4.0)
171
172
173
174
175             #estabilidade
176             if gu <= 10.0**(-2):
177                 gu_n = 1.0
178             elif gu >= 10.0**(5):
179                 gu_n= 10.0**(5)
180             else:
181                 gu_n = gu**(n-1)
182
183
184
185             if gd <= 10.0**(-2):
186                 gd_n = 1.0
187             elif gd >= 10.0**(5):
188                 gd_n= 10.0**(5)
189             else:
190                 gd_n = gd**(n-1)
191
192
193
194             if gl <= 10.0**(-2):
195                 gl_n = 1.0
196             elif gl >= 10.0**(5):
197                 gl_n= 10.0**(5)
198             else:
199                 gl_n = gl**(n-1)
200
201
202
203             if gr <= 10.0**(-2):
204                 gr_n = 1.0
205             elif gr >= 10.0**(5):
206                 gr_n= 10.0**(5)
207             else:
208                gr_n = gr**(n-1)
209
210
211
212             aux_v1 = (C2*((v[i+1,j]-v[i-1,j])/(2*dx))  +  v[i,j]*((v[i,j+1] -
                ↪  v[i,j-1])/(2.0*dy)))*Re
213             aux_v2 = (gr_n*(v[i+1,j] - v[i,j])  - gl_n*(v[i,j] -
                ↪  v[i-1,j]))/(dx**2.0)
```

```
214             aux_v3 = (2.0/(dy**2.0))*(gu_n*(v[i,j+1] - v[i,j]) - gd_n*(v[i,j]
                ↪  - v[i,j-1]))
215             aux_v4 = (1.0/(dx*dy))*(gr_n*(u[i+1,j] - u[i+1,j-1]) -
                ↪  gl_n*(u[i,j] - u[i,j-1]))
216
217
218 #
    ↪  ============================================================================
219 #               aux_v1 = (C2*((v[i+1,j]-v[i-1,j])/(2*dx))  +
    ↪  v[i,j]*((v[i,j+1] - v[i,j-1])/(2.0*dy)))*Re
220 #               aux_v2 = ((v[i+1,j] - v[i,j])  - (v[i,j] -
    ↪  v[i-1,j]))/(dx**2.0)
221 #               aux_v3 = (2.0/(dy**2.0))*((v[i,j+1] - v[i,j]) - (v[i,j] -
    ↪  v[i,j-1]))
222 #               aux_v4 = (1.0/(dx*dy))*((u[i+1,j] - u[i+1,j-1]) - (u[i,j] -
    ↪  u[i,j-1]))
223 #
    ↪  ============================================================================
224
225
226
227             v_star[i,j] = v[i,j] + (dt/Re)*(-aux_v1 + aux_v2 + aux_v3 +
                ↪  aux_v4)
228
229     return v_star
230 ##################################################
231
232 ##################################################
233 @njit
234 def update_bc_v_star(v_star, nx, ny):
235
236     for i in range (0,nx):
237         v_star[i,0] = 0
238         v_star[i,ny] = 0
239
240     for j in range (0,ny+1):
241         v_star[-1,j] = -v_star[0,j]
242         v_star[nx,j] = -v_star[nx-1,j]
243
244     return v_star
245
246 ##################################################
247
248 ##################################################
249 #     ############  - PRESSÃO
250 @njit
251 def calculate_pressure(p, u_star, v_star, nx, ny, dx, dy, dt, tol):
252
253     omega = 1.98
254     Lambda = 0.0
255     R = 0.0
```

```
256     erro = 1
257     while erro > tol:
258         R_max = 0
259         for i in range (0,nx):
260             for j in range (0,ny):
261                 if i == 0 and j == 0:
262                     Lambda = -((1/dx**2) + (1/dy**2))
263                     R = ((u_star[i+1,j] - u_star[i,j])/(dt*dx)) +
                        ↪  ((v_star[i,j+1] - v_star[i,j])/(dt*dy)) - ((p[i+1,j]
                        ↪  -p[i,j])/(dx**2) + (p[i,j+1] - p[i,j])/(dy**2))
264                 elif i == 0 and j == ny-1:
265                     Lambda = -((1.0/dx**2) + (1.0/dy**2))
266                     R = ((u_star[i+1,j] - u_star[i,j])/(dt*dx)) +
                        ↪  ((v_star[i,j+1] - v_star[i,j])/(dt*dy)) - ((p[i+1,j]
                        ↪  -p[i,j])/(dx**2) + (p[i,j-1] - p[i,j])/(dy**2))
267
268                 elif i == nx-1 and j == 0:
269                     Lambda = -((1.0/dx**2) + (1.0/dy**2))
270                     R = ((u_star[i+1,j] - u_star[i,j])/(dt*dx)) +
                        ↪  ((v_star[i,j+1] - v_star[i,j])/(dt*dy)) - ((p[i-1,j]
                        ↪  -p[i,j])/(dx**2) + (p[i,j+1] - p[i,j])/(dy**2))
271                 elif i == nx-1 and j == ny-1:
272                     Lambda = -((1.0/dx**2) + (1.0/dy**2))
273                     R = ((u_star[i+1,j] - u_star[i,j])/(dt*dx)) +
                        ↪  ((v_star[i,j+1] - v_star[i,j])/(dt*dy)) - ((p[i-1,j]
                        ↪  -p[i,j])/(dx**2) + (p[i,j-1] - p[i,j])/(dy**2))
274                 elif i == 0:
275                     Lambda = -((1.0/dx**2) + (2.0/dy**2))
276                     R = ((u_star[i+1,j] - u_star[i,j])/(dt*dx)) +
                        ↪  ((v_star[i,j+1] - v_star[i,j])/(dt*dy)) - ((p[i+1,j]
                        ↪  -p[i,j])/(dx**2) + (p[i,j+1] - 2*p[i,j] +
                        ↪  p[i,j-1])/(dy**2))
277                 elif i == nx-1:
278                     Lambda = -((1.0/dx**2) + (2.0/dy**2))
279                     R = ((u_star[i+1,j] - u_star[i,j])/(dt*dx)) +
                        ↪  ((v_star[i,j+1] - v_star[i,j])/(dt*dy)) - ((p[i-1,j]
                        ↪  -p[i,j])/(dx**2) + (p[i,j+1] - 2*p[i,j] +
                        ↪  p[i,j-1])/(dy**2))
280                 elif j == 0:
281                     Lambda = -((2.0/dx**2) + (1.0/dy**2))
282                     R = ((u_star[i+1,j] - u_star[i,j])/(dt*dx)) +
                        ↪  ((v_star[i,j+1] - v_star[i,j])/(dt*dy)) - ((p[i+1,j]
                        ↪  -2*p[i,j] + p[i-1,j])/(dx**2) + (p[i,j+1] -
                        ↪  p[i,j])/(dy**2))
283                 elif j == ny-1:
284                     Lambda = -((2.0/dx**2) + (1.0/dy**2))
285                     R = ((u_star[i+1,j] - u_star[i,j])/(dt*dx)) +
                        ↪  ((v_star[i,j+1] - v_star[i,j])/(dt*dy)) - ((p[i+1,j]
                        ↪  -2*p[i,j] + p[i-1,j])/(dx**2) + (p[i,j-1] -
                        ↪  p[i,j])/(dy**2))
286                 else:
```

```
                    Lambda = -((2.0/dx**2) + (2.0/dy**2))
                    R = ((u_star[i+1,j] - u_star[i,j])/(dt*dx)) +
                     ↪  ((v_star[i,j+1] - v_star[i,j])/(dt*dy)) - ((p[i+1,j]
                     ↪  -2*p[i,j] + p[i-1,j])/(dx**2) + (p[i,j+1] - 2*p[i,j]
                     ↪  + p[i,j-1])/(dy**2))

                R = (R)/(Lambda)
                p[i,j] = p[i,j] + omega*R
                if np.abs(R) > R_max:
                    R_max = np.abs(R)

        erro = R_max



# Reaplicando as c.c.

    for i in range (0,nx):
        p[i,-1] = p[i,0]
        p[i,ny] = p[i,ny-1]
    for j in range (0,ny):
        p[-1,j] = p[0,j]
        p[nx,j] = p[nx-1,j]
    p[-1,-1] = p[0,0]
    p[-1,ny] = p[0,ny-1]
    p[nx,-1] = p[nx-1,0]
    p[nx,ny] = p[nx-1,ny-1]

    return p

#####################################################



#####################################################
#Calculando u
@njit
def calculate_u_new(u_star, nx, ny, dt, p, u, v, dx):

    for i in range (1, nx):
        for j in range (-1, ny+1):
            u[i,j] = u_star[i,j] - (dt)*((p[i,j] - p[i-1,j])/dx)

    return u
#####################################################



#####################################################
#     #Calculando v
@njit
```

```
335 def calculate_v_new(v_star, nx, ny, dt, p, u, v, dy):
336     for i in range (-1, nx+1):
337         for j in range (1, ny):
338             v[i,j] = v_star[i,j] - (dt)*((p[i,j] - p[i,j-1])/dy)
339 
340     return v
341 
342 ################################################
343 
344 
345 ################################################
346 # função de corrente
347 @njit
348 def calculate_stream_function(psi, nx, ny, v, u, dx, dy, tol):
349 
350     for i in range (0,nx):
351         for j in range (0, ny):
352             if i == 0 or i == nx or j == 0 or j == ny:
353                 psi[i,j] = 0
354 
355     Lambda = -(((2.0)/(dx**2.0)) + ((2.0)/(dy**2.0)))
356     erro = 1
357     while erro > tol:
358         R_max = 0
359         for i in range (nx):
360             for j in range (ny):
361                 R = -((v[i,j] - v[i-1,j])/(dx)) + ((u[i,j] - u[i,j-1])/(dy))
                    ↪  - ((psi[i+1,j] -2*psi[i,j] + psi[i-1,j])/(dx**2) +
                    ↪  (psi[i,j+1] - 2*psi[i,j] + psi[i,j-1])/(dy**2))
362                 R = R/Lambda
363                 psi[i,j] = psi[i,j] + 1.98*R
364                 if np.abs(R) > R_max:
365                     R_max = np.abs(R)
366         erro = R_max
367     return psi
368 
369 
370 
371 ################################################
372 
373 @njit
374 def check_for_incompressibility(u_new, v_new, N, dx, dy, div):
375 
376   for i in range(0,N):
377     for j in range(0,N):
378       div[i,j] = (u_new[i+1,j] - u_new[i,j])/dx + (v_new[i,j+1] -
        ↪  v_new[i,j])/dy
379 
380   return div
381 ########################################
382 
```

```
383 ########################################
384
385 @njit
386 def comparativo_lit(Re, n, u_comp, v_comp, nx, ny, u_plot, v_plot):
387     #Retirando os pontos do campo de velocidades para comparação (Comparado
     ↪  com Ghia) (Linhas médias).
388     if n == 1.0:
389
390         for i in range (0, nx):
391             for j in range (0, ny+1):
392                     if i == 65: #int(nx/2.0):
393                         u_comp[j] = u_plot[i,j]
394
395         for j in range (ny):
396             for i in range (nx+1):
397                     if j == 65: #int(ny/2.0):
398                         v_comp[i] = v_plot[i,j]
399
400     return u_comp, v_comp
401
402
403 ##########################################################################################3
404
405
406 #
     ↪  ============================================================================
407 #      ### TEMPERATURA - PERFIL
408 #
     ↪  ============================================================================
409 #
     ↪  ============================================================================
410 #      for i in range(1, nx):
411 #          for j in range (1, ny):
412 #              aux_t1 = u[i,j]*((theta[i+1,j] - theta[i-1,j])/(2*dx)) +
     ↪  v[i,j]*((theta[i,j+1] - theta[i,j-1])/(2*dy))
413 #              aux_t2 = (theta[i+1,j] - 2*theta[i,j] + theta[i-1,j])/(dx**2)
     ↪  + (theta[i,j+1] - 2*theta[i,j] + theta[i,j-1])/(dy**2)
414 #              R = dt*(aux_t1 + aux_t2)
415 #              theta[i,j] = theta[i,j] + R
416 #
417 #      for i in range (nx+1):
418 #          theta[i,-1] = -theta[i,0]
419 #          theta[i,ny] = 2 - theta[i,ny-1]
420 #      for j in range (ny):
421 #          theta[-1,j] = 2*t_p[j] - theta[0,j]
422 #          theta[nx,j] = 2*t_p[j] - theta[nx-1,j]
423 #
424 #      theta[-1,-1] = theta[0,0]
425 #      theta[-1,ny] = theta[0,ny-1]
426 #      theta[nx,-1] = theta[nx-1,0]
427 #      theta[nx,ny] = theta[nx-1,ny-1]
```

```
428 #
429 #
    ↪ ============================================================================
430
431
432 #
    ↪ ============================================================================
433 #         if i == 0 and j == 0:
434 #             t_plot[i,j] = (1/3)*(theta[i,j] + theta[i, j-1] +
    ↪ theta[i-1,j])
435 #         elif i == 0 and j == ny:
436 #             t_plot[i,j] = (1/3)*(theta[i,j] + theta[i, j-1] +
    ↪ theta[i-1,j-1])
437 #         elif i == nx and j == 0:
438 #             t_plot[i,j] = (1/3)*(theta[i,j] + theta[i-1,j] +
    ↪ theta[i-1,j-1])
439 #         elif i == nx and j == ny:
440 #             t_plot[i,j] = (1/3)*(theta[i, j-1] + theta[i-1,j] +
    ↪ theta[i-1,j-1])
441 #         else:
442 #             t_plot[i,j] = (1/4)*(theta[i,j] + theta[i, j-1] +
    ↪ theta[i-1,j] + theta[i-1,j-1])
443 #
444 #
    ↪ ============================================================================
```