ESTUDO DA IMPLEMENTAÇÃO DE VÁLVULAS DE TESLA EM TROCADORES DE CALOR UTILIZANDO O MÉTODO DE ELEMENTOS FINITOS

Pedro Mattos da Silva

Projeto de Graduação apresentado ao Curso de Engenharia Mecânica da Escola Politécnica, Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Engenheiro.

Orientador: Gustavo Rabello dos Anjos

Rio de Janeiro

Janeiro de 2023

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
Departamento de Engenharia Mecânica
DEM/POLI/UFRJ

ESTUDO DA IMPLEMENTAÇÃO DE VÁLVULAS DE TESLA EM TROCADORES DE CALOR UTILIZANDO O MÉTODO DE ELEMENTOS FINITOS

Pedro Mattos da Silva

PROJETO FINAL SUBMETIDO AO CORPO DOCENTE DO DEPARTAMENTO DE ENGENHARIA MECÂNICA DA ESCOLA POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO MECÂNICO.

Aprovado por:

Prof. Gustavo Rabello dos Anjos

Prof. Roney Leon Thompson

Prof. Marcelo José Colaço

RIO DE JANEIRO, RJ - BRASIL
JANEIRO DE 2023

Silva, Pedro Mattos da

ESTUDO DA IMPLEMENTAÇÃO DE VÁLVULAS DE TESLA EM TROCADORES DE CALOR UTILIZANDO O MÉTODO DE ELEMENTOS FINITOS/ Pedro Mattos da Silva. – Rio de Janeiro: UFRJ/Escola Politécnica, 2023.

XII, 73 p.: il.; 29, 7cm.

Orientador: Gustavo Rabello dos Anjos

Projeto de Graduação – UFRJ/ Escola Politécnica/ Curso de Engenharia Mecânica, 2023.

Referências Bibliográficas: p. 53 – 55.

1. Válvula de Tesla. 2. Método de Elementos Finitos. 3. Mecânica dos Fluidos. 4. Transferência de Calor. 5. Python. I. Anjos, Gustavo Rabello dos. II. Universidade Federal do Rio de Janeiro, UFRJ, Curso de Engenharia Mecânica. III. ESTUDO DA IMPLEMENTAÇÃO DE VÁLVULAS DE TESLA EM TROCADORES DE CALOR UTILIZANDO O MÉTODO DE ELEMENTOS FINITOS.

"Tudo o que temos que decidir é o que fazer com o tempo que nos é dado." – Gandalf

Agradecimentos

Agradeço primeiramente a Deus pelo dom da vida.

Agradeço à minha família, especialmente ao meu pai José Robson e à minha mãe Ana Lúcia, por todo o suporte prestado durante a minha jornada até aqui, sem o apoio deles nada disso seria possível.

Agradeço à minha namorada Gabriele por ter estado ao meu lado nos últimos anos, tanto nos momentos felizes como nos momentos difíceis da minha vida, sem ela a caminhada teria sido muito mais pesada.

Agradeço a todos os companheiros da Mecânica, mas especialmente ao meu amigo Luis Felipe, por toda a parceria durante a formação e por ter sido o principal responsável pela escolha do tema desse trabalho.

Agradeço também à equipe Minerva eRacing por ter me tirado da zona de conforto, me ensinado muito sobre engenharia mecânica e trabalho em equipe e me proporcionado diversas experiências incríveis.

Finalmente, agradeço ao professor Gustavo Rabello, que com seu altíssimo conhecimento de mecânica dos fluidos computacional e excelente didática ao ministrar a disciplina, me inspirou a seguir com essa área para o projeto final.

Resumo do Projeto de Graduação apresentado à Escola Politécnica/UFRJ como parte dos requisitos necessários para a obtenção do grau de Engenheiro Mecânico

ESTUDO DA IMPLEMENTAÇÃO DE VÁLVULAS DE TESLA EM TROCADORES DE CALOR UTILIZANDO O MÉTODO DE ELEMENTOS FINITOS

Pedro Mattos da Silva

Janeiro/2023

Orientador: Gustavo Rabello dos Anjos

Programa: Engenharia Mecânica

A Válvula de Tesla é um dispositivo que foi inventado por Nikola Tesla em 1920 e tem como função fornecer uma maior resistência à passagem de fluido em uma direção do que na outra. Nesse trabalho é analisada a implementação de válvulas de Tesla em trocadores de calor microfluídicos. Para tanto são levadas em conta dois aspectos: a diodicidade e a transferência de calor propriamente dita. Para a realização do estudo, são implementadas as equações de Navier-Stokes e de conservação de energia em um programa iterativo em *Python* utilizando-se o método de elementos finitos. A partir do método, são realizadas simulações de escoamentos em uma válvula de Tesla para diferentes números de Reynolds e analisados os valores da diodicidade para cada um deles. Além disso, considerando as paredes da válvula à temperatura constante, é calculada a temperatura de saída do fluido. O valor da temperatura de saída é comparado com o valor obtido como referência para um tipo de canal mais comumente encontrado em trocadores de calor. Os resultados obtidos demonstraram que a diodicidade cresce conforme se aumenta o número de Reynolds. Além disso, com relação à transferência de calor, não foram observadas vantagens na utilização do dispositivo de Tesla em comparação com a geometria de referência.

Abstract of Undergraduate Project presented to POLI/UFRJ as a partial fulfillment of the requirements for the degree of Mechanical Engineer

STUDY OF THE IMPLEMENTATION OF TESLA VALVES IN HEAT EXCHANGERS BY FINITE ELEMENTS METHOD

Pedro Mattos da Silva

January/2023

Advisor: Gustavo Rabello dos Anjos

Department: Mechanical Engineering

The Tesla Valve is a device invented by Nikola Tesla in 1920. That apparatus offers a greater resistance to fluid flow in one direction than in the other. In this work, the implementation of Tesla valves in microfluidic heat exchangers is analyzed. For that, two aspects are taken into account: diodicity and heat transfer itself. To carry out the study, the Navier-Stokes and energy equations are implemented in an iterative *Python* program using the finite element method. From the method, simulations of flows in a Tesla valve for different Reynolds numbers are performed and the diodicity values are computed. Besides that, considering the walls of the valve at constant temperature, the outlet temperature of the fluid is calculated. That value is compared with the value obtained as a reference for a type of channel most commonly found in heat exchangers. The achieved results demonstrated that diodicity increases as the Reynolds number increases. Whit regard to heat transfer, no advantages were observed in using the Tesla device compared to the reference geometry.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Válvula de Tesla

A válvula de Tesla é um dispositivo que foi inventado por Nikola Tesla em 1920 [1]. Tal artefato é definido pelo mesmo como uma espécie de válvula de retenção sem partes móveis, uma vez que é um conduíte completamente rígido e oferece uma menor resistência à passagem de fluido em uma direção (sentido direto) do que em outra direção (sentido inverso). A Figura 1.1 a seguir representa o esquema dessa válvula, sendo o escoamento da esquerda para a direita no sentido inverso e o da direita para a esquerda no sentido direto.

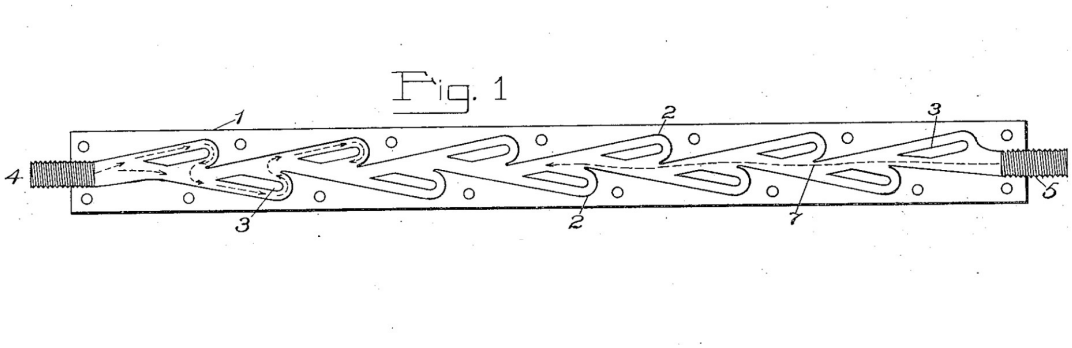


Figura 1.1: Esboço do dispositivo valvular apresentado por Tesla

O funcionamento desse dispositivo é explicado por seu próprio inventor da seguinte forma: "o interior desse conduíte é provido de alargamentos, reentrâncias, projeções, chicanas ou vasos os quais, enquanto não oferecem praticamente nenhuma resistência para a passagem do fluido em uma direção, além do atrito superfi-

cial, constituem uma barreira quase intransponível para esse escoamento na direção contrária devido a maiores ou menores expansões, contrações, deflexões, mudanças de direção, paradas e partidas e concomitantes, rápidas e sucessivas transformações das energias de pressão e velocidade."(TESLA, N., "Valvular conduit", U.S. Patent No. 1,329,559, 1920, p. 1, tradução nossa).

A eficácia desse tipo de válvula pode ser medida pela grandeza denominada diodicidade, que é definida como a razão entre a perda de carga no sentido inverso e a perda de carga no sentido direto para uma mesma vazão de fluido.

$$D_i = \left(\frac{\Delta P_i}{\Delta P_d} \right) \tag{1.1}$$

Segundo Tesla, os efeitos da perda de carga do conduíte valvular são maximizados quando a vazão de fluido é fornecida em pulsos, especialmente quando estes são repentinos e em altas frequências.

1.2 Motivação

Em novembro de 2021, a companhia chinesa de tecnologia Xiaomi anunciou seu novo modelo de arrefecimento a ser utilizado em seus smartphones mais avançados denominada "*Loop LiquidCool Technology*" [2]. Inspirada em uma tecnologia utilizada na indústria aeroespacial, a empresa diz ter conseguido uma capacidade de arrefecimento duas vezes maior do que as de câmaras de vapor tradicionais.

Resumidamente, essa nova tecnologia é composta por um ciclo contendo um evaporador, um condensador e uma câmara de reabastecimento. No evaporador, quando o smartphone está com uma alta carga de processamento e, consequentemente, aquecido, o fluido refrigerante é vaporizado e se expande em direção ao condensador, uma região mais fria do smartphone onde o fluido é condensado. Em seguida, o fluido segue por canais porosos até a câmara de reabastecimento por meio do fenômeno de capilaridade. Na câmara de reabastecimento foi implementada uma microestrutura de válvulas de Tesla, de modo a evitar que os gases do evaporador retornem no sentido inverso do escoamento. As figuras à seguir trazem representações da"*Loop LiquidCool Technology*".

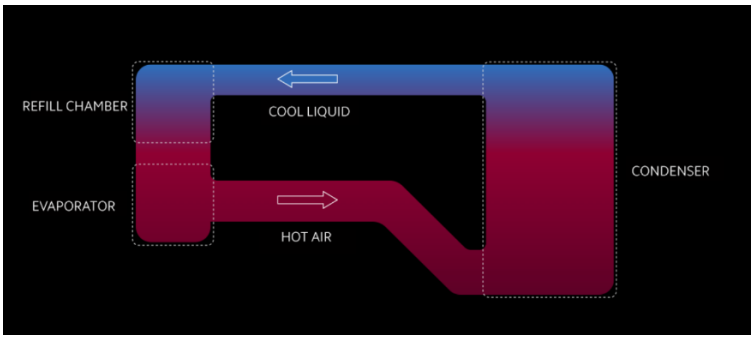


Figura 1.2: Representação do ciclo de resfriamento proposto pela Xiaomi

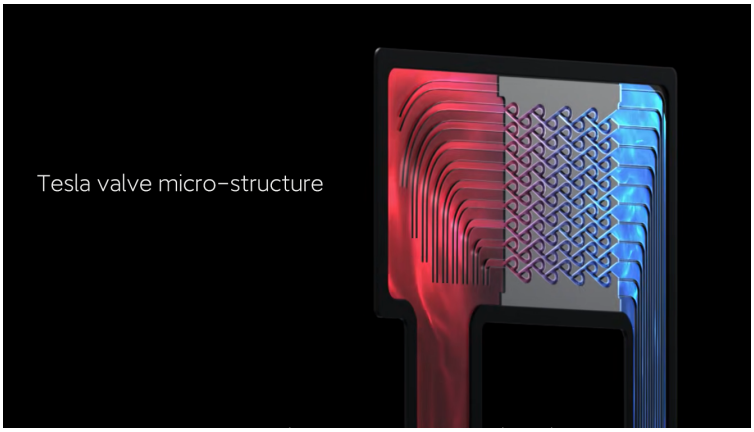


Figura 1.3: Representação da microestrutura de válvulas de Tesla proposta pela Xiaomi

De maneira geral, a utilização de válvulas de Tesla tem sido relativamente pouco explorada pela literatura. Todavia, o recente anúncio da Xiaomi trouxe um novo holofote para a aplicabilidade desses dispositivos.

1.3 Objetivo

O objetivo desse trabalho é investigar a aplicabilidade de válvulas de Tesla em trocadores de calor microfluídicos, primeiramente avaliando diodicidade para diferentes números de Reynolds e seguidamente avaliando a eficiência na transferência de calor desse tipo de geometria em comparação com geometrias mais comumente utilizadas em trocadores.

O estudo foi realizado por meio de um modelo numérico utilizando o Método de Elementos Finitos (MEF) à partir das formulações de Navier-Stokes e das equações

de conservação de massa e energia. Nesse contexto, foram utilizados o métodos de Galerkin para a discretização espacial dos termos difusivos e da pressão, o semi-Lagrangeano para o tratamento do termo convectivo, que é naturalmente instável, e de diferenças finitas para a discretização temporal. Para as equações de energia, foi utilizado somente o método de Galerkin e o de diferenças finitas no sentido de resolver o campo de temperaturas. Com esse propósito, foram utilizados softwares abertos tanto para a geração das malhas computacionais (*Gmsh*), como para a operacionalização do método e solução das equações (*Python*) e para o pós-processamento (*Paraview*).

Capítulo 2

Revisão Bibliográfica

2.1 Método de Elementos Finitos

O chamado Método de Elementos Finitos (MEF) é um método para solução de problemas de engenharia que vão desde a análise de esforços mecânicos, até simulações de dinâmica dos fluídos e transferência de calor. Embora existam registros anteriores da utilização desse método, o primeiro a utilizar o nome MEF propriamente dito foi o estadunidense Ray Clough em 1960[3], que na época era professor da universidade da Califórnia no campus de Berkley e trabalhava na Boeing[4], sendo o método utilizado para analisar os esforços em fuselagens de aviões.

O MEF consiste em subdividir uma geometria de análise em pequenos elementos, transformando o corpo contínuo em uma malha de elementos finitos de tal maneira que, ao resolver as equações de interesse em cada nó da malha de maneira simultânea, chega-se à uma solução aproximada do problema. A figura 2.1 mostra um exemplo de uma malha de elementos finitos presente em Lewis[5].

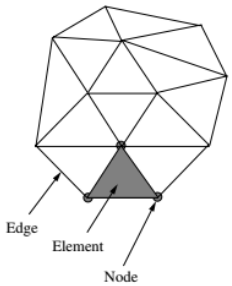


Figura 2.1: Exemplo de malha de elementos finitos com representação de aresta, elemento e nós

Após a publicação dos textos de Clough, a cidade de Berkley virou um polo de pesquisa e desenvolvimento do MEF, desenvolvimento este que foi propulsionado pelo avanço da tecnologia de computadores. Em 1965 foi criado um dos primeiros softwares de elementos finitos, denominado *NASTRAN*, a partir de um forte investimento da NASA. Em seguida, ocorreu a criação do *Ansys* em 1969, um dos mais populares softwares de elementos finitos até os dias de hoje, e a fundação do *ABAQUS* em 1978 [4].

A formulação do Método de Elementos Finitos utilizada nesse trabalho está presente em Fish[4], Lewis[5] e Anjos[6].

2.2 Válvula de Tesla

Como exposto na seção 1.2, válvulas de Tesla têm sido relativamente pouco exploradas pela literatura e pela indústria de maneira geral. Forster *et al.* realizou análises comparando a diodicidade para diferentes velocidades de escoamento em regime permanente em microcanais e publicou sobre sua aplicação em microbombas em 1995[7]. Em 2005, Gamboa, Morris e Forster realizaram um processo de otimização da geometria desse tipo de dispositivo utilizando um *software* comercial de CFD (*Computational Fluid Dynamics*) e conseguiram um aumento médio na diodicidade de 25% para Reynolds variando de 0 a 2000 [8]. A figura 2.2 mostra a diferença entre a geometria padrão utilizada por Forster *et al.* em 1995 e a obtida pelos autores Gamboa, Morris e Forster após a otimização.

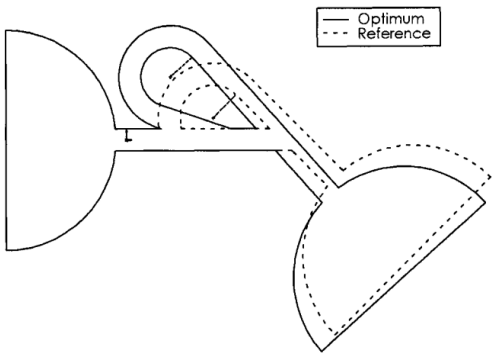


Figura 2.2: Comparação geometria de referência e geometria otimizada de uma válvula de Tesla

Paudel *et al.* estudou os efeitos térmicos em válvulas de Tesla [9]. Por meio de simulações considerando um fluido entrando no conduíte à temperatura constante e as paredes do conduíte também à temperatura constante, chegaram à conclusão de que a diodicidade diminui conforme se aumenta a temperatura de entrada do fluido para líquidos (o efeito contrário ocorre para gases). Além disso, ao comparar a eficiência de troca de calor entre a válvula de Tesla e um canal reto de mesma área ou mesmo volume, chegaram à conclusão de que a transferência de calor no dispositivo de Tesla é significantemente mais eficiente do que no canal reto, isto é, a temperatura de saída do fluído naquele é mais próxima à temperatura da parede do que neste. Entretanto, os autores não exploraram comparações com canais que, além de possuírem a mesma área ou mesmo volume, tivessem um formato semelhante, ou seja, uma região de ocupação parecida. Nesse sentido, tal análise será abordada neste trabalho.

Capítulo 3

Metodologia

3.1 Formulação Utilizada

3.1.1 Equação de Conservação da Massa

A partir do princípio da conservação de massa, sabe-se que, dado um fluido escoando através de um volume de controle $\forall$, a taxa de variação da massa de $\forall$ é igual à diferença entre a quantidade de massa de fluido que entra e que sai desse volume de controle. Esse princípio pode ser representado, em sua forma integral, pela equação 3.1 [10].

$$\frac{\partial}{\partial t}\int_{VC}\rho d\forall + \int_{SC}\rho \mathbf{v}\cdot d\overrightarrow{A} = 0 \tag{3.1}$$

Analisando-se o primeiro termo, considerando-se que a massa específica é constante em todo o volume, pode-se colocar o termo ρ para fora da integral. Dessa forma, a integral torna-se o próprio volume de controle analisado, conforme mostra a equação a seguir:

$$\frac{\partial}{\partial t}\int_{VC}\rho d\forall = \frac{\partial \rho}{\partial t}\int_{VC} d\forall \tag{3.2}$$

Para o segundo termo, a partir do teorema de Gauss, sabe-se que:

$$\int_{SC}\rho \mathbf{v}\cdot d\overrightarrow{A} = \int_{VC}\nabla\cdot\rho\mathbf{v}d\forall \tag{3.3}$$

Portanto, a equação da conservação de massa pode ser escrita na forma:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \mathbf{v} = 0 \tag{3.4}$$

Considerando que o escoamento é incompressível e, com isso, sua massa específica não se altera no tempo, temos que o primeiro termo da equação 3.3 se anula, sobrando apenas o segundo termo que, para $\rho \neq 0$, torna a seguinte equação é verdadeira:

$$\nabla \cdot \mathbf{v} = 0 \tag{3.5}$$

3.1.2 Equações de Conservação da Quantidade de Movimento

A equação de conservação da quantidade de movimento aplicada à fluidos é obtida a partir do balanço dessa propriedade dentro de um volume de controle e pode ser apresentada em sua forma vetorial do seguinte modo [11]:

$$\rho \left[\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v} \right] = -\nabla p + \nabla \cdot \tau + \rho \mathbf{g} \tag{3.6}$$

Em que τ é o chamado tensor de viscosidade. Para fluidos Newtonianos e incompressíveis, tal valor é definido como:

$$\tau = \mu(\nabla \mathbf{v} + \nabla \mathbf{v}^T) \tag{3.7}$$

Considerando que:

$$\nu = \frac{\mu}{\rho} \tag{3.8}$$

Pode-se escrever a equação da conservação de quantidade de movimento na forma da chamada equação de Navier-Stokes, como apresentado a seguir.

$$\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v} = -\frac{1}{\rho} \nabla p + \nabla \cdot \left[\nu(\nabla \mathbf{v} + \nabla \mathbf{v}^T) \right] + \mathbf{g} \tag{3.9}$$

O termo $(\mathbf{v} \cdot \nabla \mathbf{v})$ é o chamado termo convectivo e está associado ao transporte da quantidade de movimento pelo próprio escoamento. O termo $\{\nabla \cdot [\nu(\nabla \mathbf{v} + \nabla \mathbf{v}^T)]\}$

é o chamado termo difusivo, estando associado ao transporte da quantidade de movimento a partir da difusão molecular das partículas do fluido.

3.1.3 Transferência de Calor

A equação de governo para a análise da transferência de calor em fluídos pode ser obtida a partir da equação de conservação de energia sobre um volume de controle. A dedução dessa equação pode ser encontrada em Lewis[5] e, para um caso sem geração, considerando a condutividade térmica constante, possui a seguinte forma:

$$\frac{\partial T}{\partial t} + \mathbf{v} \cdot \nabla T = \alpha \nabla^2 T \tag{3.10}$$

Em que T é a temperatura do fluído e α é a chamada difusividade térmica que é definida como:

$$\alpha = \frac{k}{\rho c_p} \tag{3.11}$$

Na equação 3.10, analogamente à equação de Navier-Stokes, o termo $(\mathbf{v} \cdot \nabla T)$ é o termo convectivo e o termo $(\alpha \nabla^2 \mathrm{T})$ é o termo difusivo.

3.1.4 Números Adimensionais

Os chamados números adimensionais, representam combinações específicas de grandezas dimensionais caracterizadoras dos fluidos, como velocidade, pressão, massa específica, viscosidade, capazes de descrever o comportamento de escoamentos, principalmente quanto às forças que são predominantes. Nesse trabalho, os números adimensionais de interesse são o número de Reynolds (Re) e o número de Prandtl (Pr).

O número de Reynolds é a grandeza adimensional de maior relevância na mecânica dos fluidos e é definido como:

$$Re = \frac{LU}{\nu} \tag{3.12}$$

Nessa equação, L é o comprimento característico do escoamento e U é a velocidade. Fisicamente, o número de Reynolds representa a razão entre as forças de inércia (associadas à convecção) e as forças viscosas (associadas à difusão). Nesse

sentido, em um escoamento com baixo Re, as forças difusivas tendem a dominar sobre as forças convectivas e o escoamento é controlado e chamado laminar. Para Reynolds altos, as forças de inércia passam a dominar tornando o escoamento caótico, levando ao chamado regime turbulento.

O número de Prandtl é a grandeza adimensional que representa a relação entre a difusividade viscosa e a difusividade térmica.

$$Pr = \frac{\nu}{\alpha} \tag{3.13}$$

3.2 Método das Diferenças Finitas

O Método das Diferenças Finitas (MDF) é um método utilizado para aproximar derivadas a partir de valores conhecidos da função próximos do ponto de interesse. Tal método é obtido a partir da expansão de funções em série de Taylor.

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2!}f''(x) + \frac{h^3}{3!}f'''(x) + ... \tag{3.14}$$

A série de Taylor é uma série infinita e a aproximação será mais precisa quanto maior for o número de termos a serem considerados. Para se obter uma aproximação de primeira ordem da derivada, desconsidera-se os termos contendo derivadas de ordem 2 em diante e obtém-se:

$$f'(x) = \frac{f(x+h) - f(x)}{h} \tag{3.15}$$

Tal aproximação será utilizada para a discretização temporal do problema analisado.

3.3 Método de Elementos Finitos

As disposições gerais sobre o MEF foram expostas na seção 2.1. O método pode ser resumido nas seguintes etapas [5]:

1. Obtenção do modelo matemático a partir do problema físico;
2. Discretização espacial e temporal do modelo matemático;
3. Discretização do domínio;

4. Geração da malha computacional;
5. Solução do problema;
6. Pós-processamento e análise de convergência.

A obtenção do modelo matemático é obtido a partir da interpretação do problema físico e aplicação das equações básicas de governo do problema, como as de conservação de massa, quantidade de movimento e energia, resultando em uma equação diferencial. Tal equação é denominada *forma forte*.

A discretização espacial envolve a obtenção da forma variacional do problema, chamada de *forma fraca*, seguida da utilização das chamadas funções de forma para caracterizar o valor de cada variável dentro de um elemento de malha. A justaposição dessas caracterizações em cada elemento é utilizada para descrever todo o campo.

A *forma fraca* representa uma forma integral de escrever uma equação de governo de modo que ela seja equivalente à equação original, porém com a ordem reduzida.

Será demonstrada a implementação da $formafraca$ e a utilização do MEF a partir das equações 3.9 (Navier-Stokes) e 3.3 (conservação de massa). Que são as equações de governo para o problema estudado nesse trabalho.

$$\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v} = -\frac{1}{\rho}\nabla p + \nabla \cdot \left[\nu(\nabla \mathbf{v} + \nabla \mathbf{v}^T)\right] + \mathbf{g} \tag{3.16}$$

$$\nabla \cdot \mathbf{v} = 0 \tag{3.17}$$

O termo difusivo da equação 3.16 pode ser escrito como:

$$\nabla \cdot [\nu(\nabla \mathbf{v} + \nabla \mathbf{v}^T)] = \nu[\nabla \cdot (\nabla \mathbf{v}) + \nabla \cdot (\nabla \mathbf{v}^T)] = \nu[\nabla^2 \mathbf{v} + \nabla \underbrace{(\nabla \cdot \mathbf{v})}_{=0}] \tag{3.18}$$

Portanto, a equação de Navier-Stokes assume a seguinte forma:

$$\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{v} + \mathbf{g} \tag{3.19}$$

Tais equações devem ser resolvidas de forma acoplada, uma vez que a equação 3.19, para o caso de escoamento bidimensional que será estudado, representa na verdade duas equações, uma para cada direção (x e y), e existem três variáveis primitivas no problema: as duas componentes da velocidade e a pressão. Por isso a necessidade de se acoplar a equação de conservação de massa ao sistema.

Para chegar-se à *forma fraca*, primeiramente é preciso multiplicar as equações 3.19 e 3.17 por funções arbitrárias, as quais chamaremos de $w(\mathbf{x})$ e $q(\mathbf{x})$, respectivamente. Aqui, o vetor $\mathbf{x}$ representa as coordenadas x e y, e a composição das funções $w(\mathbf{x})$ para as equações nas direções x e y será chamada de $\mathbf{w}$. Em seguida, é feita a integração das equações no domínio do problema. Além disso, será desconsiderada a influência do campo gravitacional no escoamento.

$$\int_{\Omega} \left\{ \frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v} + \frac{1}{\rho} \nabla p - \nu \nabla^2 \mathbf{v} \right\} \cdot \mathbf{w} d\Omega = 0 \tag{3.20}$$

$$\int_{\Omega} \left[\nabla \cdot \mathbf{v} \right] q d\Omega = 0 \tag{3.21}$$

Organizando os termos da equação 3.20 chega-se a:

$$\int_{\Omega} \underbrace{\left\{ \frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v} \right\}}_{\frac{D\mathbf{v}}{Dt}} \cdot \mathbf{w} d\Omega + \int_{\Omega} \left\{ \frac{1}{\rho} \nabla p \right\} \cdot \mathbf{w} d\Omega - \int_{\Omega} \left\{ \nu \nabla^2 \mathbf{v} \right\} \cdot \mathbf{w} d\Omega = 0 \tag{3.22}$$

O primeiro termo da equação 3.22 pode ser escrito na forma de derivada material como mostrado acima, uma vez que tal formulação será essencial para o desenvolvimento do método semi-Lagrangeano que será abordado posteriormente. Além disso, realiza-se a integração por parte no segundo termo, para que seja possível a separação do termo de contorno e no terceiro termo para, além de separar os termos de contorno, reduzir-se a ordem das derivadas.

$$\int_{\Omega} \frac{1}{\rho} \nabla p \cdot \mathbf{w} d\Omega = \frac{1}{\rho} \int_{\Gamma} \mathbf{w} p \cdot \mathbf{n} d\Gamma - \frac{1}{\rho} \int_{\Omega} p \nabla \cdot \mathbf{w} d\Omega \tag{3.23}$$

$$\int_{\Omega} \nu \nabla^2 \mathbf{v} \cdot \mathbf{w} d\Omega = \nu \int_{\Gamma} \mathbf{w} \nabla \mathbf{v} \cdot \mathbf{n} d\Gamma - \nu \int_{\Omega} \nabla \mathbf{w} : \nabla \mathbf{v} d\Omega \tag{3.24}$$

Para problemas de fluidos incompressíveis, como é o caso em análise, as condições de contorno são todas do tipo *Dirichlet*, na qual o valor da variável é conhecido, ou *Neumann* homogêneo, na qual a taxa de variação da variável no contorno é zero. Nesse tipo de situação, convém e satisfaz as equações que o valor da função $w(\mathbf{x})$ nesses pontos seja igual a zero. Dessa forma, a *forma fraca* final do problema com as devidas substituições está apresentada na equação 3.25.

$$\int_\Omega \frac{D\mathbf{v}}{Dt} \cdot \mathbf{w} d\Omega - \frac{1}{\rho} \int_\Omega p \nabla \cdot \mathbf{w} d\Omega + \nu \int_\Omega \nabla \mathbf{w} : \nabla \mathbf{v} d\Omega = 0 \tag{3.25}$$

$$\int_\Omega \left[\nabla \cdot \mathbf{v}\right] q d\Omega = 0 \tag{3.26}$$

O MEF consiste em afirmar que as soluções tentativas $\mathbf{v(x)}$, isto é, $u(\mathbf{x})$ e $v(\mathbf{x})$, e $p(\mathbf{x})$ que satisfazem as equações acima para qualquer $w(\mathbf{x})$ e $q(\mathbf{x})$ com $w(\Gamma) = 0$ é a solução do problema, uma vez que as formas forte e fraca são equivalentes.

Ao discretizarmos o problema, uma solução tentativa geralmente utilizada é a do chamado método de Galerkin segundo o qual as soluções tentativas para cada elemento têm a seguinte forma:

$$u(\mathbf{x}, t) \approx \sum_{i=1}^{NV} N_i(\mathbf{x}) u_n(t) \tag{3.27}$$

$$v(\mathbf{x}, t) \approx \sum_{i=1}^{NV} N_i(\mathbf{x}) v_n(t) \tag{3.28}$$

$$w_x(\mathbf{x}, t) \approx \sum_{j=1}^{NV} N_j(\mathbf{x}) w_{x_n}(t) \tag{3.29}$$

$$w_y(\mathbf{x}, t) \approx \sum_{j=1}^{NV} N_j(\mathbf{x}) w_{y_n}(t) \tag{3.30}$$

$$p(\mathbf{x}, t) \approx \sum_{i=1}^{NP} N_i(\mathbf{x}) p_n(t) \tag{3.31}$$

$$q(\mathbf{x}, t) \approx \sum_{i=j}^{NP} N_j(\mathbf{x}) q_n(t) \tag{3.32}$$

Nas equações acima, NV e NP, representam respectivamente o número de pontos do domínio em que serão discretizado as velocidades e a pressão. Na seção 3.3.1 será demonstrada a necessidade de que esses dois valores sejam distintos. As funções $N_{i,j}$ são as denominadas funções de forma, que são escolhidas de modo à satisfazer as igualdades acima e são amplamente discutidas pela literatura. Aqui utilizaremos as funções de forma apresentadas em Lewis[5].

Após a realização das devidas substituições, são obtidas as seguintes equações para cada elemento de malha:

$$\sum_i \int_{\Omega^e} \sum_j \frac{Du_i}{Dt} N_i N_j w_{x_j} d\Omega - \frac{1}{\rho} \sum_i \int_{\Omega^e} \sum_j N_i \frac{\partial Nj}{\partial x} w_{x_j} p_i d\Omega + \nu \sum_i \int_{\Omega^e} \sum_j \left(\frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} \right) w_{x_j} u_i d\Omega = 0 \tag{3.33}$$

$$\sum_i \int_{\Omega^e} \sum_j \frac{Dv_i}{Dt} N_i N_j w_{y_j} d\Omega - \frac{1}{\rho} \sum_i \int_{\Omega^e} \sum_j N_i \frac{\partial Nj}{\partial y} w_{y_j} p_i d\Omega + \nu \sum_i \int_{\Omega^e} \sum_j \left(\frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} \right) w_{y_j} v_i d\Omega = 0 \tag{3.34}$$

$$\sum_i \int_{\Omega^e} \sum_j \left(\frac{\partial N_i}{\partial x} u_i + \frac{\partial N_i}{\partial y} v_i \right) N_j q_j d\Omega = 0 \tag{3.35}$$

É necessário destacar que pode-se dividir cada uma das equações por suas respectivas funções peso. Além disso, é possível se chegar a algumas relações nas equações acime que constituem as chamadas matrizes elementares, que serão discutidas com maiores detalhes na próxima seção.

$$\int_{\Omega^e} N_i N_j d\Omega^e \rightarrow \text{Matriz elementar de massa } \mathbf{m^e} \tag{3.36}$$

$$\int_{\Omega^e} N_i \frac{\partial N_j}{\partial x} d\Omega^e \rightarrow \text{Matriz elementar de gradiente } \mathbf{g_x^e} \tag{3.37}$$

$$\int_{\Omega^e} \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} d\Omega^e \rightarrow \text{Matriz elementar de rigidez } \mathbf{k_x^e} \tag{3.38}$$

Nesse sentido, conforme exposto por Lewis, é possível escrever a justaposição dos somatórios das matrizes elementares em formato de matrizes globais. Dessa forma, é interessante organizar todas essas equações em um sistema linear único a ser resolvido para cada passo de tempo. Ao substituir-se os valores das matrizes globais nas equações 3.33, 3.34 e 3.35, é obtido:

$$\mathbf{M} \frac{Du}{Dt} - \frac{1}{\rho} \mathbf{G_x} p + \nu \mathbf{K} u = 0 \tag{3.39}$$

$$\mathbf{M} \frac{Dv}{Dt} - \frac{1}{\rho} \mathbf{G_y} p + \nu \mathbf{K} v = 0 \tag{3.40}$$

$$\mathbf{D}\,\mathbf{v} = 0 \tag{3.41}$$

Sendo $\mathbf{D}$ a chamada matriz de divergente, definida como:

$$\mathbf{D} = \mathbf{G}^T \tag{3.42}$$

3.3.1 Discretização do Domínio

A discretização espacial significa dividir o domínio em um número finito de blocos chamados de elementos. Tais blocos podem possuir diferentes formas e tamanhos. Quanto maior o número de divisões feitas em um domínio, mais precisa tende a ser a solução, porém, um maior poder computacional será exigido.

Nesse trabalho, a discretização será em um domínio bidimensional e serão utilizados elementos de forma triangular, que são uma das geometrias mais simples para o MEF, todavia, o problema exige uma restrição quanto à utilização do elemento triangular, como será explorado à seguir.

Condição de LBB

Ao serem utilizados elementos triangulares simples para a discretização do domínio, teremos o mesmo número de pontos para análise das velocidades e da pressão. Nesse sentido, a matriz final do sistema linear gerado a partir do método de Galerkin será simétrica, e poderão ocorrer problemas ao se tentar invertê-la para a obtenção da solução do sistema, como o fato da matriz inversa possivelmente não existir [6].

A condição de Ladyzhenskaya-Babuska-Brezzi, também conhecida como *LBB* é uma condição de estabilidade, no que tange a discretização do domínio, para a solução das equações de Navier-Stokes utilizando o método de Galerkin para elementos finitos.

Uma alternativa à escolha do elemento triangular simples, de modo que a condição de *LBB* seja satisfeita, é a utilização do chamado elemento triangular *MINI* da família *Taylor-Hood*, no qual a pressão é avaliada nos vértices do triângulo, e a velocidade é avaliada tanto nos vértices, quanto no centroide do elemento, como mostra a figura a seguir retirada de Anjos[6].

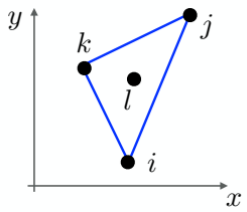


Figura 3.1: Exemplo de elemento triangular MINI.

Montagem das Matrizes

Como citado no início dessa seção, os somatórios das funções de forma que são utilizadas na resolução de problemas a partir do MEF podem ser reescritos como matrizes elementares que são posteriormente acopladas de forma a transformar o problema em um único sistema linear a ser resolvido.

A transformação das funções de forma em matrizes a partir da geometria dos elementos está amplamente discutida na literatura. Para a estruturação das matrizes para o elemento *MINI*, precisa-se primeiramente estruturar as do elemento triangular simples, visto que aquelas são uma ampliação dessas.

Considerando $x_i,\ y_i,\ x_j,\ y_j,\ x_k,\ y_k$ as coordenadas dos vértices do elemento, temos que sua área pode ser obtida como:

$$A = \frac{1}{2} det \begin{bmatrix} 1 & x_i & y_i \\ 1 & x_j & y_j \\ 1 & x_k & y_k \end{bmatrix} \tag{3.43}$$

Além disso, é necessário prescrever os seguintes parâmetros:

$$\begin{aligned} a_i &= x_j y_k - x_k y_j \qquad b_i = y_j - y_k \qquad c_i = x_k - x_j \\ a_j &= x_k y_i - x_i y_k \qquad b_j = y_k - y_i \qquad c_j = x_i - x_k \\ a_k &= x_i y_j - x_j y_i \qquad b_k = y_i - y_j \qquad c_k = x_j - x_i \end{aligned} \tag{3.44}$$

Nesse sentido, as funções de forma podem ser escritas como:

$$N_i = \frac{1}{2A}(a_i + b_i x + c_i y) \tag{3.45}$$

$$N_j = \frac{1}{2A}(a_j + b_j x + c_j y) \tag{3.46}$$

$$N_k = \frac{1}{2A}(a_k + b_k x + c_k y) \tag{3.47}$$

Dessa forma, a partir das definições apresentadas nas equações 3.36, 3.37 e 3.38 aplicadas para cada elemento, chega-se as seguintes matrizes locais:

$$\mathbf{m^e} = \frac{A}{12}\begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} \tag{3.48}$$

$$\mathbf{g_x^e} = \frac{1}{6}\begin{bmatrix} b_i & b_j & b_k \\ b_i & b_j & b_k \\ b_i & b_j & b_k \end{bmatrix} \tag{3.49}$$

$$\mathbf{g_y^e} = \frac{1}{6}\begin{bmatrix} c_i & c_j & c_k \\ c_i & c_j & c_k \\ c_i & c_j & c_k \end{bmatrix} \tag{3.50}$$

$$\mathbf{k^e} = \frac{1}{4A}\begin{bmatrix} b_i^2 + c_i^2 & b_i b_j + c_i c_j & b_i b_k + c_i c_k \\ b_k b_j + c_i c_j & b_j^2 + c_j^2 & b_k b_j + c_k c_j \\ b_i b_k + c_i c_k & b_k b_j + c_k c_j & b_k^2 + c_k^2 \end{bmatrix} \tag{3.51}$$

Para o elemento *MINI* no qual a velocidade é analisada tanto nos vértices do triângulo, como no seu centroide, é necessário definir:

$$z = \frac{1}{4A}(b_j^2 + b_j b_k + b_k^2 + c_j^2 + c_j c_k + c_k^2) \tag{3.52}$$

Nesse sentido, as matrizes elementares ficam da seguinte forma:

$$\mathbf{m^e_{MINI}} = \frac{A}{840}\begin{bmatrix} 83 & 13 & 13 & 45 \\ 13 & 83 & 13 & 45 \\ 13 & 13 & 83 & 45 \\ 45 & 45 & 45 & 243 \end{bmatrix} \tag{3.53}$$

$$\mathbf{g^e_{x_{MINI}}} = \begin{bmatrix} (9/20)\mathbf{g_x^e} + \mathbf{g_x^e}^T & & \\ & & \\ -(9/40)b_i & -(9/40)b_j & -(9/40)b_k \end{bmatrix}_{(4,3)}^{(3,3)} \tag{3.54}$$

$$\mathbf{g}^{\mathbf{e}}_{\mathbf{y}_{\mathbf{MINI}}} = \begin{bmatrix} (9/20)\mathbf{g}^{\mathbf{e}}_{\mathbf{y}} + \mathbf{g}^{\mathbf{e}}_{\mathbf{y}}{}^{T} & \\ & \\ & \\ & {\scriptstyle (3,3)} \\ -(9/40)c_i & -(9/40)c_j & -(9/40)c_k \end{bmatrix}_{(4,3)} \tag{3.55}$$

$$\mathbf{g}^{\mathbf{e}}_{\mathbf{v}_{\mathbf{x}_{\mathbf{MINI}}}} = \begin{bmatrix} & & & -(9/40)b_{\mathrm{i}} \\ (11/20)\mathbf{g}^{\mathbf{e}}_{\mathbf{x}} + (9/20)\mathbf{g}^{\mathbf{e}}_{\mathbf{x}}{}^{T} & & & -(9/40)b_{\mathrm{j}} \\ & & & -(9/40)b_{\mathrm{k}} \\ (9/40)b_i & (9/40)b_j & (9/40)b_k & 0 \end{bmatrix}_{(4,4)} \tag{3.56}$$

$$\mathbf{g}^{\mathbf{e}}_{\mathbf{v}_{\mathbf{y}_{\mathbf{MINI}}}} = \begin{bmatrix} & & & -(9/40)c_{\mathrm{i}} \\ (11/20)\mathbf{g}^{\mathbf{e}}_{\mathbf{y}} + (9/20)\mathbf{g}^{\mathbf{e}}_{\mathbf{y}}{}^{T} & & & -(9/40)c_{\mathrm{j}} \\ & & & -(9/40)c_{\mathrm{k}} \\ (9/40)c_i & (9/40)c_j & (9/40)c_k & 0 \end{bmatrix}_{(4,4)} \tag{3.57}$$

$$\mathbf{k}^{\mathbf{e}}_{\mathbf{MINI}} = \begin{bmatrix} & & & -(27/10)z \\ \mathbf{k}^{\mathbf{e}} + (9/10)z & & & -(27/10)z \\ & & & -(27/10)z \\ -(27/10)z & -(27/10)z & -(27/10)z & (81/10)z \end{bmatrix}_{(4,4)} \tag{3.58}$$

Essas definições podem ser encontradas em Anjos[6]. É importante notar que as matrizes 3.54, 3.55, 3.56, 3.57 e 3.58 são variações das matrizes do elemento linear dispostas dentro de matrizes maiores acrescentando-se outros termos. As matrizes $\mathbf{g}^{e}_{\mathbf{x}_{\mathbf{MINI}}}$ e $\mathbf{g}^{e}_{\mathbf{y}_{\mathbf{MINI}}}$ serão acopladas ao sistema para multiplicar os termos de pressão, por isso elas tem dimensão (4,3).

As matrizes $\mathbf{g}^{\mathbf{e}}_{\mathbf{v}_{\mathbf{x}_{\mathbf{MINI}}}}$ e $\mathbf{g}^{\mathbf{e}}_{\mathbf{v}_{\mathbf{y}_{\mathbf{MINI}}}}$, são matrizes de gradiente no espaço de funções da velocidade. Elas não serão utilizadas na solução das equações de Navier-Stokes, pois o termo convectivo foi acoplado dentro da derivada material para a implementação do método semi-Lagrangeano, que será demonstrado adiante. Todavia, tais matrizes serão utilizadas para a solução da equação de temperatura, pois, nesse caso, serão utilizados Reynolds menores de modo a garantir-se a estabilidade do problema mesmo com a utilização dos métodos tradicionais.

Faz-se necessária a montagem das matrizes para cada elemento da malha. Posteriormente, essas matrizes são justapostas de modo a ser construído um sistema linear único que englobe todo o domínio.

3.4 Método semi-Lagrangeano

A abordagem Euleriana para escoamentos de fluídos diz respeito à análise das propriedades do escoamento em coordenadas espaciais fixas, enquanto diferentes partículas de fluidos passam por essas coordenadas. O uso exclusivo desse tipo de abordagem para a solução numérica das equações de Navier-Stokes, principalmente para número de Reynolds altos, tende a gerar um alto grau de instabilidade, uma vez que o termo convectivo das equações é não-linear.

A abordagem Lagrangeana consiste em coordenadas móveis que acompanham as partículas de fluído, sendo comumente utilizada para a solução de problemas de física de partículas. A utilização exclusiva de coordenadas Lagrangeanas para a solução de problemas de escoamento também é instável, uma vez que as trajetórias das partículas tornam-se caóticas em um curto espaço de tempo [11].

Nesse sentido, uma das alternativas para se tratar o termo convectivo nessas equações é com a utilização do método semi-Lagrangeano. Tal método consiste em, para cada passo de tempo, calcular a posição das partículas no passo de tempo anterior para cada nó da malha partir da equação da trajetória.

$$\mathbf{x_d^n} = \mathbf{x^{n+1}} - \mathbf{v^n \Delta t} \tag{3.59}$$

Essa posição da partícula é denominada ponto de saída e pode estar em diferentes regiões, inclusive fora do domínio como demonstrado na figura 3.2. Nesse caso, assume-se a posição do vértice mais próximo à posição encontrada.

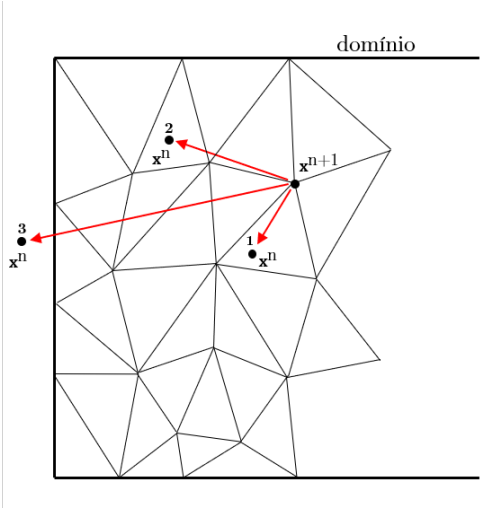


Figura 3.2: Possíveis trajetórias calculadas a partir da posição da partícula em $\mathbf{x}^{n+1}$

Uma vez calculada a posição da partícula no passo de tempo anterior, é feita uma interpolação dos valores da velocidade no elemento no qual a partícula se encontra a partir dos valores da propriedade nos vértices do elemento, obtendo assim uma aproximação para a velocidade no ponto de saída.

A partir dessa abordagem, o termo da derivada material das equações de Navier-Stokes pode ser substituído por uma aproximação de primeira ordem no tempo como segue:

$$\frac{D\mathbf{v}}{Dt} = \frac{\mathbf{v_i^{n+1}} - \mathbf{v_d^n}}{\Delta t} \tag{3.60}$$

Nessa equação, $\mathbf{v_i^{n+1}}$ se torna a icógnita do problema, e o termo $\mathbf{v_d^n}$ é a velocidade da partícula no passo de tempo anterior, calculada a partir da interpolação.

A função para a implementação do semi -Lagrangeano foi desenvolvido pelo professor Gutavo Rabello dos Anjos, orientador desse projeto, e está presente no apêndice A.

Capítulo 4

Implementação do Código Numérico

Aqui serão apresentadas apenas os principais pontos relacionados à implementação do código numérico. O código completo utilizado na solução do problema, bem como o da implementação do método semi-Lagrangeano encontram-se dispostos no apêndice A.

4.1 Equações de Governo

As equações de governo para a solução das equações de Navier-Stokes utilizando o MEF foram discutidas na seção 3.3 e estão representadas pelas equações 3.39, 3.40 e 3.41. Aplicando-se as substituições referentes ao método semi-Lagrangeano, chega-se a:

$$\mathbf{M}\frac{u_i^{n+1} - u_d^n}{\Delta t} - \frac{1}{\rho}\mathbf{G_x}p^{n+1} + \nu\mathbf{K}u^{n+1} = 0 \tag{4.1}$$

$$\mathbf{M}\frac{v_i^{n+1} - v_d^n}{\Delta t} - \frac{1}{\rho}\mathbf{G_y}p^{n+1} + \nu\mathbf{K}v^{n+1} = 0 \tag{4.2}$$

$$\mathbf{D}\,\mathbf{v}^{n+1} = 0 \tag{4.3}$$

Reorganizando os termos, acoplando as equações e implementando as condições de contorno, podemos descrever o problema como um único sistema linear.

$$\underbrace{\begin{bmatrix} \frac{\mathbf{M}}{\Delta t} + \nu\mathbf{K} & & -\frac{1}{\rho}\mathbf{G_x} \\ & \frac{\mathbf{M}}{\Delta t} + \nu\mathbf{K} & -\frac{1}{\rho}\mathbf{G_y} \\ \mathbf{D_x} & \mathbf{D_y} & \end{bmatrix}}_{A} \cdot \underbrace{\begin{bmatrix} u^{n+1} \\ v^{n+1} \\ p^{n+1} \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} \frac{\mathbf{M}}{\Delta t}u^n + c.c._u \\ \frac{\mathbf{M}}{\Delta t}v^n + c.c._v \\ 0 + c.c._p \end{bmatrix}}_{b} \tag{4.4}$$

Considerando NV o número de nós em que está sendo avaliada a velocidade (vértices e centroides dos elementos) e NP o número de nós em que se está sendo avaliada a pressão (vértices), o número total de linhas e colunas da matriz A acima é $2NV + NP$, conforma mostrado a seguir.

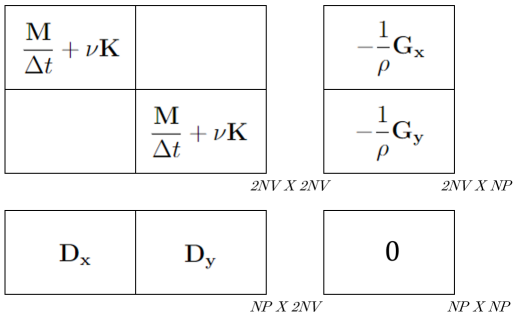


Figura 4.1: Dimensões das matrizes que compõem a matriz A

Pode-se utilizar o mesmo raciocínio apresentado na seção 3.3 para resolver a equação 3.10 (equação da temperatura) utilizando o MEF, chegando-se ao seguinte sistema linear:

$$\left[\frac{\mathbf{M}}{\Delta t} + \alpha\mathbf{K} + \mathbf{v} \cdot (\mathbf{G_x}, \mathbf{G_y})\right] T^{n+1} = \frac{\mathbf{M}}{\Delta t} T^n \tag{4.5}$$

4.2 Geração da Malha Computacional

Para a geração da malha computacional, foi utilizado o software aberto *Gmsh*, que é um software gratuito capaz de gerar diversos tipos de malhas com diferentes formas de elementos e padrões de maneira intuitiva.

O *Gmsh* possui um módulo chamado *Geometry*, no qual pode-se construir o objeto a ser discretizado, definindo a geometria e os contornos do domínio. Além disso, o software permite a importação de geometrias construídas em outros softwares CAD, desde que estejam no formato *STEP*.

No módulo *Mesh*, é possível realizar a discretização do domínio em diferentes elementos. Além disso, é possível, entre outras funções, definir os tipos de elementos que serão utilizados, o refinamento dos elementos, além de campos específicos de refinamento para cada região do domínio. Após a criação da malha computacional, um arquivo *.mesh* deve ser exportado.

4.3 Importação da Malha no Python

Após a criação e exportação da malha no *Gmsh*, é preciso importa-la para o software *Python*. No programa, cada nó da malha carregará uma informação de posição (x e y). Além disso, faz-se necessário criar a chamada matriz de conectividade, também conhecida como matriz IEN, que armazena as informações sobre os nós de cada elemento da malha separadamente.

4.3.1 Criação da Matriz de Conectividade

A figura 4.2 apresenta uma malha triangular simples de oito elementos. Nela, os elementos estão circulados e representados em vermelho, os vértices estão representados em verde e os centroides dos elementos estão representados em lilás.

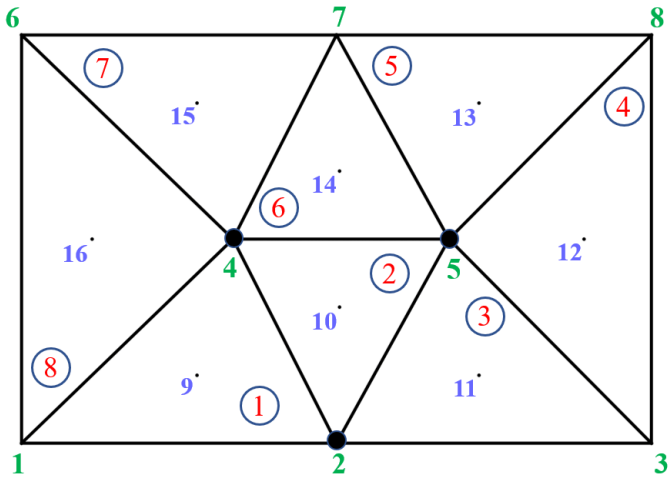


Figura 4.2: Malha triangular simples de oito elementos.

A matriz IEN correspondente à essa malha deve possuir uma linha para cada elemento, e em cada uma dessas linhas, deve-se guardar as informações dos vértices e dos centroides correspondentes. Nesse sentido, a matriz terá quatro colunas, as

três primeiras conterão os vértices do elemento, e a quarta terá a informação do centroide. Ademais, deve-se garantir que, para cada elemento, os vértices sejam armazenados seguindo um mesmo sentido (horário ou anti-horário).

Dessa forma, uma possível IEN para a malha da figura 4.2, adotando-se o sentido anti-horário, seria:

$$\mathbf{IEN} = \begin{bmatrix} 1 & 2 & 4 & 9 \\ 2 & 5 & 4 & 10 \\ 2 & 3 & 5 & 11 \\ 3 & 8 & 5 & 12 \\ 5 & 8 & 7 & 13 \\ 4 & 5 & 7 & 14 \\ 4 & 7 & 6 & 15 \\ 1 & 4 & 6 & 16 \end{bmatrix}$$

A montagem dessa estrutura torna-se extremamente útil, pois carrega a identidade de cada elemento, além de permitir que sejam acessadas informações sobre as posições dos nós dos elemento a partir dos índices da IEN.

4.3.2 Estruturas Auxiliares

Como explicado na seção 3.4, na utilização do método semi-Lagrangeano, para cada passo de tempo, são calculadas as posições das partículas no passo de tempo anterior por meio da equação da trajetória. Dessa forma, para a implementação desse método, faz-se necessária a criação de estruturas auxiliares quer permitam que cada elemento "conheça" informações sobre seus elementos vizinhos. As estruturas auxiliares que serão utilizadas são chamadas de **oElem** e **neighElem** [6].

A matriz denominada **oElem** carrega as informações sobre qual é o elemento vizinho que compartilha a aresta diametralmente oposta a determinado vértice. Essa matriz não leva em conta a posição dos centroides. Considerando novamente a malha da figura 4.2:

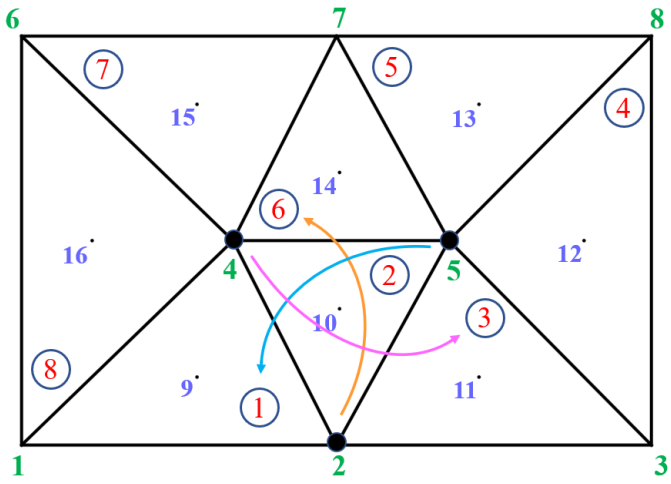


Figura 4.3: Malha triangular simples, formação de estruturas auxiliares.

Percebe-se que a IEN referente ao elemento 2, desconsiderando a quarta coluna da IEN (correspondente aos centroides), é:

$$\mathbf{IEN}[2] = \begin{bmatrix} 2 & 5 & 4 \end{bmatrix}$$

Por conseguinte, adquirindo-se a informação do elemento que compartilha a aresta diametralmente oposta a cada um desses vértices do elemento 2, como mostrado na figura 4.3, chega-se ao vetor **oElem** correspondente:

$$\mathbf{oElem}[2] = \begin{bmatrix} 6 & 1 & 3 \end{bmatrix}$$

Caso não exista nenhum elemento compartilhando a aresta oposta ao vértice (como é o caso do vértice 4 em relação ao elemento 1) é atribuído o valor de -1 nessa posição do vetor. A matriz **oElem**, é a composição desses vetores referentes a cada um dos elementos da matriz **IEN**.

Uma outra estrutura auxiliar importante é a matriz denominada **neighElem**, que armazena, para cada vértice, o índice de todos os elementos vizinhos a ele. Nesse sentido, considerando a figura 4.3, temos que **neighElem**[1] = [1 8], **neighElem**[2] = [1 2 3], **neighElem**[3] = [11 12], **neighElem**[4] = [1 2 6 7 8] e assim por diante. A matriz **neighElem** é a composição de todos esses vetores.

4.4 Montagem das Matrizes Globais

Uma vez realizada a importação do domínio discretizado e montadas as estruturas auxiliares, faz-se necessária a construção das matrizes globais presentes nas equações de governo 4.1, 4.2, 4.3 e 4.5, a partir da justaposição das matrizes elementares, que são calculadas, para cada elemento, conforme mostrado na seção 3.3.1. A seguir, é apresentada a parte do código que é responsável por essa justaposição.

```
for e in range(0,ne): # Loop percorrendo cada um dos elementos
# Obtencao dos indices globais a partir da matriz de conectividade
    for ilocal in range(0,4):
        iglobal = int(IEN[e,ilocal])
        for jlocal in range(0,4):
            jglobal = int(IEN[e,jlocal])

        # Acoplamento, na matriz global, da matriz local
        # referente ao elemento e
            K[iglobal,jglobal] += kelem[ilocal,jlocal]
            M[iglobal,jglobal] += melem[ilocal,jlocal]

            Gvx[iglobal,jglobal] += gvxele[ilocal,jlocal]
            Gvy[iglobal,jglobal] += gvyele[ilocal,jlocal]

        for jlocal in range(0,3):
            jglobal = int(IEN[e,jlocal])
            Gx[iglobal,jglobal] += gxele[ilocal,jlocal]
            Gy[iglobal,jglobal] += gyele[ilocal,jlocal]

        # As matrizes Gvx e Gvy tem dimensao NVxNV e sao utilizadas na
            equacao da temperatura. As matrizes Gx e Gy possuem dimensao
            NVxNP e sao utilizadas nas equacoes de Navier-Stokes.

Dx = np.transpose(Gx)
Dy = np.transpose(Gy)
```

4.5 Verificação do Código Numérico

4.5.1 Escoamento de Hagen-Poiseuille

O escoamento de Hagen-Poiseuille consiste num escoamento dentro de um canal composto de duas placas paralelas, de modo que ao estar totalmente desenvolvido, o perfil de velocidade torna-se parabólico. É um problema simples de escoamento laminar que é muito utilizado para verificação de códigos numéricos, uma vez que tal escoamento possui solução analítica.

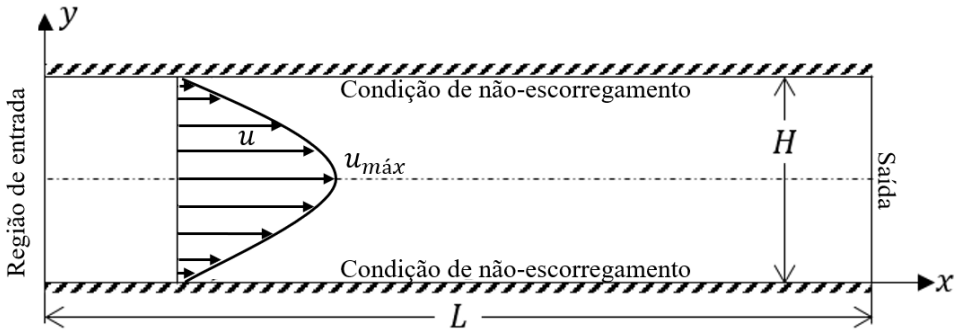


Figura 4.4: Representação do escoamento de Poiseuille.

Será realizada uma simulação do escoamento de Hagen-Poiseuille utilizando o código proposto nesse trabalho. Será considerado um fluido hipotético escoando a 20 °C enquanto as paredes estarão em temperatura constante de 50 °C. O resultado do perfil de velocidade u em função da posição y será comparado com a solução analítica. Além disso será verificado se o perfil da distribuição de temperaturas é parabólico como se espera.

Solução Analítica

A solução analítica para a velocidade do escoamento bidimensional completamente desenvolvido a uma distância y da placa inferior no escoamento de Poiseuille está descrita em Santos[12] e tem a seguinte forma:

$$u(y) = -\frac{1}{2\mu}\frac{\partial p}{\partial x}(yH - y^2) \tag{4.6}$$

O gradiente de pressão é dado pela seguinte expressão:

$$\frac{\partial p}{\partial x} = -12\frac{\mu Q}{H^3} \tag{4.7}$$

Considerando a velocidade de entrada constante de magnitude U, tem-se:

$$Q = UH \tag{4.8}$$

$$\frac{\partial p}{\partial x} = -12\frac{\mu U}{H^2} \tag{4.9}$$

Portanto, substituindo a equação 4.9 na equação 4.6, chega-se a:

$$u(y) = 6U\frac{(yH - y^2)}{H^2} \tag{4.10}$$

A partir da equação 4.10 pode-se inferir que a velocidade máxima ocorre em $y = H/2$ e possui valor de $1{,}5\,U$.

Solução numérica

Para a simulação numérica, foi considerado uma domínio de comprimento total de 3m e distância entre placas de 1m. As condições de contorno podem ser observadas na figura a seguir.

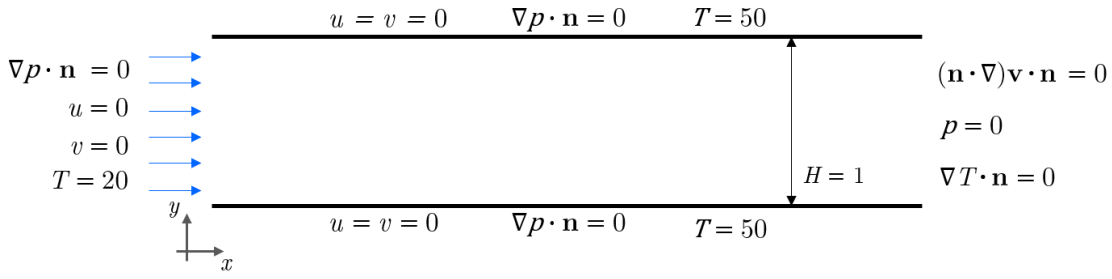


Figura 4.5: Condições de contorno da simualção do escoamento de Poiseuille.

Como critério de parada para a simulação, foi analisada a diferença entre o somatório do campo de velocidades entre uma iteração e outra, e definido que é atingido o regime permanente quando tal resultado é menor que $1{,}00\times10^{-6}$m/s, conforme mostra a equação 4.11.

$$\Delta \mathbf{v} = \sum_{i=1}^{NV} \left| \mathbf{v}_i^n - \mathbf{v}_i^{n-1} \right| < 1,00 \times 10^{-6} \tag{4.11}$$

Os parâmetros utilizados na simulação estão apresentados na tabela 4.1.

Fluido Hipotético			
ρ [kg/m^3]	1,00	α [m^2/s]	0,10
μ [Pa $\cdot$ s]	0,10	U [m/s]	1,00
ν [m^2/s]	0,10	Re	10
c_p [kJ/kg·K]	2,00	Pr	1,00
k [W/m·K]	0,20	Δt [s]	$1,00 \times 10^{-2}$

Tabela 4.1: Parâmetros do escoamento para a simulação de Poiseuille

As simulações foram realizadas utilizando-se seis malhas de diferentes graus de refinamento a fim de ser possível observar o grau de convergência, isto é, o quanto o resultado é influenciado pelo fator de refinamento. Tal análise foi feita comparando os resultados obtidos para a velocidade u nas simulações com os valores calculados analiticamente por meio da equação 4.10. Os resultados numéricos foram obtidos a partir do software *ParaView*, traçando-se uma linha vertical no domínio em $x = 1,5$m, onde o escoamento já se encontra totalmente desenvolvido, e obtendo-se os valores da velocidade u em função da coordenada y.

Os parâmetros das malhas utilizadas estão presentes na tabela 4.2.

Malha	Número de elementos
1	482
2	650
3	922
4	1496
5	2966
6	4544

Tabela 4.2: Malhas utilizadas nas simulações com seus respectivos números de elementos.

A figura 4.6 apresenta um esquema da malha de número 4 a fim de exemplificação.

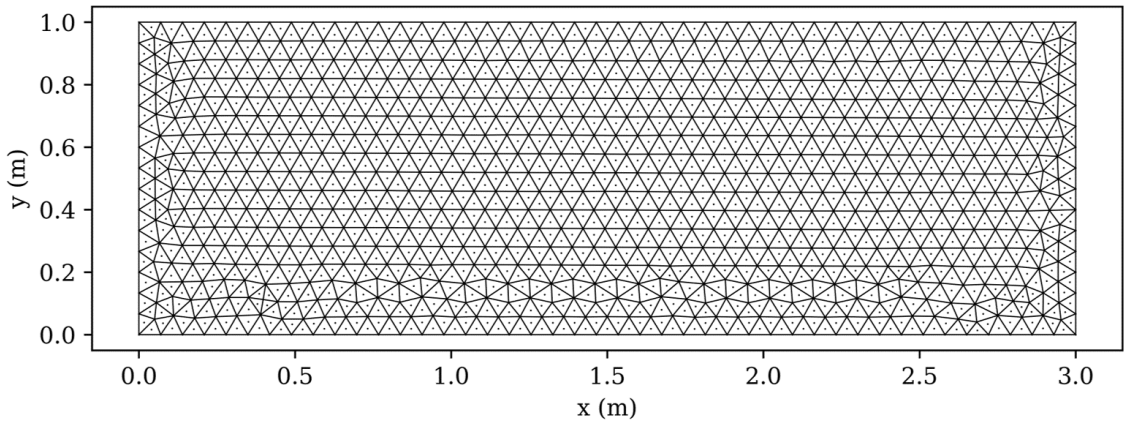


Figura 4.6: Malha utilizada na simulação de Poiseuille.

O gráfico da figura a seguir apresenta o erro relativo máximo entre as solução numéricas e a solução analítica em função do número de elementos das malhas.

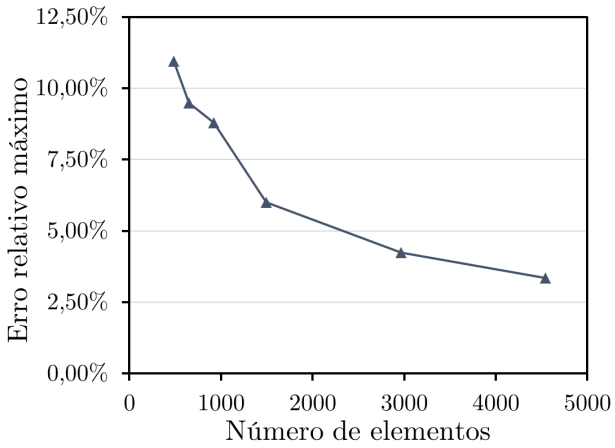
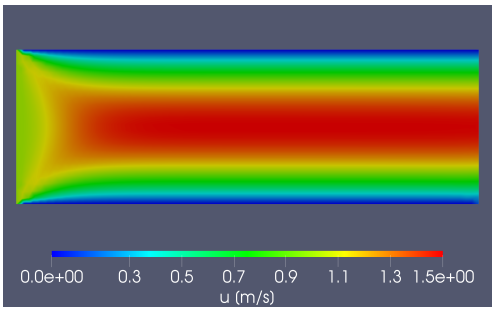


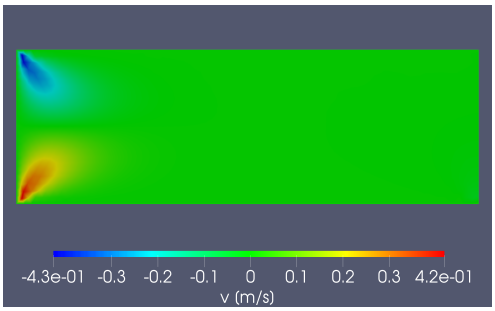
Figura 4.7: Erro relativo máximo obtido entre as soluções numéricas e a solução analítica em função do número de elementos para o escoamento de Poiseuille.

A partir da análise do gráfico, é possível inferir que os resultados convergiram com um erro médio abaixo de 5,0% a partir da malha de 2966 elementos.

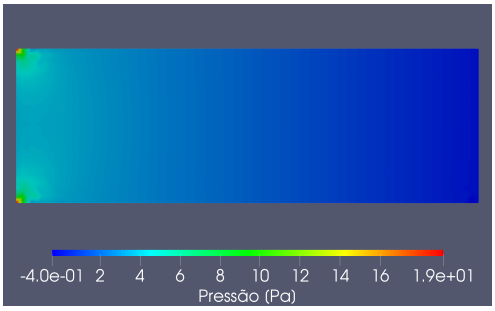
Os resultados da simulação para os campos de velocidades, pressões e temperaturas para a malha de número 5 estão apresentados na figura a seguir.



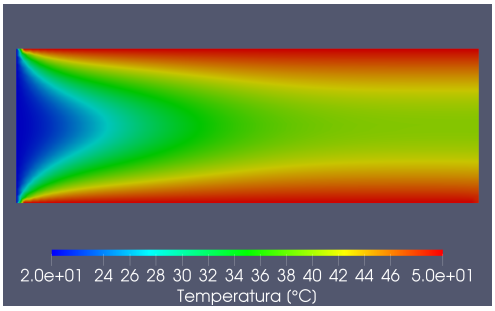
(a) Velocidade na direção x.



(b) Velocidade na direção y.



(c) Campo de pressão.



(d) Campo de temperatura.

Figura 4.8: Resultados da simulação numérica do escoamento de Poiseuille.

Percebe-se que, como esperado, tanto o perfil do campo de velocidade na direção x como o do campo de temperaturas são parabólicos. O gráfico da figura 4.9 apresenta uma sobreposição dos resultados da solução numérica e analítica para a velocidade na direção x em função da coordenada y para a malha de número 5.

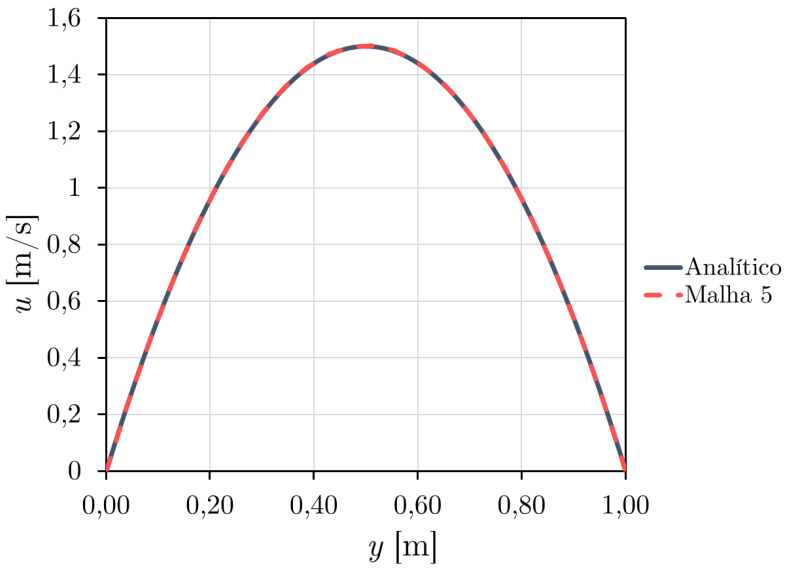


Figura 4.9: Velocidade u em função da coordenada y no escoamento de Poiseuille.

Pela análise do gráfico, é possível observar que o resultado numérico para esse escoamento é bastante semelhante ao resultado analítico.

4.5.2 Escoamento lid-driven

O escoamento conhecido como *lid driven* simula o comportamento de um fluido dentro de uma cavidade com tampa quando a tampa se move com uma velocidade determinada. É um caso também bastante utilizado para a verificação de simulações, pois apesar da solução analítica não ser facilmente encontrada, exemplos de soluções numéricas utilizando os mais variados métodos estão amplamente disponíveis na literatura. O esquema desse escoamento, juntamente com as condições de contorno, está apresentado na figura 4.10.

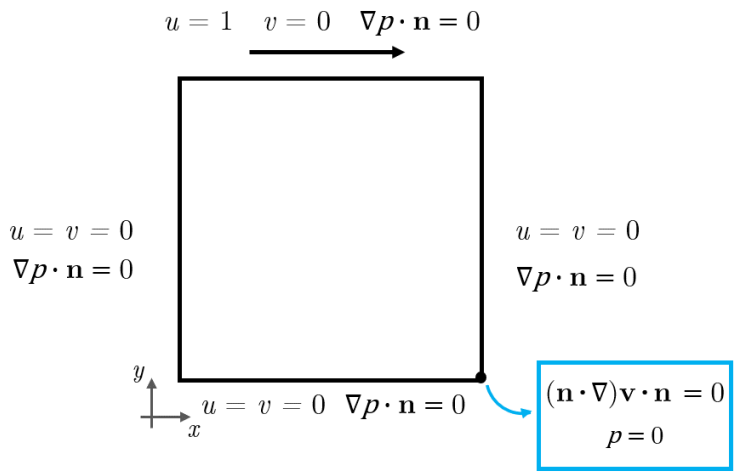
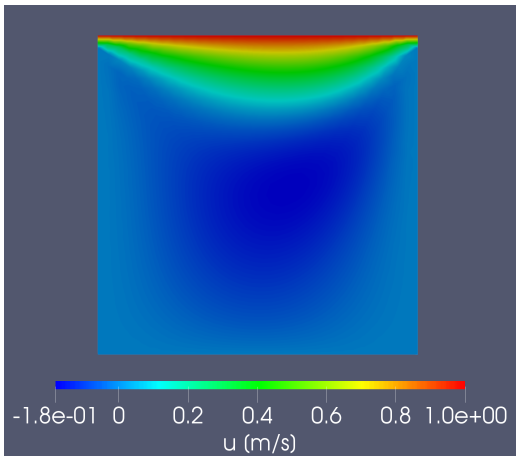


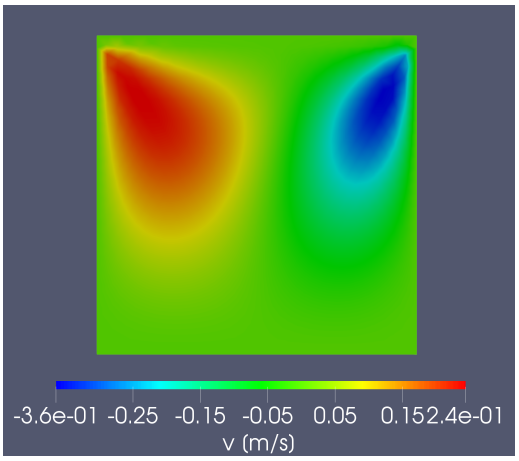
Figura 4.10: Esquema do escoamento *lid-driven*

Foi considerada uma cavidade com dimensões de 1×1m e foi utilizada uma malha com o mesmo grau de refino da malha de número 6 do escoamento de Poiseuille apresentada na seção 4.5.1.

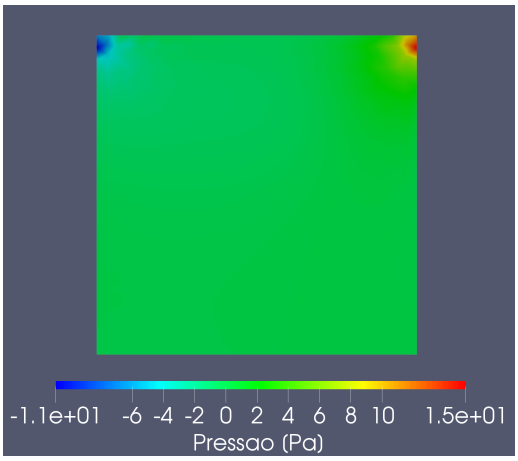
A simulação foi realizada considerando $Re = 100$ e um passo de tempo de $1,00 \times 10^{-3}$s. Os resultados da simulação estão apresentados na figura a seguir.



(a) Velocidade na direção x.



(b) Velocidade na direção y.



(c) Campo de pressões.

Figura 4.11: Resultado da simulação *lid-driven*.

Com o *ParaView*, foi realizado o cálculo da velocidade u em função da coordenada y na posição $x = 0{,}5$ m e comparado com os valores obtidos numericamente por Ghia[13] para o mesmo número de Reynolds. O gráfico da figura 4.12 apresenta os resultados obtidos.

A partir da análise do gráfico, é possível observar que as resultados apresentam comportamentos qualitativamente bastaste semelhantes. Desconsiderando-se o ponto em que u é aproximadamente zero, o erro relativo máximo encontrado foi de 20,08%.

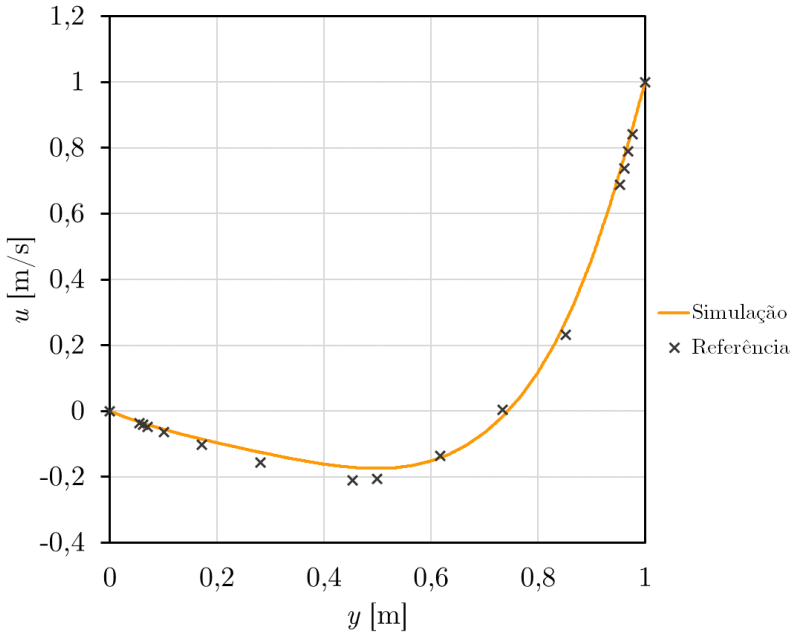


Figura 4.12: Comparação dos resultados obtidos com os valores de referência.

4.6 Verificação do Modelo da Válvula

Como explicado na seção 1.3, o objetivo desse trabalho é investigar a aplicabilidade de válvulas de Tesla em trocadores de calor microfluídicos, primeiramente avaliando diodicidade para diferentes números de Reynolds e seguidamente avaliando a eficiência na transferência de calor desse tipo de geometria em comparação com geometrias mais comumente utilizadas em trocadores.

Para tanto, foram criados dois modelos de válvulas de Tesla, que estão mostrados nas figuras 4.13 e 4.14. Um dos modelos consiste em uma válvula de Tesla única, que foi utilizada para a verificação do modelo numérico, isto é, para testar a adequação da malha e do passo de tempo. O outro modelo consiste em duas válvulas conectadas em série, que foi o de fato utilizado nos estudos desse trabalho, com o objetivo de aumentar os efeitos da diodicidade e utilizando-se os mesmos parâmetros de simulação da válvula única já validada.

Os modelos foram criados utilizando-se o software *SolidWorks* e foram inspirados nas geometrias otimizadas desses dispositivos propostos por Gamboa, Morris e Forster [8].

Figura 4.13: Modelo de válvula de Tesla única.

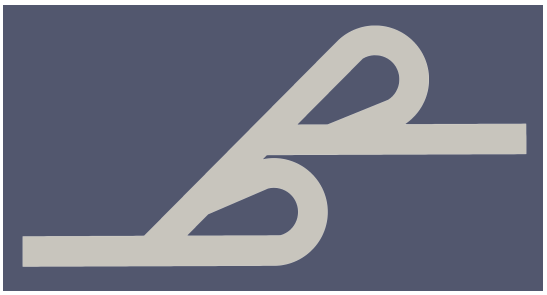


Figura 4.14: Modelo com duas válvulas em série

A altura da entrada do conduíte é de 1 milímetro e os comprimentos de entrada e saída do fluído são de 4 mm. Foi utilizada água a 20°C como fluido de trabalho. As propriedades da água foram obtidas diretamente da biblioteca *CoolProp* do *Python*.

Escolha do Passo de Tempo

O método semi-Lagrangeano é incondicionalmente estável, independentemente do passo de tempo. No entanto, quanto menor o passo de tempo melhor é a aproximação da derivada material da equação de Navier-Stokes, e mais precisa é a solução. Nesse sentido, para a simulação numérica, primeiramente desejou-se encontrar o passo de tempo ideal para a simulação. Para tanto, foi construída uma malha inicial e testado o escoamento para $\mathrm{Re} = 1000$ com diferentes passos de tempo, uma vez que esse é o número de Reynolds mais alto que será simulado. Além disso, nesse primeiro momento foi testado apenas o escoamento no sentido inverso, por ser o mais caótico.

Para números de Reynolds altos e geometrias complexas como a da válvula de Tesla, o regime do escoamento não é permanente, isto é, as variações dos campos de velocidades e de pressão entre um passo de tempo e outro nunca tendem a zero. Portanto, foi analisado o comportamento da diferença relativa das propriedades do fluido entre uma interação e outra utilizando norma 1, isto é:

$$\Delta \mathbf{v}_{rel} = \sum_{i=1}^{NV} \left| \frac{\mathbf{v}_i^n - \mathbf{v}_i^{n-1}}{\mathbf{v}_i^n} \right| \tag{4.12}$$

$$\Delta p_{rel} = \sum_{i=1}^{NP} \left| \frac{p_i^n - p_i^{n-1}}{p_i^n} \right| \tag{4.13}$$

Como dito anteriormente, essas diferenças relativas não tendem a zero, pois o escoamento não chega ao regime permanente. Dessa forma, desejou-se acompanhar o comportamento desses valores com o passar do tempo de modo a averiguar a estabilidade e precisão do resultado.

A malha utilizada nos testes, bem como seus parâmetros e as condições de contorno estão representados na figura 4.15. Ela foi construída de modo a apresentar um maior refinamento nas paredes do escoamento, onde existe um maior gradiente de velocidades. As condições de contorno são as mesmas do escoamento de Poiseuille, ou seja, velocidades nulas nas paredes e prescritas na entrada, gradiente da velocidade nulo na saída, pressão nula na saída e gradientes de pressão nulos na entrada e nas paredes.

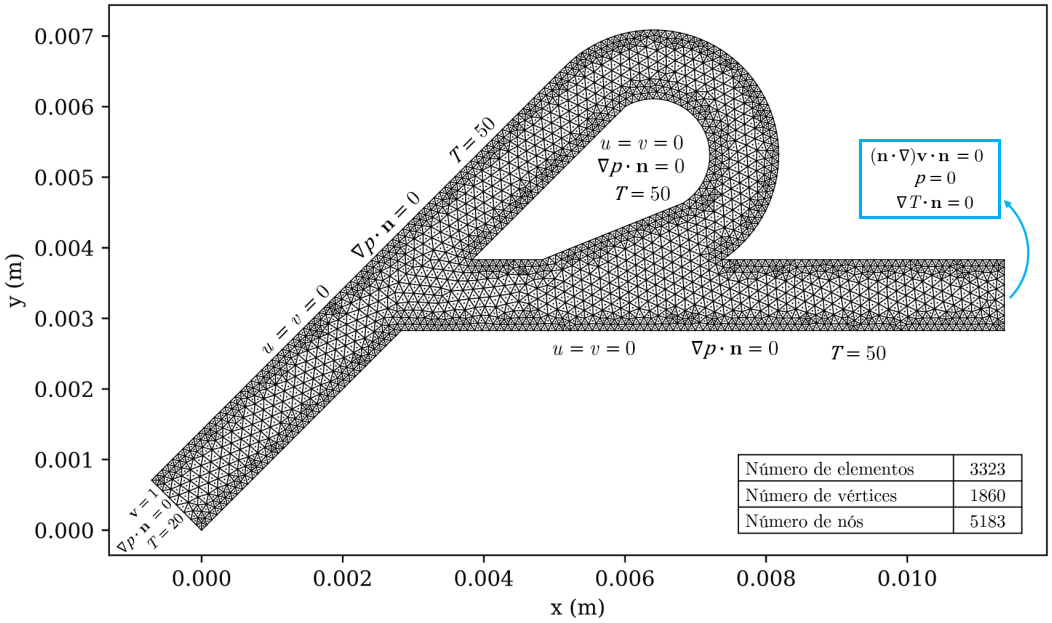


Figura 4.15: Malha para teste da válvula única

As propriedades do fluido e os parâmetros do escoamento estão representados na tabela a seguir:

Fluido	Água (20°C)
ρ [kg/m^3]	998,21
μ [Pa · s]	$1,00 \times 10{-}3$
ν [m^2/s]	$1,00 \times 10^{-6}$
c_p [kJ/kg·K]	4184,05
k [W/m·K]	$5,98 \times 10^{-1}$
α [m^2/s]	$1,43 \times 10^{-7}$
U [m/s]	1,00
Re	1000
Pr	7,01

Tabela 4.3: Propriedades do escoamento para verificação do modelo numérico

Nesse caso, U representa o módulo da velocidade de entrada, cujo vetor tem direção de 45° com relação aos eixos coordenados. A utilização de passos de tempo na ordem de 10^{-3}s ou superiores geraram resultados bastante caóticos. As figuras a seguir demonstram os resultados obtidos para passos de tempo de 5×10^{-4}s e $2,5 \times 10^{-4}$s.

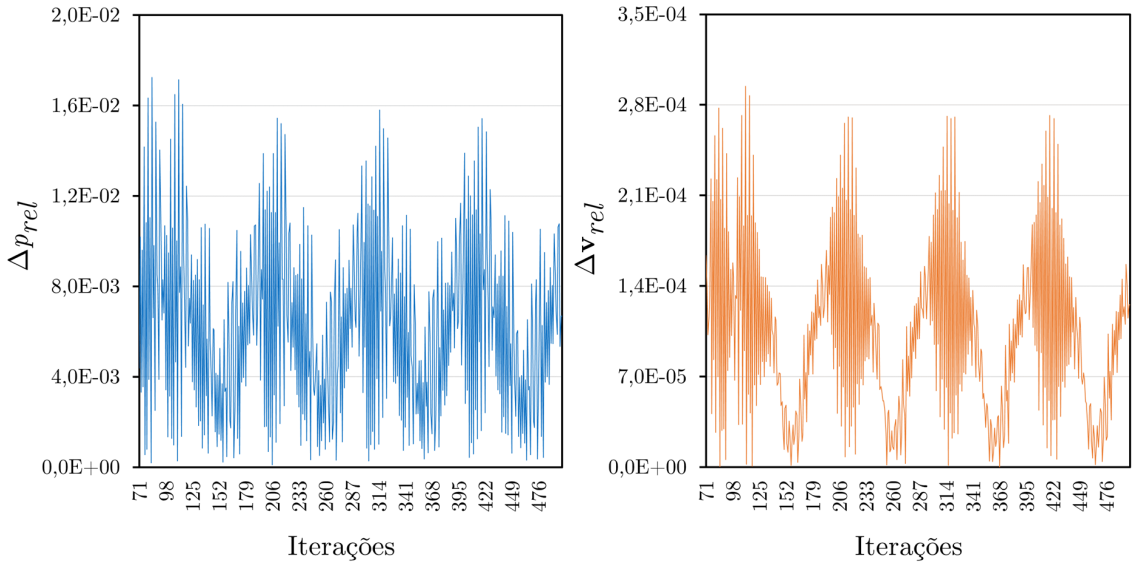


Figura 4.16: Resultados de $\Delta \mathbf{v}_{rel}$ e Δp_{rel} em função do número de iterações para $\Delta t = 5 \times 10^{-4}$s.

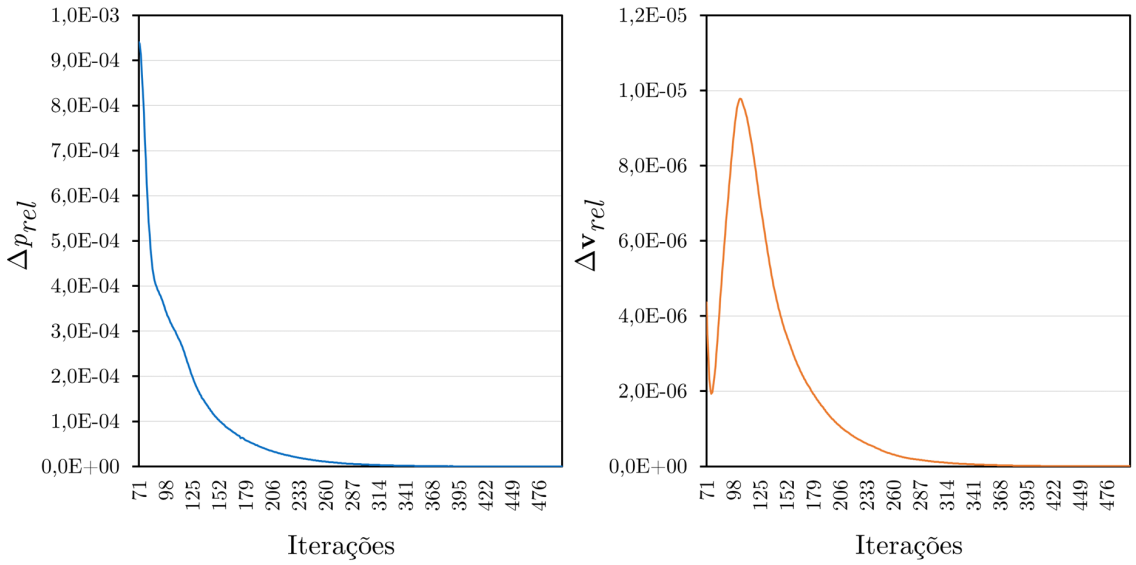


Figura 4.17: Resultados de $\Delta \mathbf{v}_{rel}$ e Δp_{rel} em função do número de iterações para $\Delta t = 2,5 \times 10^{-4}$s.

Observando os gráficos acima, pode-se observar que os resultados para $\Delta t = 2,5 \times 10^{-4}$s são muito mais precisos do que os para $\Delta t = 5 \times 10^{-4}$s, uma vez que as diferenças relativas deste oscilam em uma faixa de variação bem mais elevada do que os daquele. Testes com passos de tempo ainda menores mostraram-se tão precisos quanto o com $\Delta t = 2,5 \times 10^{-4}$s. Além disso, foi observado que os valores das diferenças relativas passam a variar em um grau extremamente pequeno a partir de aproximadamente 480 iterações, quando os valores de Δp_{rel} se estabilizam abaixo de $1,0 \times 10^{-7}$. Nesse sentido, em todas as simulações realizadas a partir daqui, foi utilizado um passo de tempo de $2,5 \times 10^{-4}$s e foram consideradas 480 iterações para a convergência do escoamento.

Teste de Convergência de Malha

Uma vez escolhidos o passo de tempo e o número de iterações, o próximo passo foi escolher o grau de refinamento da malha. Para tanto, foram geradas diferentes malhas alterando-se o grau de refinamento dos elementos. Cada uma das malhas criadas foi testada e os resultados foram comparados no que diz respeito à perda de carga do escoamento, isto é, a diferença de pressão entre a entrada e saída da válvula, uma vez que é a perda de carga que será utilizada para o cálculo da diodicidade, objeto de estudo desse trabalho.

Os canais de entrada e saída da válvula possuem 4 mm de comprimento, e as pressões foram medidas em linhas localizadas a 1 mm antes da entrada da válvula e 1 mm depois da saída da mesma, conforme mostra a figura 4.18, a partir da média dos valores da pressão nessas regiões.

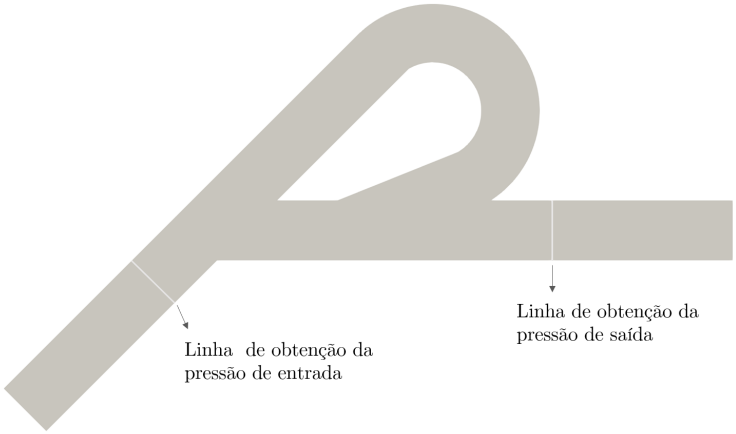


Figura 4.18: Linhas para obtenção da pressão

Foram utilizadas cinco malhas com graus de refinamento diferentes. A tabela 4.4 classifica as malhas conforme o número de elementos e a figura 4.19 mostra os resultados obtidos para a perda de carga nas simulações das diferentes malhas.

Malha	Número de Elementos
1	3323
2	3773
3	4400
4	5111
5	6072

Tabela 4.4: Malhas utilizadas nas simulações para validação do modelo numérico

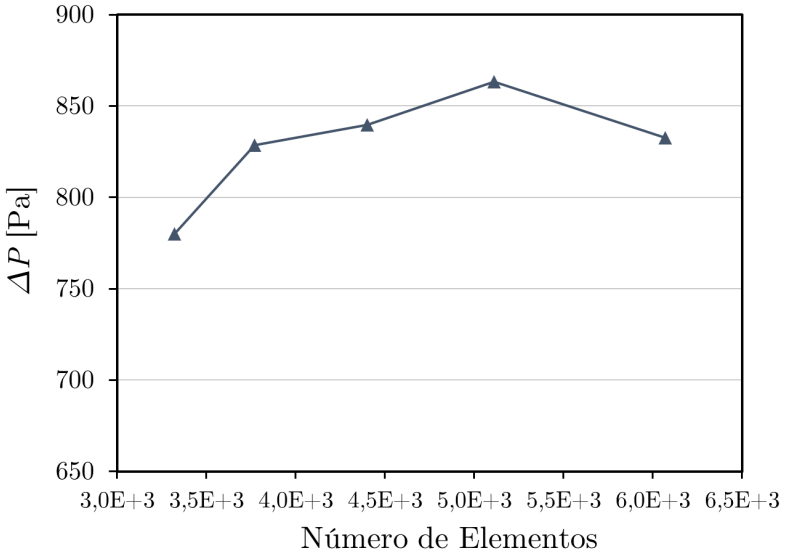
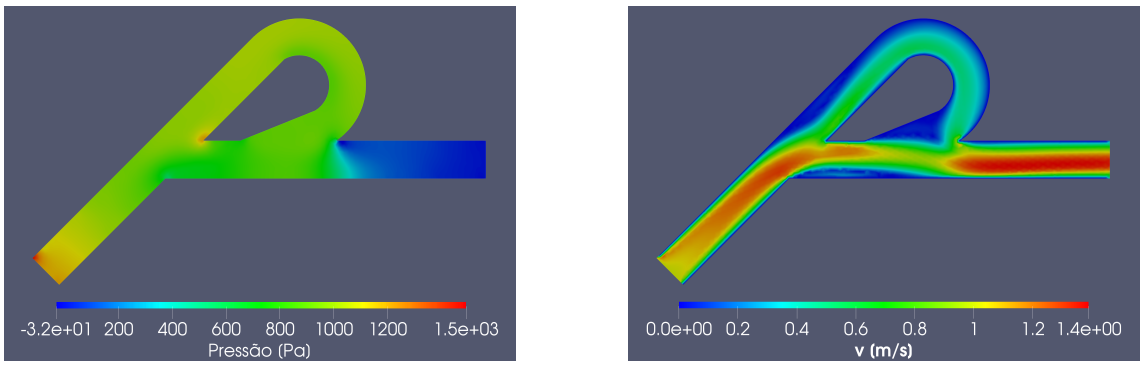


Figura 4.19: Perda de carga em função do número de elementos da malha

Observando o gráfico da figura acima, é possível perceber que o resultado da malha de número 1 diverge bastante das outras e que a partir da malha de número 2 eles começam a convergir. O erro relativo da malha de número 2 com relação à malha de número 4 é de 4% e com a relação à malha de número 5 é de 0,48%. O refinamento da malha está associado a uma propriedade do *Gmsh* chamada *Global Mesh Size Factor* (GMSF). Nesse sentido, uma vez que a malha de número 4 se mostrou adequada, foi utilizado o mesmo GMSF dessa malha no modelo das válvulas de Tesla em série, de modo a ter-se um mesmo número de elementos por unidade de área.



(a) Campo de pressões.

(b) Campo do módulo da velocidade.

Figura 4.20: Resultados da simulação para a malha de 3773 elementos, $\Delta t = 2{,}5{\times}10^{-4}$, $Re = 1000$.

A figura 4.20 apresenta os resultados para os campos de pressões e velocidades obtidos na simulação da malha de número 4.

Uma vez escolhidos o passo de tempo, número de iterações e grau de refinamento da malha, pôde-se validar o modelo e dar-se início às simulações utilizando a geometria com duas válvulas em série.

Capítulo 5

Resultados e Discussões

5.1 Diodicidade em Função do Número de Reynolds

Conforme apresentado na seção 1.1, a diodicidade é a grandeza capaz de mensurar a eficácia das válvulas de Tesla. Ela é definida como a razão entre a perda do escoamento ao fluir no sentido inverso e a perda do mesmo ao fluir no sentido direto para uma mesma vazão de fluido.

$$D_i = \left(\frac{\Delta P_i}{\Delta P_d} \right) \tag{5.1}$$

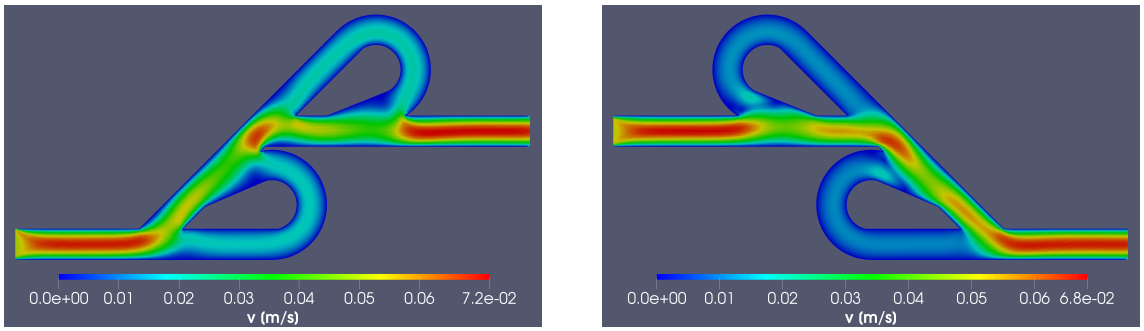
Para a análise do comportamento da diodicidade em função do número de Reynolds, foi utilizada a geometria com duas válvulas em série como apresentado na seção 4.6. Assim como no modelo utilizado na verificação do código numérico, foi considerada uma altura do canal de entrada de 1mm e os comprimentos de entrada e saída são de 4mm. Para a geração da malha foi utilizado o mesmo *Global Mesh Size Factor* utilizado na geometria com a válvula única já validada, isto é, o mesmo número de elementos por unidade de área.

Os parâmetros das malhas utilizadas nas simulações estão apresentados na tabela 5.1. É válido destacar que, apesar dos domínios utilizados nos casos de escoamento no sentido inverso e direto serem idênticos e apenas espelhos um do outro, existe uma pequena diferença entre o número de elementos das malhas devido ao próprio algoritmo de geração do *Gmsh*.

Sentido Inverso		Sentido Direto	
Número de elementos	6163	Número de elementos	6155
Numero de vértices	3417	Numero de vértices	3413
Número de nós	9580	Número de nós	9568

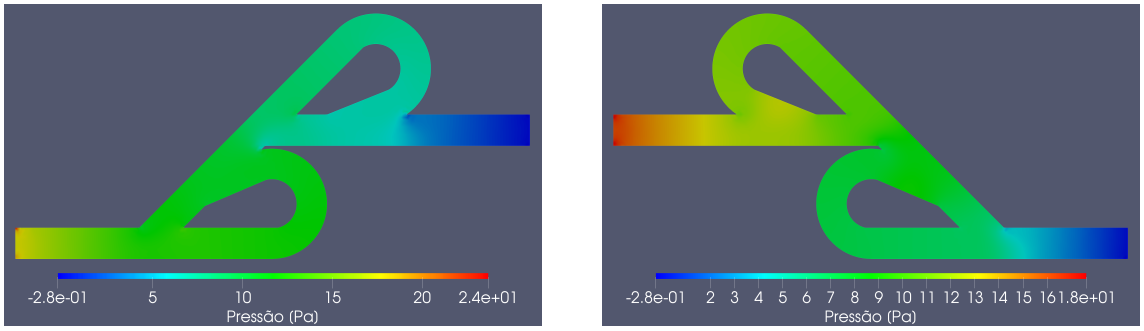
Tabela 5.1: Parâmetros das malhas utilizadas nas simulações.

Assim como na verificação, foi considerado água a 20ºC como fluido de trabalho. As propriedades estão dispostas na tabela 4.3 da seção 4.6. A diferença aqui é que a velocidade U de entrada do fluido foi variada de modo a serem atingidos diferentes números de Reynolds. Foram testadas cinco faixas de Reynolds: 50, 100, 200, 500 e 1000. O passo de tempo utilizado foi de $2,5 \times 10^{-4}$ e foram consideradas 480 iterações. Os resultados das simulações estão apresentados nas figuras 5.1, 5.2, 5.3, 5.4, 5.5, 5.6, 5.7, 5.8, 5.9 e 5.10. O escoamento em todos os casos é da esquerda para a direita.



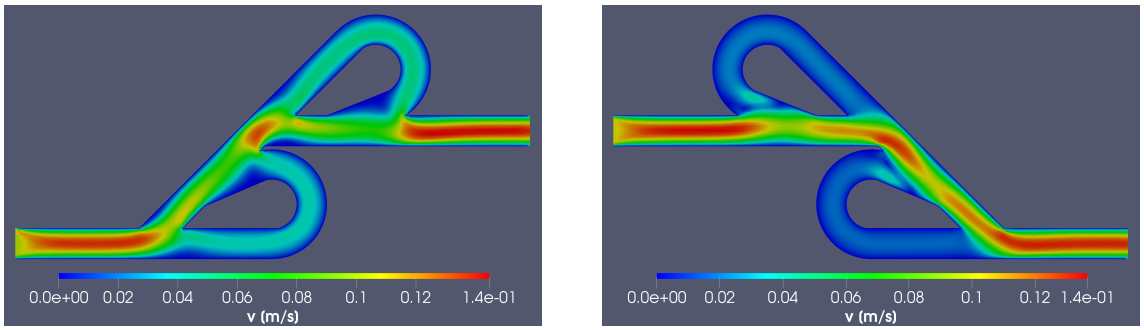
(a) Sentido inverso. (b) Sentido direto.

Figura 5.1: Resultados das simulações do módulo da velocidade para $Re = 50$.



(a) Fluxo inverso. (b) Fluxo direto.

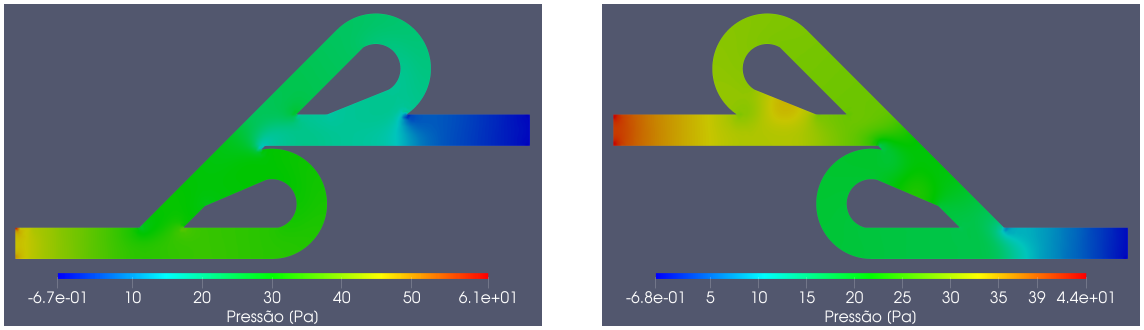
Figura 5.2: Resultados das simulações do campo de pressões para $Re = 50$.



(a) FluSentido inverso.

(b) Sentido direto.

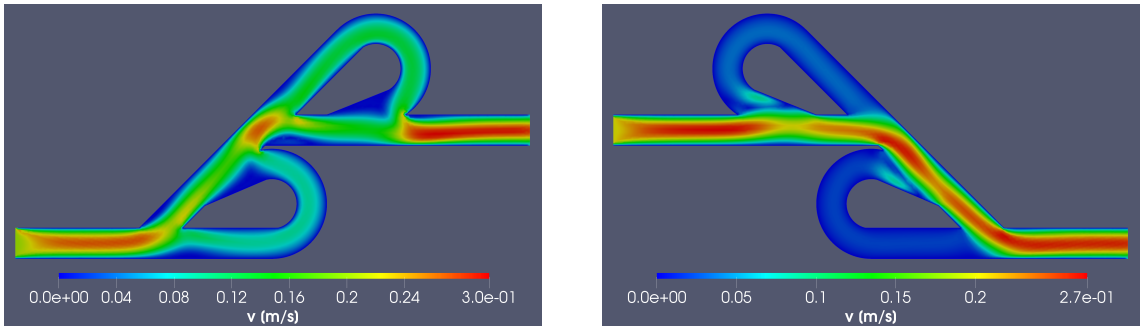
Figura 5.3: Resultados das simulações do módulo da velocidade para $Re = 100$.



(a) Sentido inverso.

(b) Sentido direto.

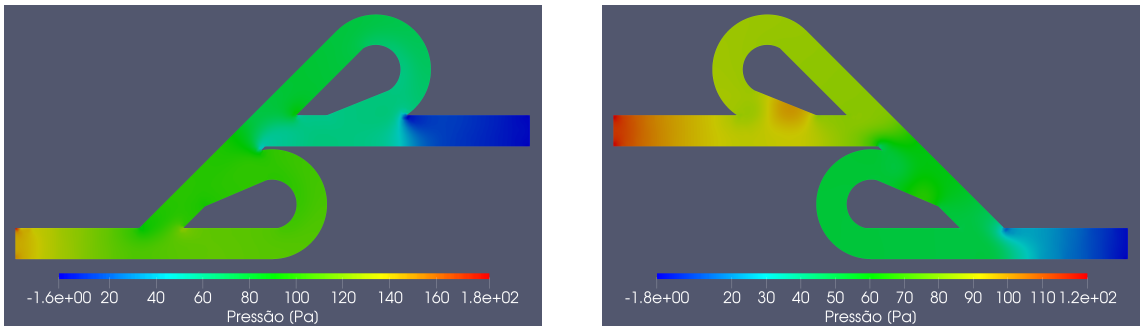
Figura 5.4: Resultados das simulações do campo de pressões para $Re = 100$.



(a) Sentido inverso.

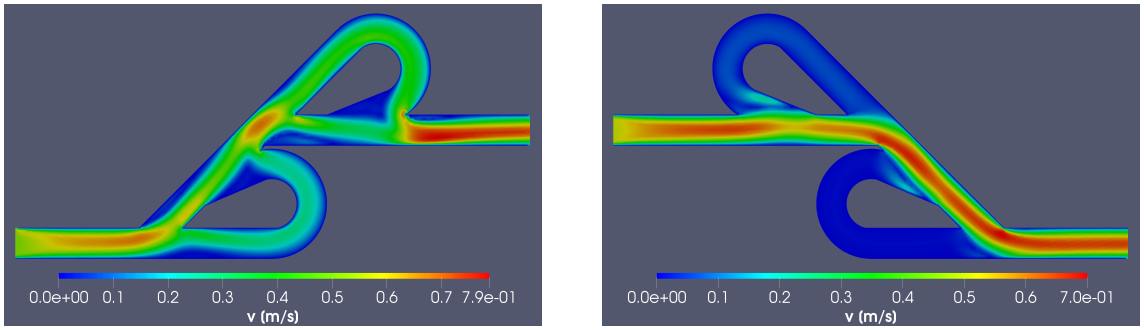
(b) Sentido direto.

Figura 5.5: Resultados das simulações do módulo da velocidade para $Re = 200$.



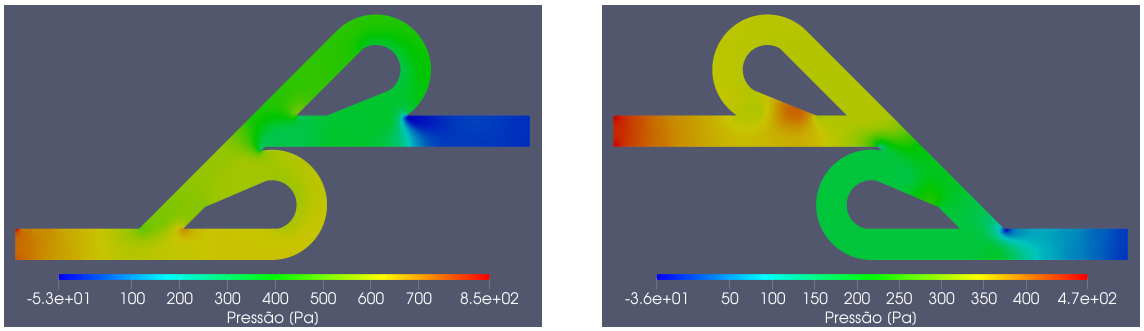
(a) Sentido inverso. (b) Sentido direto.

Figura 5.6: Resultados das simulações do campo de pressões para $Re = 200$.



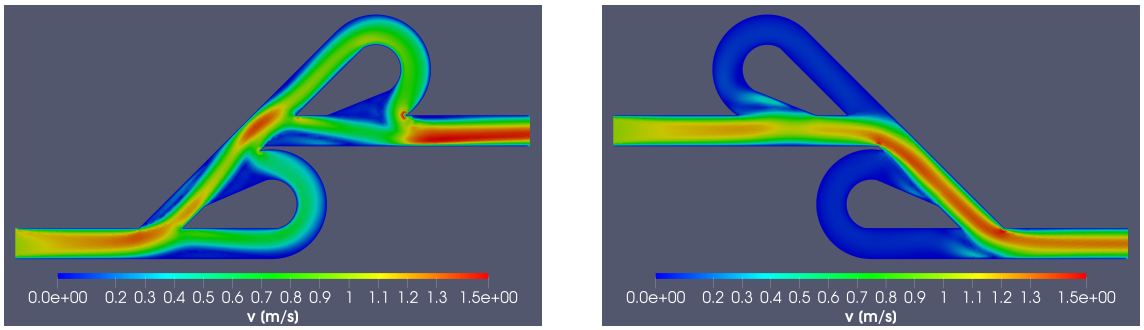
(a) Sentido inverso. (b) Sentido direto.

Figura 5.7: Resultados das simulações do módulo da velocidade para $Re = 500$.



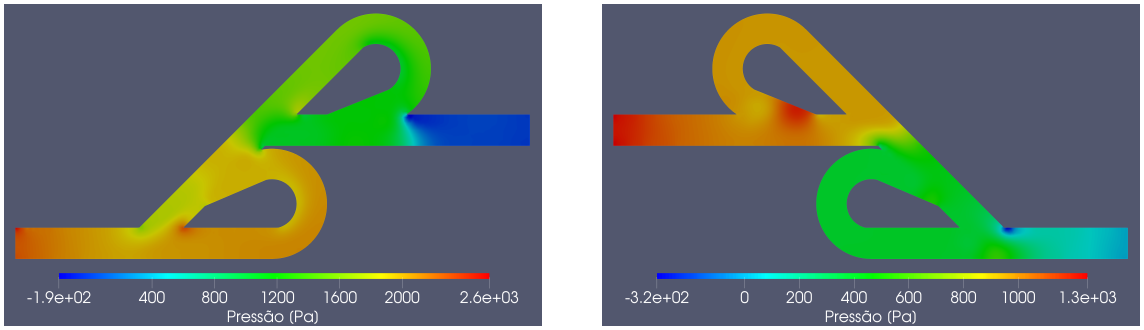
(a) Sentido inverso. (b) Sentido direto.

Figura 5.8: Resultados das simulações do campo de pressões para $Re = 500$.



(a) Sentido inverso. (b) Sentido direto.

Figura 5.9: Resultados das simulações do módulo da velocidade para $Re = 1000$.



(a) Sentido inverso. (b) Sentido direto.

Figura 5.10: Resultados das simulações do campo de pressões para $Re = 1000$.

A partir da observação das imagens, é possível perceber que, para o fluxo inverso, quanto maior o número de Reynolds, maior a porção de fluido que passa pelos canais laterais das válvulas em comparação com a que escoa pela região central. Já para o fluxo direto, independentemente do número de Reynolds, o fluido tende a seguir o caminho central do dispositivo.

O gráfico da imagem 5.11 representa a diodicidade em função do número de Reynolds. Os valores da diodicidade foram obtidos a partir da obtenção da pressão à 1mm antes da entrada da válvula e 1mm após a saída da mesma, utilizando-se o software *ParaView* da mesma forma como mostrado na seção 4.6.

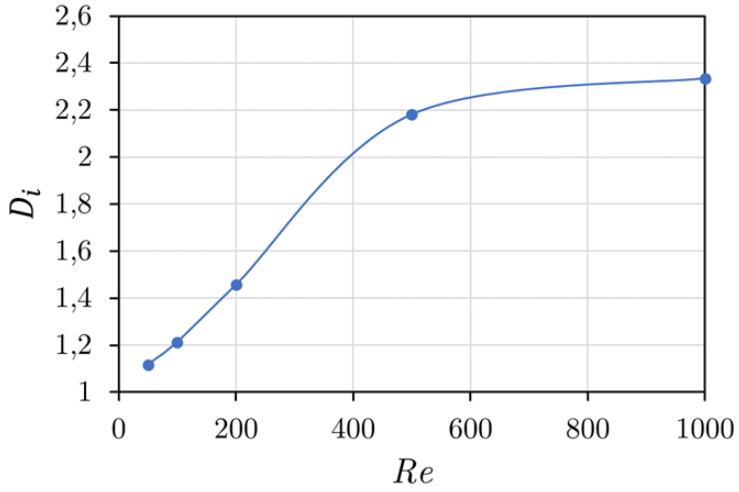


Figura 5.11: Resultado diodicidade em função do número de Reynolds.

A partir da análise do gráfico é possível observar que, como esperado, a diodicidade cresce em função do número de Reynolds, atingindo o valor máximo de 2,33 para $Re = 100$. Porém a taxa de crescimento da diodicidade não é constante, ela parece aumentar até a faixa de $Re = 200$ e depois passa a diminuir até se tornar bem reduzida para números de Reynolds maiores que 500.

Tal comportamento foi também observado por Gamboa, Morris e Forster [8], apesar dos valores absolutos da diodicidade obtidos pelos autores terem sido menores, devido ao fato de terem utilizado um modelo de válvula única.

5.2 Transferência de Calor

Para verificar se a válvula de Tesla se mostra como uma alternativa adequada em relação à transferência de calor, foi construída uma outra geometria para ser utilizada como referência para comparação. Tal geometria foi construída com um perfil de zigue-zague, que é um formato mais comumente encontrado em trocadores de calor microfluídicos. Além disso, buscou-se obter aproximadamente a mesma área e o mesmo perímetro que a do dispositivo de Tesla.

A figura a seguir mostra a geometria de referência já com a malha constituída (foram omitidos os centroides) sobreposta à geometria de Tesla a título de comparação, bem como os parâmetros da malha gerada.

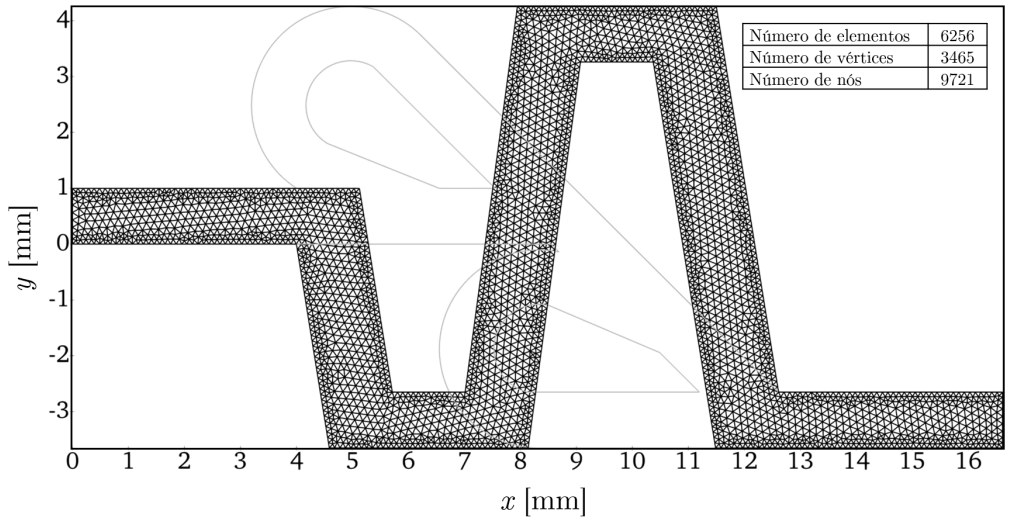


Figura 5.12: Malha do trocador em zigue-zague sobreposta à válvula de Tesla.

A tabela a seguir representa a comparação entre os parâmetros geométricos dos dois domínios.

Válvula de Tesla		Conduite de referência	
Área total [mm^2]	35,31	Área total [mm^2]	36,26
Perímetro [mm]	65,37	Perímetro [mm]	65,29

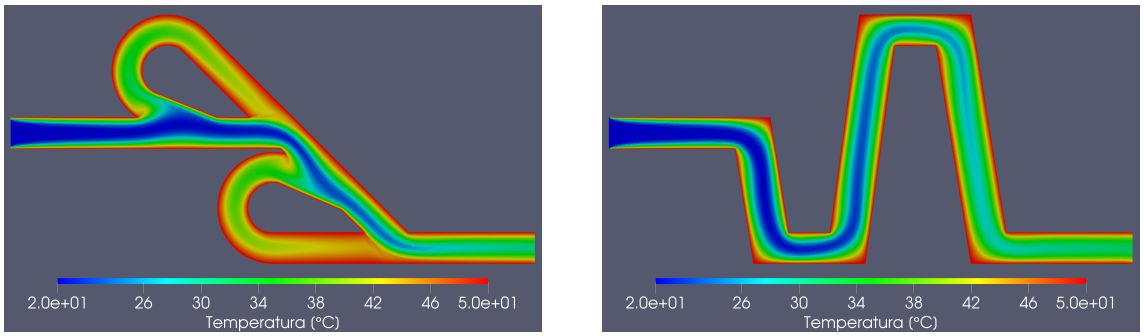
Tabela 5.2: Parâmetros dos modelos utilizadas nas simulações.

Foram realizadas simulações para a obtenção do campo de temperaturas e posterior análise da eficiência comparando a temperatura de saída do fluido para as duas geometrias. Foi considerada água escoando a 20 °C e as paredes à temperatura constante de 50 °C. As simulações foram realizadas com $Re = 50$ e foi considerado o escoamento no sentido direto para o conduíte de Tesla, que é o sentido desejado em um trocador de calor. Como critério de parada para a simulação, foi analisada a diferença entre o somatório do campo de temperaturas entre uma iteração e outra, e definido que é atingido o regime permanente quando tal resultado é menor que 0,2°C, conforme mostra a equação 5.2.

$$\Delta T = \sum_{i=1}^{NT} \left| T_i^n - T_i^{n-1} \right| < 0,2 \tag{5.2}$$

Além disso, uma vez que o atingimento do regime permanente de temperaturas exige um alto número de iterações e consequentemente um alto tempo de processa-

mento computacional, os campos de velocidades e pressões só foram calculados até a 480º iteração e considerados constantes a partir de então, uma vez que, após esse patamar, a variação dessas propriedades é mínima. O passo de tempo utilizado foi de $2,5 \times 10^{-4}$s. A figura a seguir apresenta os resultados obtidos.



(a) Válvula de Tesla.

(b) Modelo de referência.

Figura 5.13: Resultados obtidos dos campos de temperaturas para $Re = 50$.

Foram obtidos os valores das temperaturas de saída a partir do cálculo do valor médio dessa propriedade a 0,5mm do final dos conduítes, utilizando-se o software *ParaView*. A eficiência de transferência de calor foi calculada da seguinte forma:

$$\epsilon = \frac{T_i - T_o}{T_i - T_w} \tag{5.3}$$

Na equação, T_i representa a temperatura de entrada, T_o representa a temperatura de saída e T_w representa a temperatura da parede. A tabela 5.3 apresenta os resultados obtidos.

Conduíte	$T_{saída}$ [°C]	Eficiência
Tesla	37,72	59,07%
Referência	37,41	58,03%

Tabela 5.3: Resultados para a temperatura de saída e eficiência

A partir dos resultados apresentados, pode-se observar que a a utilização de válvulas de Tesla, em comparação com um modelo comum de trocador, não apresenta desvantagens no que diz respeito à transferência de calor, mas sim uma ligeira vantagem.

Capítulo 6

Conclusões

Nesse trabalho foi desenvolvido um programa em *Python* para a solução de escoamentos utilizando apenas softwares abertos a partir do qual foi feita a análise da viabilidade da implementação de válvulas de Tesla em trocadores de calor. Tais dispositivos, que são uma espécie de válvula de retenção sem partes móveis, fornecem uma maior resistência à passagem do fluido em uma direção (sentido inverso) do que em outra (sentido direto). A razão entre a perda de carga no sentido inverso e a perda no sentido direto caracteriza a grandeza denominada diodicidade. As válvulas de Tesla têm sido relativamente pouco exploradas pela literatura e majoritariamente utilizadas em aplicações de bombeamento.

A motivação para o estudo foi o recente anúncio da empresa Xiaomi de que estaria utilizando esse tipo de válvula nos trocadores de calor de seus smartphones mais avançados de modo a aumentar-se a eficiência de transferência de calor, uma vez que a menor perda de carga em um sentido de escoamento garantiria o correto escoamento de fluido na direção desejada, evitando-se refluxos e consequentemente perda de eficiência.

Para a realização do estudo, foram implementadas as equações de Navier-Stokes e de conservação de energia em um programa iterativo em *Python* utilizando-se o método de elementos finitos. Tal método foi implementado seguindo as formulações presentes em Fish[4], Lewis[5] e Anjos[6].

Foi realizada a verificação do código realizando-se simulações de escoamentos conhecidos, como o escoamento de Poiseuille e o escoamento da cavidade (*lid-driven*), considerando-se um fluido hipotético e comparados os resultados com referências

disponíveis na literatura. Em seguida, foi construído um modelo geométrico de válvula de Tesla única e foram testados escoamentos para $Re = 1000$, no sentido inverso do fluxo, considerando diferentes passos de tempo e diferentes graus de refinamento de malha de modo a ser obtida uma configuração adequada para o modelo estudado.

Em seguida, foram realizadas as simulações dos escoamentos nos sentidos direto e inverso e obtidos os valores da diodicidade para diferentes números de Reynolds considerando duas válvulas em série. Foram testados escoamentos com números de Reynolds variando de 50 a 1000. A partir da análise dos resultados foi observado que, quanto maior o número de Reynolds, maior a diodicidade. Além disso, a taxa de variação da diodicidade se mostrou crescente para Reynolds variando de 50 a 200 e decrescente para Reynolds entre 200 e 1000.

Por fim, foi feita a análise da transferência de calor, comparando os resultados obtidos utilizando a geometria de Tesla com os resultados obtidos utilizando outra geometria mais comumente encontrada em trocadores de calor, para Re=50. Nas simulações, foi considerado o fluido adentrando o escoamento a 20°C e as paredes do domínio a temperatura constante de 50°C. A partir da análise dos resultados, não foram observadas diferenças significativas na eficiência da transferência de calor dos dois conduítes. Para trabalhos futuros, espera-se que sejam feitos mais testes para o cálculo do campo de temperaturas para diferentes números de Reynolds e testes para a transferência de calor da válvula no sentido inverso.

Referências Bibliográficas

[1] TESLA, N., "Valvular conduit", *U.S. Patent No. 1,329,559*, 1920.

[2] "Xiaomi Introduces Loop LiquidCool Technology", Beijing, China, 5 Nov. 2021. Disponível em: https://www.mi.com/global/discover/article?id=2571. Acesso em: 18 set. 2022.

[3] CLOUGH, R. W., "The Finite Element Method in Plane Stress Analysis". 1960.

[4] FISH, J., BELYTSCHKO, T., *A First Course in Finite Elements*. 1st ed. John Wiley & Sons, Inc., 2007.

[5] LEWIS, R. W., NITHIARASU, P., SEETHARAMU, K. N., *Fundamentals of the Finite Element Method for Heat and Fluid Flow*. 1st ed. John Wiley & Sons, Inc., 2004.

[6] ANJOS, G. R., "Notas de aula: Projeto Final: Dinâmica dos Fluidos Computacional - Elementos Finitos", 2020.

[7] FORSTER, F. K. *et al.* "Design, Fabrication and Testing of Fixed-Valve Micro-Pumps", Proc. ASME Fluid Eng. Div., 234, pp. 39–44, 1995.

[8] GAMBOA, A. R., MORRIS, C. J., FORSTER, F. K., "Improvements in fixed-valve micropump performance through shape optimization of valves", ASME Journal of Fluids Engineering, 127 (2), pp. 339–346., 2005.

[9] PAUDEL, B., JAMAL, T., THOMPSON, S., et al., "Thermal Effects on Micro-Sized Tesla Valves". In: *American Society of Mechanical Engineers, Fluids Engineering Division (Publication) FEDSM*, v. 2, 08 2014.

[10] PRITCHARD, P. J., *Fox and McDonald's Introduction to Fluid Mechanics*. 8th ed. John Wiley & Sons, Inc., 2011.

[11] ANJOS, G. R., "Solução do Campo Hidrodinâmico em Células Eletroquímicas Pelo Método de Elementos Finitos", Dissertação (Mestrado), Universidade Federal do Rio de Janeiro, 2007.

[12] SANTOS, F. O., "Simulação numérica de escoamentos de fluidos utilizando diferenças finitas generalizadas", Dissertação (Mestrado), Universidade de São Paulo, 2005.

[13] GHIA, U., GHIA, K. N., SHIN, C. T., "High-Re Solutions for Incompressible Flow Using the Navier-Stokes Equations and a Multigrid Method", Journal of Computational Physics, 48, 387-411, 1982.

[14] NGUYEN, N.-T., TRUONG, T.-Q., "Simulation and Optimization of Tesla Valves". 2003.

[15] THOMPSON, S. M., PAUDEL, B. J., JAMAL, T., et al., "Numerical Investigation of Multistaged Tesla Valves", *Journal of Fluids Engineering-transactions of The Asme*, v. 136, pp. 081102, 2014.

[16] NGUYEN, Q.-M., ABOUEZZI, J., RISTROPH, L., "Early turbulence and pulsatile flows enhance diodicity of Tesla's macrofluidic valve", *Nature Communications*, v. 12, 2021.

[17] ANJOS, G. R., "COMPUTAÇÃO CIENTÍFICA PARA ENGENHEIROS", Universidade Federal do Rio de Janeiro, 2020.

[18] Silva, J. A. "Método de Elementos Finitos em Python com a abordagem lagrangiana euleriana para as oscilações de um cilindro em um escoamento transversal", TCC (Graduação) – Curso de Engenharia Mecânica, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2020.

[19] Florentino, Y. S. "Estudo de escoamentos para arrefecimento de eletrônicos através de Método de Elementos Finitos", TCC (Graduação) – Curso de Engenharia Mecânica, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2022.

[20] Cunha, L. H. C. "Formulação Corrente-Vorticidade em Problemas de Transferência de Calor Conjugado usando MEF", TCC (Graduação) – Curso

de Engenharia Mecânica, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2019.

Apêndice A

Códigos Fonte

A.1 Função para implementação do método semi-Lagrangeano

```
## ===================================================================== ##
#  this is file SL.py, created at 19-Feb-2021                           #
#  maintained by Gustavo Rabello dos Anjos                              #
#  e-mail: gustavo.rabello@gmail.com                                    #
## ===================================================================== ##

# Classe SL para navierStokes2dFEM, navierStokes2dFEMGauss and 3D version
# compute -> gera matriz de conveccao conv(numVerts,numVerts)
# getDepartElem -> define a trajetoria para tras e inicia busca
# jumpToElem -> busca de trajetoria
# computeIntercept -> se ponto cair fora do dominio, interpola no contorno
# testElement -> verifica se eh o elemento que contem o ponto
# setCentroid -> interpola valores de numVerts nos centroids
# setBC -> ajuste de c.c. de velocidade em vxd,vyd

import numpy as np
from scipy.sparse import lil_matrix,csr_matrix

class Linear:
```

```
def __init__(_self,_IEN,_X,_Y,_neighborElem,_oface,_velU,_velV):

 _self.X = _X
 _self.Y = _Y
 _self.IEN = _IEN
 _self.numNodes = len(_X)
 _self.numElems = len(_IEN)
 _self.numVerts = _self.numNodes-_self.numElems
 _self.neighborElem = _neighborElem
 _self.oface = _oface
 _self.velU = _velU
 _self.velV = _velV

 _self.l1 = 0.0
 _self.l2 = 0.0
 _self.l3 = 0.0

 _self.conv = lil_matrix((_self.numVerts,_self.numVerts),
     dtype='float32')

def compute(_self,_dt):
 _self.getDepartElem(_dt)

def getDepartElem(_self,_dt):
 for i in range(0,_self.numVerts):
  mele = _self.neighborElem[i][0]
  xp = _self.X[i] - _self.velU[i]*_dt
  yp = _self.Y[i] - _self.velV[i]*_dt

  _self.jumpToElem(mele,i,xp,yp)

def jumpToElem(_self,_destElem,_iiVert,_R2X,_R2Y):
 v1 = _self.IEN[_destElem][0]
 v2 = _self.IEN[_destElem][1]
```

```
 v3 = _self.IEN[_destElem][2]

 if _self.testElement(_destElem,_R2X,_R2Y):
  _self.conv[_iiVert,v1] = _self.l1
  _self.conv[_iiVert,v2] = _self.l2
  _self.conv[_iiVert,v3] = _self.l3
 else:
  if (_self.l1<=_self.l2) and (_self.l1<=_self.l3):
   vjump=0
   ib1=v2
   ib2=v3
  if (_self.l2<=_self.l1) and (_self.l2<=_self.l3):
   vjump=1
   ib1=v1
   ib2=v3
  if (_self.l3<=_self.l1) and (_self.l3<=_self.l2):
   vjump=2
   ib1=v1
   ib2=v2

  if _self.oface[_destElem,vjump] != -1:
   _self.jumpToElem(_self.oface[_destElem,vjump],_iiVert,_R2X,_R2Y)
  else:
   Bl1,Bl2 = _self.computeIntercept(_iiVert,_R2X,_R2Y,ib1,ib2)
   _self.conv[_iiVert,ib1] = Bl1
   _self.conv[_iiVert,ib2] = Bl2

def computeIntercept(_self,_i,_R2X,_R2Y,_ib1,_ib2):
 R1X = _self.X[_i]
 R1Y = _self.Y[_i]

 B1X = _self.X[_ib1]
 B2X = _self.X[_ib2]
 B1Y = _self.Y[_ib1]
```

```
 B2Y = _self.Y[_ib2]

 a1 = B1X-B2X; b1 = R1X-_R2X; c1 = R1X-B2X
 a2 = B1Y-B2Y; b2 = R1Y-_R2Y; c2 = R1Y-B2Y

 det  = (a1*b2) - (a2*b1)
 detx = (c1*b2) - (c2*b1)

 x1 = detx/det

 # return C++ equivalent Bl1 and Bl2 (see femSIM2d)
 return x1,1.0-x1

def testElement(_self,_mele,_xp,_yp):
 v1 = _self.IEN[_mele][0]
 v2 = _self.IEN[_mele][1]
 v3 = _self.IEN[_mele][2]
 EPSlocal = 1e-04

 A = (1.0/2.0) * ( _self.X[v2]*_self.Y[v3] +
                   _self.X[v1]*_self.Y[v2] +
                   _self.Y[v1]*_self.X[v3] -
                   _self.X[v1]*_self.Y[v3] -
                   _self.Y[v2]*_self.X[v3] -
                   _self.Y[v1]*_self.X[v2] )

 A1 = (1.0/2.0) * ( _self.X[v2]*_self.Y[v3] +
                    _xp*_self.Y[v2] +
                    _yp*_self.X[v3] -
                    _xp*_self.Y[v3] -
                    _self.Y[v2]*_self.X[v3] -
                    _yp*_self.X[v2] )
```

```
 A2 = (1.0/2.0) * ( _xp*_self.Y[v3] +
                    _self.X[v1]*_yp +
                    _self.Y[v1]*_self.X[v3] -
                    _self.X[v1]*_self.Y[v3] -
                    _yp*_self.X[v3] -
                    _self.Y[v1]*_xp )

 _self.l1 = A1/A
 _self.l2 = A2/A
 _self.l3 = 1.0 - _self.l2 - _self.l1

 return ( (_self.l1>=0.0-EPSlocal) and (_self.l1<=1.0+EPSlocal) and \
        (  _self.l2>=0.0-EPSlocal) and (_self.l2<=1.0+EPSlocal) and \
        (  _self.l3>=0.0-EPSlocal) and (_self.l3<=1.0+EPSlocal) )

def setCentroid(_self,_vxd,_vyd):
 # interpolando no centroid
 zeros = np.zeros( (_self.numNodes-_self.numVerts),dtype='float')
 _vxd = np.append(_vxd,zeros)
 _vyd = np.append(_vyd,zeros)
 for e in range(0,_self.numElems):
  [v1,v2,v3,v4] = _self.IEN[e]
  _vxd[v4] = (_vxd[v1] + _vxd[v2] + _vxd[v3])/3.0
  _vyd[v4] = (_vyd[v1] + _vyd[v2] + _vyd[v3])/3.0
 return [_vxd,_vyd]

def setBC(_self,_idv,_vbc,_vd):
 # impondo c.c. de velocidade
 # condicao de contorno (Dirichlet) para vxd,vyd
 for i in _idv:
  _vd[i] = _vbc[i]
 return _vd
```

A.2 Código para a solução utilizando o MEF

```
"""
Solucao Equacoes de Navier-Stokes e Transferencia de Calor
Created on Sat Jun 12 11:17:27 2021
@author: Pedro Mattos da Silva
"""

import meshio
import numpy as np
import matplotlib.pyplot as plt
import timeit
from CoolProp.CoolProp import PropsSI
import math
import pandas as pd
import SL    # Importa a funcao semi-Lagrangeano


start = timeit.default_timer() # Inicio da contagem do tempo de simulacao

D = 1/1000     # [m] Definicao da escala do dominio

# Importacao da malha
msh = meshio.read('simples4v3.msh')
X = D*msh.points[:,0]
Y = D*msh.points[:,1]

IENa = msh.cells['triangle']
IENbound = msh.cells['line']
IENboundTypeElem = list(msh.cell_data['line']['gmsh:physical'] - 1)
boundNames = list(msh.field_data.keys())
IENboundElem = [boundNames[elem] for elem in IENboundTypeElem]
npoints = len(X)
ne = IENa.shape[0]
```

```
cc = np.unique(IENbound.reshape(IENbound.size))
ccName = [[]for i in range(len(X))]
for elem in range(0,len(IENbound)):
    ccName[IENbound[elem][0]] = IENboundElem[elem]
    ccName[IENbound[elem][1]] = IENboundElem[elem]

# Implementacao dos centroides e criacao da matriz oElem
# (IENa -> sem centroides; IEN -> com centroides)
cx = []
IEN = np.zeros((ne, 4), dtype = 'int')
IEN[:,:-1] = IENa
Xc = X.copy()
Yc = Y.copy()
oElem = -1*np.ones((ne,3), dtype='int')
for i in range (0,ne):
    # Centroides
    IEN[i,3] = npoints+i
    cx = (X[IEN[i,0]]+ X[IEN[i,1]] + X[IEN[i,2]])/3
    cy = (Y[IEN[i,0]]+ Y[IEN[i,1]] + Y[IEN[i,2]])/3
    Xc = np.append(Xc,cx)
    Yc = np.append(Yc,cy)

    # Criacao oElem
    for j in range (0,3):
        a = [IENa[i,j-2],IENa[i,j-1]]
        for e in range (0, ne):
            if e!= i and a[0] in IENa[e] and a[1] in IENa[e]:
                oElem[i,j] = e
                break

npointsc = len(Xc)

# Plotagem da malha
plotsize = (8,5)
```

```
fig1, ax1 = plt.subplots(figsize=plotsize, dpi = 1200)
ax1.set_aspect('equal')
ax1.triplot(X,Y,IENa,'k-', linewidth = 0.5)
ax1.plot(Xc,Yc, 'ko', marker = '.', markersize = 1.5, markeredgewidth=0.0)
plt.xlabel('x (m)')
plt.ylabel('y (m)')
plt.show()

# Criacao da matriz neighElem
neighElemnp = np.zeros((npoints,1), dtype = 'int')
neighElem = neighElemnp.tolist()
for i in range(0, len(neighElem)):
    b = []
    for j in range(0,len(IENa)):
        if i in IENa[j]:
            b = b + [j]
    neighElem[i] = b


# Importacao das propriedades do fluido
Pfluido = 101325  #[Pa]
Tfluido = 20      #[Celsius]
var = 'Water'

rho = PropsSI('D', 'T', 273.15+Tfluido, 'P', Pfluido, var) # [kg/m^3]
k = PropsSI('L', 'T', 273.15+Tfluido, 'P', Pfluido, var) # [W/m*K]
cp = PropsSI('C', 'T', 273.15+Tfluido, 'P', Pfluido, var) # [kJ/kg*K]
mi = PropsSI('V', 'T', 273.15+Tfluido, 'P', Pfluido, var) # [Pa*s]
nu = mi/rho       # [m^2/s]
alpha = k/(rho*cp) # [m^2/s]

# Propriedades e condicoes iniciais
U = 0.1      # [m/s] (entrada do fluido)
dt = 0.00025   # [s]
```

```
nIter = 480
Re = U*D/nu
Pr = nu/alpha

print ("Re: ", Re)
print("Pr: ", Pr)
print("Elementos: ",len(IEN))
print("Pontos: ", npoints)
print("dt: ", dt, " s")

# Montagem das matrizes
K = np.zeros( (npointsc,npointsc),dtype='float' )
M = np.zeros( (npointsc,npointsc),dtype='float' )

Gx = np.zeros( (npointsc,npoints),dtype='float' )
Gy = np.zeros( (npointsc,npoints),dtype='float' )

Gvx = np.zeros( (npointsc,npointsc),dtype='float' )
Gvy = np.zeros( (npointsc,npointsc),dtype='float' )

for e in range(0,ne): # Loop por todos os elementos
    # Definicao do elemento e
    v1 = IEN[e,0]
    v2 = IEN[e,1]
    v3 = IEN[e,2]
    v4 = IEN[e,3]

    area = 0.5*np.linalg.det([ [1,X[v1],Y[v1]],
                               [1,X[v2],Y[v2]],
                               [1,X[v3],Y[v3]] ] )

    # Matriz 'm' elementar
    melem = (area/840)*np.array([[83,13,13,45],
                                 [13,83,13,45],
```

```
                                [13,13,83,45],
                                [45,45,45,243]])

bi = Y[v2] - Y[v3]
bj = Y[v3] - Y[v1]
bk = Y[v1] - Y[v2]
ci = X[v3] - X[v2]
cj = X[v1] - X[v3]
ck = X[v2] - X[v1]

z = (1.0/(4*area))*(bj**2+bj*bk+bk**2+cj**2+cj*ck+ck**2)

# Matriz 'k' elementar
kxelem = (1.0/(4*area))*np.array([ [bi*bi,bi*bj,bi*bk],
                                   [bj*bi,bj*bj,bj*bk],
                                   [bk*bi,bk*bj,bk*bk]])
kyelem = (1.0/(4*area))*np.array([ [ci*ci,ci*cj,ci*ck],
                                   [cj*ci,cj*cj,cj*ck],
                                   [ck*ci,ck*cj,ck*ck]])
kelema = (kxelem + kyelem)
d = (9/10)*z*np.ones((3,3),dtype = 'float')
kelema += d
kelem = (-27*z/10)*np.ones((4,4), dtype = 'float')
kelem[:3,:3] = kelema
kelem[3,3] = (81/10)*z

# Matrizes gx e gx elementares
gxelea = (1/6)*np.array([[bi,bj,bk],[bi,bj,bk],[bi,bj,bk]])
gyelea = (1/6)*np.array([[ci,cj,ck],[ci,cj,ck],[ci,cj,ck]])

gxele = np.zeros((4,3), dtype = 'float')
gyele = np.zeros((4,3), dtype = 'float')

gxele[3,0] = bi
```

```
gxele[3,1] = bj
gxele[3,2] = bk

gxele = (-9/40)*gxele
gxele[:3] = (9/20)*gxelea+np.transpose(gxelea)

gyele[3,0] = ci
gyele[3,1] = cj
gyele[3,2] = ck

gyele = -(9/40)*gyele
gyele[:3] = (9/20)*gyelea+np.transpose(gyelea)

# Matrizes gvx e gvy elementares
gvxele = np.zeros((4,4), dtype = 'float')
gvyele = np.zeros((4,4), dtype = 'float')

gvxele[3,0] = (9/40)*bi
gvxele[3,1] = (9/40)*bj
gvxele[3,2] = (9/40)*bk
gvxele[0,3] = -(9/40)*bi
gvxele[1,3] = -(9/40)*bj
gvxele[2,3] = -(9/40)*bk

gvyele[3,0] = (9/40)*ci
gvyele[3,1] = (9/40)*cj
gvyele[3,2] = (9/40)*ck
gvyele[0,3] = -(9/40)*ci
gvyele[1,3] = -(9/40)*cj
gvyele[2,3] = -(9/40)*ck

gvxele[:3,:3] = (11/20)*gxelea+(9/20)*np.transpose(gxelea)

gvyele[:3,:3] = (11/20)*gyelea+(9/20)*np.transpose(gyelea)
```

```
    # Montagem das matrizes globais
    for ilocal in range(0,4):
      iglobal = int(IEN[e,ilocal])
      for jlocal in range(0,4):
        jglobal = int(IEN[e,jlocal])

        K[iglobal,jglobal] += kelem[ilocal,jlocal]
        M[iglobal,jglobal] += melem[ilocal,jlocal]

        Gvx[iglobal,jglobal] += gvxele[ilocal,jlocal]
        Gvy[iglobal,jglobal] += gvyele[ilocal,jlocal]


      for jlocal in range(0,3):
        jglobal = int(IEN[e,jlocal])
        Gx[iglobal,jglobal] += gxele[ilocal,jlocal]
        Gy[iglobal,jglobal] += gyele[ilocal,jlocal]

Dx = np.transpose(Gx)
Dy = np.transpose(Gy)


# Implementacao das condicoes de contorno
bvxcc = np.zeros((npointsc),dtype = 'float') # Velocidade em x
bvycc = np.zeros((npointsc),dtype = 'float') # Velocidade em x
bpcc = np.zeros((npoints),dtype = 'float') # Pressao
bTcc = np.zeros((npointsc),dtype = 'float') # Temperatura

ccv = cc.tolist() #vetor contendo apenas os pontos do contorno

for i in cc:
    if ccName[i] == 'top':
        bvxcc[i] = 0
```

```
            bvycc[i] = 0
            bpcc[i] = 0
            bTcc[i] = 50
        if ccName[i] == 'bottom':
            bvxcc[i] = 0
            bvycc[i] = 0
            bpcc[i] = 0
            bTcc[i] = 50
        if ccName[i] == 'left':
            bvxcc[i] = U
            bvycc[i] = 0
            bpcc[i] = 0
            bTcc[i] = 20
        if ccName[i] == 'right':
            bvxcc[i] = 0
            bvycc[i] = 0
            bpcc[i] = 0
            bTcc[i] = 0
            ccv.remove(i)
        if ccName[i] == 'hole':
            bvxcc[i] = 0
            bvycc[i] = 0
            bpcc[i] = 0
            bTcc[i] = 50

# Inicializacao dos campos
vx = np.zeros((npointsc), dtype = 'float')
vy = np.zeros((npointsc), dtype = 'float')
p = np.zeros((npoints), dtype = 'float')
T = 20*np.ones((npointsc), dtype = 'float') # Inicializando com 20 graus
v = np.zeros((npoints), dtype = 'float') # Modulo da velocidade

# Criacao da matriz A global que acopla velocidade e pressao
A = np.zeros((2*npointsc+npoints,2*npointsc+npoints), dtype = 'float')
```

```
A[:npointsc,:npointsc] = M/dt + nu*K
A[npointsc:2*npointsc,npointsc:2*npointsc] = M/dt + nu*K
A[0:npointsc,2*npointsc:2*npointsc+npoints] = -(1/rho)*Gx
A[npointsc:2*npointsc,2*npointsc:2*npointsc+npoints] = -(1/rho)*Gy
A[2*npointsc:2*npointsc+npoints,0:npointsc] = Dx
A[2*npointsc:2*npointsc+npoints,npointsc:2*npointsc] = Dy

# Imposicao das condicoes de contorno na matriz A
for i in cc:
    if ccName[i] == 'top':
        A[i,:] =0
        A[i,i] =1
        A[i+npointsc,:] = 0
        A[i+npointsc,i+npointsc] = 1
    if ccName[i] == 'bottom':
        A[i,:] =0
        A[i,i] =1
        A[i+npointsc,:] = 0
        A[i+npointsc,i+npointsc] = 1
    if ccName[i] == 'left':
        A[i,:] =0
        A[i,i] =1
        A[i+npointsc,:] = 0
        A[i+npointsc,i+npointsc] = 1
    if ccName[i] == 'right':
        A[i+2*npointsc,:] = 0
        A[i+2*npointsc,i+2*npointsc] = 1
    if ccName[i] == 'hole':
        A[i,:] =0
        A[i,i] =1
        A[i+npointsc,:] = 0
        A[i+npointsc,i+npointsc] = 1

# Vetores que armazenam as diferencas relativas
```

```
dprel = []
dvrel = []

dT = 1   # Variavel que calcula a diferenca do campo entre iteracoes
dv = 10  # Variavel que calcula a diferenca do campo entre iteracoes

n = 0

# Loop solucao
while (n < nIter): #Escolha do criterio de parada
    dp = np.sum(p)
    dv = np.sum(v)

    # Calculo das velocidades nos vertices pelo semi-Lagrangeano
    sl = SL.Linear(IEN,Xc,Yc,neighElem,oElem,vx,vy)
    sl.compute(dt)
    vxd = sl.conv*vx[0: npoints]
    vyd = sl.conv*vy[0: npoints]

    # Interpolacao para o centroide dos valores obtidos
    [vxd,vyd] = sl.setCentroid(vxd,vyd)

    # Impondo a c.c. de velocidade
    vxd = sl.setBC(ccv,bvxcc,vxd)
    vyd = sl.setBC(ccv,bvycc,vyd)

    # Vetor RHS b (com conveccao explicita)
    b1 = (M/dt)@vxd # Semi-Lagrangian x
    b2 = (M/dt)@vyd # Semi-Lagrangian y

    # Acoplamento das velocidades e pressoes no vetor b chamado de 'bo'
    bp = np.zeros((npoints),dtype = 'float')
    bo  = np.append(b1,b2)
    bo = np.append(bo,bp)
```

```
# Implementacao das c.c. no vetor bo
for i in ccv:
    bo[i] = bvxcc[i]
    bo[i+npointsc] = bvycc[i]
for i in cc:
    if ccName[i] == 'right':
        bo[i+2*npointsc] = bpcc[i]

x = np.linalg.solve(A,bo) # Solucao do sistema

# Separacao dos resultados
vx = x[:npointsc]
vy = x[npointsc:2*npointsc]
p = x[2*npointsc:3*npointsc]

# Calculo do modulo das velocidades apenas nos vertices
for i in range (0, npoints):
    v[i] = math.sqrt(vx[i]**2 + vy[i]**2)

# Calculo das diferencas relativas para analise da solucao
dv = abs(np.sum(v)-dv)
dp = abs(np.sum(p)-dp)

dvr = dv/np.sum(v)
dpr = dp/np.sum(p)
print("dvrel: ", dvr)
print("dprel: ", dpr)
dprel.append(dpr)
dvrel.append(dvr)

# Temperatura
# O calculo da temperatura apresenta instabilidade para Reynolds altos
dT = np.sum(T)
```

```
vg = np.dot(np.diag(vxd),Gvx) + np.dot(np.diag(vyd),Gvy)
At = (M/dt + alpha*K + vg) # Matriz A -> temperatura
bt = (M/dt)@T              # Vetor b -> temperatura

# Imposicao das condicoes de contorno da temperatura
for i in cc:
    if ccName[i] != 'right':
        At[i,:] = 0
        At[i,i] = 1
        bt[i] = bTcc[i]

T = np.linalg.solve(At,bt) # Solucao do sistema

dT = abs(np.sum(T)-dT)    # Diferenca do campo entre iteracoes
print("dT: ", dT)

# Escolha de um passo para salvamento de um arquivo vtk com os
# resultados e um arquivo csv contendo as diferencas relativas
# para analise
if (n+20)%20 == 0:
    try:
        df = pd.DataFrame(list(zip(dprel,dvrel)), \
                        columns=['dp/p','dv/v'])
        df.to_csv('nome_arquivo.csv', index=True)
        point_data = {'Pressao' : p, 'u' : vx[:npoints], \
                    'v' : vy[:npoints], 'V' : v, \
                        'Temperatura' : T[:npoints]}
        meshio.write_points_cells('nome_arquivo2.vtk',\
                          msh.points,msh.cells,point_data=point_data)
    except Exception:
        pass

print(n)
n  += 1
```

```
stop = timeit.default_timer() # Finaliza a contagem de tempo


print ("Re: ", Re)
print("Pr: ", Pr)
print("Elementos: ",len(IEN))
print("Pontos: ", npoints)
print("dt: ", dt)
print("Iteracoes: ", n)
print("Tempo de simulacao: ", stop - start, " s")
```