Politécnica UFRJ

ANÁLISE DA EFICIÊNCIA DE FILTRAGEM EM EMISSÕES DE BIOCOMBUSTÍVEIS POR MEIO DE SIMULAÇÃO NUMÉRICA EM MALHAS NÃO ESTRUTURADAS

Anna Bárbara Serejo Coimbra

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 Engenheira.

Orientador: Gustavo Rabello dos Anjos

Rio de Janeiro

Outubro de 2023

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

ANÁLISE DA EFICIÊNCIA DE FILTRAGEM EM EMISSÕES DE BIOCOMBUSTÍVEIS POR MEIO DE SIMULAÇÃO NUMÉRICA EM MALHAS NÃO ESTRUTURADAS

Anna Bárbara Serejo Coimbra

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, Ph.D.

Prof. Átila Pantaleão Silva Freire, Ph.D.

Prof. Juliana Braga Rodrigues Loureiro, D.Sc.

RIO DE JANEIRO, RJ - BRASIL
OUTUBRO DE 2023

Coimbra, Anna Bárbara Serejo

Análise da Eficiência de Filtragem em Emissões de Biocombustíveis por meio de simulação numérica em malhas não estruturadas/ Anna Bárbara Serejo Coimbra. – Rio de Janeiro: UFRJ/Escola Politécnica, 2023.

XVI, 117 p.: il.; 29, 7cm.

Orientador: Gustavo Rabello dos Anjos

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

Referências Bibliográficas: p. 68 – 70.

1. Análise numérica. 2. Mecânica dos fluidos. 3. Biocombustíveis. 4. Filtros de partículas. I. Rabello dos Anjos, Gustavo. II. Universidade Federal do Rio de Janeiro, UFRJ, Curso de Engenharia Mecânica. III. Análise da Eficiência de Filtragem em Emissões de Biocombustíveis por meio de simulação numérica em malhas não estruturadas.

Dedico esse trabalho ao meu pai, que me ensinou a apreciar as ciências, humanas e exatas, a literatura e a matemática.

Pra que nossa esperança

Seja mais que a vingança

Seja sempre um caminho

Que se deixa de herança

Ivan Lins, *Novo Tempo*

Agradecimentos

Agradeço aos meus pais por todo o apoio e suporte que me deram ao longo de toda a minha trajetória acadêmica e em especial durante a minha graduação. Obrigada por me incentivarem a sonhar alto e apoiarem os meus sonhos.

Aqui gostaria também de lembrar dos amigos que me acompanharam durante a graduação, especialmente nesses últimos anos marcados por tanta incerteza. Carolina Coutinho, Mariana Medeiros, João Pedro Rodrigues, Barbara Temer, Nicole Quintella, Rodrigo Tanajura, Amanda Kezen e Thomás Costa, muito obrigada por todos os momentos de bate-papo entre aulas e almoços no fundão. A companhia e a troca de ideias que tive com vocês ao longo da graduação tornou meu dia a dia na Ilha do Fundão muito mais feliz, vivo e interessante. Outra pessoa que enche meu cotidiano de alegria e risadas é o meu namorado, Gabriel Pessanha. Desejo agradecer a ele em especial por todo o apoio, companheirismo e por enriquecer a minha vida. Você me estimula a ser uma pessoa melhor e a vida é mais feliz com você. A todos vocês eu dedico o poema "The Orange", da poeta Wendy Cope.

Agradeço ao amigo da engenharia civil, Victor Trindade, por toda a paciência com minhas dúvidas sobre como funcionam inúmeras burocracias relacionadas à UFRJ e por me convidar para fazer parte de seus projetos de representação discente. Essas experiências foram muito importantes para mim e ter a sua companhia certamente tornou a experiência mais divertida.

Gostaria de agradecer também à todo o corpo docente do Departamento de Engenharia Mecânica da UFRJ e, principalmente, ao meu orientador, Professor Gustavo Rabello dos Anjos. Muito obrigada por todo o apoio e paciência e por me apresentar a área de mecânica dos fluidos computacional. Tudo que aprendi sob sua orientação abriu portas para mim e sou muito grata por isso.

Ainda no Departamento de Engenharia Mecânica, gostaria de agradecer aos Pro-

fessores Átila Pantaleão e Juliana Loureiro por me receberem de braços abertos no Núcleo Interdisciplinar de Dinâmica dos Fluidos, quando eu ainda estava no primeiro período da graduação. Muito obrigada por terem me oferecido essa oportunidade tão importante para mim, que foi minha primeira experiência em iniciação científica e pesquisa acadêmica.

Por fim, agradeço também à Agência Nacional do Petróleo, Gás Natural e Biocombustíveis - ANP, à Financiadora de Estudos e Projetos – FINEP e ao Ministério da Ciência, Tecnologia e Inovação – MCTI que, por meio do Programa de Recursos Humanos ANP para o Setor Petróleo e Gás – PRH-ANP/MCTI, garantiram os recursos necessários para a realização deste trabalho por meio da bolsa de graduação do programa PRH 8: Engenharia Mecânica para o Uso Eficiente de Biocombustíveis.

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

ANÁLISE DA EFICIÊNCIA DE FILTRAGEM EM EMISSÕES DE BIOCOMBUSTÍVEIS POR MEIO DE SIMULAÇÃO NUMÉRICA EM MALHAS NÃO ESTRUTURADAS

Anna Bárbara Serejo Coimbra

Outubro/2023

Orientador: Gustavo Rabello dos Anjos

Departamento: Engenharia Mecânica

As mudanças climáticas e suas consequências fazem com que seja necessário que as atividades humanas passem por um processo de descarbonização. Diante desse cenário, os biocombustíveis se apresentam como uma boa alternativa pois provocam uma quantidade menor de emissões de gás carbônico em relação aos combustíveis fósseis. Esse potencial para diminuição de emissões com a adoção de biocombustíveis pode ser ainda maior por meio da adoção de filtros de partículas mais eficientes na filtragem das emissões de combustão. Por isso, o objetivo desse trabalho é estudar diferentes geometrias de canais porosos desse tipo de filtro e identificar quais fatores geométricos tornam esses filtros mais eficientes, contribuindo para uma melhor compreensão sobre esse fenômeno. Os canais porosos foram simulados utilizando a formulação corrente-vorticidade aplicada no método de elementos finitos em um código escrito em **Python** pela autora especificamente para este trabalho. Os parâmetros analisados foram a velocidade do escoamento, a vorticidade e a função corrente. A partir dos resultados obtidos, foi observado que o tamanho e a distribuição dos obstáculos que representam o meio poroso dentro do canal são fatorem que influenciam na eficiência da filtragem.

Palavras-chave: Análise numérica; Mecânica dos fluidos; Biocombustíveis; Filtros de partículas.

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

ANALYSIS OF FILTRATION EFFICIENCY IN BIOCOMBUSTION EMISSIONS THROUGH NUMERICAL SIMULATION ON UNSTRUCTURED GRIDS

Anna Bárbara Serejo Coimbra

October/2023

Advisor: Gustavo Rabello dos Anjos

Department: Mechanical Engineering

Climate change and its consequences make it necessary for human activities to undergo a decarbonization process. Given this scenario, biofuels present themselves as a good alternative as they cause fewer carbon dioxide emissions compared to fossil fuels. This potential for reducing emissions with the adoption of biofuels can be even greater through the adoption of particulate filters that are more efficient in filtering combustion emissions. Therefore, the objective of this work is to study different geometries of porous channels of this type of filter and identify which geometric factors make these filters more efficient, contributing to a better understanding of this phenomenon. The porous channels were simulated with a **Python** code written by the author specifically for this work. This code used the current-vorticity formulation applied in the finite element method and the parameters analyzed were flow speed, vorticity and current function. From the results obtained, it was observed that the size and distribution of the obstacles that represent the porous medium within the channel are factors that influence the filtration efficiency.

Keywords: Numerical analysis; Fluid mechanics; Biofuels; Particle filters.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviações e Siglas

MEF ..... Método de Elementos Finitos

GEE ..... Gases de Efeito Estufa

EPE ..... Empresa de Pesquisa Energética

DPF ..... Filtro de Partículas Diesel

IEN ..... Matriz de Conectividade

Lista de Símbolos

τ ..... Tensão de Cisalhamento

μ ..... Viscosidade Dinâmica

ρ ..... Massa Específica ou Densidade

ν ..... Viscosidade Cinemática

Re ..... Número de Reynolds

$\mathbf{V}$ ..... Velocidade do Escoamento

u ..... Velocidade do Escoamento (na direção x)

v ..... Velocidade do Escoamento (na direção y)

L ..... Comprimento Característico

ψ ..... Função Corrente

ω_z ..... Vorticidade

$\mathbf{F}$ ..... Força

m ..... Massa

σ ..... Força de Superfície

p ..... Pressão

$\mathbf{g}$ ..... Gravidade

d ..... Deslocamentos

$\mathbf{K}$ ..... Matriz de Rigidez

M ..... Matriz de Massa

G ..... Matriz de Gradiente

α ..... Fator de Qualidade do Filtro

θ ..... Densidade dos obstáculos

Capítulo 1

Introdução

A quantidade de emissões produzidas pelas atividades humanas é maior do que o meio ambiente e a atmosfera conseguem absorver sem grandes consequências. De fato, caso o atual paradigma energético não seja alterado, o equilíbrio ambiental da Terra será levado a um limite, desastres naturais serão mais frequentes e a vida no planeta será ameaçada (IPCC, 2022). Diante desse contexto, a elaboração de alternativas energéticas é essencial e, entre as fontes de energia que se apresentam como substitutos para os combustíveis fósseis, os biocombustíveis têm tido destaque.

Por serem provenientes de biomassa, a utilização de biocombustíveis como o biodiesel no lugar de diesel mineral promove uma diminuição das emissões. Essa característica por si só é um grande incentivo para a adoção de biocombustíveis diante de compromissos nacionais e internacionais que visam reverter a ameaças que o aquecimento global e as mudanças climáticas representam. No entanto, é possível diminuir ainda mais as emissões provenientes de biocombustíveis por meio da adoção de filtros de partículas mais eficientes em máquinas que utilizem essa fonte de energia, como carros.

Um filtro de partículas padrão utilizado para filtrar os gases provenientes da combustão que ocorre em motores de carros possui uma geometria típica, esquematizada na figura 1.1. No entanto, outras geometrias são possíveis de serem adotadas na construção desses filtros. O objetivo deste trabalho é estudar como o escoamento se comporta em canais de filtros como esses ao adotar geometrias alternativas.

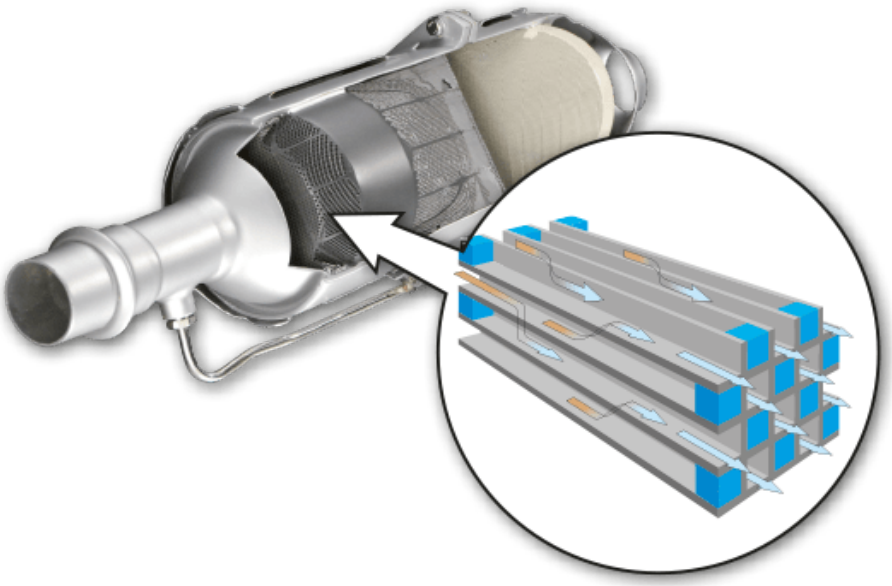

Figura 1.1: Ilustração de um filtro de partículas diesel destacando a organização geométrica dos seus canais. Fonte: METAL-5

Para isso, levando em consideração que a composição de um filtro é semelhante a um meio poroso, a porosidade será emulada por meio de diferentes obstáculos distribuídos de formas diversas ao longo de um canal por onde um escoamento transversal atravessa esses obstáculos. Esses escoamentos serão simulados numericamente a partir de códigos numéricos desenvolvidos pela autora do texto para este trabalho. As diferentes geometrias por onde passarão os escoamentos também foram geradas geradas a partir de códigos numéricos produzidos pela autora especificamente para esse fim que descrevem as geometrias desejadas na linguagem do software **Gmsh**. A partir das simulações executadas, pretende-se evidenciar quais características geométricas são desejáveis para que o processo de filtragem seja mais eficiente. Por meio dos resultados obtidos, este trabalho busca promover uma melhor compreensão sobre o fenômeno estudado e os escoamentos através de filtros.

Para atingir essa meta, uma revisão bibliográfica de trabalhos de temas semelhantes foi realizada em conjunto com uma recordação de conceitos indispensáveis da mecânica dos fluidos. Essas recapitulações estão apresentadas no capítulo 2 deste trabalho e incluem definições de parâmetros importantes, exposição das equações da continuidade e da conservação da quantidade de movimento, dedução das equações de Navier-Stokes e de sua forma adimensional. Além disso, nesse mesmo capítulo,

o método de elementos finitos (MEF) e a formulação corrente-vorticidade são apresentados.

No capítulo 3, as quatro etapas da metodologia escolhida para este trabalho são expostas. A primeira etapa consiste na elaboração das malhas responsáveis por discretizar o domínio das geometrias desejadas. A geração dessas malhas foi realizada por meio do software **Gmsh**, com auxílio do código escrito em **Python** feito pela autora. Em seguida, foi feita a apresentação do código numérico escrito em **Python** que utiliza MEF para solucionar as equações da formulação corrente-vorticidade para os pontos das malhas geradas. Esse código numérico, escrito pela autora do texto, produz as simulações que mostram o comportamento do escoamento ao longo do tempo para cada geometria estudada.

Em relação ao processamento de resultados obtidos por meio desse código numérico, a visualização de dados foi realizada no software **Paraview**. Nesse programa, foi possível extrair informações essenciais sobre o escoamento para avaliar a qualidade de cada tipo de filtro simulado. Para essa avaliação, foram definidos dois novos parâmetros: a densidade de obstáculos, obtida pela razão entre a área ocupada pelos obstáculos e a área do canal, e o fator de qualidade de filtragem, que relaciona a velocidade horizontal de saída e de entrada do canal.

Após definir a metodologia adotada, a verificação do código numérico para solução da formulação corrente-vorticidade foi realizada e apresentada no capítulo 4. Para a verificação, um problema clássico de mecânica dos fluidos, o escoamento entre placas planas, foi reproduzido utilizando o código numérico desenvolvido. Os resultados numéricos e analíticos foram, então, comparados e se demonstrou que há compatibilidade entre os resultados.

No capítulo 5 os resultados para as simulações realizadas foram expostas acompanhadas de algumas discussões sobre as implicações desses resultados. A qualidade da filtragem de cada uma das geometrias foi estimada e geometrias de mesmo tipo tiveram suas qualidades comparadas entre si.

Enfim, o capítulo 6 recapitula os temas abordados ao longo do trabalho e avalia os resultados obtidos. A partir desses resultados, foram apresentadas sugestões de temas de trabalho que podem contribuir para um entendimento do problema em questão que vá além do presente estudo.

Capítulo 2

Revisão Bibliográfica

2.1 Biocombustíveis e filtros de escapamento

Biocombustíveis se destacam por serem fontes de energia que se propõem como uma alternativa aos combustíveis de origem fóssil, uma vez que a matéria prima dos biocombustíveis é a biomassa. Essa característica faz com que eles emitam menos partículas poluentes e gases de efeito estufa (GEE), o biodiesel brasileiro, por exemplo, emite menos 70% GEE do que o diesel mineral (CERRI *et al.*, 2017) e, diante da necessidade de reverter o desequilíbrio ambiental provocado pelo aumento da temperatura média do planeta, os biocombustíveis podem fazer parte de uma estratégia de descarbonização. De fato, essa aptidão para a descarbonização fez com que o Plano Nacional de Energia 2050 (MINISTÉRIO DE MINAS E ENERGIA – MME. e EPE, 2020), realizado pela Empresa de Pesquisa Energética (EPE), destacasse a importância dos biocombustíveis para descarbonizar o setor de transportes e outras atividades cuja eletrificação encontre dificuldades.

Com o objetivo de diminuir as emissões de a nível mundial e diminuir ou reverter os efeitos das mudanças climáticas, diversos compromissos internacionais foram e estão sendo desenvolvidos. Um dos principais acordos internacionais nessa direção é o Acordo de Paris, por meio do qual 195 países, incluindo o Brasil, se comprometeram em atingir diferentes metas de diminuição de emissões para que o aumento da temperatura média no planeta fique limitado a preferencialmente 1,5^{o}C e não mais do que 2°C. Para que essas metas sejam atingidas, pesquisas na área de energias renováveis e alternativas são fundamentais e há uma janela de oportunidade

para estudos, como o presente trabalho de conclusão de curso, investigarem como diminuir ainda mais as emissões dessa fonte de energia.

Uma forma de realizar essa diminuição de emissões é por meio de filtros de exaustão mais eficientes. Esses filtros retêm uma parcela das emissões da combustão de biocombustíveis, permitindo que menos poluentes cheguem na atmosfera. Nos trabalhos de DE SOUZA (2021), DE CERQUEIRA (2022) e SPESANI (2023), diferentes geometrias de filtros junto com suas respectivas eficiências de filtragem foram analisadas. Porém, devido ao grande número de possibilidades geométricas para esses filtros, ainda há espaço para novas análises.

Antes de sugerir novas geometrias para filtros de exaustão, é importante entender como é a geometria padrão de um desses filtros e o seu funcionamento. O tipo de filtro que será analisado se chama Filtro de Partículas Diesel (DPF) e é utilizado para filtrar o material particulado da combustão tanto do diesel quanto do biodiesel. A sua geometria é composta por inúmeros pequenos canais que formam uma estrutura semelhante a uma colmeia, na qual os gases e o material particulado da combustão são forçados a passar por vários filtros em sequência até esse escoamento encontrar a atmosfera. Um desenho esquemático desse tipo de filtro pode ser visto abaixo:

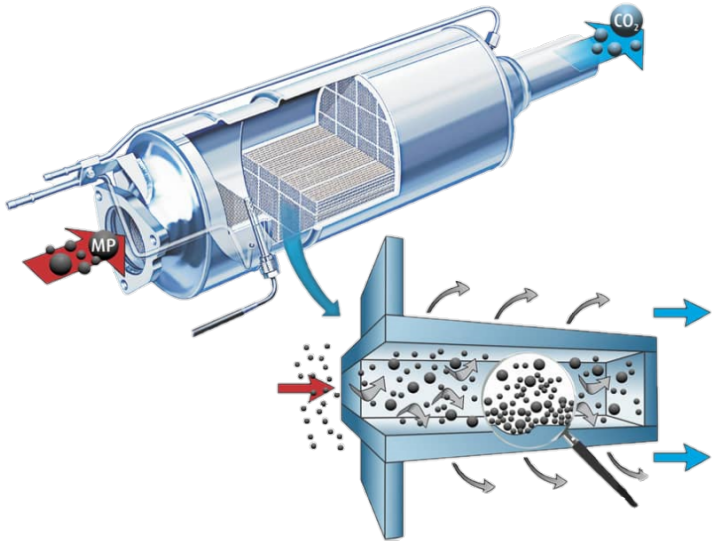

Figura 2.1: Desenho esquemático de um filtro de partículas diesel onde é possível ver o material particulado (MP) entrar e o gás carbônico (CO_2) remanescente, após a passagem pelo filtro, sair. Em destaque é possível ver a estrutura de um dos inúmeros canais sequenciais do filtro. Fonte: O MECÂNICO (2020)

Vale destacar que as paredes de material filtrante podem ser interpretadas como um meio poroso cuja disposição de obstáculos é aleatória. Fotos realizadas por SEONG *et al.* (2019) com raio X em vistas 2D de amostras de filtros evidenciam a estrutura porosa das paredes desses filtros e podem ser visualizadas na figura 2.2.

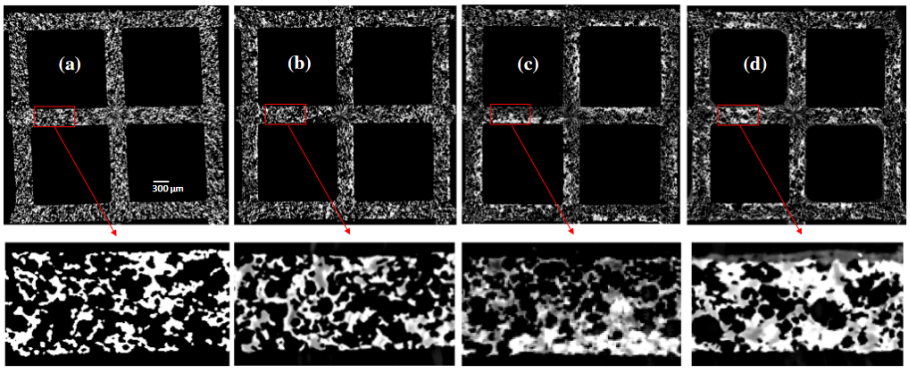

Figura 2.2: Fotos feitas com raio X de vistas 2D de amostras de filtros de partículas. Fonte: SEONG *et al.* (2019).

Após visualizar a figura 2.2 e notar o aspecto poroso das paredes que compõem um filtro DPF, é conveniente ressaltar que as partículas que um filtro desse tipo é capaz de capturar são extremamente pequenas. Segundo o estudo de ROSSOMANDO *et al.* (2021), um filtro DPF típico atinge eficiência máxima de filtragem com partículas de tamanho entre 10 e 40 nm. Portanto, é importante ter em mente que a granulometria das partículas filtradas por esses filtros é dessa ordem e que a escala do problema abordado é bastante pequena. A pequenez dessa escala foi um dos fatores que fez com que as equações que descrevem o escoamento estudado fossem utilizadas em sua forma adimensional. Essa decisão foi tomada para evitar problemas de convergência e de limitação computacional no código numérico já que, ao se utilizar dimensões reais, os valores de intervalo de tempo entre iterações precisavam ser extremamente pequenos para descrever corretamente o escoamento.

Como MESQUIDA (2019) constatou em sua revisão de literatura em seu trabalho sobre filtros de partículas em veículos automotivos, a relevância desses filtros aumentou nos últimos anos tanto por razões ambientais quanto de saúde. Regulações automotivas restringem a quantidade de partículas que pode ser emitida por veículos e, uma vez que existe uma associação entre a poluição em cidades e um aumento nas internações por doenças respiratórias (ARBEX *et al.*, 2012), pode se tornar de inte-

resse público a exigência de filtros DPF com filtração mais eficiente. Essa questão é mais um motivador para as pesquisas sobre esses filtros.

Além disso, ao se considerar o uso de filtros DPF para filtrar os rejeitos da combustão de biocombustíveis especificamente, biocombustíveis podem promover um funcionamento mais eficiente desses filtros do que combustíveis de origem fóssil. Isso ocorre porque a capacidade filtrante do filtro DPF se regenera após ele atingir certas temperaturas e queimar as fuligens capturadas. Estudos realizados por RODRÍGUEZ-FERNÁNDEZ *et al.* (2017) mostram que o processo de regeneração dos filtros DPF é mais econômico ao se utilizar biocombustíveis. A conclusão desse estudo incentiva que a associação de filtros DPF com biocombustíveis continue sendo investigada. Seguindo essa tendência, o presente estudo considera essa associação promissora como mais um motivador para o estudo do tema.

2.2 Revisão de conceitos

Definições importantes

As ferramentas mais importantes para a elaboração do presente trabalho são as equações fundamentais que regem a mecânica dos fluidos. Portanto, convém fazer uma breve revisão dessas equações e de outros conceitos importantes que compõem a base teórica dos fenômenos estudados.

Uma das principais características que distingue os fluidos, sejam líquidos ou gasosos, dos sólidos é a forma como os fluidos respondem à uma aplicação de uma força tangencial à sua superfície. Enquanto os sólidos sofrem uma deformação finita, os fluidos são deformados continuamente sob a aplicação de uma força tangencial. A tensão de cisalhamento τ provocada por essa força nos fluidos é, por sua vez, proporcional à taxa de mudança dessa deformação. A constante de proporcionalidade que define essa relação é a viscosidade dinâmica μ. Assim, a tensão de cisalhamento τ nos fluidos, que pode ser interpretada como sendo a força tangencial por unidade de área, é definida em um plano xy pela equação 2.1, na qual $\frac{dV}{dy}$ é a variação da velocidade.

$$\tau = \mu \frac{dV}{dy} \tag{2.1}$$

Fluidos para os quais se aplicam a equação 2.1 são chamados de fluidos newtonianos. As viscosidades desses fluidos podem ser consideradas constantes e no presente trabalho só trataremos sobre esse tipo de fluido.

Outra grandeza importante para as análises de mecânica dos fluidos é a massa específica ou densidade ρ, que é uma propriedade definida pela quantidade de massa por unidade de volume. Sua definição algébrica pode ser verificada na equação 2.2, na qual δm é a quantidade de massa presente em um volume δv do fluido. Em fluidos incompressíveis, o valor de ρ é constante em todo o seu volume. Para as elaborações desse trabalho, apenas fluidos incompressíveis estão sendo considerados.

$$\rho = \frac{\delta m}{\delta v} \tag{2.2}$$

É possível relacionar a viscosidade dinâmica μ e a massa específica ρ para encontrar outro parâmetro da mecânica dos fluidos: a viscosidade cinemática ν que é definida pela equação 2.3.

$$\nu = \frac{\mu}{\rho} \tag{2.3}$$

Uma vez estabelecidas as definições desses importantes parâmetros, é possível definir o número de Reynolds (Re), que é um importante parâmetro de similaridade entre diferentes escoamentos e que facilita a comparação de escoamentos definidos por equações adimensionais, como é o caso das simulações numéricas do presente estudo. O número de Reynolds também é importante para caracterizar o escoamento em questão como laminar (para baixos valores de Re) ou turbulento (para altos valores de Re). No presente trabalho, todas as simulações realizadas foram feitas utilizando números de Reynolds baixos e, portanto, escoamentos turbulentos fogem do escopo desse texto. A definição do número de Reynolds pode ser encontrada na equação 2.4, na qual u é a velocidade do escoamento e L é o comprimento característico, que consiste em uma grandeza adimensional que, em poucas palavras, define a escala do sistema físico em questão.

$$Re = \frac{\rho u L}{\mu} = \frac{uL}{\nu} \tag{2.4}$$

As classificações de análises de problemas da mecânica dos fluidos não se restringem à compressibilidade ou grau de turbulência ou de liminaridade do escoamento.

Esses problemas podem ser encarados de um ponto de vista tridimensional, bidimensional ou até mesmo unidimensional. No presente trabalho, todas as elaborações são feitas tendo em vista um escoamento bidimensional em um plano xy.

Um conceito importante na análise de escoamentos bidimensionais é a função corrente ψ, por meio da qual é possível gerar as linhas de corrente do escoamento. Uma vez em que as linhas de corrente são sempre tangenciais à velocidade do escoamento, ao saber como se dispõem as linhas de corrente, já é possível adquirir muitas informações sobre o escoamento. Daí vem a importância da função corrente, cuja definição para escoamentos incompressíveis é dada na equação 2.5. Nessa equação, u e v são as velocidades do escoamento nas direções x e y, respectivamente.

$$d\psi = udy - vdx \tag{2.5}$$

Por meio da equação 2.5, é possível definir também as velocidades u e v em função da função corrente ψ:

$$u = \frac{\partial \psi}{\partial y} \tag{2.6}$$

$$v = -\frac{\partial \psi}{\partial x} \tag{2.7}$$

Outro parâmetro importante para os escoamentos deste trabalho é a vorticidade w_z. A vorticidade quantifica a rotação local do fluido em um ponto. Ela está diretamente relacionada com a velocidade angular de um elemento do fluido e, quando o valor encontrado para a vorticidade de um escoamento é zero, isso significa que o escoamento é irrotacional. Matematicamente, a vorticidade ω_z na direção z, perpendicular ao plano xy, pode ser definida como o rotacional da velocidade $\mathbf{V}$ do escoamento e, em um escoamento bidimensional, ela se relaciona com a função corrente ψ. Essas relações estão explicitadas nas equações 2.8 e 2.9.

$$\omega_z = \nabla \times \mathbf{V} \tag{2.8}$$

$$\omega_z = -\nabla^2 \psi \tag{2.9}$$

Equações de governo

Existem três equações fundamentais que são fundamentais para toda a mecânica dos fluidos: a equação da continuidade, a equação da conservação da quantidade de movimento e a equação da conservação de energia. As duas primeiras são as mais relevantes para o presente trabalho e serão descritas a seguir.

O princípio físico no qual a equação de continuidade é o de que a massa nunca pode ser criada ou destruída. Portanto, em um volume de controle, a taxa de massa que entra nesse volume tem que se manter igual a todo momento à taxa de massa que sai do volume de controle somada à quantidade de massa que permanece no volume de controle. A equação da continuidade pode ser expressa por 2.10.

$$\frac{\partial}{\partial t}\iiint_V \rho \, dV + \iint_S \rho \mathbf{V} \cdot \mathbf{dS} = 0 \tag{2.10}$$

Na equação 2.10, as variáveis do escoamento estão sendo definidas em um espaço finito. Na aplicação de métodos numéricos, pode ser mais conveniente expressar a equação da continuidade em seu formato de derivada parcial, que define as variáveis do escoamento como um ponto nesse mesmo escoamento. Essa interpretação diferente da equação de continuidade está expressa na equação 2.11.

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{V}) = 0 \tag{2.11}$$

Como no presente trabalho a massa específica é constante no tempo e no espaço, as equações 2.10 e 2.11 podem ser simplificadas e suas novas formas podem ser verificadas, respectivamente, nas equações 2.12 e 2.13.

$$\iint_S \rho \mathbf{V} \cdot \mathbf{dS} = 0 \tag{2.12}$$

$$\nabla \cdot (\rho \mathbf{V}) = 0 \tag{2.13}$$

A equação da conservação do momento, por sua vez, se baseia na segunda lei de Newton, que diz que a força é igual à taxa de mudança do momento. Essa lei está representada pela equação 2.14.

$$\mathbf{F} = \frac{d}{dt}(m\mathbf{V}) \tag{2.14}$$

Para um sistema infinitesimal de massa dm, podemos reescrever 2.14 como 2.15 utilizando o conceito de derivada material.

$$\mathbf{dF} = dm\frac{D\mathbf{V}}{Dt} = dm\left[u\frac{\partial \mathbf{V}}{\partial x} + v\frac{\partial \mathbf{V}}{\partial y} + w\frac{\partial \mathbf{V}}{\partial z} + \frac{\partial \mathbf{V}}{\partial t}\right] \tag{2.15}$$

Para que seja possível expandir o termo à esquerda da equação 2.15, é importante lembrar que as forças que atuam sobre um elemento do fluido podem ser forças de campo e forças de superfícies. As forças resultantes da ação dessas forças nas direções x, y e z são descritas, respectivamente, nas equações 2.16, 2.17, 2.18. Nessas equações, F_B são as forças de campo, F_S são as forças de superfície e σ_{xx}, σ_{yy}, σ_{zz}, τ_{xy}, τ_{xz}, τ_{yx}, τ_{yz}, τ_{zx} e τ_{zy} são as tensões que atuam no elemento fluido.

$$\mathbf{dF}_x = dF_{B_x} + dF_{S_x} = \left(\rho g_x + \frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \tau_{yx}}{\partial y} + \frac{\partial \tau_{zx}}{\partial z}\right) dxdydz \tag{2.16}$$

$$\mathbf{dF}_y = dF_{B_y} + dF_{S_y} = \left(\rho g_y + \frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \sigma_{yy}}{\partial y} + \frac{\partial \tau_{zy}}{\partial z}\right) dxdydz \tag{2.17}$$

$$\mathbf{dF}_z = dF_{B_z} + dF_{S_z} = \left(\rho g_z + \frac{\partial \tau_{xz}}{\partial x} + \frac{\partial \tau_{yz}}{\partial y} + \frac{\partial \sigma_{zz}}{\partial z}\right) dxdydz \tag{2.18}$$

Com essas três equações, podemos substituir o termo à esquerda da equação 2.15. No entanto, para representar fluidos newtonianos incompressíveis e de viscosidade constante, que são o objeto de estudo desse trabalho, é prático expressar as tensões em termos dos campos de velocidade e de pressão. Após adotar essas simplificações, obtemos uma das formulações das equações de Navier-Stokes, expressas nas equações 2.19, 2.20 e 2.21.

$$\rho\left(\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z}\right) = \rho g_x - \frac{\partial p}{\partial x} + \mu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right) \tag{2.19}$$

$$\rho\left(\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + w\frac{\partial v}{\partial z}\right) = \rho g_y - \frac{\partial p}{\partial y} + \mu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial z^2}\right) \tag{2.20}$$

$$\rho\left(\frac{\partial w}{\partial t} + u\frac{\partial w}{\partial x} + v\frac{\partial w}{\partial y} + w\frac{\partial w}{\partial z}\right) = \rho g_z - \frac{\partial p}{\partial z} + \mu\left(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} + \frac{\partial^2 w}{\partial z^2}\right) \tag{2.21}$$

Ao converter as equações 2.19, 2.20 e 2.21 para a notação vetorial, obtém-se a equação 2.22. Seus diferentes termos possuem significados físicos importantes: $\frac{\partial \mathbf{v}}{\partial t}$ compõe o termo transiente, $\mathbf{v} \cdot \nabla \mathbf{v}$ forma o termo convectivo, $-\frac{1}{\rho}\nabla p$ representa o gradiente de pressão, $\mathbf{g}$ consiste na ação da gravidade e $\nu \nabla^2 \mathbf{v}$ é o termo difusivo.

$$\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v} = -\frac{1}{\rho}\nabla p + \mathbf{g} + \nu \nabla^2 \mathbf{v} \tag{2.22}$$

A equação 2.22 já é muito útil por si só mas é possível modificar sua formulação de forma à tornar essa equação adimensional. Adimensionalizar uma equação consiste em remover as dimensões físicas de uma equação por meio da substituição de variáveis. Essa prática faz com que seja possível generalizar soluções, facilitando a comparação entre experimentos ou simulações feitos por diferentes laboratórios e pesquisadores.

O número de Reynolds já mencionado nesse tópico é uma das variáveis adimensionais, definida a partir de propriedades do fluido e do escoamento, que permite simplificar as equações adimensionais. A equação 2.23 é obtida ao adimensionalizar a equação 2.22. Em 2.23, os termos com asteriscos são versões adimensionalizadas de parâmetros que caracterizam um escoamento. Por exemplo, $\mathbf{v^*}$ é a velocidade adimensional e $\mathbf{p^*}$ é a pressão adimensional. Termos adimensionalizados não possuem unidades, o que dispensa conversões e facilita contas. Como neste trabalho os efeitos gravitacionais serão desconsiderados, o termo gravitacional foi retirado da equação 2.23.

$$\frac{\partial \mathbf{v}^*}{\partial t^*} + \mathbf{v}^* \cdot \nabla^* \mathbf{v}^* = -\frac{1}{\rho}\nabla^* p^* + \frac{1}{Re}\nabla^{*2}\mathbf{v}^* \tag{2.23}$$

2.3 Método numérico

No livro *"Computational Fluid Dynamics: The Basics with Applications"*, o autor e engenheiro John D. Anderson define a dinâmica dos fluidos computacional, em tradução livre, como "a arte de substituir as integrais ou as derivadas parciais (conforme for o caso) nessas equações por formas algébricas discretizadas, às quais são resolvidas para obter *números* para os valores de campo do escoamento em pontos discretos no tempo e/ou no espaço."(ANDERSON, 1995) As equações

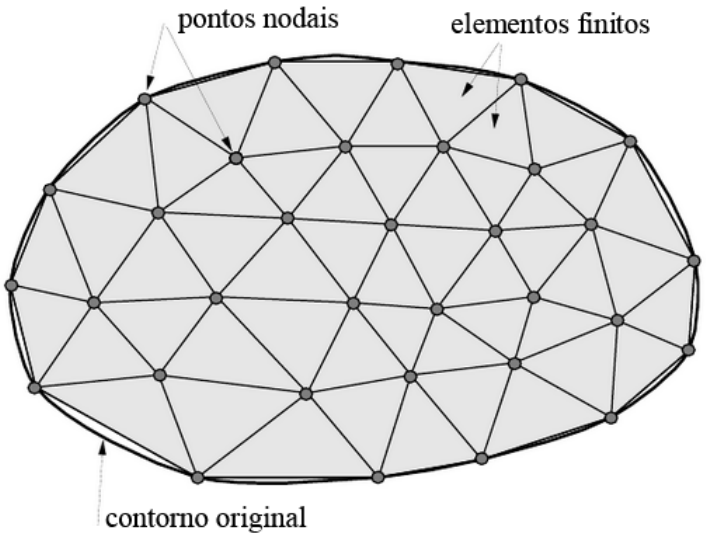

Figura 2.3: Desenho esquemático de como se é a discretização do domínio utilizando método de elementos finitos. Fonte: DE SOUZA (2003).

em questão são justamente as equações fundamentais que governam a mecânica dos fluidos, mencionadas no último tópico. Para realizar essa substituição que transforma as equações fundamentais em formas algébricas discretizadas, muitos métodos numéricos podem ser utilizados. No presente trabalho, foi utilizado o método de elementos finitos (MEF), que será brevemente introduzido.

O MEF permite que problemas complexos de diversas áreas da engenharia, incluindo a mecânica dos fluidos, consigam ser modelados e resolvidos mesmo quando não existe uma solução analítica. A essência desse poderoso método reside na discretização do objeto de estudo em inúmeros elementos ligados uns aos outros por nós. Esse conjunto de elementos e nós compõe uma malha que consiste em uma representação simplificada do objeto que se quer reproduzir.

Essa simplificação do domínio é uma das vantagens da adoção desse método numérico pois, ao invés de satisfazer as equações fundamentais da mecânica dos fluidos continuamente em todo um escoamento para conseguir uma resposta, só é necessário satisfazer essas equações em um número finito de pontos. No entanto, certos problemas exigem mais elementos do que outros para representar situações reais. Isso pode se tornar um desafio da aplicação do MEF já que, quanto mais elementos são adotados para representar um domínio, mais poder computacional é exigido.

Existem diferentes softwares que podem auxiliar na geração das malhas em que serão aplicados métodos numéricos como MEF. Assim, a partir de uma geometria fornecida ao software, os elementos finitos e seus nós são gerados. As relações de interconectividade entre os pontos das malhas são guardadas em uma matriz de conectividade IEN. Essa matriz relaciona os índices locais de cada elemento com os índices globais que representam o problema como um todo.

Essa relação de índices é essencial para montar a matriz de rigidez global do problema a partir das matrizes de rigidez locais de cada elemento e das condições de contorno. Uma forma de entender no que consiste o conceito de matriz de rigidez é relembrar a Lei de Hooke, que descreve o comportamento de muitos materiais elásticos. Ao observar a equação da Lei de Hooke em 2.24, é possível perceber que a matriz com as constantes elásticas $\mathbf{K}$ relaciona a matriz coluna com as forças aplicadas $\mathbf{F}$ aos deslocamentos provocados representados na matriz coluna $\mathbf{d}$.

$$\{\mathbf{F}\} = \{\mathbf{K}\}\{\mathbf{d}\} \tag{2.24}$$

De forma semelhante, a matriz de rigidez utilizada nesse trabalho para resolver as equações fundamentais da mecânica dos fluidos nas malhas trabalhadas relaciona as grandezas de interesse com os seus resultados por meio de um sistema de equações. Esse sistema pode, por sua vez, pode ser resolvido por métodos implícitos ou explícitos. Nesse trabalho, o método implícito será utilizado. Nele, utilizando como referência a equação 2.24, os deslocamentos são calculados por meio da inversão da matriz $\mathbf{K}$, como mostra a equação 2.25.

$$\{d_{t+\Delta t}\} = [K]^{-1}\{F_{t+\Delta t}\} \tag{2.25}$$

Antes de fazer essa conta, no entanto, é preciso definir as condições de contorno e a solução aproximada para os deslocamentos. Essa solução é encontrada por meio do que é chamado forma fraca (ou variacional) da equação que governa a simulação estudada. Essa formulação fraca é encontrada a partir da equação de governo original (formulação forte) ao ponderar ela por uma função peso e integrá-la sobre um domínio. A partir desse ponto, é possível fazer simplificações que permitem encontrar soluções aproximadas para o sistema de equações lineares obtido. No contexto desse trabalho, a equação de governo em questão é a formulação corrente-

vorticidade, que será descrita no próximo tópico e terá sua forma fraca deduzida.

2.4 Formulação Corrente-Vorticidade

2.4.1 Dedução a partir das equações de Navier-Stokes

A formulação corrente-vorticidade é uma forma de expressar as equações de Navier-Stokes, representadas em sua forma vetorial no tópico 2.2, em termos de função corrente ψ e vorticidade ω ao invés da velocidade e pressão. Essa função alternativa pode ser encontrada por meio de uma reformulação das equações de continuidade e de momento em fluidos, descritas também em 2.2. Expressar as equações de Navier-Stokes dessa forma é um caminho eficiente para evitar o acoplamento entre a velocidade e a pressão como incógnitas.

Diferentes estudos adotaram a formulação corrente-vorticidade com sucesso como DA CUNHA (2020), MOHAMED ABDELWAHED (2017) e CARNEVALE *et al.* (2018). Seguindo esse legado promissor, o presente trabalho também se propõe a utilizar essa formulação para analisar os escoamentos propostos. Assim, é conveniente realizar uma revisão de como a formulação corrente-vorticidade pode ser encontrada a partir da equação 2.22. A linha de raciocínio a seguir é baseada na dedução apresentada por DE CERQUEIRA (2022) e ANJOS.

Na seção 2.2, a vorticidade foi definida como sendo o rotacional da velocidade. Assim, é possível obter a equação da vorticidade aplicando o operador rotacional $(\nabla\times)$ na equação 2.22:

$$\nabla \times \left(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v} \right) = \nabla \times \left(-\frac{1}{\rho} \nabla p + \mathbf{g} + \nu \nabla^2 \mathbf{v} \right) \tag{2.26}$$

No entanto, na equação 2.26 o termo $\mathbf{v} \cdot \nabla \mathbf{v}$ é equivalente a $\nabla \frac{v^2}{2} - \mathbf{v} \times \nabla \times \mathbf{v}$ e pode substituido em 2.26:

$$\nabla \times \left(\frac{\partial \mathbf{v}}{\partial t} + \nabla \frac{v^2}{2} - \mathbf{v} \times \nabla \times \mathbf{v} \right) = \nabla \times \left(-\frac{1}{\rho} \nabla p + \mathbf{g} + \nu \nabla^2 \mathbf{v} \right) \tag{2.27}$$

Ao expandir a equação 2.27 de ambos os lados, encontra-se:

$$\frac{\partial \left(\nabla \times \mathbf{v} \right)}{\partial t} + \nabla \times \nabla \frac{v^2}{2} - \nabla \times (\mathbf{v} \times (\nabla \times \mathbf{v})) = -\frac{1}{\rho} \nabla \times \nabla p + \nabla \times \mathbf{g} + \nu \nabla^2 \left(\nabla \times \mathbf{v} \right) \tag{2.28}$$

Uma vez que $\nabla \times \nabla = 0$ e definindo $v^2 = \mathbf{v} \cdot \mathbf{v}$, é possível reduzir a equação 2.28 para:

$$\frac{\partial (\nabla \times \mathbf{v})}{\partial t} - \nabla \times (\mathbf{v} \times (\nabla \times \mathbf{v})) = \nabla \times \mathbf{g} + \nu \nabla^2 (\nabla \times \mathbf{v}) \tag{2.29}$$

Relembrando que a vorticidade ω equivale ao rotacional da velocidade $\nabla \times \mathbf{v}$, obtém-se uma equação mais reduzida:

$$\frac{\partial \omega}{\partial t} + \mathbf{v} \cdot \nabla \omega - \omega \cdot \nabla \omega \mathbf{v} = \nabla \times \mathbf{g} + \nu \nabla^2 \omega \tag{2.30}$$

Como nesse trabalho, todos os problemas simulados serão bidimensionais, é possível anular termo $\omega \cdot \nabla \mathbf{v}$ do lado esquerdo da equação 2.30. Isso ocorre pois a vorticidade bidimensional ω_z será sempre perpendicular a $\nabla \mathbf{v}$. Além disso, as forças gravitacionais serão desprezadas nas análises desse trabalho e, portanto, o termo $\nabla \times \mathbf{g}$ também poderá ser anulado. Adotando essas modificações, a equação 2.30 pode ser simplificada para a seguinte forma:

$$\frac{\partial \omega_z}{\partial t} + \mathbf{v} \cdot \nabla \omega_z = \nu \nabla^2 \omega_z \tag{2.31}$$

A equação 2.31 é a equação de transporte da vorticidade para problemas bidimensionais e é uma das equações que compõem a formulação corrente-vorticidade. Para obter as demais equações, é necessário relembrar das equações 2.6 e 2.7. Para um problema bidimensional, podemos interpretar o componente da velocidade do escoamento u como sendo equivalente ao componente da velocidade do escoamento em relação ao eixo x. Da mesma forma, v pode ser considerado equivalente ao componente da velocidade do escoamento em relação ao eixo y. Reformulando as equações 2.6 e 2.7 com essa notação obtém-se:

$$v_x = \frac{\partial \psi}{\partial y} \tag{2.32}$$

$$v_y = -\frac{\partial \psi}{\partial x} \tag{2.33}$$

Tendo essas equações em mente, podemos relembrar a definição de vorticidade dada pela equação 2.8 e reescrevê-la explicitando a presença os termos v_x e v_y. Novamente, é importante lembrar que o problema sendo tratado é bidimensional.

$$\omega_z = \nabla \times \mathbf{v} = \frac{\partial v_y}{\partial x} - \frac{\partial v_x}{\partial y} \tag{2.34}$$

Ao substituir as equações 2.32 e 2.33 em 2.34, encontra-se enfim a equação de Poisson para a função corrente ψ:

$$\nabla^2 \psi = -\omega_z \tag{2.35}$$

O conjunto composto pelas equações 2.31, 2.32, 2.33 e 2.35 forma o sistema de equações para a solução de escoamentos utilizando a formulação de corrente-vorticidade. A solução numérica pode ser obtida ao discretizar essas equações utilizando o método de elementos finitos.

2.4.2 Adaptação para MEF

Como foi falado no tópico 2.3, um passo muito importante para a aplicação do método numérico de elementos finitos (MEF) é a transformação da equação de interesse original (forma forte) em uma nova apresentação que permita uma solução mais direta por meio de um sistema linear. Essa nova apresentação da equação de forma forte é chamada de forma fraca. No presente tópico, as equações 2.31, 2.32, 2.33,2.35 e 2.34 terão sua forma fraca apresentada seguindo o mesmo método adotado por DE CERQUEIRA (2022). A dedução de todas as formas fracas desse trabalho se baseia no Método de Galerkin, a linha de raciocínio para a aplicação desse método nas equações em questão é apresentada no decorrer dos próximos três tópicos.

Equação de Transporte de vorticidade

Para iniciar o processo da dedução da forma fraca da equação de transporte de vorticidade, o primeiro passo consiste em ponderar a equação 2.31 com uma função peso $w(x, y)$. Uma vez ponderada, a equação resultante deve ser integrada no domínio de estudo Ω:

$$\int_\Omega w \left(\frac{\partial \omega_z}{\partial t} + \mathbf{v} \cdot \nabla \omega_z - \nu \nabla^2 \omega_z \right) d\Omega = 0 \tag{2.36}$$

A equação 2.36 pode ser, então, expandida distribuindo a função peso w e separando as integrais:

$$\int_{\Omega} w\frac{\partial \omega_z}{\partial t}d\Omega + \int_{\Omega} w\mathbf{v}\cdot\nabla\omega_z d\Omega - \int_{\Omega} w\nu\nabla^2\omega_z d\Omega = 0 \tag{2.37}$$

Para encontrar a forma fraca, a ordem da equação 2.37 precisa ser reduzida. Isso pode ser feito utilizando a identidade de Green, por meio da qual é possível integrar por partes o termo com laplaciano de segunda ordem. Assim, uma nova equação é obtida:

$$\int_{\Omega} w\frac{\partial \omega_z}{\partial t}d\Omega + \int_{\Omega} w\mathbf{v}\cdot\nabla\omega_z d\Omega - \int_{\Gamma} w\nu\nabla\omega_z d\Omega + \int_{\Omega} \nabla w\cdot\nu\nabla\omega_z d\Omega = 0 \tag{2.38}$$

Na equação 2.38, Γ é a região de contorno do domínio Ω. A próxima etapa consiste na definição da função vorticidade $(\omega_z(x,y))$ e da função peso $(w(x,y))$ como combinações lineares nas quais N_i e N_j são funções de forma (ou de interpolação) e a_i e b_j são constantes ainda não determinadas:

$$\omega_z(x,y) = \sum_{i=0}^{\infty} N_i(x,y)a_i \tag{2.39}$$

$$w(x,y) = \sum_{j=0}^{\infty} N_j(x,y)b_j \tag{2.40}$$

Como as equações 2.39 e 2.40 só precisam ser aplicadas ao numéro total de pontos ou vértices da malha que descreve o problema em questão, é possível realizar uma aproximação que aplique o somatório somente aos pontos n da malha:

$$\omega_z(x,y) \approx \sum_{i=0}^{n} N_i(x,y)a_i \tag{2.41}$$

$$w(x,y) \approx \sum_{j=0}^{n} N_j(x,y)b_j \tag{2.42}$$

Seria possível já aplicar as aproximações representadas pelas equações 2.41 e 2.42 na equação 2.38. No entanto, ao utilizar o método de Galerkin, as funções de forma N_i e N_j são consideradas iguais e mais simplificações são possíveis por meio dessa igualdade. Como $N_i(x,y) = N_j(x,y)$, podemos substituir os somatórios $\sum_i$

e $\sum_j$ por um novo somatório $\sum_e$ que se aplica a todos os elementos e da malha que descreve a geometria do problema. Depois dessa substituição, podemos dividir a equação 2.38 por w e retirar as constantes ω_z de dentro das integrais:

$$\begin{aligned} \sum_e \frac{\partial \omega_{z_i}}{\partial t} \int_\Omega N_i N_j d\Omega + \sum_e \omega_{z_i} \int_\Omega \mathbf{v} \cdot (N_j \nabla N_i)\, d\Omega \\ - \sum_e \omega_{z_i} \int_\Gamma N_j \left(\nu \nabla N_i\right) d\Gamma + \sum_e \omega_{z_i} \int_\Omega \nabla N_j \cdot \left(\nu \nabla N_i\right) d\Omega = 0 \end{aligned} \tag{2.43}$$

A partir da equação 2.43, é possível aproximar o termo $\frac{\partial \omega_z}{\partial t}$ utilizando o conceito de diferenças finitas progressivas, que calcula o termo atual tendo como referência o valor do tempo posterior. Com essa aproximação, encontra-se uma nova equação:

$$\begin{aligned} \sum_e \frac{\omega_{z_i}^{n+1} - \omega_{z_i}^{n}}{\Delta t} \int_\Omega N_i N_j d\Omega + \sum_e \omega_{z_i} \int_\Omega \mathbf{v} \cdot (N_j \nabla N_i)\, d\Omega \\ - \sum_e \omega_{z_i} \int_\Gamma N_j \left(\nu \nabla N_i\right) d\Gamma + \sum_e \omega_{z_i} \int_\Omega \nabla N_j \cdot \left(\nu \nabla N_i\right) d\Omega = 0 \end{aligned} \tag{2.44}$$

Alguns dos termos da equação 2.44 podem ser representados como matrizes. São eles:

$$\int_\Omega N_i N_j d\Omega = \mathbf{M} \tag{2.45}$$

$$\int_\Omega N_j \nabla N_i d\Omega = \mathbf{G} \tag{2.46}$$

$$\int_\Omega \nabla N_j \cdot \nabla N_i d\Omega = \mathbf{K} \tag{2.47}$$

A matriz $\mathbf{M}$ é chamada de matriz de massa, a matriz $\mathbf{G}$ é conhecida como matriz de gradiente e a matriz $\mathbf{K}$ é a matriz de rigidez ou que representa o termo viscoso. O conceito de matriz de rigidez já foi introduzido em 2.3.

Como nesse trabalho estão sendo utilizadas condições de contorno de Dirichlet em que $w_j = 0$ e a velocidade $\mathbf{v}$ e viscosidade cinemática ν estão sendo consideradas como constantes. Para cada elemento e, podemos expressar a equação 2.44 como:

$$m_{ij}^{e}\left(\frac{\omega_{z_i}^{n+1}-\omega_{z_i}^{n}}{\Delta t}\right)+\mathbf{v}\cdot g_{ij}^{e}\omega_{z_i}+\nu k_{ij}^{e}\omega_{z_i}=0 \tag{2.48}$$

A equação 2.48 consiste numa forma fraca local da equação de transporte da vorticidade. Ela se aplica em cada elemento e da malha em questão e os índices i e j variam conforme os vértices ou nós de cada elemento. Ao relacionar os valores de m, g e k locais de cada elemento uns com os outros com o auxílio da IEN, encontra-se a equação global da forma fraca do transporte de vorticidade:

$$\mathbf{M}_{ij}\left(\frac{\omega_{z_i}^{n+1}-\omega_{z_i}^{n}}{\Delta t}\right)+\mathbf{v}\cdot\mathbf{G}_{ij}\omega_{z_i}+\nu\mathbf{K}_{ij}\omega_{z_i}=0 \tag{2.49}$$

Equação de Poisson para função corrente

A linha de raciocínio para dedução da formulação fraca da equação de Poisson para a função corrente é análoga à da equação de transporte de vorticidade explicitada no tópico anterior. O ponto de partida é a equação 2.35. Novamente, essa equação será multiplicada por uma função peso $w(x, y$ e será integrada em um domínio Ω:

$$\int_{\Omega} w\left(\nabla^2\psi\right)+\int_{\Omega} w\omega_z d\Omega=0 \tag{2.50}$$

Para reduzir a ordem do termo contendo o laplaciano ∇^2, novamente é aplicada a identidade de Green. Na nova equação obtida, Γ é o contorno do domínio Ω:

$$\int_{\Gamma} w\left(\nabla\psi\right)d\Gamma-\int_{\Omega}\nabla w\cdot\nabla\psi d\Omega+\int_{\Omega} w\omega_z d\Omega=0 \tag{2.51}$$

Os termos que representam a função corrente $\psi(x, y)$, a vorticidade $\omega_z(x, y)$ e a função peso $w(x, y)$ podem ser representados como combinações lineares que se aplicam para cada elemento n da malha:

$$\psi(x,y)\approx\sum_{i=0}^{n}N_i(x,y)a_i \tag{2.52}$$

$$\omega_z(x,y)\approx\sum_{i=0}^{n}N_i(x,y)b_i \tag{2.53}$$

$$w(x,y) \approx \sum_{j=0}^{n} N_j(x,y) c_j \tag{2.54}$$

Como o método de Galerkin está sendo adotado para encontrar a forma fraca, considera-se que $\sum_j$ e $\sum_i$ podem ser substituídos por um somatório de todos os elementos da malha $\sum_e$. Além disso, ao aplicar essa consideração na equação 2.51, também retiramos da integral a vorticidade ω_z e a função corrente ψ, que são constantes:

$$\sum_{e\Gamma} \psi_i w_j \int \Gamma N_j \left(\nabla N_i\right) d\Gamma - \sum_{e} \psi_i w_j \int_{\Omega} \nabla N_j \cdot \nabla N_i d\Omega + \sum_{e} \omega_z w_j \int_{\Omega} N_j N_i d\Omega = 0 \tag{2.55}$$

A equação 2.55 pode ser dividida por w_j e, uma vez que a condição de contorno de Dirichlet $w_j = 0$ está sendo considerada, a equação da forma fraca para a função corrente em cada elemento e se torna:

$$-k_{ij}^{e} \psi_i + m_{ij}^{e} \omega_{z_i} = 0 \tag{2.56}$$

Para representar o problema global, onde os coeficientes i e j referem-se a elementos da malha, se obtém:

$$\mathbf{K}_{ij} \psi_i = \mathbf{M}_{ij} \omega_{z_i} \tag{2.57}$$

Campos de velocidade

Para deduzir a forma fraca que representa os campos de velocidade de um escoamento expresso segundo a formulação de corrente-vorticidade, um processo semelhante ao descrito nos últimos dois tópicos será executado. O primeiro passo consiste em multiplicar por uma função peso $w(x,y)$ as equações 2.32 e 2.33 e integrá-las em um domínio Ω.

$$\int_{\Omega} w \left(v_x - \frac{\partial \psi}{\partial y} \right) d\Omega \tag{2.58}$$

$$\int_{\Omega} w \left(v_y + \frac{\partial \psi}{\partial x} \right) d\Omega \tag{2.59}$$

As equações 2.58 e 2.59 são o resultado das manipulações feitas, respectivamente nas equações 2.32 e 2.33. A partir desse desenvolvimento, duas equações globais que descrevem a forma fraca do campo de velocidade são encontradas com novas matrizes de peso **G** e de rigidez **K**:

$$\mathbf{M}_{ij} v_{x_i} = \mathbf{G}_{y_{ij}} \psi_i \tag{2.60}$$

$$\mathbf{M}_{ij} v_{y_i} = -\mathbf{G}_{x_{ij}} \psi_i \tag{2.61}$$

Equação da Vorticidade 2D

A última equação cuja forma fraca terá que ser deduzida para descrever o problema abordado nesse trabalho e o código que foi implementado na realização dele é a equação da vorticidade aplicada em um espaço bidimensional. Portanto, essa equação, representada em 2.34, também será multiplicada por uma função peso e será integrada em um domínio Ω:

$$\int_{\Omega} w \left(\omega_z - \frac{\partial v_y}{\partial x} + \frac{\partial v_x}{\partial y} \right) d\Omega = 0 \tag{2.62}$$

Após procedimentos análogos aos executados nos tópicos anteriores, a matriz global obtida a partir de 2.62 será:

$$\mathbf{M}_{ij} \omega_{z_i} = \mathbf{G}_{x_{ij}} v_{y_i} - \mathbf{G}_{y_{ij}} v_{x_i} \tag{2.63}$$

Portanto, reunindo todas as formas fracas que serão utilizadas por meio do método de elementos finitos para calcular os componentes v_x e v_y da velocidade, a vorticidade ω_z e a função corrente ψ em cada ponto das malhas criadas segundo as geometrias de interesse desse trabalho para cada iteração programada, obtém-se a seguinte lista. Nessa lista os itens 3 e 4 são equações auxiliares as duas primeiras.

1. Equação de transporte de vorticidade

$$\left(\frac{M}{\Delta t} + \nu K + \boldsymbol{v} \cdot G \right) {\omega_z}^{n+1} = \frac{M}{\Delta t} {\omega_z}^{n} + c.c \tag{2.64}$$

2. Função corrente

$$K\psi = M\omega_z + c.c \tag{2.65}$$

3. Campos de velocidade v_x e v_y

$$Mv_x = G_y\psi \quad \text{e} \quad Mv_y = -G_x\psi \tag{2.66}$$

4. Vorticidade ω_z

$$M\omega_z = G_x v_x - G - yv_x \tag{2.67}$$

Na lista acima, K, M e G são matrizes que representam, respectivamente, o termo viscoso, a massa e o gradiente. O termo c.c. representa as condições de contorno para solução das equações de transporte de vorticidade e da função corrente.

Capítulo 3

Metodologia Proposta

Após realizar uma revisão sobre diversos conceitos e equações importantes para a elaboração desse trabalho, chega o momento de retornar ao objeto de estudo: filtros de exaustão da queima de biocombustíveis. Para estudá-los com as ferramentas descritas no último capítulo, serão feitas algumas simplificações. Como é possível verificar pela figura 2.1, um filtro de exaustão possuí inúmeros canais bem pequenos. Simular todos esses canais simultaneamente seria um grande desafio que exigiria muitos recursos computacionais. Por isso, ao invés de simular a estrutura de um filtro inteiro, o presente trabalho se propõe a realizar simulações de um único canal do filtro de partículas.

Além disso, para trabalhar com equações mais simples, o canal do filtro será simulado em somente duas dimensões. Como base para a resolução do escoamento bidimensional, serão usadas as equações de governo desenvolvidas no capítulo anterior. Outra simplificação que terá de ser feita é que o código numérico desenvolvido não consegue encontrar resultados convergentes para altos números de Reynolds. Por isso, todas as simulações serão feitas com valores bem baixos de número de Reynolds. No entanto, essa limitação não é necessariamente fora da realidade. Afinal, o escoamento em um filtro de partículas envolve passagens sucessivas por inúmeros filtros que provocam diminuição da pressão e, portanto, números de Reynolds mais baixos para esses escoamentos.

Quanto à abordagem para construir esse problema de dinâmica dos fluidos de forma computacional, foi adotada uma abordagem euleriana, na qual a malha que discretiza o canal fica estacionária no espaço. A utilização dessa abordagem é

propícia para o caso estudado já que o fluido é considerado incompressível e o escoamento é contínuo.

A intenção desse trabalho, no entanto, é estudar diferentes geometrias de filtros de partículas e seus canais. Os canais simulados terão uma região porosa em sua região central e a variação da geometria se dará na região porosa dos canais. Essa porosidade será emulada geometricamente por meio de obstáculos sólidos que serão colocados em cada canal simulado. Portanto, cada geometria simulada é diferente na medida em que as distribuições e números de obstáculos mudaram para cada canal.

Uma vez estabelecidas essas premissas inicias, é possível seguir para a descrição de como se realizaram as simulações numéricas desse trabalho. Cada simulação envolveu, essencialmente, três etapas: geração de geometria e malha associada, cálculo numérico das grandezas de interesse para essa geometria utilizando o método de elementos finitos, visualização dos resultados e pós processamento de dados. Cada uma dessas etapas será descrita a seguir.

3.1 Geração de geometrias e malhas

Todas as malhas geradas para esse trabalho foram montadas com o auxílio do software **Gmsh** em sua versão 3.0.6. Esse software é livre, de código aberto e fornece uma interface amigável para a criação de geometrias e malhas a partir dessas geometrias. Além disso, o **Gmsh** possibilita a customização de geometrias por meio de *scripts* associados aos seus arquivos .geo. Assim, ao invés de construir manualmente cada geometria seguindo as ferramentas da interface do programa, o usuário possui a possibilidade de definir sua geometria alterando os *scripts* associados ao seu arquivo contendo a geometria.

Esses *scripts* são escritos em uma linguagem de programação própria do **Gmsh** e em sua versão 3.0.6 os recursos que essa linguagem permite são limitados. No entanto, uma vez que essa linguagem é compreendida, é possível utilizar outras linguagens de programação, como **Python**, para programar funções que escrevam o *script* para a geometria desejada. Essa abordagem permite que as limitações da linguagem do **Gmsh** sejam superadas e concede mais controle sobre a geração da

geometria e sua malha.

Como as geometrias de interesse neste trabalho são relativamente complexas, optou-se por escrever um código em **Python** para criar o *script* de cada cada tipo de geometria que seria simulado. Além de fornecerem o *script*, cada um desses códigos foi programado para fornecer informações de interesse como as condições de contorno para cada um dos obstáculos gerados, coordenadas do centro do obstáculo e seus raios (já que todos os obstáculos simulados são circulares). Esses códigos podem ser consultados nos apêndices A, B e C deste documento e imagens das geometrias e malhas retornadas a partir dos *scripts* gerados por eles podem ser vistas abaixo:

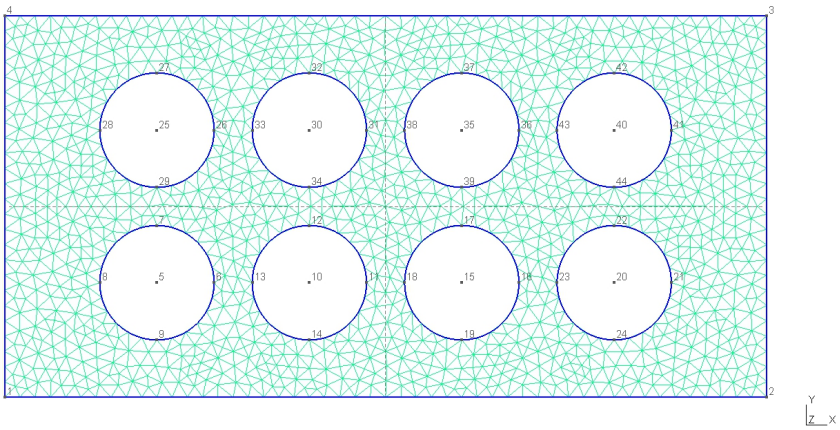

Figura 3.1: Geometria e malha geradas utilizando o software **Gmsh** para o teste com obstáculos circulares alinhados. Nessa geometria específica o diâmetro de todos os obstáculos circulares é igual a 3 unidades de comprimento. A altura e o comprimento do canal são iguais a 10 e 20 unidades de comprimento, respectivamente. A malha dessa geometria é composta por 3148 elementos. Fonte: Elaboração própria.

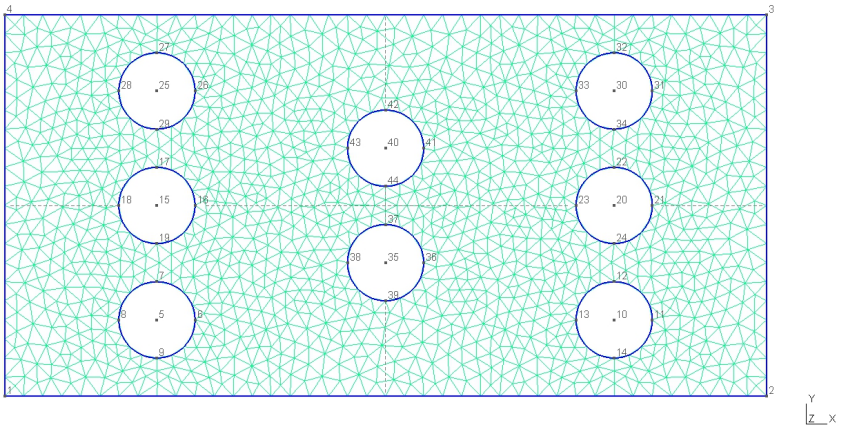

Figura 3.2: Geometria e malha geradas utilizando o software **Gmsh** para o teste com obstáculos circulares desalinhados. Nessa geometria específica o diâmetro de todos os obstáculos circulares é igual a 2 unidades de comprimento. A altura e o comprimento do canal são iguais a 10 e 20 unidades de comprimento, respectivamente. A malha dessa geometria é composta por 2984 elementos. Fonte: Elaboração própria.

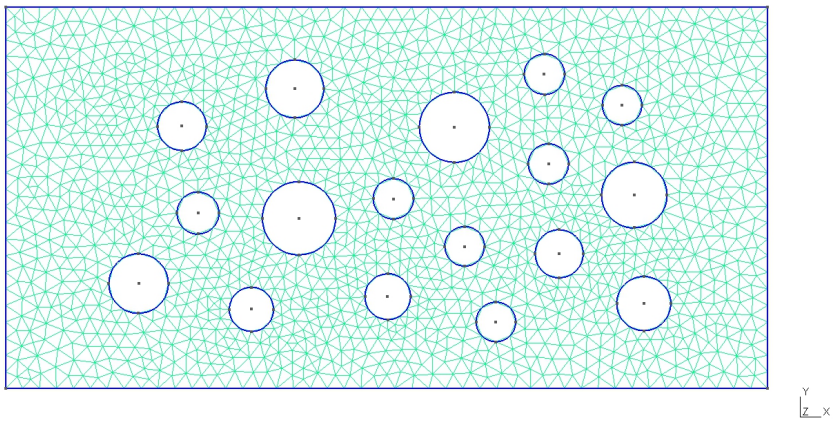

Figura 3.3: Geometria e malha geradas utilizando o software **Gmsh** para o teste com obstáculos circulares dispostos em posições aleatórias com valores de raios aleatórios. Nessa geometria específica o raio de cada obstáculo é sorteado de uma distribuição uniforme entre os valores 1 e 2. O valor médio dos raios para essa simulação foi 0,655. A altura e o comprimento do canal são iguais a 10 e 20 unidades de comprimento, respectivamente. A malha dessa geometria é composta por 3001 elementos. Fonte: Elaboração própria.

As imagens acima representam cada tipo de geometria que foi criada com o auxílio de um código em **Python**. Em resumo, três formas de emular a porosidade dos filtros de partículas foram experimentadas: obstáculos circulares de mesmo raio alinhados, obstáculos circulares de mesmo raio em que cada linha está desalinhada em relação à sua vizinha e obstáculos circulares de raios diferentes cujas posições centrais são aleatórias.

O terceiro tipo de geometria simulado, no qual os obstáculos circulares tomam posições aleatórias na geometria do canal, foi idealizado na intenção de tentar emular o aspecto orgânico e a disposição aleatória que o material filtrante dos filtros de partículas possuem, como pode ser visto na figura 2.2. No entanto, ao trabalhar com posições aleatórias para os obstáculos circulares, alguns desafios surgiram. Foi preciso estabelecer limites para as posições centrais dos obstáculos, considerando os valores de seus raios por exemplo, para que os obstáculos não cruzassem uns aos outros nem a geometria do canal.

Além disso, para que a forma adimensional das equações de governo pudessem

ser utilizadas, foi preciso estabelecer que o comprimento característico fosse igual a uma unidade de comprimento. Implementar essa condição é vantajoso pois reduz os parâmetros variáveis que descrevem o escoamento mas, especialmente no caso da geometria formada por obstáculos circulares com posições aleatórias, foi um grande desafio. Além de garantir que os obstáculos não cruzassem uns com os outros e com o contorno do canal, foi necessário assegurar que as bordas dos obstáculos mantivessem uma distância de no mínimo uma unidade de comprimento entre eles e em relação à borda do canal para que o comprimento característico tivesse um valor compatível com a forma adimensionalizada das equações de governo.

O comprimento característico, que faz parte da equação 2.4 do número de Reynolds, ficou definido como sendo a menor distância entre a borda de dois obstáculos. Essa distância foi estabelecida em todas as simulações como sendo igual a uma unidade de comprimento e, desse modo, o número de Reynolds passa a depender somente do valor definido para a viscosidade dinâmica e do valor escolhido para a velocidade inicial do escoamento. Os parâmetros escolhidos para as simulações podem ser consultados no capítulo 5.

Assim, uma vez que as geometrias dos canais foram criadas a partir da programação dos códigos desenvolvidos em **Python**, foi possível criar as malha e as matrizes de conectividade IEN no **Gmsh**. Essas malhas consistem em arquivos .msh que foram em seguida importadas pelo código numérico escrito também em **Python**, descrito no próximo tópico.

3.2 Código numérico

Para calcular as grandezas de interesse em cada nó da malha descrita no tópico anterior em todos os intervalos escolhido, foi elaborado um código numérico escrito em **Python** na sua versão 3.11 com a utilização dos pacotes *numpy* e *meshio*. Esse código pode ser consultado na íntegra no apêndice D. No entanto, é importante ressaltar que para cada geometria o código presente no apêndice D teve que ser alterado segundo a quantidade de obstáculos e as condições de contorno dos mesmos.

A execução do código numérico desenvolvido pode ser dividida em quatro etapas principais: definição dos parâmetros essenciais para o escoamento e leitura da ma-

lha, montagem das matrizes globais do sistema (deduzidas no capítulo 2), inversão de matrizes e resolução do sistema de equações e, finalmente, exportação de resultados no formato de arquivos .vtk, que serão analisados na etapa de visualização de resultados

Com a formulação corrente-vorticidade, após a execução do código é possível obter os valores do campo de velocidades, da função corrente e da vorticidade em cada ponto das malhas. No entanto, ao adotar essa formulação não é possível obter os valores das pressões. Isso gera algumas limitações como, por exemplo, não ser possível calcular a perda de carga do escoamento. No entanto, como as perdas de carga não estão sendo estudadas nesse trabalho, a utilização da formulação corrente-vorticidade continua sendo satisfatória para os fins do mesmo.

Duas outras limitações desse código numérico devem ser ressaltadas. No computador utilizado para essas simulações cujas especificações estão expostas no próximo capítulo, não ser possível rodar esse código importando malhas com número de elementos muito superiores a 3000. Essa limitação não pôde ser contornada. Além disso, não foi possível encontrar resultados convergentes para números de Reynolds muito altos. O funcionamento desse código numérico se dá somente para baixos números de Reynolds. Isso, no entanto, não é uma grande limitação uma vez que nas hipóteses utilizadas para descrever o problema abordado um baixo número de Reynolds já está sendo considerado. De fato, está sendo considerado que a passagem sucessiva por filtros diminui a pressão e a velocidade do escoamento em filtros de partículas, abaixando o número de Reynolds desses escoamentos.

3.3 Visualização de resultados

Para a visualização dos arquivos .vtk gerados na etapa de execução do código numérico descrito no tópico anterior, o software **Paraview** foi escolhido. Ele é um software livre, de código aberto e que permite muitos recursos de visualização e pós processamento de resultados. É por meio das ferramentas do **Paraview** que as imagens das simulações foram criadas e dados de interesse foram extraídos, como as velocidades média de entrada e saída do canal.

3.4 Qualidade dos filtros

De posse dos resultados das simulações, a qualidade de cada geometria é então avaliada. Para este trabalho, se escolheu definir um fator de qualidade α para a filtragem de cada geometria. A forma mais intuitiva de medir a qualidade de um filtro de partículas seria analisando a quantidade de partículas retidas no filtro. Porém, como não foi realizada simulação de partículas para este trabalho, a definição do fator de qualidade de cada canal filtrante será realizada de outra forma.

Por exemplo, é possível criar um novo fator de qualidade partindo do entendimento de que para que as partículas do escoamento sejam retidas e parem em alguma superfície da geometria do filtro o escoamento precisa desacelerar. É intuitivo pensar que em escoamentos entre paredes planas, como são os canais das simulações deste trabalho, uma diminuição na velocidade horizontal do escoamento represente uma desaceleração desejada, pois representaria mais partículas sendo retidas nas superfícies das paredes ou dos obstáculos. Quanto à velocidade no eixo vertical do escoamento, tanto sua diminuição quanto aumento podem ser considerados desejáveis, uma vez que podem significar, respectivamente, desaceleração das partículas e aceleração das partículas em direção às paredes do filtro.

Portanto, a partir dessa linha de pensamento o fator de qualidade α é definido para este trabalho. Esse fator α consistirá em uma razão entre a velocidade média do escoamento horizontal na entrada do canal e a velocidade média do escoamento horizontal na saída do canal. Ou seja, a partir de uma velocidade de entrada fixa, quanto maior a velocidade média do escoamento na saída do canal, menor será o valor encontrado para α e vice-versa. A fórmula desse fator de qualidade pode ser consultada na equação 3.1.

$$\alpha = \frac{v_x \text{ médio na entrada do canal}}{v_x \text{ médio na saída do canal}} \tag{3.1}$$

Como foi dito no tópico 3.1, três tipos de geometrias foram simuladas. Cada uma delas se baseia em um tipo de organização diferente para a disposição dos obstáculos que emulam a porosidade do filtro. Para cada tipo de organização geométrica dos obstáculos, três simulações foram realizadas, cada uma definindo obstáculos que ocupam uma área menor ou maior do canal. Ou seja, cada simulação tem obstáculos que ocupam uma área maior ou menor em relação à área total do canal. Torna-se

conveniente então definir essa relação como um outro parâmetro que caracterize as simulações. Esse parâmetro se chamará *densidade dos obstáculos* e será identificado pela letra grega θ. A densidade dos obstáculos consistirá na razão entre a área ocupada pelos obstáculos na geometria do canal e a área total do canal, que é fixa. A equação 3.2 explicita essa relação.

$$\theta = \frac{\text{Área ocupada pelos obstáculos}}{\text{Área do canal}} \tag{3.2}$$

Com esses dois parâmetros definidos, será possível relacioná-los em uma função para cada tipo de geometria simulada e as geometrias de filtro mais eficientes serão identificadas.

Capítulo 4

Verificação do código numérico

As simulações para verificação do código numérico foram realizadas no computador pessoal da autora. O modelo desse computador é um notebook Dell Inspiron 5570, com processador Intel Core i7-8550U 1.99 GHz, 4 Core, 8 Threads e 16GB de RAM. O sistema operacional utilizado é Microsoft Windows 11.

4.1 Escoamento entre placas planas

Um problema muito conhecido em mecânica dos fluidos e de fácil replicação é o escoamento entre placas planas ou escoamento de Poiseuille. Esse problema possui solução analítica e ao comparar essa solução com a numérica encontrada será possível analisar se o código numérico desenvolvido pode ser verificado. Verificações de código numérico tendo como base esse problema já foram realizadas com sucesso por DA CUNHA (2020), SPESANI (2023) e DE CERQUEIRA (2022).

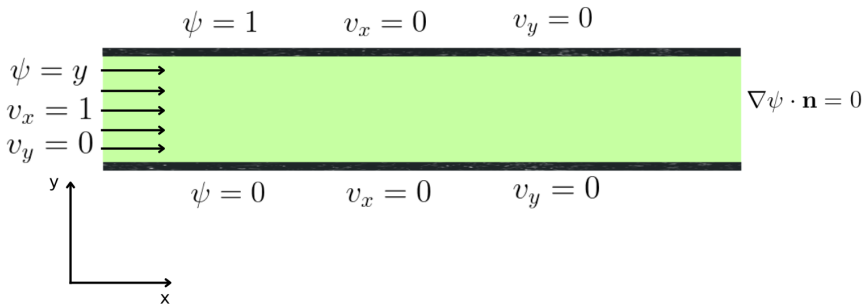

Figura 4.1: Modelo esquemático contendo as condições de contorno consideradas para a simulação de escoamento entre placas planas. As dimensões do canal são 1×6. Fonte: elaboração própria.

Um desenho esquemático desse tipo de escoamento contendo as condições de contorno aplicadas pode ser visto na figura 4.1. O canal para essa simulação terá altura igual a uma unidade de comprimento e comprimento igual a seis unidades de comprimento, de modo a ser equivalente à verificação desse mesmo experimento realizada por DA CUNHA (2020). O valor de v_x na entrada do escoamento será igual a 1 e, com essas informações é possível definir a solução analítica para a curva da velocidade esperada, que será igual à equação 4.1.

$$v_x = 6y(1 - y) \tag{4.1}$$

Portanto, para essa simulação de vericação, os valores encontrados para v_x em um corte vertical do canal devem estar compatíveis com os valores da equação 4.1, na qual y se refere à altura do canal e v_x à velocidade no eixo horizontal. Os parâmetros que caracterizam o escoamento e a simulação podem ser consultados na tabela 4.1.

Parâmetros para simulação de verificação	
v_x na entrada do escoamento	1
Massa específica ρ	1
Viscosidade dinâmica ν	0,04
Número de Reynolds Re	25
Número de iterações	1000
Intervalo de tempo entre iterações	0,1
Número de elementos na malha	3100
Números de nós	1644
Element size factor	0,073

Tabela 4.1: Parâmetros da simulação de escoamento entre placas planas.

Assim, após rodar a simulação e de posse da solução analítica disponível na equação 4.1, foi construído um gráfico comparando esses dados. Esse gráfico está disponível na figura 4.3. Os dados utilizados para ilustrar a solução numérica na figura 4.3 foram obtidos a partir de uma seleção de 50 pontos igualmente espaçados dos valores obtidos para a velocidade na direção X (v_x) de uma seção vertical do canal localizada a 5 unidades de comprimento da entrada durante a iteração 1000.

Para extrair esses valores, foi utilizada a função *Plot over line* do software **Paraview** e uma imagem da localização da extração de dados pode ser vista na figura abaixo:

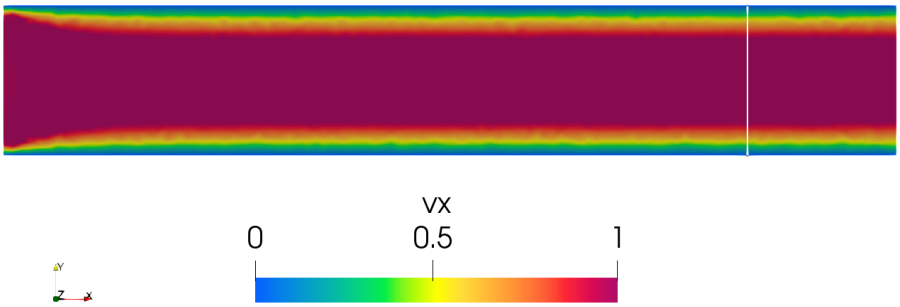

Figura 4.2: Visualização do canal criado para validar a solução numérica. O canal possui 1 unidade de comprimento de altura e um comprimento equivalente a 6 unidades de comprimento. A medição vx da imagem se refere à velocidade do escoamento na direção horizontal e a seção vertical de onde os dados foram extraídos é a da seta branca presente na imagem, que está localizada a 5 unidades de comprimento da entrada do canal do lado esquerdo. Essa captura de imagem se refere à 1000ª iteração do código numérico.

Ao analisar a figura 4.3, nota-se que há uma concordância satisfatória entre os dados. A pequena falta de conformidade entre os dados numéricos e analíticos na região central do gráfico provavelmente se dá por conta do número de iterações não ter sido suficiente para formar o estado estacionário do escoamento.

No entanto, tendo em vista as limitações computacionais impostas, o resultado dessa comparação se mostra adequado. Desse modo, o código numérico elaborado está qualificado para gerar dados condizentes com a realidade e, portanto, está verificado.

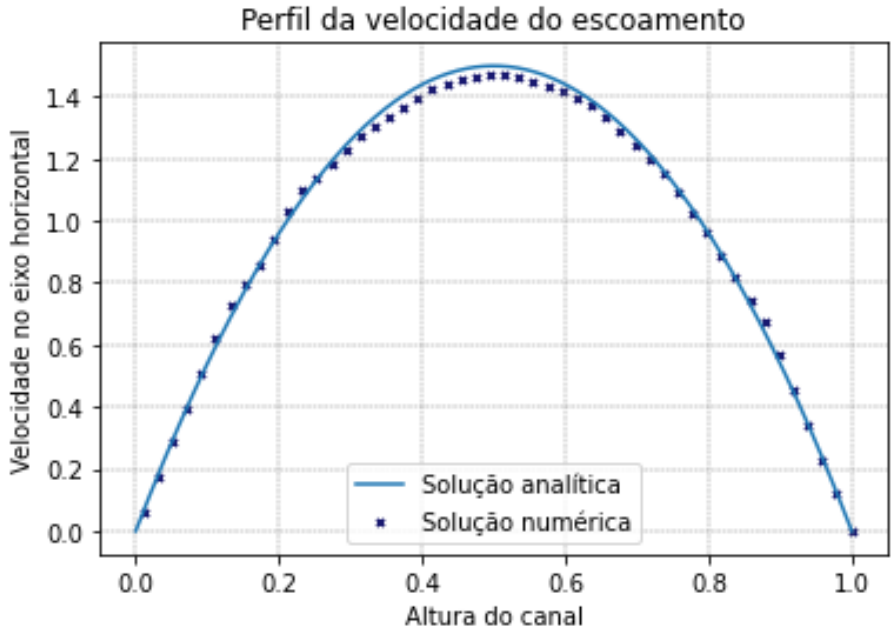

Figura 4.3: Gráfico $v_x \times y$, onde se compara os resultados da solução numérica e analítica para o perfil da velocidade do escoamento entre placas planas horizontais.

4.2 Escoamento *Lid-driven cavity*

Outro problema muito estudado e replicado nos estudos de mecânica dos fluidos computacional é o escoamento *lid-driven cavity* ou escoamento laminar em uma cavidade. Esse escoamento é caracterizado por uma velocidade de escoamento $v_x = 1$ na superfície superior de uma cavidade. As demais condições de contorno estão explicitadas na figura 4.4.

Esse teste de verificação tem como objetivo comparar os valores encontrados para v_x ao longo da reta $x = 0.5$ que atravessa a cavidade e os valores de v_y ao longo da reta $y = 0.5$ que também corta a cavidade com valores de referência conhecidos. Caso esses valores estejam compatíveis, o código numérico utilizado nesse trabalho será novamente verificado.

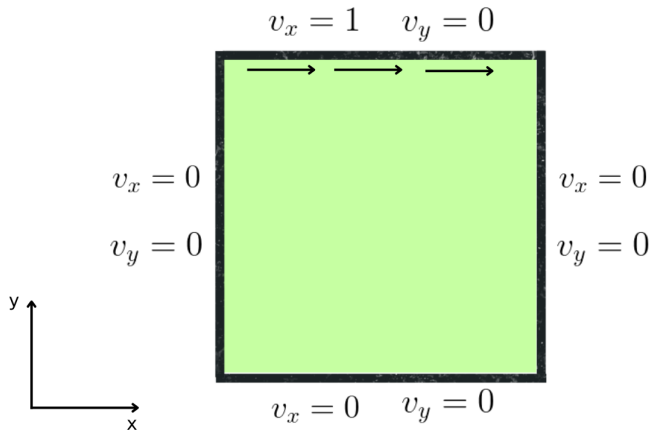

Figura 4.4: Desenho esquemático ilustrando o escoamento *Lid-driven cavity* e as condições de contorno que caracterizam o problema. As dimensões da cavidade são, em unidades de comprimento, 1×1. Fonte: elaboração própria.

Os parâmetros utilizados na simulação numérica feita para este trabalho podem ser consultados na tabela 4.2. Vale destacar que as dimensões do canal em que a simulação foi realizada são 1×1. O número de Reynolds escolhido foi igual a 10 para que essa replicação possa ser comparável com os resultados numéricos obtidos por MARCHI *et al.* (2009). Utilizar o artigo de MARCHI *et al.* (2009) como referência é conveniente pois os dados numéricos apresentados nesse artigo se mostraram compatíveis com a solução analítica apresentada por SHIH *et al.* (1989) para esse mesmo problema. Portanto, a verificação do código numérico desenvolvido com os resultados encontrados por MARCHI *et al.* (2009) consiste em uma dupla verificação.

Parâmetros para a simulação *Lid-driven*	
Massa específica ρ	1
Viscosidade dinâmica ν	0,1
Número de Reynolds Re	10
Número de iterações	500
Intervalo de tempo entre iterações	0,01
Número de elementos da malha	3166
Número de nós	1650
Element size factor	0,029

Tabela 4.2: Parâmetros da simulação de escoamento *lid-driven cavity*.

Após rodar a simulação com os parâmetros expostos, as distribuições de valores para a velocidade no eixo horizontal v_x e para a velocidade no eixo vertical v_y ao longo da cavidade foram obtidas e podem ser consultadas nas figuras abaixo:

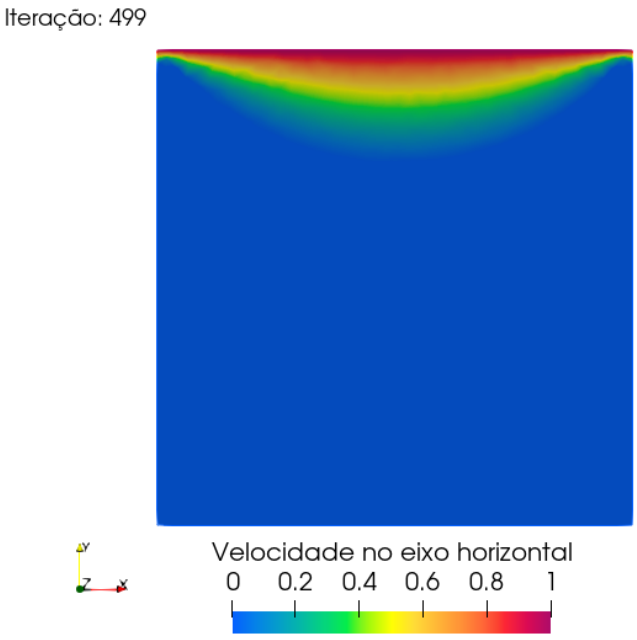

Figura 4.5: Solução numérica de v_x para o escoamento *Lid-driven cavity* com número de Reynolds igual a 10. Essa imagem representa a 500ª iteração do código feito para essa verificação.

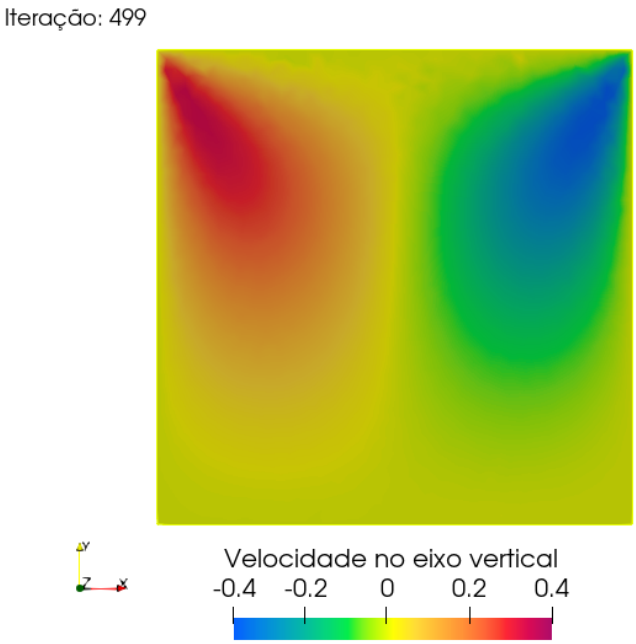

Figura 4.6: Solução numérica de v_y para o escoamento *Lid-driven cavity* com número de Reynolds igual a 10. Essa imagem representa a 500ª iteração do código feito para essa verificação.

Para obter os valores de v_x e v_y ao longo das linhas que cortam a cavidade definidas por $x = 0.5$ e $y = 0.5$, respectivamente, foi utilizada a função *Plot over line* do software **Paraview**. A visualização das linhas nas quais os dados são obtidos está disponível nas figuras 4.7 e 4.8.

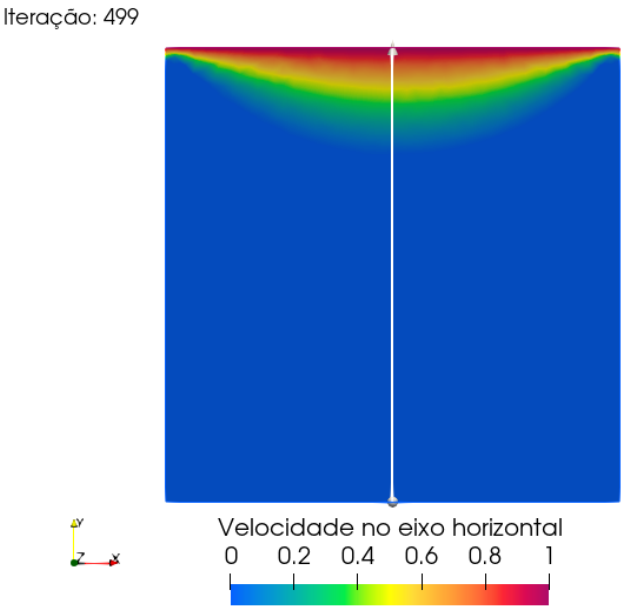

Figura 4.7: Solução numérica de v_x para o escoamento *Lid-driven cavity* com número de Reynolds igual a 10. Essa imagem representa a 500ª iteração do código feito para essa verificação.

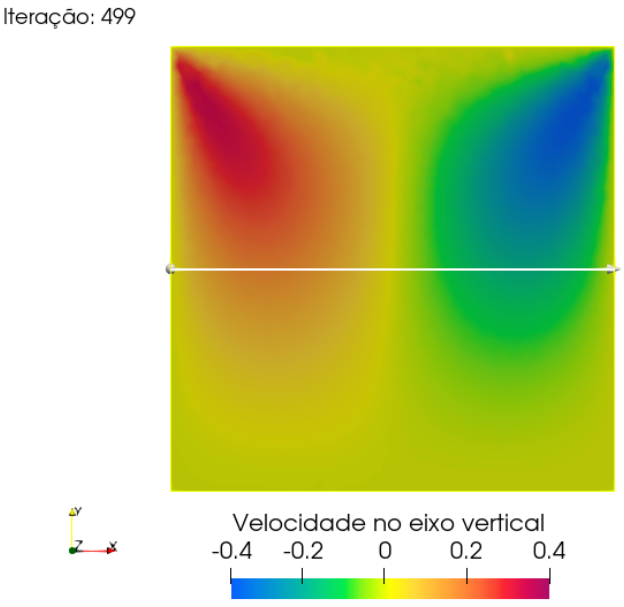

Figura 4.8: Solução numérica de v_y para o escoamento *Lid-driven cavity* com número de Reynolds igual a 10. Essa imagem representa a 500ª iteração do código feito para essa verificação.

De posse dos dados que a função *Plot over line* forneceu, foi possível criar gráficos

comparativos entre os valores obtidos por esse trabalho e por MARCHI *et al.* (2009). Como pode ser visto nas figuras 4.9 e 4.10, a concordância entre os dois trabalhos foi excelente. Portanto, o presente código numérico apresentado está novamente verificado.

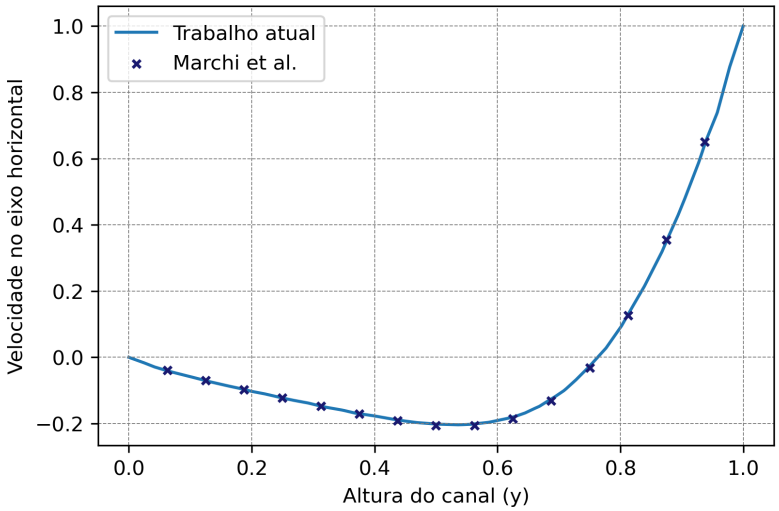

Figura 4.9: Comparação entre os resultados do presente trabalho e de MARCHI *et al.* (2009) obtidos para v_x na reta $x = 0.5$ que corta a cavidade.

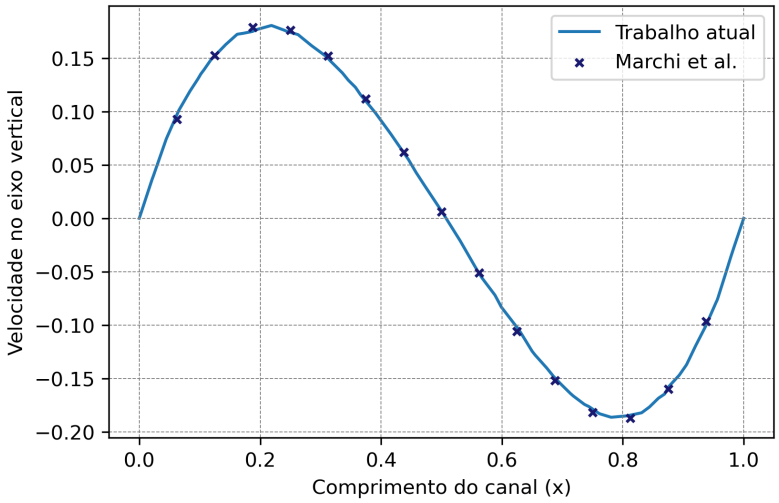

Figura 4.10: Comparação entre os resultados do presente trabalho e de MARCHI *et al.* (2009) obtidos para v_y na reta $y = 0.5$ que corta a cavidade.

Capítulo 5

Resultados

Depois de realizar a verificação do código numérico utilizado, as simulações com diferentes geometrias que emulam a porosidade de filtros de partículas foram realizadas. Conforme foi estabelecido no capítulo 3, três testes foram realizados para cada um dos três tipos de geometrias abordadas, totalizando 9 simulações. O computador utilizado para realizar essas simulações foi o mesmo cujas especificações foram mencionadas ao falar sobre a verificação do código numérico, no capítulo 4

Os parâmetros das simulações que descrevem o escoamento, como o número de Reynolds por exemplo, se manteve constante em todas as simulações. Esses parâmetros comuns podem ser consultados na tabela 5.1. Vale enfatizar que essas simulações foram realizadas considerando a forma adimensional da formulação corrente-vorticidade e o comprimento característico foi definido como a menor distância entre as bordas de dois obstáculos circulares. Em todas as simulações o valor dessa distância é igual a um, para que a forma adimensional da formulação corrente-vorticidade pudesse ser utilizada sem problemas.

As condições de contorno adotadas para as simulações de 1 a 9 estão esquematizadas na figura 5.1. A geometria do canal dessas simulações possui dimensões 10×20. É importante destacar que o obstáculo circular presente no meio do canal da figura é ilustrativo e sua condição de contorno se aplica a todos os obstáculos simulados. Portanto, todos os obstáculos de todas as simulações possuem condições de contorno de velocidade iguais a $v_x = 0$ e $v_y = 0$. Quanto a função corrente, todos os obstáculos possuem uma condição de contorno equivalente a $\psi = y_p$, em que y_p significa a coordenada do eixo y que o centro do obstáculo possui.

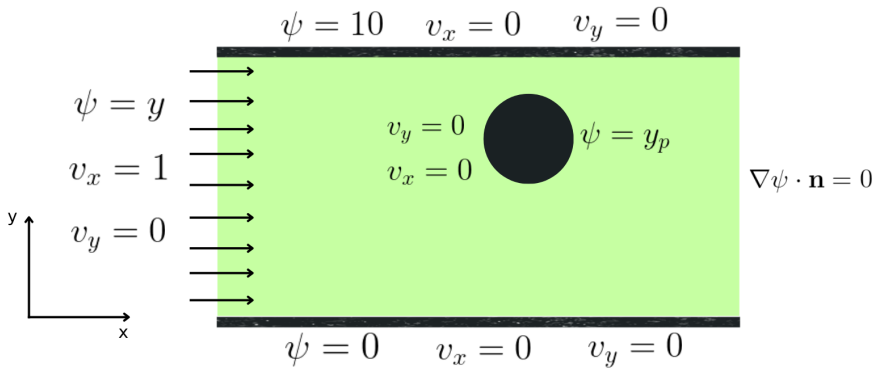

Figura 5.1: Modelo esquemático contendo as condições de contorno consideradas para a simulação de escoamento entre placas planas. As dimensões do canal são 10×20. O obstáculo circular é ilustrativo e representa todos os obstáculos simulados. Fonte: elaboração própria.

Com os parâmetros comuns a todas as simulações definidos será possível se debruçar sobre as simulações individuais ressaltando suas particularidades. As descrições e resultados obtidos para cada teste podem ser consultadas nos tópicos seguintes.

Parâmetros comuns a todos os escoamentos simulados	
Altura do canal	10
Comprimento do canal	20
v_x na entrada do canal	1,0
v_y na entrada do canal	0
Comprimento característico L	1,0
Massa específica ρ	1,0
Viscosidade dinâmica ν	0,04
Número de Reynolds Re	25
Número de iterações	1000
Intervalo de tempo entre iterações	0,05

Tabela 5.1: Tabela de parâmetros característicos de todas as simulações.

5.1 Geometrias com obstáculos alinhados

O primeiro tipo de geometria estudado consiste em um canal de filtro de partículas cuja porosidade é emulada por obstáculos circulares de raios iguais, sólidos, que estão alinhados entre si e organizados em linhas e colunas. Para as três simulação dessa categoria, o número de obstáculos e o diâmetro dos mesmos são modificados. Na tabela 5.2, é possível conferir os parâmetros utilizados para cada um desses testes, que se somam aos parâmetros globais da 5.1.

Simulação	**Diâmetro dos obstáculos**	**Nº de obstáculos**	**Elementos da malha**	**Área ocupada por obstáculos**
1	1.0	28	3186	10.99
2	2.0	15	2686	23.56
3	3.0	8	3148	28.27

Tabela 5.2: Parâmetros e informações relevantes sobre as simulações realizadas com geometrias que possuem obstáculos alinhados.

As geometrias e malhas das simulações 1, 2 e 3 estão expostas abaixo e os resultados obtidos para os cálculos da velocidade, vorticidade e função corrente do escoamento se apresentam nos subtópicos seguintes.

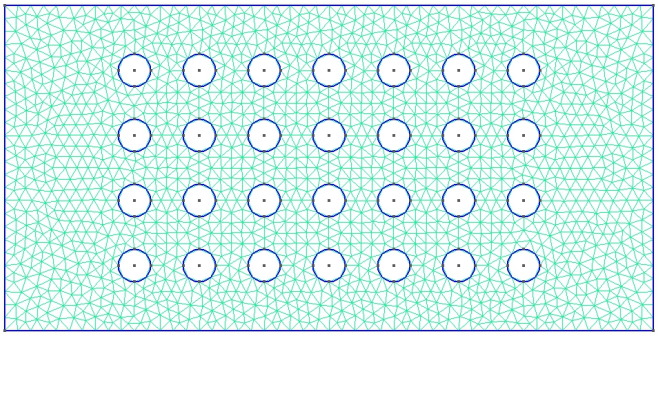

Figura 5.2: Geometria e malha feita para a simulação 1. As dimensões do canal são 10×20 e o diâmetro de todos os obstáculos circulares é igual a 1.

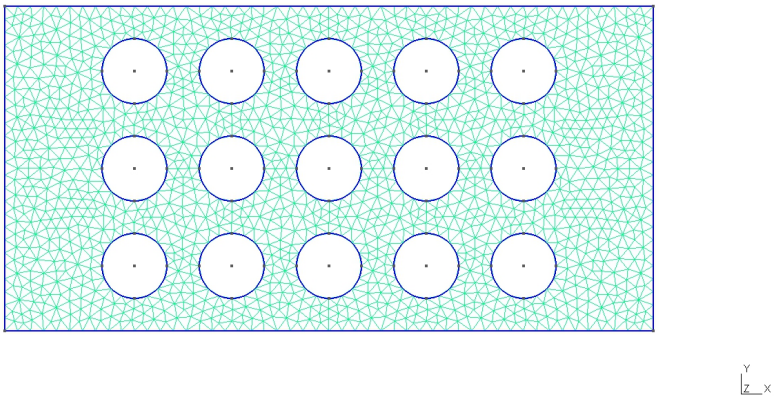

Figura 5.3: Geometria e malha feita para a simulação 2. As dimensões do canal são 10×20 e o diâmetro de todos os obstáculos circulares é igual a 2.

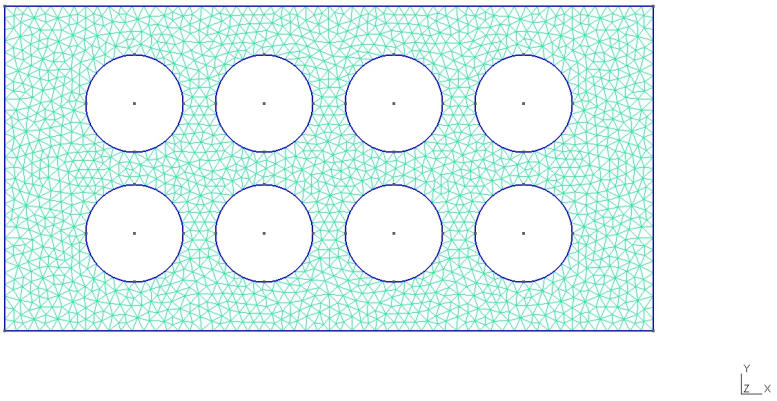

Figura 5.4: Geometria e malha feita para a simulação 3. As dimensões do canal são 10×20 e o diâmetro de todos os obstáculos circulares é igual a 3.

5.1.1 Obstáculos com diâmetro igual a 1

Assim como todas as simulações, a simulação 1 foi realizada em um canal com dimensões 10×20. Além disso, conforme foi exposto na tabela 5.2, os 28 obstáculos circulares dessa simulação possuem diâmetro igual a 1.0. A malha dessa simulação contém 3186 elementos e a área ocupada pelos obstáculos circulares é igual a 10,99 unidades de área. Considerando que a área do canal é igual a 200 unidades de área, densidade dos obstáculos θ, definida no capítulo 3, encontrada é igual a 0,05495.

Os valores para as velocidades v_x e v_y, vorticidade ω_z e função corrente ψ foram calculados para todos os nós da malha da simulação 1 em cada uma das 1000 iterações para os quais essa simulação foi submetida. Para a iteração 1000, as distribuições ao longo do escoamento no canal desses valores calculados podem ser consultadas na imagem 5.5. A iteração 1000 foi escolhida para ilustrar essa simulação pois nela foi considerado que o escoamento já estaria estabilizado em seu estado estacionário. Vale destacar que, embora e etiqueta "Time: 999" esteja caracterizando as fotos, a iteração ilustrada consiste na 1000ª iteração porque a primeira iteração foi numerada como 0.

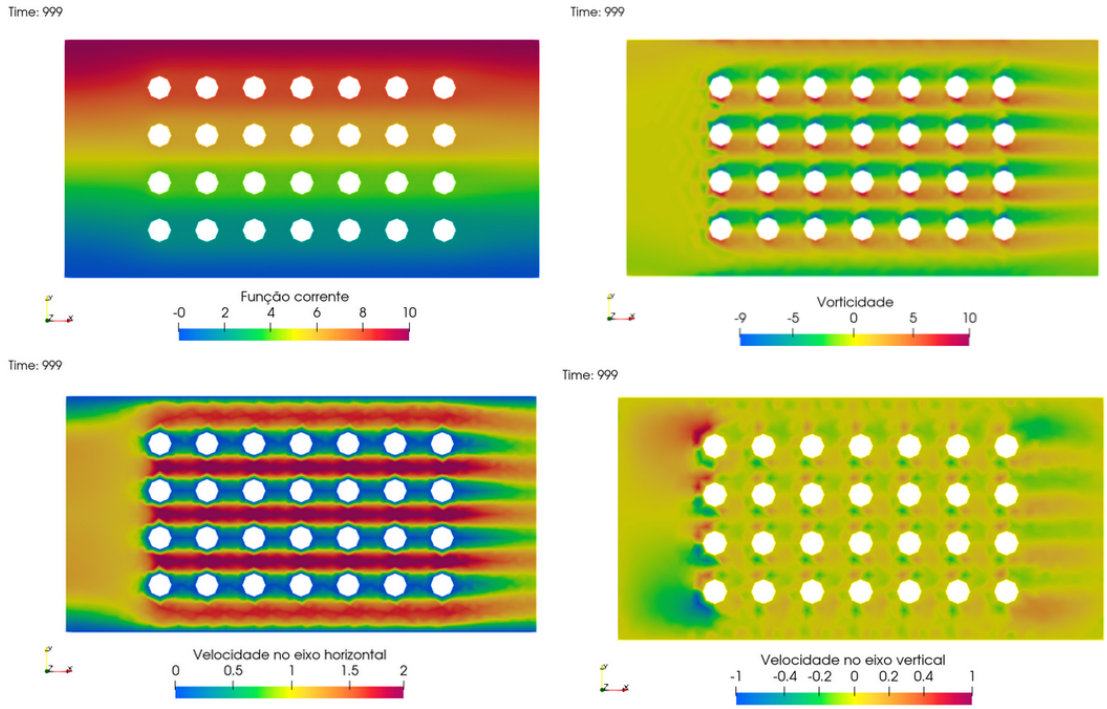

Figura 5.5: Resultados da simulação 1 mostrando os valores de v_x, v_y, ω_z e ψ calculados para a 1000ª iteração.

Para encontrar o valor da velocidade média no eixo horizontal do escoamento na saída do canal, a função *Plot Over Line* do software **Paraview** foi utilizada para obter os valores de v_x. A posição escolhida para retirar esses dados foi uma reta vertical que corta o canal a 18 unidades de comprimento da entrada do canal. Essa reta está ilustrada na figura 5.6 como a seta branca que corta o canal.

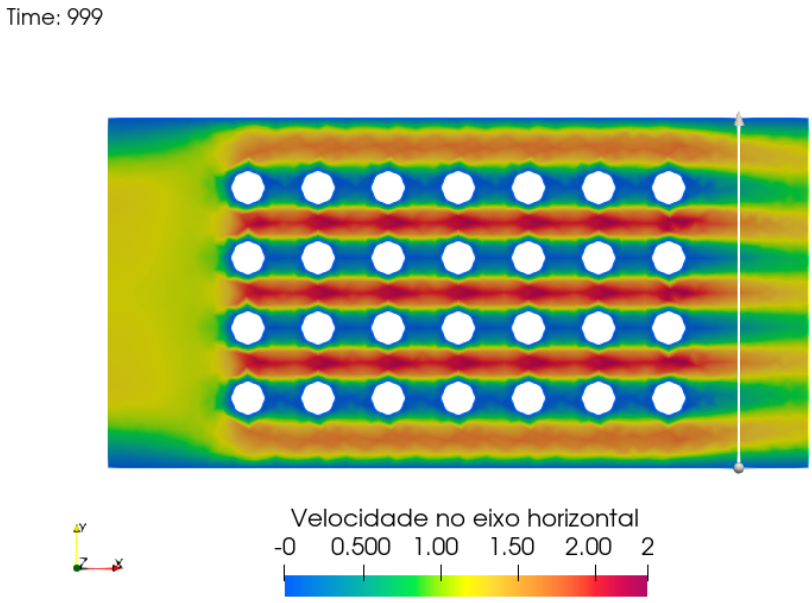

Figura 5.6: Resultado da simulação 1 para a sua 1000ª iteração e seta da função *Plot over line* localizada a 18 unidades de comprimento da entrada do canal mostrando onde essa função foi aplicada.

Por meio da função *Plot Over Line*, foi possível obter uma lista com todos os valores de v_x dos pontos que passam por essa reta. Ao fazer a média desses valores, a velocidade média da saída do escoamento na direção do eixo x foi encontrada. Para a simulação 1, esse valor foi igual a 0,987248482 unidades de velocidade. Levando em consideração que a condição de contorno na entrada do canal considera uma velocidade de entrada $v_x = 1$ igual em toda a extensão da entrada do canal, a velocidade média de entrada do escoamento na direção do eixo x é igual a um.

De posse dos valores para velocidade média de saída e de entrada do escoamento na direção do eixo x, é possível calcular o fator de qualidade do filtro α definido no capítulo 3. Arredondando o valore da velocidade de saída para três casas decimais, $\alpha = \frac{1}{0{,}987}$ e, portanto, o valor encontrado para o fator de qualidade do filtro é igual a 1,01317.

A correlação entre os valores encontrados de θ e α será realizada no tópico 5.1.4, junto com os demais valores encontrados para esses parâmetros nas simulações 2 e 3.

5.1.2 Obstáculos com diâmetro igual a 2

Quanto a simulação 2, conforme foi apresentado na tabela 5.2, os seus 15 obstáculos circulares dessa simulação possuem diâmetro igual a 2.0. A malha dessa simulação contém 2686 elementos e a área ocupada pelos obstáculos circulares é igual a 23,56 unidades de área. Como a área do canal é igual a 200 unidades de área, densidade dos obstáculos θ, definida no capítulo 3, encontrada é igual a 0,1178.

A distribuição dos valores para as velocidades v_x e v_y, vorticidade ω_z e função corrente ψ calculados ao longo do canal para todas as 1000 iterações dessa simulação podem ser visualizados na figura 5.7. Novamente, a iteração 1000 foi escolhida para ilustrar o resultado desse experimento.

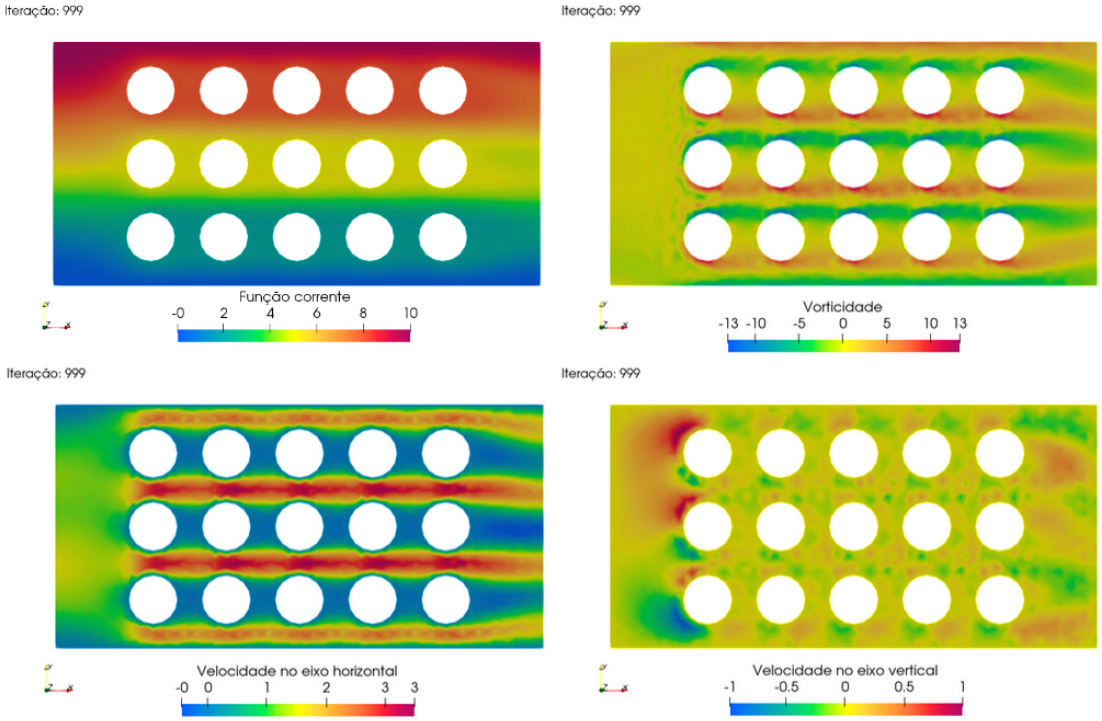

Figura 5.7: Resultados da simulação 2 mostrando os valores de v_x, v_y, ω_z e ψ calculados para a 1000ª iteração.

A função *Plot Over Line* foi utilizada para se obter os valores de v_x na mesma localização do canal de quando esse procedimento foi realizado na simulação 1. A velocidade média de saída do escoamento calculada para a simulação 2 foi igual a 0,951171213. Isso confere para a simulação 2, um fator de qualidade de filtragem α com valor igual a 1,05133.

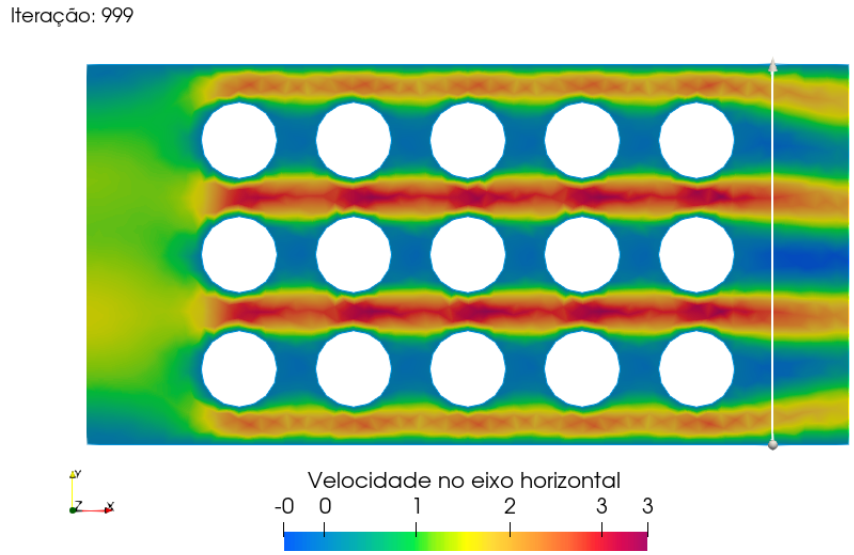

Figura 5.8: Resultado da simulação 2 para a sua 1000ª iteração e seta da função *Plot over line* localizada a 18 unidades de comprimento da entrada do canal mostrando onde essa função foi aplicada.

5.1.3 Obstáculos com diâmetro igual a 3

Em um canal de dimensões idênticas às simulações 1 e 2, conforme os dados da tabela 5.2, os 8 obstáculos circulares dessa simulação possuem diâmetro igual a 3.0. A malha dessa simulação contém 3148 elementos e a área ocupada pelos obstáculos circulares é igual a 38,27 unidades de área. De posse dos valores das dimensões do canal, encontra-se um valor para a densidade dos obstáculos θ, definida no capítulo 3, igual a 0,14135.

Ilustrações com a distribuição dos valores para as velocidades v_x e v_y, vorticidade ω_z e função corrente ψ calculados ao longo do canal para a 1000ª iteração dessa simulação estão na figura 5.9.

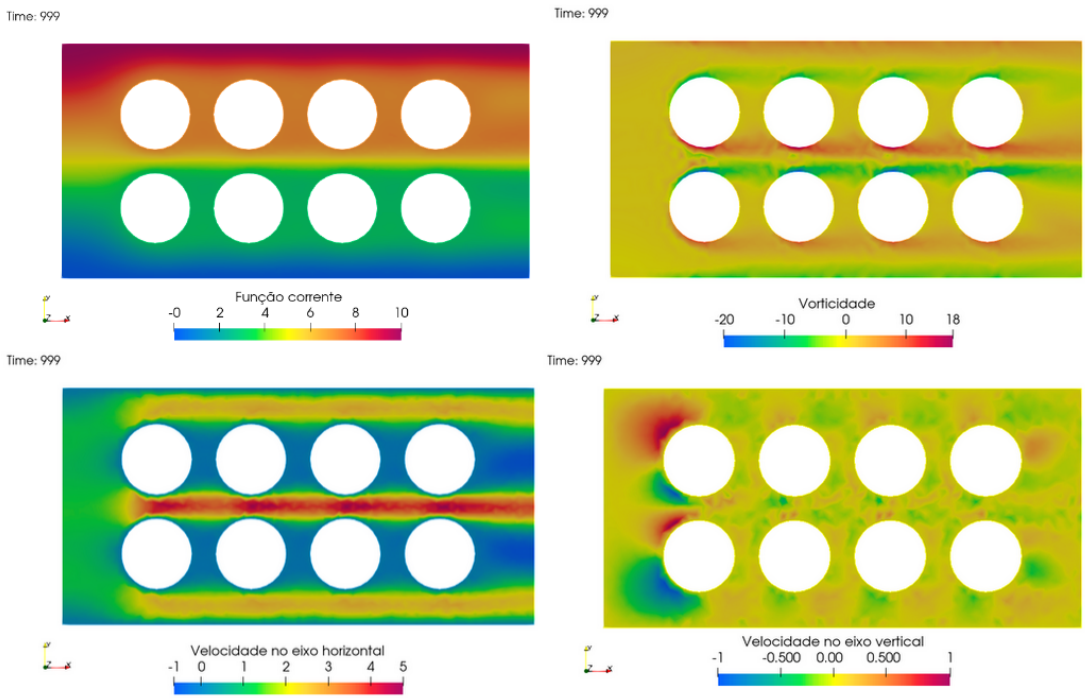

Figura 5.9: Resultados da simulação 3 mostrando os valores de v_x, v_y, ω_z e ψ calculados para a 1000ª iteração.

A função *Plot Over Line* foi utilizada para se obter os valores de v_x na mesma localização do canal de quando esse procedimento foi realizado na simulação 1 e 2. A velocidade média de saída do escoamento calculada para a simulação 3 foi igual a 0,978705598. Isso confere para a simulação 3, um fator de qualidade de filtragem α com valor igual a 1,02176.

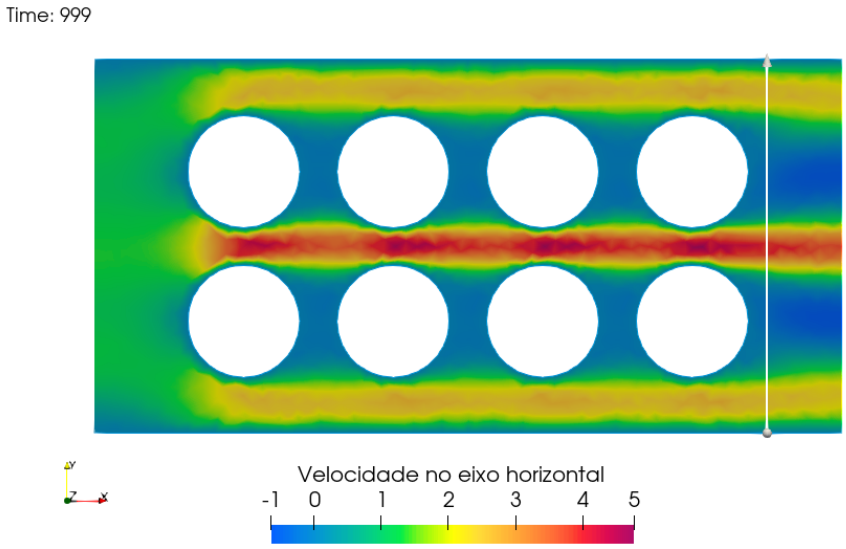

Figura 5.10: Resultado da simulação 3 para a sua 1000ª iteração e seta da função *Plot over line* localizada a 18 unidades de comprimento da entrada do canal mostrando onde essa função foi aplicada.

5.1.4 Análise de qualidade

Após realizar as simulações 1, 2 e 3, é possível associar os valores do fator de qualidade α e os valores da densidade de obstáculos θ de cada simulação em um único gráfico. Esse gráfico pode ser consultado na figura 5.11 e foi criado utilizando **Python**.

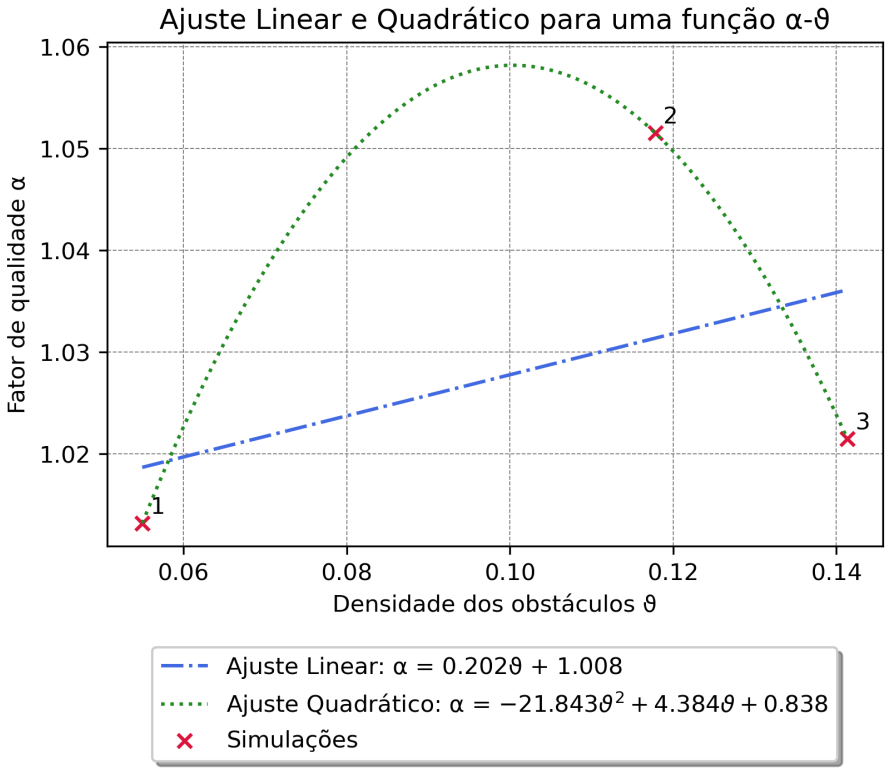

Figura 5.11: Dados de fator de qualidade α e densidade de obstáculos θ para as simulações 1, 2 e 3. Um ajuste linear e um ajuste quadrático foram feitos a partir desses dados para representar funções que relacionem esses dois parâmetros. As funções que descrevem ambos os ajustes podem ser visualizadas na legenda.

Além de indicar um ponto no gráfico para cada simulação, dois ajustes foram realizados para encontrar uma função que possa descrever as tendências demonstradas pelos dados. Os ajustes realizados foram um ajuste linear e outro quadrático. Evidentemente, a quantidade limitada de simulações dificulta interpretações conclusivas mas os dados disponíveis demonstram tendências interessantes por si só.

O ajuste linear indica uma tendência da qualidade da filtragem aumentar quando a densidade de obstáculos aumenta. Por outro lado, o ajuste quadrático informa que a densidade de obstáculos favorece a qualidade de filtragem, mas só até certo ponto. Ou seja, existiria um valor para a densidade de obstáculos que seria ótimo e ao aumentar a densidade de obstáculos além desse ponto, a eficiência de filtragem diminuiria.

A queda do fator de qualidade na terceira simulação, em comparação com a simulação 2, pode ser devido aos obstáculos maiores da simulação 3 acabarem criando

"corredores"por onde o escoamento escapa sem que hajam obstáculos em seu caminho que o desacelerem. Esse efeito seria maior quando os obstáculos são maiores pois nessa situação temos um número menor de obstáculos no canal e, com porções maiores das seções verticais do canal ocupadas, o escoamento atinge maiores velocidades do que nos casos anteriores nas localizações desses "corredores"do canal sem obstáculos.

Portanto, o comportamento de queda do fator de qualidade na simulação 3 pode ser explicado. No entanto, esse comportamento não deixa de ser inesperado. Por isso, o ideal seria que mais simulações com esse tipo de geometria fossem realizadas para que mais pontos pudessem ser acrescentados a esse ajuste. Isso contribuiria para identificar com mais precisão a relação entre o fator de qualidade α e a densidade de obstáculos θ. Infelizmente, não foi possível realizar mais simulações e aprofundar esses estudos devido ao limitado poder computacional disponível no computador pessoal em que estar simulações foram realizadas.

5.2 Geometrias com obstáculos desalinhados

A segunda abordagem geométrica para estudar filtros de partículas utilizou obstáculos circulares sólidos e desalinhados entre si com raios iguais. O desafio da formação dessa geometria foi executar esse desalinhamento de forma a manter a menor distância entre as bordas de dois obstáculos igual a um, para que o comprimento característico pudesse ser um. No final, para representar essa geometria, colunas de obstáculos circulares desalinhados foram colocadas distantes umas das outras para garantir que a menor distância entre entre dois obstáculos seria dada pelo espaço que separa dois obstáculos de uma mesma coluna. Essa distância foi definida como igual a um.

Para as três simulações dessa categoria, novamente os principais parâmetros que variaram entre elas foram o número de obstáculos e os seus diâmetros. Outras características de cada simulação podem ser consultadas na tabela 5.3. Vale reiterar que essas características se somam aos parâmetros globais presentes na tabela 5.1.

Simulação	**Diâmetro dos obstáculos**	**Nº de obstáculos**	**Elementos da malha**	**Área ocupada por obstáculos**
4	1.0	20	3252	7.85
5	2.0	8	2984	12.57
6	3.0	5	2780	17.67

Tabela 5.3: Parâmetros que caracterizam simulações realizadas com obstáculos distribuídos de forma desalinhada entre si.

As geometrias e malhas das simulações 4, 5 e 6 estão expostas abaixo e os resultados obtidos para os cálculos da velocidade, vorticidade e função corrente do escoamento se apresentam nos subtópicos seguintes. A análise de resultados também será feita nessas seções.

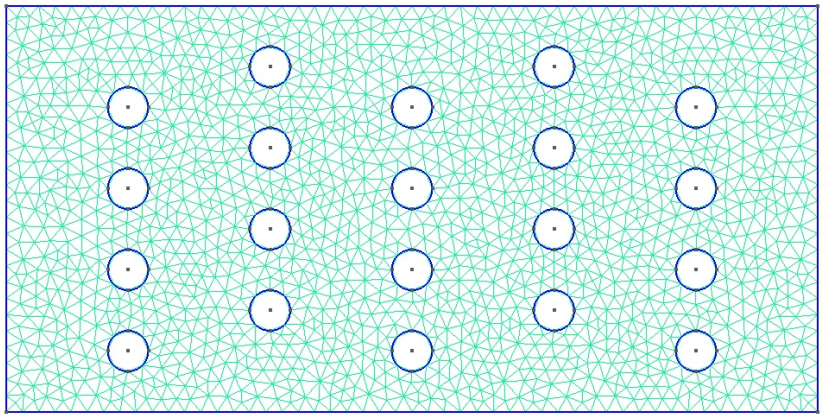

Figura 5.12: Geometria e malha feita para a simulação 4. As dimensões do canal são 10×20 e o diâmetro de todos os obstáculos circulares é igual a 1.

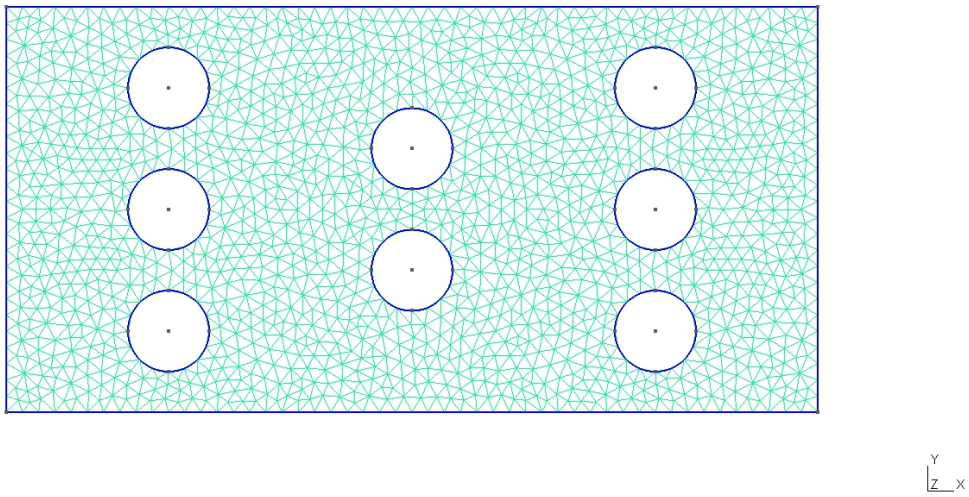

Figura 5.13: Geometria e malha feita para a simulação 5. As dimensões do canal são 10 × 20 e o diâmetro de todos os obstáculos circulares é igual a 2.

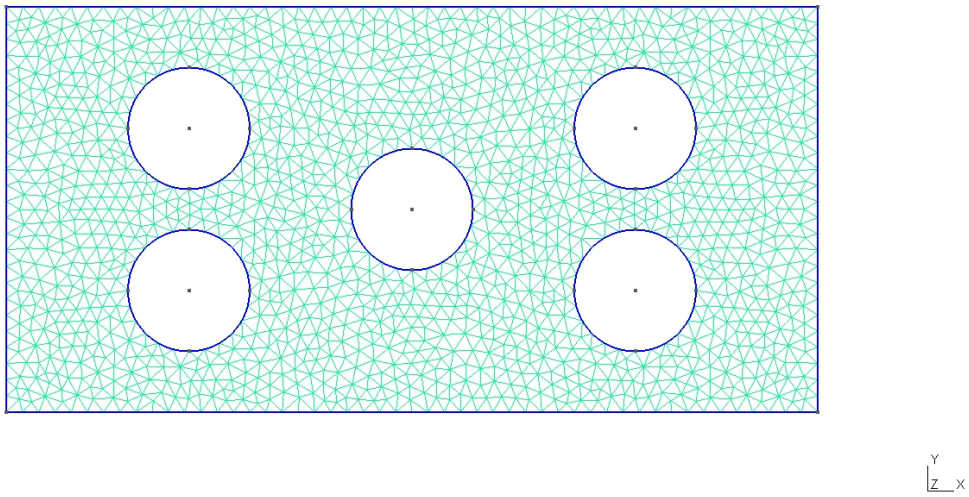

Figura 5.14: Geometria e malha feita para a simulação 6. As dimensões do canal são 10 × 20 e o diâmetro de todos os obstáculos circulares é igual a 3.

5.2.1 Obstáculos com diâmetro igual a 1

Para a primeira simulação com geometria de obstáculos desalinhados, 20 obstáculos sólidos, circulares e de diâmetro igual a 1.0 foram distribuídos no canal simulado. Conforme informado pela tabela 5.3, a malha dessa simulação possui 3252 elementos e os seus obstáculos ocupam uma área igual a 7,85 unidades de área. Esse valor confere para a simulação 4 uma densidade de obstáculos θ igual a 0,03925.

Em cada uma das iterações executadas, os valores para as velocidades v_x e v_y, vorticidade ω_z e função corrente ψ foram calculados para todos os nós da malha dessa simulação. A visualização desses dados para a 300ª iteração está disponível na figura 5.15.

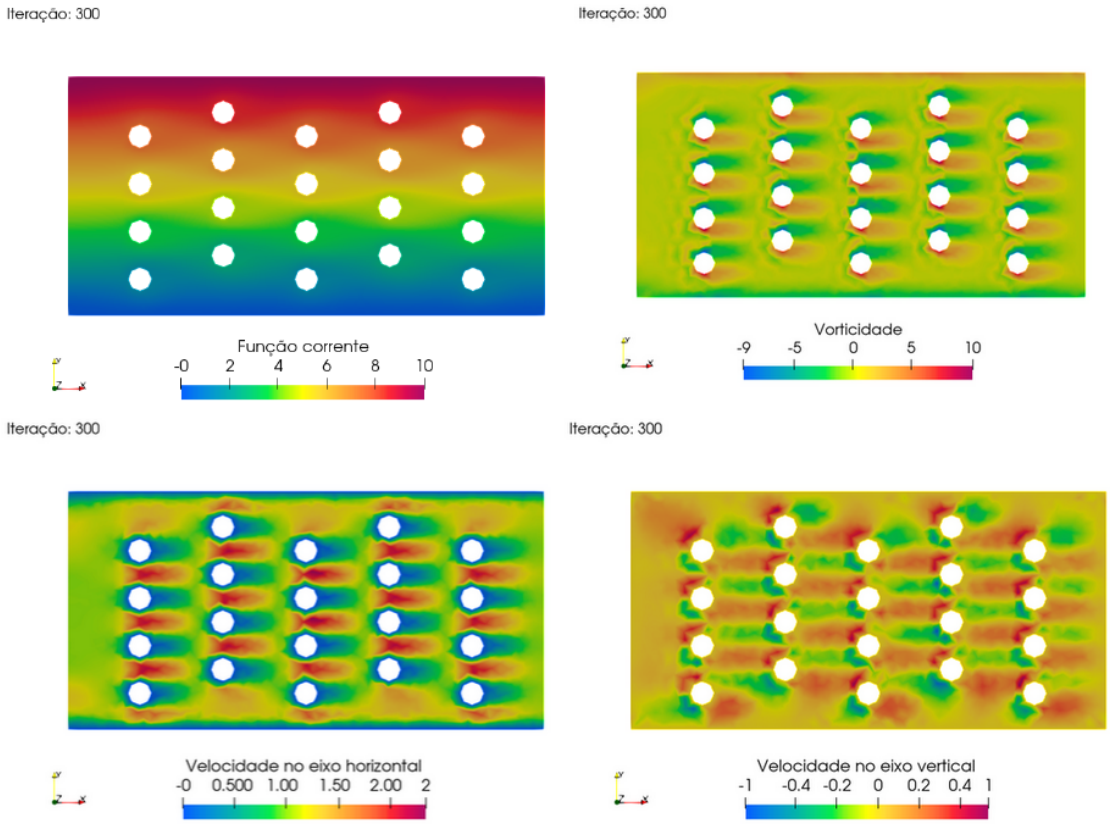

Figura 5.15: Resultados da simulação 4 mostrando os valores de v_x, v_y, ω_z e ψ calculados para a 300ª iteração.

A função *Plot Over Line* foi novamente utilizada para se obter os valores de v_x na mesma localização em que essa função foi executada para as simulações anteriores. A velocidade média de saída do escoamento calculada para a simulação 4 foi igual

a 0,983216973. Isso confere para a simulação 4 um fator de qualidade de filtragem α com valor igual a 1,01707.

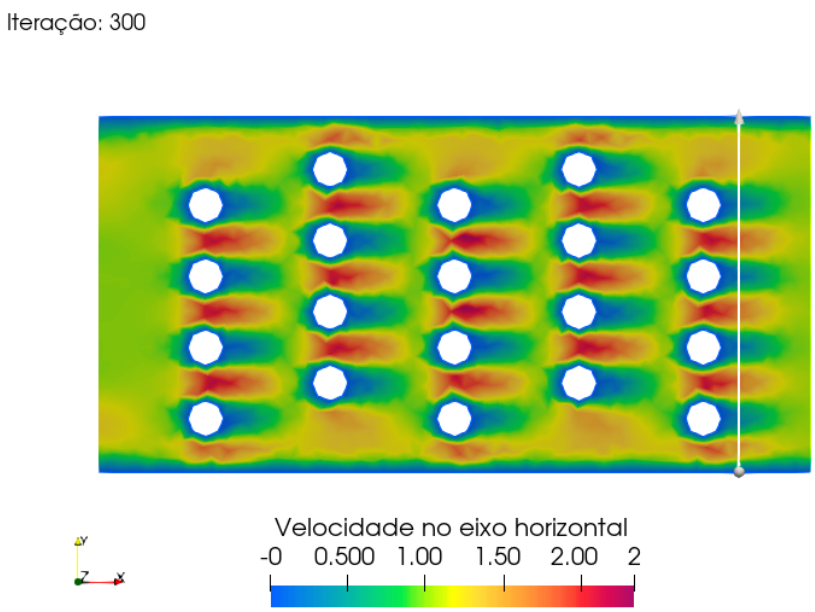

Figura 5.16: Resultado da simulação 4 para a sua 300ª iteração e seta da função *Plot over line* localizada a 18 unidades de comprimento da entrada do canal mostrando onde essa função foi aplicada.

5.2.2 Obstáculos com diâmetro igual a 2

Para a simulação 5 foram distribuídos 8 obstáculos sólidos, circulares e de diâmetro igual a 2.0 no canal simulado. Conforme informado pela tabela 5.3, a malha dessa simulação possui 2984 elementos e os seus obstáculos ocupam uma área igual a 12,57 unidades de área. Esse valor confere para a simulação 5 uma densidade de obstáculos θ igual a 0,06285.

Em cada uma das iterações executadas, os valores para as velocidades v_x e v_y, vorticidade ω_z e função corrente ψ foram calculados para todos os nós da malha dessa simulação. A visualização desses dados para a 1000ª iteração está disponível na figura 5.17.

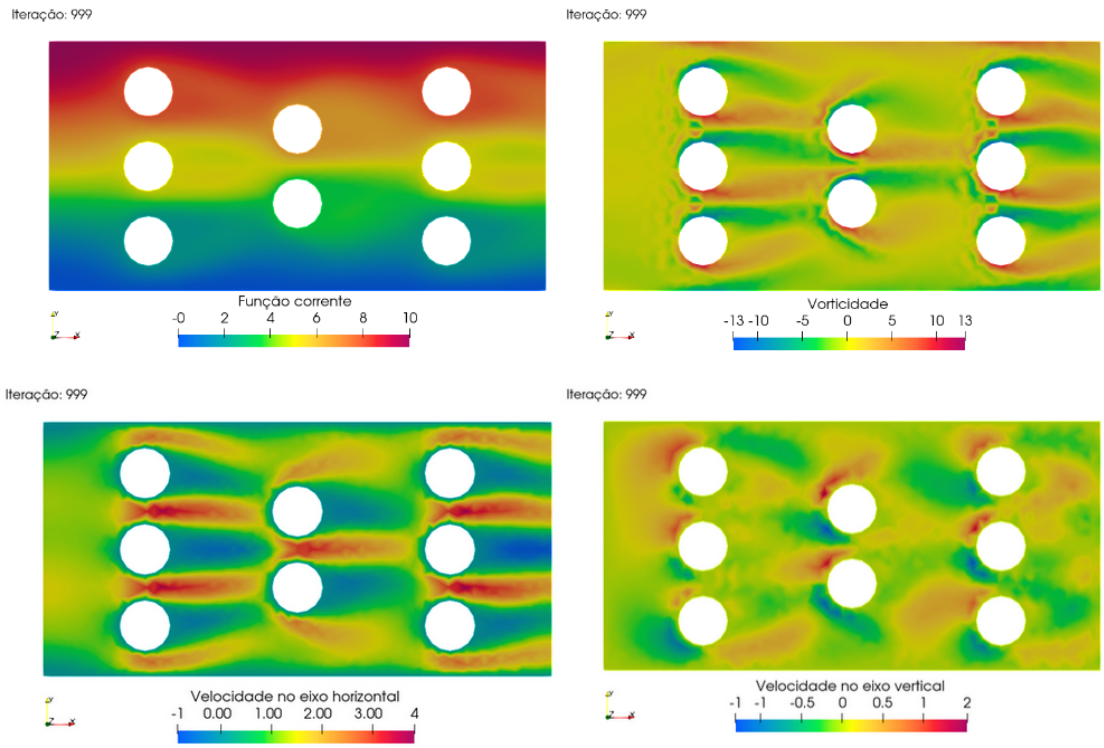

Figura 5.17: Resultados da simulação 5 mostrando os valores de v_x, v_y, ω_z e ψ calculados para a 1000ª iteração.

A função *Plot Over Line* foi utilizada para se obter os valores de v_x na mesma localização em que essa função foi executada para as simulações anteriores. A velocidade média de saída do escoamento calculada para a simulação 5 foi igual a 0,943118828. Isso confere para a simulação 5 um fator de qualidade de filtragem α com valor igual a 1,060312.

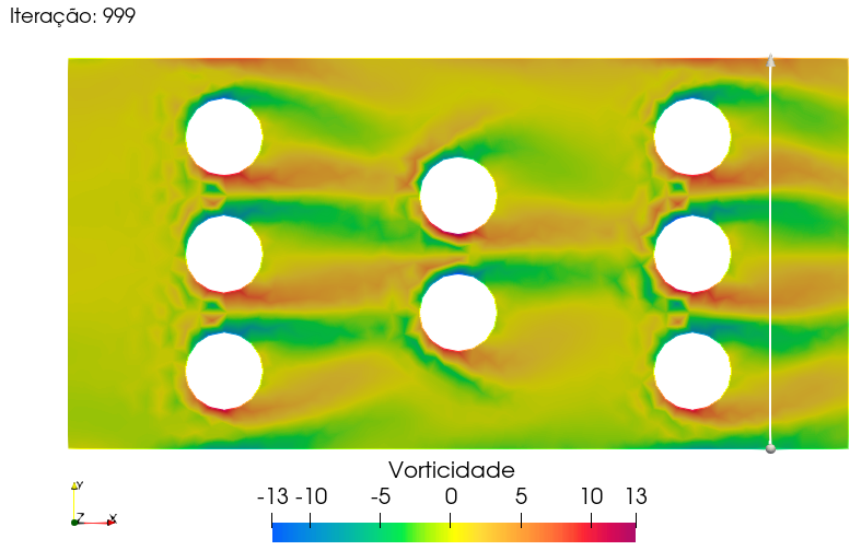

Figura 5.18: Resultado da simulação 5 para a sua 1000ª iteração e seta da função *Plot over line* localizada a 18 unidades de comprimento da entrada do canal mostrando onde essa função foi aplicada.

5.2.3 Obstáculos com diâmetro igual a 3

Na simulação 6, foram distribuídos 8 obstáculos sólidos, circulares e de diâmetro igual a 3.0 no canal simulado. Conforme informado pela tabela 5.3, a malha dessa simulação possui 2780 elementos e os seus obstáculos ocupam uma área igual a 17,67 unidades de área. Esse valor confere para a simulação 6 uma densidade de obstáculos θ igual a 0,8835.

Em cada uma das iterações executadas, os valores para as velocidades v_x e v_y, vorticidade ω_z e função corrente ψ foram calculados para todos os nós da malha dessa simulação. A visualização desses dados para a 1000ª iteração está disponível na figura 5.19.

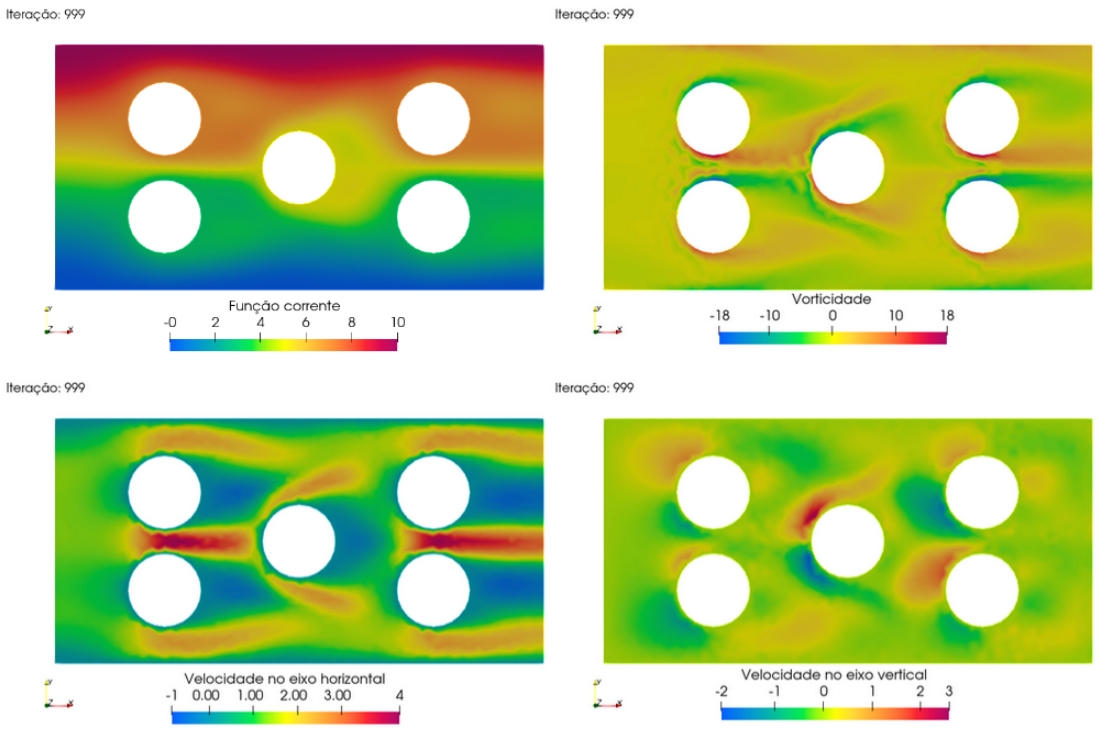

Figura 5.19: Resultados da simulação 6 mostrando os valores de v_x, v_y, ω_z e ψ calculados para a 1000ª iteração.

A função *Plot Over Line* foi utilizada para se obter os valores de v_x na mesma localização em que essa função foi executada para as simulações anteriores. A velocidade média de saída do escoamento calculada para a simulação 6 foi igual a 0,976362039. Isso confere para a simulação 6 um fator de qualidade de filtragem α com valor igual a 1,02421.

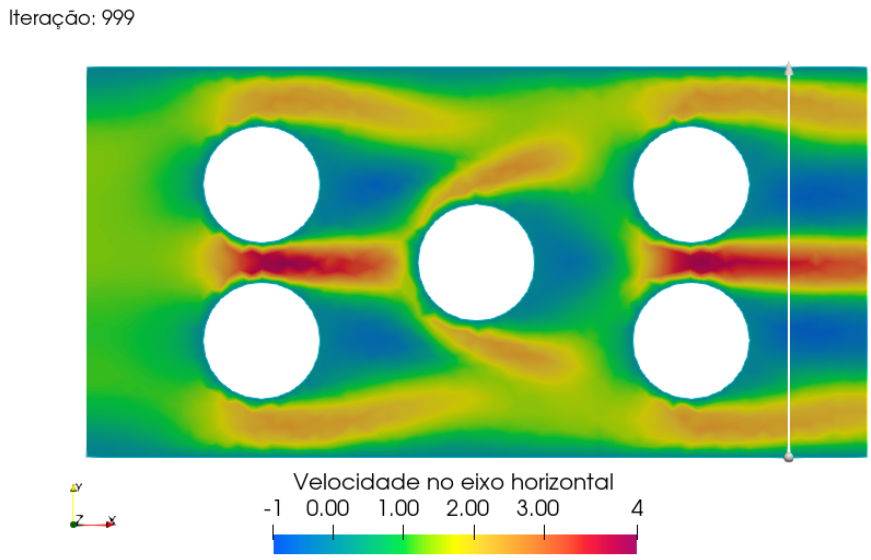

Figura 5.20: Resultado da simulação 6 para a sua 1000ª iteração e seta da função *Plot over line* localizada a 18 unidades de comprimento da entrada do canal mostrando onde essa função foi aplicada.

5.2.4 Análise de qualidade

Após obter os resultados das simulações 4, 5 e 6, é possível associar os valores do fator de qualidade α e os valores da densidade de obstáculos θ de cada simulação em um único gráfico. Esse gráfico pode ser consultado na figura 5.21 e foi criado utilizando **Python**.

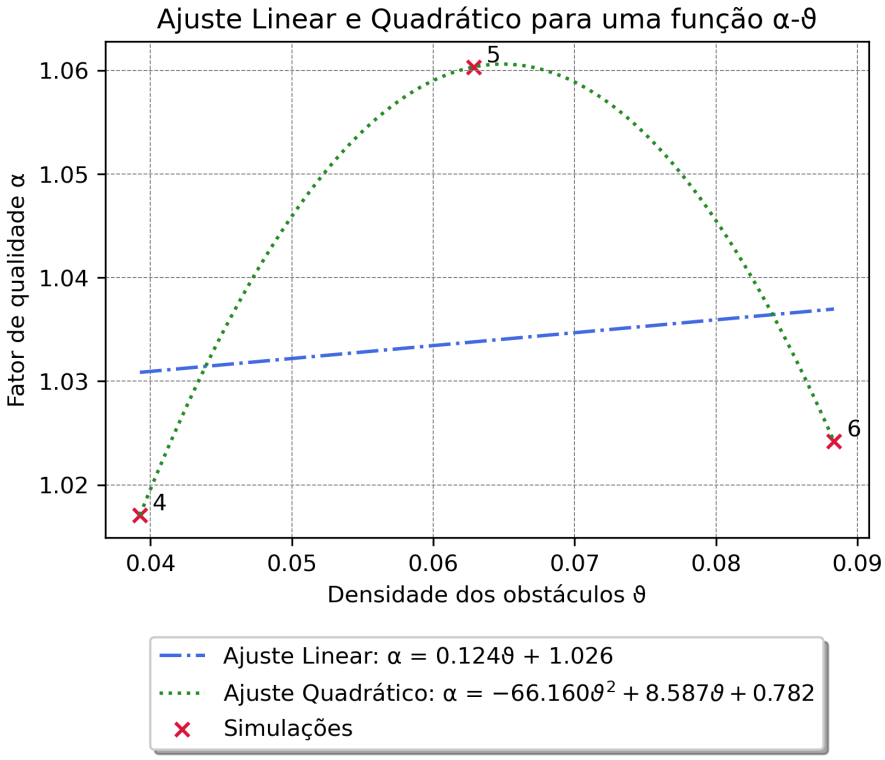

Figura 5.21: Dados de fator de qualidade α e densidade de obstáculos θ para as simulações 4, 5 e 6. Um ajuste linear e um ajuste quadrático foram feitos a partir desses dados para representar funções que relacionem esses dois parâmetros. As funções que descrevem ambos os ajustes podem ser visualizadas na legenda.

Além de indicar um ponto no gráfico para cada simulação, dois ajustes foram realizados para encontrar uma função que possa descrever as tendências demonstradas pelos dados. Os ajustes realizados foram um ajuste linear e outro quadrático. Assim como para o primeiro grupo de geometrias, é perceptível que a quantidade limitada de simulações dificulta interpretações conclusivas mas mesmo assim os dados disponíveis demonstram tendências interessantes por si só.

Os dados adquiridos para essas simulações reforçam as tendências já observadas pela análise de qualidade das primeiras três simulações. Novamente o ajuste linear indica uma relação proporcional entre a qualidade da filtragem e a densidade de obstáculos. Além disso, o ajuste quadrático reitera a tendência observada anteriormente na qual a densidade de obstáculos favorece a qualidade de filtragem até certo ponto e após esse ponto um aumento na densidade de obstáculos pode se tornar prejudicial à qualidade.

A queda do fator de qualidade na sexta simulação, em comparação com a simulação 5, pode ter ocorrido devido aos obstáculos maiores da simulação 6 acabarem criando "corredores"por onde o escoamento escapa sem que hajam obstáculos em seu caminho que o desacelerem. Essa hipótese evidenciaria um comportamento semelhante entre os escoamentos com obstáculos alinhados e os escoamentos com obstáculos desalinhados. O efeito responsável pela queda do fator de qualidade α na simulação 6 pode estar acontecendo porque nessa situação temos um número menor de obstáculos no canal e, com porções maiores das seções verticais do canal ocupadas, o escoamento atinge maiores velocidades do que nos casos anteriores nas localizações desses "corredores"do canal sem obstáculos.

Mesmo que esse seja um fenômeno não intuitivo, existem explicações possíveis para o comportamento de queda do fator de qualidade na simulação 6. Para uma melhor compreensão desse fenômeno, o ideal seria que mais simulações com esse tipo de geometria fossem realizadas para que mais pontos pudessem ser acrescentados a esse ajuste. Isso contribuiria para identificar com mais precisão a relação entre o fator de qualidade α e a densidade de obstáculos θ. Infelizmente, não foi possível realizar mais simulações e aprofundar esses estudos devido ao limitado poder computacional disponível no computador pessoal em que estar simulações foram realizadas.

5.3 Geometrias com obstáculos aleatórios

A terceira e última abordagem geométrica para estudar filtros de partículas consistiu em posicionar obstáculos sólidos circulares de diferentes raios em diferentes posições do canal sem padrão aparente. A distribuição adotada para o sorteio dos valores aleatórios conferidos para os raios e para as coordenadas horizontais e verticais de cada obstáculo foi uma distribuição uniforme, conferida pelo pacote *random* para o **Python**.

No entanto, algumas restrições foram impostas para os valores sorteados: os obstáculos não poderiam se interceptar entre si nem interceptar o canal; as bordas dos obstáculos precisavam manter uma distância de no mínimo uma unidade de comprimento entre eles mesmos e entre eles e o canal; e um par de obstáculos foi

sorteado antes de todos os outros com uma distância obrigatória de uma unidade de comprimento para garantir que a distância mínima entre dois obstáculos fosse igual a um. Essa última restrição foi imposta para que o valor escolhido para o comprimento característico do escoamento pudesse ser um.

Outro aspecto que diferencia as simulações dessa categoria para as demais é que em cada simulação o valor do raio dos obstáculos não é constante. Esses valores são sorteados para cada obstáculo dentro de um intervalo e cada simulação adota um intervalo diferente. Assim, para a Simulação 7, esses intervalos conferem uma variação do diâmetro dos obstáculos entre 1 e 2. Para a simulação 8, os diâmetros dos obstáculos variam entre 2 e 3. E, para a simulação 9, os diâmetros variam entre 3 e 4. Isso confere um valor de diâmetro médio diferente para cada simulação. Além dessa variação, o número de obstáculos de cada simulação também variou.

Os valores encontrados para o diâmetro médio, número de obstáculos e outras informações de cada simulação podem ser consultados na tabela 5.4. Os parâmetros da tabela 5.4 se somam aos parâmetros globais da tabela 5.1.

Simulação	**Diâmetro médio dos obstáculos**	**Nº de obstáculos**	**Elementos da malha**	**Área ocupada por obstáculos**
7	1,31	17	3001	12.027
8	2,54	8	2994	20.528
9	3,54	5	3036	24.829

Tabela 5.4: Parâmetros que caracterizam simulações realizadas com uma distribuição aleatória de obstáculos.

As geometrias e malhas das simulações 7, 8 e 9 estão expostas abaixo e os resultados obtidos para os cálculos da velocidade, vorticidade e função corrente do escoamento se apresentam nos subtópicos seguintes. A análise de resultados também será feita nessas seções.

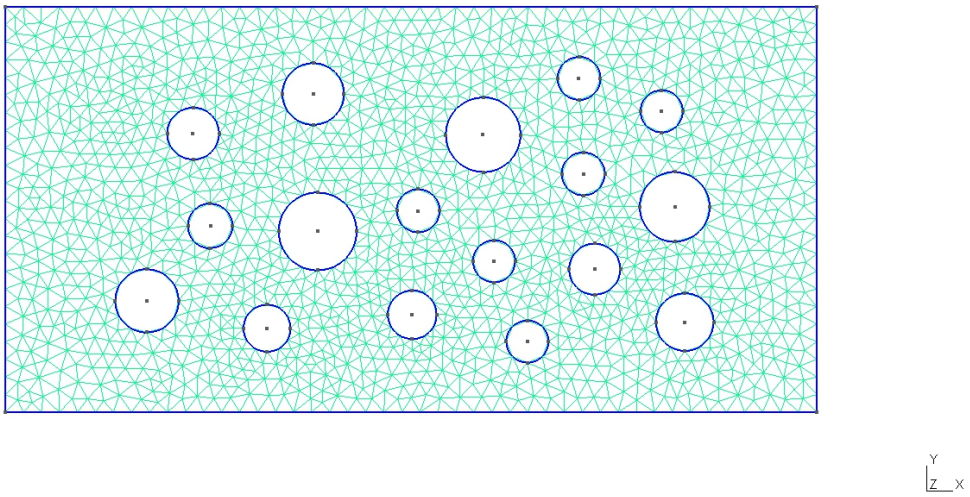

Figura 5.22: Geometria e malha feita para a simulação 7. As dimensões do canal são 10 × 20 e o diâmetro dos obstáculos varia entre 1 e 2. O diâmetro médio para essa geometria é igual a 1,31.

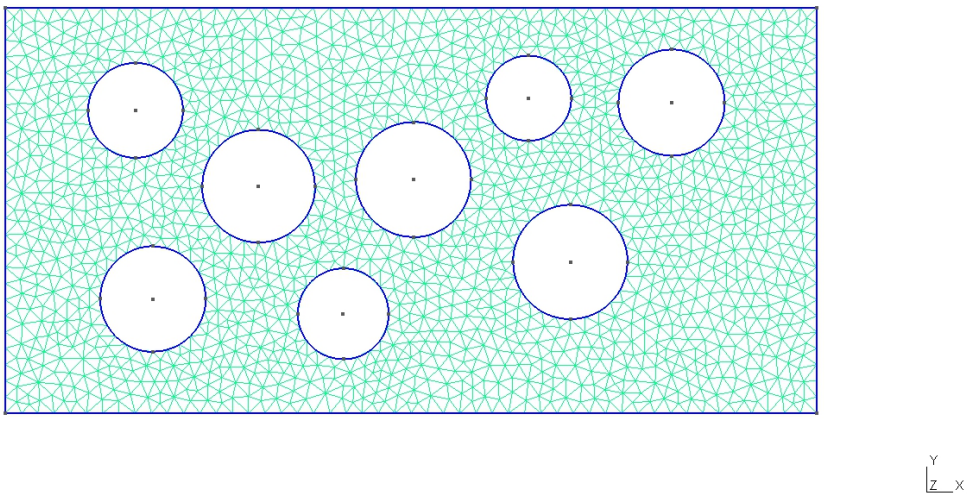

Figura 5.23: Geometria e malha feita para a simulação 8. As dimensões do canal são 10 × 20 e o diâmetro dos obstáculos varia entre 2 e 3. O diâmetro médio para essa geometria é igual a 2,54.

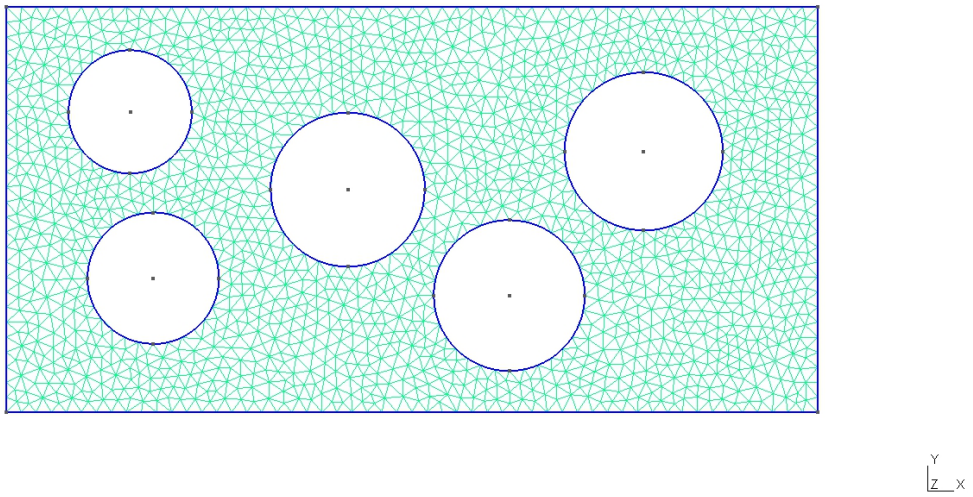

Figura 5.24: Geometria e malha feita para a simulação 9. As dimensões do canal são 10 × 20 e o diâmetro dos obstáculos varia entre 3 e 4. O diâmetro médio para essa geometria é igual a 3,54.

5.3.1 Obstáculos com diâmetros entre 1 e 2

A simulação 7 possui 17 obstáculos circulares e sólidos cujos diâmetros possuem valores variados entre 1 e 2. O diâmetro médio dos obstáculos dessa simulação é 1,31, conforme a tabela 5.4. A malha dessa simulação é composta por 3001 elementos e a área ocupada pelos obstáculos circulares dentro do canal de simulação é igual a 12,027 unidades de área. Esse valor confere para a simulação 7 uma densidade de obstáculos θ igual a 0,060135.

Em cada uma das iterações executadas, os valores para as velocidades v_x e v_y, vorticidade ω_z e função corrente ψ foram calculados para todos os nós da malha dessa simulação. A visualização desses dados para a 1000ª iteração está disponível na figura 5.25.

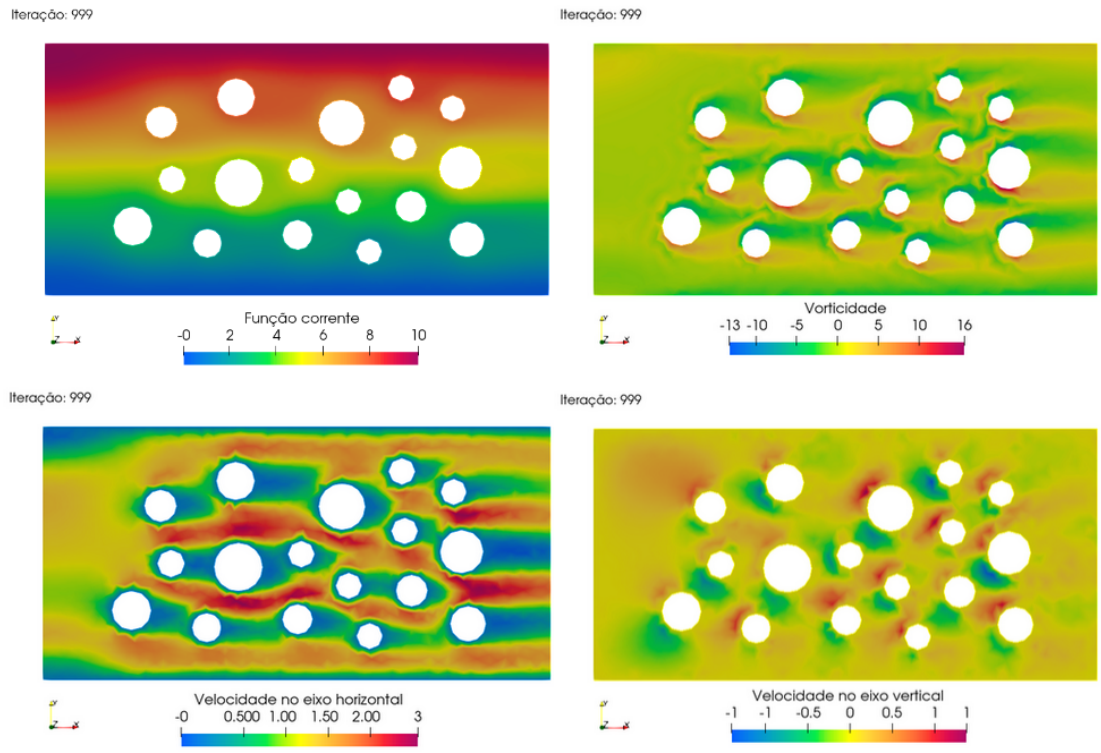

Figura 5.25: Resultados da simulação 7 mostrando os valores de v_x, v_y, ω_z e ψ calculados para a 1000ª iteração.

A função *Plot Over Line* foi utilizada para se obter os valores de v_x na mesma localização em que essa função foi executada para as simulações anteriores. A velocidade média de saída do escoamento calculada para a simulação 7 foi igual a 0,971563085. Isso confere para a simulação 7 um fator de qualidade de filtragem α com valor igual a 1,02927.

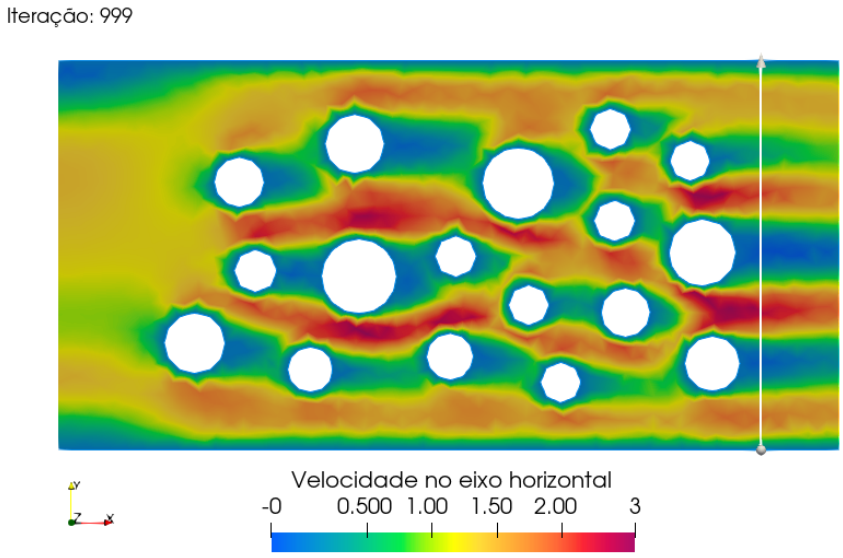

Figura 5.26: Resultado da simulação 7 para a sua 1000ª iteração e seta da função *Plot over line* localizada a 18 unidades de comprimento da entrada do canal mostrando onde essa função foi aplicada.

5.3.2 Obstáculos com diâmetros entre 2 e 3

A simulação 8 possui 8 obstáculos circulares e sólidos cujos diâmetros possuem valores variados entre 2 e 3. O diâmetro médio dos obstáculos dessa simulação é 2,54, conforme a tabela 5.4. A malha dessa simulação é composta por 2994 elementos e a área ocupada pelos obstáculos circulares dentro do canal de simulação é igual a 20,528 unidades de área. Esse valor confere para a simulação 8 uma densidade de obstáculos θ igual a 0,10264.

Em cada uma das iterações executadas, os valores para as velocidades v_x e v_y, vorticidade ω_z e função corrente ψ foram calculados para todos os nós da malha dessa simulação. A visualização desses dados para a 1000ª iteração está disponível na figura 5.27.

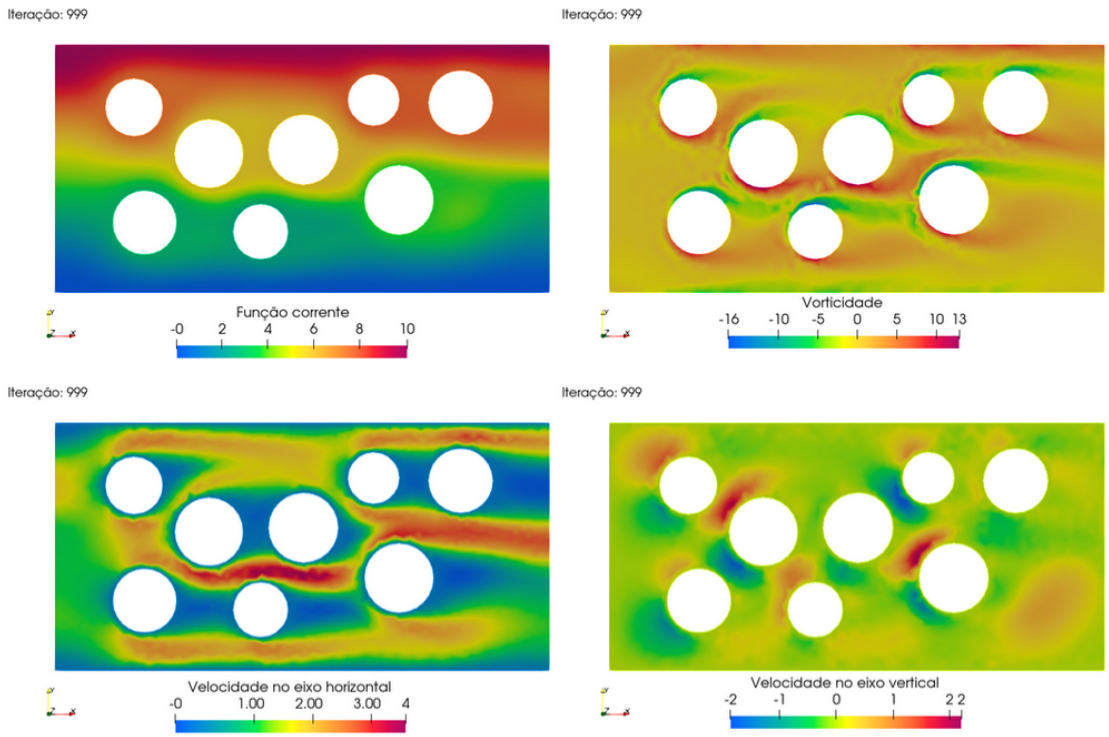

Figura 5.27: Resultados da simulação 8 mostrando os valores de v_x, v_y, ω_z e ψ calculados para a 1000ª iteração.

A função *Plot Over Line* foi utilizada para se obter os valores de v_x na mesma localização em que essa função foi executada para as simulações anteriores. A velocidade média de saída do escoamento calculada para a simulação 8 foi igual a 0,977812241. Isso confere para a simulação 8 um fator de qualidade de filtragem α com valor igual a 1,02269123.

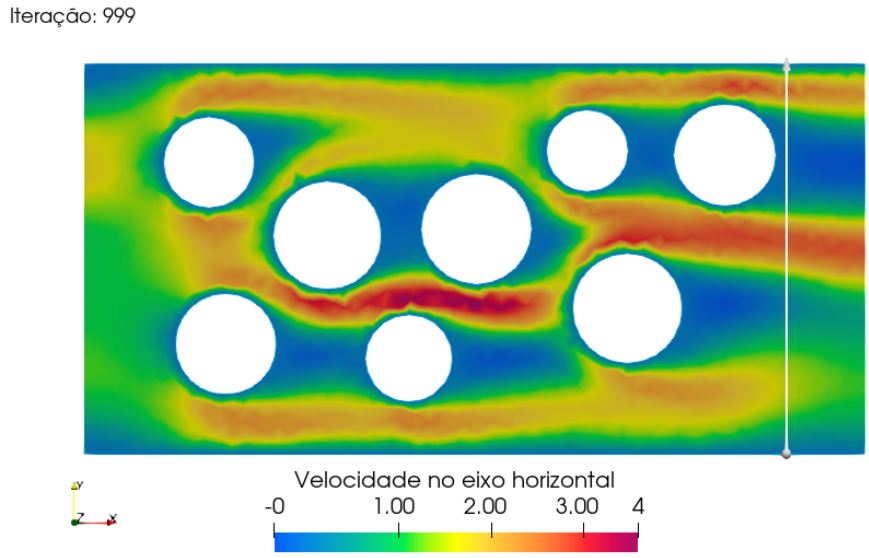

Figura 5.28: Resultado da simulação 8 para a sua 1000ª iteração e seta da função *Plot over line* localizada a 18 unidades de comprimento da entrada do canal mostrando onde essa função foi aplicada.

5.3.3 Obstáculos com diâmetro entre 3 e 4

A simulação 9 possui 5 obstáculos circulares e sólidos cujos diâmetros possuem valores variados entre 3 e 4. O diâmetro médio dos obstáculos dessa simulação é 3,54, conforme a tabela 5.4. A malha dessa simulação é composta por 3036 elementos e a área ocupada pelos obstáculos circulares dentro do canal de simulação é igual a 24,829 unidades de área. Esse valor confere para a simulação 9 uma densidade de obstáculos θ igual a 0,124145.

Em cada uma das iterações executadas, os valores para as velocidades v_x e v_y, vorticidade ω_z e função corrente ψ foram calculados para todos os nós da malha dessa simulação. A visualização desses dados para a 1000ª iteração está disponível na figura 5.29.

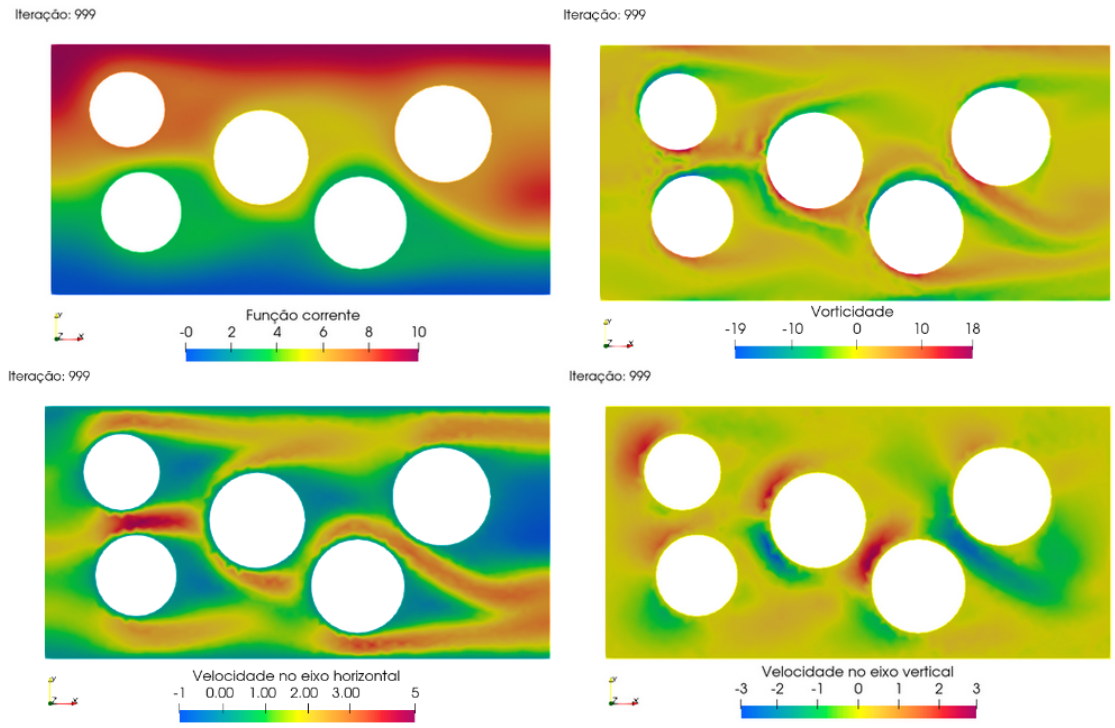

Figura 5.29: Resultados da simulação 9 mostrando os valores de v_x, v_y, ω_z e ψ calculados para a 1000ª iteração.

A função *Plot Over Line* foi utilizada para se obter os valores de v_x na mesma localização em que essa função foi executada para as simulações anteriores. A velocidade média de saída do escoamento calculada para a simulação 9 foi igual a 0,982888832. Isso confere para a simulação 9 um fator de qualidade de filtragem α com valor igual a 1,017409057.

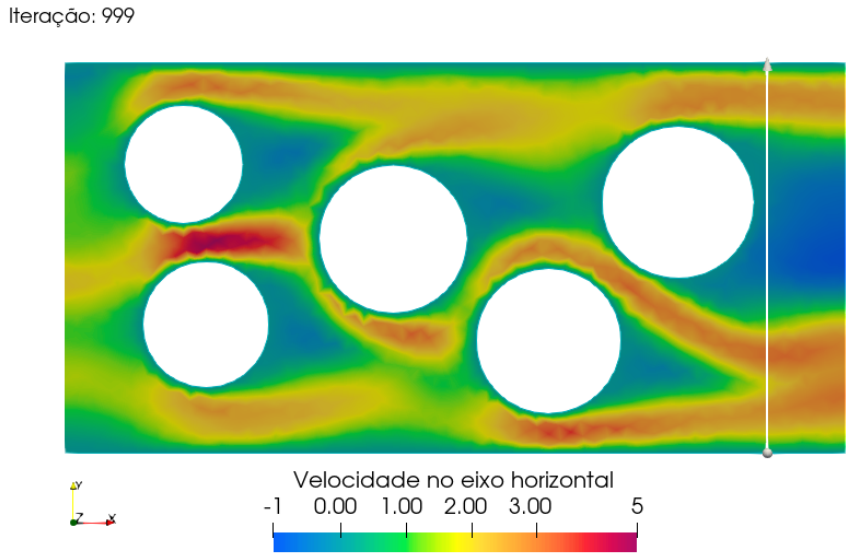

Figura 5.30: Resultado da simulação 9 para a sua 1000ª iteração e seta da função *Plot over line* localizada a 18 unidades de comprimento da entrada do canal mostrando onde essa função foi aplicada.

5.3.4 Análise de qualidade

Após obter os resultados das simulações 7, 8 e 9, é possível associar os valores do fator de qualidade α e os valores da densidade de obstáculos θ de cada simulação em um único gráfico. Esse gráfico pode ser consultado na figura 5.31 e foi criado utilizando **Python**.

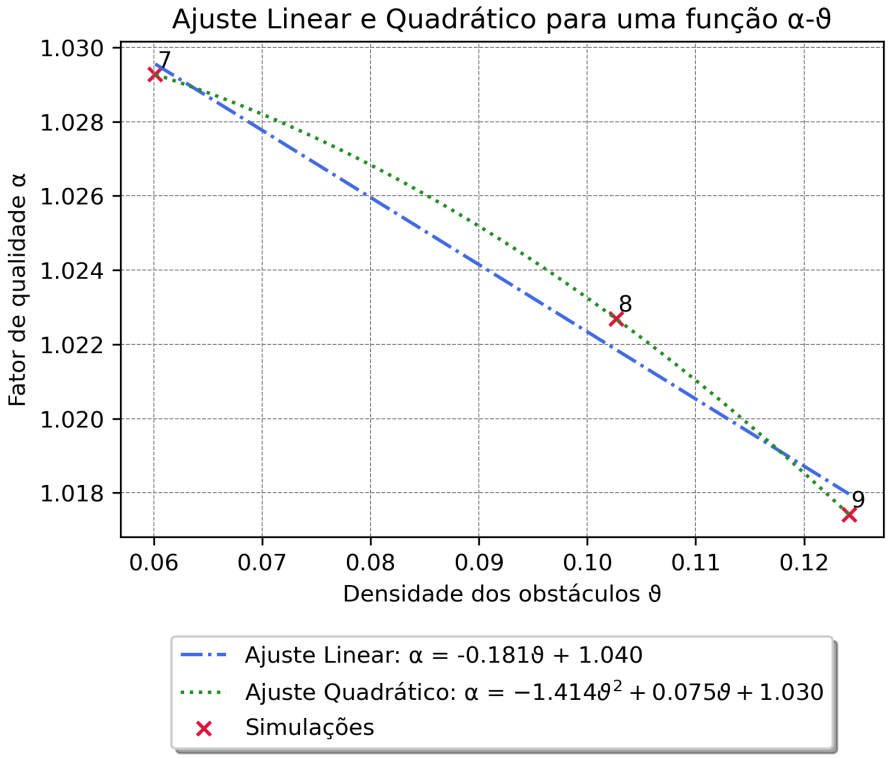

Figura 5.31: Dados de fator de qualidade α e densidade de obstáculos θ para as simulações 7, 8 e 9. Um ajuste linear e um ajuste quadrático foram feitos a partir desses dados para representar funções que relacionem esses dois parâmetros. As funções que descrevem ambos os ajustes podem ser visualizadas na legenda.

Além de indicar um ponto no gráfico para cada simulação, dois ajustes foram realizados para encontrar uma função que possa descrever as tendências demonstradas pelos dados. Os ajustes realizados foram um ajuste linear e outro quadrático. É evidente que o número limitado de simulações novamente restringe as interpretações que podem ser elaboradas a partir desses dados. No entanto, os resultados disponíveis demonstram tendências interessantes mesmo assim.

É curioso observar como, no caso dos obstáculos dispostos aleatoriamente pelo canal, as tendências observadas foram praticamente opostas das observadas para os outros grupos de simulações. O ajuste linear, por exemplo, mostra uma tendência de diminuição do fator de qualidade α na medida em que a densidade dos obstáculos aumenta. Além disso, o ajuste quadrático não parece tão representativo para esse grupo de simulações quanto o ajuste linear e não contribui para criar novos entendimentos sobre o fenômeno.

É possível que a natureza aleatória da distribuição dos obstáculos seja responsável por essa diferença de resultados e que, para esses casos, obstáculos menores sejam mais eficientes. Essa interpretação pode insinuar que distribuições aleatórias e padronizadas de obstáculos criam duas categorias diferentes de filtros. Filtros cuja porosidade pode ser emulada por obstáculos de mesmo tamanho distribuídos de forma alinhada e desalinhada possuiriam uma relação proporcional entre a qualidade da filtragem α e a densidade de obstáculos θ. Por outro lado, a segunda categoria seria composta por filtros cuja porosidade emulada por obstáculos sólidos dispostos de maneira aleatória e com tamanhos aleatórios possuiriam um outro comportamento. O comportamento dos filtros dessa segunda categoria seria o observado na imagem 5.31, ou seja, uma relação inversamente proporcional entre o fator de qualidade de filtragem α e a densidade de obstáculos θ.

É possível que, para os filtros de geometrias com obstáculos aleatórios serem mais eficientes, obstáculos sólidos com uma menor granulometria devam ser utilizados. No entanto, devido ao número limitado de simulações por conta das limitações computacionais do presente trabalho, difícil interpretar esses dados de forma conclusiva. Mesmo que esses resultados iniciais indiquem algumas tendências, é desejável que em trabalhos futuros mais simulações sejam realizadas para que as análises de qualidade possam ser mais esclarecedoras.

Capítulo 6

Conclusões

Neste trabalho de conclusão de curso, foi realizada uma breve revisão sobre o papel dos biocombustíveis nas atuais iniciativas de descarbonização das atividades de diversos países, incluindo ao Brasil. Nesse contexto, os filtros de partículas foram introduzidos como recursos interessantes que podem contribuir para uma diminuição ainda maior de emissões.

Para estudar esses filtros, foi necessário um estudo de conceitos indispensáveis da mecânica dos fluidos. Tendo como base esses conceitos, as equações da continuidade e de conservação do momento, no contexto da mecânica dos fluidos, foram deduzidas e apresentadas. A partir delas, as equações de Navier-Stokes foram definidas em sua forma dimensional e adimensional.

Por meio dessas equações, muitas variáveis importantes de escoamentos podem ser calculadas. No entanto, para evitar o acoplamento entre velocidade e pressão, escolheu-se trabalhar com a formulação corrente-vorticidade. Essa formulação foi então deduzida a partir das equações de Navier-Stokes em seu formato mais conhecido.

Depois dessas elaborações, a metodologia adotada para estudar as geometrias de filtros de partículas foi apresentada. A essência dessa metodologia consiste em, a partir de geometrias e malhas geradas, aplicar o MEF utilizando a formulação corrente-vorticidade para calcular variáveis de interesse do escoamento. Essas variáveis serão utilizadas para estudar a qualidade de filtragem de cada geometria escolhida. Portanto, foram definidos dois novos parâmetros: o fator de qualidade α e a densidade de obstáculos θ.

Para a verificação do código numérico desenvolvido, que utiliza MEF com a formulação corrente-vorticidade, dois escoamentos clássicos foram reproduzidos: o escoamento entre placas planas e o *lid-driven cavity*. O resultado de ambas as reproduções proporcionou uma verificação satisfatória, corroborando a metodologia adotada.

Após a verificação, os resultados numéricos obtidos para 9 simulações foram apresentados. Apesar da quantidade limitada de simulações, foi possível chegar a conclusões interessantes. Por meio dos resultados, evidenciou-se dois tipos de relações diferentes entre o fator de qualidade α e a densidade de obstáculos θ. Para as simulações com obstáculos circulares alinhados e desalinhados, se mostrou uma relação proporcional entre a qualidade da filtragem e a área ocupada pelos obstáculos. Ou seja, o escoamento seria mais desacelerado com a presença de mais obstáculos ocupando a área do canal.

Essa tendência, no entanto, não foi observada para as simulações com obstáculos distribuídos de forma aleatória. Na verdade, para essas simulações, se mostrou a relação inversa. Ou seja, quanto maior a área ocupada pelos obstáculos, menor o fator de qualidade do filtro. Essa observação levanta a hipótese de que distribuições aleatórias de obstáculos provoquem um comportamento diferente nos escoamentos em filtros, em comparação com distribuições mais padronizadas.

É importante destacar, entretanto, que os estudos elaborados possuem suas limitações. Por exemplo, os resultados podem ter sido influenciados pela quantidade limitada de simulações realizadas. Além disso, o comprimento do canal e a quantidade de obstáculos nas simulações podem não ter sido grande o suficiente para evidenciar uma diminuição significativa de velocidade do escoamento. É possível, inclusive, que as relações definidas por α e θ não sejam as mais relevantes para qualificar a qualidade de um filtro de partículas. Por isso, trabalhos futuros podem enriquecer essa discussão por meio de estudos como: descrever novas relações que correlacionem a geometria de um canal com a velocidade do escoamento, análise do efeito de números de Reynolds maiores nesse contexto, estudo de novos formatos de obstáculos e implementação de uma interface gráfica que facilite a execução das simulações.

Referências Bibliográficas

MARCHI, C. H., SUERO, R., ARAKI, L. K. "The lid-driven square cavity flow: numerical solution with a 1024 x 1024 grid", *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, v. 31, pp. 186–198, 2009.

IPCC. "Summary for Policymakers". In: Pörtner, H. O., Roberts, D. C., Tignor, M., et al. (Eds.), *Climate Change 2022: Impacts, Adaptation, and Vulnerability. Contribution of Working Group II to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change*, p. In Press, Cambridge, UK, Cambridge University Press, 2022. In Press.

METAL-5. "THE PARTICULATE FILTER: TECHNICAL EXPLANATIONS AND SERVICING TIPS - Metal5 - metal5.com". https://metal5.com/en/2021/10/05/the-particulate-filter-technical-explanations-and-servicing-tips/. [Accesso em 05/10/2023].

CERRI, C. E. P., YOU, X., CHERUBIN, M. R., et al. "Assessing the greenhouse gas emissions of Brazilian soybean biodiesel production", *PLOS ONE*, v. 12, n. 5, pp. 1–14, 05 2017. doi: 10.1371/journal.pone.0176948. Disponível em: <https://doi.org/10.1371/journal.pone.0176948>.

MINISTÉRIO DE MINAS E ENERGIA – MME., EPE, C. *Plano Nacional de Energia*. Ministério de Minas e Energia, 2020.

DE SOUZA, J. P. I. "Particle-Laden Multiphase FLows: A Finite Element Analysis on Biofuel Particle Emissions". 2021.

DE CERQUEIRA, J. C. "Análise Numérica do Escoamento de Resíduos de Biocombustíveis em Banco de Tubos". 2022.

SPESANI, D. M. "Análise Numérica do Escoamento de Biocombustíveis em um Meio Poroso". 2023.

O MECÂNICO. "Filtro de partículas em motores diesel elimina a fumaça preta". https://omecanico.com.br/filtro-de-particulas-em-motores-diesel-elimina-a-fumaca-preta/, 2020. Accesso em 20/09/2023.

SEONG, H., CHOI, S., MATUSIK, K. E., et al. "3D pore analysis of gasoline particulate filters using X-ray tomography: impact of coating and ash loading", *Journal of Materials Science*, v. 54, n. 8, pp. 6053–6065, 2019.

ROSSOMANDO, B., MELONI, E., DE FALCO, G., et al. "Experimental characterization of ultrafine particle emissions from a light-duty diesel engine equipped with a standard DPF", *Proceedings of the Combustion Institute*, v. 38, n. 4, pp. 5695–5702, 2021. ISSN: 1540-7489. doi: https://doi.org/10.1016/j.proci.2020.09.011. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1540748920306696>.

MESQUIDA, I. M. V. "Analysis of Flow Pattern in a Gasoline Particulate Filter using CFD". 2019.

ARBEX, M. A., SANTOS, U. D. P., MARTINS, L. C., et al. "A poluição do ar e o sistema respiratório", *Jornal Brasileiro de Pneumologia*, v. 38, pp. 643–655, 2012.

RODRÍGUEZ-FERNÁNDEZ, J., LAPUERTA, M., SÁNCHEZ-VALDEPEÑAS, J. "Regeneration of diesel particulate filters: Effect of renewable fuels", *Renewable Energy*, v. 104, pp. 30–39, 2017. ISSN: 0960-1481. doi: https://doi.org/10.1016/j.renene.2016.11.059. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0960148116310436>.

ANDERSON, J. *Computational Fluid Dynamics: The Basics with Applications*. McGraw-Hill International Editions: Mechanical Engineering. United States, McGraw-Hill, 1995. ISBN: 9780071132107. Disponível em: <https://books.google.com.br/books?id=phG_QgAACAAJ>.

DE SOUZA, R. "O Método dos Elementos Finitos Aplicado ao Problema de Condução de Calor". 05 2003.

DA CUNHA, L. H. C. "ALE Finite Element Method for Simulating Flows with the Streamfunction-Vorticity Formulation". 2020.

MOHAMED ABDELWAHED, NEJMEDDINE CHORFI, M. H. "A Stabilized Finite Element Method for Stream Function Vorticity Formulation of Navier-Stokes Equations", *Electronic Journal of Differential Equations*, v. 2017 (2017), n. 24, pp. 1–10, 2017.

CARNEVALE, L., ANJOS, G., MANGIAVACCHI, N. "Stream Function-Vorticity Formulation Applied in the Conjugated Heat Problem Using the FEM With Unstructured Mesh". 11 2018. doi: 10.26678/ABCM.ENCIT2018.CIT18-0173.

ANJOS, G. R. "Computação Científica para Engenheiros." Disponível em: <https://gustavorabello.github.io/teaching/>.

SHIH, T., TAN, C., HWANG, B. "Effects of grid staggering on numerical schemes", *International Journal for numerical methods in fluids*, v. 9, n. 2, pp. 193–212, 1989.

Apêndice A

Código para geração de geometria com obstáculos circulares alinhados

```
# -*- coding: utf-8 -*-
"""
Created on Mon Jul 31 16:46:03 2023

@author: Anna Bárbara

para criar circulos como obstaculos alinhados em linhas e colunas

"""


# dimensoes do canal
comprimento = 20
altura = 10

# refinamento da malha
lc = 1.0

# primeiro circulo
xc = 4
yc = 2

```

```
23 # raio dos circulos
24 raio = 0.5
25
26 # numero de colunas verticais e numero de linhas horizontais
27 nv = 4
28 nh = 7
29
30 # permeabilidade vertical e horizontal
31 kv = 2*raio +1 #espaco entre circulos da mesma coluna
32 kh = 2*raio +1 #espaço entre circulos da mesma linha
33
34 # criando listas
35 x_list = []
36 y_list = []
37 r_list = []
38 z_list = []
39
40
41 # inicializando as coordenadas dos circulos
42 for v in range(0,nv):
43     for h in range(0,nh):
44         x_list.append(xc + h*kh)
45         y_list.append(yc + v*kv)
46         r_list.append(raio)
47         z_list.append(0)
48
49
50 # ou usando list comprehension
51 #x_list = [xc + h * kh for v in range(nv) for h in range(nh)]
52 #y_list = [yc + v * kv for v in range(nv) for h in range(nh)]
53 #r_list = [raio] * (nv * nh)  # O raio é constante para todos os cí
    rculos
54 #z_list = [0] * (nv * nh)  # Valor fixo de 0 para coordenada z de todos
     os círculos
55
56
57 # numero de circulos que quero gerar
58 numero_circulos = nv*nh
59
```

```
60
61 r_list_rounded = [ round(elem, 2) for elem in r_list ]
62 x_list_rounded = [ round(elem, 2) for elem in x_list ]
63 y_list_rounded = [ round(elem, 2) for elem in y_list ]
64 z_list_rounded = [ round(elem, 2) for elem in z_list ]
65
66 print(numero_circulos)
67 print("r_list = " + str(r_list_rounded[:]))
68 print("x_list = " + str(x_list_rounded[:]))
69 print("y_list = " + str(y_list_rounded[:]))
70 print("z_list = " + str(z_list_rounded[:]))
71
72
73 # comecando a escrever o script
74
75 x1 = """//channel dimensions
76 length = {comprimento};
77 height = {altura};
78 lc = {lc};
79 //+
80 Point(1) = {{0, 0, 0, lc}};
81 Point(2) = {{length, 0, 0, lc}};
82 Point(3) = {{length, height, 0, lc}};
83 Point(4) = {{0, height, 0, lc}};
84 //+
85 Line(1) = {{1, 2}};
86 Line(2) = {{2, 3}};
87 Line(3) = {{3, 4}};
88 Line(4) = {{4, 1}};
89 //+
90 Line Loop(1) = {{3, 4, 1, 2}};
91 //+
92 """.format(comprimento =str(comprimento), altura=str(altura),lc = str(
       lc))
93
94
95 with open("circ_alinhados_para_gmsh.txt","w+") as f:
96     f.writelines(x1)
97
```

```
98
99  # escrevendo os pontos
100
101 with open('circ_alinhados_para_gmsh.txt', 'a') as f:
102     for i in range(numero_circulos):
103         j = i * 5 + 5  # Correctly update j for each set of points
104         x = f"""Point({j}) = {{{x_list[i]}, {y_list[i]}, {z_list[i]},
        lc}};
105 //+
106 Point({j+1}) = {{{x_list_rounded[i] + r_list_rounded[i]}, {
        y_list_rounded[i]}, {z_list[i]}, lc}};
107 //+
108 Point({j+2}) = {{{x_list_rounded[i]}, {y_list_rounded[i] +
        r_list_rounded[i]}, {z_list[i]}, lc}};
109 //+
110 Point({j+3}) = {{{x_list_rounded[i] - r_list_rounded[i]}, {
        y_list_rounded[i]}, {z_list[i]}, lc}};
111 //+
112 Point({j+4}) = {{{x_list_rounded[i]}, {y_list_rounded[i] -
        r_list_rounded[i]}, {z_list[i]}, lc}};
113 //+
114 """
115         f.writelines(x)
116
117
118 # escrevendo os circulos
119
120 with open('circ_alinhados_para_gmsh.txt', 'a') as f:
121     for i in range(numero_circulos):
122         a = i * 4 + 5
123         c = i * 5 + 5
124         b = c + 1
125         d = c + 2
126         x = f"""Circle({a}) = {{{b},{c},{d}}};
127 //+
128 Circle({a+1}) = {{{b+1},{c},{d+1}}};
129 //+
130 Circle({a+2}) = {{{b+2},{c},{d+2}}};
131 //+
```

```
Circle({a+3}) = {{{b+3},{c},{d-1}}};
//+
"""
        f.writelines(x)

# fazendo line loops

with open('circ_alinhados_para_gmsh.txt', 'a') as f:
    for i in range(numero_circulos+1):
        a = i * 4 + 1
        b = i * 4 + 2
        c = i * 4 + 3
        d = i * 4 + 4
        x = f"""Line Loop({i+2}) = {{{a},{b},{c},{d}}};
//+
"""
        f.writelines(x)

# definindo plane surface

lista = [];
for i in range (2,numero_circulos+3):
    lista.append(i)


string = "Plane Surface(1) = {"

for item in lista:
    string += str(item) + ", "

# tirando o ultimo espaco e a virgula
string = string[:-2]

string += "};\n//+"

with open('circ_alinhados_para_gmsh.txt', 'a') as f:
        x = string
        f.writelines(x)

```

```
171
172
173 # definindo physical lines do contorno do canal
174
175 with open('circ_alinhados_para_gmsh.txt', 'a') as f:
176     x2 = """Physical Line("inlet") = {4};
177 //+
178 Physical Line("outlet") = {2};
179 //+
180 Physical Line("paredeSup") = {3};
181 //+
182 Physical Line("paredeInf") = {1};
183 //+
184 """
185     f.writelines(x2)
186
187 # definindo physical lines dos obstaculos
188
189 with open('circ_alinhados_para_gmsh.txt', 'a') as f:
190     for i in range(numero_circulos):
191         a = i * 4 + 5
192         b = i * 4 + 6
193         c = i * 4 + 7
194         d = i * 4 + 8
195         x = f"""Physical Line("obstaculo{i+1}") = {{{a},{b},{c},{d}}};
196 //+
197 """
198         f.writelines(x)
199
200
201 # definindo physical surface
202
203 with open('circ_alinhados_para_gmsh.txt', 'a') as f:
204         x = """Physical Surface("surface") = {1};"""
205         f.writelines(x)
206
207
208 # printando condicao de contorno para adicionar no codigo
    corrente Vorticidade2D
```

```
209
210 for i in range(numero_circulos):
211     x = f"""if ccName[i] == 'obstaculo{i+1}':
212       vx_cc[i] = 0.0
213       vy_cc[i] = 0.0
214       psi_cc[i] = {y_list[i]}
215 """
216     print(x)
217
218 # resetar listas
219
220 r_list.clear()
221 x_list.clear()
222 y_list.clear()
223 z_list.clear()
```

Apêndice B

Código para geração de geometria com obstáculos circulares desalinhados

```
# -*- coding: utf-8 -*-
"""
Created on Mon Jul 31 16:46:03 2023

@author: Anna Bárbara

para criar circulos como obstaculos DESalinhados em linhas e colunas

"""

# codigo para gerar geometrias alinhas cujas linhas pares tenham uma
    defasagem
# nesse documento raio = 0.5

# dimensoes do canal
comprimento = 20
altura = 10

# refinamento da malha
lc = 1.0

# raio dos circulos
```

```
22 raio = 0.5
23
24 # primeiro circulo do primeiro alinhamento
25 xc1 = 3
26 yc1 = 1.5
27
28 # primeiro circulo do segundo alinhamento
29 xc2 = 6.5
30 yc2 = 2.5
31
32
33 # numero de colunas verticais e numero de linhas horizontais
34 nv1 = 4 # primeiro alinhamento
35 nh1 = 3 # primeiro alinhamento
36
37 nv2 = 4
38 nh2 = 2
39
40 # definindo L que vai ser a distancia entre as bordas dos circulos na
       mesma linha e na mesma coluna
41
42 #L = (2*raio*sqrt(2)*(2-sqrt(2)) +sqrt(2))/2
43
44 # permeabilidade vertical e horizontal
45 kv = 2* raio + 1 #espaco entre circulos da mesma coluna
46 kh = 7 #espaço entre circulos da mesma linha
47 #defasagem = 1 #defasagem das linhas pares em relação as impares
48
49 # criando listas
50 x_list = []
51 y_list = []
52 r_list = []
53 z_list = []
54
55
56 # inicializando as coordenadas dos circulos do primeiro alinhamento
57 for v in range(0,nv1):
58     for h in range(0,nh1):
59         x_list.append(xc1 + h*kh)
```

```
60          y_list.append(yc1 + v*kv)
61          r_list.append(raio)
62          z_list.append(0)
63
64 # inicializando as coordenadas dos circulos do primeiro alinhamento
65 for v in range(0,nv2):
66      for h in range(0,nh2):
67          x_list.append(xc2 + h*kh)
68          y_list.append(yc2 + v*kv)
69          r_list.append(raio)
70          z_list.append(0)
71
72 # ou usando list comprehension
73 #x_list = [xc + h * kh for v in range(nv) for h in range(nh)]
74 #y_list = [yc + v * kv for v in range(nv) for h in range(nh)]
75 #r_list = [raio] * (nv * nh)  # O raio é constante para todos os cí
       rculos
76 #z_list = [0] * (nv * nh)  # Valor fixo de 0 para coordenada z de todos
       os círculos
77
78
79 # numero de circulos que quero gerar
80 numero_circulos = nv1*nh1 + nv2*nh2
81
82
83 r_list_rounded = [ round(elem, 2) for elem in r_list ]
84 x_list_rounded = [ round(elem, 2) for elem in x_list ]
85 y_list_rounded = [ round(elem, 2) for elem in y_list ]
86 z_list_rounded = [ round(elem, 2) for elem in z_list ]
87
88 print(numero_circulos)
89 print("r_list = " + str(r_list_rounded[:]))
90 print("x_list = " + str(x_list_rounded[:]))
91 print("y_list = " + str(y_list_rounded[:]))
92 print("z_list = " + str(z_list_rounded[:]))
93
94 # comecando a escrever o script
95
96 x1 = """//channel dimensions
```

```
length = {comprimento};
height = {altura};
lc = {lc};
//+
Point(1) = {{0, 0, 0, lc}};
Point(2) = {{length, 0, 0, lc}};
Point(3) = {{length, height, 0, lc}};
Point(4) = {{0, height, 0, lc}};
//+
Line(1) = {{1, 2}};
Line(2) = {{2, 3}};
Line(3) = {{3, 4}};
Line(4) = {{4, 1}};
//+
Line Loop(1) = {{3, 4, 1, 2}};
//+
""".format(comprimento =str(comprimento), altura=str(altura),lc = str(
    lc))


with open("circ_alinhados_para_gmsh.txt","w+") as f:
    f.writelines(x1)


# escrevendo os pontos

with open('circ_alinhados_para_gmsh.txt', 'a') as f:
    for i in range(numero_circulos):
        j = i * 5 + 5  # Correctly update j for each set of points
        x = f"""Point({j}) = {{{x_list[i]}, {y_list[i]}, {z_list[i]},
    lc}};
//+
Point({j+1}) = {{{x_list_rounded[i] + r_list_rounded[i]}, {
    y_list_rounded[i]}, {z_list[i]}, lc}};
//+
Point({j+2}) = {{{x_list_rounded[i]}, {y_list_rounded[i] +
    r_list_rounded[i]}, {z_list[i]}, lc}};
//+
Point({j+3}) = {{{x_list_rounded[i] - r_list_rounded[i]}, {
```

```
    y_list_rounded[i]}, {z_list[i]}, lc}};
//+
Point({j+4}) = {{{x_list_rounded[i]}, {y_list_rounded[i] -
    r_list_rounded[i]}, {z_list[i]}, lc}};
//+
"""
        f.writelines(x)


# escrevendo os circulos

with open('circ_alinhados_para_gmsh.txt', 'a') as f:
    for i in range(numero_circulos):
        a = i * 4 + 5
        c = i * 5 + 5
        b = c + 1
        d = c + 2
        x = f"""Circle({a}) = {{{b},{c},{d}}};
//+
Circle({a+1}) = {{{b+1},{c},{d+1}}};
//+
Circle({a+2}) = {{{b+2},{c},{d+2}}};
//+
Circle({a+3}) = {{{b+3},{c},{d-1}}};
//+
"""
        f.writelines(x)

# fazendo line loops

with open('circ_alinhados_para_gmsh.txt', 'a') as f:
    for i in range(numero_circulos+1):
        a = i * 4 + 1
        b = i * 4 + 2
        c = i * 4 + 3
        d = i * 4 + 4
        x = f"""Line Loop({i+2}) = {{{a},{b},{c},{d}}};
//+
"""
```

```
169         f.writelines(x)
170
171 # definindo plane surface
172
173 lista = [];
174 for i in range (2,numero_circulos+3):
175     lista.append(i)
176
177
178 string = "Plane Surface(1) = {"
179
180 for item in lista:
181     string += str(item) + ", "
182
183 # tirando o ultimo espaco e a virgula
184 string = string[:-2]
185
186 string += "};\n//+"
187
188 with open('circ_alinhados_para_gmsh.txt', 'a') as f:
189         x = string
190         f.writelines(x)
191
192
193
194 # definindo physical lines do contorno do canal
195
196 with open('circ_alinhados_para_gmsh.txt', 'a') as f:
197     x2 = """Physical Line("inlet") = {4};
198 //+
199 Physical Line("outlet") = {2};
200 //+
201 Physical Line("paredeSup") = {3};
202 //+
203 Physical Line("paredeInf") = {1};
204 //+
205 """
206     f.writelines(x2)
207
```

```
# definindo physical lines dos obstaculos

with open('circ_alinhados_para_gmsh.txt', 'a') as f:
    for i in range(numero_circulos):
        a = i * 4 + 5
        b = i * 4 + 6
        c = i * 4 + 7
        d = i * 4 + 8
        x = f"""Physical Line("obstaculo{i+1}") = {{{a},{b},{c},{d}}};
//+
"""
        f.writelines(x)


# definindo physical surface

with open('circ_alinhados_para_gmsh.txt', 'a') as f:
        x = """Physical Surface("surface") = {1};"""
        f.writelines(x)


# printando condicao de contorno para adicionar no codigo
    correnteVorticidade2D

for i in range(numero_circulos):
    x = f"""if ccName[i] == 'obstaculo{i+1}':
     vx_cc[i] = 0.0
     vy_cc[i] = 0.0
     psi_cc[i] = {y_list[i]}
"""
    print(x)

# resetar listas

r_list.clear()
x_list.clear()
y_list.clear()
z_list.clear()
```

Apêndice C

Código para geração de geometria com obstáculos circulares de posições aleatórias

```
# -*- coding: utf-8 -*-
"""
Created on Thu Jul 27 22:36:08 2023

@author: Anna Bárbara

Criando esse .py aqui para gerar circulos que nao se interceptam, de
    qualquer intervalo de raio e
distancia entre circulos maior do que 1 dentro de um canal

"""

import random
import math


# calcular distancia entre pontos
def distancia_entre_pontos(x1, y1, x2, y2):
    return math.sqrt((x1 - x2)**2 + (y1 - y2)**2)

# dimensoes do canal
comprimento = 20.0
```

```
22 altura = 10.0
23
24 # refinamento da malha
25 lc = 1.0
26
27 # numero de circulos que quero gerar
28 numero_circulos = 5
29
30 # intervalos para valores aleatorios
31 xMin = 3.0
32 xMax = 17.0
33 rMin = 1.5
34 rMax = 2.0
35
36 # criando listas
37 x_list = []
38 y_list = []
39 r_list = []
40 z_list = []
41
42
43 # criando os primeiros dois circulos antes de forma a garantir que eles
       necessariamente tenham uma distancia
44 # igual a um para que o comprimento caracteristico seja um
45
46 r1 = rMin + random.random()*(rMax - rMin)  # Intervalo para r1
47 x1 = xMin + random.random()*(xMax - xMin)
48 yMin1 = r1 + 1
49 yMax1 = altura - (r1 + 1)
50 y1 = yMin1 + random.random()*(yMax1 - yMin1)
51
52 r_list.append(r1)
53 x_list.append(x1)
54 y_list.append(y1)
55 z_list.append(0)
56
57 # inicializa uma flag para verificar se o segundo circulo esta dentro
      dos limites
58 segundo_circulo_valido = False
```

```
59
60 # loop para gerar o segundo circulo e garantir que esteja dentro dos
       limites
61 while not segundo_circulo_valido:
62     r2 = rMin + random.random()*(rMax - rMin)  # sorteia raio do
       segundo circulo
63
64     # calcula a distancia que eles tem que ter entre si
65     distancia = (r1 + r2) + 1
66
67     # define um angulo aleatorio para essa distancia
68     angle = random.uniform(0, 2 * math.pi)
69
70     # calcula as coordenadas para o segundo circulo
71     x2 = x1 + distancia * math.cos(angle)
72     yMin2 = r2 + 1
73     yMax2 = altura - (r2 + 1)
74     y2 = y1 + distancia * math.sin(angle)
75
76     # verifica se o segundo circulo esta dentro dos limites
77     if x2 > xMax or x2 < xMin or y2 > yMax2 or y2 < yMin2:
78         segundo_circulo_valido = False
79     else:
80         segundo_circulo_valido = True
81         # adiciona circulo válido para listas
82         r_list.append(r2)
83         x_list.append(x2)
84         y_list.append(y2)
85         z_list.append(0)
86
87 # parte que gera circulos que nao se cruzam, diam=1, distancia entre
       pontos tem que ser maior que raio + 1
88 # como distancia entre pontos eh maior que raios + 1, comprimento
       caract pode ser 1
89
90 while len(r_list) < numero_circulos:
91     r = rMin + random.random()*(rMax - rMin)
92     x = xMin + random.random()*(xMax-xMin)
93     yMin = r + 1  # tem que somar 1 para que a distancia entre o
```

```
    circulo e a parede seja maior do que 1 e comp caract poder ser 1
    yMax = altura - (r + 1)
    y = yMin + random.random()*(yMax-yMin)

    # verifica se o circulo atual nao se cruza com nenhum circulo
    anterior
    is_valid_circle = True
    for i in range(len(x_list)):
        d = distancia_entre_pontos(x, y, x_list[i], y_list[i])
        if d < 1 + (r + r_list[i]):
            is_valid_circle = False
            break

    # adiciona circulo valido para listas
    if is_valid_circle:
        r_list.append(r)
        x_list.append(x)
        y_list.append(y)
        z_list.append(0)


r_list_rounded = [ round(elem, 2) for elem in r_list ]
x_list_rounded = [ round(elem, 2) for elem in x_list ]
y_list_rounded = [ round(elem, 2) for elem in y_list ]
z_list_rounded = [ round(elem, 2) for elem in z_list ]


print("r_list = " + str(r_list_rounded[:]))
print("x_list = " + str(x_list_rounded[:]))
print("y_list = " + str(y_list_rounded[:]))
print("z_list = " + str(z_list_rounded[:]))



x1 = """//channel dimensions
length = {comprimento};
height = {altura};
lc = {lc};
//+
```

```
Point(1) = {{0, 0, 0, lc}};
Point(2) = {{length, 0, 0, lc}};
Point(3) = {{length, height, 0, lc}};
Point(4) = {{0, height, 0, lc}};
//+
Line(1) = {{1, 2}};
Line(2) = {{2, 3}};
Line(3) = {{3, 4}};
Line(4) = {{4, 1}};
//+
Line Loop(1) = {{3, 4, 1, 2}};
//+
""".format(comprimento =str(comprimento), altura=str(altura),lc = str(
    lc))


with open("diam_entre_3_4_8_circulos.txt","w+") as f:
    f.writelines(x1)


#escrevendo os pontos

with open('diam_entre_3_4_8_circulos.txt', 'a') as f:
    for i in range(numero_circulos):
        j = i * 5 + 5  # Correctly update j for each set of points
        x = f"""Point({j}) = {{{x_list[i]}, {y_list[i]}, {z_list[i]},
    lc}};
//+
Point({j+1}) = {{{x_list_rounded[i] + r_list_rounded[i]}, {
    y_list_rounded[i]}, {z_list[i]}, lc}};
//+
Point({j+2}) = {{{x_list_rounded[i]}, {y_list_rounded[i] +
    r_list_rounded[i]}, {z_list[i]}, lc}};
//+
Point({j+3}) = {{{x_list_rounded[i] - r_list_rounded[i]}, {
    y_list_rounded[i]}, {z_list[i]}, lc}};
//+
Point({j+4}) = {{{x_list_rounded[i]}, {y_list_rounded[i] -
    r_list_rounded[i]}, {z_list[i]}, lc}};
```

```
164 //+
165 """
166         f.writelines(x)
167
168
169 #escrevendo os circulos
170
171 with open('diam_entre_3_4_8_circulos.txt', 'a') as f:
172     for i in range(numero_circulos):
173         a = i * 4 + 5
174         c = i * 5 + 5
175         b = c + 1
176         d = c + 2
177         x = f"""Circle({a}) = {{{b},{c},{d}}};
178 //+
179 Circle({a+1}) = {{{b+1},{c},{d+1}}};
180 //+
181 Circle({a+2}) = {{{b+2},{c},{d+2}}};
182 //+
183 Circle({a+3}) = {{{b+3},{c},{d-1}}};
184 //+
185 """
186         f.writelines(x)
187
188 # fazendo line loops
189
190 with open('diam_entre_3_4_8_circulos.txt', 'a') as f:
191     for i in range(numero_circulos+1):
192         a = i * 4 + 1
193         b = i * 4 + 2
194         c = i * 4 + 3
195         d = i * 4 + 4
196         x = f"""Line Loop({i+2}) = {{{a},{b},{c},{d}}};
197 //+
198 """
199         f.writelines(x)
200
201 # definindo plane surface
202
```

```
lista = [];
for i in range (2,numero_circulos+3):
    lista.append(i)


string = "Plane Surface(1) = {"

for item in lista:
    string += str(item) + ", "

#tirando o ultimo espaco e a virgula
string = string[:-2]

string += "};\n//+"

with open('diam_entre_3_4_8_circulos.txt', 'a') as f:
        x = string
        f.writelines(x)



# definindo physical lines do contorno do canal

with open('diam_entre_3_4_8_circulos.txt', 'a') as f:
    x2 = """Physical Line("inlet") = {4};
//+
Physical Line("outlet") = {2};
//+
Physical Line("paredeSup") = {3};
//+
Physical Line("paredeInf") = {1};
//+
"""
    f.writelines(x2)

# definindo physical lines dos obstaculos

with open('diam_entre_3_4_8_circulos.txt', 'a') as f:
    for i in range(numero_circulos):
```

```
242         a = i * 4 + 5
243         b = i * 4 + 6
244         c = i * 4 + 7
245         d = i * 4 + 8
246         x = f"""Physical Line("obstaculo{i+1}") = {{{a},{b},{c},{d}}};
247 //+
248 """
249         f.writelines(x)
250
251
252 # definindo physical surface
253
254 with open('diam_entre_3_4_8_circulos.txt', 'a') as f:
255         x = """Physical Surface("surface") = {1};"""
256         f.writelines(x)
257
258
259 # printando condicao de contorno para adicionar no codigo
    correnteVorticidade2D
260
261 for i in range(numero_circulos):
262     x = f"""if ccName[i] == 'obstaculo{i+1}':
263      vx_cc[i] = 0.0
264      vy_cc[i] = 0.0
265      psi_cc[i] = {y_list[i]}
266 """
267     print(x)
268
269 # teste
270
271 print(r_list[0], r_list[1])
272 print(x_list[0],x_list[1])
273 print(y_list[0], y_list[1])
274
275 # resetar listas
276
277 r_list.clear()
278 x_list.clear()
279 y_list.clear()
```

```
280 z_list.clear()
```

Apêndice D

Código numérico com formulação corrente-vorticidade e MEF para resolver escoamento

O código presente nesse apêndice foi executado para obter os resultados da simulação 7. Para executar as simulações, todos esses códigos são equivalentes. As únicas configurações que mudam são as condições de contorno dos obstáculos, o número de obstáculos e o nome do arquivo .msh importado. Portanto, o código presente nesse apêndice representa todas as simulações de 1 a 9.

```
# -*- coding: utf-8 -*-
"""
Created on Tue May 23 22:01:09 2023

@author: Anna Bárbara

assumindo rho = 1, vx = 1, H = Diametro Caracteristico = SEMPRE 1

"""

import meshio
import numpy as np
#import matplotlib
#import matplotlib.pyplot as plt

# propriedade do fluido
```

```
#nu = 1.0 # Re=1
nu = 0.04 # Re=25 Re=rho * vx * H/mu = vx*H/nu = Re=25
dt = 0.005
nIter = 1000

# leitura de malha e classificacao de contorno por nome (ccName)
#mshname = 'canal.msh'
mshname = 'diam_entre_1_e_2_17_circulos_2a.msh'
msh = meshio.read('./' + mshname)
X = np.array(msh.points[:,0])
Y = np.array(msh.points[:,1])
npoints = len(X)
IEN = msh.cells[1].data # triangles
ne = len(IEN)
IENbound = msh.cells[0].data # lines
IENboundTypeElem = list(msh.cell_data['gmsh:physical'][0] - 1)
boundNames = list(msh.field_data.keys())
IENboundElem = [boundNames[elem] for elem in IENboundTypeElem]

# cria lista de nos do contorno
cc = np.unique(IENbound.reshape(IENbound.size))
ccName = [[] for i in range( len(X) )]
# prioridade 4
for elem in range(0,len(IENbound)):
 if IENboundElem[elem] == 'inlet':
  ccName[ IENbound[elem][0] ] = IENboundElem[elem]
  ccName[ IENbound[elem][1] ] = IENboundElem[elem]
# prioridade 3
for elem in range(0,len(IENbound)):
   if IENboundElem[elem] == 'outlet':
     ccName[ IENbound[elem][0] ] = IENboundElem[elem]
     ccName[ IENbound[elem][1] ] = IENboundElem[elem]
for elem in range(0,len(IENbound)):
 if IENboundElem[elem] == 'obstaculo1':
  ccName[ IENbound[elem][0] ] = IENboundElem[elem]
  ccName[ IENbound[elem][1] ] = IENboundElem[elem]
for elem in range(0,len(IENbound)):
 if IENboundElem[elem] == 'obstaculo2':
  ccName[ IENbound[elem][0] ] = IENboundElem[elem]
```

```
  ccName[ IENbound[elem][1] ] = IENboundElem[elem]
for elem in range(0,len(IENbound)):
 if IENboundElem[elem] == 'obstaculo3':
  ccName[ IENbound[elem][0] ] = IENboundElem[elem]
  ccName[ IENbound[elem][1] ] = IENboundElem[elem]
for elem in range(0,len(IENbound)):
 if IENboundElem[elem] == 'obstaculo4':
  ccName[ IENbound[elem][0] ] = IENboundElem[elem]
  ccName[ IENbound[elem][1] ] = IENboundElem[elem]
for elem in range(0,len(IENbound)):
 if IENboundElem[elem] == 'obstaculo5':
  ccName[ IENbound[elem][0] ] = IENboundElem[elem]
  ccName[ IENbound[elem][1] ] = IENboundElem[elem]
for elem in range(0,len(IENbound)):
 if IENboundElem[elem] == 'obstaculo6':
  ccName[ IENbound[elem][0] ] = IENboundElem[elem]
  ccName[ IENbound[elem][1] ] = IENboundElem[elem]
for elem in range(0,len(IENbound)):
 if IENboundElem[elem] == 'obstaculo7':
  ccName[ IENbound[elem][0] ] = IENboundElem[elem]
  ccName[ IENbound[elem][1] ] = IENboundElem[elem]
for elem in range(0,len(IENbound)):
 if IENboundElem[elem] == 'obstaculo8':
  ccName[ IENbound[elem][0] ] = IENboundElem[elem]
  ccName[ IENbound[elem][1] ] = IENboundElem[elem]
for elem in range(0,len(IENbound)):
 if IENboundElem[elem] == 'obstaculo9':
  ccName[ IENbound[elem][0] ] = IENboundElem[elem]
  ccName[ IENbound[elem][1] ] = IENboundElem[elem]
for elem in range(0,len(IENbound)):
 if IENboundElem[elem] == 'obstaculo10':
  ccName[ IENbound[elem][0] ] = IENboundElem[elem]
  ccName[ IENbound[elem][1] ] = IENboundElem[elem]
for elem in range(0,len(IENbound)):
 if IENboundElem[elem] == 'obstaculo11':
  ccName[ IENbound[elem][0] ] = IENboundElem[elem]
  ccName[ IENbound[elem][1] ] = IENboundElem[elem]
for elem in range(0,len(IENbound)):
 if IENboundElem[elem] == 'obstaculo12':
```

```
95    ccName[ IENbound[elem][0] ] = IENboundElem[elem]
96    ccName[ IENbound[elem][1] ] = IENboundElem[elem]
97  for elem in range(0,len(IENbound)):
98   if IENboundElem[elem] == 'obstaculo13':
99    ccName[ IENbound[elem][0] ] = IENboundElem[elem]
100   ccName[ IENbound[elem][1] ] = IENboundElem[elem]
101 for elem in range(0,len(IENbound)):
102  if IENboundElem[elem] == 'obstaculo14':
103   ccName[ IENbound[elem][0] ] = IENboundElem[elem]
104   ccName[ IENbound[elem][1] ] = IENboundElem[elem]
105 for elem in range(0,len(IENbound)):
106  if IENboundElem[elem] == 'obstaculo15':
107   ccName[ IENbound[elem][0] ] = IENboundElem[elem]
108   ccName[ IENbound[elem][1] ] = IENboundElem[elem]
109 for elem in range(0,len(IENbound)):
110  if IENboundElem[elem] == 'obstaculo16':
111   ccName[ IENbound[elem][0] ] = IENboundElem[elem]
112   ccName[ IENbound[elem][1] ] = IENboundElem[elem]
113 for elem in range(0,len(IENbound)):
114  if IENboundElem[elem] == 'obstaculo17':
115   ccName[ IENbound[elem][0] ] = IENboundElem[elem]
116   ccName[ IENbound[elem][1] ] = IENboundElem[elem]
117
118 # prioridade 2
119 for elem in range(0,len(IENbound)):
120  if IENboundElem[elem] == 'paredeInf':
121   ccName[ IENbound[elem][0] ] = IENboundElem[elem]
122   ccName[ IENbound[elem][1] ] = IENboundElem[elem]
123 # prioridade 1
124 for elem in range(0,len(IENbound)):
125  if IENboundElem[elem] == 'paredeSup':
126   ccName[ IENbound[elem][0] ] = IENboundElem[elem]
127   ccName[ IENbound[elem][1] ] = IENboundElem[elem]
128
129
130 # definicao dos vetores de condicoes de contorno para vx,vy e psi
131 vx_cc = np.zeros( (npoints),dtype='float' )
132 vy_cc = np.zeros( (npoints),dtype='float' )
133 psi_cc = np.zeros( (npoints),dtype='float' )
```

```
for i in cc:
  if ccName[i] == 'paredeInf':
    vx_cc[i] = 0.0
    vy_cc[i] = 0.0
    psi_cc[i] = 0.0
  if ccName[i] == 'paredeSup':
    vx_cc[i] = 0.0
    vy_cc[i] = 0.0
    psi_cc[i] = 10.0
  if ccName[i] == 'inlet':
    vx_cc[i] = 1.0
    vy_cc[i] = 0.0
    psi_cc[i] = Y[i]
  if ccName[i] == 'obstaculo1':
    vx_cc[i] = 0.0
    vy_cc[i] = 0.0
    psi_cc[i] = 7.420693098478472
  if ccName[i] == 'obstaculo2':
    vx_cc[i] = 0.0
    vy_cc[i] = 0.0
    psi_cc[i] = 5.065195034878112
  if ccName[i] == 'obstaculo3':
    vx_cc[i] = 0.0
    vy_cc[i] = 0.0
    psi_cc[i] = 3.525576565694484
  if ccName[i] == 'obstaculo4':
    vx_cc[i] = 0.0
    vy_cc[i] = 0.0
    psi_cc[i] = 2.74795195315145
  if ccName[i] == 'obstaculo5':
    vx_cc[i] = 0.0
    vy_cc[i] = 0.0
    psi_cc[i] = 6.872411848990697
  if ccName[i] == 'obstaculo6':
    vx_cc[i] = 0.0
    vy_cc[i] = 0.0
    psi_cc[i] = 7.8500241877312495
  if ccName[i] == 'obstaculo7':
    vx_cc[i] = 0.0
```

```
173       vy_cc[i] = 0.0
174       psi_cc[i] = 4.457930066554496
175     if ccName[i] == 'obstaculo8':
176       vx_cc[i] = 0.0
177       vy_cc[i] = 0.0
178       psi_cc[i] = 3.7190574143965662
179     if ccName[i] == 'obstaculo9':
180       vx_cc[i] = 0.0
181       vy_cc[i] = 0.0
182       psi_cc[i] = 2.402238452263 56
183     if ccName[i] == 'obstaculo10':
184       vx_cc[i] = 0.0
185       vy_cc[i] = 0.0
186       psi_cc[i] = 6.843165964688799
187     if ccName[i] == 'obstaculo11':
188       vx_cc[i] = 0.0
189       vy_cc[i] = 0.0
190       psi_cc[i] = 2.2196401281684945
191     if ccName[i] == 'obstaculo12':
192       vx_cc[i] = 0.0
193       vy_cc[i] = 0.0
194       psi_cc[i] = 2.0733350997162856
195     if ccName[i] == 'obstaculo13':
196       vx_cc[i] = 0.0
197       vy_cc[i] = 0.0
198       psi_cc[i] = 8.234670933086807
199     if ccName[i] == 'obstaculo14':
200       vx_cc[i] = 0.0
201       vy_cc[i] = 0.0
202       psi_cc[i] = 4.595241097347385
203     if ccName[i] == 'obstaculo15':
204       vx_cc[i] = 0.0
205       vy_cc[i] = 0.0
206       psi_cc[i] = 1.7392475424215599
207     if ccName[i] == 'obstaculo16':
208       vx_cc[i] = 0.0
209       vy_cc[i] = 0.0
210       psi_cc[i] = 5.875979816429652
211     if ccName[i] == 'obstaculo17':
```

```
212     vx_cc[i] = 0.0
213     vy_cc[i] = 0.0
214     psi_cc[i] = 4.965149668823173
215
216
217 # inicializando com zeros a matriz A (densa) e o vetor do lado direito
        b
218 K = np.zeros( (npoints,npoints), dtype='float')
219 M = np.zeros( (npoints,npoints), dtype='float')
220 Gx = np.zeros( (npoints,npoints), dtype='float')
221 Gy = np.zeros( (npoints,npoints), dtype='float')
222
223 for e in range(0,ne):
224  v1,v2,v3 = IEN[e]
225
226  # calcula a area do triangulo
227  area = 0.5*np.linalg.det([[1.0,X[v1],Y[v1]],
228                            [1.0,X[v2],Y[v2]],
229                            [1.0,X[v3],Y[v3]]])
230
231  # definir a matriz kelem
232  b1 = Y[v2]-Y[v3]
233  b2 = Y[v3]-Y[v1]
234  b3 = Y[v1]-Y[v2]
235
236  c1 = X[v3]-X[v2]
237  c2 = X[v1]-X[v3]
238  c3 = X[v2]-X[v1]
239
240  kxelem = (1.0/(4*area))*np.array([[b1*b1,b1*b2,b1*b3],
241                                    [b2*b1,b2*b2,b2*b3],
242                                    [b3*b1,b3*b2,b3*b3]])
243  kyelem = (1.0/(4*area))*np.array([[c1*c1,c1*c2,c1*c3],
244                                    [c2*c1,c2*c2,c2*c3],
245                                    [c3*c1,c3*c2,c3*c3]])
246  kelem = kxelem+kyelem
247  melem = (area/12)*np.array([[2.0,1.0,1.0],
248                              [1.0,2.0,1.0],
249                              [1.0,1.0,2.0]])
```

```
250
251  gxelem = (1.0/6.0)*np.array([[b1,b2,b3],
252                               [b1,b2,b3],
253                               [b1,b2,b3]])
254
255  gyelem = (1.0/6.0)*np.array([[c1,c2,c3],
256                               [c1,c2,c3],
257                               [c1,c2,c3]])
258
259  for ilocal in range(0,3):
260   iglobal = IEN[e,ilocal]
261   for jlocal in range(0,3):
262    jglobal = IEN[e,jlocal]
263
264    K[iglobal,jglobal]  += kelem[ilocal,jlocal]
265    M[iglobal,jglobal]  += melem[ilocal,jlocal]
266    Gx[iglobal,jglobal] += gxelem[ilocal,jlocal]
267    Gy[iglobal,jglobal] += gyelem[ilocal,jlocal]
268
269 # inversao da matriz de massa
270 Minv = np.linalg.inv(M)
271 # calculo da condicao de contorno para omega_z
272 omega_z_cc = Minv@(Gx@vy_cc - Gy@vx_cc)
273 # outra forma de resolver o sistema linear
274 #omega_z_cc = np.linalg.solve(M,(Gx@vy - Gy@vx))
275
276 #print ("... gravando em VTK passo de tempo: " + str(n))
277 point_data = {'psi_cc' : psi_cc}
278 data_vx_cc = {'vx_cc' : vx_cc}
279 data_vy_cc = {'vy_cc' : vy_cc}
280 data_omega_z_cc = {'omega_z_cc' : omega_z_cc}
281 point_data.update(data_vx_cc)
282 point_data.update(data_vy_cc)
283 point_data.update(data_omega_z_cc)
284 meshio.write_points_cells('condicaoDeContorno.vtk',
285                           msh.points,
286                           msh.cells,
287                           point_data=point_data,
288                           )
```

```
289
290 # condicao inicial de vx,vy
291 vx = np.zeros( (npoints),dtype='float' )
292 vy = np.zeros( (npoints),dtype='float' )
293 psi = np.zeros( (npoints),dtype='float' )
294
295 for i in cc:
296  vx[i] = vx_cc[i]
297  vy[i] = vy_cc[i]
298
299 # calculo da condicao inicial de omega_z
300 omega_z = omega_z_cc.copy()
301
302 point_data = {'psi' : psi}
303 data_vx = {'vx' : vx}
304 data_vy = {'vy' : vy}
305 data_omega_z = {'omega_z' : omega_z}
306 point_data.update(data_vx)
307 point_data.update(data_vy)
308 point_data.update(data_omega_z)
309 meshio.write_points_cells('condicaoInicial.vtk',
310                           msh.points,
311                           msh.cells,
312                           point_data=point_data,
313                           )
314
315 # LOOP no TEMPO
316 for n in range(0,nIter):
317  # calculo da condicao de contorno de omega_z
318  omega_z_cc = Minv@(Gx@vy - Gy@vx)
319  # montagem da matriz A
320  vx_diag = np.diag(vx)
321  vy_diag = np.diag(vy)
322  # montagem da matriz A e do vetor b de transporte de vorticidade
323  A = M/dt + nu*K + vx_diag@Gx + vy_diag@Gy # implicito para conv e
     difusao
324  b = (M/dt)@omega_z
325
326  # condicao de contorno para o sistema linear Ax=b
```

```
for i in cc:
 if ccName[i] == 'paredeSup' or \
    ccName[i] == 'paredeInf' or \
    ccName[i] == 'obstaculo1' or \
    ccName[i] == 'obstaculo2' or \
    ccName[i] == 'obstaculo3' or \
    ccName[i] == 'obstaculo4' or \
    ccName[i] == 'obstaculo5' or \
    ccName[i] == 'obstaculo6' or \
    ccName[i] == 'obstaculo7' or \
    ccName[i] == 'obstaculo8' or \
    ccName[i] == 'obstaculo9' or \
    ccName[i] == 'obstaculo10' or \
    ccName[i] == 'obstaculo11' or \
    ccName[i] == 'obstaculo12' or \
    ccName[i] == 'obstaculo13' or \
    ccName[i] == 'obstaculo14' or \
    ccName[i] == 'obstaculo15' or \
    ccName[i] == 'obstaculo16' or \
    ccName[i] == 'obstaculo17' or \
    ccName[i] == 'inlet':
  A[i,:] = 0.0 # zerando a linha
  A[i,i] = 1.0 # colocando 1 na diagonal
  b[i]   = omega_z_cc[i]

# solucao do sistema linear para omega_z
omega_z = np.linalg.solve(A,b)

# solucao da Eq. de funcao-corrente
# montagem da matriz A e do vetor b de funcao-corrente
Apsi = K.copy()
bpsi = M@omega_z

# condicao de contorno para o sistema linear Ax=b
for i in cc:
 if ccName[i] == 'paredeSup' or \
    ccName[i] == 'paredeInf' or \
    ccName[i] == 'obstaculo1' or \
    ccName[i] == 'obstaculo2' or \
```

```
366       ccName[i] == 'obstaculo3' or \
367       ccName[i] == 'obstaculo4' or \
368       ccName[i] == 'obstaculo5' or \
369       ccName[i] == 'obstaculo6' or \
370       ccName[i] == 'obstaculo7' or \
371       ccName[i] == 'obstaculo8' or \
372       ccName[i] == 'obstaculo9' or \
373       ccName[i] == 'obstaculo10' or \
374       ccName[i] == 'obstaculo11' or \
375       ccName[i] == 'obstaculo12' or \
376       ccName[i] == 'obstaculo13' or \
377       ccName[i] == 'obstaculo14' or \
378       ccName[i] == 'obstaculo15' or \
379       ccName[i] == 'obstaculo16' or \
380       ccName[i] == 'obstaculo17' or \
381       ccName[i] == 'inlet':
382     Apsi[i,:] = 0.0 # zerando a linha
383     Apsi[i,i] = 1.0 # colocando 1 na diagonal
384     bpsi[i]   = psi_cc[i]
385
386
387 # solucao do sistema linear para PSI
388 psi = np.linalg.solve(Apsi,bpsi)
389
390 # extracao dos campos de velocidade vx,vy a partir de PSI
391 vx =       Minv@(Gy@psi)
392 vy = -1.0*Minv@(Gx@psi)
393 # outra forma de resolver o sistema linear
394 #vx =  np.linalg.solve(M,(Gy@psi))
395 #vy = -np.linalg.solve(M,(Gx@psi))
396
397 for i in cc:
398   if ccName[i] == 'paredeSup' or \
399       ccName[i] == 'paredeInf' or \
400       ccName[i] == 'obstaculo1' or \
401       ccName[i] == 'obstaculo2' or \
402       ccName[i] == 'obstaculo3' or \
403       ccName[i] == 'obstaculo4' or \
404       ccName[i] == 'obstaculo5' or \
```

```
405     ccName[i] == 'obstaculo6' or \
406     ccName[i] == 'obstaculo7' or \
407     ccName[i] == 'obstaculo8' or \
408     ccName[i] == 'obstaculo9' or \
409     ccName[i] == 'obstaculo10' or \
410     ccName[i] == 'obstaculo11' or \
411     ccName[i] == 'obstaculo12' or \
412     ccName[i] == 'obstaculo13' or \
413     ccName[i] == 'obstaculo14' or \
414     ccName[i] == 'obstaculo15' or \
415     ccName[i] == 'obstaculo16' or \
416     ccName[i] == 'obstaculo17' or \
417     ccName[i] == 'inlet':
418   vx[i]    = vx_cc[i]
419   vy[i]    = vy_cc[i]
420
421 print ("... gravando em VTK passo de tempo: " + str(n).zfill(4))
422 point_data = {'psi' : psi}
423 data_vx = {'vx' : vx}
424 data_vy = {'vy' : vy}
425 data_omega_z = {'omega_z' : omega_z}
426 point_data.update(data_vx)
427 point_data.update(data_vy)
428 point_data.update(data_omega_z)
429 meshio.write_points_cells('solucao_'+ str(n).zfill(4) +'.vtk',
430                           msh.points,
431                           msh.cells,
432                           point_data=point_data,
433                           )
```