

UNIVERSIDADE DE SÃO PAULO

Instituto de Ciências Matemáticas e de Computação

**Animação de fumaça em malhas não-estruturadas usando
RBF-FD**

Gabriel Lucas da Silva

Dissertação de Mestrado do Programa de Pós-Graduação em Ciências
de Computação e Matemática Computacional (PPG-CCMC)

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

Gabriel Lucas da Silva

**Animação de fumaça em malhas não-estruturadas usando
RBF-FD**

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências – Ciências de Computação e Matemática Computacional. *EXEMPLAR DE DEFESA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. Afonso Paiva Neto

**USP – São Carlos
Agosto de 2025**

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

d586a da Silva, Gabriel Lucas
Animação de fumaça em malhas não-estruturadas
usando RBF-FD / Gabriel Lucas da Silva; orientador
Afonso Paiva Neto. -- São Carlos, 2025.
62 p.

Tese (Doutorado - Programa de Pós-Graduação em
Ciências de Computação e Matemática Computacional)
-- Instituto de Ciências Matemáticas e de
Computação, Universidade de São Paulo, 2025.

1. RBF-FD. 2. Animação de fumaça. 3. Malhas
não-estruturadas. I. Neto, Afonso Paiva, orient. II.
Título.

Gabriel Lucas da Silva

Smoke animation on unstructured meshes using RBF-FD

Dissertation submitted to the Instituto de Ciências
Matemáticas e de Computação – ICMC-USP – in
accordance with the requirements of the Computer
and Mathematical Sciences Graduate Program, for
the degree of Master in Science. *EXAMINATION
BOARD PRESENTATION COPY*

Concentration Area: Computer Science and
Computational Mathematics

Advisor: Prof. Dr. Afonso Paiva Neto

USP – São Carlos
August 2025

AGRADECIMENTOS

Gostaria de expressar minha profunda gratidão à minha família, que sempre esteve ao meu lado, oferecendo apoio incondicional e incentivo em cada etapa desta jornada. Aos meus amigos, pela compreensão, paciência e pelas palavras de encorajamento nos momentos mais desafiadores. Agradeço também aos meus colegas de trabalho, pela colaboração e pelas valiosas trocas de conhecimento que enriqueceram esta pesquisa. A todos, minha sincera gratidão por tornarem essa conquista possível.

Gostaria de expressar minha profunda gratidão à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo apoio financeiro proporcionado por meio da bolsa de estudos PROEX, processo número PROEX-12620283/M. Este suporte, durante o período de abril de 2021 a março de 2023, foi fundamental para a realização desta pesquisa, permitindo-me dedicar integralmente ao desenvolvimento deste trabalho e ao avanço do conhecimento na área de computação gráfica. Agradeço pela confiança e pelo investimento no meu potencial acadêmico.

RESUMO

DA SILVA, G. L. **Animação de fumaça em malhas não-estruturadas usando RBF-FD.** 2025. 71 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2025.

Durante as últimas décadas, os estudos de simulações numéricas de escoamento de fluidos e de novas técnicas de processamento geométrico têm sido áreas de intensa pesquisa e com aplicações que vão da engenharia industrial à indústria do entretenimento. Nesse projeto de mestrado, propomos o estudo de novas técnicas de animação de escoamento de fumaça em domínios arbitrários usando interpolações que não necessitam de malha. Essas técnicas consistem na aplicação do método numérico sem malha conhecido como RBF (Radial Basis Function) e sua versão com diferenças finitas generalizadas (RBF-FD).

Palavras-chave: RBF-FD, Animação de fumaça, Malhas não-estruturada, Domínios arbitrários.

ABSTRACT

DA SILVA, G. L. **Smoke animation on unstructured meshes using RBF-FD.** 2025. 71 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2025.

Over the last decades, studies of numerical simulations of fluid flow and new geometric processing techniques have been areas of intense research with applications ranging from industrial engineering to the entertainment industry. In this master's project, we propose the study of new smoke flow animation techniques in arbitrary domains using mesh-free interpolations. These techniques consist of applying the meshless numerical method known as RBF (Radial Basis Function) and its version with generalized finite differences (RBF-FD).

Keywords: RBF-FD, Smoke animation, Unstructured mesh, Arbitrary domains.

LISTA DE ILUSTRAÇÕES

Figura 1 – Animação de fumaça no jogo <i>Mass Effect 3</i> .	19
Figura 2 – Tipos de simulações de escoamento de fluidos	20
Figura 3 – Tipos de células presentes na abordagem euleriana em duas dimensões.	21
Figura 4 – Estêncil em grade uniforme bidimensional.	22
Figura 5 – Redução de RBF-FD em FD no caso 1D, mostrando como é necessário estar organizado a vizinhança; colinear e equidistante.	32
Figura 6 – Estêncil RBF-FD em malha triangular 2D. O ponto central f_i (em verde escuro) possui vizinhos em posições não uniformes, conectados pelas linhas tracejadas azuis com distâncias r_j variáveis.	33
Figura 7 – O ponto analisado, x_i^n , mostra o caminho contrário (em vermelho) à velocidade do escoamento, e o ponto final, x_i^{n-1} , tem seu valor copiado para o ponto inicial. Em azul podemos ver os pontos utilizados na interpolação deste exemplo.	36
Figura 8 – Fluxograma do pipeline de simulação de fluidos.	40
Figura 9 – Exemplo de interseção entre raio de <i>backtracking</i> e fronteira do domínio. A malha é representada em azul, P_1 é o ponto de origem (vermelho) do raio, a direção do raio é indicada pela seta vermelha, P_2 é o ponto final (preto), e o ponto de interseção com a fronteira é mostrado em verde.	42
Figura 10 – Exemplo de malha para método de <i>collocation</i> . Os nós 1, 2, 3 e 4 (azul ciano) estão na fronteira com vetores normais \mathbf{n} apontando para fora, e o nó 5 (laranja) é interno.	45
Figura 11 – Ilustração da criação de um nó fantasma para tratamento de fronteiras. O nó fantasma \mathbf{x}_g é criado na direção normal \mathbf{n} a partir do nó de fronteira \mathbf{x}_b , a uma distância h .	48
Figura 12 – Primeiro experimento: Comparação da eficácia da técnica de nós fantasma em malha circular. Cada linha mostra, da esquerda para direita: simulação sem ghost nodes, simulação com ghost nodes, distribuição espacial do divergente (gráfico 3D) e evolução temporal do divergente máximo. As três linhas representam instantes inicial, intermediário e final da simulação, respectivamente.	59

Figura 13 – Segundo experimento: simulação de fumaça em geometria irregular (formato de país). Cada linha apresenta, da esquerda para direita: visualização da densidade de fumaça sobre a malha, distribuição espacial do divergente (gráfico 3D) e evolução temporal do divergente máximo. As três linhas correspondem aos instantes inicial, intermediário e final da simulação. 60

Figura 14 – Terceiro experimento: simulação de fumaça em domínio com fronteiras altamente irregulares e regiões de alta curvatura. Cada linha apresenta, da esquerda para direita: visualização da densidade de fumaça sobre a malha, distribuição espacial do divergente (gráfico 3D) e evolução temporal do divergente máximo. As três linhas correspondem aos instantes inicial, intermediário e final da simulação. 61

LISTA DE TABELAS

Tabela 1 – Resumo dos artigos revisados.	27
--	----

LISTA DE SÍMBOLOS

- u** — Campo de velocidades
 p — Campo de pressão
 ρ — Densidade do fluido
 ν — Coeficiente de viscosidade dinâmica
g — Vetor de forças externas (gravidade)
w — Campo de velocidades intermediário
 ∇ — Operador gradiente (nabla)
 $\nabla \cdot$ — Operador divergente
 ∇^2 — Operador laplaciano
 \mathcal{L} — Operador diferencial genérico
 Δt — Passo de tempo da simulação
 Δx — Espaçamento da malha (direção x)
 Δx_{local} — Espaçamento local da malha
 n — Índice temporal (superscript): \mathbf{u}^n é a velocidade no tempo t^n
 $\phi(r)$ — Função de base radial
 $\Phi_k(\mathbf{x})$ — RBF centrada no ponto \mathbf{x}_k
 r — Distância radial $r = \|\mathbf{x} - \mathbf{x}_k\|$
 s — Exponente da spline poliharmônica ($\phi(r) = r^s$)
 ω_j — Pesos RBF-FD para o ponto j
 ω_j^L — Pesos RBF-FD para o operador laplaciano
 ω_j^n — Pesos RBF-FD para a derivada normal
 $\tilde{\omega}_j^L$ — Pesos efetivos do laplaciano (com ghost node)
 α_k — Coeficientes de interpolação RBF
 β_j — Coeficientes dos polinômios na interpolação RBF
 $P_j(x)$ — Polinômios da base para interpolação RBF

\mathbf{x}_i — Posição do ponto/nó i na malha

\mathbf{x}^* — Posição de origem após backtracking

\mathbf{x}_b — Nô de fronteira

\mathbf{x}_g — Nô fantasma (ghost node)

\mathbf{n} — Vetor normal unitário à fronteira

h — Distância do nô fantasma à fronteira

α — Parâmetro de ajuste para distância do ghost node

\mathcal{S}_i — Estêncil do nô i (conjunto de vizinhos)

N — Número de pontos no estêncil ou na malha

M — Número de polinômios na base RBF

Ω — Domínio computacional

i, j, k — Índices de pontos/nós na malha

d — Dimensão do domínio ($d = 2$ para 2D)

m — Grau máximo dos polinômios

SUMÁRIO

1	INTRODUÇÃO	19
1.1	Objetivos	23
1.2	Roteiro	23
2	REVISÃO BIBLIOGRÁFICA	25
3	FUNDAMENTAÇÃO TEÓRICA	29
3.1	Aspectos numéricos	29
3.2	Método RBF-FD	30
3.3	Animação de fumaça	33
3.3.1	<i>Advecção Semi-Lagrangiana</i>	35
4	METODOLOGIA	37
4.1	Pipeline do <i>Stable Fluids</i>	38
4.1.1	<i>Termo de Advecção</i>	38
4.1.2	<i>Termo do Gradiente de Pressão</i>	38
4.1.3	<i>Termo de Força Externa</i>	39
4.1.4	<i>Projeção do Campo de Velocidades</i>	39
4.1.5	<i>Algoritmo Completo</i>	40
4.2	Interseção com a Fronteira	40
4.2.1	<i>Algoritmo</i>	41
4.2.2	<i>Desempenho</i>	42
4.2.3	<i>Interpolação</i>	43
4.3	Malhas arbitrárias	44
4.3.1	<i>Precisão do RBF</i>	44
4.4	Tratamento de Fronteiras	44
4.4.1	<i>Método de Collocation para Condições de Neumann</i>	45
4.4.2	<i>Nós Fantasmas</i>	46
4.4.3	<i>Simplificação da Abordagem de Flyer et al.</i>	47
4.4.4	<i>Cálculo do Laplaciano com Nós Fantasmas</i>	49
4.4.4.1	<i>Pesos RBF-FD com estêncil Estendido</i>	49
4.4.4.2	<i>Aplicação da Condição de Neumann</i>	49
4.4.4.3	<i>Eliminação do Nô Fantasma e Pesos Efetivos</i>	50
4.4.4.3.1	Considerações sobre estabilidade numérica	51

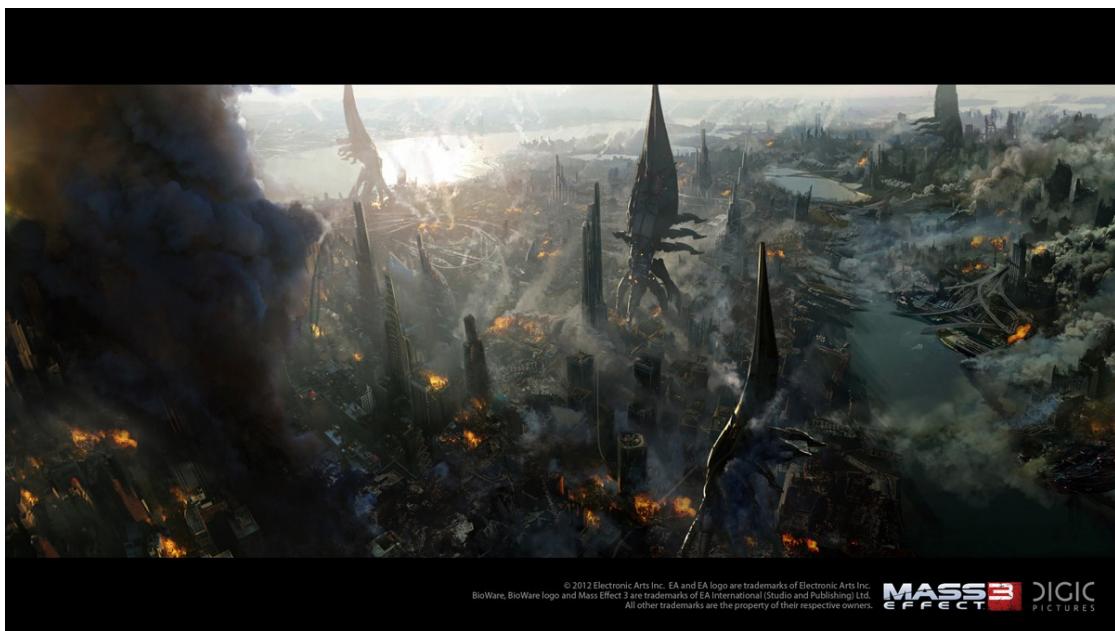
4.4.4.4	<i>Construção da Matriz do Laplaciano</i>	51
4.4.4.5	<i>Exemplo: Sistema Linear com Nós Fantasma</i>	52
4.5	Aspectos Computacionais	53
4.5.1	<i>Otimização de Desempenho</i>	53
4.5.2	<i>Visualização</i>	54
4.5.3	<i>Geração de Malhas</i>	54
4.5.4	<i>Resolução do Sistema Linear</i>	54
4.5.5	<i>Estruturas de Dados</i>	55
5	RESULTADOS	57
5.1	<i>Análise do Divergente</i>	57
5.2	<i>Visualização da Simulação</i>	57
5.3	<i>Conservação de Massa</i>	58
6	CONCLUSÃO	63
6.1	<i>Publicações</i>	63
6.2	<i>Desafios e Trabalhos Futuros</i>	64
	REFERÊNCIAS	65
	APÊNDICES	67
	APÊNDICE A – DEPENDÊNCIAS DO PROJETO	69
A.1	<i>Bibliotecas Principais</i>	69
A.2	<i>Arquivo de Dependências Completo</i>	70
A.3	<i>Ambiente de Execução</i>	70

CAPÍTULO
1

INTRODUÇÃO

As indústrias do entretenimento, jogos (Figura 1) e engenharia têm grande interesse em simulações de fluidos para integrar em seus respectivos produtos. Em geral, simulações baseadas em física tendem a apresentar resultados visualmente realísticos e convincentes, podendo levar uma pessoa leiga na área a confundir uma gravação real com uma simulação (HUANG; GONG; HAN, 2015). É necessário que o observador da simulação seja convencido de sua veracidade, pois isso aumenta a imersão na experiência de um filme ou jogo. Para alcançar esse nível de qualidade desejado, é preciso trabalhar com técnicas computacionalmente custosas para minimizar a perda de detalhes na simulação. Portanto, é importante utilizar métodos com alta precisão e otimização do tempo de execução para poder aplicar a técnica em tempo real, no caso de jogos, e poupar

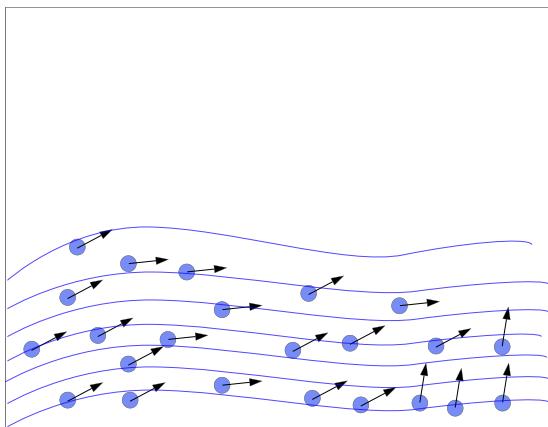
Figura 1 – Animação de fumaça no jogo *Mass Effect 3*.



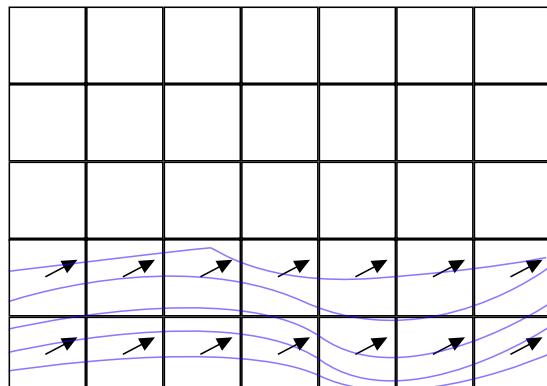
Fonte: <<https://www.fxguide.com/fxfeatured/cinematics-case-study-mass-effect-3/>>

Figura 2 – Tipos de simulações de escoamento de fluidos

(a) Simulação pela visão lagrangiana.



(b) Simulação pela visão euleriana.



Fonte: Elaborada pelo autor.

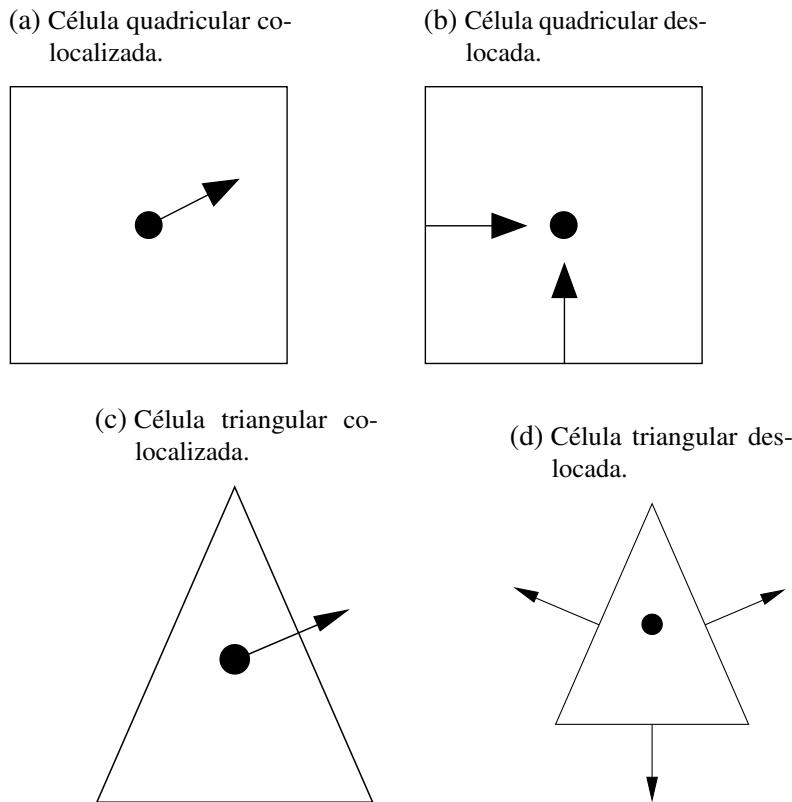
tempo na produção de efeitos especiais em filmes.

O comportamento de fluidos no mundo real é descrito de maneira contínua. Quando se discretiza este fenômeno para simular computacionalmente, é necessário realizar um passo de discretização do domínio. Por conta disto, simulações de fluidos podem ser abordadas de duas maneiras mais comumente usadas: visão lagrangiana e visão euleriana, além da forma híbrida.

Na visão lagrangiana (ver Figura 2a), o fluido é discretizado em forma de um sistema de partículas, com cada uma delas representando uma parte da massa fluídica (volume). Cada partícula tem uma posição no domínio e carrega consigo algumas propriedades daquela parte do fluido que a mesma está discretizando, como velocidade, posição, pressão e força externa. A simulação é realizada pela determinação, em cada passo de tempo, da posição e velocidade de cada partícula do sistema. Já na abordagem euleriana (ver Figura 2b), a discretização do espaço é feita em uma grade ou malha. Esta grade é dividida em células, e estas armazenam as propriedades do fluido que estão passando por aquela região do espaço, sendo responsáveis por gerir como o fluido irá escoar.

Existem diversas características que diferem na representação da malha. Por exemplo, existem malhas regulares e irregulares. Em malhas regulares, os centros de cada célula são equidistantes e colineares, enquanto nas malhas irregulares não existe esta restrição. Outro fator que pode distinguir é a topologia de cada célula da malha: existem malhas com topologia triangular, quadrangular ou mistas. Malhas triangulares se moldam melhor em fronteiras irregulares, como objetos complexos, porém, este benefício é contrabalançado pelo grande custo computacional. Malhas quadrangulares têm a desvantagem de perder muitos detalhes por conta da estrutura simples. Grades híbridas têm a vantagem de poder usar células triangulares em fronteiras com objetos complexos, e células quadrangulares em espaços abertos, melhorando o custo computacional ([HUANG; GONG; HAN, 2015](#)). Além disso, os valores do campo escalar

Figura 3 – Tipos de células presentes na abordagem euleriana em duas dimensões.

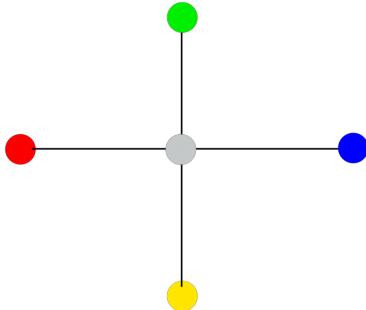


Fonte: Elaborada pelo autor.

ou campo vetorial podem ser armazenados no centro de cada célula ou no centro de cada face. Caso ambos estejam no centro, esta malha é chamada de co-localizada (do inglês, *collocated*), e no caso do campo escalar estar no centro da célula e a velocidade na face, esta malha é chamada de deslocada (do inglês, *staggered*), conforme ilustrado na Figura 3. O foco do nosso trabalho será a simulação de fluidos bidimensional por meio da abordagem euleriana com uso de malhas triangulares co-localizadas.

Em geral, simulações de fluidos baseadas em física têm como base as Equações de Navier-Stokes (ENS) para fluidos incompressíveis ([GRESHO, 1991](#)). Estas são um conjunto de equações diferenciais parciais que descrevem o escoamento dos fluidos. Para resolver as ENS, é necessária a aplicação de operadores diferenciais de cálculo vetorial, tais como: gradiente, divergente e laplaciano. Estes operadores são aplicados em funções contínuas; no entanto, não é possível representar funções contínuas em computadores, logo, estas devem ser discretizadas. Na abordagem euleriana, o método de Diferenças Finitas (do inglês, *Finite Differences* - FD) visa solucionar este problema para grades uniformes. A aplicação de FD em uma célula da grade necessita acessar informações dos vizinhos; estes vizinhos são chamados de estêncil. Em grades uniformes bidimensionais, o estêncil é sempre composto por 4 vizinhos em posições fixas (acima, abaixo, esquerda e direita), conforme mostrado na Figura 4. Como esta vizinhança é fixa em grades uniformes, podemos aplicar este método de forma eficiente.

Figura 4 – Estêncil em grade uniforme bidimensional.



Fonte: Elaborada pelo autor.

Quando trabalhamos com malhas triangulares, temos uma **malha não-estruturada**, definida como uma discretização do domínio onde os elementos (triângulos, no caso 2D) não seguem um padrão regular de conectividade, ou seja, cada vértice pode ter um número variável de elementos adjacentes. Neste contexto, o conceito de estêncil se torna mais complexo, pois a quantidade e configuração dos vizinhos de uma célula não são mais uniformes. Em malhas não-estruturadas, é comum utilizar o conceito de *k-anel* (do inglês, *k-ring*), que define a vizinhança de um elemento da malha de forma topológica. O *1-anel* de uma célula ou vértice é definido como o conjunto de todos os elementos que compartilham pelo menos uma aresta com o elemento central. Diferentemente do *1-anel* em malhas regulares (como mostrado na Figura 4), onde sempre há 4 vizinhos em posições fixas, em malhas não-estruturadas o *1-anel* pode ter um número variável de vizinhos sem restrições de posição. Por exemplo, em uma malha triangular, o *1-anel* de um vértice interno é composto por todos os triângulos que contêm aquele vértice (podendo ser 5, 6, 7 ou mais triângulos), enquanto o *1-anel* de um triângulo é formado pelos três triângulos adjacentes que compartilham suas arestas. Este conceito pode ser estendido para *k-anéis*, onde o *2-anel* inclui os vizinhos dos vizinhos, e assim sucessivamente. A escolha do tamanho do estêncil (ou seja, quantos anéis incluir) impacta diretamente a precisão e o custo computacional dos métodos numéricos aplicados à malha.

A propriedade de vizinhança variável impossibilita o uso direto de FD em malhas não-estruturadas, sendo necessário o uso de outros métodos para a resolução dos operadores diferenciais, tais como os métodos de Elementos Finitos e Volumes Finitos. Em animação computacional, o método de Volumes Finitos (do inglês, *Finite Volumes* - FV) é o mais utilizado para animação de fluidos em malhas não-estruturadas ([KLINGNER et al., 2006](#)).

O método baseado em Funções de Base Radial (do inglês, *Radial Basis Function* - RBF) ([FASSHAUER, 2007](#)) realiza uma interpolação independente da conectividade da malha, e sua aproximação de FD, conhecida como *Radial Basis Function Finite Differences* (RBF-FD) ([WRIGHT; FORNBERG, 2006](#)), resolve os operadores diferenciais necessários através de uma aproximação de alta ordem. Com a aplicação do RBF-FD em malhas não-estruturadas, pretendemos resolver as ENS e representar visualmente a simulação de fumaça.

Um desafio fundamental na resolução numérica das ENS é a correta aplicação de condições de fronteira, essenciais para garantir o comportamento físico apropriado do fluido nos limites do domínio. Existem diferentes tipos de condições de fronteira, sendo as mais comuns as condições de Dirichlet, que prescrevem o valor da função no contorno, e as condições de Neumann, que prescrevem o valor da derivada normal no contorno.

Em particular, as condições de Neumann são cruciais para modelar fronteiras com fluxo livre e paredes impermeáveis, mas sua implementação em malhas não-estruturadas apresenta dificuldades significativas. Enquanto métodos tradicionais como FD e FV têm procedimentos bem estabelecidos para aplicar essas condições em grades uniformes, a extensão para malhas triangulares não-estruturadas utilizando RBF-FD ainda carece de investigação aprofundada na literatura de animação de fluidos.

1.1 Objetivos

O objetivo principal desta dissertação de mestrado é desenvolver uma técnica eficiente de animação de fumaça bidimensional em malhas não-estruturadas utilizando o método RBF-FD, abordando as limitações atuais na aplicação de condições de contorno em malhas triangulares.

1.2 Roteiro

O resto do texto se divide da seguinte maneira. No Capítulo 2, abordaremos como evoluiu a simulação de fluidos computacional ao longo do tempo, avaliando os métodos canônicos até os métodos mais atuais. No Capítulo 3, apresentaremos a fundamentação teórica que baseia nossa solução e enunciaremos os problemas a serem resolvidos. No Capítulo 4, explicaremos os principais métodos utilizados na nossa pesquisa, como o RBF-FD, e como aplicar as condições de contorno de Neumann neste caso. No Capítulo 5, apresentaremos os resultados obtidos com o método proposto, incluindo análise quantitativa do divergente do campo de velocidades e visualização qualitativa das simulações. Por fim, no Capítulo 6, discutiremos as contribuições do trabalho, publicações realizadas e direções para trabalhos futuros.



REVISÃO BIBLIOGRÁFICA

Este capítulo apresenta uma revisão bibliográfica focada em métodos de simulação de fluidos aplicados à computação gráfica, com ênfase em animação de fumaça. As primeiras abordagens para animação de fumaça foram baseadas em métodos não-físicos, sem a utilização das ENS, por meio da animação de texturas (HUANG; GONG; HAN, 2015). Dos métodos baseados em física, o trabalho de Jos Stam (STAM, 1999) tem grande relevância, pois introduziu na literatura o método de advecção semi-lagrangiana, que aumentou o desempenho na simulação de fluidos, dado que é uma solução simples e incondicionalmente estável para o problema da advecção. No entanto, o método apresenta dissipação numérica significativa. Além disso, o método proposto por Stam faz uso de grades co-localizadas. A advecção semi-lagrangiana utiliza os valores de velocidades no centro da célula para retroceder no passo de tempo e copiar a velocidade da posição do passo anterior para a célula atual.

Em 2003, Jos Stam publicou um artigo sobre simulação de fluidos para jogos, focando no desempenho e detalhes de implementação (STAM, 2003). Neste trabalho, o autor apresenta mais detalhes sobre a implementação da malha, explicando o método de células fantasma para auxiliar na computação de operadores diferenciais por meio de FD. O trabalho aborda as condições de fronteira de Neumann, em que a componente normal da velocidade das células fantasma é uma cópia das velocidades dentro da grade, porém com o sinal trocado, fazendo com que a componente perpendicular da velocidade do campo na fronteira seja nula, enquanto a componente tangencial permanece livre. Também é apresentada a condição de fronteira de Dirichlet para o campo de pressão, onde os valores são copiados mantendo o sinal. Como nosso foco não é em grades regulares, para leitores interessados em se aprofundar nesse tópico recomendamos o *survey* (HUANG; GONG; HAN, 2015).

Os trabalhos de Stam mencionados utilizam domínios de simulação simples: um deles utiliza fronteira periódica em uma *Axis Aligned Bounding Box* (AABB) (STAM, 1999), o outro somente uma AABB (STAM, 2003). Este tipo de fronteira facilita a aplicação das condições de

contorno mencionadas anteriormente. Porém, malhas regulares têm dificuldade para representar domínios com fronteiras complexas. Logo, a melhor escolha para trabalhar com tais fronteiras é o uso de malhas triangulares (2D) e tetraedrais (3D) (PARK *et al.*, 2010). A abordagem deste artigo é utilizar uma malha híbrida, em que é possível aproveitar os benefícios dos dois tipos de malhas. A parte regular é usada para preencher espaços vazios (sem fluido), enquanto a parte triangular é usada nas fronteiras, preservando mais detalhes durante a simulação. Nas células triangulares é utilizada uma representação de malha deslocada, com a restrição de cada triângulo conter o próprio circuncentro.

Como malhas não-estruturadas (triangulares em 2D e tetraedrais em 3D) se adaptam melhor às fronteiras complexas, o uso de malhas dinâmicas torna-se fundamental quando há obstáculos que podem se mover durante a simulação. Essa é a abordagem usada em (KLINGNER *et al.*, 2006), que utiliza malhas tetraedrais dinâmicas. Para tais malhas, o método numérico utilizado é o de FV. A malha é recalculada a cada passo de tempo da simulação, exigindo que sua geração seja eficiente e rápida. O tipo de geração de malha escolhido foi o de Delaunay, cujo refinamento é complexo, pois deve manter a conformidade da malha. Foi necessário adaptar a advecção semi-lagrangiana para acomodar a troca de malhas durante o passo de tempo. Além disso, a reconstrução da malha utiliza uma técnica na qual a malha se adapta bem a obstáculos e é refinada com base em critérios arbitrários.

Métodos com malha baseada em árvores são amplamente utilizados em abordagens adaptativas (WANG, ; GREAVES, 2004; NAKANISHI *et al.*, 2020), pois esta estrutura facilita o aumento ou a diminuição da resolução da malha em regiões específicas (refinamento). A principal desvantagem é que o refinamento de *quadtrees* (2D) e *octrees* (3D) gera malhas não-estruturadas, onde a conectividade entre elementos varia. Isto causa complicações nos métodos numéricos mais tradicionais (FD e FV), criando a necessidade de determinar independentemente os vizinhos em cada ponto. Uma solução é usar RBF-FD para realizar a interpolação, que é independente da estrutura da malha. Embora existam diversos trabalhos que utilizam RBF-FD para resolução de Equações Diferenciais Parciais (EDP) (ORUC, 2021; TOJA-SILVA; FAVIER; PINELLI, 2014), incluindo as ENS (MAHMOOD *et al.*, 2019), o primeiro artigo a usar este método para simular fluidos na área de computação gráfica (NAKANISHI *et al.*, 2020) tem como principais contribuições: uma nova técnica para simulação de fluido completamente adaptativa com abordagem híbrida (euleriana e lagrangiana); e o uso de RBF-FD para aproximar os operadores diferenciais em árvores arbitrárias, enquanto a adaptatividade é calculada durante a simulação com o intuito de ajustar a grade à interface do líquido.

Durante a pesquisa do acervo para construir esta revisão bibliográfica, identificamos uma lacuna na literatura quanto ao uso de RBF-FD para animação de fumaça em malhas triangulares não-estruturadas. Apesar de encontrarmos resultados relevantes (ORUC, 2021; TOJA-SILVA; FAVIER; PINELLI, 2014), nenhum deles aborda especificamente malhas triangulares arbitrárias com fronteiras complexas. O trabalho que mais se aproxima é o método proposto por Nakanishi et

Tabela 1 – Resumo dos artigos revisados.

Artigo	Malha	Célula	Método Numérico	Adaptativa	Dinâmica	Dim.
(STAM, 1999) & (STAM, 2003)	Co-localizada	Quad	FD	Não	Não	2D/3D
(PARK <i>et al.</i> , 2010)	Deslocada	Tri/Quad	FV	Não	Sim	2D/3D
(KLINGNER <i>et al.</i> , 2006)	Deslocada	Tet	FV	Não	Sim	3D
(NAKANISHI <i>et al.</i> , 2020)	Deslocada	Quadtree/Octree	RBF-FD	Sim	Não	2D/3D
Nossa proposta	Co-localizada	Tri	RBF-FD	Não	Não	2D

Fonte: Elaborada pelo autor.

al. ([NAKANISHI *et al.*, 2020](#)). Entretanto, o foco deste trabalho é construir uma grade adaptativa baseada em árvores (quadtree/octree), não em malhas triangulares. Nossa proposta é preencher essa lacuna na literatura e desenvolver uma simulação de fluidos em animação computacional utilizando RBF-FD em malhas triangulares bidimensionais não-estruturadas com geometrias arbitrárias.



FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os fundamentos teóricos necessários para o desenvolvimento de uma técnica de simulação de fluidos bidimensional em malhas não-estruturadas. Primeiramente, na Seção 3.1, são abordados os aspectos numéricos da discretização de operadores diferenciais, introduzindo o método de Diferenças Finitas (FD) para grades regulares. Em seguida, na Seção 3.2, apresentamos o método RBF-FD, que generaliza FD para malhas não-estruturadas e permite a aplicação de operadores diferenciais em nuvens de pontos arbitrárias. Por fim, na Seção 3.3, descrevemos as Equações de Navier-Stokes para fluidos incompressíveis e o pipeline clássico de simulação de fumaça, incluindo a advecção semi-lagrangiana e a projeção de Helmholtz-Hodge para garantir a condição de divergente-nulo.

3.1 Aspectos numéricos

Para resolver as ENS que regem a simulação de fluidos precisamos usar um método numérico para discretizar os operadores diferenciais utilizados, sendo eles: laplaciano, divergente e gradiente. Um método usado para grades regulares é o de FD. Na abordagem Euleriana, por conta da discretização ser feita no espaço, não temos informações sobre as derivadas do material simulado, logo é necessário adotar um método para conseguir estes dados. Este método vem da definição de derivada já conhecida, seja $f(x)$ uma função diferenciável e h um incremento infinitesimal:

$$f'(x) = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h} \quad (3.1)$$

Porém, como o nosso domínio é discreto, precisamos adaptar este conceito. Assumindo uma grade regular com uma distância Δx entre as amostras e uma função diferenciável $f(x)$, em que f_i é a i -ésima amostra. Algumas aproximações de primeira ordem para as derivadas parciais são:

$$\text{FD progressiva: } \left. \frac{\partial f}{\partial x} \right|_{x_i} \approx \frac{f_{i+1} - f_i}{\Delta x} \quad (3.2)$$

$$\text{FD regressiva: } \left. \frac{\partial f}{\partial x} \right|_{x_i} \approx \frac{f_i - f_{i-1}}{\Delta x} \quad (3.3)$$

Uma aproximação de segunda ordem para a derivada também pode ser obtida:

$$\text{FD central: } \left. \frac{\partial f}{\partial x} \right|_{x_i} \approx \frac{f_{i+1} - f_{i-1}}{2\Delta x} \quad (3.4)$$

Além do cálculo da derivada segunda:

$$\left. \frac{\partial^2 f}{\partial x^2} \right|_{x_i} = \left. \frac{\partial}{\partial x} \left(\frac{\partial f}{\partial x} \right) \right|_{x_i} \approx \frac{\left(\frac{f_{i+1} - f_i}{\Delta x} \right) - \left(\frac{f_i - f_{i-1}}{\Delta x} \right)}{\Delta x} = \frac{f_{i+1} - 2f_i + f_{i-1}}{\Delta x^2}. \quad (3.5)$$

Definindo um campo vetorial 2D $\mathbf{u} = (u, v)^\top$ e um campo escalar p , os operadores diferenciais utilizados, gradiente, divergente e laplaciano, respectivamente, têm a seguinte forma:

$$\text{Gradiente: } \nabla p = \left(\frac{\partial p}{\partial x}, \frac{\partial p}{\partial y} \right)^\top \quad (3.6)$$

$$\text{Divergente: } \nabla \cdot \mathbf{u} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}, \quad (3.7)$$

$$\text{Laplaciano: } \nabla^2 p = \frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2}. \quad (3.8)$$

E suas respectivas formas com a aproximação com FD:

$$\nabla p \approx \left(\frac{p_{i+1,j} - p_{i-1,j}}{\Delta x}, \frac{p_{i,j+1} - p_{i,j-1}}{\Delta y} \right)^\top, \quad (3.9)$$

$$\nabla \cdot \mathbf{u} \approx \frac{u_{i+1,j} - u_{i-1,j}}{\Delta x} + \frac{v_{i,j+1} - v_{i,j-1}}{\Delta y}, \quad (3.10)$$

$$\nabla^2 p \approx \frac{p_{i+1,j} - 2p_{i,j} + p_{i-1,j}}{\Delta x^2} + \frac{p_{i,j+1} - 2p_{i,j} + p_{i,j-1}}{\Delta y^2}. \quad (3.11)$$

3.2 Método RBF-FD

Uma RBF é uma função radialmente simétrica entre o centro \mathbf{x}_k e o ponto avaliado \mathbf{x} , ambos no domínio $\Omega \subset \mathbb{R}^d$. Matematicamente pode ser representada como $\Phi_k(\mathbf{x}) = \phi(||\mathbf{x} - \mathbf{x}_k||)$, onde $\phi(r)$ representa uma função escalar $[0, \infty)$ e $||\cdot||$ denota a norma euclidiana. Existem várias escolhas para ϕ , uma é a *spline poliharmônica*:

$$\phi(r) = r^s, \text{ com } s = 1, 3, 5, \dots \quad (3.12)$$

Para realizar a solução dos operadores diferenciais na nossa malha não-estruturada, escolhemos a técnica de RBF-FD. Seja uma nuvem de pontos $\{x_k\}_{k=1}^N$ e os valores das funções $y_k = y(x_k) \in \mathbb{R}$, queremos encontrar um interpolador $S_y : \Omega \rightarrow \mathbb{R}$, tal que:

$$S_y(x_i) = y_i, \forall i = 1, \dots, N. \quad (3.13)$$

A forma generalizada de um interpolador RBF é dada por:

$$S_y(x) = \sum_{k=1}^N \alpha_k \phi(||x - x_k||) + \sum_{j=1}^M \beta_j P_j(x). \quad (3.14)$$

No qual $\{P_1(x), \dots, P_M(x)\}$ é a base para o espaço $M = \binom{m+d}{d}$ -dimensional \prod_m^d de todos os polinômios multivariados com grau $\leq m$. Para o caso bidimensional ($d = 2$), a base polinomial é construída da seguinte forma:

- Para $m = 1$ (linear), $M = 3$: $\{P_1, P_2, P_3\} = \{1, x, y\}$
- Para $m = 2$ (quadrático), $M = 6$: $\{P_1, \dots, P_6\} = \{1, x, y, x^2, xy, y^2\}$

Neste trabalho, utilizamos polinômios de grau $m = 2$, resultando em $M = 6$ termos polinomiais. Para garantir que os coeficientes α_k e β_j são únicos, precisamos impor a seguinte restrição:

$$\sum_{k=1}^N \alpha_k P_j(x_k) = 0, \forall 1, \dots, M. \quad (3.15)$$

Com isso temos o seguinte sistema para resolver:

$$\begin{bmatrix} \mathbf{A} & \mathbf{P} \\ \mathbf{P}^\top & \mathbf{O} \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{bmatrix} = \begin{bmatrix} \mathbf{y} \\ \mathbf{0} \end{bmatrix}, \quad (3.16)$$

onde \mathbf{A} é uma matriz de ordem N com $\mathbf{A}_{ij} = \phi(||x_i - x_j||)$, \mathbf{P} é uma matriz de ordem $N \times M$ e $\mathbf{P}_{ij} = P_j(x_i)$, \mathbf{O} é uma matriz nula de ordem M , $\boldsymbol{\alpha} = [\alpha_1, \dots, \alpha_N]^\top$, $\boldsymbol{\beta} = [\beta_1, \dots, \beta_M]^\top$, $\mathbf{y} = [y_1, \dots, y_N]^\top$ e $\mathbf{0}$ é o vetor nulo de tamanho M . Para mais detalhes de como é feita a solução desse sistema, consultar (NAKANISHI *et al.*, 2020).

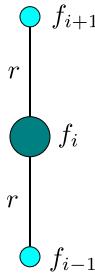
Seja \mathcal{L} o operador diferencial que queremos aproximar, e \mathcal{X}_i a vizinhança composta por uma nuvem de pontos $\{x_k\}_{k=1}^N$. Para uma dada posição x_i , queremos aproximar $\mathcal{L}y$ avaliado no ponto central x_i como uma combinação linear dos valores das funções $\{y_k\}_{k=1}^N$, tal que:

$$\mathcal{L}y = \sum_{k=1}^N \omega_k y_k. \quad (3.17)$$

Similar ao interpolador RBF, os pesos ω são obtidos resolvendo o seguinte sistema:

$$\begin{bmatrix} \mathbf{A} & \mathbf{P} \\ \mathbf{P}^\top & \mathbf{O} \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega} \\ \boldsymbol{\gamma} \end{bmatrix} = \begin{bmatrix} \mathcal{L}\phi \\ \mathcal{L}\mathbf{P} \end{bmatrix}, \quad (3.18)$$

Figura 5 – Redução de RBF-FD em FD no caso 1D, mostrando como é necessário estar organizado a vizinhança; colinear e equidistante.



Fonte: Elaborada pelo autor.

O método de FD é bastante usado em malhas regulares, pois, se beneficia da uniformidade do espaçamento das amostras. Quando trabalhamos com RBF-FD, temos uma generalização de FD para uma nuvem de pontos qualquer. Caso essa vizinhança esteja organizada de maneira uniforme, como uma grade regular, temos a redução de RBF-FD para FD. No caso 1D, ilustrado pela Figura 5, queremos calcular $\partial f_i / \partial x_k \approx \omega_{i-1} f_{i-1} + \omega_{i+1} f_{i+1}$. A matriz para cálculo de pesos fica da seguinte forma:

$$\begin{bmatrix} \phi(0) & \phi(2r) & 1 & (x_{i-1})_k \\ \phi(2r) & \phi(0) & 1 & (x_{i+1})_k \\ 1 & 1 & 0 & 0 \\ (x_{i-1})_k & (x_{i+1})_k & 0 & 0 \end{bmatrix} \begin{bmatrix} \omega_{i-1} \\ \omega_{i+1} \\ \gamma_{i-1} \\ \gamma_{i+1} \end{bmatrix} = \begin{bmatrix} \frac{\partial \phi}{\partial z}(r) \\ \frac{\partial \phi}{\partial z}(r) \\ 0 \\ 1 \end{bmatrix}, \quad (3.19)$$

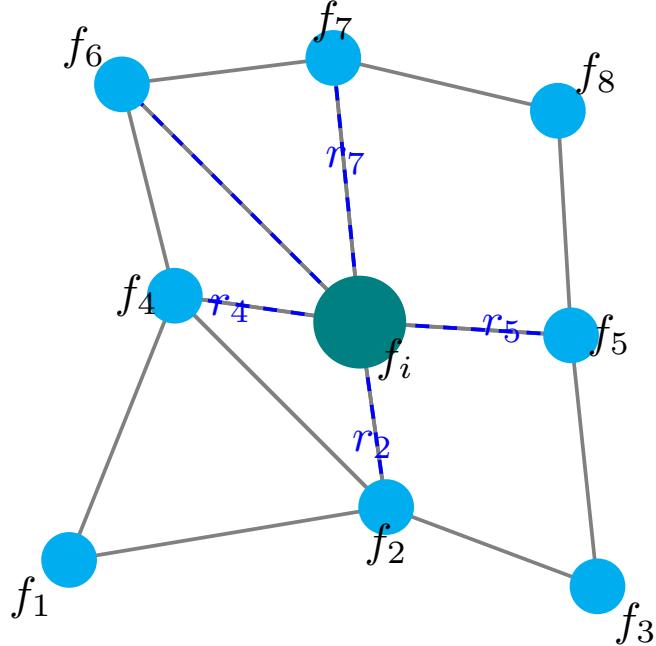
com $r = \|x_{i+1} - x_i\| = \|x_{i-1} - x_i\|$ e $(\cdot)_k$ a k-ésima coordenada do ponto. Resolvendo o sistema (3.19), temos $\omega_{i+1} = 1/2r$ e $\omega_{i-1} = -1/2r$. Portanto,

$$\frac{\partial f_i}{\partial x_k} \approx \frac{f_{i+1} - f_{i-1}}{2r}. \quad (3.20)$$

Como podemos ver, a Equação (3.20) coincide com a aproximação por FD central (3.4).

A principal vantagem do método RBF-FD é sua capacidade de generalizar para malhas não-estruturadas, como malhas triangulares. A Figura 6 ilustra um estêncil típico em uma malha triangular 2D, onde o ponto central f_i possui vizinhos em posições arbitrárias, cada um a uma distância r_j diferente. O mesmo procedimento de cálculo de pesos descrito anteriormente é aplicado, porém com uma matriz de dimensão maior (proporcional ao número de vizinhos no estêncil) e sem a simplificação de equidistância.

Figura 6 – Estêncil RBF-FD em malha triangular 2D. O ponto central f_i (em verde escuro) possui vizinhos em posições não uniformes, conectados pelas linhas tracejadas azuis com distâncias r_j variáveis.



Fonte: Elaborada pelo autor.

3.3 Animação de fumaça

As ENS regem uma simulação de escoamento de fluidos, sua forma final para fluidos incompressíveis é representada pelas seguintes equações:

$$\dot{\mathbf{u}} = -(\mathbf{u} \cdot \nabla) \mathbf{u} - \frac{1}{\rho} \nabla p + v \nabla^2 \mathbf{u} + \mathbf{g}, \quad (3.21)$$

$$\nabla \cdot \mathbf{u} = 0. \quad (3.22)$$

onde \mathbf{u} denota o campo de velocidade, p a pressão e ρ a densidade do fluido. A Equação (3.22) é a condição de incompressibilidade do fluido. O primeiro termo da Equação (3.21), $(\mathbf{u} \cdot \nabla)$, é conhecido como o termo de convecção, responsável pelo transporte de propriedades do fluido ao longo do escoamento. A advecção em específico é a convecção da velocidade. O segundo termo é o gradiente de pressão local, $\frac{1}{\rho} \nabla p$. O fluido escoa conforme o gradiente negativo da pressão. O terceiro termo, $v \nabla^2 \mathbf{u}$, é o termo de viscosidade ou difusão, que considera forças de cisalhamento. A constante v é o coeficiente de viscosidade dinâmica, com efeito de gerar resistência ao escoamento, criando forças que difundem as velocidades através do fluido e podem criar turbulências se existirem gradientes de alta velocidade. O quarto e último termo é o de forças externas, que exerce forças como a gravidade \mathbf{g} e força centrífuga, atuando uniformemente através do fluido. Além disso, também pode incluir forças de interação do usuário com a simulação.

Quando tratamos de fumaça, podemos considerá-la como um fluido invíscido, isto é, $\nu = 0$. Logo, podemos reescrever a equação de momento (3.21) da seguinte forma:

$$\dot{\mathbf{u}} = -(\mathbf{u} \cdot \nabla) \mathbf{u} - \frac{1}{\rho} \nabla p + \mathbf{g} \quad (3.23)$$

Assim, dado um campo vetorial de velocidades \mathbf{w} e o campo escalar de pressão p , o pipeline de animação de fumaça proposto por Stam, conhecido como *Stable Fluids*, pode ser simplificado em 3 passos ([STAM, 1999](#)):

Passo 1: O primeiro passo acelera o campo de velocidade conforme o campo de velocidade externa, \mathbf{g} , que pode ser afetado por forças aplicadas pelo usuário. Neste trabalho utilizamos o método de Euler para resolver a integração temporal, logo:

$$\mathbf{w}^n = \mathbf{w}^{n-1} + \Delta t \cdot \mathbf{g}, \quad (3.24)$$

onde Δt é o passo de tempo da simulação.

Passo 2: Este passo se resume ao transporte de propriedades do fluido ao longo do escoamento e é resolvido pela técnica semi-lagrangiana, que consiste em: para cada nó da malha no passo de tempo n , x_i^n , verificar o valor de um atributo nesta posição, v_i^n ; andar no sentido contrário do escoamento (*backtracking*), encontrando a posição no passo anterior, x_i^{n-1} ; e copiar o valor da posição final, v_i^{n-1} , para a posição inicial (ver Figura 7). Como não é garantido que x_i^{n-1} será um valor disponível na nossa malha, precisamos utilizar um método de interpolação para aproximar v_i^{n-1} . O método escolhido, no caso 2D com malha estruturada, é a interpolação bilinear, que necessita de 4 pontos em topologia retangular para interpolar o valor corretamente.

Passo 3: Nenhuma das operações anteriores se preocupa em manter o campo com divergente-nulo, condição (3.22). Logo, este passo aproxima o campo calculado para o campo divergente-nulo através de uma projeção. De acordo com a decomposição de Helmholtz-Hodge, um campo vetorial suave \mathbf{w} pode sempre ser decomposto na soma de dois campos vetoriais \mathbf{u} e \mathbf{v} , tal que \mathbf{u} é divergente-nulo e \mathbf{v} é um campo com rotacional nulo.

$$\mathbf{w} = \mathbf{u} + \mathbf{v}, \quad (3.25)$$

$$\nabla \cdot \mathbf{u} = 0 \text{ e } \nabla \times \mathbf{v} = \mathbf{0}. \quad (3.26)$$

Se um campo tem rotacional nulo, isso implica que existe um campo p , que satisfaz:

$$\mathbf{v} = \nabla p \quad (3.27)$$

Substituindo a Equação (3.25) na Equação (3.27), temos:

$$\mathbf{w} = \mathbf{u} + \nabla p \quad (3.28)$$

Aplicando o divergente dos dois lados, pela Equação (3.26) segue que:

$$\nabla^2 p = \nabla \cdot \mathbf{w} \quad (3.29)$$

Esta equação é conhecida como a *Equação de Poisson*. Com isso conseguimos projetar o campo \mathbf{w} , em uma aproximação que seja divergente-nulo. Para isso basta resolver a Equação (3.29) usando as discretizações FD dadas pelas Equações (3.11) e (3.10) impondo a condição de Neumann $\nabla p \cdot \mathbf{n} = 0$ na fronteira do domínio, onde \mathbf{n} é o vetor normal na fronteira do domínio, obtemos o campo de pressão p . Finalmente, \mathbf{u} é dado por:

$$\mathbf{u} = \mathbf{w} - \nabla p.$$

3.3.1 Advecção Semi-Lagrangiana

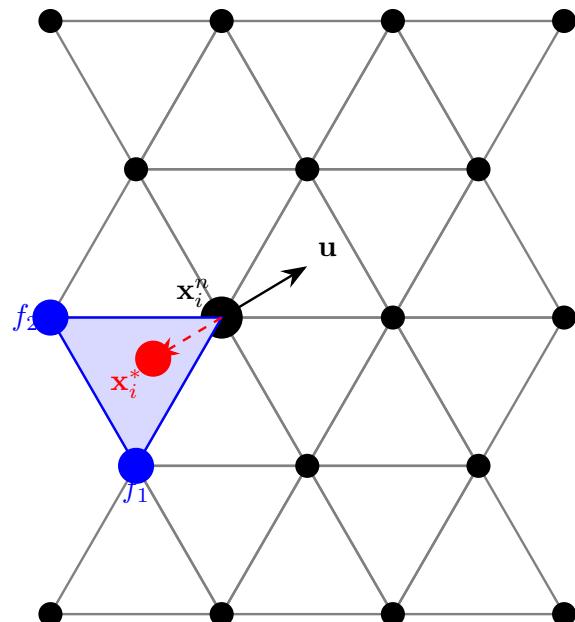
O método semi-lagrangiano para advecção, introduzido por [Stam \(1999\)](#), é uma técnica incondicionalmente estável para resolver o termo não-linear $(\mathbf{u} \cdot \nabla) \mathbf{u}$ das Equações de Navier-Stokes. O método baseia-se na técnica das características, que pode ser entendida intuitivamente como o rastreamento reverso das partículas de fluido.

Para obter a velocidade $\mathbf{u}(\mathbf{x}, t + \Delta t)$ em um ponto \mathbf{x} no tempo $t + \Delta t$, o algoritmo: (i) retrocede o ponto \mathbf{x} através do campo de velocidades \mathbf{w}_1 por um tempo Δt ; (ii) define uma trajetória $\mathbf{p}(\mathbf{x}, s)$ correspondente a uma linha de corrente parcial; e (iii) atribui a nova velocidade como sendo a velocidade que a partícula tinha em sua posição anterior (ver Figura 7):

$$\mathbf{w}_2(\mathbf{x}) = \mathbf{w}_1(\mathbf{p}(\mathbf{x}, -\Delta t)) \quad (3.30)$$

A estabilidade incondicional do método é consequência direta desta formulação, pois o valor máximo do novo campo nunca será maior que o maior valor do campo anterior. Diferentemente das abordagens por diferenças finitas que requerem passos de tempo pequenos satisfazendo a condição $\Delta t < \Delta x / |\mathbf{u}|$, o método semi-lagrangiano permanece estável para qualquer passo de tempo.

Figura 7 – O ponto analisado, x_i^n , mostra o caminho contrário (em vermelho) à velocidade do escoamento, e o ponto final, x_i^{n-1} , tem seu valor copiado para o ponto inicial. Em azul podemos ver os pontos utilizados na interpolação deste exemplo.



Fonte: Elaborada pelo autor.



METODOLOGIA

Neste capítulo, apresentamos a metodologia desenvolvida para a simulação de fumaça em malhas não-estruturadas utilizando o método RBF-FD. A abordagem proposta combina técnicas de simulação de fluidos baseadas no trabalho seminal *Stable Fluids* proposto por [Stam \(1999\)](#) com uma nova formulação para lidar com malhas arbitrárias através de funções de base radial, conforme usado por [Nakanishi *et al.* \(2020\)](#).

Nossa metodologia está estruturada em quatro seções principais. Inicialmente, apresentamos uma adaptação do pipeline de [Stam \(1999\)](#) para o contexto de malhas não-estruturadas, detalhando a discretização e resolução de cada termo das Equações de Navier-Stokes utilizando RBF-FD.

A segunda seção introduz uma solução para o problema de intersecção entre raios e polígonos, contribuição fundamental para o tratamento adequado de partículas que ultrapassam os limites do domínio durante a simulação. Esta solução estende as capacidades do método semi-lagrangiano tradicional ([STAM, 1999](#)) para operar em malhas arbitrárias, superando as limitações das abordagens anteriores que se restringiam a malhas estruturadas ([STAM, 2003](#)).

A terceira seção aborda os desafios específicos relacionados ao uso de malhas não estruturadas. Apresentamos estratégias para melhorar a precisão do método RBF nas fronteiras do domínio, incluindo o desenvolvimento de técnicas especiais para o tratamento de nós fantasma, utilizando o trabalho desenvolvido em [Flyer, Barnett e Wicker \(2016\)](#).

Por fim, discutimos os aspectos computacionais da implementação, detalhando as estruturas de dados utilizadas e as otimizações realizadas para garantir a eficiência do método em aplicações práticas.

4.1 Pipeline do *Stable Fluids*

A implementação numérica das Equações de Navier-Stokes incompressíveis usando o método RBF-FD requer a discretização adequada de cada um dos termos apresentados na Seção 3.3. Nesta seção, detalharemos como cada termo foi implementado, mantendo a consistência com as propriedades matemáticas do escoamento e garantindo a estabilidade numérica da simulação.

4.1.1 Termo de Advecção

O termo de advecção, $(\mathbf{u} \cdot \nabla)\mathbf{u}$, representa o transporte das propriedades do fluido pelo próprio campo de velocidades. Para sua implementação usando RBF-FD, adotamos uma abordagem semi-lagrangiana similar à apresentada por Stam ([STAM, 1999](#)), porém adaptada ao contexto de malhas não-estruturadas. O procedimento pode ser descrito em três etapas principais:

Etapa 1: Para cada ponto \mathbf{x}_i da malha computacional no tempo t^n , onde $i \in \{1, \dots, N_{total}\}$ é o índice do ponto na malha, calculamos a posição de origem \mathbf{x}_i^* através do *backtracking*:

$$\mathbf{x}_i^* = \mathbf{x}_i - \Delta t \mathbf{u}(\mathbf{x}_i, t^n) \quad (4.1)$$

onde \mathbf{x}_i^* denota a posição de partida da partícula que chegará em \mathbf{x}_i após um passo de tempo Δt , seguindo o campo de velocidades \mathbf{u} .

Etapa 2: Como \mathbf{x}_i^* geralmente não coincide com um ponto da malha, utilizamos interpolação RBF para aproximar o valor da velocidade nesta posição. A função de interpolação é construída usando os N pontos mais próximos de \mathbf{x}_i^* , onde N é o número de pontos do estêncil:

$$\mathbf{u}(\mathbf{x}_i^*, t^{n-1}) = \sum_{j=1}^N \lambda_j \phi(\|\mathbf{x}_i^* - \mathbf{x}_j\|) \quad (4.2)$$

onde $\phi(r)$ é a função de base radial escolhida e λ_j são os coeficientes de interpolação determinados pela solução de um sistema linear local.

Etapa 3: O valor interpolado é então atribuído ao ponto \mathbf{x}_i no tempo t^n :

$$\mathbf{u}(\mathbf{x}_i, t^n) = \mathbf{u}(\mathbf{x}_i^*, t^{n-1}) \quad (4.3)$$

4.1.2 Termo do Gradiente de Pressão

O termo do gradiente de pressão, $-\frac{1}{\rho} \nabla p$, é implementado diretamente usando os pesos RBF-FD previamente calculados para o operador gradiente. Para um ponto \mathbf{x}_i e seu estêncil de N pontos, temos:

$$(\nabla p)_i = \sum_{j=1}^N \omega_{ij}^\nabla p_j \quad (4.4)$$

onde i é o índice do ponto onde o gradiente está sendo calculado, j percorre os N pontos do estêncil \mathcal{S}_i , p_j é o valor da pressão no ponto j , e ω_{ij}^∇ são os pesos RBF-FD para o operador gradiente.

4.1.3 Termo de Força Externa

O termo de força externa \mathbf{g} incorpora as forças volumétricas que atuam sobre o fluido, como a gravidade e forças de interação do usuário. No contexto da simulação de fumaça, este termo é particularmente importante para modelar o efeito de empuxo térmico. A implementação deste termo é direta, sendo adicionado explicitamente ao campo de velocidades:

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \Delta t \mathbf{g} \quad (4.5)$$

onde o superscript n denota o passo de tempo atual ($t = n \cdot \Delta t$) e $n+1$ o próximo passo de tempo.

Para o caso específico da força de empuxo térmico, utilizamos a aproximação de Boussinesq:

$$\mathbf{g} = \beta(T - T_\infty) \mathbf{g}_0 \quad (4.6)$$

onde β é o coeficiente de expansão térmica, T é a temperatura local, T_∞ é a temperatura ambiente e \mathbf{g}_0 é o vetor da aceleração da gravidade.

4.1.4 Projeção do Campo de Velocidades

A etapa de projeção é fundamental para garantir a condição de incompressibilidade do fluido ($\nabla \cdot \mathbf{u} = 0$). Seguindo a decomposição de Helmholtz-Hodge apresentada na Seção 3.3, implementamos a projeção em três etapas:

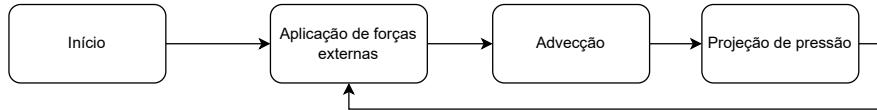
Primeiro, calculamos o divergente do campo de velocidades intermediário \mathbf{w} usando os pesos RBF-FD para o operador divergente:

$$(\nabla \cdot \mathbf{w})_i = \sum_{j=1}^N \omega_{ij}^{\text{div}} \cdot \mathbf{w}_j \quad (4.7)$$

Em seguida, resolvemos a equação de Poisson para a pressão:

$$\nabla^2 p = \nabla \cdot \mathbf{w} \quad (4.8)$$

Figura 8 – Fluxograma do pipeline de simulação de fluidos.



Fonte: Elaborada pelo autor.

utilizando os pesos RBF-FD para o operador Laplaciano, resulta no sistema linear:

$$\sum_{j=1}^N \omega_{ij}^L p_j = (\nabla \cdot \mathbf{w})_i \quad (4.9)$$

Para este sistema, aplicamos a condição de contorno de Neumann na fronteira do domínio:

$$\nabla p \cdot \mathbf{n} = 0 \quad (4.10)$$

Por fim, o campo de velocidades livre de divergência é obtido subtraindo o gradiente de pressão do campo intermediário:

$$\mathbf{u}_i = \mathbf{w}_i - \nabla p_i \quad (4.11)$$

onde ∇p_i é calculado usando a Equação (4.4).

4.1.5 Algoritmo Completo

O algoritmo completo para a simulação do escoamento, ilustrado na Figura 8, pode ser resumido nas seguintes etapas:

1. Atualização das forças externas usando as Equações (4.5) e (4.6);
2. Advecção do campo de velocidades através das Equações (4.1)–(4.3);
3. Projeção do campo para garantir a incompressibilidade usando as Equações (4.7)–(4.11).

Este procedimento é repetido a cada passo de tempo da simulação, garantindo a evolução estável e fisicamente consistente do escoamento.

4.2 Interseção com a Fronteira

Durante o processo de advecção semi-lagrangiana descrito na Etapa 1 da Seção 4.1.1, o *backtracking* a partir de um ponto \mathbf{x}_i pode resultar em uma posição \mathbf{x}_i^* que se encontra fora do domínio computacional. Este cenário é particularmente comum em malhas não-estruturadas

com fronteiras complexas, onde partículas próximas aos contornos podem, ao serem advectadas, ultrapassar os limites do domínio. Para garantir que a interpolação seja realizada corretamente e que as condições de contorno sejam respeitadas, é fundamental determinar com precisão onde o raio de *backtracking* intersecta a fronteira da malha.

A solução deste problema requer o cálculo eficiente da interseção entre um raio (definido pela trajetória de *backtracking*) e os segmentos que compõem a fronteira do polígono que delimita o domínio computacional. Um algoritmo amplamente utilizado para calcular essa interseção baseia-se no método de *Ray Casting* (lançamento de raio), que consiste em disparar um raio a partir de um ponto de origem na direção desejada e determinar onde ele cruza as arestas do polígono. O algoritmo examina cada aresta do polígono para verificar se ocorre uma interseção com o raio, determinando o ponto exato onde essa interseção acontece.

4.2.1 Algoritmo

Para calcular a interseção entre um raio e um polígono em 2D, focamos apenas na interseção do raio com as arestas do polígono. Um método eficaz para isso é o algoritmo de interseção de segmento de linha, como o algoritmo de Cohen-Sutherland ou o algoritmo de Liang-Barsky ([FOLEY et al., 1996](#)). Vamos considerar o algoritmo de Cohen-Sutherland, que é mais simples de entender e implementar para esse cenário específico.

Para cada aresta do polígono, representada pelos pontos P_1 e P_2 , calculamos a interseção com o raio utilizando a equação paramétrica da reta. Sejam $P_1(x_1, y_1)$ e $P_2(x_2, y_2)$ os pontos que definem a aresta e (x, y) o ponto de interseção com o raio. A equação paramétrica da reta é dada por:

$$x = x_1 + t(x_2 - x_1) \quad (4.12)$$

$$y = y_1 + t(y_2 - y_1) \quad (4.13)$$

onde t é um parâmetro que varia de 0 a 1. Substituindo essas equações na equação do raio, obtemos uma equação:

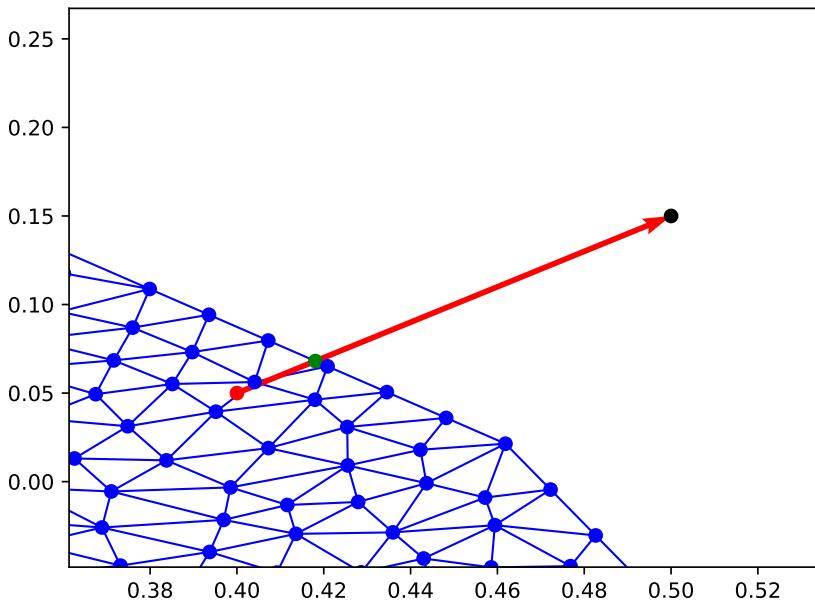
$$t = (x - x_1)(x_2 - x_1) = (y - y_1)(y_2 - y_1) \quad (4.14)$$

Se t estiver no intervalo $[0, 1]$, então a interseção ocorre dentro do segmento de linha P_1P_2 . Calculando t para ambos os eixos, podemos verificar se o ponto de interseção está dentro do segmento de linha.

Este procedimento é repetido para cada aresta do polígono. Se houver interseção, o algoritmo retorna o ponto de interseção mais próximo ao ponto de origem do raio. Se não houver interseção com nenhuma das arestas, então o raio não intersecta o polígono.

Este algoritmo é simples de implementar em 2D, fornecendo uma solução robusta para calcular a interseção entre um raio e um polígono.

Figura 9 – Exemplo de interseção entre raio de *backtracking* e fronteira do domínio. A malha é representada em azul, P_1 é o ponto de origem (vermelho) do raio, a direção do raio é indicada pela seta vermelha, P_2 é o ponto final (preto), e o ponto de interseção com a fronteira é mostrado em verde.



Fonte: Elaborada pelo autor.

4.2.2 Desempenho

É possível aplicar uma vetorização a esse cálculo para tentar melhorar o desempenho, fazendo com que seja calculada a interseção de um raio com todas as arestas de um polígono em um único passo. A técnica de vetorização permite uma abordagem eficiente e paralela para calcular a interseção de um raio com múltiplas arestas de um polígono em 2D, proporcionando uma solução escalável e otimizada para esse problema.

Embora a técnica de vetorização ofereça uma abordagem eficiente para calcular a interseção de um raio com múltiplas arestas de um polígono em 2D, implementá-la diretamente em Python pode resultar em um desempenho insatisfatório para conjuntos de dados maiores. Isso ocorre devido à natureza interpretada do Python e à falta de otimização de baixo nível para operações matriciais.

Para superar essa limitação de desempenho, podemos recorrer à técnica de compilação just-in-time (JIT) oferecida pela biblioteca Numba¹. Numba é uma biblioteca que compila

¹ <https://numba.pydata.org/>

funções Python em código de máquina otimizado, melhorando significativamente o desempenho, especialmente para operações numéricas intensivas.

Ao utilizar Numba, podemos decorar a função que calcula a interseção do raio com múltiplas arestas com `@jit` para permitir a compilação JIT. Isso transformará a função em código de máquina otimizado, proporcionando uma melhoria significativa no desempenho.

Além disso, ao trabalhar com grandes conjuntos de dados, podemos aproveitar técnicas de paralelismo oferecidas por Numba para distribuir as operações em múltiplos núcleos de CPU, acelerando ainda mais o processo de cálculo.

Portanto, ao utilizar Numba, podemos melhorar significativamente o desempenho do algoritmo de cálculo da interseção do raio com múltiplas arestas, tornando-o adequado para lidar eficientemente com conjuntos de dados maiores em tempo real ou aplicações de alto desempenho.

4.2.3 Interpolação

Durante o processo de *backtracking*, quando a posição \mathbf{x}_i^* não coincide com um ponto da malha, é necessário interpolar os valores das propriedades do fluido a partir dos pontos vizinhos. Esse cenário caracteriza um problema de interpolação em **dados irregulares**, onde os pontos de amostragem (posições após backtracking) não coincidem com os vértices da malha computacional. Nas malhas triangulares, dois métodos comuns de interpolação são utilizados: baricêntrica e RBF (Radial Basis Function). O método de interpolação baricêntrica é uma abordagem direta que interpola valores dentro de cada elemento triangular usando uma combinação linear dos valores conhecidos nos vértices do triângulo, sendo o método que adotamos neste trabalho para o backtracking devido à sua eficiência e estabilidade. Por outro lado, o método RBF utiliza funções de base radial para interpolar os valores, permitindo uma maior flexibilidade na representação de dados em posições arbitrárias.

No entanto, embora o método RBF ofereça essa flexibilidade, ele é conhecido por sofrer de baixa precisão perto das bordas dos elementos triangulares. Essa baixa precisão é atribuída à maneira como as funções de base radial são distribuídas, resultando em uma interpolação menos confiável em regiões próximas às fronteiras. Como consequência, a interpolação RBF pode gerar campos que não são divergentes nulos, o que significa que as propriedades físicas não estão sendo corretamente representadas, criando artefatos na simulação. Esses artefatos podem distorcer os resultados da simulação e comprometer a precisão do modelo, especialmente em problemas onde a conservação de massa e outras leis físicas são críticas.

Além disso, exploramos a influência dos métodos de interpolação do *backtracking* na simulação, comparando os resultados obtidos com os métodos baricêntrico e RBF, visando identificar as diferenças de desempenho e suas implicações na precisão das previsões de dispersão de fumaça.

4.3 Malhas arbitrárias

Para demonstrar a capacidade do método em lidar com **geometrias complexas** — definidas como domínios com fronteiras de alta curvatura, não-convexos ou com múltiplas regiões de concavidade — criamos malhas triangulares arbitrárias aproveitando uma biblioteca de malhas de países, que fornece uma representação precisa das fronteiras de países. Essa biblioteca permite gerar malhas de alta resolução para uma variedade de regiões geográficas, oferecendo uma base sólida para a simulação de dispersão de fumaça em diferentes ambientes. É importante destacar que as malhas de países são descritas apenas pelas fronteiras, sem uma triangulação prévia. Para realizar a triangulação das fronteiras e obter as malhas triangulares necessárias para a simulação, utilizamos um wrapper Python da biblioteca Triangle ([SHEWCHUK, 1996](#)), uma ferramenta conhecida por sua capacidade de realizar triangulações de Delaunay com restrições de maneira eficiente e robusta. Vale ressaltar que a triangulação pode gerar malhas com características que dificultam a solução de problemas de interpolação, especialmente quando utilizamos métodos baseados em RBF (Radial Basis Function). Essas malhas complexas podem apresentar regiões onde a solução do problema de RBF se torna instável ou até mesmo impossível de ser determinada devido à natureza irregular da malha.

4.3.1 Precisão do RBF

Observamos resultados mais precisos em malhas com topologia convexa e sem uma densidade de vértices alta na fronteira, pois, na solução de problemas por interpolação por funções de base radial (RBF), a proximidade entre vértices pode gerar problemas na obtenção de resultados precisos. Isso se deve à maneira como os pesos da função de interpolação são calculados. A função RBF utiliza uma operação que eleva a distância entre os vértices a um expoente para determinar a influência de cada ponto na interpolação. Essa operação pode levar à instabilidade na solução quando os vértices estão muito próximos uns dos outros. Em casos de extrema proximidade entre vértices, a matriz de coeficientes do sistema linear a ser solucionado torna-se mal condicionada. Isso significa que a solução pode ser altamente sensível a pequenas alterações nos dados, resultando em resultados imprecisos e oscilações indesejáveis na função interpolada.

4.4 Tratamento de Fronteiras

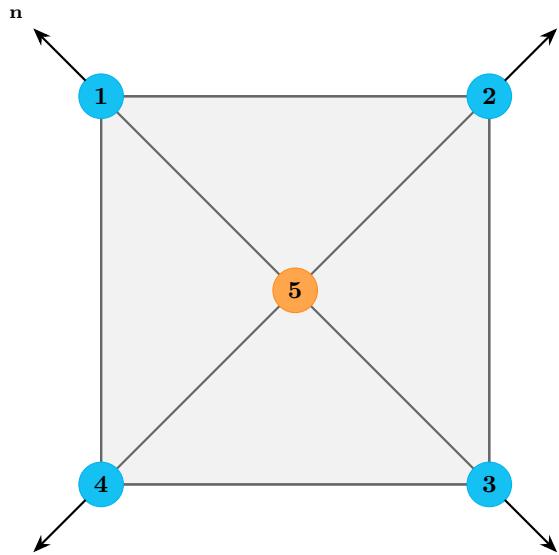
O tratamento adequado das fronteiras é essencial para manter a precisão e estabilidade do método RBF-FD em malhas não-estruturadas. Nossa abordagem para o tratamento de fronteiras representa uma simplificação significativa da proposta por [Flyer, Barnett e Wicker \(2016\)](#), mantendo as vantagens de precisão enquanto reduz a complexidade computacional. Nesta seção, apresentamos inicialmente o método tradicional de *collocation* e suas limitações, seguido pela

técnica de nós fantasma e a formulação matemática completa para sua aplicação na solução da equação de Poisson.

4.4.1 Método de Collocation para Condições de Neumann

A imposição de condições de contorno de Neumann na resolução da equação de Poisson pode ser realizada através do método de *collocation*. Este método substitui a equação diferencial nos nós de fronteira pela equação da condição de contorno, resultando em um sistema linear que incorpora automaticamente as restrições de fronteira.

Figura 10 – Exemplo de malha para método de *collocation*. Os nós 1, 2, 3 e 4 (azul ciano) estão na fronteira com vetores normais \mathbf{n} apontando para fora, e o nó 5 (laranja) é interno.



Fonte: Elaborada pelo autor.

Para ilustrar este conceito, considere um domínio simples com 5 nós, conforme mostrado na Figura 10, onde os nós 1, 2, 3 e 4 estão na fronteira e o nó 5 é interno. A equação de Poisson $\nabla^2 p = \nabla \cdot \mathbf{u}$, onde \mathbf{u} é o campo vetorial, vamos simplificar e usar $\nabla \cdot \mathbf{u} = b$, deve ser satisfeita em todos os nós, enquanto a condição de Neumann $\frac{\partial p}{\partial \mathbf{n}} = 0$ deve ser imposta nos nós de fronteira.

No método RBF-FD, a aproximação dos operadores diferenciais em um nó \mathbf{x}_i utiliza um conjunto de nós vizinhos, denominado estêncil e denotado por $\mathcal{S}_i = \{\mathbf{x}_j\}_{j=1}^{N_i}$, onde N_i é o número de pontos no estêncil do nó i . Utilizando este conjunto de vizinhos, a derivada normal pode ser aproximada por:

$$\frac{\partial p}{\partial \mathbf{n}} = \nabla p \cdot \mathbf{n} \approx \sum_{j \in \mathcal{S}_i} \omega_j^n p_j = 0 \quad (4.15)$$

onde os pesos da derivada normal são dados por $\omega_j^n = n_x \omega_j^x + n_y \omega_j^y$, sendo ω_j^x e ω_j^y os pesos RBF-FD para as derivadas parciais e $\mathbf{n} = (n_x, n_y)$ o vetor normal unitário à fronteira.

O sistema linear resultante combina as equações do Laplaciano para os nós internos com as equações de Neumann para os nós de fronteira:

$$\begin{bmatrix} \omega_1^n & \omega_2^n & 0 & \omega_4^n & \omega_5^n \\ \omega_1^n & \omega_2^n & \omega_3^n & 0 & \omega_5^n \\ 0 & \omega_2^n & \omega_3^n & \omega_4^n & \omega_5^n \\ \omega_1^n & 0 & \omega_3^n & \omega_4^n & \omega_5^n \\ \omega_1^L & \omega_2^L & \omega_3^L & \omega_4^L & \omega_5^L \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ b_5 \end{bmatrix} \quad (4.16)$$

As primeiras quatro linhas correspondem às condições de Neumann nos nós de fronteira (1, 2, 3 e 4), enquanto a última linha corresponde à equação de Poisson no nó interno (5). Este sistema pode ser resolvido diretamente para obter os valores de pressão que satisfazem simultaneamente a equação de Poisson no interior do domínio e a condição de Neumann na fronteira.

Embora o método de *collocation* seja conceitualmente simples e direto, ele apresenta limitações importantes em termos de precisão. O problema fundamental está na assimetria dos estêncils próximos à fronteira. No exemplo da Figura 10, observe que o nó 1, localizado no canto superior esquerdo, possui vizinhos apenas nas direções para baixo e para a direita. Esta distribuição assimétrica de pontos no estêncil prejudica a qualidade da aproximação RBF-FD dos operadores diferenciais, resultando em erros maiores para nós de fronteira quando comparados aos nós internos.

Para ilustrar esta limitação, considere o cálculo do Laplaciano $\nabla^2 p$ no nó 1. O estêncil para este nó contém apenas os nós 2, 4 e 5, todos localizados em um mesmo semiplano em relação ao nó 1. Esta configuração unilateral dificulta a captura precisa do comportamento da função em todas as direções, especialmente na direção normal à fronteira, onde não há nós disponíveis para balancear o estêncil. Este problema se agrava em malhas não-estruturadas com geometrias complexas, onde a distribuição irregular de nós amplifica a assimetria dos estêncils de fronteira.

Estas limitações motivam a busca por técnicas alternativas que possam melhorar a precisão das aproximações em nós de fronteira, mantendo a estrutura geral do método RBF-FD. A seguir, apresentamos a técnica de nós fantasma, que oferece uma solução elegante para este problema.

4.4.2 Nós Fantasmas

A técnica de nós fantasma foi desenvolvida especificamente para superar as limitações do método de *collocation* descritas anteriormente. A ideia central é balancear artificialmente os estêncils assimétricos de fronteira através da adição temporária de nós virtuais fora do domínio computacional. Estes nós fantasma são posicionados estratégicamente para simetrizar o estêncil,

melhorando significativamente a precisão das aproximações RBF-FD em nós de fronteira.

A Figura 11 ilustra o conceito de nó fantasma em uma malha triangular. O nó de fronteira destacado \mathbf{x}_b (em vermelho) está localizado na fronteira superior do domínio. O nó fantasma \mathbf{x}_g (em azul tracejado) é criado fora do domínio, na direção do vetor normal \mathbf{n} , a uma distância h controlada pelo parâmetro α .

Para cada nó da fronteira \mathbf{x}_b , criamos um nó fantasma \mathbf{x}_g na direção normal à fronteira:

$$\mathbf{x}_g = \mathbf{x}_b + h\mathbf{n} \quad (4.17)$$

onde \mathbf{n} é o vetor normal unitário à fronteira no nó \mathbf{x}_b e h é a distância do nó fantasma, definida como:

$$h = \alpha \cdot \Delta x_{local} \quad (4.18)$$

onde Δx_{local} representa o espaçamento local da malha, calculado como a distância média entre o nó de fronteira e seus vizinhos no estêncil, e α é um parâmetro de ajuste que controla a influência do nó fantasma. Em nossos experimentos, utilizamos $\alpha = 0.1$, valor que oferece um bom equilíbrio entre a melhoria na simetria do estêncil e a estabilidade numérica.

O vetor normal \mathbf{n} em cada nó de fronteira \mathbf{x}_b é calculado como a média normalizada das normais das arestas de fronteira adjacentes ao nó. Se \mathbf{e}_1 e \mathbf{e}_2 são as duas arestas de fronteira que compartilham o nó \mathbf{x}_b , com normais \mathbf{n}_1 e \mathbf{n}_2 respectivamente (obtidas por rotação de 90° do vetor da aresta), então:

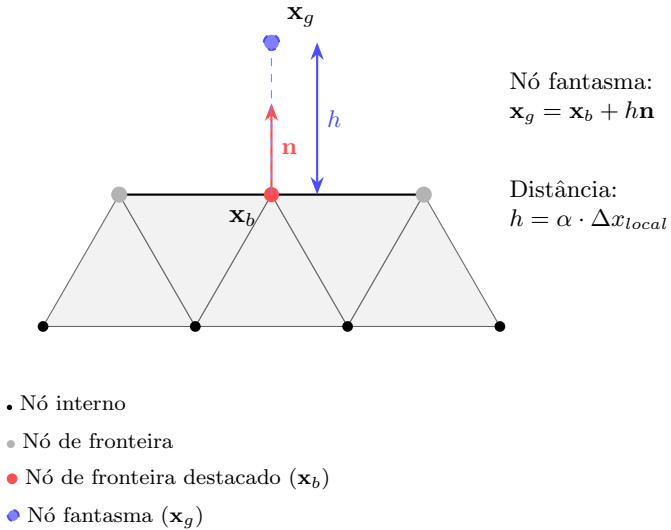
$$\mathbf{n} = \frac{\mathbf{n}_1 + \mathbf{n}_2}{\|\mathbf{n}_1 + \mathbf{n}_2\|} \quad (4.19)$$

Esta abordagem garante que a normal aponte consistentemente para fora do domínio, mesmo em regiões de alta curvatura da fronteira.

4.4.3 Simplificação da Abordagem de Flyer et al.

Embora a técnica de nós fantasma seja promissora, diferentes abordagens para sua implementação podem resultar em complexidades computacionais distintas. A abordagem proposta por Flyer, Barnett e Wicker (2016) para o tratamento de fronteiras com nós fantasma é bastante sofisticada, envolvendo a reflexão de todos os nós internos através da fronteira e a criação de múltiplos nós fantasma. Neste método completo, cada nó de fronteira possui diversos nós fantasma em seu estêncil, oriundos das reflexões de seus vizinhos internos. Esta estratégia resulta em estêncils altamente balanceados e precisão elevada, mas requer a solução de sistemas lineares adicionais para cada configuração de estêncil.

Figura 11 – Ilustração da criação de um nó fantasma para tratamento de fronteiras. O nó fantasma \mathbf{x}_g é criado na direção normal \mathbf{n} a partir do nó de fronteira \mathbf{x}_b , a uma distância h .



Fonte: Elaborada pelo autor.

Nossa implementação adota uma simplificação substancial desta abordagem: para cada nó de fronteira, criamos apenas *um único nó fantasma temporário* na direção normal, conforme ilustrado na Figura 11. Esta simplificação traz vantagens computacionais significativas em diversos aspectos.

Primeiramente, em termos de eficiência computacional, evita-se a necessidade de resolver sistemas lineares adicionais para determinar os valores nos múltiplos nós fantasma, como requerido na abordagem completa de Flyer, Barnett e Wicker (2016). A criação de um único nó fantasma por nó de fronteira também resulta em código mais simples e manutenível, facilitando a implementação e depuração do método. Do ponto de vista de uso de memória, não é necessário armazenar as posições e relações de múltiplos nós fantasma refletidos, reduzindo significativamente os requisitos de armazenamento.

A vantagem mais relevante para a simulação de fluidos está na resolução direta do Laplaciano. Para resolver a equação de Poisson na etapa de projeção, necessitamos apenas dos pesos do operador Laplaciano. Com a simplificação proposta, estes pesos são obtidos diretamente através de uma formulação matemática que combina os pesos padrão do Laplaciano com uma correção baseada na condição de Neumann, conforme será detalhado na próxima subseção.

Esta simplificação é particularmente vantajosa para o problema de simulação de fluidos, onde a equação de Poisson deve ser resolvida a cada passo de tempo. Na prática, construímos a matriz esparsa do Laplaciano uma única vez durante a inicialização, utilizando pesos modificados para os nós de fronteira e pesos padrão RBF-FD para os nós internos. Durante a simulação, a resolução da equação de Poisson reduz-se à multiplicação desta matriz pelos valores de divergência e à solução do sistema linear resultante.

Embora a abordagem simplificada possa resultar em precisão ligeiramente inferior à do método completo de Flyer, Barnett e Wicker (2016) em problemas gerais, ela se mostra adequada para a simulação de fluidos incompressíveis, onde a ênfase está na conservação de massa e na estabilidade da simulação. A próxima subseção detalha os aspectos matemáticos do cálculo dos pesos RBF-FD com nós fantasma.

4.4.4 Cálculo do Laplaciano com Nós Fantasmas

A presença do nó fantasma no estêncil de fronteira afeta especificamente o cálculo do operador Laplaciano. O método RBF-FD para calcular os pesos permanece inalterado; a única modificação é a inclusão temporária do nó fantasma \mathbf{x}_g no estêncil durante o cálculo. O que diferencia esta abordagem é o tratamento subsequente destes pesos para incorporar a condição de contorno de Neumann e eliminar a dependência do nó fantasma.

4.4.4.1 Pesos RBF-FD com estêncil Estendido

Considere um nó de fronteira \mathbf{x}_b com seu estêncil original $\mathcal{S} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$. Ao incluir o nó fantasma \mathbf{x}_g no estêncil, formamos o estêncil estendido $\mathcal{S}_g = \mathcal{S} \cup \{\mathbf{x}_g\}$. Os pesos RBF-FD para os operadores derivadas direcionais (∂_x e ∂_y) e Laplaciano são calculados da mesma forma que para nós internos, resolvendo o sistema linear local definido no capítulo anterior. A diferença é que agora obtemos pesos adicionais correspondentes ao nó fantasma.

Com o estêncil estendido, o operador Laplaciano é aproximado por:

$$\nabla^2 u(\mathbf{x}_b) \approx \sum_{j=1}^N \omega_j^L u_j + \omega_g^L u_g \quad (4.20)$$

onde $u_j = u(\mathbf{x}_j)$ são os valores da função nos nós do estêncil, $u_g = u(\mathbf{x}_g)$ é o valor no nó fantasma, e ω_j^L e ω_g^L são os pesos RBF-FD calculados normalmente com o estêncil estendido. Da mesma forma, obtemos pesos para as derivadas parciais: ω_j^x , ω_g^x , ω_j^y e ω_g^y .

Como o nó fantasma não faz parte do domínio original e não possui valores físicos associados, é necessário eliminar a dependência de u_g na expressão acima. Esta eliminação é realizada através da condição de contorno de Neumann, conforme descrito a seguir.

4.4.4.2 Aplicação da Condição de Neumann

A condição de contorno de Neumann homogênea, utilizada na etapa de projeção da Equação (4.10), pode ser expressa como:

$$\frac{\partial u}{\partial \mathbf{n}} = \nabla u \cdot \mathbf{n} = 0 \quad \text{em } \mathbf{x}_b \quad (4.21)$$

Utilizando RBF-FD para aproximar o gradiente, temos:

$$\nabla u(\mathbf{x}_b) \cdot \mathbf{n} \approx \sum_{j=1}^N \omega_j^n u_j + \omega_g^n u_g = 0 \quad (4.22)$$

onde os pesos da derivada normal são dados por:

$$\omega_j^n = n_x \omega_j^x + n_y \omega_j^y \quad (4.23)$$

sendo ω_j^x e ω_j^y os pesos RBF-FD para as derivadas parciais em x e y , respectivamente, e $\mathbf{n} = (n_x, n_y)$ o vetor normal unitário à fronteira.

Com a condição de Neumann estabelecida, podemos agora expressar o valor do nó fantasma u_g em termos dos valores conhecidos nos nós do domínio e, consequentemente, eliminar sua dependência na aproximação do Laplaciano.

4.4.4.3 Eliminação do Nó Fantasma e Pesos Efetivos

O passo fundamental que modifica o cálculo do Laplaciano em nós de fronteira é a eliminação da dependência do valor no nó fantasma u_g . Como u_g não faz parte do domínio original, precisamos expressá-lo em termos dos valores conhecidos nos nós do domínio. Isso é feito através da condição de Neumann.

Da Equação (4.22), podemos isolar o valor da função no nó fantasma:

$$u_g = -\frac{1}{\omega_g^n} \sum_{j=1}^N \omega_j^n u_j \quad (4.24)$$

Substituindo a Equação (4.24) na Equação (4.20), realizamos a eliminação do nó fantasma:

$$\begin{aligned} \nabla^2 u(\mathbf{x}_b) &\approx \sum_{j=1}^N \omega_j^L u_j + \omega_g^L u_g \\ &= \sum_{j=1}^N \omega_j^L u_j + \omega_g^L \left(-\frac{1}{\omega_g^n} \sum_{j=1}^N \omega_j^n u_j \right) \\ &= \sum_{j=1}^N \omega_j^L u_j - \frac{\omega_g^L}{\omega_g^n} \sum_{j=1}^N \omega_j^n u_j \\ &= \sum_{j=1}^N \left(\omega_j^L - \frac{\omega_g^L}{\omega_g^n} \omega_j^n \right) u_j \\ &= \sum_{j=1}^N \tilde{\omega}_j^L u_j \end{aligned} \quad (4.25)$$

onde os pesos efetivos do Laplaciano são:

$$\tilde{\omega}_j^L = \omega_j^L - \frac{\omega_g^L}{\omega_g^n} \omega_j^n \quad (4.26)$$

Esta é a fórmula central para o cálculo do Laplaciano em nós de fronteira. Os pesos efetivos $\tilde{\omega}_j^L$ combinam os pesos padrão do Laplaciano ω_j^L com uma correção baseada nos pesos da derivada normal. Esta correção, dada pelo termo $-\frac{\omega_g^L}{\omega_g^n} \omega_j^n$, incorpora automaticamente a condição de contorno de Neumann e elimina completamente a dependência do nó fantasma. O resultado são pesos que envolvem apenas os nós do domínio original, mas fornecem aproximações mais precisas do Laplaciano em nós de fronteira devido ao balanceamento do estêncil proporcionado pelo ghost node.

4.4.4.3.1 Considerações sobre estabilidade numérica.

A Equação (4.26) apresenta uma potencial instabilidade quando $\omega_g^n \approx 0$, o que causaria uma divisão por zero ou valores numericamente instáveis. Esta situação pode ocorrer em configurações geométricas específicas onde o nó fantasma está posicionado de forma que sua contribuição para a derivada normal seja aproximadamente nula. Na prática, isso acontece raramente quando o parâmetro α é escolhido adequadamente (em nosso caso, $\alpha = 0.1$), garantindo que o nó fantasma esteja suficientemente afastado da fronteira na direção normal. Em nossa implementação, não observamos este problema nas malhas testadas, mas recomenda-se verificar a magnitude de ω_g^n durante a construção dos pesos para detectar possíveis configurações problemáticas.

Com os pesos efetivos definidos, podemos agora descrever como eles são utilizados na prática para construir o sistema linear global que resolve a equação de Poisson na etapa de projeção.

4.4.4.4 Construção da Matriz do Laplaciano

Na prática computacional, a matriz esparsa do operador Laplaciano é construída uma única vez durante a fase de inicialização da simulação. Para cada nó i do domínio, a linha i da matriz contém os pesos do Laplaciano calculados de acordo com a posição do nó:

$$L_{ij} = \begin{cases} \tilde{\omega}_j^L & \text{se } i \text{ é nó de fronteira (usando Equação (4.26))} \\ \omega_j^L & \text{se } i \text{ é nó interno (pesos RBF-FD padrão)} \end{cases} \quad (4.27)$$

Esta matriz L é então utilizada diretamente na resolução da equação de Poisson durante a etapa de projeção. A cada passo de tempo da simulação, a equação $\nabla^2 p = \nabla \cdot \mathbf{w}$ é resolvida através do sistema linear $L\mathbf{p} = \mathbf{b}$, onde \mathbf{b} contém os valores de divergência calculados em cada nó e \mathbf{p} é o vetor de pressões a ser determinado. A condição de Neumann nas fronteiras já está

automaticamente incorporada nos pesos efetivos $\tilde{\omega}_j^L$, não sendo necessário tratamento adicional das condições de contorno.

Os pesos $\tilde{\omega}_j^L$ resultantes deste processo são pré-calculados e armazenados para todos os nós de fronteira, sendo reutilizados durante toda a simulação sem necessidade de recálculo a cada passo de tempo. Esta estratégia garante eficiência computacional, uma vez que o custo de calcular os pesos com ghost nodes é pago apenas uma vez na inicialização.

É importante notar que, para nós internos, a Equação (4.26) reduz-se naturalmente aos pesos padrão do Laplaciano. Como nós internos não possuem vetor normal à fronteira definido, podemos considerar $\mathbf{n} = (0, 0)$ para estes nós. Consequentemente, pela Equação (4.23), os pesos da derivada normal tornam-se $\omega_j^n = n_x \omega_j^x + n_y \omega_j^y = 0$. Substituindo este resultado na Equação (4.26), obtemos $\tilde{\omega}_j^L = \omega_j^L - \frac{\omega_g^L}{\omega_g^n} \cdot 0 = \omega_j^L$, confirmando que os pesos efetivos coincidem com os pesos RBF-FD padrão para nós internos, conforme especificado na Equação (4.27).

4.4.4.5 Exemplo: Sistema Linear com Nós Fantasma

Para ilustrar a aplicação completa da formulação com nós fantasma, retornemos ao domínio da Figura 10 com 5 nós. Ao contrário do método de *collocation* apresentado na Equação (4.16), onde as linhas correspondentes aos nós de fronteira impõem diretamente a condição de Neumann, o sistema linear com nós fantasma mantém a equação de Poisson em todos os nós, utilizando os pesos efetivos $\tilde{\omega}_j^L$ calculados pela Equação (4.26) para os nós de fronteira:

$$\begin{bmatrix} \tilde{\omega}_{11}^L & \tilde{\omega}_{12}^L & 0 & \tilde{\omega}_{14}^L & \tilde{\omega}_{15}^L \\ \tilde{\omega}_{21}^L & \tilde{\omega}_{22}^L & \tilde{\omega}_{23}^L & 0 & \tilde{\omega}_{25}^L \\ 0 & \tilde{\omega}_{32}^L & \tilde{\omega}_{33}^L & \tilde{\omega}_{34}^L & \tilde{\omega}_{35}^L \\ \tilde{\omega}_{41}^L & 0 & \tilde{\omega}_{43}^L & \tilde{\omega}_{44}^L & \tilde{\omega}_{45}^L \\ \omega_{51}^L & \omega_{52}^L & \omega_{53}^L & \omega_{54}^L & \omega_{55}^L \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix} \quad (4.28)$$

onde $b_5 = (\nabla \cdot \mathbf{w})_5$ é o valor da divergência no nó interno. Embora o sistema com nós fantasma mantenha a equação de Poisson $\nabla^2 p = b$ em todos os nós, o lado direito permanece zero nos nós de fronteira (1, 2, 3, 4), pois a condição de Neumann homogênea já está incorporada nos pesos efetivos $\tilde{\omega}_{ij}^L$.

A diferença fundamental em relação ao método de *collocation* da Equação (4.16) está na matriz de coeficientes. No *collocation*, as quatro primeiras linhas representam a condição $\nabla p \cdot \mathbf{n} = 0$ usando pesos da derivada normal ω_j^n . Já na formulação com nós fantasma, todas as cinco linhas representam o operador Laplaciano $\nabla^2 p$.

As primeiras quatro linhas da matriz utilizam os pesos efetivos $\tilde{\omega}_{ij}^L$ calculados pela Equação (4.26), que diferem dos pesos padrão RBF-FD por incorporarem a correção baseada na condição de Neumann através do termo $-\frac{\omega_g^L}{\omega_g^n} \omega_j^n$. Esta correção é não-nula para nós de

fronteira, pois estes possuem vetor normal $\mathbf{n} \neq (0, 0)$ bem definido, resultando em $\omega_j^n \neq 0$ pela Equação (4.23). Consequentemente, para os nós 1, 2, 3 e 4, temos $\tilde{\omega}_{ij}^L \neq \omega_{ij}^L$, e estes pesos efetivos fornecem aproximações mais precisas do Laplaciano devido ao balanceamento do estêncil proporcionado pelos nós fantasma.

Por outro lado, a última linha, correspondente ao nó interno 5, utiliza os pesos RBF-FD padrão ω_{5j}^L . Como demonstrado anteriormente, para nós internos, $\mathbf{n} = (0, 0)$ implica $\omega_j^n = 0$, e portanto $\tilde{\omega}_{5j}^L = \omega_{5j}^L$ pela Equação (4.26). Assim, embora pudéssemos utilizar a notação de pesos efetivos também na quinta linha, ela seria redundante pois os pesos efetivos coincidem com os pesos padrão para nós internos.

Esta abordagem unificada, onde todos os nós satisfazem a mesma equação diferencial com pesos apropriadamente modificados na fronteira, simplifica consideravelmente a implementação computacional e melhora a precisão das aproximações em nós de fronteira através do balanceamento proporcionado pelos nós fantasma.

A formulação matemática apresentada nesta seção define completamente o tratamento de fronteiras no método RBF-FD. A próxima seção aborda os aspectos práticos da implementação computacional, incluindo as estruturas de dados utilizadas e as otimizações realizadas para garantir eficiência na simulação.

4.5 Aspectos Computacionais

A implementação do método proposto envolveu diversas decisões de projeto visando balancear eficiência computacional com facilidade de desenvolvimento e manutenção. Python foi escolhido como linguagem principal devido à sua versatilidade e rico ecossistema de bibliotecas científicas, embora uma implementação em C++ pudesse oferecer melhor performance.

4.5.1 Otimização de Desempenho

Para mitigar as limitações de desempenho do Python em operações numéricas intensivas, utilizamos a biblioteca Numba para compilação just-in-time (JIT) em pontos críticos do código. Em particular, o algoritmo de intersecção raio-polígono foi otimizado através desta técnica, resultando em uma aceleração significativa no processo de backtracking, essencial para a advecção semi-lagrangiana.

A compilação JIT permite que segmentos críticos do código sejam convertidos em código de máquina otimizado durante a execução, aproximando o desempenho de linguagens compiladas como C++. Esta otimização foi particularmente efetiva nas operações de intersecção geométrica e interpolação, que são executadas intensivamente durante a simulação.

4.5.2 Visualização

A visualização da simulação foi implementada utilizando OpenGL, com uma abordagem direta onde cada vértice da malha é colorido de acordo com a densidade local do fluido. Esta escolha permite uma renderização eficiente em tempo real, essencial para aplicações interativas e validação visual dos resultados.

O sistema de renderização foi projetado para minimizar transferências de dados entre CPU e GPU, mantendo na memória da placa gráfica apenas as informações necessárias para a visualização. A densidade do fluido é mapeada para uma escala de cores que permite a visualização clara das estruturas do escoamento.

4.5.3 Geração de Malhas

A geração de malhas triangulares a partir de contornos de países foi realizada utilizando a biblioteca Triangle ([SHEWCHUK, 1996](#)), que implementa o algoritmo de triangulação de Delaunay com restrições. Essa biblioteca foi escolhida por sua robustez e capacidade de lidar com geometrias complexas. A abordagem permite a construção confiável de malhas computacionais a partir de dados geográficos que originalmente continham apenas informações de contorno.

O processo de geração de malha inclui etapas de pré-processamento para garantir a qualidade dos elementos gerados. Inicialmente, realiza-se a verificação e correção de auto-intersecções nos contornos de entrada. Em seguida, aplica-se refinamento adaptativo baseado na curvatura local, garantindo maior densidade de elementos em regiões de geometria complexa. Por fim, executa-se a suavização dos elementos para melhorar a qualidade geral da malha, otimizando aspectos como razão de aspecto e ângulos internos dos triângulos.

4.5.4 Resolução do Sistema Linear

A resolução da equação de Poisson a cada passo de tempo da simulação requer a solução de um sistema linear esparsa $L\mathbf{p} = \mathbf{b}$. Para sistemas de grande porte, métodos diretos como a decomposição LU podem ser computacionalmente custosos tanto em tempo quanto em memória. Por isso, optamos por utilizar o método iterativo BiCGSTAB (*Biconjugate Gradient Stabilized*), que é particularmente eficiente para sistemas não-simétricos, como o resultante da formulação com nós fantasma.

O método BiCGSTAB é acelerado através de um precondicionador ILU (*Incomplete LU*), que aproxima a decomposição LU da matriz original, melhorando significativamente a taxa de convergência do método iterativo. O precondicionador é calculado uma única vez durante a fase de inicialização da simulação, sendo reutilizado em todos os passos de tempo subsequentes.

A tolerância de convergência utilizada foi de 10^{-10} , valor que garante precisão adequada para a etapa de projeção sem comprometer o desempenho computacional. O custo por iteração do

BiCGSTAB é proporcional ao número de entradas não-nulas da matriz, tornando-o especialmente adequado para as matrizes esparsas geradas pelo método RBF-FD.

4.5.5 Estruturas de Dados

A implementação utiliza estruturas de dados otimizadas para operações comuns em simulações de fluidos. As propriedades do fluido, como densidade, velocidade e pressão, são armazenadas em arrays contíguos em memória, maximizando a eficiência de acesso e permitindo operações vetorizadas. As matrizes de diferenciação RBF-FD são representadas por estruturas esparsas, explorando o fato de que cada linha contém apenas um número pequeno de entradas não-nulas correspondentes ao estêncil local. Para operações de vizinhança, como a busca dos k vizinhos mais próximos durante a interpolação, utilizam-se árvores de busca espacial que reduzem a complexidade computacional destas operações.

Estas escolhas de implementação permitem um equilíbrio entre facilidade de desenvolvimento e desempenho computacional, resultando em um sistema capaz de simular escoamentos de fumaça em malhas não-estruturadas com eficiência satisfatória para aplicações práticas.



RESULTADOS

Neste capítulo, apresentamos os resultados obtidos com o método proposto. Embora a implementação tenha sido inicialmente concebida para ser interativa, a complexidade do tratamento de malhas arbitrárias nos levou a adotar uma abordagem não-interativa. Os resultados são avaliados tanto quantitativamente, através da análise do divergente do campo de velocidades, quanto qualitativamente, por meio da visualização da simulação.

5.1 Análise do Divergente

Uma propriedade fundamental em simulações de fluidos incompressíveis é a condição de divergente nulo no campo de velocidades. Nos experimentos subsequentes, se apresenta a distribuição espacial do divergente na malha em um instante específico da simulação em forma de um gráfico tridimensional, assim como se mostra a evolução temporal do divergente máximo ao longo da simulação em forma de um gráfico bidimensional.

A análise destes resultados revela um aumento gradual do divergente ao longo do tempo, particularmente pronunciado próximo às fronteiras do domínio. Este comportamento está diretamente relacionado às limitações do nosso método em regiões de geometria complexa.

5.2 Visualização da Simulação

Para avaliar qualitativamente o método proposto, realizamos três experimentos distintos de simulação de fumaça em diferentes geometrias. As Figuras 12, 13 e 14 apresentam a evolução temporal de cada experimento em três instantes: inicial, intermediário e final (de cima para baixo).

Em todos os experimentos, utilizamos um **injetor de fumaça** que aplica uma fonte constante de velocidade vertical e densidade em uma região específica da malha. O injetor

permanece ativo durante a primeira metade da simulação (500 frames) e é então desligado, permitindo observar o comportamento do fluido em regime livre. Por isso, é esperado que o divergente seja alto no início da simulação (devido à injeção de massa) e apresente uma queda repentina quando o injetor é desativado.

No primeiro experimento (Figura 12), comparamos o impacto da solução de nós fantasma simplificada que adotamos. Especialmente nesse exemplo, não utilizamos o algoritmo de interseção raio-polígono. Como podemos analisar pelos gráficos, temos uma estabilidade muito maior no valor do divergente a cada passo de tempo e também analisamos que os pontos onde o divergente diferente de zero passaram de estar por toda a malha para estarem só na fronteira.

O segundo experimento (Figura 13) apresenta uma geometria mais complexa, testando a capacidade do método em lidar com fronteiras irregulares. Nota-se que, mesmo com a maior complexidade geométrica, o método consegue capturar os principais fenômenos físicos esperados. Aqui, começam a se manifestar algumas das limitações do método, como os efeitos da perda de massa e instabilidades próximas às fronteiras mais complexas.

No terceiro experimento (Figura 14), exploramos uma configuração ainda mais desafiadora, com fronteiras altamente irregulares. Nesse experimento, podemos analisar que o divergente não se estabiliza em um valor baixo, acarretando na perda de massa ainda mais rápida. Mas ainda assim temos uma qualidade no escoamento do fluido.

Em todos os casos, é possível observar a capacidade do método em reproduzir comportamentos físicos qualitativamente corretos, como:

- Formação e evolução de estruturas vorticais;
- Interação adequada com as fronteiras do domínio;
- Difusão realista da densidade da fumaça.

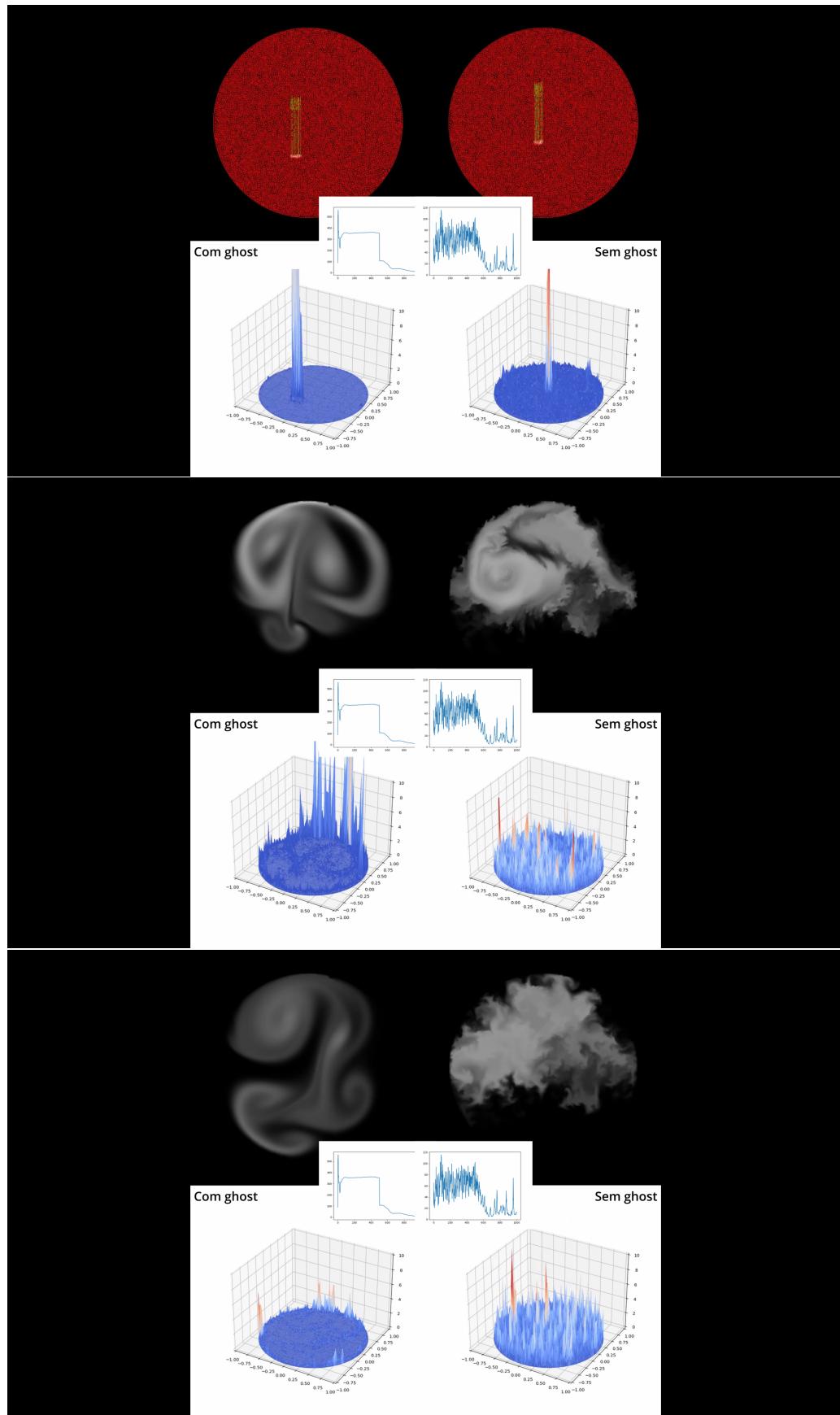
No entanto, também são evidentes alguns artefatos numéricos, particularmente em regiões de alta curvatura da fronteira e nos estágios mais avançados da simulação, onde a perda de massa se torna mais pronunciada.

5.3 Conservação de Massa

Um desafio significativo identificado durante os testes foi a perda gradual de massa ao longo da simulação, como pode ser observado em todos os experimentos. Este fenômeno é principalmente atribuído a dois fatores:

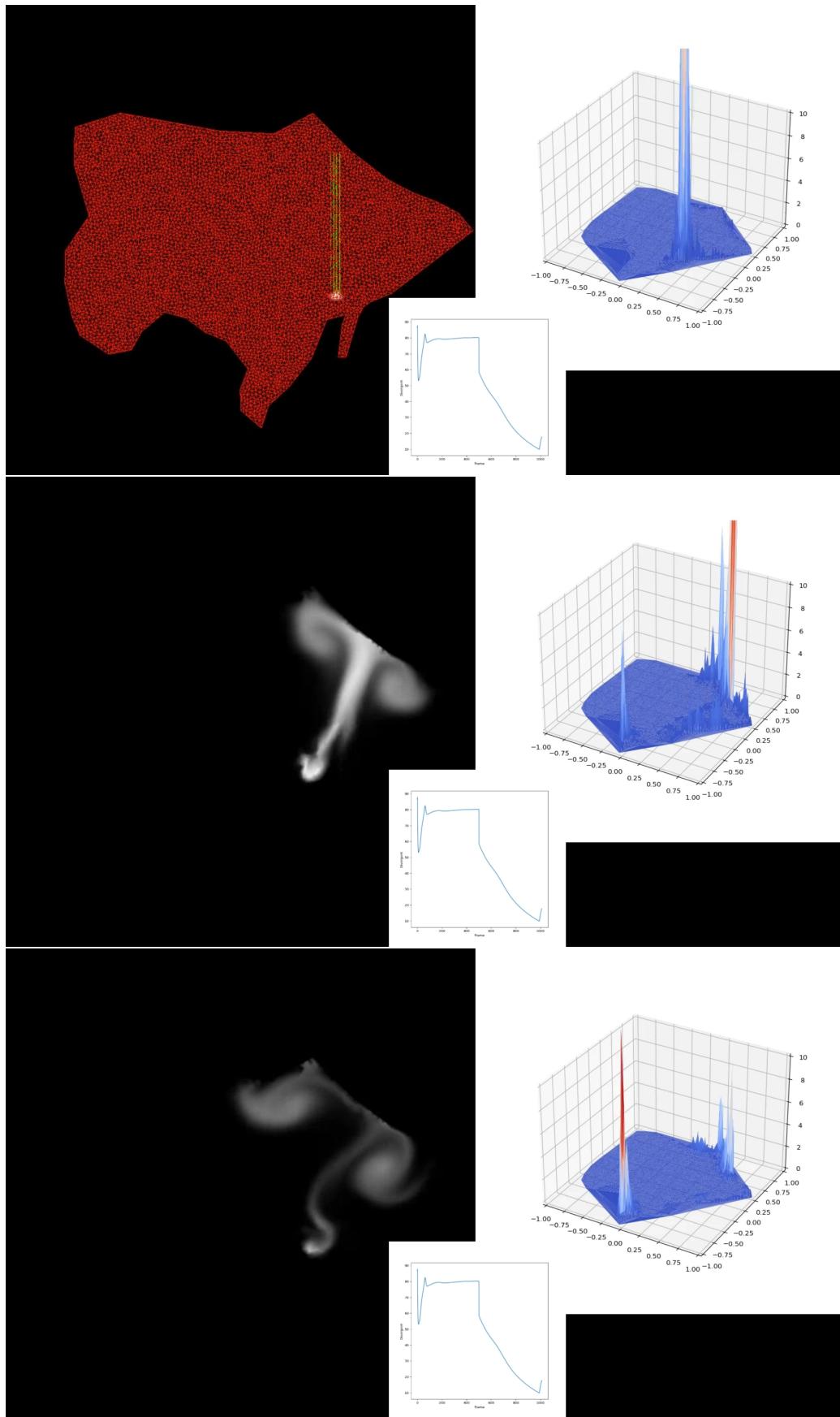
1. Imprecisões na aproximação das condições de contorno junto à fronteira;
2. Instabilidades numéricas em regiões onde nós da malha estão muito próximos.

Figura 12 – Primeiro experimento: Comparação da eficácia da técnica de nós fantasma em malha circular. Cada linha mostra, da esquerda para direita: simulação sem ghost nodes, simulação com ghost nodes, distribuição espacial do divergente (gráfico 3D) e evolução temporal do divergente máximo. As três linhas representam instantes inicial, intermediário e final da simulação, respectivamente.



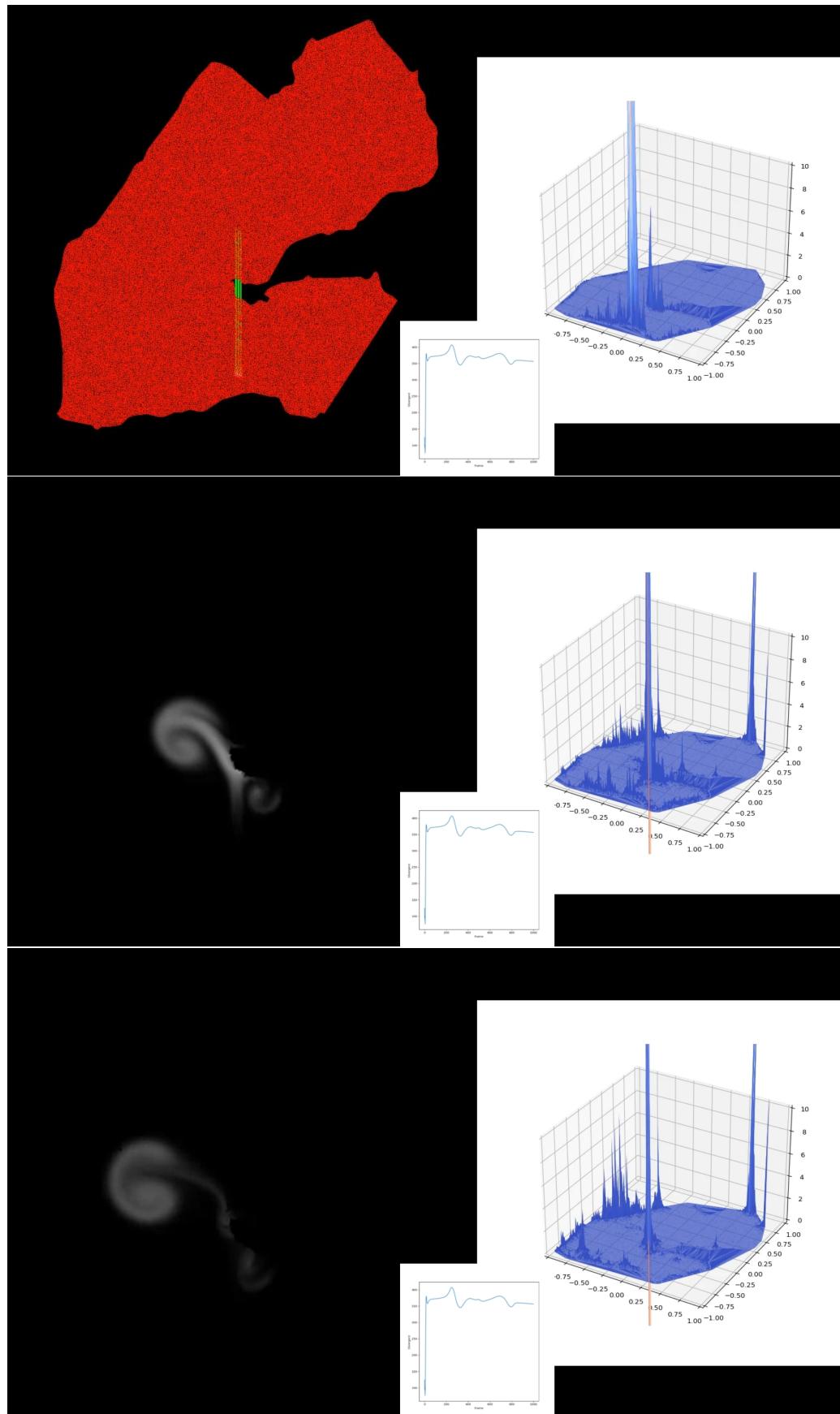
Fonte: Elaborada pelo autor.

Figura 13 – Segundo experimento: simulação de fumaça em geometria irregular (formato de país). Cada linha apresenta, da esquerda para direita: visualização da densidade de fumaça sobre a malha, distribuição espacial do divergente (gráfico 3D) e evolução temporal do divergente máximo. As três linhas correspondem aos instantes inicial, intermediário e final da simulação.



Fonte: Elaborada pelo autor.

Figura 14 – Terceiro experimento: simulação de fumaça em domínio com fronteiras altamente irregulares e regiões de alta curvatura. Cada linha apresenta, da esquerda para direita: visualização da densidade de fumaça sobre a malha, distribuição espacial do divergente (gráfico 3D) e evolução temporal do divergente máximo. As três linhas correspondem aos instantes inicial, intermediário e final da simulação.



Fonte: Elaborada pelo autor.

A escolha da função de base radial poliarmônica mostrou-se particularmente sensível a estas configurações geométricas, contribuindo para a instabilidade do método em fronteiras complexas.



CONCLUSÃO

Este trabalho apresentou uma metodologia para simulação de fumaça em malhas não-estruturadas bidimensionais utilizando o método RBF-FD. A abordagem proposta combina técnicas estabelecidas de simulação de fluidos baseadas no método *Stable Fluids* com uma formulação que permite operar em malhas triangulares arbitrárias através de funções de base radial. Desenvolvemos soluções específicas para desafios inerentes a malhas não-estruturadas, incluindo o tratamento de fronteiras complexas através de nós fantasma temporários e um algoritmo eficiente para determinação de interseção raio-polígono durante a advecção semi-lagrangiana. Os resultados demonstraram que o método é capaz de reproduzir fenômenos físicos qualitativamente corretos, como a formação de estruturas vorticais e a interação adequada com fronteiras irregulares.

As principais contribuições deste trabalho incluem: (i) a adaptação do método *Stable Fluids* para operar em malhas triangulares não-estruturadas utilizando RBF-FD; (ii) o desenvolvimento de uma técnica de nós fantasma temporários que preserva a estrutura original da malha e simplifica a implementação das condições de contorno; (iii) a implementação de um algoritmo eficiente de interseção raio-polígono otimizado com compilação JIT para viabilizar o *backtracking* semi-lagrangiano em malhas arbitrárias; e (iv) a demonstração da viabilidade do método em domínios com geometrias irregulares, incluindo contornos geográficos reais.

6.1 Publicações

Parte dos resultados preliminares desta pesquisa foi apresentada no XLI Congresso Nacional de Matemática Aplicada e Computacional (CNMAC) em 2023, realizado em Bonito, Mato Grosso do Sul. O trabalho intitulado “Animação de fumaça em malhas não-estruturadas usando o método RBF-FD” foi publicado nos anais do evento ([SILVA; PAIVA; PAGLIOSA, 2023](#)) e apresentou os fundamentos da metodologia proposta, incluindo os primeiros experimentos com malhas triangulares não-estruturadas e a adaptação do método semi-lagrangiano para este

contexto. A apresentação no CNMAC proporcionou feedback valioso da comunidade científica, contribuindo para o aprimoramento e refinamento da metodologia desenvolvida nesta dissertação.

6.2 Desafios e Trabalhos Futuros

Durante o desenvolvimento deste trabalho, identificamos alguns desafios que representam oportunidades interessantes para investigações futuras. A conservação de massa ao longo de simulações prolongadas permanece como um aspecto que pode ser aprimorado, principalmente em regiões próximas às fronteiras onde a aproximação das condições de contorno introduz imprecisões. Este comportamento está relacionado à sensibilidade do método RBF-FD à proximidade entre nós e à escolha da função de base radial, particularmente a função poliarmônica utilizada neste trabalho.

Outra direção promissora para trabalhos futuros é a extensão do método para domínios não-simplesmente conexos, como geometrias com buracos ou múltiplas componentes conexas. Esta extensão requer o desenvolvimento de técnicas específicas para o tratamento das descontinuidades topológicas e das condições de contorno em fronteiras internas. Uma abordagem possível seria a decomposição do domínio em regiões simplesmente conexas ou a investigação de formulações alternativas para o cálculo dos pesos RBF-FD em topologias mais complexas.

O desempenho computacional do método também apresenta oportunidades de otimização. Embora tenhamos utilizado compilação JIT para acelerar operações críticas, a exploração de técnicas de paralelização mais sofisticadas poderia viabilizar simulações interativas, expandindo significativamente o potencial de aplicação do método. Além disso, a investigação de funções de base radial alternativas que apresentem melhor comportamento numérico em configurações geométricas desafiadoras pode aumentar a robustez do método.

Uma extensão natural deste trabalho seria a adaptação da metodologia para simulações tridimensionais. A transição de malhas triangulares para malhas tetraedrais mantém a estrutura conceitual do método RBF-FD, porém introduz desafios computacionais adicionais devido ao aumento significativo no número de graus de liberdade. Neste contexto, técnicas de refinamento adaptativo e paralelização se tornam ainda mais críticas para viabilizar simulações em tempo razoável. A extensão para 3D também permitiria explorar aplicações mais realistas em computação gráfica, como simulações de fumaça volumétrica em ambientes complexos.

Por fim, o desenvolvimento de técnicas de pré-processamento para otimizar a distribuição dos nós na malha e a implementação de métodos de correção de massa local representam caminhos naturais para o aprimoramento da precisão do método. Estas melhorias tornariam a abordagem ainda mais robusta e aplicável a uma gama mais ampla de problemas práticos em animação computacional, consolidando o método RBF-FD como uma alternativa viável para simulação de fluidos em domínios com geometrias complexas.

REFERÊNCIAS

FASSHAUER, G. E. **Meshfree Approximation Methods with Matlab**. WORLD SCIENTIFIC, 2007. Disponível em: <<https://www.worldscientific.com/doi/abs/10.1142/6437>>. Citado na página 22.

FLYER, N.; BARNETT, G. A.; WICKER, L. J. Enhancing finite differences with radial basis functions: Experiments on the navier–stokes equations. **Journal of Computational Physics**, v. 316, p. 39–62, 2016. ISSN 0021-9991. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0021999116300195>>. Citado nas páginas 37, 44, 47, 48 e 49.

FOLEY, J. D.; DAM, A. van; FEINER, S. K.; HUGHES, J. F. **Computer Graphics: Principles and Practice**. [S.l.]: Addison-Wesley Professional, 1996. Citado na página 41.

GREAVES, D. A quadtree adaptive method for simulating fluid flows with moving interfaces. **Journal of Computational Physics**, Academic Press Inc., v. 194, p. 35–56, 2 2004. ISSN 00219991. Citado na página 26.

GRESHO, P. M. Incompressible fluid dynamics: Some fundamental formulation issues. **Annual Review of Fluid Mechanics**, v. 23, n. 1, p. 413–453, 1991. Disponível em: <<https://doi.org/10.1146/annurev.fl.23.010191.002213>>. Citado na página 21.

HUANG, Z.; GONG, G.; HAN, L. Physically-based smoke simulation for computer graphics: a survey. **Multimedia Tools and Applications**, v. 74, p. 7569–7594, 2015. ISSN 15737721. Citado nas páginas 19, 20 e 25.

KLINGNER, B. M.; FELDMAN, B. E.; CHENTANEZ, N.; O'BRIEN, J. F. **Fluid Animation with Dynamic Meshes**. 2006. Citado nas páginas 22, 26 e 27.

MAHMOOD, R.; KOUSAR, N.; REHMAN, K. U.; MOHASAN, M. Lid driven flow field statistics: A non-conforming finite element simulation. **Physica A: Statistical Mechanics and its Applications**, Elsevier B.V., v. 528, 8 2019. ISSN 03784371. Citado na página 26.

NAKANISHI, R.; NASCIMENTO, F.; CAMPOS, R.; PAGLIOSA, P.; PAIVA, A. Rbf liquids: An adaptive pic solver using rbf-fd. **ACM Transactions on Graphics**, Association for Computing Machinery, v. 39, 11 2020. ISSN 15577368. Citado nas páginas 26, 27, 28, 31 e 37.

ORUC, O. A radial basis function finite difference (rbf-fd) method for numerical simulation of interaction of high and low frequency waves: Zakharov–rubenchik equations. **Applied Mathematics and Computation**, Elsevier Inc., v. 394, 4 2021. ISSN 00963003. Citado na página 26.

PARK, J.; SEOL, Y.; CORDIER, F.; NOH, J. A smoke visualization model for capturing surface-like features. **Computer Graphics Forum**, v. 29, n. 8, p. 2352–2362, 2010. Disponível em: <<https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2010.01719.x>>. Citado nas páginas 26 e 27.

SHEWCHUK, J. R. **Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator.** 1996. 203–222 p. Applied Computational Geometry Towards Geometric Engineering. Springer-Verlag. Wrapper Python disponível em: <<https://pypi.org/project/triangle/>>. Disponível em: <<https://www.cs.cmu.edu/~quake/triangle.html>>. Citado nas páginas 44 e 54.

SILVA, G. L. da; PAIVA, A.; PAGLIOSA, P. Animação de fumaça em malhas não-estruturadas usando o método rbf-fd. **Proceeding Series of the Brazilian Society of Computational and Applied Mathematics**, v. 10, n. 1, 2023. ISSN 2359-0793. Disponível em: <<https://proceedings.sbmac.org.br/sbmac/article/view/4252>>. Citado na página 63.

STAM, J. Stable fluids. **Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1999**, p. 121–128, 1999. Citado nas páginas 25, 27, 34, 35, 37 e 38.

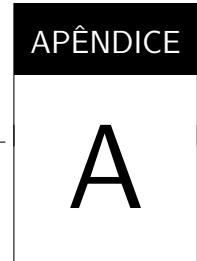
_____. Real-time fluid dynamics for games. **Proceedings of the Game Developer Conference**, v. 18, p. 17, 2003. ISSN 09574174. Disponível em: <<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.12.6736&rep=rep1&type=pdf>>. Citado nas páginas 25, 27 e 37.

TOJA-SILVA, F.; FAVIER, J.; PINELLI, A. Radial basis function (rbf)-based interpolation and spreading for the immersed boundary method. **Computers and Fluids**, Elsevier Ltd, v. 105, p. 66–75, 12 2014. ISSN 00457930. Citado na página 26.

WANG, Z. J. **A QUADTREE-BASED ADAPTIVE CARTESIAN/QUAD GRID FLOW SOLVER FOR NAVIER-STOKES EQUATIONS.** Citado na página 26.

WRIGHT, G. B.; FORNBERG, B. Scattered node compact finite difference-type formulas generated from radial basis functions. **Journal of Computational Physics**, v. 212, n. 1, p. 99–123, 2006. ISSN 0021-9991. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0021999105003116>>. Citado na página 22.

Apêndices



DEPENDÊNCIAS DO PROJETO

Este apêndice lista as principais bibliotecas Python utilizadas na implementação do método proposto, juntamente com suas respectivas versões. A reprodução dos experimentos descritos neste trabalho requer a instalação destas dependências nas versões especificadas ou compatíveis.

A.1 Bibliotecas Principais

NumPy 1.22.4 Biblioteca fundamental para computação numérica, utilizada para operações com arrays e álgebra linear.

SciPy 1.10.1 Biblioteca para computação científica, utilizada para resolver sistemas lineares esparsos (BiCGSTAB com precondicionador ILU), construção de matrizes esparsas e busca de vizinhos mais próximos (KD-Tree).

Numba 0.60 Compilador JIT (Just-In-Time) para Python, utilizado para otimização de operações numéricas intensivas, especialmente no algoritmo de intersecção raio-polígono.

Triangle 20230923 Wrapper Python para a biblioteca Triangle de triangulação de Delaunay, utilizada para geração de malhas triangulares a partir de contornos.

Matplotlib 3.7.1 Biblioteca de visualização, utilizada para geração de gráficos e análise dos resultados.

OpenCV-Python 4.7.0.72 Biblioteca de visão computacional, utilizada para processamento de imagens e gravação de vídeos da simulação.

PyOpenGL 3.1.7 Binding Python para OpenGL, utilizado para renderização em tempo real da simulação.

Shapely 2.0.1 Biblioteca para manipulação e análise de objetos geométricos planares.

libpysal 4.7.0 Biblioteca de análise espacial, utilizada para operações de vizinhança em malhas.

tqdm 4.65.0 Biblioteca para barras de progresso, utilizada durante a inicialização e processamento da simulação.

A.2 Arquivo de Dependências Completo

O arquivo requirements.txt completo, que pode ser utilizado para instalação automática via pip install -r requirements.txt, contém as seguintes especificações:

```
numpy<=1.22.4
scipy==1.10.1
numba==0.60
triangle==20230923
matplotlib==3.7.1
opencv-python==4.7.0.72
PyOpenGL==3.1.7
shapely==2.0.1
libpysal<=4.7.0
tqdm==4.65.0
Pillow==10.0.1
networkx==3.1
scikit-learn==1.2.2
Rtree==1.0.1
trimesh==3.22.0
Cython==0.29.35
```

A.3 Ambiente de Execução

Os experimentos foram executados em um sistema Linux com Python 3.10. A utilização de versões diferentes das bibliotecas listadas pode resultar em comportamentos distintos ou incompatibilidades. Recomenda-se a criação de um ambiente virtual Python para isolar as dependências do projeto.

