

# 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)

S586a      Silva, Gabriel Lucas da  
                  Animação de fumaça em malhas não-estruturadas  
                  usando RBF-FD / Gabriel Lucas da Silva; orientador  
                  Afonso Paiva Neto. -- São Carlos, 2025.  
                  52 p.

Dissertação (Mestrado - 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. 53 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. 53 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> .	17
Figura 2 – Tipos de simulações de escoamento de fluidos	18
Figura 3 – Tipos de células presentes na abordagem euleriana em duas dimensões.	19
Figura 4 – Tipos de estêncis em grades uniformes.	20
Figura 5 – Exemplo de malha conforme.	20
Figura 6 – Exemplo de malha não-conforme, com círculo vermelho para evidenciar uma aresta que mostra a propriedade da malha.	21
Figura 7 – Redução de RBF-FD em FD no caso 1D, mostrando como é necessário estar organizado a vizinhança; colinear e equidistante.	30
Figura 8 – 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.	32
Figura 9 – Exemplo de intersecção raio polígono. Malha em azul, $P_1$ ponto vermelho, direção do raio como seta vermelha, $P_2$ ponto preto, Ponto de intersecção em verde.	38
Figura 10 – Primeiro experimento: Comparação da eficácia da técnica de vértice fantasma.	46
Figura 11 – Segundo experimento: simulação da fumaça em uma geometria simples irregular.	47
Figura 12 – Terceiro experimento: simulação em um domínio com fronteiras irregulares complexas.	48



## **LISTA DE TABELAS**

---

---

Tabela 1 – Resumo dos artigos revisados. . . . .	25
--	----



# SUMÁRIO

---

---

1	<b>INTRODUÇÃO</b>	17
1.1	<i>Objetivos</i>	21
1.2	<i>Roteiro</i>	21
2	<b>REVISÃO BIBLIOGRÁFICA</b>	23
3	<b>FUNDAMENTAÇÃO TEÓRICA</b>	27
3.1	<i>Aspectos numéricos</i>	27
3.2	<i>Método RBF-FD</i>	28
3.3	<i>Animação de fumaça</i>	30
3.3.1	<i>Advecção Semi-Lagrangiana</i>	32
4	<b>METODOLOGIA</b>	33
4.1	<i>Pipeline do Stam</i>	34
4.1.1	<i>Termo de Advecção</i>	34
4.1.2	<i>Termo do Gradiente de Pressão</i>	34
4.1.3	<i>Termo de Força Externa</i>	35
4.1.4	<i>Projeção do Campo de Velocidades</i>	35
4.1.5	<i>Algoritmo Completo</i>	36
4.2	<i>Intersecção raio polígono</i>	36
4.2.1	<i>Algoritmo</i>	36
4.2.2	<i>Desempenho</i>	37
4.2.3	<i>Interpolação</i>	38
4.3	<i>Malhas arbitrárias</i>	39
4.3.1	<i>Precisão do RBF</i>	40
4.4	<i>Tratamento de Fronteiras</i>	40
4.4.1	<i>Vértices Fantasma Temporários</i>	40
4.4.2	<i>Cálculo dos Pesos RBF-FD</i>	41
4.5	<i>Aspectos Computacionais</i>	42
4.5.1	<i>Otimização de Desempenho</i>	42
4.5.2	<i>Visualização</i>	42
4.5.3	<i>Geração de Malhas</i>	42
4.5.4	<i>Estruturas de Dados</i>	43

<b>5</b>	<b>RESULTADOS</b>	<b>45</b>
5.1	Análise do Divergente	45
5.2	Visualização da Simulação	45
5.3	Conservação de Massa	49
5.4	Limitações e Trabalhos Futuros	50
<b>REFERÊNCIAS</b>		<b>51</b>

CAPÍTULO  
1

---

# INTRODUÇÃO

---

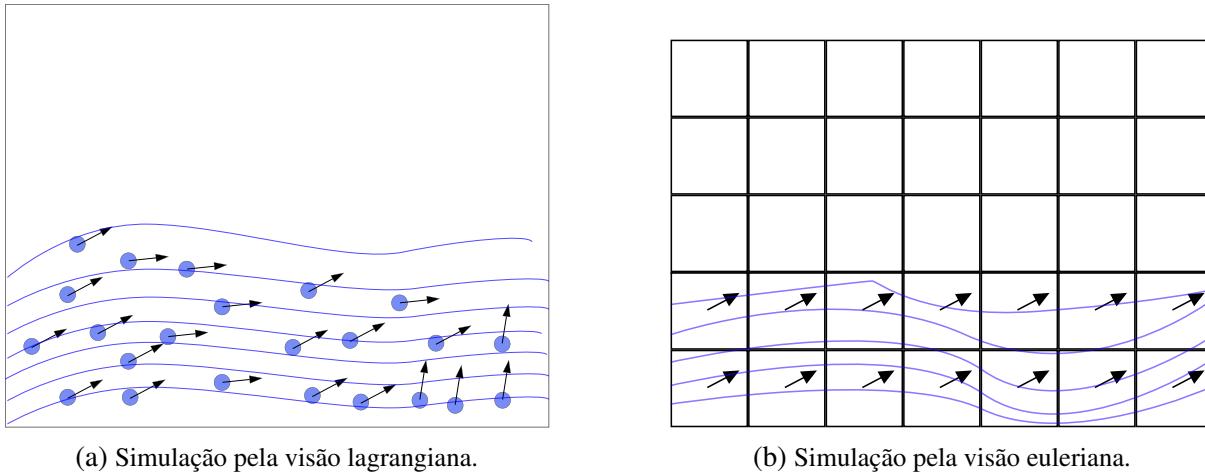
As indústrias do entretenimento, jogos (Figura 1) e engenharia têm grande interesse em simulações de fluidos para integrar em seus respectivos produtos. Em geral, simulações baseadas em física tendem a apresentar resultados visualmente realísticos e convincentes, podendo levar uma pessoa leiga na área a confundir uma gravação real com uma simulação (HUANG; GONG; HAN, 2015). É necessário que o observador da simulação seja convencido da 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 perder o mínimo de detalhes possível 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



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



Fonte: Elaborada pelo autor.

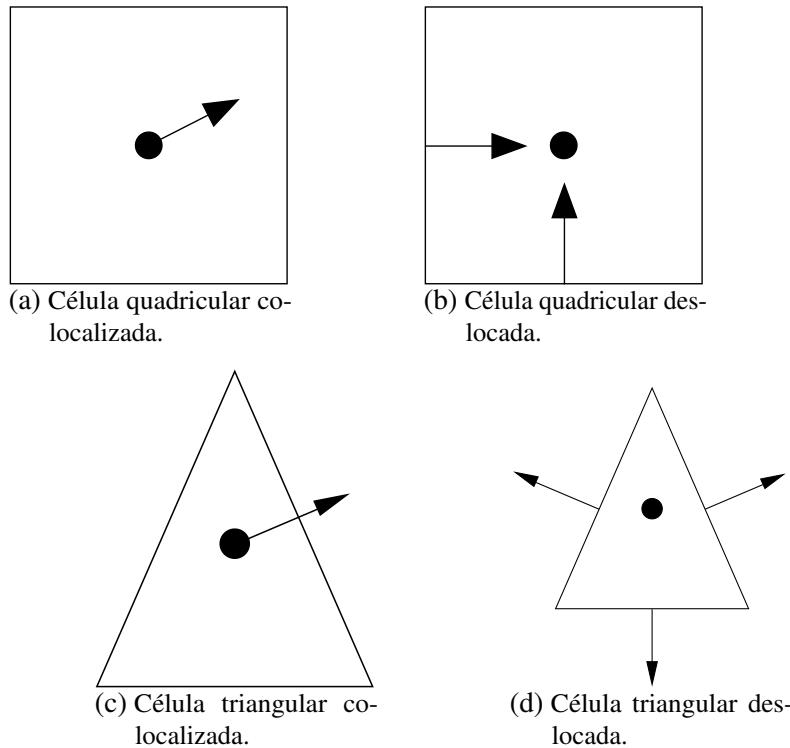
de jogos, e poupar tempo na produção de efeitos especiais em filmes.

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 tetraédrica, hexaédrica ou mistas. Malhas tetraedrais se moldam melhor em fronteiras irregulares, como objetos complexos, porém, este benefício é contrabalançado pelo grande custo computacional. Malhas hexaédricas têm a desvantagem de perder muitos detalhes por conta da estrutura simples. Grades híbridas têm a vantagem de poder usar células tetraédricas em fronteiras com objetos complexos, e células hexaédricas em espaços abertos, melhorando o custo computacional (**HUANG; GONG; HAN, 2015**). Além disso, os valores do campo escalar ou campo vetorial

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



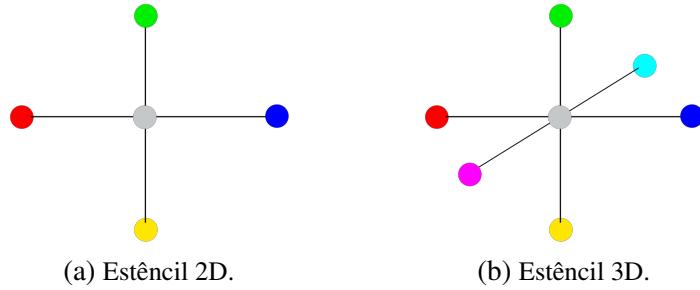
Fonte: Elaborada pelo autor.

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 por meio da abordagem euleriana com uso de malhas tetraedrais deslocadas.

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, no caso 2D são 4 e no 3D são 6, conforme mostrado na Figura 4. Como esta vizinhança é fixa em grades uniformes, podemos aplicar este método de forma eficiente.

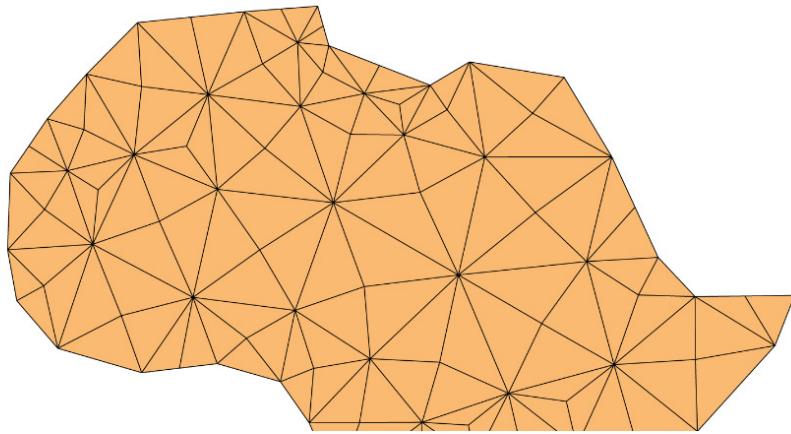
Quando trabalhamos com malhas tetraedrais, temos que aceitar uma malha não-estruturada. Esta propriedade impossibilita o uso de FD, pois a vizinhança de uma célula não é fixa, sendo necessário o uso de outros métodos para a resolução destes operadores diferenciais em malhas

Figura 4 – Tipos de estêncis em grades uniformes.



Fonte: Elaborada pelo autor.

Figura 5 – Exemplo de malha conforme.



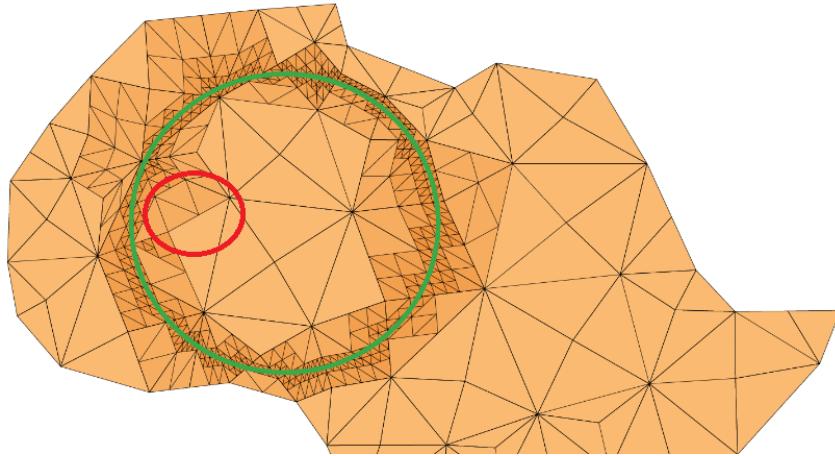
Fonte: ([NASCIMENTO et al., 2014](#))

não-estruturadas, 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 fumaça em malhas não-estruturadas ([KLINGNER et al., 2006](#)).

Outra propriedade importante de malhas que deve ser considerada é a conformidade, que consiste na topologia da mesma. Uma malha conforme (ver Figura 5) contém todas as arestas da face completa, ou seja, não existe um vértice que resida no interior de uma aresta. Na literatura, encontramos diversos problemas geométricos ([MAHMOOD et al., 2019](#); [ČERVENÝ; DOBREV; KOLEV, 2019](#)) em malhas não-conformes (ver Figura 6). No entanto, há uma lacuna na resolução das ENS nesse tipo de malha. Para resolver numericamente as ENS, é interessante utilizar um método que seja independente da malha, isto é, métodos de interpolação sem malha.

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

Figura 6 – Exemplo de malha não-conforme, com círculo vermelho para evidenciar uma aresta que mostra a propriedade da malha.



Fonte: Adaptado de ([NASCIMENTO \*et al.\*, 2014](#))

## 1.1 Objetivos

O objetivo principal deste trabalho é desenvolver uma técnica eficiente de animação de fumaça em malhas não-estruturadas utilizando o método RBF-FD, abordando as limitações atuais na aplicação de condições de contorno em malhas tetraedrais.

## 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 demonstraremos a fundamentação teórica que baseia nossa solução e enunciar 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. Por fim, No capítulo 5, daremos um contexto geral do nosso projeto, explicitando as expectativas, hipóteses e o cronograma para o desenvolvimento do projeto.





## REVISÃO BIBLIOGRÁFICA

---



---

Inicialmente a animação de fumaça foi baseada em métodos não-físicos, sem a utilização das ENS. Isso foi possível 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. Porém, o método sofre gravemente em relação à dissipação numérica. Além disso, o método proposto por Stam faz o uso de grades co-localizadas. A advecção semi-lagrangiana usa 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). Aqui ele mostra mais detalhes sobre a implementação da malha, explicando o método de células fantasmas para auxiliar na computação de operadores diferenciais por meio de FD. Além de trabalhar com as condições de fronteira de Neumann, em que a velocidade das células fantasmas são cópias das velocidades dentro da grade, porém com o sinal trocado, fazendo com que a velocidade do campo na fronteira seja nula. E a condição de fronteira de Dirichlet para o campo de pressão, onde os valores são copiados, mantendo o sinal. Como o nosso foco não é em grades regulares, caso queira se aprofundar mais nesse tópico recomendamos o *survey* (HUANG; GONG; HAN, 2015).

Dos artigos citados, um deles utiliza uma 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 tem dificuldade para comportar domínios(fronteiras) complexos. Logo, a melhor escolha para trabalhar com estas 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 dos benefícios dos dois tipos de malhas. A parte regular é usada

para preencher espaços vazios (sem fluido), já a parte triangular é usada nas fronteiras, perdendo menos 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 tetraedrais se adaptam melhor às fronteiras complexas, o uso de malhas tetraedrais dinâmicas são o foco quando temos obstáculos que podem se mover durante a simulação. Essa é a abordagem usada em (KLINGNER *et al.*, 2006). Como a malha não tem uma estrutura regular, o método numérico utilizado é o de FV. O cálculo da malha é feita a cada passo de tempo da simulação. Logo, a geração da malha deve ser eficiente e rápida. O tipo geração de malha escolhida foi a de Delaunay, em que o refinamento é complexo, pois mantém a conformidade da malha. Fez-se necessário a alteração da advecção semi-lagrangiana para se adaptar à troca de malhas durante o passo de tempo. Além disso, a reconstrução da malha utiliza uma técnica no qual a malha se adapta bem à obstáculos, e é refinada baseado em critérios arbitrários.

Métodos com malha baseada em árvores são famosos em abordagens adaptativas (WANG, ; GREAVES, 2004; NAKANISHI *et al.*, 2020), pois esta estrutura tem facilidade em aumentar ou diminuir a resolução da malha em regiões específicas (refinamento). A principal desvantagem é que o refinamento de *quadtrees* (2D) e *octrees* (3D) podem gerar malhas não conformes. Isto causa complicações nos métodos numéricos mais tradicionais (FD e FV), criando a necessidade de resolver independentemente a determinação dos vizinhos nestes pontos de não conformidade. Uma solução é usar RBF-FD para conseguir fazer a interpolação livre de malha. Apesar de existirem diversos trabalhos que usam RBF-FD para resolução de uma Equações Diferenciais Parciais (EDP) (ORUC, 2021; TOJA-SILVA; FAVIER; PINELLI, 2014), incluindo as ENS (MAHMOOD *et al.*, 2019). O primeiro artigo à 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). 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, não foi encontrado na literatura um método capaz de fazer animação de fumaça em uma malha não-estruturada e não-conforme. Apesar de encontrar resultados relevantes (ORUC, 2021; TOJA-SILVA; FAVIER; PINELLI, 2014), nenhum deles satisfaz todos os requisitos apresentados na Tabela 1. O trabalho que mais cumpre todos os requisitos é o método proposto por Nakanishi et al. (NAKANISHI *et al.*, 2020). Entretanto, o foco deste trabalho é fazer uma grade adaptativa, e a estrutura de dados é baseada em árvore. A nossa proposta é preencher essa lacuna na literatura e produzir uma simulação de fluidos em animação computacional com uma malha de triângulos/tetraedros não-conforme.

Tabela 1 – Resumo dos artigos revisados.

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

Fonte: Feito pelo autor.





# FUNDAMENTAÇÃO TEÓRICA

---



---

## 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:

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

Porém, como o nosso domínio é discreto, precisamos adaptar este conceito. Assumindo uma grade regular com uma distância  $\Delta x$  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:

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

Definindo um campo vetorial 3D  $\mathbf{u} = (u, v, w)^\top$  e um campo escalar  $p$ . Os operadores diferenciais utilizados, gradiente, divergente e laplaciano, respectivamente, tem a seguinte forma:

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

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

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

E suas respectivas formas com a aproximação com FD:

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

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

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

## 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  $\phi$ , 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,  $\{\mathbf{x}_k\}_{k=1}^N$ , os valores das funções  $y_k = y(x_k) \in \mathbb{R}$ , queremos achar 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  $\Pi_m^d$  de todos os polinômios multivariados com grau  $\leq m$ . Para garantir que os coeficientes  $\alpha_k$  e  $\beta_j$  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  $A_{ij} = \phi(\|x_i - x_j\|)$ ,  $\mathbf{P}$  é uma matriz de ordem  $N \times M$  e  $P_{ij} = P_j(x_i)$ ,  $\mathbf{O}$  é uma matriz nula de ordem  $M$ ,  $\boldsymbol{\alpha} = [\alpha_1, \dots, \alpha_N]^\top$ ,  $\boldsymbol{\beta} = [\beta, \dots, \beta_M]^\top$ ,  $\mathbf{y} = [y_1, \dots, y_N]^\top$  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  $\omega$  são obtidos resolvendo o seguinte sistema:

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

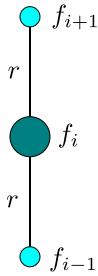
O método de FD é bastante usado em malhas regulares, pois, se beneficia da uniformidade do espaçamento das amostras. Quando trabalhamos com RBF-FD, temos uma generalização de FD para uma nuvem de pontos qualquer. Caso essa vizinhança esteja organizada de maneira uniforme, como uma grade regular, temos a redução de RBF-FD para FD. No caso 1D, ilustrado pela Figura 7, 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,

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

Figura 7 – Redução de RBF-FD em FD no caso 1D, mostrando como é necessário estar organizado a vizinhança; colinear e equidistante.



Fonte: Elaborada pelo autor.

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

### 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:

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

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

onde  $\mathbf{u}$  denota o campo de velocidade,  $p$  a pressão e  $\rho$  a densidade de 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, este é 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 termo gradiente de pressão local,  $\frac{1}{\rho} \nabla p$ . O fluido escoa conforme o gradiente negativo da pressão. O terceiro termo,  $\nu \nabla^2 \mathbf{u}$ , é o termo de viscosidade ou difusão, este considera forças de cisalhamento. A constante  $\nu$  é o coeficiente de viscosidade dinâmica, com efeito de gerar resistência ao escoamento, criando forças que difundem as velocidades através do fluido e pode criar turbulências se existirem gradientes de alta velocidade. O quarto e último termo é o termo de forças externas, este exerce forças como a gravidade  $\mathbf{g}$  e força centrífuga que atuam uniformemente através do fluido. Além de também poder incluir forças de interação do usuário com a simulação.

Quando tratamos de fumaça podemos considerar 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 de Stam pode ser simplificar 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  $\Delta t$  é o passo de tempo da simulação.

**Passo 2:** Este se resume ao transporte de propriedades do fluido ao longo do escoamento, este passo é resolvido pela técnica semi-lagrangiana. Que consiste em pegar 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*), achando 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 8). 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 precisa 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:

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

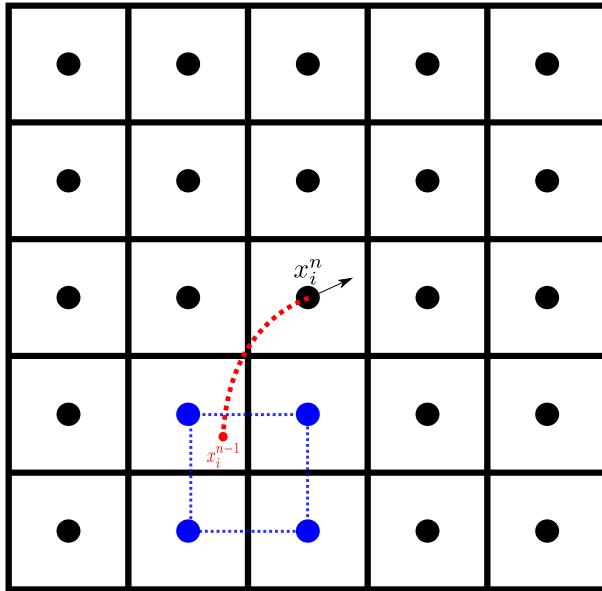
Aplicando o divergente dos dois lados, pela Equação (3.26) segue que:

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

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:

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

Figura 8 – 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.

### 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:

Retrocede o ponto  $\mathbf{x}$  através do campo de velocidades  $\mathbf{w}_1$  por um tempo  $\Delta t$ . Define uma trajetória  $\mathbf{p}(\mathbf{x}, s)$  correspondente a uma linha de corrente parcial. E atribui a nova velocidade como sendo a velocidade que a partícula tinha em sua posição anterior (Figura 8):

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

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.



## 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 de [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 vértices 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.

O método é avaliado analisando as limitações conhecidas da abordagem que também são discutidas, proporcionando uma visão completa e crítica da proposta.

## 4.1 Pipeline do Stam

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:

Para cada ponto  $\mathbf{x}_i$  da malha computacional no tempo  $t^n$ , calculamos a posição de origem  $\mathbf{x}_i^*$  através do *backtracking*:

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

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 stencil:

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

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

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 stencil de  $N$  pontos, temos:

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

onde  $w_{ij}^{\nabla}$  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:

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

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  $\beta$  é o coeficiente de expansão térmica,  $T$  é a temperatura local,  $T_{\infty}$  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:

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

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

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

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

$$\sum_{j=1}^N w_{ij}^{\nabla^2} p_j = (\nabla \cdot \mathbf{w})_i \quad (4.9)$$

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  $\nabla p_i$  é calculado usando a Equação (4.4).

### 4.1.5 Algoritmo Completo

O algoritmo completo para a simulação do escoamento pode ser resumido nas seguintes etapas:

Atualização das forças externas usando as Equações (4.5) e (4.6) Advecção do campo de velocidades através das Equações (4.1)-(4.3) 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 Intersecção raio polígono

A intersecção entre um raio e um polígono é um problema comum em computação gráfica e geometria computacional, fundamental em diversas aplicações, como jogos, simulações físicas e renderização de imagens. Um algoritmo amplamente utilizado para calcular essa intersecção é o algoritmo de intersecção de raio e polígono baseado no método de "Ray Casting"(lançamento de raio). Esse método consiste em disparar um raio a partir de um ponto de origem na direção desejada e determinar quantas vezes ele cruza as arestas do polígono. O algoritmo examina cada aresta do polígono para verificar se ocorre uma intersecção com o raio. Utilizando conceitos matemáticos como o teste de intersecção entre segmentos de reta e o teste de paridade, o algoritmo determina se o raio intersecta o polígono e, se o fizer, onde exatamente essa intersecção ocorre.

### 4.2.1 Algoritmo

Para calcular a intersecção entre um raio e um polígono em um ambiente 2D, focamos apenas na intersecção do raio com as arestas do polígono. Um método eficaz para isso é o algoritmo de intersecção de segmento de linha, como o algoritmo de Cohen-Sutherland ou o algoritmo de Liang-Barsky. 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.

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

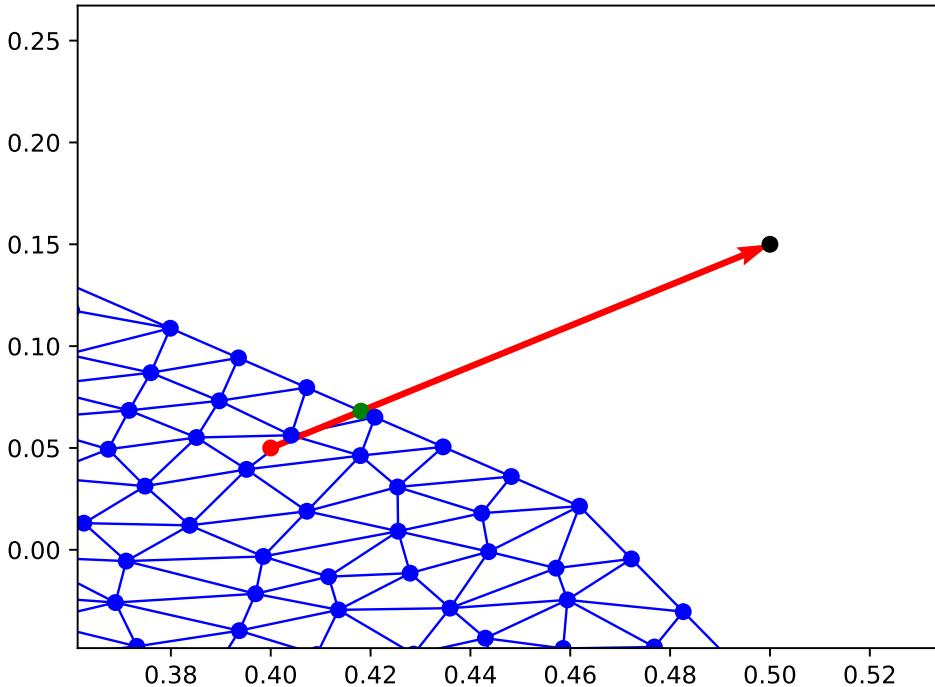
### 4.2.2 Desempenho

É possível aplicar uma vetorização a esse cálculo para tentar melhorar o desempenho, fazendo com que seja calculada a intersecção 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 múltiplos segmentos de aresta de um polígono em um ambiente 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 múltiplos segmentos de aresta de um polígono em um ambiente 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 por bibliotecas como Numba em Python. Numba é uma biblioteca de Python que compila funções Python em código de máquina otimizado, melhorando significativamente o desempenho, especialmente para operações numéricas intensivas.

Figura 9 – Exemplo de intersecção raio polígono. Malha em azul,  $P_1$  ponto vermelho, direção do raio como seta vermelha,  $P_2$  ponto preto, Ponto de intersecção em verde.



Fonte: elaborado pelo autor

Ao utilizar Numba, podemos decorar a função que calcula a interseção do raio com múltiplos segmentos de aresta 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 em Python, podemos melhorar significativamente o desempenho do algoritmo de cálculo da interseção do raio com múltiplos segmentos de aresta, tornando-o adequado para lidar eficientemente com conjuntos de dados maiores em tempo real ou aplicações de alto desempenho.

#### 4.2.3 Interpolação

O algoritmo de backtracking e os métodos de interpolação estão conectados por sua aplicação em problemas de otimização e modelagem computacional. Enquanto o algoritmo de backtracking é utilizado para explorar soluções em espaços de busca complexos, ajustando

continuamente o caminho para alcançar uma solução ótima, os métodos de interpolação são empregados para estimar valores desconhecidos em regiões discretizadas, preenchendo lacunas no conhecimento dos dados. Essas técnicas são frequentemente combinadas em aplicações que envolvem modelagem numérica, como simulações de sistemas físicos ou computacionais, onde a precisão e a eficiência são essenciais. Por exemplo, em simulações de fluxo de fluidos, o algoritmo de backtracking pode ser utilizado para otimizar a distribuição de propriedades em um sistema fluido, enquanto os métodos de interpolação são usados para preencher lacunas nos dados observados, permitindo uma representação mais precisa do comportamento do fluido em regiões onde as medições são escassas ou inexistentes. Assim, a combinação dessas técnicas oferece uma abordagem poderosa para resolver uma variedade de problemas de modelagem e otimização em ciência e engenharia.

Nas malhas triangulares, usamos dois métodos comuns de interpolação: trilinear e RBF (Radial Basis Function). O método trilinear é uma abordagem direta que interpola valores dentro de cada célula triangular usando uma combinação linear dos valores conhecidos nos vértices da célula. Por outro lado, o método RBF utiliza funções de base radial para interpolar os valores, permitindo uma maior flexibilidade na representação de dados irregulares.

No entanto, embora o método RBF ofereça essa flexibilidade, ele é conhecido por sofrer de baixa precisão perto das bordas das células 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 trilinear e RBF (Radial Basis Function), visando identificar as diferenças de desempenho e suas implicações na precisão das previsões de dispersão de fumaça.

## 4.3 Malhas arbitrárias

Para criar malhas triangulares arbitrárias, aproveitamos uma biblioteca de malhas de países, que fornece uma representação precisa da topografia global em forma de fronteiras. 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. No entanto, 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 porte da biblioteca Triangle para Python, uma

ferramenta conhecida por sua capacidade de realizar triangulações de maneira eficiente e robusta. Essa abordagem possibilitou a triangulação da malha conforme necessário para a simulação. No entanto, vale ressaltar que, a triangulação pode gerar malhas com características que desafiam 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

Conseguimos perceber resultados mais concisos em malhas com topologia convexa e sem uma densidade de vértices alta na fronteira, pois, na solução de problemas por interpolação radial de base (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, conhecida como "métrica de dispersão", pode levar a 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 difere da proposta por Flyer, Barnett e Wicker (2016) ao implementar um sistema de vértices fantasma temporário que não modifica a estrutura da malha original.

### 4.4.1 Vértices Fantasma Temporários

Em vez de adicionar permanentemente vértices fantasma à malha computacional, nossa implementação cria vértices fantasma temporários apenas durante o processo de cálculo dos pesos RBF-FD. Para cada vértice da fronteira  $\mathbf{x}_b$ , criamos um vértice fantasma  $\mathbf{x}_g$  na direção normal à fronteira:

$$\mathbf{x}_g = \mathbf{x}_b + h\mathbf{n} \quad (4.15)$$

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

$$h = \alpha \cdot \Delta x_{local} \quad (4.16)$$

onde  $\Delta x_{local}$  é uma medida do espaçamento local da malha e  $\alpha$  é um parâmetro de ajuste que controla a influência do vértice fantasma.

#### 4.4.2 Cálculo dos Pesos RBF-FD

O cálculo dos pesos RBF-FD para vértices próximos à fronteira segue um processo em três etapas:

Inicialmente, identifica-se o estêncil original do vértice de fronteira.

Um vértice fantasma é temporariamente adicionado ao estêncil, expandindo a matriz de interpolação:

$$\begin{bmatrix} \mathbf{A} & \mathbf{a}_g \\ \mathbf{a}_g^T & \phi(0) \end{bmatrix} \begin{bmatrix} \lambda \\ \lambda_g \end{bmatrix} = \begin{bmatrix} \mathbf{b} \\ b_g \end{bmatrix} \quad (4.17)$$

onde  $\mathbf{a}_g$  contém as avaliações da função de base radial entre o vértice fantasma e os demais pontos do estêncil.

Os pesos são calculados considerando a condição de contorno apropriada. Para uma condição de Neumann, por exemplo:

$$\frac{\partial u}{\partial n} = 0 \quad \text{na fronteira} \quad (4.18)$$

Esta abordagem apresenta várias vantagens em relação à adição permanente de vértices fantasma:

- Mantém a estrutura original da malha inalterada
- Reduz o custo computacional por não aumentar o número total de vértices
- Simplifica a implementação das condições de contorno
- Permite ajuste local da influência dos vértices fantasma através do parâmetro  $\alpha$

Os pesos resultantes deste processo são armazenados e utilizados durante toda a simulação, eliminando a necessidade de recálculo a cada passo de tempo e mantendo a eficiência computacional do método.

## 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, que implementa o algoritmo de triangulação de Delaunay com restrições. Esta 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, incluindo:

- Verificação e correção de auto-intersecções nos contornos
- Refinamento adaptativo baseado na curvatura local
- Suavização dos elementos para melhorar a qualidade da malha

#### **4.5.4 Estruturas de Dados**

A implementação utiliza estruturas de dados otimizadas para operações comuns em simulações de fluidos:

- Arrays contíguos em memória para armazenamento eficiente de propriedades do fluido
- Estruturas esparsas para matrizes de diferenciação RBF-FD
- Árvores de busca espacial para operações de vizinhança

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 10, 11 e 12 apresentam a evolução temporal de cada experimento em três instantes: inicial, intermediário e final (de cima para baixo). Em todo começo de simulação é esperado que tenhamos o divergente alto, pois estamos injetando fumaça no campo. Esse injetor é desligado na metade da simulação, por isso a queda repentina no gráfico de divergente máximo pelo tempo.

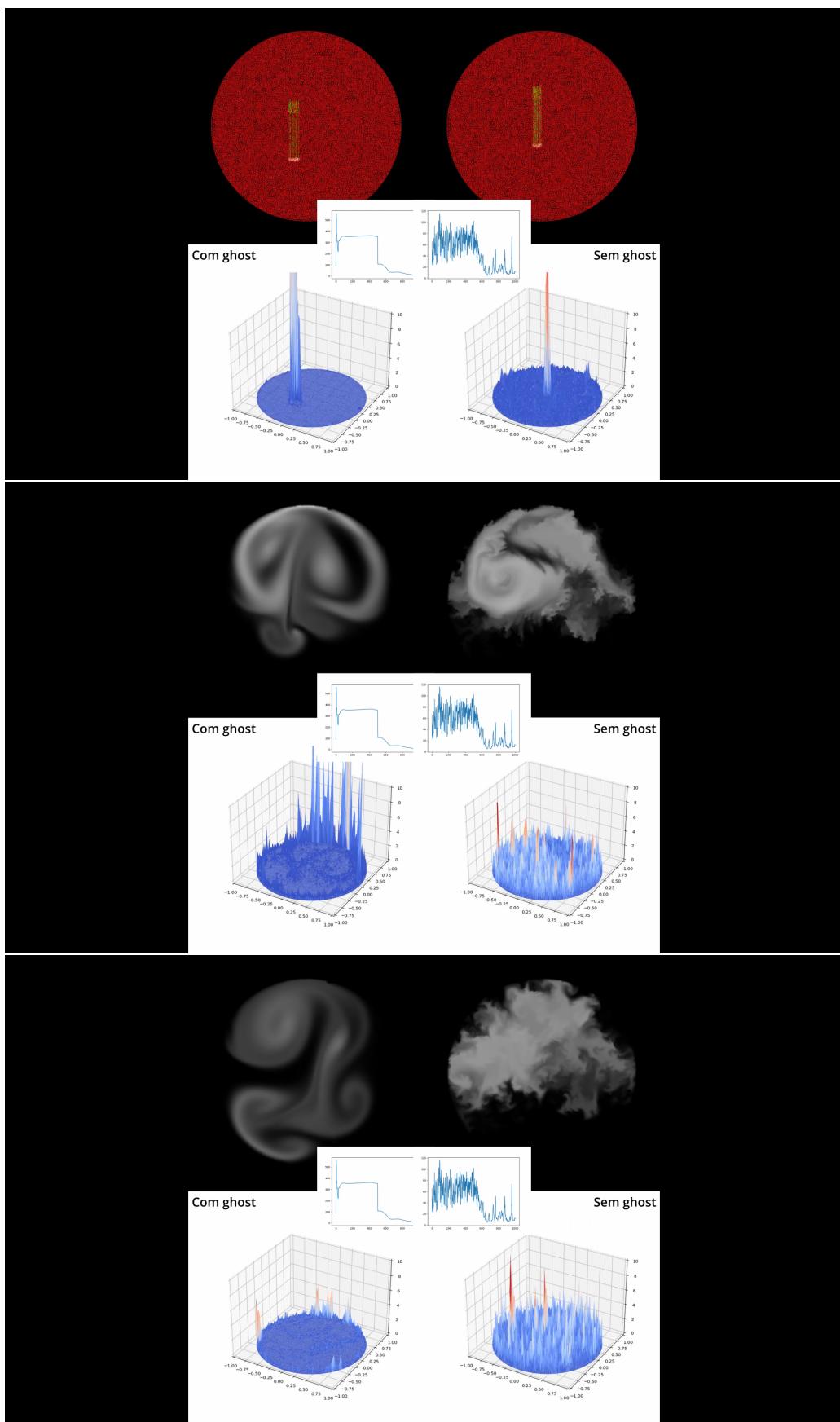


Figura 10 – Primeiro experimento: Comparação da eficácia da técnica de vértice fantasma.

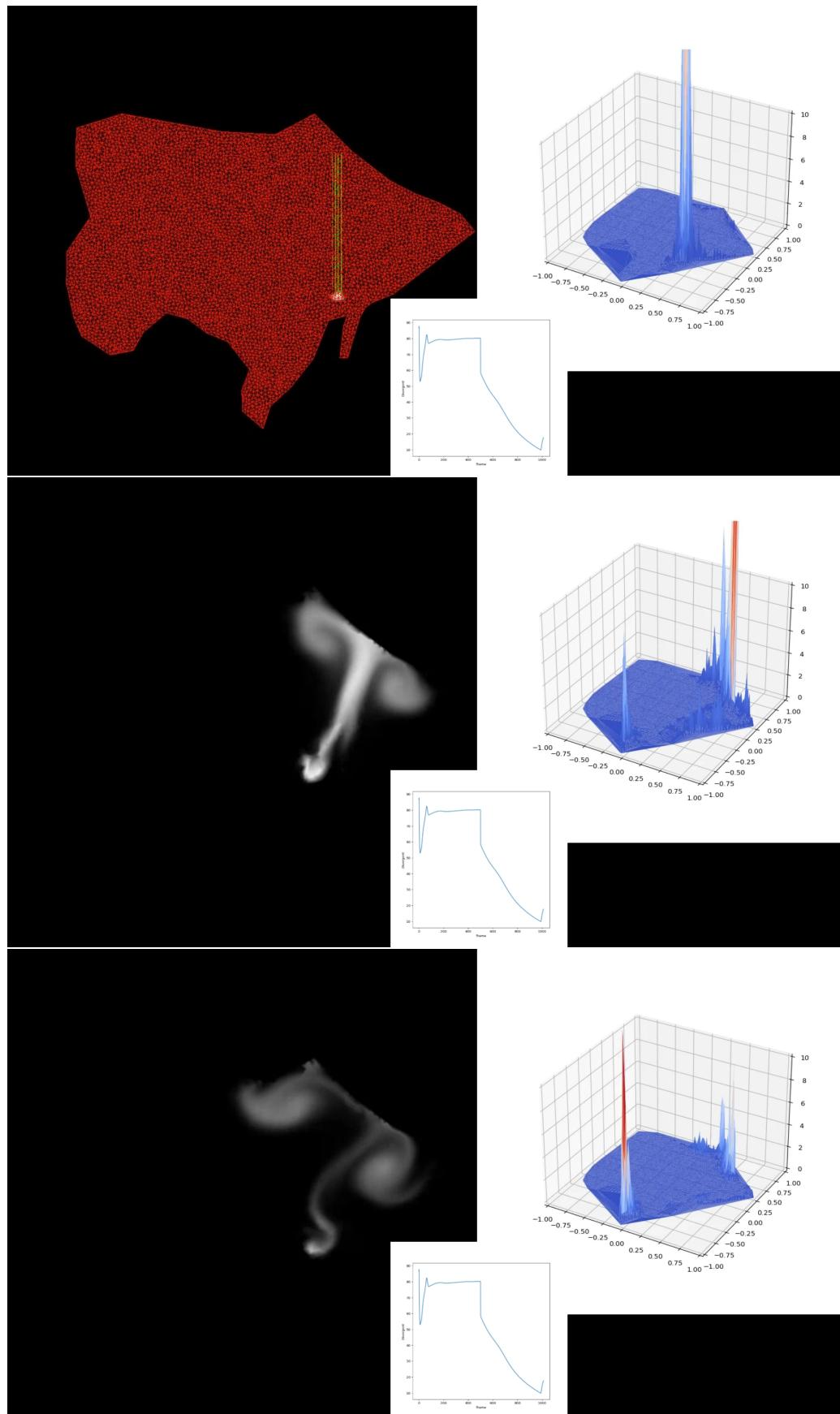


Figura 11 – Segundo experimento: simulação da fumaça em uma geometria simples irregular.

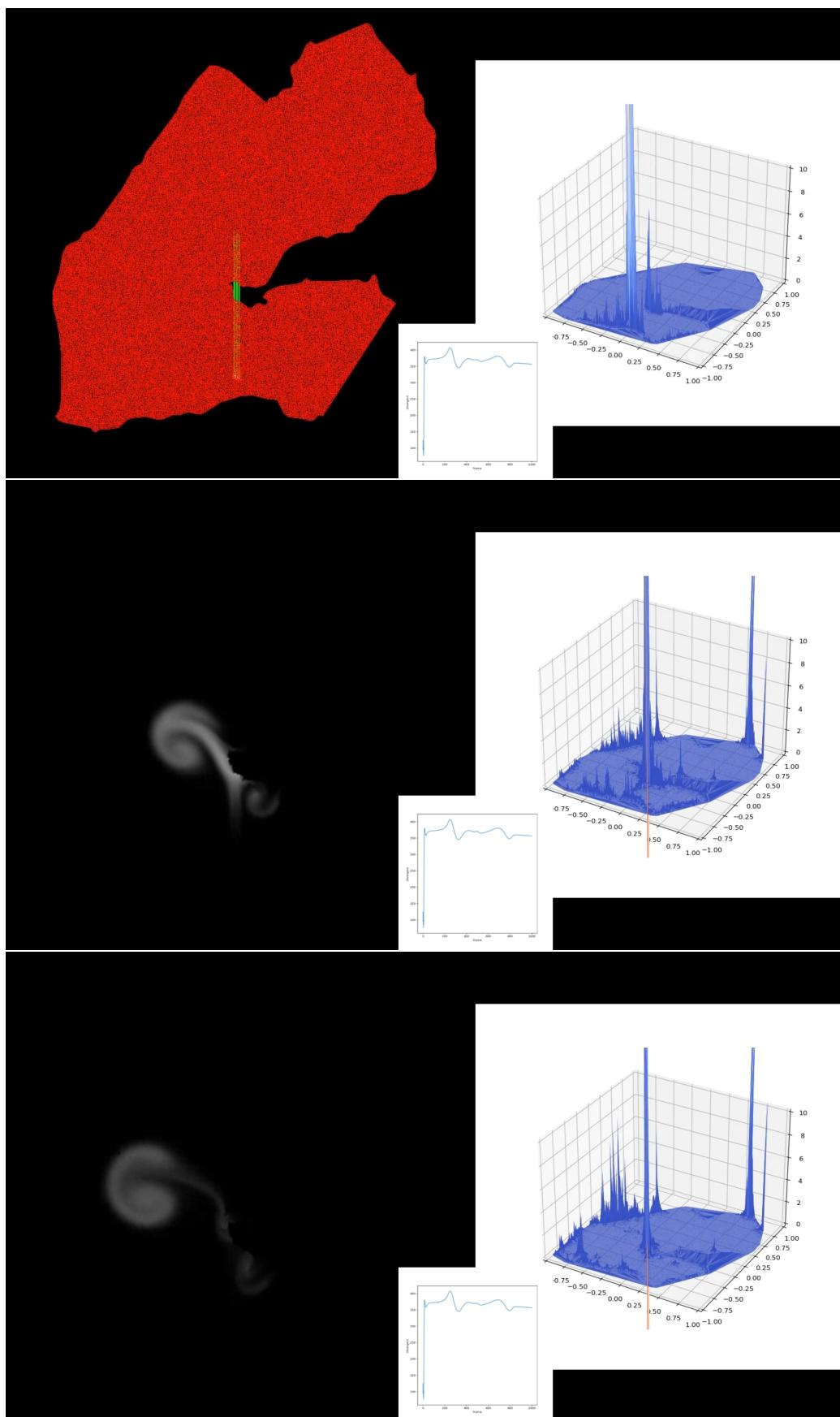


Figura 12 – Terceiro experimento: simulação em um domínio com fronteiras irregulares complexas.

No primeiro experimento (Figura 10), comparamos o impacto da solução de ghost simplificado 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 11) 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 12), 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
- Intereração 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 vértices da malha estão muito próximos

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.

## 5.4 Limitações e Trabalhos Futuros

As principais limitações identificadas em nossa implementação podem ser sumarizadas em:

- Perda de massa ao longo do tempo
- Instabilidade em fronteiras complexas
- Sensibilidade à proximidade entre vértices
- Custo computacional elevado para malhas densas
- Incapacidade de lidar com domínios não-simplesmente conexos

Particularmente relevante foi a tentativa mal sucedida de estender o método para domínios com buracos (domínios não-simplesmente conexos). Esta limitação está possivelmente relacionada a dois fatores: a dificuldade do método RBF-FD em lidar com as descontinuidades topológicas introduzidas pelos buracos e a complexidade adicional no tratamento das condições de contorno em fronteiras internas.

Para trabalhos futuros, sugerimos algumas direções promissoras:

- Investigação de funções de base radial alternativas que apresentem maior estabilidade em geometrias complexas
- Desenvolvimento de técnicas de pré-processamento para otimizar a distribuição dos vértices na malha
- Implementação de métodos de correção de massa local
- Exploração de técnicas de paralelização para viabilizar simulações interativas
- Adaptação do método para suportar domínios não-simplesmente conexos, possivelmente através de:
  - Decomposição do domínio em regiões simplesmente conexas
  - Desenvolvimento de técnicas específicas para tratamento de fronteiras internas
  - Investigação de formulações alternativas para o cálculo dos pesos RBF-FD em topologias complexas

Estas melhorias poderiam endereçar as principais limitações identificadas, tornando o método mais robusto e aplicável a uma gama mais ampla de problemas práticos, incluindo simulações em domínios geometricamente mais complexos.

## 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 20.

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 33 e 40.

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 24.

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 19.

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 17, 18 e 23.

KLINGNER, B. M.; FELDMAN, B. E.; CHENTANEZ, N.; O'BRIEN, J. F. **Fluid Animation with Dynamic Meshes**. 2006. Citado nas páginas 20, 24 e 25.

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 nas páginas 20 e 24.

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 24, 25, 29 e 33.

NASCIMENTO, F. de C.; PAIVA, A.; FIGUEIREDO, L. H. de; STOLFI, J. Approximating implicit curves on plane and surface triangulations with affine arithmetic. **Computers & Graphics**, v. 40, p. 36–48, 2014. Citado nas páginas 20 e 21.

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 24.

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 23 e 25.

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 23, 25, 30, 32, 33 e 34.

\_\_\_\_\_. 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 23, 25 e 33.

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 24.

WANG, Z. J. A QUADTREE-BASED ADAPTIVE CARTESIAN/QUAD GRID FLOW SOLVER FOR NAVIER-STOKES EQUATIONS. Citado na página 24.

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 20.

ČERVENÝ, J.; DOBREV, V.; KOLEV, T. Nonconforming mesh refinement for high-order finite elements. **SIAM Journal on Scientific Computing**, v. 41, p. C367–C392, 1 2019. ISSN 1064-8275. Disponível em: <<https://pubs.siam.org/doi/10.1137/18M1193992>>. Citado na página 20.

