Representação de superfícies livres utilizando partição da unidade implícita no sistema freeflow

Luís Felipe da Costa Ladeira

Representação de superfícies livres utilizando partição da unidade implícita no sistema freeflow

Luís Felipe da Costa Ladeira

Orientador: **Prof. Dr. Antônio Castelo Filho**

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. *VERSÃO REVISADA*

USP – São Carlos
Agosto de 2011

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

dr da Costa Ladeira, Luis Felipe
Representação de superfície utilizando partição da unidade implícita no sistema freeflow / Luis Felipe da Costa Ladeira; orientador Antônio Castelo Filho -- São Carlos, 2011.
96 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, 2011.

1. Representação de superfícies. I. Castelo Filho, Antônio, orient. II. Título.

Para minha família...

Epigrafe

Mesmo que a rota da minha vida me
conduza a uma estrela, nem por isso
fui dispensado de percorrer os caminhos
do mundo.

José Saramago

Agradecimentos

Agradeço primeiramente à minha família que me deu apoio em todas as horas e ocasiões. Meus pais, minha irmã e meus sobrinhos, por todo afeto que me dispensaram sempre.

Presto meu agradecimento ao meu orientador, o professor Antônio Castelo Filho, pela oportunidade de trabalhar e desenvolver um projeto que foi de muito interesse ao longo desses anos. Agradeço também aos professores que me examinaram na banca de qualificação, professores Fabrício Simeoni de Sousa e Gustavo Carlos Buscaglia, pelas sugestões e críticas construtivas. Agradeço meu amigo e professor João Paulo Gois, pelas inúmeras vezes que se mostrou prestativo em ajudar em tempos de dificuldade.

Gostaria de deixar um grande agradecimento à agência FAPESP que amparou minha pesquisa neste projeto de mestrado.

Em especial, gostaria de agradecer minha amiga e companheira Lia, que teve fé em mim e soube deixar seus votos por meu sucesso da forma mais clara e carinhosa possível.

Resumo

Este trabalho consiste em introduzir uma nova abordagem de representação de superfície no ambiente de simulação *Freeflow2D*. Consiste em usar Partição da Unidade Implícita para estimar da superfície a geometria, normais e curvatura. Procurando se valer das vantagens de métodos do tipo *meshless* (sem malha) conservando no entanto a malha Lagrangiana, no interesse de manter o fácil acesso de vizinhança, inserção e eliminação de pontos.

Abstract

The objective of this work is to introduce a new approache of surface representation within the *Freeflow* system. It consists of using implicit functions by means of Partition of Unit Implicit to estimate surface geometry, normals and curvature. Aiming at the advantages of meshless methods of surface representation whilst keeping the Lagrangian mesh in order to preserve ease of access of geometric vicinity, particle insertion and removal.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Contextualização e Motivação

O objetivo deste capítulo é expor uma breve contextualização histórica do trabalho em questão, mencionando trabalhos prévios realizados, a motivação que os levou a serem desenvolvidos e dar exemplos das inúmeras aplicações angariadas graças ao seu desenvolvimento. Serão abordadas separadamente duas linhas de trabalho relevantes para o projeto em questão, a saber: a Dinâmica dos Fluidos Computacional (DFC) e o tratamento computacional da superfície livre. Ademais, será delineada a interdependência entre os assuntos de tal forma que se adequam ao contexto deste trabalho de mestrado. Trataremos em seguida dos objetivos alcançados. Finalmente, o fim do capítulo compreende uma síntese da organização global do conteúdo do documento.

1.1 Dinâmica dos Fluidos Computacional

Desde há muito tempo o estudo de escoamentos de fluidos é objeto de interesse no campo das observações humanas sobre o mundo físico. Até a presente data, o parecer científico tem incorporado a maior quantidade disponível de ferramentas de análise do fenômeno dado que as equações repectetivas aos modelos teóricos apresentam uma notável complexidade. Além de serem representativas no que tange a modelagem em

termos de equações diferenciais de uma vasta gama de evoluções dinâmicas de sistemas – pois das respectivas equações chamadas de Navier-Stokes pode-se extrair, mediante simplificações, equações parciais que modelam diversos outros objetos de estudo da física, química, engenharia e etc. – o estudo de fluidos em movimento em si atesta sua valia dada a ubiquidade dos fenômenos que representam.

De fato, na natureza encontramos como objeto particular de estudo da mecânica dos fluidos em fenômenos tais quais movimento das águas em rios, lagos e mares, de massas de ar, do vento, etc. Como consequência imediata, a maturidade diante da compreensão deste tema largamente presente na natureza angaria resultados cujas aplicações se estendem em bastantes áreas: aerodinâmica (aviação, minimização da perda de energia em transportes marítimos e terrestres ligada à força de arraste, o *design* de veículos), biologia (compreensão do funcionamento de flagelos nas células), metereologia (estudo das variações climáticas, implicações de depredações ambientais e poluição), medicina (hemodinâmica), etc.

A maior parte dos sistemas físicos são descritos por equações diferenciais parciais não lineares. Igualmente, as equações de Navier-Stokes trata de equações de evolução temporal. Sua não linearidade estende consideravelmente sua análise no campo teórico da matemática pura, e apesar de sua aparente simplicidade, seu estudo está longe de acabado. Se, por um lado, dispomos de uma boa teoria de existência e unicidade de soluções regulares para a restrição das equações ao domínio bidimensional, ainda há muito a ser desenvolvido para estender tais resultados para a análise das equações em um domínio tridimensional.

Não obstante, em virtude da mencionada complexidade das equações de Navier-Stokes, a solução analítica da equação está além do alcance das ferramentas matemáticas de que se dispõe atualmente. No interesse de extrair a maior quantidade possível de conhecimento sobre a evolução de escoamento de fluidos segundo várias propriedades materiais e condições iniciais, emerge nos anos 1950 uma nova vertente,

a saber, a mecânica dos fluidos computacional. A subsequente evolução dos computadores viabilizou cálculos de soluções numéricas das equações que culminou em um sistema de retroalimentação na interdependênica do conhecimento numérico, analítico e experimental sobre as equações.

Nesse ínterim, na representação dos fenômenos a serem estudados em modelos computacionais urge fineza, dado que uma aproximação insuficientemente precisa acarreta amiúde em resultados numéricos que não correspondem ao modelo físico do problema. Assim, há relevância no estudo teórico dos modelos computacionais, cada qual podendo representar uma circunstância particular de escoamento.

Dentre estas circunstâncias encontramos o caso do escoamento multi-fásico. Trata-se do escoamento de dois ou mais fluidos que possuem, entre si, propriedades materiais diferentes. Como consequência imediata, os parâmetros das equações de Navier-Stokes divergem entre si, dependendo da região do domínio em que se encontra isto é, de qual fase está a representar.

A região de fronteira entre diferentes fases, chamada interface, requer particular astúcia de modelagem: é frequentemente necessário um número maior de pontos nesta região a fim de que se tenha conhecimento preciso dos desdobramentos da superfície.

1.2 Modelagem Geométrica

Conforme citado acima, um caso de especial interesse da mecânica dos fluidos é o escoamento multifásico. Neste cenário, há mais de um tipo de fluido envolvido na evolução no tempo das equações de Navier-Stokes, o que implica em diferentes propriedades materiais distribuídas ao longo do domínio físico.

A *interface* que separa os diferentes fluidos pode sucumbir a uma série de fenômenos de difícil manipulação no âmbito computacional, como deformações e mudança de tologia. Algumas propriedades geométricas são evocadas no seio da solução das equações

de Navier-Stokes, como o vetor normal da superfície nas condições de fronteira e a curvatura no cálculo da tensão superficial.

Podemos dividir as abordagens de representação da superfície em duas grandes vertentes: capturas de superfície (*front-capturing*) e acompanhamento de superfície (*front-tracking*). Métodos do tipo *front-tracking* desenvolvem algum tipo de estrutura que representa a superfície explicitamente, como as partículas marcadoras do método MAC, que se movem ao longo de uma malha fixa Euleriana.

Nesses métodos de captura de superfície:

- Tem-se a geometria explícita da interface;
- Facilidade na inserção e remoção de pontos;
- Em vista da conectividade na ocasião de representação por malhas lagrangianas, há fácil acesso aos pontos de uma determinada vizinhança;
- Algum procedimento algébrico se faz necessário se quisermos uma estimativa precisa de normais e curvaturas, como aproximações locais;
- Não se lida naturalmente com mudanças topológicas da superfície, a implementação é mais complexa e custosa;
- Embora tenhamos alto custo, essas mudanças topológicas gozam de maior controle quando representadas por malhas, como critérios para romper ou unir duas interfaces diferentes;

A representação de uma superfície a partir de pontos não organizados é um ramo que vem gozando de um rápido crescimento. É um ramo de pesquisa de interesse para modelagem geométrica e computação gráfica. Ela lida com o problema de se construir uma função para recuperar as informações geométricas de uma superfície que daria origem aos pontos não organizados.

Esta área de pesquisa ganhou proeminência na década de oitenta, após o trabalho de Boissonant ([35]). Sua relevância científica pode ser atestada ante a grande área de atuação que seu desenvolvimento aufere, da qual a indústria de entretenimentos (cinema, jogos), a modelagem topográfica, reconstrução de superfície e áreas médicas são exemplos a citar.

Técnicas na linha do *front-capturing* trabalham superfícies como regiões de alta variação, sem representar a interface por elementos explícitos. As mudanças topológicas são retratadas com maior naturalidade, como ruptura ou junção de duas ou mais superfícies. Resumidamente e de forma abrangente, podemos dizer que nestes métodos:

- Variações topológicas são tratadas com maior naturalidade;
- Menor controle de critério nessas mudanças topológicas;
- Por não haver estrutura de malha lagrangiana, o custo das operações pode ser reduzido e grande quantidade de pontos pode ser utilizada;
- Funções indicadoras estão intrinsecamente ligadas à esse tipo de abordagem, tornando fácil o acesso a distâncias algébricas;
- Também é facilitado o acesso a propriedades geométricas como vetores normais à superfície e cálculos da curvatura em qualquer ponto;
- Preciso de algum tipo de árvore ou tabela *hash* para conhecer vizinhança dos pontos;
- Algum processo de reamostragem será evocado no tratamento da superfície, não havendo conectividade entre os pontos (e.g. algoritmos do tipo *marching cubes*).

Tendo em vista essas peculiaridades de cada abordagem, surge naturalmente o interesse em abrigar todas as vantagens possíveis em uma técnica que goze dos benefícios

de cada uma destas técnicas. Falamos então de um método *híbrido*, usar uma representação de superfície do tipo dos métodos de captura de superfície e integrá-la ao lado de uma representação por malhas.

1.3 Objetivos Alcançados

Levantamento bibliográfico. Esta área de pesquisa sendo multidisciplinar envolve diversas vertentes da ciência e da engenharia. Neste trabalho foi feito uma organização dos trabalhos prévios, colocando-os como referência para a realização do projeto desenvolvido e explicando seus respectivos métodos de forma resumida e elucidativa.

Organização didática do conteúdo. Com um capítulo de conceitos básicos para familiarizar o leitor com as ferramentas da matemática, computação e física envolvidas nos algoritmos que compreendem tais métodos de reconstrução de superfície, este documento pretende cuidar da necessidade do leigo buscando se familiarizar com a terminologia científica do trabalho em questão.

Nova abordagem geométrica no sistema *Freeflow2D*. No interesse de elaborar a polivalência do ambiente *Freeflow*, este projeto desenvolveu uma nova maneira de representar a superfície e efetuar cálculos de normal e curvatura. Desde sua concepção original, o ambiente de simulação *Freeflow* havia mantido sua maneira inicial de representação da superfície livre.

1.4 Estrutura da Dissertação

O **Capítulo 2** é dedicado a revisar os conceitos de matemática, física e computação que serão utilizados ao longo do documento. O enfoque foi de familiarizar o leitor com a terminologia dos trabalhos envolvidos, explicar conceitos matemáticos recorrentes na literatura e deduzir as equações de Navier-Stokes. O objetivo deste trabalho de mestrado é aplicar um método baseado em funções implícitas em um ambiente de

simulação de escoamentos. O algoritmo foi desenvolvido no sistema *Freeflow2d*, descrito no **Capítulo 3**. Este sistema abriga o método GENSMAC para resolver escoamentos com superfícies livres. São explicados o algoritmo do método GENSMAC, um algoritmo chamado TSUR-2D responsável por remover ondulações em simulações de alto número de Reynolds, a maneira como as células são catalogadas, a estrutura de dados *half-edge2D* que representa a superfície livre, a maneira como se calculam as normais e a curvatura e como é tratada a mudança de topologia. O **Capítulo 4** discorre sobre métodos estudados que inspiraram a concepção do trabalho e está divido em duas partes: métodos ditos Lagrangianos e métodos baseados em partículas[1]. O projeto desenvolvido será detalhado no **Capítulo 5**, detalhando primeiro o que se dispõe em cada célula (representações locais) e como estas infomarções são agregadas em uma única função (global), via funções do tipo peso e seguindo a teoria da partição da unidade implícita. Em seguida são deduzidos os cálculos para obtenção de normais e curvaturas. Ademais, são dados detalhes da implementação e um algoritmo em pseudo-código. O **Capítulo 6** mostra resultados da superfície em quatro cenários: uma seção preliminar mostrando a partição da unidade funcionando nas representações locais, uma seção estudando convergência de superfícies sem movimento, uma seção sobre movimento rígido e uma seção em que a superfície é utilizada em simulações de mecânica dos fluidos.

[1] Do inglês *Point Based Surfaces*.

Capítulo 2

Conceitos Básicos

O objetivo do presente capítulo é pincelar algumas noções matemáticas de que nos serviremos nos capítulos seguintes e estabelecer uma linguagem e notação do conteúdo posterior. Além disso, será exposto algumas idéias de métodos utilizados na representação de superfície.

2.1 Superfícies e Curvas de Nível

Nesta seção introduzimos uma perspectiva sobre a definição de superfícies implícitas como iso-superfícies de uma função

$$f : \begin{cases} \Omega \subset \mathbb{R}^n & \longrightarrow & \mathbb{R} \\ \vec{x} = (x_1, \cdots, x_n) & \mapsto & f(x_1, \cdots, x_n) \end{cases}.$$

Definição 2.1 *Dada uma constante arbitrária C, um* conjunto de nível[1] *é obtido na união de todos os pontos $\vec{x}$ do domínio tais que*

$$f(\vec{x}) = C.$$

[1] Em inglês, conjunto de nível se diz *level set*.

Quando $n = 2$, falamos de curvas de nível *e em* superfícies de nível *para $n = 3$. Em geral, o conjunto de nível terá uma dimensão inferior ao domínio da função e para $n > 3$, falamos em* hipersuperfícies de nível.

2.2 O vetor gradiente

Definição 2.2 *Seja*

$$f : \begin{cases} \Omega \subset \mathbb{R}^n & \longrightarrow & \mathbb{R} \\ (x_1, \cdots, x_n) & \mapsto & f(x_1, \cdots, x_n) \end{cases}$$

uma função de n variáveis a valores reais em um aberto Ω de $\mathbb{R}^n$. Considerando a base canôninica $(e_i)_{1 \leq i \leq n}$ de $\mathbb{R}^n$, dizemos que f admite uma derivada parcial na direção e_i no ponto $\vec{x}$ *se e somente se a função $t \mapsto f(\vec{x} + t\vec{e_i})$ admite derivada no ponto 0. Notamos então*

$$\frac{\partial f}{\partial x_i}(\vec{x}) = \frac{df}{dt}(\vec{x} + t\vec{e_i}) \mid_{t=0} . \tag{2.1}$$

Define-se também o vetor gradiente$\vec{\nabla} f$ *de f no ponto p pelo vetor de $\mathbb{R}^n$ cujas coordenadas são compostas pelas derivadas parciais de f:*

$$\vec{\nabla} f = \left(\frac{\partial f}{\partial x_1}, \cdots, \frac{\partial f}{\partial x_n} \right) \tag{2.2}$$

Supondo que $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ seja uma função regular, digamos de classe $C^1(\mathbb{R}^n; \mathbb{R})$ que define implicitamente uma família de superfícies segundo curvas de nível delimitadas pelos conjuntos

$$\Gamma_\kappa = \{\vec{x} \in \mathbb{R}^n \mid f(\vec{x}) = \kappa\}. \tag{2.3}$$

Fixamos agora um parâmetro κ para definir uma superfície. Considerando uma curva $\gamma : [0,1] \to \Gamma_\kappa$ sobre a superfície Γ_κ. Como $\gamma(t)$ pertence à superfície para todo

t pertencente ao intervalo [0,1], temos *a fortiori* que $f(\gamma(t)) = \kappa$, isto é,

$$f \circ \gamma \mid_{[0,1]} \equiv \kappa \tag{2.4}$$

Logo, efetuando a derivada com relação à variável t e levando em consideração a regra da cadeia, obtemos:

$$\vec{\nabla} f(\gamma(t)) \cdot \vec{\gamma}'(t) = 0. \tag{2.5}$$

Ora, cabe notar que $\gamma(t)$ é um ponto p pertencente à superfície Γ_κ e que o vetor $\gamma'(t)$ engendrado pela derivada da curva $\gamma(t)$ é um vetor pertencente ao plano tangente de Γ_κ. Mais geralmente, como qualquer ponto, assim como qualquer vetor tangente (por intermédio de derivação) podem ser obtidos por composição adequada de curvas γ sobre a superfície, a equação supracitada pode ser reescrita na forma: para qualquer ponto $p \in \Gamma_\kappa$ e para qualquer vetor $\vec{\tau}$ pertencente ao plano tangente $\Pi_p(\Gamma_\kappa)$ da superfície Γ_κ em p, temos:

$$\vec{\nabla} f(p) \cdot \vec{\tau} = 0. \tag{2.6}$$

Em outras palavras, provamos

Teorema 2.1 *Seguindo a notação acima,* $\vec{\nabla} f(p)$ *é o vetor normal ao plano* $\Pi_p(\Gamma_\kappa)$.

2.3 Multiplicadores de Lagrange

Seja $f(x,y)$ uma função continuamente diferenciável que queremos maximizar respeitando a restrição:

$$g(x,y) = C,$$

para algum escalar C.

O método dos multiplicadores de Lagrange transforma o problema de minimização em um problema de auto-valores por buscar uma solução da forma:

$$\begin{cases} \vec{\nabla} f & = & \lambda \vec{\nabla} g & \text{, para algum } \lambda \\ \\ g(x,y) & = & C \end{cases}$$

2.4 Malha Euleriana e Malha Lagrangiana

Definição 2.3 *No cenário de escoamentos multifásicos ocorrendo em um domínio* $\Omega \subset \mathbb{R}^3$*, seja* $\mathcal{S} : \mathcal{R}^+ \to \Omega$ *a função que retorna para cada tempo* $t \in \mathbb{R}^+$ *a região* $\mathcal{S}(t)$ *que delimita a fronteira entre as diferentes fases do escoamento. Uma tal região é chamada* Superfície Livre.

Definição 2.4 *Seja* $\Omega = [a_1,b_1] \times \cdots \times [a_n,b_n] \subset \mathbb{R}^n$ *um retângulo munido de uma malha retangular, formada pela união (disjunta) de elementos da forma:*

$$[a_1 + (i_1)\Delta x_1, a_1 + (i_1 + 1)\Delta x_1] \times \cdots \times [a_n + (i_n)\Delta x_n, a_n + (i_n + 1)\Delta x_n]. \qquad (2.7)$$

Tais elementos são chamados ***células*** *da malha.*

Definição 2.5 *Denomina-se* Malha Euleriana *uma malha cartesiana estruturada, o conjunto de todas as células inscritas no domínio computacional, conforme descritas na definição acima. São subdivisões fixas ao longo de uma simulação e particionam todo o domínio.*

Definição 2.6 *Quando a região da superfície livre é tratada por pontos com conectividade, sua malha associada é chamada* Malha Lagrangiana.

No cenário de escoamento multifásico as propriedades materiais de cada fluido sobre todo o domínio são representadas por uma malha euleriana e são utilizadas para solução das equações de Navier-Stokes. A superfície livre, por sua vez, necessita de um refinamento maior e será descrito por uma malha lagrangiana, que é atualizada

a cada passo de tempo Δt, consoante à configuração atual da superfície. Uma representação bidimensional de uma superfície livre de malha lagrangiana imersa em uma malha euleriana pode ser vista na figura 2.1.

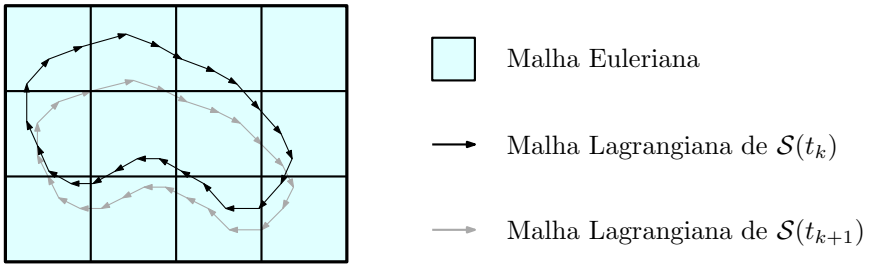


Figura 2.1: Após o passo de tempo, a malha lagrangiana da superfície $\mathcal{S}(t_k)$ é atualizada segundo sua posição após a evolução no tempo de $\mathcal{S}(t_{k+1})$. A malha euleriana permanece imóvel.

2.5 Projeções

Trataremos aqui as projeções respectivas à representação implícita da interface e o conjunto subjacente engendrado, que por definição comporta os pontos projetados neles mesmos. Uma formulação mais especificamente orientada às representações implícitas e locais da superfície será dada adiante.

A projeção é definida localmente, aproximando-se um conjunto de uma vizinhança de pontos por alguma classe especial de lugares geométricos $\mathcal{G}$, por exemplo por planos ou esferas. A escolha de $\mathcal{G}$ dependerá do problema a ser resolvido e faz parte do conjunto de características que compoem cada método. Tal escolha afetará igualmente o custo computacional do resultado final, assim como sua precisão e acurácia. Imersa em um contexto de aproximação local, é natural imaginarmos que tal subespaço de possibilidades geométricas tem maior funcionalidade quando seus elementos podem ser definidos por uma cadeia finita de parâmetros.

A definição da superfície é dada seja pela intersecção da malha lagrangiana com a interface implicitamente definida, seja da projeção de um ponto na interface, uma vez

que o ponto já se encontra em sua vizinhança.

Em um âmbito geral, uma função F definida em um espaço vetorial $\mathcal{U}$ (e.g. $\mathcal{U} = C[a,b]$) com valores em um espaço vetorial $\mathcal{V}$, ambos espaços associados ao mesmo corpo $\mathbb{K}$, é dita linear se for aditiva e homogênea. Isto é:

$$\begin{aligned} &F : \mathcal{U} \longrightarrow \mathcal{V} \\ &F(u_1 + u_2) = F(u_1) + F(u_2) \qquad \forall u_1, u_2 \in \mathcal{U}, \\ &F(\lambda u) = \lambda F(u) \qquad\qquad\qquad \forall u \in \mathcal{U}, \forall \lambda \in \mathbb{K} \end{aligned} \tag{2.8}$$

Tal função é dita formar uma **transformação linear** de $\mathcal{U}$ em $\mathcal{V}$ e o espaço de todas transformações desta forma é notado $\mathcal{L}(\mathcal{U},\mathcal{V})$. O operador projeção é um exemplo de transformação linear, tomando $\mathcal{U} = \mathcal{V}$ e notando que a imagem da aplicação age como subespaços de $\mathcal{U}$.

Um operador projetor pode ser definido como qualquer transformação linear satisfazendo

$$\begin{cases} P : \mathcal{U} \longrightarrow \mathcal{U}, \\ \quad P^2 = P. \end{cases} \tag{2.9}$$

Tal aplicação determina uma decomposição do espaço de origem $\mathcal{U}$ como soma direta de dois subespaços: a imagem $\Im(P)$ de P e o núcleo $\ker P$ de P. Para se apreender completamente um operador em uma aplicação específica é útil observar os dois subespaços relacionados $\Im(P)$ e $\ker P$, além das relações entre si e com o espaço total $\mathcal{U}$:

$$\begin{aligned} \mathcal{U} &= \Im(P) \oplus \ker P \\ \Im(P) &= \{y \in \mathcal{U} \mid \exists x \in \mathcal{U} \text{ t.q. } P \cdot x = y\} \\ \ker P &= \{u \in \mathcal{U} \mid P \cdot u = 0\} \end{aligned} \tag{2.10}$$

Para espaços pré-hilbertianos, isto é, espaços vetoriais munidos de um produto interno, a projeção adere uma forma mais explícita. Em particular, se consideramos

um plano definido por

$$\Pi = \{\vec{x} \in \mathbb{R}^n \mid \vec{n} \cdot \vec{x} = \kappa\}$$

em que $\|n\| = 1$, a projeção de um ponto $\vec{u}$ no plano Π é obtido caminhando na direção da normal $\vec{n}$ até o ponto de incidência no plano:

$$\begin{aligned} \vec{p} + \lambda\vec{n} \in \Pi &\implies <\vec{n},\vec{u} + \lambda\vec{n}> = \kappa \\ &\implies \lambda = \kappa - <\vec{n},\vec{u}> \end{aligned}$$

2.6 Mínimos Quadrados

Sejam $\mathcal{P} = \{(x_k\,, y_k\,) \;:\; k \in I\}$ uma nuvem de pontos, $\mathcal{F} = \{f_\ell\}_{\ell=0}^{d}$ uma família de funções e $\underline{\alpha} = (\alpha_0\,, \ldots\,, \alpha_d\,)$ um vetor de parâmetros. Escrevendo

$$F_{\underline{\alpha}}(\cdot) = \sum_{\ell=0}^{d} \alpha_\ell f_\ell(\cdot) \tag{2.11}$$

e definindo

$$\mathcal{J}(\underline{\alpha}) = \sum_{k\in I} (y_k - F_{\underline{\alpha}}(x_k))^2 \tag{2.12}$$

podemos enunciar o problema de mínimos quadrados como a busca por um parâmetro $\underline{\alpha}$ que satisfaz

$$\underline{\alpha} = \arg\, \min_{\underline{\beta}} \mathcal{J}(\underline{\beta}) \tag{2.13}$$

e procurando pelos pontos críticos, queremos $\underline{\beta}$ tal que

$$\frac{\partial}{\partial \beta_i}\, \mathcal{J}(\underline{\beta}) = 0 \tag{2.14}$$

para todo $0 \leq i \leq d$.

Calculando o membro da esquerda da equação acima

$$\begin{aligned} \sum_k \frac{\partial}{\partial \beta_i} \left(y_k - F_{\underline{\beta}}(x_k)\right)^2 &= \sum_k (y_k - F_{\underline{\beta}}(x_k)) \frac{\partial}{\partial \beta_i} F_{\underline{\beta}}(x_k) \\ &= \sum_k (y_k - F_{\underline{\beta}}(x_k))\, f_i(x_k) \end{aligned} \tag{2.15}$$

e igualando-o a 0 (zero) conforme (2.14), obtemos

$$\sum_{0 \le j \le d} \beta_j \sum_{k \in I} f_j(x_k)\, f_i(x_k) = \sum_{k \in I} y_k\, f_i(x_k) \tag{2.16}$$

para todo $i = 0, \ldots, d$. Isto remete ao sistema linear:

$$\mathbf{M} \cdot \mathbf{x} = \mathbf{v} \tag{2.17}$$

em que

$$\mathbf{M} = \left[\sum_k f_j(x_k)\, f_i(x_k) \right]_{ij} \tag{2.18}$$

é a matriz do sistema linear e

$$\mathbf{v} = \left[\sum_k y_k\, f_i(x_k) \right]_i \tag{2.19}$$

é o vetor coluna. A solução $\mathbf{x} = [\beta_0, \cdots, \beta_d]^T$ é o vetor de parâmetros procurado.

2.6.1 Polinômios

Tomando a família de funções $\mathcal{F} = \{f_\ell(x) = x^\ell\}$.

A matriz do sistema linear deriva imediatamente da equação (2.18):

$$\mathbf{M} = \left[\sum_k x_k^{i+j} \right]_{ij} \tag{2.20}$$

assim como o vetor coluna em (2.19)

$$\mathbf{v} \;=\; \left[\sum_k y_k\, x_k^i\right]_i \tag{2.21}$$

2.6.2 Esfera

Tomando a família de funções $\mathcal{F} = \{1,x,y,x^2+y^2\}$, no interesse de aproximar um conjunto de pontos $\{x_k,y_k\}_k$ pela esfera dada implicitamente por

$$ax + by + c = x^2 + y^2 \Rightarrow ax + by + c - (x^2+y^2) = 0, \tag{2.22}$$

consideremos para $\underline{\alpha} = (a,b,c)$ a função $f_{\underline{\alpha}}(x,y) = ax + by + c - (x^2+y^2)$.

A função a ser minimizada difere de (2.12) pois em vez de aproximar $f(x_k) \approx y_k$, queremos $f(x_k,y_k) \approx 0$, logo:

$$\mathcal{J}(\underline{\alpha}) = \sum_{k\in I} (f_{\underline{\alpha}}(x_k,y_k))^2 \tag{2.23}$$

Nota-se que (2.15) fica:

$$\begin{aligned}\sum_k f_{\underline{\alpha}}(x_k,y_k)\, f_i(x_k,y_k) \;&=\; \sum_k (ax_k + by_k + c - x_k^2 - y_k^2)\, f_i(x_k,y_k) \\ &=\; \sum_k \Big((ax_k + by_k + c)\, f_i(x_k,y_k) \\ &\qquad - (x_k^2 + y_k^2)\, f_i(x_k,y_k) \Big)\end{aligned} \tag{2.24}$$

e igualando a última expressão à 0, obtemos o sistema linear de matriz:

$$\mathbf{M} \;=\; \left[\, M_i^T \cdot M_j \,\right]_{ij} \tag{2.25}$$

e de vetor coluna

$$\mathbf{v} \;=\; \left[\, M_i \cdot M_4 \right]_i \tag{2.26}$$

em que:

$$\begin{aligned} M_1 &= [\,x_k\,]_{k\in I}\,, & M_2 &= [\,y_k\,]_{k\in I}\,, \\ M_3 &= [\,1\,]_{k\in I}\,, & M_4 &= \left[\,x_k^2+y_k^2\,\right]_{k\in I} \end{aligned} \tag{2.27}$$

e a solução $\mathbf{x} = [a,b,c]^T$ do sistema linear (2.17) é o parâmetro procurado.

Tomemos agora a equação que define implicitamente uma esfera

$$(x-c_x)^2 + (y-c_y)^2 = r^2 \tag{2.28}$$

tendo assim como centro $\vec{c} = [c_x,c_y]$ e raio $r > 0$. Expandindo

$$x^2 - 2x\,c_x + c_x^2 + y^2 - 2y\,c_y + c_y^2 = r^2 \tag{2.29}$$

e rearranjando os termos, obtemos:

$$\left(2\,c_x\right)x + \left(2\,c_y\,\right)y + \left(r^2 - (c_x^2+c_y^2)\right) - \left(x^2+y^2\right) = 0 \tag{2.30}$$

o que mostra que a aproximação acima efetuada é bem o ajuste por esferas. Ademais, determinamos a correspondência:

$$\begin{bmatrix} c_x \\ c_y \end{bmatrix} = \frac{1}{2}\begin{bmatrix} a \\ b \end{bmatrix} \qquad r = \sqrt{c + \frac{a^2+b^2}{4}} \tag{2.31}$$

2.6.3 Polinômios Trigonométricos

Tomando a família de funções $\mathcal{F} = \{f_\ell(x) = \sin(\alpha_\ell x)\} \bigcup \{f_\ell(x) = \cos(\beta_\ell x)\}$.

Alterando um pouco a notação de tal forma que decompomos o parâmetro

$$\underline{\alpha} = [a,b]^T \tag{2.32}$$

de tal maneira que

$$\begin{cases} \underline{a} = (a_0\,,\ldots,a_n) \\ \\ \underline{b} = (b_0\,,\ldots,b_n) \end{cases} .$$

a função aproximada assume a forma

$$F(\underline{a},\underline{b},x) = F_{\underline{a}\,\underline{b}}(x) = \sum a_k\,\sin\alpha_k\,x + b_k\,\cos\beta_k\,x$$

Quero o parâmetro que satisfaça

$$\begin{bmatrix} \underline{a} \\ \\ \underline{b} \end{bmatrix} = \arg\min_{(\underline{a},\underline{b})} \sum_k (y_k - F_{\underline{a}\,\underline{b}}(x_k))^2$$

ou seja, que minimize a função

$$\mathcal{J}(\underline{a},\underline{b}) = \sum_k (y_k - F_{\underline{a}\,\underline{b}}(x_k))^2$$

Obtenho o sistema linear decomposto em blocos abaixo:

$$\begin{bmatrix} \boldsymbol{M}_{\,1\,1} & \boldsymbol{M}_{\,1\,2} \\ \boldsymbol{M}_{\,2\,1} & \boldsymbol{M}_{\,2\,2} \end{bmatrix} \begin{bmatrix} \underline{a} \\ \underline{b} \end{bmatrix} = \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \end{bmatrix} \tag{2.33}$$

em que

$$
\begin{aligned}
\boldsymbol{M}_{\,1\,1} &= \left[\sum_j \sin\alpha_k\, x_j\ \sin\alpha_i\, x_j\right]_{i\,j} \\
\boldsymbol{M}_{\,1\,2} &= \left[\sum_j \sin\alpha_k\, x_j\ \cos\alpha_i\, x_j\right]_{i\,j} \\
\boldsymbol{M}_{\,2\,1} &= \left[\sum_j \cos\alpha_k\, x_j\ \sin\alpha_i\, x_j\right]_{i\,j} \\
\boldsymbol{M}_{\,2\,2} &= \left[\sum_j \cos\alpha_k\, x_j\ \cos\alpha_i\, x_j\right]_{i\,j} \\
v_1 &= \left[\sum_k y_k\, x_j\ \sin\alpha_i\, x_j\right]_i \\
v_2 &= \left[\sum_k y_k\, x_j\ \cos\alpha_i\, x_j\right]_i
\end{aligned}
$$

2.7 Equações de Navier-Stokes

Neste capítulo introduzimos as equações governantes do movimento de um fluido incompressível e viscoso. São chamadas equações de Navier-Stokes, em referência à Claude-Louis Navier e George Gabriel Stokes. Para uma orientação da dedução cabal destas equações o que se é aqui apresentado não basta, no entanto para o propósito da dissertação o conteúdo aqui exposto é largamente suficiente, a saber, desenvolver a noção de como estas equações aparecem no estudo da natureza e quais princípios dão margem para sua procedência.

Dentre os princípios considerados contamos com a *Segunda Lei de Newton* aplicada ao escoamento de fluidos. Tal princípio afirma que o produto da massa e aceleração é iigual à soma das forças externas agindo sobre o corpo. Aplicado ao movimento de fluidos estamos lidando com duas forças: as que agem na fronteira (atrito e pressão) e as que agem sobre a massa do corpo(forças gravitacionais). Outro princípio considerado

é que a tensão no fluido é oriunda de termos viscosos conjunta à pressão.

Em complementação à visão de movimento estritamente ligado à partículas e dinâmica de corpos rígidos (como na Mecânica Clássica em seu estádio inicial), a visão da mecânica celeste de astros, a dualidade ondular/particular adjacente à mecânica quântica e propriedades de grandes números associados à mecânica estatística temos a idéia da mecânica do meio contínuo. Conformemente, a formulação matemática da mecânica de fluidos aqui apresentada repousa sobre o conceito de *continuidade do meio*. Ou seja, a despeito da natureza atômica da matéria, no sentido em que as moléculas que a compoem não preenchem plenamente todo o espaço que ocupam (em outras palavra, há "ausência de matéria" entre as moléculas), matematicamente consideraremos o meio como um *continuum*. Assim, cada elemento infinitesimal $d\Omega$ de um meio Ω é considerado um ponto material e estabelece-se desta maneira uma bijeção entre o meio e algum subconjunto $D \subset \mathbb{R}^3$.

2.7.1 Equação da Continuidade

A primeira equação considerada será derivado de um princípio fundamental da mecânica clássica conhecido como a lei de Lomonosov-Lavoisier, ou princípio de conservação de massa.

Há conservação de massa em todo sistema material fechado no decorrer das transformações à que se submete, quaisquer que sejam estas transformações.

A continuidade do meio implica que a massa do espaço tridimensional de algum domínio $D \subset \mathbb{R}^3$ é resultado de uma integral de alguma função densidade

$$\begin{array}{rccc} \rho: & [0,+\infty)\times D & \longrightarrow & \mathbb{R}^+ \\ & (t,\vec{x}) & \mapsto & \rho(t,\vec{x}) \end{array}$$

que por sua vez representaria qual é o elemento de massa dm do meio em um determinado instante t no ponto $\vec{x}$, a saber $\rho(t,\vec{x})$. A massa de uma região Ω é tão logo obtida

trivialmente pela integração da função densidade $\rho(\cdot,\cdot)$ nesta região do espaço:

$$\mathcal{M} = m(\Omega,t) = \int_{\Omega} \rho(t,\vec{x})dV. \tag{2.34}$$

O princípio acima afirma que a variação de $m(\cdot,\cdot)$ no tempo é nula[2], assim:

$$\begin{aligned} \mathcal{M}' = \frac{d}{dt}m(\Omega,t) &= \int_{\Omega}\left(\frac{\partial\rho}{\partial t} + \nabla\cdot(\rho\vec{u})\right)dV = 0 \\ &= \int_{\Omega}\left(\frac{D}{Dt}\rho\right)dV = 0 \end{aligned} \tag{2.35}$$

para qualquer região $\Omega \subset D$, onde aproveitamos para introduzir a notação do operador $\frac{D}{Dt}(\cdot) = \frac{\partial}{\partial t}(\cdot) + \nabla(\cdot)$ chamado de derivada *Lagrangiana* ou *substantiva*. Podemos suspender a integral ao considerar que a validade da fórmula integral ser legítima para toda região de volume é semelhante à igualdade em todo ponto:

$$\boxed{\frac{D}{Dt}\rho = 0.} \tag{2.36}$$

A equação acima constitui a primeira das equações de Navier-Stokes e é conhecida por *equação da continuidade*. O primeiro termo representa a taxa de aumento da densidade no volume de controle e a segunda representa a taxa de fluxo de massa passando pela superfície de controle (a fronteira do volume de controle) por unidade de volume.

Quando o interesse é modelisar fluxos incompressíveis, obtemos imediatamente da definição

$$\rho \equiv \text{cte.}$$

e a equação é simplificada de maneira que se torna uma equação de continuidade de

[2]Mais precisamente, a região de volume homóloga Ω_{hom} da região Ω apresenta mesma massa.

volume:

$$\nabla \cdot \vec{u} = 0. \tag{2.37}$$

A interpretação física de tal nulidade absoluta do divergente do vetor velocidade é que a taxa de dilatação local do volume é nula.

2.7.2 Conservação de Momento

A ***Segunda Lei de Newton*** afirma que a resultante das forças aplicadas à um corpo é igual à taxa de variação da quantidade de movimento. No contexto de um meio contínuo, considerando novamente um subconjunto Ω do domínio D:

$$\frac{D}{Dt}\int_{\Omega} \rho\vec{u}\, dV = \int_{\partial\Omega} f_{\partial\Omega}\, dS + \int_{\Omega} \rho f_{\Omega}\, dV \tag{2.38}$$

em que f_{Ω} é a força por unidade de massa que aqui será tomada como a força gravitacional g, $\partial\Omega$ é a superfície que contorna Ω, $f_{\partial\Omega}$ é a força agindo sobre esta borda, podendo ser escrita na forma do tensor $\underline{\underline{\sigma}}$ das tensões agindo sobre o vetor normal $\vec{n}$ da superfície:

$$f_{\partial\Omega} = \underline{\underline{\sigma}} \cdot \vec{n} \tag{2.39}$$

e aplicando o teorema de Gauss sobre a primeira integral do RHS de (2.38) e fazendo as mesmas considerações sobre integrais em subconjuntos arbitrários do domínio (para obtenção da equação 2.36 da continuidade):

$$\frac{\partial}{\partial t}(\rho\vec{u}) + \nabla \cdot (\rho\vec{u} \cdot \vec{u}) = \nabla \cdot \underline{\underline{\sigma}} + \rho g \tag{2.40}$$

e finalmente, após substituir a equação da continuidade e expandir o tensor

$$\sigma \;=\; -p\underline{\underline{\mathbb{1}}} \;+\; \tau \tag{2.41}$$

em uma componente de pressão (p, multiplicada pelo tensor $\underline{\underline{\mathbb{1}}}$ tal que $\underline{\underline{\mathbb{1}}}_{ij} = 1$ se e somente se $i = j$ e 0 senão) e outra de tensões viscosas (τ), obtemos finalmente a equação de conservação da quantidade de movimento

$$\boxed{\rho \frac{D}{Dt}(\vec{u}) = \nabla p + \nabla \cdot \tau + \rho g.} \tag{2.42}$$

2.7.3 Condições de Fronteira

Considerando uma superfície livre na fronteira entre dois fluidos do cenário de escoamentos multifásicos, envolvendo um fluido de baixa densidade. Nesta superfície, as condições de fronteira para pressão e velocidade são dadas por:

$$\begin{aligned} (\underline{\underline{\sigma}} \cdot \vec{n}) \cdot \vec{n} &= \sigma\kappa\lambda \\ (\underline{\underline{\sigma}} \cdot \vec{n}) \cdot \vec{l} &= 0 \\ (\underline{\underline{\sigma}} \cdot \vec{n}) \cdot \vec{m} &= 0 \end{aligned} \tag{2.43}$$

em que σ é a tensão superficial, κ é a curvatura e λ é uma constante de relaxação. Os vetores $\vec{n}$, $\vec{l}$ e $\vec{m}$ são vetores unitários nas direções normal e tangencial á superfície, respectivamente.

Capítulo 3

FreeFlow

Algoritmos desenvolvidos para resolver equações diferenciais numericamente frequentemente geram grandes quantidades de dados. No caso da integração numérica das equações de Navier-Stokes simulando um escoamento multifásico, um programa gerará dados que se traduzem como informações do problema estudado, mais especificamente, sobre a geometria da interface, a pressão nos pontos do domínio computacional, a velocidade, a temperatura, etc. Faz-se conveniente assim haver algum programa associado que represente graficamente estas informações, um *visualizador*.

O *Freeflow-2D* é um sistema integrado que se utiliza para a modelagem, simulação e visualização de escoamentos bidimensionais de fluidos Newtonianos incompressíveis com superfícies livres. O objetivo deste capítulo é introduzir este ambiente de simulação que abriga o método GENSMAC para resolver escoamentos com superfícies livres, pois é no seio deste ambiente que se desenvolveu a representação da superfície utilizando o método da Partição da Unidade Implícita (PUI). As discretizações das equações diferenciais utilizadas pelo método GENSMAC não serão explicitadas neste documento, apenas haverá indicações de leituras sugeridas que cobrem exaustivamente este assunto.

A separação em três módulos distintos do *Freeflow2D* permite a extensão de cada um deles, que se comunicam um com o outro por intermédio de arquivos de entrada e

saída. Desta maneira, o ambiente foi estruturado para permitir futuras extensões em cada módulo separadamente. Os módulos são:

- *Modflow2D*: é o módulo que modela as características geométricas do ambiente do problema considerado, inicializa as variáveis e atribui valores às constantes adimensionais do escoamento considerado.

- *Simflow2D*: esse módulo implementa a discretização das equações governantes para um fluido Newtoniano incompressível utilizando a técnica *GENSMAC2D*. É a parte central do ambiente *Freeflow2D*.

- *Visflow2D* disponibiliza a visualização dos dados gerados pelo módulo *Simflow2D* utilizando ferramentas de computação gráfica.

Finalmente, o método MAC em que foi inspirado seu sucessor GENSMAC tem recebido notável retomada de interesse na publicação científica, segundo [3]. Os sistemas *Freeflow2D*, *FreeflowAxi* e *Freeflow3D* foram objetos de estudo de um extenso número de trabalhos de pesquisa, artigos, teses e dissertações.

3.1 Simulador

Dentre os primeiros tratamentos computacionais das equações de Navier-Stokes encontramos o método MAC[14], acrônimo do inglês *Marker and Cell*, introduzido por Harlow & Welch, em que é utilizada a técnica da malha deslocada (*staggered mesh*, em inglês) e partículas marcadoras. Analogamente, Tomé & McKee desenvolveram um método de escoamentos bidimensionais com superfície livre e as partículas marcadoras fornecem a localização da interface no domínio, chamado GENSMAC [31].

Em uma série de extensões do trabalho, o método se desenvolveu para simular escoamentos incompressíveis tridimensionais de fluidos Newtonianos isotérmicos [55]. Este método, chamado GENSMAC3D, fora incorporado em um ambiente de simulação

a que se deu o nome de *Freeflow3d* [15]. O *FreeFlow* abriga um agregado de rotinas para representar escoamentos variados, entre os quais podemos citar como exemplos os respectivos ao movimento de containers [26], escoamento de fluidos não-isotérmicos, escoamentos multifásicos e inclusão de efeitos de tensão superficial [28] e escoamento de fluidos Newtonianos generalizados [29]. O nome dado ao simulador que resolve as equações de Navier-Stokes do domínio bidimensional (o chamado *solver*, em inglês) é *Simflow2D*.

3.1.1 Malha Deslocada

Dependendo da escolha de modelagem numérica das equações de Navier-Stokes podemos nos deparar com aproximações inconsistentes com a realidade física do problema. A citar, acontece em certas ocasiões um campo oscilatório de pressão na solução numérica, ainda que se utilise discretizações de alta ordem, em virtude do aparecimento de grupos de equações desacopladas entre si.

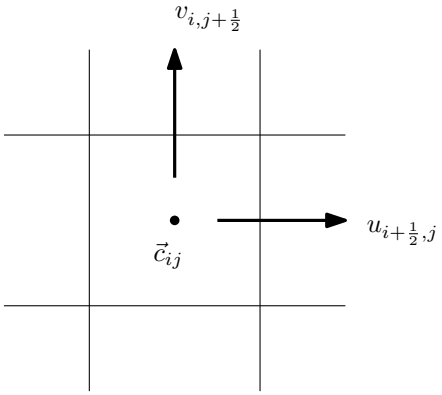


Figura 3.1: Ilustração de uma malha deslocada: no centro encontramos a posição $\vec{c}_{ij}$ e na malha deslocada os valores do vetor velocidade $\vec{u}(\vec{x}) = (u(\vec{x}), v(\vec{x}))$, cada coordenada deslocada na sua direção respectiva.

Uma possível abordagem que contorna esta complicação é a introdução de índices deslocados $f_{i+\frac{1}{2}}$ para uma função $f(x)$ no conjunto de equações de tal forma que o desacoplamento não acontece. Para tanto, estes valores (e.g. o campo de velocidades)

considerados na solução numérica da equação estarão deslocados, e a malha respectiva que armazena estes valores recebe igualmente o nome de *malha deslocada*. Na figura 3.1, observa-se para cada célula $\mathcal{C}_{ij}$, os valores que se armazena do campo de velocidades

$$\vec{u}(\vec{x}) = \Big(u(\vec{x}), v(\vec{x})\Big)$$

são: $U(i,j) = u_{i+\frac{1}{2},j} = u(x + (i + \frac{1}{2})\Delta x, y + j\Delta y)$ para a velocidade na direção $\hat{x}$ e $V(i,j) = v_{i,j+\frac{1}{2}} = v(x + i\Delta x, y + (j + \frac{1}{2})\Delta y)$ para a velocidade na direção $\hat{y}$, em que temos duas matrizes U e V armazenando os valores na malha deslocada da velocidade nas direções $\hat{x}$ e $\hat{y}$, respectivamente. Pode-se recuperar a velocidade em um ponto (x,y) qualquer mediante interpolação dos pontos mais próximos da malha deslocada.

3.1.2 O Método GENSMAC

Porquanto a solução numérica das equações de Navier-Stokes, no que tange suas discretizações, independem do tipo de representação da superfície livre – e vice-versa – o algoritmo do método GENSMAC será exposto apenas resumidamente neste documento, abordando a idéia estrutural do programa. Uma visão mais detalhada do código pode ser encontrada em [31], [23] e [25]. Nesta seção será utilizado o seguinte padrão: todo campo contendo um til ($\widetilde{}$) será um campo intermediário do método, isto é, será utilizado para o cálculo do campo final nos passos seguintes do algoritmo. Dito isto, os passos do método GENSMAC em cada instante $t = t_n$ são:

a) Um campo de pressão $\widetilde{p}$ que satisfazendo as condições de contorno da superfície livre é calculado.

b) O campo de velocidades intermediário $\widetilde{u}$ é calculado mediante uma versão adimensional da equação de conservação de movimento :

$$\frac{\partial}{\partial t}\widetilde{u} = -(u \cdot \nabla)u - \nabla\widetilde{p} + \frac{1}{Re}\nabla^2 u + \frac{1}{Fr^2}g. \tag{3.1}$$

c) Resolvendo uma equação de Poisson

$$\nabla^2 \Psi = \nabla \cdot \widetilde{u} \tag{3.2}$$

com condições de fronteira (do tipo von Neumann ou do tipo Dirichlet), obtém-se um potencial Ψ.

d) O campo de velocidades final é calculado:

$$u(\vec{x},t) = \widetilde{u}(\vec{x},t) - \nabla\Psi. \tag{3.3}$$

e) O campo de pressão final é calculado:

$$p^{n+1} = \widetilde{p} + \frac{\Psi}{\delta t}. \tag{3.4}$$

f) Calcula-se a movimentação da interface utilizando o campo de velocidade atualizado por intermédio das EDOs:

$$\begin{cases} \frac{d}{dt}x &= u \\ \\ \frac{d}{dt}y &= v \end{cases}. \tag{3.5}$$

Após o término do ciclo, o campo de velocidades u obtido satisfará, para $t_{n+1} = t_n + \delta t$, a equação de conservação de massa:

$$\nabla \cdot u = 0. \tag{3.6}$$

O último passo (f) pode ser discretizado pelo método de Euler implícito ou explícito.

3.1.3 Eliminador de ondulações TSUR-2D

Em simulações de alto número de Reynolds (acima de 50), ondulações numéricas da interface no nível da subcélula são passíveis de acontecer, por exemplo, em situações em que localmente há diminuição da superfície. Estas ondulações se dão por conta da diferença do campo de velocidade variando de célula para célula.

A fim de eliminar tais ondulações foi desenvolvido um filtro chamado TSUR-2D, acrônimo do nome em inglês *Trapezoidal Sub-grid Undulation Removal*, removedor trapezoidal de ondulações na sub-malha. A formulação para problemas tridimensionais do chamado TSUR-3D pode ser encontrada em [1] e [28].

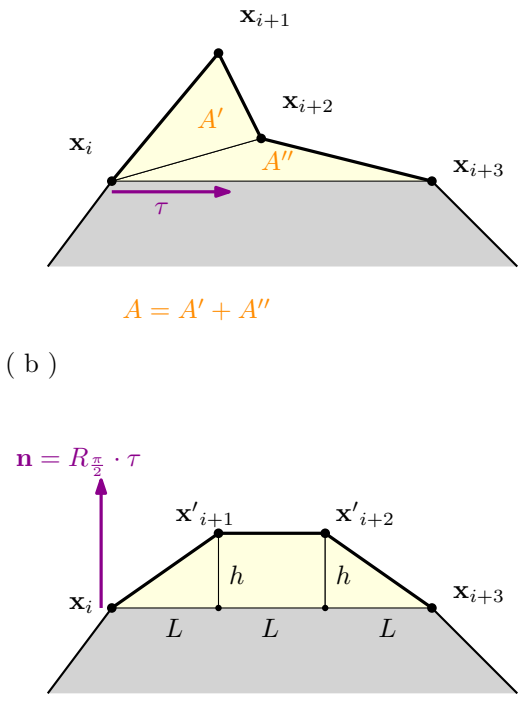


Figura 3.2: Ilustração da ação do eliminador de ondulações bidimensional TSUR-2D aplicado na aresta ($\mathbf{x}_{i+1}$,$\mathbf{x}_{i+2}$).

O eliminador de ondulações moverá, para cada aresta ($\vec{x}_{i+1}$,$\vec{x}_{i+2}$) da malha lagrangiana, os vértices $\vec{x}_{i+1}$ e $\vec{x}_{i+2}$ a ela associados de tal maneira a conservar o volume.

Na notação desta seção, $\vec{x}'_{i+1}$ e $\vec{x}'_{x+2}$ designam as posições para onde serão movidos os vértices $\vec{x}_{i+1}$ e $\vec{x}_{i+2}$, respectivamente, ao considerarmos o algoritmo TSUR-2D aplicado na aresta $(\vec{x}_{i+1},\vec{x}_{i+2})$. Utilizando então quatro pontos consecutivos da malha lagrangiana, representados na figura (3.2a): $\vec{x}_i,\vec{x}_{i+1},\vec{x}_{i+2}$ e $\vec{x}_{i+3}$. Primeiro calcula-se a área formada pelo quadrilátero $(\vec{x}_i,\vec{x}_{i+1},\vec{x}_{i+2},\vec{x}_{i+3})$ (conforme indica a figura, isto pode ser feito calculando separadamente as áreas A' e A'' dos triângulos $(\vec{x}_i,\vec{x}_{i+1},\vec{x}_{i+2})$ e $(\vec{x}_i,\vec{x}_{i+2},\vec{x}_{i+3})$, respectivamente).

Definindo

$$\vec{\tau} = (\tau_x,\tau_y)^t = \frac{\vec{x}_{i+3}-\vec{x}_i}{\|\vec{x}_{i+3}-\vec{x}_i\|}$$

$$\vec{n} = (\tau_y,-\tau_x)^t = R_{\pi/2}\cdot\vec{\tau}$$

em que $R_{\pi/2}$ é matriz de rotação de $\pi/2$, as novas posições $\vec{x}'_{i+1}$ e $\vec{x}'_{i+2}$ designadas são:

$$\begin{cases} \vec{x}'_{i+1} = \vec{x}_i + L\vec{\tau} + h\vec{n} \\ \\ \vec{x}'_{i+2} = \vec{x}_i + 2L\vec{\tau} + h\vec{n} \end{cases} \tag{3.7}$$

em que $L = \frac{\|\vec{x}_{i+3}-x_i\|}{3}$ e $h = \frac{A}{2L}$. A figura (3.2b) mostra o resultado final do eliminador de ondulações aplicado na aresta $(\vec{x}_{i+1},\vec{x}_{i+2})$ e a formulação matemática garante conservação da área do novo quadrilátero

$$(\vec{x}_i,\vec{x}'_{i+1},\vec{x}'_{i+2},\vec{x}_{i+3})$$

graças à escolha de h.

A figura (3.3) foi extraída de [1] e mostra uma comparação entre duas simulações, uma sem o TSUR-2D representada pela linha tracejada e outra com o TSUR-2D representada pela linha pontilhada. Trata-se de uma simulação em um domínio discretizado usando uma malha uniforme de 50×60 células. As ondulações da superfície tracejada

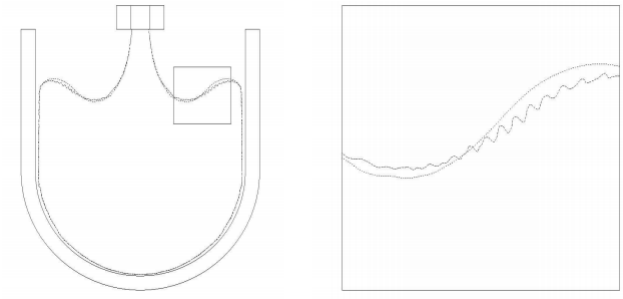


Figura 3.3: Ação do TSUR: na linha tracejada a simulação sem ajuste de ondulações e na linha pontilhada a simulação com eliminação das ondulações.

(sem TSUR) são erros numéricos que comprometem o cálculo da tensão superficial. As ondulações são nitidamente eliminadas sob a ação do TSUR-2D, rendendo os resultados mais confiáveis.

3.1.4 Rótulo das Células

A essência do método MAC são as partículas virtuais marcadoras e as células definidas em uma malha euleriana [3]. A maneira como as células são catalogadas no método GENSMAC depende da relação entre a célula, a superfície que delimita as diferentes fases e sua localização no domínio. Esta classificação das células serve o propósito de situar a superfície livre, os contornos do problema e as células onde há entrada ou saída de fluido.

Desta maneira, podemos classificar as células como do tipo:

- **FULL(F)** - quando a célula se encontra inteiramente contida no interior do fluido ;
- **EMPTY(E)** - quando toda a célula está no exterior do fluido ;
- **SURFACE(S)** - para células contendo partículas marcadoras e havendo em sua vizinhança ao menos uma célula do tipo E;

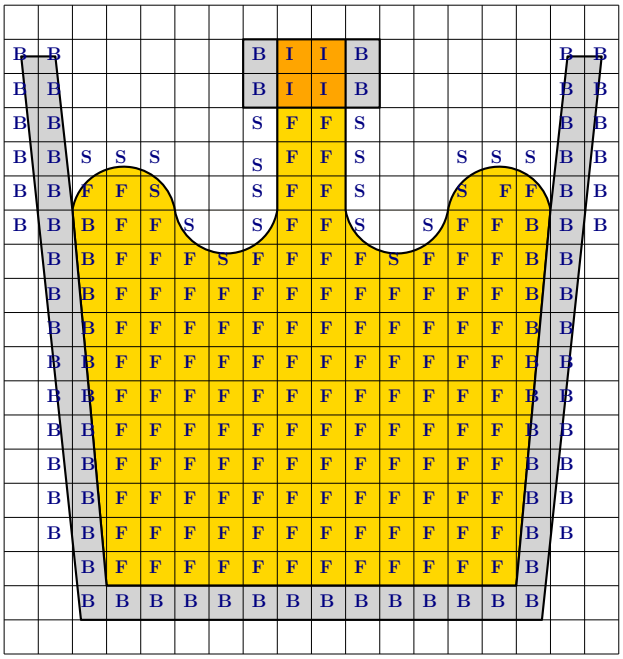


Figura 3.4: Rótulo das Células.

- **BOUNDARY(B)** - para células da extremidade, pertencentes ao domínio rígido da simulação ;
- **INFLOW(I)** - células em que há entrada de fluido ;
- **OUTFLOW(O)** - células em que há saída de fluido ;

A figura (3.4) ilustra uma configuração de células e seus rótulos em um instante de um escoamneto bidimensionais, em que as do tipo **EMPTY** são as células sem rótulo.

3.1.5 A Interface

Considerando um domínio bidimensional como uma simulação de um problema 2D ou como uma seção transversal de um domínio 3D, no cenário de escoamento de fluidos, a fronteira que separa regiões do domínio em que as propriedades materiais dos fluidos distam entre si – ou seja, a frente que separa dois fluidos diferentes – é chamada interface. Ademais, a terminologia usada nesta seção e ilustrada na figura (3.5) entende

por *slice* ou fatia todo o agregado de uma componente conexa de fluido e por *face* cada componente conexa projetada no domínio. Na figura, temos o exemplo de duas faces pertencentes à mesma fatia $\mathcal{S}_1$ (a saber $\mathcal{F}_1$ e $\mathcal{F}_2$) pois ambas pertencem à mesma componente conexa do fluido. As partículas marcadoras na fronteira reconstroem a frente do fluido via polinômios afins por partes.

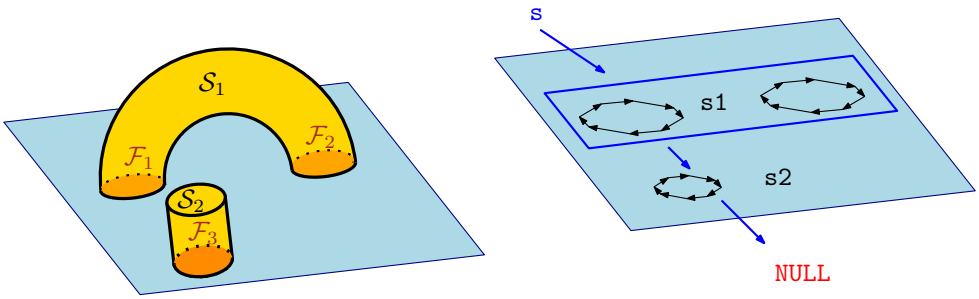


Figura 3.5: Visualização em um domínio bidimensional de três componentes conexas $\mathcal{F}_1$, $\mathcal{F}_2$ e $\mathcal{F}_3$, na hierarquia da lista de *slices* : a cabeça da lista `s` aponta para `s1`, que contem `f1` e `f2`, o próximo slice da lista é `s2`, que contém `f3`.

O *Freeflow2D* usa como estrutura de dados para a interface o *half-edge2D* ([2]). Esta se baseia em uma representação topológica centrada na aresta, divida em duas semi-arestas, que caminham em direções opostas. A partir da semi-aresta, recupera-se informações sobre os vértices, as arestas e as faces. A figura (3.6) mostra a hierarquia da estrutura de dados *half-edge2D* do *simflow2D* com a notação inspirada na sintaxe da linguagem C: dado um ponteiro de *slices* apontando para uma fatia `s`, o acesso aos campos `s->prv` e `s->nxt` retorna a fatia anterior e posterior da lista, respectivamente; ademais, a figura mostra a hierarquia dos demais objetos topológicos do *slice* e como se dá o acesso à lista de vértices, de faces e arestas.

Alternativamente à representação da malha lagrangiana por *half-edge*, poderia também haver a estrutura de dados *mate-face*. Estas têm malhas associadas à um perfil, isto é, informações armazenadas definindo a dimensão da malha, tipo de dados e operações efetuadas. As informações geométricas são armazenadas em vértices, como a posição - a partir de que se faz todas as operações geométricas - e um indicador para

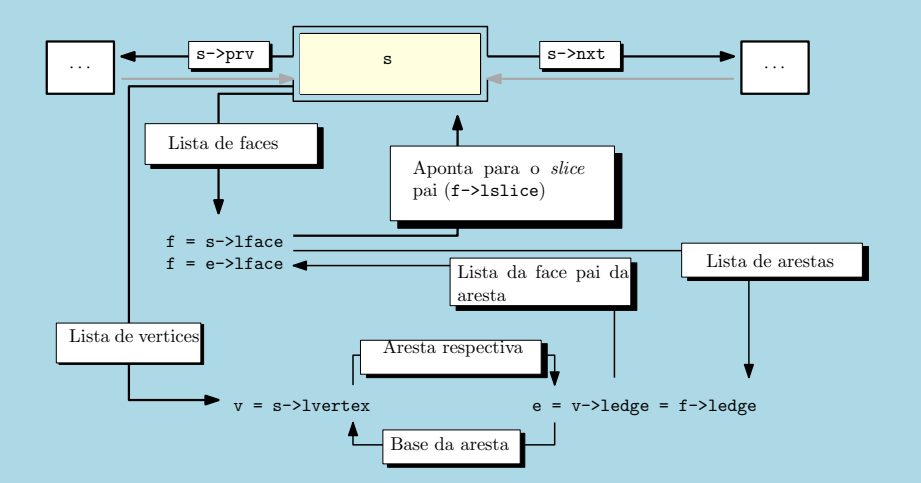


Figura 3.6: Hierarquia da estrutura de dados semi-aresta (*half-edge2D*) no *freeflow2D*.

a célula incidente. Conforme descrito em [33], pode-se representar as arestas e faces explícitamente ou implicitamente, possibilitando assim poupar memória. No futuro, o *Freeflow* talvez incorpore a estrutura de dados *half-edge* como representação alternativa da superfície livre.

A superfície representada pela malha lagrangiana terá participação nas equações de Navier-Stokes no cálculo das condições de fronteira. Para tanto, necessita-se uma estimativa dos vetores normais à superfície. A frente participa igualmente nos efeitos da tensão superficial para cujo cálculo é requerido o valor da curvatura, o que conduz naturalmente à próxima seção.

3.2 Cálculo de Normal e Curvatura

A aproximação inicial do vetor está intimamente ligada com a rotulação das células da malha euleriana. Dada uma célula $\mathcal{C}_{ij}$, o procedimento retornará uma estimativa do vetor $\vec{n}_{ij}$ respectivo ao vetor normal à superfície no centro $\vec{c}_{ij}$ da célula $\mathcal{C}_{ij}$. A figura (3.6) permite uma visualização do princípio: o vetor normal $\vec{n}_{ij}$ favorece a direção onde

haverá células do tipo `EMPTY`. Chamaremos este vetor de **normal de referência** e ele será obtido observando o rótulo das células $\mathcal{C}_{i-1,j}$,$\mathcal{C}_{i,j-1}$,$\mathcal{C}_{i+1,j}$ e $\mathcal{C}_{i,j+1}$. Temos então as 8 possibilidades seguintes:

$$\vec{n}_{ij} \in \left\{ \left(\pm 1, 0 \right), \left(0, \pm 1 \right), \left(\pm \frac{1}{\sqrt{2}}, \pm \frac{1}{\sqrt{2}} \right) \right\}. \tag{3.8}$$

Esta normal de referência é ilustrada com dois exemplos na figura 3.7: no caso (a), apenas uma célula vizinha é do tipo `EMPTY`, em cuja direção aponta o vetor de referência; no caso (b), havendo duas células do tipo `EMPTY` o algoritmo opta por fazer uma média das duas direções. Vale mencionar que o índice (i,j) em $\mathcal{C} = \mathcal{C}_{ij}$ foi omitido na figura para deixar a ilustração mais leve em notação.

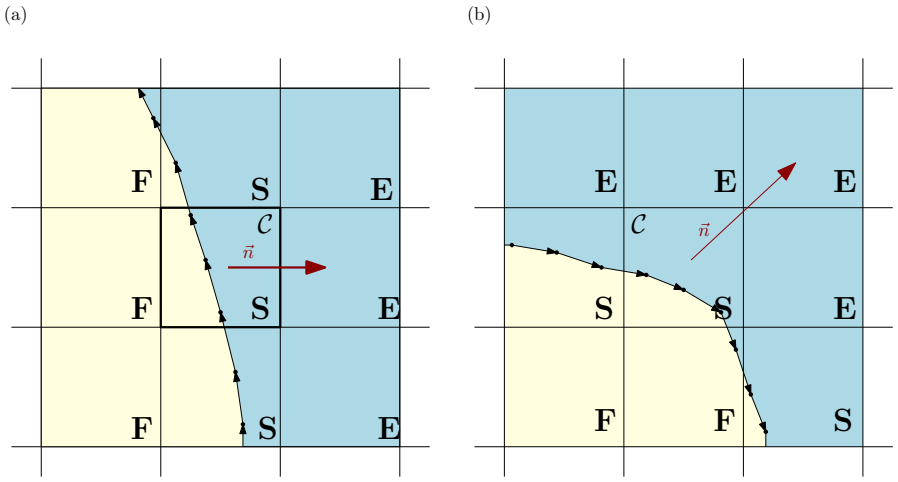


Figura 3.7: Visualização da primeira estimativa da normal: (a) somente uma célula do tipo `EMPTY` é contígua e em sua direção e sentido aponta o vetor normal $\vec{n}$; (b) duas células do tipo `EMPTY` são vizinhas, o vetor normal $\vec{n}$ aponta na direção média.

Na ausência de tensão superficial, a estimativa da normal de referência é suficiente (como vetor normal definitivo) e como se pode notar, seu cálculo é quase imediato. No entanto, havendo tensão superficial presente, faz-se necessário uma aproximação mais refinada da normal, assim como da curvatura.

Conforme mencionado anteriormente, o vetor normal e a curvatura da superfície

são necessários na resolução das equações de Navier-Stokes. Mais especificamente, elas são importantes no cálculo das condições de fronteira e na ação da tensão superficial.

Seja $\mathcal{P}$ o conjunto das partículas que definem a superfície. Para todo centro $\vec{c}_{ij}$ de toda célula $\mathcal{C}_{ij}$, calcula-se a normal e a curvatura conforme segue:

(1) Dado um critério de vizinhança $\epsilon_1 > 0$ e a bola $B_1 = B(\vec{c}_{ij},\epsilon_1)$, aproxima-se todos os pontos da intersecção $\mathcal{P} \cap B_1$ por um plano π, segundo o método dos mínimos quadrados;

(2) Seja $\vec{n}_\pi$ a normal do plano π calculado no item (1);

(3) Usando $\vec{n}_\pi$ como um dos eixos cartesianos, constroe-se um novo sistema de coordenadas ;

(4) Dado um novo critério de vizinhança $\epsilon_2 > 0$ e $B2 = B(\vec{c}_{ij},\epsilon_2)$, consideramos a intersecção $\mathcal{P} \cap B_2$ e calculamos os coeficientes a,b,c,d e e do parabolóide

$$p(\xi,\eta) = a\xi^2 + b\xi\eta + c\eta^2 + d\xi + e\eta + f \tag{3.9}$$

que aproxima os pontos da interseção, estando os elementos já descritos no novo sistema de coordenadas.

O **cálculo da normal** depende da direção do vetor normal $\vec{n}_\pi$ obtido no item (2). Sejam $\vec{e}_x$, $\vec{e}_y$ e $\vec{e}_z$ vetores da base canônica de $\mathbb{R}^3E$ e seja $\vec{n}_{ij}$ o vetor normal de referência em (3.8). Seja ainda

$$\vec{e} = \arg \max_{\vec{e'} \in \{e_x,e_y,e_z\}} | < \vec{n}_{ij},\vec{e'} > |.$$

Então, caso $\vec{e} = \vec{e}_x$, a equação do plano descrito no item (1) é escrito na forma $x = \alpha y + \beta z + \gamma$ (respectivamente $y = \alpha z + \beta x + \gamma$ caso $\vec{e} = \vec{e}_y$ e $z = \alpha x + \beta y + \gamma$

caso $\vec{e} = \vec{e}_z$). Então o vetor normal da célula $\mathcal{C}_{ij}$, isto é, o vetor normal da superfície no ponto c_{ij} vale

$$\mathtt{s}\ (<\vec{n'},\vec{n}_i j>)\frac{\vec{n'}}{\|n'\|_2} \tag{3.10}$$

em que $n' = (-1,\alpha,\beta)$ (resp., a primeira permutação cíclica caso $\vec{e} = \vec{e}_y$, a segunda permutação cíclica se $\vec{e} = \vec{e}_z$) e $\mathtt{s}(\cdot)$ é a função sinal.

A **curvatura**, por sua vez, é calculada mediante a fórmula

$$\kappa = -2\left(\frac{a}{(1+d^2)^{3/2}} + \frac{b}{(1+e^2)^{3/2}}\right) \tag{3.11}$$

em que os coeficientes a,b,d e e provém da aproximação descrita no item (4) acima.

3.3 Mudança de Topologia

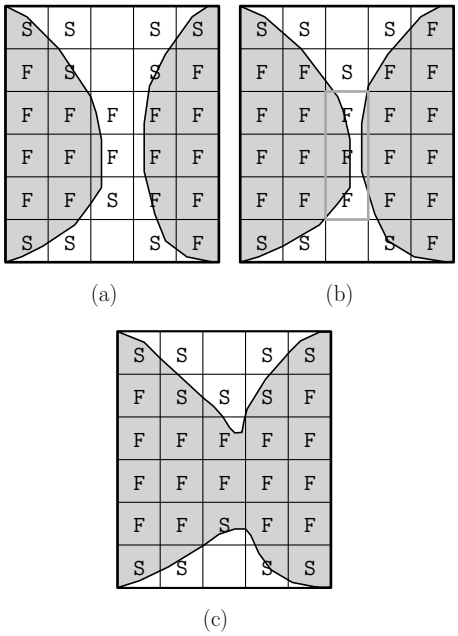


Figura 3.8: Junção de duas componentes conexas de fluido.

O procedimento de junção de duas superfícies livres acontece quando há proxi-

midade entre elas. Neste caso, quando as diferentes superfícies estão suficientemente próximas a ponto de terem células do tipo FULL em comum, acontece a união dos fluidos. O processo foi ilustrado na figura (3.8): na figura (a), dois segmentos conexos estão se aproximando; em (b), no passo de tempo seguinte, a área assinalada em cinza evidencia células do tipo FULL comuns aos dois segmentos distintos; finalmente, em (c), ainda no mesmo passo de tempo que na figura anterior, observamos a união dos dois fluidos.

A cisão de um fluido ocorre de maneira análoga. Após duas regiões não vizinhas da interface se aproximarem, de tal forma que suas células respectivas do tipo SURFACE estão postas lado a lado, acontece o rompimento da parede. O processo está ilustrado na figura (3.9)

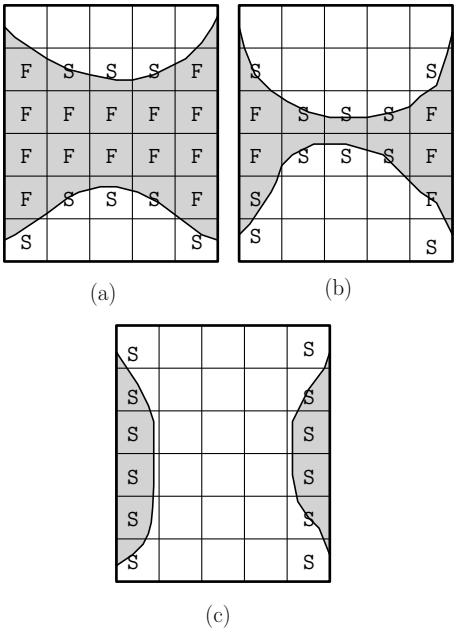


Figura 3.9: Divisão da parede do fluido.

Capítulo 4

Trabalhos prévios

Trataremos neste capítulo sobre trabalhos que inspiraram a concepção do projeto desenvolvido, dividindo o conteúdo em dois troncos principais: métodos lagrangeanos e métodos baseados em partículas. Sobre os métodos lagrangeanos discutiremos uma idéia de método de acompanhamento de fronteira [10] que elabora as equações de escoamento multifásico utilizando funções características para delimitar as diferentes fases do fluido. Estenderemos a discussão sobre funções características, dando um exemplo de como construí-las conforme tratado em [8]. É digno de nota que outras duas representações lagrangianas – as estruturas de dado *half-edge* e *mate face* – são tratadas na seção 3.1.5, no capítulo sobre o *Freeflow*. Será discutido também alguns métodos baseados em partículas, introduzindo o conceito de *level set*.

4.1 Métodos Lagrangeanos

Métodos com partículas marcadoras consiste em distribuir em uma interface uma quantidade representativa de pontos que evoluirão no tempo para representar a evolução da interface numericamente. Tal evolução das partículas é regida em função do campo de velocidades e falamos em *conectividade* quando há uma malha lagrangiana que liga

estes pontos usando uma lista (pontos conectados por elementos). Os pontos e os elementos (o objeto da fronteira) são armazenados em listas encadeadas que contém ponteiros para o objeto anterior e o próximo objeto da lista, o que faz com que a adição e remoção de objetos seja realizado de maneira simples.

Este método, dito lagrangeano, engendra uma simulação numérica acurada, no entanto possui limitações como:

- A dificuldade de simular os processos de quebra e fusão das interfaces. Não é evidente tratar as mudanças de topologia.

- Também é difícil manter a densidade das partículas consistente com o nível de discretização. Elas acumulam-se em algumas áreas enquanto minguam em outras. Estratégias de criação e eliminação de partículas são necessárias para lidar com o problema.

- A maioria dos métodos exclusivamente Lagrangianos matêm uma conectividade entre partículas necessárias para reconstruir a superfície livre a partir das partículas que se movem com ela. Tal conectividade define uma malha da interface em movimento. Se a malha se distorce demais a reconstrução se torna fisicamente incoerente, levando ao colapso da simulação.

O método de acompanhamento de fronteira com partículas marcadoras tem sido aplicado em soluções de uma grande variedade de problemas, tais como instabilidade de fluidos, fluidos reagindo quimicamente, fadiga de material e trocas de fases.

Em um esquema baseado em conectividade a vizinhança de pontos é levada em consideração segundo uma lista, que é mantida e necessária para cada ponto da interface. Existe grande interesse em tratar com cautela quando se trabalha com conectividade, pois a geometria da interface que dela depende pode sofrer deturpações que comprometeriam uma simulação.

Sem a presença de conectividade, algumas operações sobre a interface, assim como alguns fenômenos físicos (a citar, mudança de topologia), são reduzidas sensivelmente em termos de complicações. Inserção e eliminação de pontos são atividades que não envolvem grandes preocupações, por exemplo, pois pretere o rearranjo dos pontos da vizinhança que preservam a conectividade, o que contrariamente não seria o caso.

Por conseguinte, há interesse em representar a superfície sem uso de conectividade, como é feito em [8], um método de acompanhamento de frente sem conectividade. É possível reconstruir informações importantes mediante tais métodos, como normais e curvatura. A forma como esta idéia será desenvolvida nesta seção repousa na utilização de uma função característica $\mathbb{1}_\Omega$, que também é chamada de indicadora ou *Heaviside* dependendo do artigo, que vale

$$\mathbb{1}_\Omega(\vec{x}) = \begin{cases} 1, \text{ se } \vec{x} \in \Omega \\ 0, \text{ senão.} \end{cases} \tag{4.1}$$

Suponhamos, ademais, que a velocidade em cada ponto da superfície implícita seja dado por $\vec{u}(\vec{x})$, conhecida em cada ponto $\vec{x}$ da interface[1]. A partir deste campo de velocidades deseja-se mover todos os pontos na superfície. A maneira mais simples de fazê-lo é resolver a equação diferencial ordinária

$$\frac{d\vec{x}}{dt} = \vec{u}(\vec{x}) \tag{4.2}$$

para todo ponto $\vec{x}$ na fronteira (i.e. para todo $\vec{x}$ tal que $\phi(\vec{x}) = 0$). Uma vez que existem infinitos pontos na interface, isto significa discretizar a interface com um número finito de pontos.

Por exemplo, é possível usar segmentos no caso bidimensional, ou triângulos no caso

[1] É tipicamente o caso em métodos que seguem a linha do MAC de haver uma malha deslocada (cf. seção 3.1.1) a partir de que se obtém valores usados na interpolação do campo de velocidades para encontrar $\vec{u}$ na posição $\vec{x}$.

tridimensional, e mover os vértices destes segmentos/triângulos. Isto não é difícil de conseguir se a conectividade não mudar e os elementos da superfície não se distorcerem muito. Infelizmente, mesmo os campos mais triviais de velocidade pode causar uma enorme discrepância nos elementos de fronteira(segmentos ou triângulos), e a acurácia do método pode deteriorar rapidamente se não for feita uma modificação periódica da discretização a fim de amenizar as deformações emergentes, suavizando e regularizando inacurácias dos elementos de superfície.

4.1.1 Formulação das Equações de Escoamento Usando Funções Características

Dentre os variados interesses em abordar um problema de escoamento de fluidos segundo a técnica *Front-Tracking* é a destreza com que se pode tratar os casos de escoamento multifásico, em particular, no que tange a reformulação do conjunto de equações de tal maneira que é possível condensá-las em um conjunto compacto de equações uma vez que se dispõe de uma função como a característica $\mathbb{1}_\Omega$. Em outras palavras, não há conjuntos de equações distintos respectivos à cada fase do domínio físico e sim um único conjunto de equações para todo o domínio.

Em âmbito de escoamento bifásico, a função indicadora fornece os pontos de uma ou outra superfície das que compoem as duas fases pela imagem recíproca de um ponto. Mediante o auxílio de $\mathbb{1}_\Omega$ e de "funções" deltas de Dirac δ[2].

Mais precisamente, a discontinuidade das propriedades materiais implicam que as equações governantes são regidas por operadores diferenciais em sua forma fraca. A função característica, por ser constante em todo ponto do domínio alheio à interface, engendra tal interface quando se considera os pontos não-nulos de seu gradiente.

[2] As aspas são apenas para não deixar totalmente de lado o formalismo matemático, segundo o qual δ não corresponde à uma função no sentido estrito, mas a uma distribuição temperada.

Lembrando que δ respeita a seguinte propriedade:

$$\int_R \delta(x)\phi(x)dx = \begin{cases} \phi(0), \text{ se } 0 \in R \\ 0, \text{ senão} \end{cases} \tag{4.3}$$

em que R é algum intervalo de integração. Assim, a função característica pode ser escrita na forma:

$$\mathbb{1}_\Omega(x,y,t) = \int_\Omega \delta(x-x')\delta(y-y')d\Omega \tag{4.4}$$

e como vale a fórmula de mudança de variáveis

$$\frac{d}{da}f(a-b) = -\frac{d}{db}f(a-b)$$

para qualquer função f diferenciável a valores reais, podemos calcular o gradiente da função característica $\mathbb{1}_\Omega$ pela fórmula abaixo, em que trocamos $\nabla = (\frac{\partial}{\partial x}, \frac{\partial}{\partial y})$ por $\nabla' = (\frac{\partial}{\partial x'}, \frac{\partial}{\partial y'})$:

$$\begin{array}{rcl} \nabla I & = & \int_\Omega \nabla\Big(\delta(x-x')\delta(y-y')\Big)d\Omega \\ & = & -\int_\Omega \nabla'\Big(\delta(x-x')\delta(y-y')\Big)d\Omega \end{array} \tag{4.5}$$

e pelo teorema de Gauss (do divergente) aplicado ao gradiente, obtemos uma integral em dimensão inferior:

$$\nabla I = -\int_{\partial\Omega} \delta(x-x')\delta(y-y')dS \tag{4.6}$$

Para o caso em que a densidade do fluido em cada fase é constante, podemos escrevê-la com o auxílio da função indicadora:

$$\rho(x,y,t) = \rho_i I(x,y,t) + \rho_0(1 - I(x,y,t)). \tag{4.7}$$

em que ρ_i é a densidade onde $I = 1$ e ρ_0 é a densidade onde $I = 0$.

Calculando o gradiente da densidade:

$$\begin{aligned} \nabla\rho &= \rho_i\nabla I - \rho_0\nabla I = (\rho_i - \rho_0)\nabla I \\ &= (\rho_0 - \rho_i)\rho\int \delta(x - x^{'})\delta(y - y^{'})\mathbf{n}^{'}ds^{'}. \end{aligned} \tag{4.8}$$

Com auxílio dos operadores elaborados acima, escreve-se as equações de Navier-Stokes em um formato válido para todo o domínio considerado, independente se propriedades materiais como a densidade ρ e a viscosidade μ mudam de forma discontínua.

Seja o campo de velocidade $\mathbf{u}$, a pressão P e f a força de corpo. A equação de movimento é dada por:

$$\frac{\partial\rho\mathbf{u}}{\partial t} + \nabla.\rho\mathbf{u}\mathbf{u} = -\nabla P + \rho\mathbf{f} + \nabla.\mu(\nabla\mathbf{u} + \nabla^T\mathbf{u}) + \int \sigma\kappa^{'}\mathbf{n}^{'}\delta^{\beta}(x - x^{'})ds^{'} \tag{4.9}$$

Sobre a equação:

- $\mathbf{x}$ é o ponto no qual a equação é calculada e $\mathbf{x}^{'}$ é um ponto na fronteira.
- $\mathbf{n}$ é um vetor unitário normal da interface em $\mathbf{x}$.
- As forças de superfície são adicionadas na interface.
- δ^{β} é uma função δ em duas ou três dimensões (respectivamente $\beta = 2$ ou 3) construída por multiplicações de funções δ unidimensionais.
- κ é a curvatura, para fluidos em duas dimensões, ou o dobro a curvatura média, para fluidos em três dimensões.

Uma vez obtida a equação 4.9, qualquer discretização feita sobre a malha euleriana fixa do problema pode ser utilizada para resolver o problema bifásico. A diferença entre as fases, caracterizada pela função densidade ρ pode ser atualizada resolvendo

uma equação de Poisson discretizada e suas propriedades materiais distribuídas ao longo da malha fixa por intermédio de suavizações envolvendo funções do tipo peso. As partículas, por sua vez, podem evoluir no tempo segundo um esquema de integração da equação 4.2, por exemplo pelo método de Euler explícito.

4.1.2 Construção da Função Indicadora

Nesta seção será construída uma função indicadora de um conjunto $\Omega \in \mathbb{R}^n$. O procedimento de obtenção da função indicadora I que segue adiante é tal qual feito em [8]. Antes de alcançar I, são construidas funções intermediárias que englobam, na sua totalidade I_a e $\tilde{I_a}$. A primeira dá valores apenas nos centros das células da malha, sendo necessária uma extensão para todo o domínio. A segunda é esta extensão obtida por interpolação.

Calculando I_a

Iniciamos com o conhecimento *a priori* de uma coleção de pontos pertencentes à interface. Esses pontos serão úteis na construção da condição inicial da equação de Laplace

$$\Delta I_a = 0 \tag{4.10}$$

que será resolvida numericamente.

As condições iniciais são obtidas fazendo $I_a = 0$ nas células periféricas da malha e $I_a = 1$ nas células que contém algum ponto da coleção pertencente à interface. A função I_a obtida após a resolução numérica da equação de Laplace. Nas células que interceptam Ω a função I_a vale 1, nas células periféricas I_a vale 0 e nos demais pontos I_a recebe algum valor entre 0 e 1.

Para finalizar, reajusta-se a função I_a uma vez fixado um parâmetro $\epsilon > 0$. Obtemos então uma nova função $I_{a,\epsilon}$:

$$I_{a,\epsilon} = \begin{cases} 0, \text{ se } I_a < 1 - \epsilon \\ I_a, \text{ senão.} \end{cases} \tag{4.11}$$

Suspenderemos adiante o índice ϵ da função obtida, isto é denotaremos por I_a a função $I_{a,\epsilon}$.

Definição 4.1 *Considerando o que foi discutido acima, chamaremos de* ***células interiores*** *aquelas em que a função indicadora* I_a *vale* 1*, assim como em todas as suas células vizinhas. Analogamente,* ***células exteriores*** *são aquelas em que* I_a *vale* 0 *em seu centro e no centro de suas células vizinhas.*

Calculando $\tilde{I_a}$

A aplicação I_a obtida acima fornece valores da função indicadora apenas nos centros $\vec{x}_g$ de cada célula g da malha. O autor sugere uma suavização por intermédio do uso de B-splines. Constroe-se, assim, uma nova função $\tilde{I_a}(\cdot)$:

$$\tilde{I_a}(\vec{x}) = \sum_g I_a(\vec{x}_g) S(\vec{x} - \vec{x}_g) \tag{4.12}$$

em que $S(\cdot)$ é produto tensorial de B-splines unidimensionais M:

$$S(\vec{x} - \vec{x}_g) = M(\vec{x} - \vec{x}_g; \Delta x) M(\vec{y} - \vec{y}_g, \Delta y) M(\vec{z} - \vec{z}_g, \Delta z). \tag{4.13}$$

O resultado é uma função $\vec{I_a}$ que aproxima a função indicadora, variando entre 0 e 1 em uma região de diâmetro h, na vizinhança da interface.

Calculando I

Encontramos a função indicadora I procurada adicionando correções na função

$\tilde{I_a}$. O interesse agora é deixar a função constante (dada uma escolha $I_c \in (0,1)$ de constante) ao longo da interface $\partial\Omega$:

$$I \mid_{\partial\Omega} \equiv I_c$$

Para fazê-lo, insere-se correções δI_p ao longo dos N_p pontos conhecidos da interface:

$$I(x) = \sum_p \delta I_p S(\vec{x} - \vec{x}_p) + \tilde{I_a}(\vec{x}) \tag{4.14}$$

e determinar estas correções δI_p remete à resolução das equações:

$$\begin{cases} \vdots \\ I(x_p) = I_c \qquad \text{para } p = 1, \cdots, N_p \\ \vdots \end{cases} \tag{4.15}$$

$$\begin{cases} \vdots \\ \sum_q \delta I_q S(\vec{x}_p - \vec{x}_q) = I_c - \tilde{I_a}(\vec{x}) \qquad \text{para } p = 1, \cdots, N_p \\ \vdots \end{cases} \tag{4.16}$$

4.2 Métodos Baseados em Partículas

À partir da década de 1980 surgiram técnicas computacionais para tratar nuvens de pontos de topologia e geometrias arbitrárias, seguindo o trabalho de Boissonnat [35]. O interesse por tais técnicas acentuou em virtude de sua serventia em diversas áreas de aplicação porquanto ultimamente encontramos a geometria computacional tridimensional presente como recurso no cinema, jogos, mapeamento topográfico, medicina, etc.

Em particular, podemos citar abordagens *point based surfaces* tais como as técnicas level set, que reconstroem superfícies à partir de nuvens de pontos através de funções

level set e que permitem considerar a superfície representada como uma curva de nível desta função.

Segundo Alexa[5], uma das razões que justifica a crescente popularidade do método foi o surgimento de *scanners* de objetos tridimensionais. Nesses, a taxa de amostragem é alta, assim como o ruido presente – as chamadas informações espúrias. Outra razão para a crescente popularidade do método em computação gráfica, segundo aponta Alexa, é o surgimento de mecanismos de scan acurados que propiciam um conjunto denso de pontos, que serve de representação inicial do modelo físico. A evolução da superfície à partir de então, fica a cargo do próprio método *level set*.

Outro aspecto atraente da abrodagem por nuvem de pontos é seu caráter interdisciplinar. Seu estudo e desenvolvimento faz apelo a conhecimentos de estatística [52], geometria computacional ([37], [38], [39], [43], [44], [41]), programação linear [36], métodos de aproximações locais (mínimos quadrados) ([46],[5],[45],[48]), métodos de aproximação global ([53], [51], [54], [47]), equações diferenciais parciais ([40], [49], [50], [47]) e teoria de Morse discreta [42].

As técnicas empregadas em *level set* (conjuntos de nível) reconstroem superfícies considerando um conjunto finito de pontos $\{\mathbf{x}_i = (x_i,y_i,z_i) \in \mathbb{R}^3 \mid i \in I\}$, para algum conjunto de índices I. Nessas técnicas, a representação da superfície $\mathcal{S}$ prescinde do uso de malhas, como por exemplo tomando a como uma curva de nível de uma função $\Phi : \Omega \in \mathbb{R}^3 \to \mathbb{R}$:

$$\mathcal{S} = \{\mathbf{x} \mid \Phi(\mathbf{x}) = C\}$$

e utilizando técnicas de cálculo numérico para recuperar pontos $\mathbf{x}$ que satisfaçam

$$\Phi(\mathbf{x}) = C$$

em que C é alguma constante real. Uma tal função $\Phi(\cdot)$ é chamada *level set*.

Em computação gráfica, chamamos de *Level Set* a técnica empregada para reconhe-

cer uma superfície como uma curva de nível de uma função, que por sua vez também é chamada de *Level Set*. Esta técnica recebeu considerável atenção nas últimas décadas por sua elegância, simplicidade e eficiência. O procedimento é motivado pela geometria diferencial e a noção que uma superfície pode ser localmente aproximada por uma função.

Tão logo aplicada em Mecânica dos Fluidos, a função level-set quando usada para representar uma interface que separa dois fluidos com propriedades materiais distintas trata alguns fenômenos com muita naturalidade enquanto outras técnicas apresentam dificuldades em conciliar sua complexidade intrínseca. Como exemplo notável, temos a mudança de topologia. O próprio aspecto matemático do tratamento de superfícies por intermédio de funções *level set* lida com esta questão de forma natural, que se extende à sua aplicação computacional.

Figura 4.1: Exemplos de *scanners* (imagens de http://www.cyberware.com)

Chamamos de **processo de amostragem**, ou simplesmente **amostragem**, o re-

sultado do ato de gerar pontos (ou efetuar uma escolha representativa de pontos) a partir de uma superfície. Em inglês, o processo é chamado de *sampling*. O número de pontos escolhidos depende da precisão desejada, muito embora uma quantidade excessiva de pontos possa causar ambiguidade, o que atravanca a eficiência do algoritmo. As aproximações engendradas pelos métodos obedecem alguma ordem $\mathcal{O}(h^r)$, assim, em função da ordem da precisão almejada e das características intrínsecas de cada problema (como por exemplo um objeto que contenha regiões de alta curvatura) deve-se determinar a quantidade de pontos necessários.

4.2.1 Introdução do método

Dado um conjunto de pontos $P = \{\vec{p}_i\}$ (obtidos possivelmente por um aparelho de *scan* 3D), definimos uma superfície suave $\mathcal{S}_P$ (superfície MLS) baseada nos pontos do input.Uma possível maneira de se triar os pontos é substituir P definindo a superfície $\mathcal{S}_P$ e a partir desta escolher uma amostra $R = \{\vec{r}_i\}$ mais refinada de pontos. Estes são chamados **pontos de representação**. Em seguida, define-se a respectiva superfície $\mathcal{S}_R$, que por sua vez aproxima $\mathcal{S}_P$.

Se temos $\Phi_0, \ldots, \Phi_N$, uma família $\mathcal{F}$ de funções linearmente independentes, podemos construir uma aplicação que associa à um vetor de parâmetros

$$\vec{a} = (a_0, \cdots, a_N)$$

uma aplicação pertencente ao conjunto de todas as combinações lineares de $\mathcal{F}$:

$$\begin{array}{ccc} \mathbb{R}^N & \longrightarrow & \mathbb{R}^N \cdot \mathcal{F} \\ \vec{a} & \longmapsto & F_{\vec{a}}(\cdot) = \sum_i a_i \Phi_i(\cdot) \end{array} \tag{4.17}$$

Uma vez que a escolha da família de funções $\mathcal{F}$ for estabelecida, a função *level set* $F_{\vec{a}}$ procurada será confundida com a busca pelo parâmetro $\vec{a}$, e tal função representará

localmente uma superfície implícita que aproxima o conjunto de pontos da superfície original.

Observação.- Cabe notar que a aproximação da superfície pelo isocontorno da função *level set* é segundo um ponto $\vec{p}$. Em outras palavras, o parâmetro $\vec{a}$ depende de $\vec{p}$.

A superfície definida pelo conjunto de pontos é uma variedade e espera-se que seja suave, mais especificamente C^∞, dado que os pontos estejam suficientemente próximos da superfície suave a ser representada. Para se computar um ponto na superfície somente uma vizinhança local deste ponto é requisitada.

Considere um ponto fixo $\vec{p}$. Tal ponto dará origem ao parâmetro $\vec{a}$ e por conseguinte a função $F_{\vec{a}}$ que representa a superfície implícita locamente pela curva de nível. Obviamente, a geometria da curva que queremos determinar com este procedimento deve priorizar a ação dos pontos $\vec{p}_i$ quanto mais próximos estiverem de $\vec{p}$.

Para a consecução de tal efeito, atribuímos pesos w_i aos pontos $\vec{p}_i$, determinados em função da distância desses pontos à $\vec{p}$, e então estabelecemos as equações que darão origem ao parâmetro $\vec{a}$. No capítulo introdutório foi mostrado um exemplo de função peso dado na forma de uma gaussiana. Esta função rapidamente aniquila a ação dos pontos na medida em que nos afastamos de $\vec{p}$, garantindo que estamos lidando com o problema à guisa legitimamente local.

O transporte da interface é determinado por um vetor velocidade $\vec{u}$, o qual pode depender de uma grande quantidade de variáveis, entre as quais posição, tempo, geometria da interface, etc. ou pode ser dado externamente como, por exemplo, a velocidade material na simulação de fluxo de um fluido.

4.2.2 Funções *level set*

A idéia principal por detrás do método *level set* é mergulhar uma interface Γ em $\mathbb{R}$ que seja a fronteira de uma região aberta $\Omega \subset \mathbb{R}^3$ dada pela 0-isosuperfície de uma função

$\phi(\vec{x}, t)$, definida em um domínio de maior dimensão. Tal função, denominada **função *level set***, possui as propriedades que seguem:

$$\phi(\vec{x},t) > 0 \qquad \text{para } \vec{x} \in \Omega$$

$$\phi(\vec{x},t) \leq 0 \qquad \text{para } \vec{x} \notin \Omega$$

em que incluiu-se $\phi = 0$ na partição dos valores negativos de ϕ afim de evitar a necessidade de tratar um caso isolado. A interface encontra-se entre $\{x|\phi(x) > 0\}$ e $\{x|\phi(x) = 0\}$, mas pode ser obviamente identificada como

$$\{x|\phi(x) = 0\}.$$

Vale notar que ϕ é uma função em $\mathbb{R}^3$, o que reduz sensivelmente a complexidade na hora de descrever a interface, principalmente quando submetida à mudanças topológicas.

Quando o campo de velocidades é dado externamente, a evolução da equação para a função *level set* pode ser dada por

$$\phi_t + \vec{u} \cdot \nabla\phi = 0. \tag{4.18}$$

Esta equação necessita ser resolvida apenas localmente, na proximidade da interface.

É conveniente tomar ϕ pela distância sinalizada da interface, de tal maneira que $\|\nabla\phi\| = 1$. Isto garante que a função *level set* variará à guisa suave, satisfazendo regularidade suficiente para trabalhar com métodos numéricos acurados de alta ordem. Infelizmente, a função *level set* pode rapidamente deixar de ser uma função sinalizada, especialmente ao trabalharmos com fluxos sofrendo mudanças topológicas extremas. Assim, algoritmos de reinicialização são úteis se quisermos manter as propriedades da função sinalizada ao resolvermos até o estado estacionário (ao passo que tomamos um

tempo fictício $\tau \longrightarrow \infty$) a equação

$$\phi_\tau + \text{sgn}(\phi_0)(\|\nabla\phi\| - 1) = 0 \tag{4.19}$$

em que $\text{sgn}(\phi_0)$ é uma função de sinal em uma dimensão que pode ser aproximada por

$$\text{sgn}(\phi_0) = \frac{\phi_0}{\sqrt{\phi_0^2 + (\Delta x)^2}} \tag{4.20}$$

Quantidades geométricas podem ser calculadas a partir das funções *level set*, incluindo o versor normal

$$\vec{N} = \frac{\nabla\phi}{\|\nabla\phi\|}, \tag{4.21}$$

e a curvatura,

$$\kappa = \nabla \cdot \left(\frac{\nabla\phi}{\|\nabla\phi\|} \right). \tag{4.22}$$

As derivadas espaciais nas equações acima podem ser calculadas usando as aproximações padrão de diferenças finitas centradas de segunda ordem quando os denominadores são não nulos.

4.2.3 Mapeamento Local por Planos

Seguindo [5] , dado um conjunto de pontos $P = \{p_i\}$,$i \in \{0,\ldots,N-1\}$ extraídos de uma superfície $\mathcal{S}_P$ queremos projetar um ponto r na superfície $\mathcal{S}$ que aproxima os pontos p_i. Nesta seção, a superfície $\mathcal{S}_P$ é um plano.

A equação de um plano Π de normal $\vec{n}$ passando por algum ponto $\vec{q} = (q_0,q_1,q_2)$ se escreve:

$$\vec{n} \cdot \vec{x} = \kappa \tag{4.23}$$

em que $\kappa = \vec{n} \cdot \vec{q}$. Deduzimos trivialmente uma função *level set* $F_{[\vec{n},\kappa]} : \mathbb{R}^3 \longrightarrow \mathbb{R}$ que define o plano implicitamente:

$$\begin{aligned} F_{[\vec{n},\kappa]}(\vec{x}) &= \vec{n}\cdot\vec{x}-\kappa \\ \Pi &= \{\vec{x}\in\mathbb{R}^3 \mid F_{[\vec{n},\kappa]}(\vec{x})=0\} \end{aligned} \tag{4.24}$$

Em analogia ao que foi escrito acima, estamos definindo o plano local de acordo com um vetor parâmetro $\vec{a}=[\vec{n},\kappa]=(n_0,n_1,n_2,\kappa)$ e a função *level set* $F_{\vec{a}}(\cdot)=F_{[\vec{n},\kappa]}(\cdot)$ que o define implicitamente.

O plano Π que queremos determinar é segundo um ponto $\vec{p}$ e Alexa o denomina **domínio de referência**. Para encontrar o parâmetro $\vec{a}$ em função deste ponto, é visada a minimização do funcional:

$$\sum_{i=0}^{N-1}\left(F_{\vec{a}}(\vec{p}_i)\right)^2 w_i \tag{4.25}$$

em que w_i são pesos fornecidos por $\theta(\|\vec{p}-\vec{p}_i\|)$ para alguma função suave, monótona e decrescente a valores não negativos em todo espaço.

A função *level set* usada por Alexa, no entanto, é ligeiramente distinta. Motivado pelo interesse de priorizar o peso dos pontos próximos *ao plano* ao invés de $\vec{p}$, consideramos a projeção $\vec{q}$ de $\vec{q}$ no plano Π (ainda a determinar) e atribuímos os pesos conforme a distância dos pontos p_i à projeção q do domínio de referência.

Explicitamente temos

$$\begin{aligned} \vec{q} &= \vec{p}+\lambda\vec{n} \\ \lambda &= \kappa-<\vec{n},\vec{p}> \end{aligned}$$

e como, por definição, temos

$$\kappa=<\vec{n},\vec{q}>\Longrightarrow F_{\vec{a}}(\vec{p}_i)=\left(\vec{n},\vec{p}_i-\vec{p}-\lambda\vec{n}\right)$$

o funcional fica

$$\sum_{i=0}^{N-1} \left(\vec{n}, \vec{p}_i - \vec{p} - \lambda \vec{n} \right)^2 \theta(\|\vec{p}_i - \vec{p} - \lambda \vec{n}\|). \tag{4.26}$$

1. Definimos o operador $\mathcal{Q}(r) = q = r + tn$, o mínimo de (4.10) com o menor r e o plano tangente local H próximo de r analogamente. O domínio local de referência é então dado por um sistema de coordenadas ortogonal de H tal que q é a origem deste sistema.

2. **Mapa Local:** O domínio de referência para r é usado para calcular uma aproximação polinomial local da superfície em uma vizinhança de r. Seja q_i a projeção de p_i em H e f_i a altura de p_i em H, ou seja, $f = n \cdot (p_i - q)$. Computamos os coeficientes de uma aproximação polinomial g de tal forma que o erro ponderado dos mínimos quadrados

$$\sum_{i=1}^{N} \big(g(x_i,y_i) - f_i \big)^2 \theta(\|p_i - q\|) \tag{4.27}$$

seja minimizado. Aqui, (x_i, y_i) pela representação de q_i em um sistema de coordenadas local de H. Observe que novamente as distâncias usadas para a função peso são a partir da projeção r em H. A projeção $\mathcal{P}$ de r em $\mathcal{S}_P$ é definida pelo valor do polinômio na origem, isto é

$$\mathcal{P}(r) = q + g(0,0)n = r + (t + g(0,0))n. \tag{4.28}$$

4.3 Adaptação de Superfícies Algébricas

No trabalho [3] apresentou-se uma nova definição de *Point Set Surface* (PSS) da vertente *Moving Least Squares Surface* (Superfícies MLS), ajustando esferas algébricas. A vantagem central desta abordagem em relação às aproximações locais por planos algébricos é a melhoria na estabilidade da projeção no contexto de baixa taxa de

amostragem e na presença de alta curvatura. Ademais, o método fornece uma boa estimativa da curvatura média da superfície sem maiores custos adicionais e permite uma boa manipulação de superfícies agudas.

Dado um conjunto de pontos $P = \{p_i \in \mathbb{R}^d\}$, define-se uma superfície suave $\mathcal{S}_P$ aproximando P por intermédio do *Surface Moving Least Squares*, por ajuste de esferas. O método trabalha tanto com nuvens de pontos quanto pontos munidos do vetor normal. A presença destes vetores normais leva à simplificação, maior eficiência e maior robustez dos algoritmos de aproximação.

Considerando superfície compreendida por uma esfera centrada no ponto $\vec{c} = (C_x,C_y,C_z)$ e com raio de tamanho R, trabalha-se a sua equação para chegar em uma função implícita que a define:

$$\begin{aligned} R^2 &= (x - C_x)^2 + (y - C_y)^2 + (z - C_z)^2 \\ 0 &= \Big(x^2 + y^2 + z^2\Big) - 2\Big(xC_x + yC_y + zC_z\Big) + \Big(C_x^2 + C_y^2 + C_z^2 - R^2\Big) \end{aligned}$$

$$0 = X^T \cdot X - 2X^T \cdot \begin{bmatrix} C_x \\ C_y \\ C_z \end{bmatrix} + \Big(C_x^2 + C_y^2 + C_z^2 - R^2\Big)$$

e finalmente obtemos:

$$[1, X^T, X^T \cdot X] \cdot \begin{bmatrix} C_x^2 + C_y^2 + C_z^2 - R^2 \\ -2C_x \\ -2C_y \\ -2C_z \\ 1 \end{bmatrix} = 0 \tag{4.29}$$

$$[1, X^T, X^T \cdot X] \cdot \begin{bmatrix} \|\vec{c}\|^2 - R^2 \\ -2\vec{c} \\ 1 \end{bmatrix} = 0 \tag{4.30}$$

Assim, associamos uma superfície a um parâmetro $\vec{a} \in \mathbb{R}^5$, correspondendo ao vetor coluna da equação escrita acima. Logo, é lícito definir uma esfera algébrica pela definição do campo escalar

$$F_{\vec{a}}(\vec{x}) = [1, \vec{x}^T, \vec{x}^T\vec{x}] \cdot \vec{a}$$

(onde $\vec{a} = [a_0, \ldots, a_{d+1}] \in \mathbb{R}^5$ é um vetor de coeficientes escalares que descrevem a esfera) e considerando sua 0-isosuperfície (isto é, sua curva de nível em zero). Para $a_{d+1} \neq 0$, o centro $\vec{c}$ e o raio r correspondentes à superfície são dados por:

$$\begin{aligned} \vec{c} &= -\tfrac{1}{2a_{d+1}}[a_1, \ldots, a_d]^T \\ r &= \left(\|\vec{c}\|^2 - \tfrac{a_0}{a_d+1}\right)^{1/2} \end{aligned} \tag{4.31}$$

Assim como nos procedimentos de minimização considerados no capítulo de conceitos básicos (seção 2.5), tomaremos uma família de funções

$$f_1(\vec{x}) = 1, f_2(\vec{x}) = x_1, f_3(\vec{x}) = x_2, f_4(\vec{x}) = x_3, f_5(\vec{x}) = (\vec{x})^T \cdot \vec{x} \tag{4.32}$$

de tal maneira que - servindo-se da notação na seção 2.5 - o conjunto abaixo

$$\{\, \vec{x} \mid F_{\vec{b}}(\vec{x}) = 0\} \tag{4.33}$$

define uma esfera (de onde vem o nome *esferas algébricas*). Os pontos de controle

fornecidos $P = \{\vec{p}_k\}_k$ e os pesos w_i conduzem à função de energia que se quer minimizar:

$$\begin{aligned} J(\vec{b}) &= \sum_i w_i (F_{\vec{b}}(p_i))^2 \\ &= \sum_i w_i (\sum_k b_k f_k(p_i))^2 \end{aligned} \tag{4.34}$$

e cuja manipulação algébrica fornece a forma matricial:

$$J(\vec{b}) = (\vec{b})^T \cdot \mathbf{M} \cdot \vec{b} \tag{4.35}$$

em que a matriz $\mathbf{M} = (M_{kl})_{kl}$ é dada por

$$M_{kl} = \sum_i w_i f_k(\vec{p}_i) f_l(p_i). \tag{4.36}$$

Lembrando que o vetor normal da superfície *level-set* - neste caso a superfície AMLS - é calculado segundo

$$\vec{n}(\vec{x}) = \vec{\nabla} F_{\vec{a}}(\vec{x}) \tag{4.37}$$

a menos de normalização. Como este problema de minimização admite a solução trivial, uma formulação para não obter sempre a solução nula - que é evidentemente indesejada, pois rende todo o domínio idêntico ao conjunto em (4.33) - é a condição de *Pratt*:

$$\|\nabla F_{\vec{b}}\| = 1 \tag{4.38}$$

cujo o cálculo explícito fornece

$$b_2^2 + b_3^2 + b_4^2 - 4b_1 b_2 = 1 \tag{4.39}$$

e que admite a forma matricial

$$[\vec{b}]^T \cdot \mathbf{C} \cdot \vec{b} = 1 \tag{4.40}$$

em que $\mathbf{C} = \left(C_{ij}\right)_{ij} \in \mathcal{M}_{5\times 5}$ e tem seus coeficientes dados por

$$C_{ij} = \begin{cases} 1, & \text{se } (i,j) \in \{(2,2),(3,3),(4,4)\} \\ -2, & \text{se } (i,j) \in \{(1,5),(5,1)\} \\ 0, & \text{caso contrário} \end{cases} \tag{4.41}$$

O parâmetro $\vec{a}$ procurado é obtido via método dos multiplicadores de Lagrange, solução de:

$$\mathbf{M} \cdot \vec{a} = \lambda \mathbf{C} \cdot \vec{a} \tag{4.42}$$

para o menor auto-valor λ e normalizado tal que $[\vec{a}]^T \cdot \mathbf{C} \cdot \vec{a} = 1$.

Esse parâmetro $\vec{a}$, conforme menciando acima, define implicitamente uma esfera pelo conjunto (4.33) e o método é chamado *algebraic moving least squares*(AMLS). Uma extensão do trabalho procurou encontrar este parâmetro orientando-se pelos passos abaixo. Antes, será necessário estabelecer a notação $\vec{a} = [a_1, \vec{a}'] \in \mathbb{R}^5$, com $a_1 \in \mathbb{R}$ e $\vec{a}' \in \mathbb{R}^4$.

PASSO A. Usando a fórmula (4.37) pelo parâmetro obtido em (4.42), calcula-se para todo ponto $\vec{p}_i \in P$ o vetor normal $\vec{N}_i = \vec{n}_i(\vec{p}_i)$. A orientação do vetor é corrigida pelo vetor normal $\widetilde{N_i}$ que se dispõe em cada ponto p_i, cuidando que o ângulo entre o novo vetor $\vec{N}_i$ obtido não faça ângulo maior que $\pi/2$ com $\widetilde{N_i}$.

PASSO B. Primeiro, obtém-se o vetor $\vec{a}'$ minimizando

$$\sum_i w_i \|\nabla F_{\vec{a}'}(p_i) - \vec{N}_i\|^2 \tag{4.43}$$

e em seguida o termo faltante a_1 minimizando

$$\sum_i w_i (F_{\vec{b}}(p_i))^2 \tag{4.44}$$

de tal forma que respeite $\vec{b}' = \vec{a}'$.

Dado caráter robusto da superfície reconstruída por este método, que produz uma superfície AMLS, o nome dado foi *Robust Algebraic Moving Least Squares*(RAMLS).

Capítulo 5

Projeto

O objetivo do desenvolvimento deste trabalho foi cumprir com três metas principais: primeiramente, encontrar uma maneira natural de estender a representação de superfície que se mantinha conforme a implementação original desde a concepção do *Freeflow* há mais de 20 anos, utilizando algum uma formulação por funções implícitamente; em seguida, comparar esta nova representação de superfície com aquela de que se dispunha inicialmente no método GENSMAC; e finalmente, implementar um algoritmo que abrisse caminho para posteriores extensões na maneira de se representar a superfície. Neste capítulo será exposta a escolha de representação local da interface, será explicado o conceito matemático de partição da unidade implícita e como é utilizado para reconstruir toda a interface a partir das representações locais.

Conforme será visto nas seções que seguem, os polinômios calculados por mínimos quadrados que aproximam a interfacem são obtidos de forma inteiramente análoga à maneira como se calculam as estimativas de normais e curvaturas no *Freeflow*, o que cobre a primeira meta citada acima. Os resultados de comparação podem ser conferidos no capítulo de resultados e a possibilidade de extensão do trabalho é discutida no último capítulo sobre trabalhos futuros.

5.1 Representações Locais

Seja $\mathcal{S}$ uma interface descrita por uma curva fechada sobre um domínio em que há uma malha euleriana e $\mathcal{P} = \{\vec{p}_k\}_k \subset \mathcal{S}$ uma nuvem de pontos pertencentes a esta superfície. De início vale ressaltar que as células de interesse nesta malha euleriana são as que interceptam a interface (cf. figura 5.1). Estas células podem ser do tipo `SURFACE` ou `FULL`, conforme a classificação das células tratadas no capítulo sobre o *Freeflow*. Na figura 5.1, as células de interesse são distinguidas pela presença explicitada do seu respectivo centro de célula.

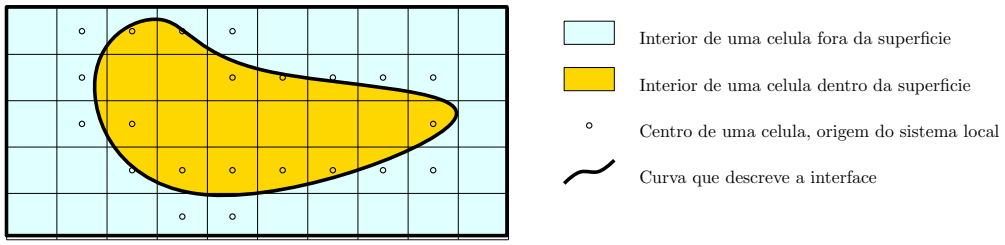


Figura 5.1: Centros das células que interceptam a interface.

Ao acessar tais células, recupera-se um sistema de referência. Assim, seja (i,j) o índice de uma célula $\mathcal{C}_{ij}$ que intercepta a interface. Associado a esta célula há um *sistema de referência local*

$$\mathcal{R}_{ij} = \left\{ \vec{c}_{ij} \ , \ [\vec{u}_{ij}, \vec{v}_{ij}] \ , \ r_{ij} \ , \ \varphi_{ij}(\cdot) \right\} \tag{5.1}$$

em que $\vec{c}_{ij}$ é o centro da célula, $[\vec{u}_{ij},\vec{v}_{ij}]$ são dois vetores unitários formando um sistema local de eixos, $r_{ij} > 0$ é o raio de atuação da representação local e

$$\begin{array}{rccc} \varphi_{ij}: & \mathbb{R} & \longrightarrow & \mathbb{R} \\ & \xi & \mapsto & \eta = \varphi_{ij}(\xi) \end{array}$$

é uma função cujo gráfico aproxima a interface na vizinhança $B_{ij} := B(\vec{c}_{ij},r_{ij})$ da célula $\mathcal{C}_{ij}$:

$$\mathcal{S} \cap B_{ij} \;\approx\; \left\{ \left[\xi,\varphi(\xi)\right]_{(i,j)} \;\middle|\; \xi \in [-r_{ij},r_{ij}] \right\}, \tag{5.2}$$

havendo aqui introduzida a notação $[\cdot,\cdot]_{(i,j)}$ que designa as coordenadas locais da célula $\mathcal{C}_{ij}$, isto é

$$\left[\xi,\eta\right]_{(i,j)} = [\vec{c}_{ij}] + \xi[\vec{u}_{ij}] + \eta[\vec{v}_{ij}].$$

Uma célula pode ser dividida em subcélulas e conter várias representações locais. A figura 5.2 mostra uma célula sendo dividida em $(k+1)^2$ células, para $k = 0,1,2,3$ (isto é, k subdivisões na direção x e k subdivisões na direção y). Nem toda subcélula conterá partículas da interface e por conseguinte uma tal subcélula não precisará de uma representação local. Mas para aquelas que interceptam a interface haverá uma representação de subcélula $\mathcal{R}^{\ell}_{ij}$, em que ℓ é um índice entre 1 e $(k+1)^2$ que situa a subcélula[1] na célula $\mathcal{C}_{ij}$. Não obstante, no interesse de manter compacta a notação matemática da formulação aqui elaborada, será considerado que no nível de uma célula há necessidade de somente uma representação local, suspendendo o índice ℓ na notação dos vetores e escalares que sucedem.

O centro $\vec{c}_{ij}$ é o ponto médio da célula obtido trivialmente. Se a célula é dada pelo domínio $\mathcal{C}_{ij} = [x_i, x_{i+1}] \times [y_j, y_{j+1}]$, então o centro é dado por

$$\vec{c}_{ij} = \left[\frac{x_i + x_{i+1}}{2}, \frac{y_j + y_{j+1}}{2}\right] \tag{5.3}$$

Os eixos locais $[\vec{u}_{ij}, \vec{v}_{ij}]$ são obtidos aproximando os pontos $B_{ij} \cap \mathcal{P}$ por uma esfera,

[1] Alternativamente, para os aficcionados da indexação típica da linguagem C por exemplo, pode-se considerar os índices de 0 a $(k+1)^2 - 1$.

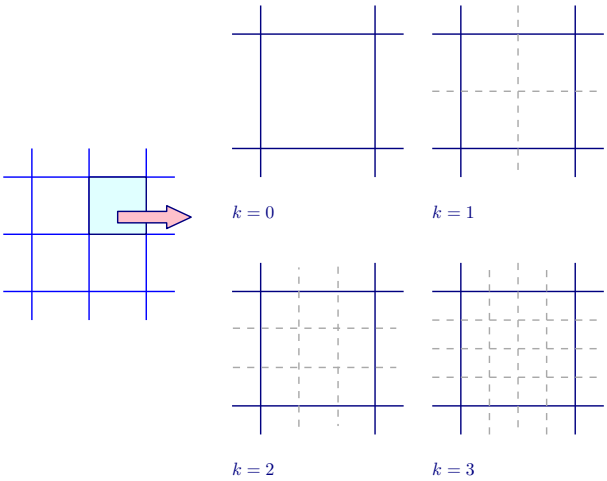


Figura 5.2: Subdivisões da célula.

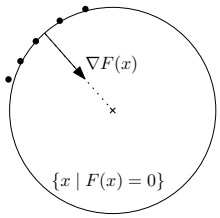


Figura 5.3: Esfera que aproxima os pontos e retorna uma normal que servirá de eixo local.

ajustando o parâmetro $\underline{a} = (a,b,c)$ da função[2]

$$F_{\underline{a}}(\vec{x}) = F_{\underline{a}}(x,y) = ax + by + c - (x^2 + y^2) \tag{5.4}$$

de tal maneira que

$$\underline{a} = \arg\min_{\underline{b}} \left[\sum_{p \in \mathcal{P}} F_{\underline{b}}(\vec{p}) \right], \tag{5.5}$$

conforme ilustrado na figura 5.3.

Este parâmetro $\underline{a}$ define uma esfera implicitamente pela fórmula

$$F_{\underline{a}} = 0$$

[2] A rigor, o parâmetro é melhor representado por $\underline{a}_{ij} = (a_{ij},b_{ij},c_{ij})$, sendo que haverá um cada célula $\mathcal{C}(i,j)$.

e é a esfera que aproxima localmente a interface. Obtém-se pela fórmula da direção normal de uma curva definida implicitamente o vetor normal $\widetilde{n}_{ij}$ da interface, a menos da orientação correta:

$$\widetilde{n}_{ij} = \frac{\nabla F_{\underline{a}}(\vec{c}_{ij})}{\|\nabla F_{\underline{a}}(\vec{c}_{ij})\|}. \tag{5.6}$$

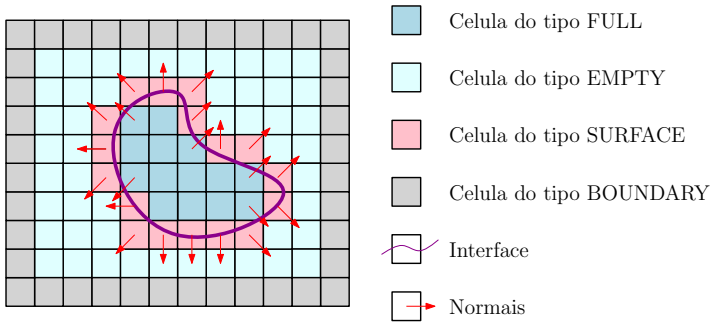


Figura 5.4: Orientação da normal segundo tipo de célula.

Finalmente, a correção da orientação do vetor normal $\widetilde{n}_{ij}$ depende de um vetor de referência $\vec{n}^R_{ij}$. Para tal há duas possibilidades:

(i) Associados à cada ponto da nuvem de pontos $\mathcal{P}$ há os respectivos vetores normais de cada ponto $\mathcal{N} = \{\vec{n}_k\}_k$ (com a indexação de tal forma que $\vec{n}_k \in \mathcal{N}$ seja a normal associada à $\vec{p}_k \in \mathcal{P}$) e o vetor de referência $\vec{n}^R_{ij}$ é a média do conjunto $C := \{\vec{n}_k \in \mathcal{N} \mid \vec{p}_k \in B_{ij}\}$:

$$\vec{n}^R_{ij} = \frac{1}{|C|} \sum_{\vec{n} \in C} \vec{n} \tag{5.7}$$

em que $|C|$ é a cardinalidade (i.e. número de elementos) do conjunto C;

(ii) Observando o rótulo das células vizinhas, é escolhida a direção da célula vizinha do tipo `EMPTY` (cf. figura 5.4), analogamente ao que foi descrito na seção sobre normais no capítulo do *Freeflow*;

Para corrigir a orientação de $\widetilde{n}_{ij}$, considera-se a função sinal $s : \mathbb{R} \to \{-1,1\}$:

$$s(x) = \begin{cases} 1 & \text{se } x \geq 0 \\ \\ -1 & \text{caso contrário.} \end{cases} \tag{5.8}$$

e o vetor $\vec{n}_{ij}$ com a boa orientação do vetor normal local é obtido

$$\vec{n}_{ij} = s(< \vec{n}^R_{ij}, \widetilde{n}_{ij} >) \, \widetilde{n}_{ij} \tag{5.9}$$

em que $< \cdot, \cdot >$ designa o produto interno.

Assim, obtido o vetor normal $\vec{n}_{ij}$, calculamos os eixos locais:

$$\begin{cases} \vec{u}_{ij} &= \vec{n}_{ij} \\ \\ \vec{v}_{ij} &= R_{\frac{\pi}{2}} \cdot \vec{n}_{ij} \end{cases} \tag{5.10}$$

em que $R_{\frac{\pi}{2}}$ é a matriz de rotação de $\frac{\pi}{2}$.

O raio de ação r_{ij} depende da escolha do usuário e o capítulo de resultados mostra a diferença de eficiência na aproximação ao mudar sua escolha. Seu propósito será melhor elucidado na seção seguinte, ao ser discutido funções do tipo peso e partições da unidade.

A função φ é uma representação local da superfície na vizinhança B_{ij} da célula. Neste trabalho φ é obtido por mínimos quadrados tal qual polinômio que aproxima os pontos da nuvem $\mathcal{P}$ contidos na vizinhança da célula $\mathcal{C}(i,j)$, escritos nas coordenadas do sistema de eixos locais; não obstante, a formulação matemática é inteiramente idêntica para interpolações locais, inclusive na construção da partição da unidade implícita na seção seguinte.

Mais precisamente, seja

$$V_{ij} = \left\{ \vec{\xi} = [\xi, \eta] = [\vec{x}]_{(i,j)} \;\middle|\; \vec{x} \in B_{ij} \right\} \tag{5.11}$$

os pontos da vizinhança B_{ij} escritos segundo coordenadas do sistema de referência local $\mathcal{R}_{ij}$. Então φ é o polinômio obtido por mínimos quadrados de tal forma que

$$\varphi = \arg \min_{f \in \mathbb{P}_n} \sum_{\vec{\xi} \in V} \left[\eta - f(\xi)\right]^2 \tag{5.12}$$

Este capítulo contém um apêndice com uma aproximação polinomial desenvolvida neste trabalho que conserva o volume da integral numérica dos pontos de entrada (cf. seção 5.6), sendo uma alternativa de representação local. Porém, a seção de resultados mostra por que razão não é uma boa escolha de aproximação por mínimos quadrados.

Finalmente, a representação (5.1) assume a forma enxuta

$$\boxed{\mathcal{R}_{ij} = \left\{ \vec{c}_{ij} \;,\; r_{ij} \;,\; \phi_{ij}(\cdot) \right\}} \tag{5.13}$$

se considerarmos as funções ξ_{ij},$\eta_{ij} : \mathbb{R}^2 \to \mathbb{R}$ definidas como

$$\begin{cases} \xi_{ij}(\vec{x}) &= \pi_1([\vec{x}]_{(i,j)}) \\ \\ \eta_{ij}(\vec{x}) &= \pi_2([\vec{x}]_{(i,j)}) \end{cases} \tag{5.14}$$

em que π_k retorna a k-ésima coordenada, para $k = 1,2$. Com efeito, a função φ_{ij} pode desta forma ser descrita na forma alternativa

$$\phi_{ij}(\vec{x}) = \eta_{ij}(\vec{x}) - \varphi_{ij} \circ \xi_{ij}(\vec{x}) \tag{5.15}$$

e o problema admite uma formulação equivalente: ao invés de procurar uma função φ_{ij} que satisfaz (5.2), procuramos ϕ_{ij} da forma (5.15) que satisfaz

$$\phi_{ij}(\vec{x}) \approx 0 \; \forall \, \vec{x} \in B_{ij} \tag{5.16}$$

5.2 Partição da Unidade Implícita (PUI)

Seja $\Omega \subset \mathbb{R}^n$ um conjunto limitado. Sejam $\{\Omega_i\}_{i=1}^M$ tal que cada Ω_i é limitado e

$$\Omega \subset \bigcup_{1 \leq i \leq M} \Omega_i. \tag{5.17}$$

Dizemos que uma família $\{w_i\}_{i=1}^M$ de funções não negativas é uma *partição da unidade k-estável* se

(i) $\text{supp}(w_i) \subseteq \Omega_i$

(ii) $\sum_{i=1}^{M}(w_i) \equiv 1$

(iii) para todo $\alpha \in \mathbb{N}^d$ com $|\alpha| < k$, existe uma constante C tal que

$$\|D^\alpha w_i\|_{L_\infty(\Omega_i)} \leq \frac{C_\alpha}{\delta_i^{|\alpha|}}, \; \forall i \tag{5.18}$$

em que $\delta_j = \sup_{x,y \in \Omega_i} \|x - y\|_2$.

Considerando um conjunto de índices I tal que $(i,j) \in I$ se e somente se a célula $\mathcal{C}(i,j)$ contém partículas - e que possui portanto um sistema de referência local $\mathcal{R}_{ij}$ - queremos construir uma função que reúne todas aproximações locais $\phi_{ij}(\cdot)$ em uma única aproximação global $P(\cdot)$.

Para tanto, consideramos as funções do tipo *peso*. As características desejadas para uma função w deste tipo são:

- ser a valores positivos $w : \mathbb{R}^2 \to \mathbb{R}^+$;
- ter suporte compacto

$$\exists\ r\,, \vec{c} \quad \text{t. q.} \qquad w\Big|_{\{\vec{x}\,|\,\|\vec{x}-\vec{c}\|>r\}} \equiv 0\,;$$

- ser decrescente conforme se afasta de um centro $\vec{c}$

$$\|\vec{x}-\vec{c}\| < \|\vec{y}-\vec{c}\| \iff \|w(\vec{x})\| \leq \|w(\vec{y})\|\ ;$$

Elaborando com maior especificidade esta última propriedade, por intermédio uma função $\theta : \mathbb{R} \to \mathbb{R}^+$ decrescente, as funções peso w_{ij} definidas para cada $(i,j) \in I$ se escrevem

$$w_{ij}(\vec{x}) = \frac{W_{ij}(\vec{x})}{\sum\limits_{(k,l)\in I} W_{kl}(\vec{x})} \tag{5.19}$$

com

$$W_{ij}(\vec{x}) = \theta\Big(\frac{\|\vec{x}-\vec{c}_{ij}\|}{r_{ij}}\Big). \tag{5.20}$$

Dentre as funções que respeitam estas características temos

$$\theta(t) = \begin{cases} (1-t^2)^4 & \text{se } t < 1 \\ 0 & \text{caso contrário.} \end{cases} \tag{5.21}$$

A propriedade (ii) da definição acima será útil no sentido que, dada uma função partição da unidade

$$P(x) = w_0(\vec{x})f_0(\vec{x}) + w_1(\vec{x})f_1(\vec{x}) + \cdots + w_n(\vec{x})f_n(\vec{x}) \tag{5.22}$$

de tal forma que

$$w_0(\vec{x}) + w_1(\vec{x}) + \cdots + w_n(\vec{x}) = 1, \tag{5.23}$$

então fixado $\vec{x}$ e definindo as constantes $\alpha_k = w_0(\vec{x})$, o valor $P(\vec{x})$ da função partição da unidade no ponto $\vec{x}$ é uma média ponderada de cada $f_k(\vec{x})$

$$P(\vec{x}) = \alpha_0 f_0(\vec{x}) + \alpha_1 f_1(\vec{x}) + \cdots + \alpha_n f_n(\vec{x}) \tag{5.24}$$

pois

$$\alpha_0 + \alpha_1 + \cdots + \alpha_n = 1 \tag{5.25}$$

Assim, a idéia de utilizar uma função peso $w_{ij}(\cdot)$, em nível intuitivo, é fazer com que a influência da representação local $\phi(\vec{x})$ na representação global $P(\vec{x})$ seja de acordo com a proximidade do ponto $\vec{x}$ ao ponto $\vec{c}$.

Neste ínterim, obtidas as representações locais $\mathcal{R}_{ij}$ para todo $(i,j) \in I$ a construção da função partição da unidade procede tal qual

$$\boxed{P(\vec{x}) = \sum_{(i,j)\in I} w_{ij}(\vec{x})\, \phi_{ij}(\vec{x})} \tag{5.26}$$

em que w_{ij} é dada por (5.19) e a representação local implícita ϕ_{ij} é dada por (5.13).

Uma vez obtida uma representação da superfície por função implícita pode-se obter imediatamente pelas fórmulas que seguem, para cada ponto $\vec{x}$, o vetor normal à interface neste ponto

$$\boxed{\vec{n}(\vec{x}) = \left. \frac{\left(\frac{\partial P}{\partial x}\right)^2 + \left(\frac{\partial P}{\partial y}\right)^2}{\left[\left(\frac{\partial P}{\partial x}\right)^2 + \left(\frac{\partial P}{\partial y}\right)^2\right]^{1/2}} \right|_{\vec{x}}} \tag{5.27}$$

e a curvatura κ da interface neste ponto

$$\kappa(\vec{x}) = \left. \frac{\left(\frac{\partial P}{\partial x}\right)^2 \frac{\partial^2 P}{\partial y^2} - 2\frac{\partial P}{\partial x}\frac{\partial P}{\partial y}\frac{\partial^2 P}{\partial x \partial y} + \left(\frac{\partial P}{\partial y}\right)^2 \frac{\partial P}{\partial x}}{\left[\left(\frac{\partial P}{\partial x}\right)^2 + \left(\frac{\partial P^2}{\partial y}\right)^2\right]^{3/2}} \right|_{\vec{x}} \tag{5.28}$$

Além de serem necessárias para o cálculo da normal e da curvatura, as derivadas da função partição da unidade serão utilizadas na projeção de um ponto à superfície PUI.

5.3 Derivadas da Partição da Unidade

Primeiro, indroduzimos as funções numerador $g(\cdot)$ e denominador $h(\cdot)$ da PUI

$$P(\vec{x}) = \frac{\sum_k w_k(\vec{x})\, f_k(\vec{x})}{\sum w_k(\vec{x})} =: \frac{g(\vec{x})}{h(\vec{x})} \tag{5.29}$$

e seguimos calculando a primeira derivada

$$\frac{\partial}{\partial x} P = \frac{\frac{\partial g}{\partial x}\, h - g\, \frac{\partial h}{\partial x}}{h^2} = \frac{G_x}{H} \tag{5.30}$$

$$\frac{\partial}{\partial y} P = \frac{\frac{\partial g}{\partial y}\, h - g\, \frac{\partial h}{\partial y}}{h^2} = \frac{G_y}{H} \tag{5.31}$$

em que definimos as funções G_x e G_y para auxiliar no cálculo da derivada segunda:

$$\frac{\partial^2}{\partial x^2} P = \frac{\frac{\partial G_x}{\partial x}\, H - G_x\, \frac{\partial H}{\partial x}}{H^2} \tag{5.32}$$

$$\frac{\partial^2}{\partial y^2} P = \frac{\frac{\partial G_y}{\partial y}\, H - G_y\, \frac{\partial H}{\partial y}}{H^2} \tag{5.33}$$

$$\frac{\partial^2}{\partial x \partial y} P = \frac{\frac{\partial G_x}{\partial y}\ H - G_x\ \frac{\partial H}{\partial y}}{H^2} \tag{5.34}$$

e as derivadas das funções numerador

$$\begin{aligned}
\frac{\partial G_x}{\partial x} &= \frac{\partial^2 g}{\partial x^2}\ h - g\ \frac{\partial^2 h}{\partial x^2} \\
\frac{\partial G_y}{\partial y} &= \frac{\partial^2 g}{\partial y^2}\ h - g\ \frac{\partial^2 h}{\partial y^2} \\
\frac{\partial G_x}{\partial y} &= \frac{\partial^2 g}{\partial x \partial y}\ h - g\ \frac{\partial^2 h}{\partial x \partial y} \\
\frac{\partial g}{\partial x} &= \sum_k \frac{\partial w_k}{\partial x} f_k + w_k \frac{\partial f_k}{\partial x} \\
\frac{\partial g}{\partial y} &= \sum_k \frac{\partial w_k}{\partial y} f_k + w_k \frac{\partial f_k}{\partial y}
\end{aligned} \tag{5.35}$$

e as derivadas das funções denominador

$$\begin{aligned}
\frac{\partial H}{\partial x} &= 2h \frac{\partial h}{\partial x} \\
\frac{\partial H}{\partial y} &= 2h \frac{\partial h}{\partial y} \\
\frac{\partial h}{\partial x} &= \sum_k \frac{\partial w_k}{\partial x} \\
\frac{\partial h}{\partial y} &= \sum_k \frac{\partial w_k}{\partial y}
\end{aligned} \tag{5.36}$$

A fim de calcular estas derivadas, vale introduzir a função característica

$$\mathbb{1}_{\mathcal{A}}(t) = \begin{cases} 1 & \text{se } t \in \mathcal{A} \\ 0 & \text{caso contrário.} \end{cases} \tag{5.37}$$

onde $\mathcal{A}$ é um conjunto. As duas funções introduzidas abaixo servirão para deixar a

notação menos carregada:

$$\begin{aligned}\Gamma_{ij}(\vec{x}) &= 1 - \frac{(x-c_x^{ij})^2+(y-c_y^{ij})^2}{r_{ij}^2} \\ \chi_{ij}(\vec{x}) &= \mathbb{1}_{[0,1]}\left(\frac{\|\vec{x}-\vec{c}_{ij}\|}{r_{ij}}\right)\end{aligned}$$

e cabe ressaltar que as coordenadas do centro $\vec{c}_{ij}$ se escrevem

$$\vec{c}_{ij} = [\ c_x^{ij}\ \ c_y^{ij}\ ].$$

Assim, seguem as derivadas da função peso:

$$\begin{aligned}
\frac{\partial w_{ij}}{\partial x}(\vec{x}) &= -\frac{8}{r_{ij}^2}(x-c_x^{ij})\left[\Gamma_{ij}(\vec{x})\right]^3\ \chi_{ij}(\vec{x}) \\
\frac{\partial w_{ij}}{\partial y}(\vec{x}) &= -\frac{8}{r_{ij}^2}(y-c_y^{ij})\left[\Gamma_{ij}(\vec{x})\right]^3\ \chi_{ij}(\vec{x}) \\
\frac{\partial^2 w_{ij}}{\partial x \partial y}(\vec{x}) &= \frac{64}{r_k^4}(x-c_x^{ij})(y-c_y^{ij})\left[\Gamma_{ij}(\vec{x})\right]^2\ \chi_{ij}(\vec{x}) \\
\frac{\partial^2 w_{ij}}{\partial x^2}(\vec{x}) &= \left(\frac{64}{r_{ij}^4}(x-c_x^{ij})^2\left[\Gamma_{ij}(\vec{x})\right]^2 - \frac{8}{r_{ij}^2}\ \left[\Gamma_{ij}(\vec{x})\right]^3\right)\chi_{ij}(\vec{x}) \\
\frac{\partial^2 w_{ij}}{\partial y^2}(\vec{x}) &= \left(\frac{64}{r_{ij}^4}(y-c_y^{ij})^2\left[\Gamma_{ij}(\vec{x})\right]^2 - \frac{8}{r_{ij}^2}\ \left[\Gamma_{ij}(\vec{x})\right]^3\right)\chi_{ij}(\vec{x})
\end{aligned} \tag{5.38}$$

5.4 Projetando na superfície PUI

Seja

$$\begin{cases} f: & \mathbb{R}^m & \rightarrow & \mathbb{R}^n \\ & \mathbf{x} & \mapsto & f(\mathbf{x}) \end{cases} \tag{5.39}$$

uma função de classe $\mathcal{C}^1$. Enunciaremos a seguir o método de Newton Generalizado, no intuito de projetar um ponto $\mathbf{x_0} \in \mathbb{R}^m$ no lugar geométrico definido por

$$\{\mathbf{x} \mid f(\mathbf{x}) \equiv \mathbf{0}\}. \tag{5.40}$$

Seja $M = \mathbf{J}(f,\mathbf{x}) \in \mathcal{M}_{m\times n}(\mathbb{R})$ a matriz jacobiana de $f(\cdot)$ no ponto $\mathbf{x}$ e $M^+ = [\mathbf{J}(f,\mathbf{x})]^+ \in \mathcal{M}_{n\times m}(\mathbb{R})$ sua *pseudo-inversa no sentido de Moore-Penrose.* Isto é, satisfaz as propriedades abaixo

(i) $MM^+M = M$

(ii) $M^+MM^+ = M^+$

(iii) $(MM^+)^T = MM^+$

(iv) $(M^+M)^T = M^+M$

em que as propriedades (iii) e (iv) tratariam do conjugado M^* no caso complexo (exigindo assim que MM^+ e M^+M fossem hermitianas, em adição a serem simétricas).

O *Método de Newton Generalizado* é um método iterativo para encontrar zeros de funções, sendo o iterador definido por, dado um ponto inicial $\mathbf{x}_i$, uma função f e a matriz $[\mathbf{J}(f,\mathbf{x})]^+$ tais quais definidos acima:

$$\begin{aligned} \mathbf{x}_0 &= \mathbf{x}_i \\ \mathbf{x}_{k+1} &= \mathbf{x}_k - [J(f,\mathbf{x}_k)]^+ \cdot f(\mathbf{x}_k) \end{aligned} \tag{5.41}$$

No caso de interesse, em que a função PUI é da forma

$$\begin{cases} P: & \mathbb{R}^2 & \to & \mathbb{R} \\ & \mathbf{x} & \mapsto & P(\mathbf{x}) \end{cases} \tag{5.42}$$

e a matriz jacobiana é o vetor gradiente

$$\mathbf{J}(P,\mathbf{x}) = \left[\frac{\partial}{\partial x} P(\mathbf{x}) \, , \frac{\partial}{\partial y} P(\mathbf{x}) \right] = \vec{\nabla} P \Big|_{\mathbf{x}} \tag{5.43}$$

observa-se que a pseudo-inversa é a matriz coluna

$$[\mathbf{J}(P,\mathbf{x})]^{+} = \frac{(\vec{\nabla} P)^T}{\|\vec{\nabla} P\|^2} \Big|_{\mathbf{x}} \tag{5.44}$$

que satisfaz todas as propriedades (i-iv).

5.5 Implementação

Nesta seção, utilizando os conceitos elaborados acima, serão dados os passos para obter a representação da superfície. Estaremos considerando como dados fornecidos pelo problema para cada início de ciclo computacional:

- Uma nuvem de pontos $\mathcal{P} = \{\vec{x}_k\}_{k \in K}$, indexada por um conjunto K;
- O conjunto das normais $\mathcal{N} = \{\vec{n}_k\}_{k \in K}$, respeitando a indexação K (i.e. $\vec{n}_k$ é normal de $\vec{x}_k$);
- Uma família de células $\mathcal{C}_{ij}$ indexadas por $(i,j) \in I$, contendo partículas da interface ($\mathcal{C}_{ij} \cap \mathcal{P} \neq \emptyset$) e uma matriz `mat[i][j]` de `double` para armazenar os sistemas locais de referência $\mathcal{R}_{ij}$ a serem calculados ao longo do ciclo, exceto os centro $\vec{c}_{ij}$ que são conhecidos e fixos ao longo de toda malha euleriana - conforme indica (5.3) - e os raios r_{ij} que são pré-definidos pelo usuário;

dadas algumas observações: o conjunto das normais $\mathcal{N}$ não precisa ser fornecido como condição inicial do problema, no entanto à partir do segundo ciclo computacional estará presente por ter sido calculada usando o item (ii) na página 67 e podendo ser utilizada doravante pelo procedimento descrito no item (i); a fim de evitar ambiguidades ressaltamos que a matriz `mat[i][j]` não é o coeficiente m_{ij} de uma matriz $\texttt{mat} = \left(m_{ij}\right)_{ij}$ e sim uma matriz $m^{ij} = (m^{ij}_{kl})_{kl} \in \mathcal{M}_{m\times n}$ referente ao sistema de referência local da célula $\mathcal{C}_{ij}$;

Uma consideração importante: todas as matrizes $A = A(\mathcal{P})$ e todos os vetores $b = b(\mathcal{P})$ tais que a solução do sistema linear

$$A \cdot \underline{\alpha} = b \tag{5.45}$$

fornece um vetor coluna $\underline{\alpha}$, parâmetro de uma função $F_{\underline{\alpha}}(\cdot)$ que minimiza

$$\sum_{k\in K} \|F_{\underline{\alpha}}(\vec{x}_k)\|^2 \tag{5.46}$$

ou que minimiza

$$\sum_{k\in K} \|y_k - F_{\underline{\alpha}}(x_k)\|^2 \ , \tag{5.47}$$

são calculados com contribuições independentes de cada $\vec{x}_k$, ou seja, $A = A(\mathcal{P})$ e $b = b(\mathcal{P})$ podem ser escritos da forma:

$$\begin{array}{rcl} A(\mathcal{P}) & = & A_0 \ + \ \sum_k \mathcal{A}(\vec{x}_k) \\ b(\mathcal{P}) & = & b_0 \ + \ \sum_k \mathcal{B}(\vec{x}_k) \end{array} \tag{5.48}$$

em que A_0, b_0 são constantes e $\mathcal{A}$,$\mathcal{B} : \mathbb{R}^2 \to \mathbb{R}$ são funções, referidas como *incrementadoras*. Denotaremos por $\mathcal{A}_{\texttt{esf}}$ (resp. $\mathcal{A}_{\texttt{pol}}$) a função que incrementa o sistema linear para obtenção do parâmetro que ajusta uma esfera (resp. ajusta o polinômio

da representação local) a totalidade dos pontos inseridos na incrementação - nos casos considerados, serão pontos de uma vizinhança.

Finalmente, as informações a seguir servirão para aumentar a clareza do algoritmo dado no final desta seção, além de estabelecer critérios e notações para deixá-lo mais compacto:

- A rotina `indice`$(\vec{x})$ retorna o índice (i,j) da célula $\mathcal{C}_{ij}$ que contém o ponto $\vec{x}$.

- Será denotado por M^{ij} o espaço de memória de `mat`$[i][j]$ utilizado para armazenar a matriz do sistema linear, tanto para ajuste de esferas quanto de polinômios;

- As matrizes `mat`$[i][j]$ alocadas para cada célula $\mathcal{C}_{ij}$ são variáveis globais, prescindindo estarem presentes nas chamadas de rotinas que as utilizam;

- A rotina `slin_esf`$(i,j,\vec{x})$ incrementa a matriz do sistema linear que aproxima pontos por esfera; mais especificamente, é o procedimento reponsável pela incrementação

$$M^{ij} \leftarrow M^{ij} + \mathcal{A}_{\texttt{esf}}(\vec{x}) \tag{5.49}$$

  e analogamente a rotina `slin_pol`$(i,j,\vec{x})$ realiza a incrementação:

$$M^{ij} \leftarrow M^{ij} + \mathcal{A}_{\texttt{pol}}([\vec{x}]_{(i,j)}) \tag{5.50}$$

  em que $\vec{x}$ é passado como parâmetro segundo as coordenadas do sistema de eixos locais da célula $\mathcal{C}_{ij}$, dada a natureza da aproximação da função local conforme explicitado em (5.2);

- A rotina `sol_slin_esf`(i,j,A,b) (resp. `sol_slin_pol`) encontra a solução do sistema linear (5.45) e armazena a solução na matriz `mat`$[i][j]$ na posição designada do parâmetro da esfera (resp. do parâmetro do polinômio da representação local);

- A rotina `calc_norm_esf`(i,j) calcula a normal da célula $\mathcal{C}_{ij}$ segundo (5.6), admitindo-se que o parâmetro (5.4) já tenha sido calculado;

- A rotina `P`(i,j,x) calcula $P(\vec{x})$ usando(5.15), segundo (5.29);

- A rotina `dxp`(i,j,x) calcula $\frac{\partial}{\partial x}P(\vec{x})$ usando(5.15), segundo (5.30);

- A rotina `dyp`(i,j,x) calcula $\frac{\partial}{\partial y}P(\vec{x})$ usando(5.15), segundo (5.31);

- A rotina `dxxp`(i,j,x) calcula $\frac{\partial^2}{\partial x^2}P(\vec{x})$ usando(5.15), segundo (5.32);

- A rotina `dyyp`(i,j,x) calcula $\frac{\partial^2}{\partial y^2}P(\vec{x})$ usando(5.15), segundo (5.33);

- A rotina `dxyp`(i,j,x) calcula $\frac{\partial^2}{\partial x \partial y}P(\vec{x})$ usando(5.15), segundo (5.34);

- A rotina `w`(i,j,x) calcula $P(\vec{x})$ usando(5.15);

- A rotina `dxw`(i,j,x) calcula $\frac{\partial}{\partial x}w_{ij}(\vec{x})$;

- A rotina `dyw`(i,j,x) calcula $\frac{\partial}{\partial y}w_{ij}(\vec{x})$;

- A rotina `dxxw`(i,j,x) calcula $\frac{\partial^2}{\partial x^2}w_{ij}(\vec{x})$;

- A rotina `dyyw`(i,j,x) calcula $\frac{\partial^2}{\partial y^2}w_{ij}(\vec{x})$;

- A rotina `dxyw`(i,j,x) calcula $\frac{\partial^2}{\partial x \partial y}w_{ij}(\vec{x})$;

As funções do tipo peso mencionadas nos últimos itens assim como suas derivadas são obtidas em (5.38).

ALGORITMO

Calculando as Representações Locais

Inicializando as matrizes alocadas:

for $(i,j) \in I$ **do**

$m^{ij} \leftarrow \mathbf{0}$

end for

Calculando o sistema linear das esferas locais:

for $k \in K$ **do**

$(i,j) \leftarrow$ `indice`$(\vec{x}_k)$

for $(p,q) \in [i-3,i+3] \times [j-3,j+3]$ **do**

if $\big((p,q) \in I\big) \wedge \big(\|\vec{c}_{pq} - \vec{x}_k\| < r_{pq}\big)$ **then**

`slin_esf`$(p,q,\vec{x}_k)$

end if

end for

end for

Resolvendo os sistemas lineares das esferas locais, calculando as normais das células e reinicializando as matrizes dos sistemas lineares M^{ij}:

for $(i,j) \in I$ **do**

`sol_slin_esf`(i,j,A,b)

`calc_norm_esf`(i,j)

$M^{ij} \leftarrow \mathbf{0}$

end for

Calculando o sistema linear dos polinômios locais:

for $k \in K$ **do**

$(i,j) \leftarrow$ `indice`$(\vec{x}_k)$

for $(p,q) \in [i-3,i+3] \times [j-3,j+3]$ **do**

if $\big((p,q) \in I\big) \wedge \big(\|\vec{c}_{pq} - \vec{x}_k\| < r_{pq}\big)$ **then**

`slin_pol`$(p,q,\vec{x}_k)$

end if

end for

end for

Resolvendo os sistemas lineares dos polinômios locais:

for $(i,j) \in I$ **do**

`sol_slin_esf`(i,j,A,b)

end for

Calculando $D^{\alpha}P(\vec{x})$

Dado um ponto $\vec{x}$ e havendo calculados todos sistemas de referência $\mathcal{R}_{ij}$, $\forall (i,j) \in I$, o procedimento abaixo calcula `P` $= P(\vec{x})$ *e todas as suas derivadas parciais até segunda ordem:*

Inicializando variáveis:

$(i,j) \leftarrow$ `indice`$(\vec{x})$

`h` $\leftarrow 0$,`hx` $\leftarrow 0$,`hy` $\leftarrow 0$

`hxx` $\leftarrow 0$,`hyy` $\leftarrow 0$,`hxy` $\leftarrow 0$

`g` $\leftarrow 0$,`gx` $\leftarrow 0$,`gy` $\leftarrow 0$

`gxx` $\leftarrow 0$,`gyy` $\leftarrow 0$,`gxy` $\leftarrow 0$

for $(p,q) \in [i-3,i+3] \times [j-3,j+3]$ **do**

if $\|\vec{x} - \vec{c}_{pq}\| < r_{pq}$ **then**

Calcula valor e derivadas da representação local (5.13):

`phi`,`phix`,`phiy`,`phixx`,`phiyy`,`phixy`.

Calculando H (Denominador da PUI)

$\texttt{h} \leftarrow \texttt{h} + \texttt{w}(p,q,\vec{x})$

$\texttt{hx} \leftarrow \texttt{hx} + \texttt{dxw}(p,q,\vec{x})$

$\texttt{hy} \leftarrow \texttt{hy} + \texttt{dyw}(p,q,\vec{x})$

$\texttt{hxx} \leftarrow \texttt{hxx} + \texttt{dxxw}(p,q,\vec{x})$

$\texttt{hyy} \leftarrow \texttt{hyy} + \texttt{dyyw}(p,q,\vec{x})$

$\texttt{hxy} \leftarrow \texttt{hxt} + \texttt{dxyw}(p,q,\vec{x})$

Calculando G (Numerador da PUI)

$\texttt{g} \leftarrow \texttt{g} + \texttt{w} * \texttt{phi}$

$\texttt{gx} \leftarrow \texttt{gx} + \texttt{w} * \texttt{phix} + \texttt{wx} * \texttt{phi}$

$\texttt{gy} \leftarrow \texttt{gy} + \texttt{w} * \texttt{phiy} + \texttt{wy} * \texttt{phi}$

$\texttt{gxx} \leftarrow \texttt{gxx} + \texttt{wxx} * \texttt{phi} + 2 * \texttt{wx} * \texttt{phix} + \texttt{w} * \texttt{phixx}$

$\texttt{gyy} \leftarrow \texttt{gyy} + \texttt{wyy} * \texttt{phi} + 2 * \texttt{wy} * \texttt{phiy} + \texttt{w} * \texttt{phiyy}$

$\texttt{gxy} \leftarrow \texttt{gxy} + \texttt{wxy} * \texttt{phi} + \texttt{wx} * \texttt{phiy} + \texttt{wy} * \texttt{phix} + \texttt{w} * \texttt{phixy}$

end if

end for

if $\texttt{h} > 0$ **then**

Calculando $P(\vec{x})$:

$\texttt{P} \leftarrow \texttt{g}/\texttt{h}$

Cálculos intermediarios

$\texttt{Gx} \leftarrow (\texttt{dgdx} * \texttt{h} - \texttt{g} * \texttt{dhdx})$, $\texttt{Gy} \leftarrow (\texttt{dgdy} * \texttt{h} - \texttt{g} * \texttt{dhdy})$

$\texttt{H} \leftarrow \texttt{h} * \texttt{h}$, $\texttt{Hx} \leftarrow 2 * \texttt{h} * \texttt{hx}$, $\texttt{Hy} \leftarrow 2 * \texttt{h} * \texttt{hy}$

Calculando $DP(\vec{x})$:

$\texttt{Px} \leftarrow Gx/H$

$\texttt{Py} \leftarrow Gy/H$

Calculos intermediarios

$\texttt{Gxx} \leftarrow \texttt{h} * \texttt{gxx} - \texttt{g} * \texttt{hxx}$, $\texttt{Gyy} = \texttt{h} * \texttt{gyy} - \texttt{g} * \texttt{hyy}$,

$\texttt{Gxy} \leftarrow \texttt{h} * \texttt{gxy} + \texttt{hy} * \texttt{gx} - \texttt{hx} * \texttt{gy} - \texttt{g} * \texttt{hxy}$

Calculando D^2P

$\texttt{Pxx} \leftarrow (\texttt{H} * \texttt{dGxdx} - \texttt{Gx} * \texttt{dHdx})/(\texttt{H} * \texttt{H})$

$\texttt{Pyy} \leftarrow (\texttt{H} * \texttt{dGydy} - \texttt{Gy} * \texttt{dHdy})/(\texttt{H} * \texttt{H})$

$\texttt{Pxy} \leftarrow (\texttt{H} * \texttt{dGxdy} - \texttt{Gx} * \texttt{dHdy})/(\texttt{H} * \texttt{H})$

end if

5.6 Apêndice: mínimos quadrados com imposição de volume

Nesta seção deduziremos as contas do que chamaremos de *polinômio volumétrico*. Dada uma nuvem de pontos $\mathcal{P} = \{(x_k,y_k)\}_{k\in K}$ queremos determinar um polinômio p tal que sua integral seja numericalmente igual ao volume formado pela nuvem de pontos com

relação ao eixo das abcissas. Isto é, queremos $p \in \mathbb{P}_n(\mathbb{R})$

$$p(x) = \sum_{j=0}^{n} \alpha_j \, x^j \tag{5.51}$$

com a imposição

$$\int p(x)\, dx = \mathcal{V}(\{\mathcal{P}\}) \tag{5.52}$$

em que $\mathcal{V}(\{\mathcal{P}\})$ é o volume formado por $\{(x_k,y_k)\}$, e.g. pela fórmula do trapézio

$$\mathcal{V}(\{\mathcal{P}\}) = \sum_{k\in K} \frac{1}{2}(x_{k+1} - x_k)(y_{k+1} + y_k) \tag{5.53}$$

Desenvolvendo o lado esquerdo de (5.52)

$$\int p(x)\, dx = \sum_{j=0}^{n} \alpha_j \int_t^T x^j \, dx = \sum_{j=0}^{n} \alpha_j \left[\frac{x^{j+1}}{j+1} \right]_t^T ,$$

estabelecendo $V := \mathcal{V}(\mathcal{P})$ e definindo $\Delta_k = T^k - t^k$, obtem-se:

$$\sum_{j=0}^{n} \alpha_j \, \frac{\Delta_{k+1}}{j+1} = V \, . \tag{5.54}$$

Isolando o parâmetro α_0 da equação (5.54)

$$\alpha_0 = \left[V - \sum_{j=1}^{n} \alpha_j \, \frac{\Delta_{k+1}}{j+1} \right] (\Delta_1)^{-1}, \tag{5.55}$$

Para tornar a notação mais leve definimos:

$$A_j(x) = x^j - \frac{\Delta_{j+1}}{(j+1)\,(T-t)} = x^j - \frac{\Delta_{j+1}}{(j+1)\,\Delta_1}, \tag{5.56}$$

de maneira que para $x = 0$

$$A_j(0) = -\frac{\Delta_{j+1}}{(j+1)\,\Delta_1}$$

e substituindo em (5.55)

$$\alpha_0 = \frac{V}{\Delta_1} + \sum_{j=1}^{n} \alpha_j \; A_j(0). \tag{5.57}$$

Assim, ao substituirmos (5.57) em (5.51) estamos impondo a condição (5.52):

$$\begin{aligned} p(x) = \alpha_0 + \sum_{j=1}^{n} \alpha_j \, x^j &= \frac{V}{\Delta_1} + \sum_{j=1}^{n} \alpha_j \; A_j(0) + \sum_{j=1}^{n} \alpha_j \, x^j \\ = \frac{V}{\Delta_1} + \sum_{j=1}^{n} \alpha_j \, [x^j + \; A_j(0)] &= \frac{V}{\Delta_1} + \sum_{j=1}^{n} \alpha_j \; A_j(x) \end{aligned}$$

e o problema se traduz então em minimizar a função de energia abaixo

$$\begin{aligned} J(\underline{\alpha}) &= \sum_{k \in K} (y_k - p(x_k))^2 \\ &= \sum_{k \in K} \left(y_k - \frac{V}{\Delta_1} - \sum_{j=1}^{n} \alpha_j \; A_j(x_k) \right)^2 \end{aligned} \tag{5.58}$$

que por sua vez se traduz em encontrar $\underline{\alpha}$, solução de

$$\frac{\partial}{\partial \alpha_i} J(\underline{\alpha}) = 0.$$

Notando que o termo $y_k - \dfrac{V}{\Delta_1}$ independe de $\underline{\alpha}$, o cálculo da derivada de J segue

$$\frac{\partial}{\partial \alpha_i} J(\underline{\alpha}) = \sum_{k \in K} \left(y_k - \frac{V}{\Delta_1} - \sum_{j=1}^{n} \alpha_j \; A_j(x_k) \right) \; (-2) \, A_i(x_k).$$

temos igualando a expressão do lado direito a 0 (zero) com alguma manipulação algébrica:

$$\sum_{k \in K} \left(y_k - \frac{V}{\Delta_1} \right) \; A_i(x_k) = \sum_{k \, , j} \alpha_j \; A_j(x_k) \, A_i(x_k)$$

e como os α_j não dependem de k, podemos escrever

$$\sum_{k\in K}\left(y_k - \frac{V}{\Delta_1}\right) A_i(x_k) = \sum_{j=1}^{n}\alpha_j \sum_{k=1}^{N\ Pts} A_j(x_k)\, A_i(x_k)$$

para $i = 1,2,\cdots,n$, o sistema linear cuja solução fornece o parâmetro α procurado.

Capítulo 6

Resultados

Neste capítulo serão apresentados os testes realizados em que se projeta os pontos na superfície formada pela Partição da Unidade Implícita. O capítulo se divide em quatro seções: a primeira com testes preliminares que permite visualizar como a partição da unidade atua na reconstrução da superfície global a partir de representações locais, além de mostrar como o método de polinômios volumétricos (discutido no apêndice do capítulo do projeto) são boas aproximações locais no que tange a conservação do volume, mas deixa de sê-lo quando englobado pela partição da unidade; a segunda seção mostra a aproximação por PUI nos casos de superfície sem movimento, alterando certos parâmetros (como o número de subcélulas e a extensão do raio de ação) e dá a ordem de convergência da aproximação; a terceira seção considera o movimento rígido e a quarta seção analisa a qualidade das simulações do *Freeflow2D* quando projeta os pontos da inteface na superfície PUI.

6.1 Preliminares

Esta seção mostra o funcionamento da PUI em um âmbito preliminar, querendo dizer que não prima pela validação do código, prima em expor a maneira como trata a

superfície e os resultados visuais do processo envolvido.

6.1.1 Análise do polinômio volumétrico

Antes de tudo, começamos por mostrar os resultados que levaram a descartar a idéia do polinômio volumétrico descrito na seção 5.6, uma idéia que tem por objetivo concentrar-se na conservação de massa. Como notação, φ_v se refere ao polinômio volumétrico enquanto φ_g é aquele obtido pelos mínimos quadrados conforme descrito em 2.6.1.

O polinômio φ_g é obtido fixando-se o parâmetro do termo constante de tal forma que ao aproximar o conjunto de pontos $\mathcal{P} = \{(x_k,y_k)\}_k$ esta imposição resulte em:

$$\int_{x_m}^{x_M} \varphi_g(x)\ dx = \mathcal{V}(\mathcal{P}) \tag{6.1}$$

em que $\mathcal{V}(\cdot)$ calcula o volume trapezoidal da nuvem de pontos e

$$\begin{aligned} x_m &= \min_k \{x_k\} \\ x_M &= \max_k \{x_k\}. \end{aligned} \tag{6.2}$$

A figura 6.1 mostra a ação do polinômio de conservação de volume φ_v no caso em que há tão poucos pontos disponíveis que φ_g coincide com o polinômio que os interpola. De fato, o cenário de baixa amostragem é aquele onde há maior divergência de comportamento entre φ_v e φ_g conforme veremos. Observa-se, no entanto, que apesar de φ_g efetivamente respeitar melhor a imposição dos mínimos quadrados, o polinômio φ_v define uma curva cuja integral calcula um volume mais próximo do volume original.

Não obstante, constatamos que ao aumentarmos a quantidade de pontos da amostra, o polinômio φ_v tende ao polinômio φ_g.

Rodando

$N_{\texttt{itera}} \leftarrow 100$

for $N_{\texttt{pts}} = 6 \rightarrow 20$ **do**

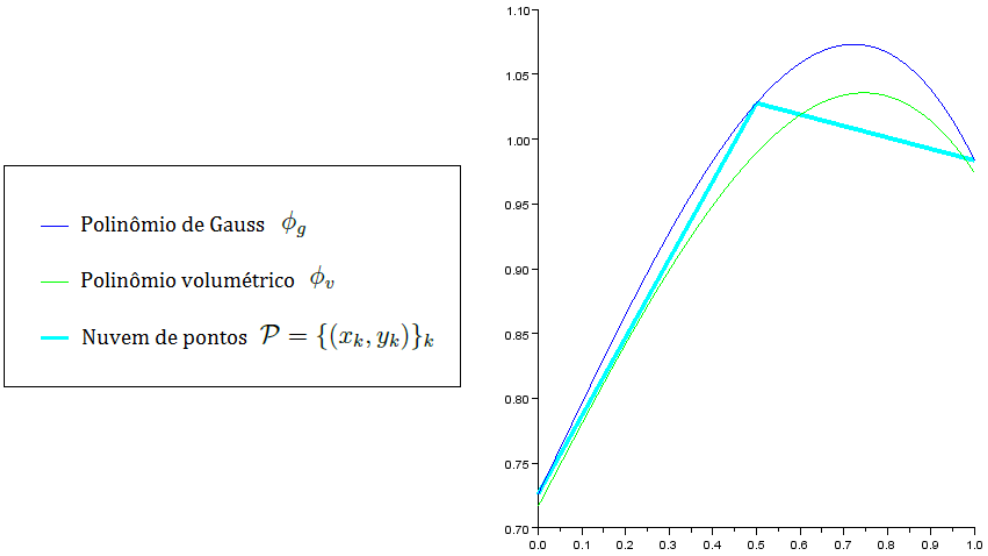


Figura 6.1: Os pontos gerados aleatoriamente definem a curva em ciano, a curva em azul é obtida pelos mínimos quadrados convencionalmente e a curva em verde prima pela conservação do volume.

$d_m \leftarrow 0$

for $i = 1 \rightarrow N_{\texttt{itera}}$ **do**

$[\varphi_v, \varphi_g, d] = \texttt{rand\_pts}(N_{\texttt{pts}})$

$d_m \leftarrow d_m + d$

end for

$d_m \leftarrow d_m / N_{\texttt{iter}}$

Imprime d_m (cf. tabela 6.1)

end for

em que `randpts(N_pts)` gera $N_{\texttt{pts}}$ pontos distríbuidos entre [0,1] e os aproxima por polinomios φ_g (Gauss) e φ_v (volumétrico), que por sua vez são passados como resposta no final da rotina, junto com a distânica $d = d(\varphi_g,\varphi_v)$ entre eles, observa-se que ao passo que aumentamos a distribuição de pontos os polinômios distam cada vez menos entre si. Os resultados impressos de d_m estão dispostos na tabela 6.1, a figura 6.2 mostra uma série de aproximações feitas usando o algoritmo acima e ao aumentar $N_{\texttt{pts}}$

Tabela 6.1: Convergência no sentido $\|\varphi_g - \varphi_v\| \xrightarrow{N_{\texttt{pts}} \to 0} 0$.

Número de pontos($N_{\texttt{pts}}$)	Distância média(d_m)
6	0.0365982
7	0.0359748
8	0.0316452
9	0.0302314
10	0.0228842
11	0.0278375
12	0.0227615
13	0.0258454
14	0.0192709
15	0.0183202
16	0.0172339
17	0.0177548
18	0.0185347
19	0.0167338
20	0.0124837

podemos notar que os gráficos de φ_g (em azul) e φ_v (em verde) se aproximam.

Para fins de comparação, vamos considerar que cada sistema de referência $\mathcal{R}_{ij}$ local contenha duas aproximações locais distintas:

$$\mathcal{R}_{ij} \;=\; \{\; \vec{c}_{ij} \;,\; r_{ij} \;,\; \varphi^g_{ij} \;,\; \varphi^v_{ij} \;\} \tag{6.3}$$

com um polinômio φ^g_{ij} obtido pelo método dos mínimos quadrados de Gauss e o outro φ^v_{ij} sendo o polinômio obtido fixando um parâmetro para conservar o volume. Por haver duas representações locais, a comparação se estende a duas superfícies $\mathcal{S}_g = \{\vec{x} \mid P_g(\vec{x}) = 0\}$ e $\mathcal{S}_v = \{\vec{x} \mid P_v(\vec{x})\}$ geradas por duas funções partição da unidade, a saber:

$$\begin{aligned} P_g(\vec{x}) &= \sum_{(i,j)\in I} w_{ij}(\vec{x})\Big(\eta_{ij}(\vec{x}) - \varphi^g_{ij} \circ \xi_{ij}(\vec{x})\Big) \\ P_v(\vec{x}) &= \sum_{(i,j)\in I} w_{ij}(\vec{x})\Big(\eta_{ij}(\vec{x}) - \varphi^v_{ij} \circ \xi_{ij}(\vec{x})\Big) \end{aligned} \tag{6.4}$$

em que as funções w_{ij} do tipo peso são definidas segundo (5.19), ξ_{ij} e η_{ij} passam $\vec{x}$ para coordenadas locais $\vec{\eta}$.

Assim, é possível porque o polinômio φ_g calcula melhor volume que o polinômio φ_v. O exemplo da elipse (que será retomado na próxima seção) mostra esta situação na prática: em uma dada célula $\mathcal{C}$ observamos a diferença entre o volume calculado pelos polinômios.

A célula $\mathcal{C}$ em questão tinha por volume

$$\mathcal{V}(\mathcal{C} \cap \mathcal{S}) = -0.449674$$

e, nas coordenadas locais, encontramos $x_m = -0.7377785$ e $x_M = 0.6371347$. A figura 6.3 auxilia a visualização dos volumes que serão calculados a seguir. O polinômio volumétrico calculado foi:

$$\varphi_v(x) = -0.294295 + 0.0003132x - 0.2045788x^2 \tag{6.5}$$

e observa-se que

$$\int_{x_m}^{x_M} \varphi_v(x)\, dx = -0.449674 \tag{6.6}$$

ou seja, como pretendíamos, a integral do polinômio coincidiu com o volume inicial. Entretanto, ao calcular o volume da nuvem de pontos que passa por esse polinômio obtemos:

$$\mathcal{V}(\{x_k, \varphi_v(x_k)\}) = -0.4504160. \tag{6.7}$$

O mesmo procedimento aplicado ao polinômio φ_g forneceu:

$$\int_{x_m}^{x_M} \varphi_g(x)\, dx = -0.4489354 \tag{6.8}$$

diferindo do volume inicial. Porém, ao considerarmos o volume dos pontos passando

pelo gráfico de φ_g

$$\mathcal{V}(\{x_k, \varphi_g(x_k)\}) = -0.4496791 \tag{6.9}$$

observamos que o resultado final é um polinômio que preserva melhor a propriedade geométrica da superfície. Isto acontece porque a imposição (6.1) se dá sobre a fórmula da integral analítica do polinômio volumétrico. Esta imposição, apesar de efetivamente gerar um polinômio que conserva o volume ao ser integrado, não o conserva pela fórmula de quadratura (6.7). Esta imprecisão será conduzida a uma má aproximação da superfície $\mathcal{S}_v$, que conforme mencionado acima, priorizava respeitar a geometria dos pontos de controle $\mathcal{P}$, o que encerra a discussão.

6.1.2 Ação da função peso na representação local

O próximo tópico é observar de perto o funcionamento da partição da unidade. A função PUI se escreve

$$P(x,y) = \frac{\sum_{(i,j)\in I} W_{ij}(x,y) f_{ij}(x,y)}{\sum_k W_{ij}(x,y)} \tag{6.10}$$

com W_{ij} definido em (5.20), podemos observar inicialmente apenas a aproximação referente a uma célula apenas, isto é, uma função

$$f_0(x,y) w_0(x,y)$$

e estudar o comportamento de seu gráfico.

Conforme podemos observar temos a função ϕ na figura (6.6) e a função peso w_k na figura 6.7. O produto ϕw_k mostra o efeito da função peso na multiplicação, conforme ilustra a figura 6.8: próximo do centro, onde a função peso vale 1, o produto se comporta como o polinômio; e na extremidade onde a função peso suaviza até valer 0, também podemos ver o produto das funções fazer o mesmo.

Se levarmos em consideração todas as representações polinomiais locais, obteremos uma superfície conforme mostrada na figura (6.9). Para analizar quantitativamente as derivadas da função partição da unidade, vale considerar o numerador da função:

$$P(\vec{x}) = \frac{\sum_k \phi_k(\vec{x}) w_k(\vec{x})}{\sum_k w_k(\vec{x})} =: \frac{g(\vec{x})}{h(\vec{x})} \tag{6.11}$$

e lembrar do resultado de cálculo diferencial em que a derivada de uma função f num ponto x em que é positiva (resp. negativa) se f é crescente (resp. decrescente) em uma vizinhança de x. Uma seção transversal da função g é analisada na figura (1.10) e confirmada na figura (1.11).

Assim, cabem três observações ao olharmos para o gráfico de $D^n P$ e $D^n g$:

- ficar atento à coerência geométrica da derivada
- perceber que no caso de P, espera-se que suas derivadas remetam à derivadas dos polinômios de menor grau
- entender como varia a ação de diferentes raios de ação (i.e., o tamnanho do suporte da função peso);

E como podemos bem visualizar na figura 6.12, quanto menor a influencia da função peso, mais característico fica o polinômio local. No entanto, embora representações locais são mais precisas localmente, um raio de ação muito curto implica numa queda brusca nos valores da função, e se tomados pequenos demais, provoca picos muito altos em suas derivadas.

Em se tratando de observar como a função peso atua na reconstituição da superfície, a figura 6.4 mostra uma circunferência (em vermelho) e uma superfície PUI (em azul) com representações locais afins. Nesta imagem, pela natureza grosseira da aproximação - seja pela escolha de polinômios do primeiro grau, seja pela baixa quantidade de representações locais (células demasiadas grandes) - pode-se ver a maneira como a Partição

da Unidade liga duas representações contíguas. A figura 6.5 mostra o que acontece quando se aumenta a quantidade de representações locais e mesmo com representações locais afins a superfície reconstruída fica nitidiamente mais próxima da original.

6.2 Superfície sem movimento

Esta seção dedica ao teste da função partição da unidade visando observar somente propriedades de caráter geométrico da superfície, isto é,

- A ordem de convergência da distância
- A ordem de convergência da normal
- A ordem de convergência da curvatura

Dada uma propriedade geométrica $\mathcal{G}(x,y)$ obtida em um ponto (x,y) do domínio, dizemos que uma família de aproximações $\{\mathcal{G}_h\}_h$ é convergente de ordem k para uma norma $\|\cdot\|$ se

$$\|g - g_h\| = \mathcal{O}(h^k) \tag{6.12}$$

quando $h \longrightarrow 0$. Assim sendo, considerando o erro $\epsilon_h = \|g - g_h\|$ para um refinamento h do problema e fazendo

$$\epsilon_h \approx h^k \tag{6.13}$$

obtemos mediante manipulação algébrica:

$$\ln(\epsilon_h) = \ln(h^k) = k\ln(h).$$

A distância entre superfícies considerada é a distância de Hausdorff, e é definida entre duas superfícies $\mathcal{S}$ e $\mathcal{S}'$ como:

$$d(\mathcal{S},\mathcal{S}') = \max\left(\max_{\mathbf{x}\in\mathcal{S}}\min_{\mathbf{y}\in\mathcal{S}'}\|\mathbf{x}-\mathbf{y}\|, \max_{\mathbf{x}\in\mathcal{S}}\max_{\mathbf{y}\in\mathcal{S}'}\min\|\mathbf{x}-\mathbf{y}\|\right) \tag{6.14}$$

Se $\mathcal{S}$ denota a superfície original e $\mathcal{S}'$ a superfície calculada pela PUI, para cada $\vec{x} \in \mathcal{S}'$ consideramos o ponto $\vec{y}_x$ de $\mathcal{S}$ mais próximo de x. Assim, o erro das normais é calcualdo segundo

$$E^N_\infty(\mathcal{S},\mathcal{S}') = \max_{x\in\mathcal{S}'}\|\vec{n}(\vec{x}) - \vec{n}(\vec{y}_x)\| \tag{6.15}$$

$$E^N_k(\mathcal{S},\mathcal{S}') = \left(\int_{x\in\mathcal{S}'}\left(\|\vec{n}(\vec{x}) - \vec{n}(\vec{y}_x)\|_{L^k}\ dx\right)^k\right)^{1/k}, \quad k = 1,2. \tag{6.16}$$

Para o erro da curvatura, em que $\kappa'(\cdot)$ é a função que retorna a curvatura segundo a superfície PUI e $\kappa(\cdot)$ é a curvatura da superfície real, temos :

$$E^C_\infty(\mathcal{S},\mathcal{S}') = \max_{x\in\mathcal{S}'}\|\kappa'(\vec{x}) - \kappa(\vec{y}_x)\| \tag{6.17}$$

$$E^C_k(\mathcal{S},\mathcal{S}') = \left(\int_{x\in\mathcal{S}'}\left(\|\kappa'(\vec{x}) - \kappa(\vec{y}_x)\|_{L^k}\ dx\right)^k\right)^{1/k}, \quad k = 1,2. \tag{6.18}$$

Assim, procederemos a série de testes de duas maneiras: primeiro construiremos no modelador do *Freeflow* superfícies com uma partícula por célula, alterando a malha euleriana a cada vez de tal forma a se obter maior número de células (e portanto, de representações polinomiais); em seguida, construiremos um exemplo a partir do qual aumentaremos a quantidade de subcéluas, havendo inicialmente distribuído partículas em abundância.

No primeiro caso, o raio de ação do polinômio em cada célula terá que ser grande o bastante para incluir pelo menos $deg(p) + 1$ pontos, dado que queremos uma representação polinomial p de grau $deg(p)$. Como a superfície é construída para ter uma

partícula por célula, isto implica em obter funções peso com suporte intersectando uma grande região do suporte de outras funções peso das células vizinhas (cf. figura 6.13).

Examinando a convergência para o caso da elipse dada por

$$\frac{x^2}{9} + \frac{y^2}{4} = 1 \tag{6.19}$$

tomando o espaçamento igual à $\Delta = 1{,}0.5{,}0.25, \cdots, 2^{-6}$ (isto é, o tamanho das células).

Para primeira série de testes (uma partícula por célula) temos tabela 6.2 que fornece os erros da normal, para cada Δ, da superfície calculada por diferenças finitas e da PUI calculada explicitamente. Vale ressaltar que a estrutura da tabela é da forma

Δ	E
$\cdots$	$\cdots$
	Erro segundo $DP\big\|_{\vec{x}}$ Erro segundo $\frac{P(x+h)-P(x-h)}{2h}$
$\cdots$	$\cdots$

ou seja, para cada tipo de erro ($E = E_1, E_2$ ou E_∞), a cada par de linhas (separadas pelas divisórias), a primeira linha mostra o respectivo erro calculado segundo os valores de DP, implementados diretamente à partir das fórmulas na seção 5.3 e a segunda linha segundo a fórmula de diferenças finitas, também implementados. Também vale notar que o valor da discretização tomado foi $h = 10^{-5}$ e que os resultados obtidos são muito próximos.[1]

[1] Isto é de particular interesse para adicionar novas rotinas, pois não é necessário calcular analiticamente as derivadas, somente inserir os cálculos das novas representações locais ou funções de tipo peso (ou ambas as duas).

Tabela 6.2: Erro da normal para uma partícula por célula.

Δ	E_∞^N	E_1^N	E_2^N
1	0.077571096458	0.017690805327	0.025392165266
	0.077571096473	0.017690805334	0.025392165266
0.5	0.009653192748	0.002783106414	0.003458601453
	0.009653192762	0.002783106420	0.003458601453
0.25	0.004501135418	0.000822068241	0.001049599294
	0.004501135409	0.000822068244	0.001049599294
0.125	0.001182132859	0.000205731660	0.000254434304
	0.001182132892	0.000205731666	0.000254434304

Tabela 6.3: Erro da curvatura para uma partícula por célula.

Δ	E_∞^C	E_1^C	E_2^C
1	0.286260654891	0.083777673585	0.125785427842
	0.286260646284	0.083777673289	0.125785425361
0.5	0.163015710126	0.047169865413	0.064378346877
	0.163015710400	0.047169865847	0.064378347307
0.25	0.098660408877	0.020860896775	0.034819510660
	0.098660409339	0.020860896874	0.034819510805
0.125	0.061913132059	0.008808851132	0.013570457946
	0.061913133341	0.008808851300	0.013570458214

A tabela 6.3 fornece os dados de convergência da curvatura e assim como foi feito para o cálculo da normal, temos duas maneiras de se calcular, a saber, usando as fórmulas analíticas da seção 5.3 e por diferenças finitas.

Em seguida construímos uma elipse com grande abundância de pontos para poder calcular funções PUI variando o número de subcélulas (ou seja, o k), o raio do suporte das funções peso e o grau do polinômio sem ter que ser preocupar com a quantidade de pontos na vizinhança de cada célula (a notar que cada subcélula $\mathcal{C}(i,j)^k$ deve conter pelo menos $n+1$ pontos em $supp(w^k_{(i,j)})$ para calcular um polinômio de grau n). Para

o tamanho do raio, escolhemos

$$r = \alpha\sqrt{(\Delta x)^2 + (\Delta y)^2} \tag{6.20}$$

e alteramos nos exemplos a seguir o tamanho do parâmetro α no interesse de encontrar qual corresponde à melhor escolha do parâmetro.

Os exemplos que seguem, conforme observamos os dados de convergência, mostram que o que mais define a superfície é a quantidade de representações locais que se dispõe, isto é, grande abundância de pontos não melhoram a aproximação local se não há maior quantidade de polinômios na representação local.

As tabelas 6.4, 6.5, 6.6, 6.7, 6.8, 6.9, 6.10, 6.11 e 6.12 fornecem dados da convergência das distâncias, normais e curvaturas alterando α. Destas tabelas observa-se um princípio interessante, já anunciado na seção anterior: quanto menor o raio de ação, maiores os picos na derivada e o que se ganha nas representações locais (por incluir na vizinhança apenas pontos muito próximos) se perde na precisão das derivadas, dado que a função peso tem suporte muito restrito.

As tabelas 6.13 mostra o cálculo do erro relativo do volume da superfície na estrutura inerente do *Freeflow2D*, isto é, a estrutura de dados *half-edge2D* e ao lado, a título de comparação, temos o erro relativo do volume calculado na superfície PUI, ligeiramente mais próximos do volume calculado analiticamente.

As tabelas 6.14 e 6.15 calculam erros de distâncias, normais e curvaturas utilizando um outro tipo de função peso, chamadas de Wendland. Trata de tomar para cada célula $\mathcal{C}_{ij}$ a função

$$w_{ij} = \frac{\widetilde{W}_{ij}(\vec{x})}{\sum\limits_{(p,q)\in I} \widetilde{W}_{pq}(\vec{x})} \tag{6.21}$$

Tabela 6.4: Convergência da distância usando $\alpha = 0.55$.

Δ	$d(\mathcal{S}_1,\mathcal{S}_2)$
1	$8.835817490569 \times 10^{-3}$
0.5	$5.76778840403 \times 10^{-4}$
0.25	$7.3919640780 \times 10^{-5}$
0.125	$9.999054897 \times 10^{-6}$
0.0625	$1.408296267 \times 10^{-6}$
0.031250	$2.86557067 \times 10^{-7}$
0.015625	3.6303352×10^{-8}

com $W_{ij}(\vec{x}) = \widetilde{\theta}(\|\vec{x} - \vec{c}_{ij}\|)$ e

$$\theta(t) = (1-t)^4(t+1). \tag{6.22}$$

Tabela 6.5: Convergência da normal usando $\alpha = 0.55$.

Δ	E_∞^N	E_1^N	E_2^N
1	0.052956184155	0.009470506386	0.011939762317
	0.052956184154	0.009470506425	0.011939762317
0.5	0.009510102241	0.001885092313	0.002117134395
	0.009510102515	0.001885092322	0.002117134395
0.25	$3.504439685 \times 10^{-3}$	$4.85393557 \times 10^{-4}$	$5.84629379 \times 10^{-4}$
	$3.504439973 \times 10^{-3}$	$4.85393560 \times 10^{-4}$	$5.84629379 \times 10^{-4}$
0.125	$6.96468592 \times 10^{-4}$	$1.29072479 \times 10^{-4}$	$1.42636759 \times 10^{-4}$
	$6.96468892 \times 10^{-4}$	$1.29072482 \times 10^{-4}$	$1.42636759 \times 10^{-4}$
0.0625	$2.02425523 \times 10^{-4}$	2.8381437×10^{-5}	3.2993044×10^{-5}
	$2.02425797 \times 10^{-4}$	2.8381437×10^{-5}	3.2993044×10^{-5}
0.031250	6.2951644×10^{-5}	7.972840×10^{-6}	9.599875×10^{-6}
	6.2951978×10^{-5}	7.972842×10^{-6}	9.599875×10^{-6}
0.015625	1.6368606×10^{-5}	2.005128×10^{-6}	2.269762×10^{-6}
	1.6368575×10^{-5}	2.005131×10^{-6}	2.269762×10^{-6}

Tabela 6.6: Convergência da curvatura usando $\alpha = 0.55$.

Δ	E_∞^C	E_1^C	E_2^C
1	2.886474880785	0.108135497307	0.398750317532
	2.886474956658	0.108135498592	0.398750327490
0.5	0.710569399703	0.034742356819	0.082540902935
	0.710569490346	0.034742357925	0.082540910439
0.25	0.159052943123	0.013271715452	0.023120803077
	0.159053029335	0.013271716338	0.023120808738
0.125	0.050152899195	0.006926563874	0.012037128649
	0.050152924231	0.006926567555	0.012037141273
0.062500	0.030296435904	0.003220805197	0.005556011769
	0.030296518716	0.003220808390	0.005556018918
0.031250	0.016554909352	0.001572846080	0.002545145375
	0.016556109315	0.001572849998	0.002545162079
0.015625	0.016582355066	0.000859266369	0.001484980781
	0.016585467754	0.000859274977	0.001485028259

Tabela 6.7: Convergência da distância para $\alpha = 0.6$.

Δ	$d(\mathcal{S},\mathcal{S}')$
1	0.006041747307708
0.5	0.000688954752442
0.25	0.000073092571424
0.125	0.000010635571400
0.0625	0.000001206428440
0.03125	0.000000175107298
0.015625	0.000000032297708

Tabela 6.8: Convergência da normal e curvatura para $\alpha = 0.6$.

Δ	E_8^N	E_1^N	E_2^N
1	0.078296523883	0.012294252466	0.014174629831
0.5	0.014485417488	0.002898614739	0.003155088174
0.25	0.003048514949	0.000589203374	0.000627937432
0.125	0.000906611800	0.000150336023	0.000168853051
0.0625	0.000302457010	0.000036569079	0.000040302330
0.03125	0.000103024315	0.000009457279	0.000010433710
0.015625	0.000047216715	0.000002404938	0.000002664256
Δ	E_8^C	E_1^C	E_2^C
1	2.024011923710	0.077479388393	0.164223199317
0.5	0.839617476391	0.034084918377	0.067672418876
0.25	0.082632735387	0.013029323892	0.020166517436
0.125	0.055605148248	0.006874625100	0.011478901532
0.0625	0.044693594474	0.003307952127	0.005406379418
0.03125	0.052271218763	0.001731677411	0.003064927576
0.015625	0.060263666294	0.000889524259	0.001747227696

Tabela 6.9: Convergência da distância para $\alpha = 0.65$.

Δ	$d(\mathcal{S},\mathcal{S}')$
1	0.008795827846022
0.5	0.000972090826568
0.25	0.000071991120359
0.125	0.000010033516615
0.0625	0.000001872626942
0.03125	0.000000256818028
0.015625	0.000000038534959

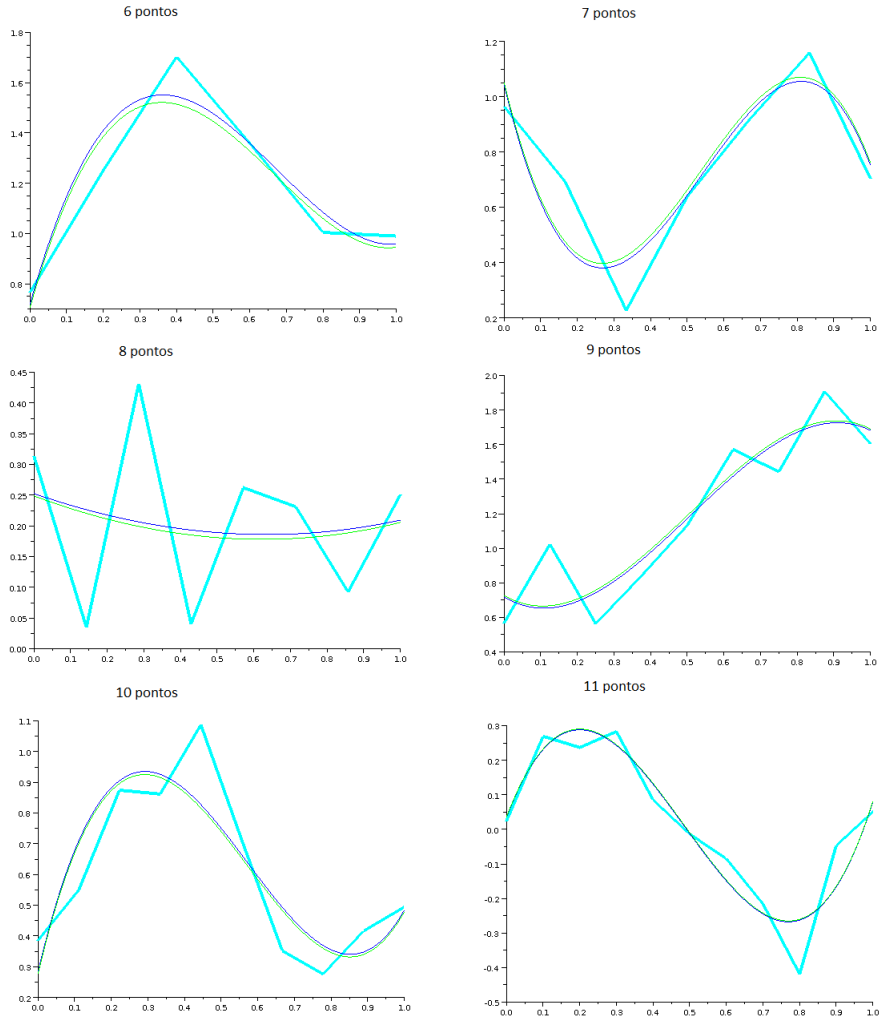
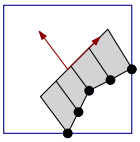
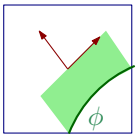


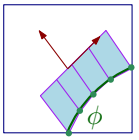
Figura 6.2: Pontos aleatoriamente distribuídos que quando em abundância fazem com que $p_v \to p_g$.



$\mathcal{V}(\mathcal{P})$: volume formado pelos pontos da interface dados *a priori*.



$\int_{x_m}^{x_M} \phi(x)dx$: calculado analiticamente, respectivo ao volume entre abcissa e a curva ϕ local.



$\mathcal{V}(\{x_k, \phi(x_k)\}_k)$: volume trapezoidal de pontos passando por $\{(x, \phi(x))\}$

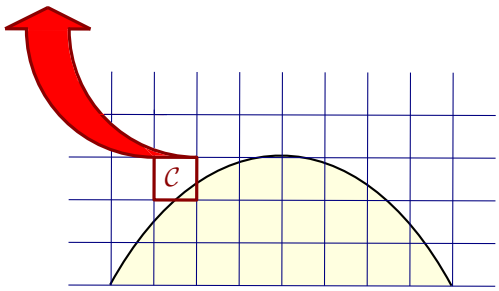


Figura 6.3: Três volumes calculados dada uma aproximação local.

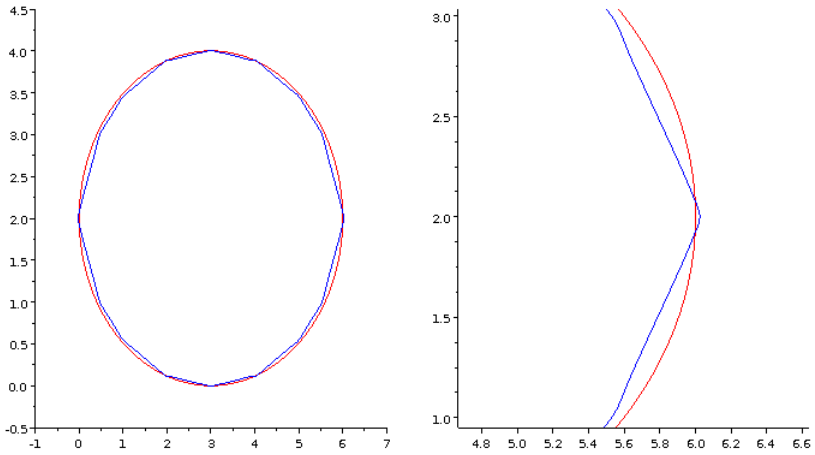


Figura 6.4: PUI construída com polinômios afins.

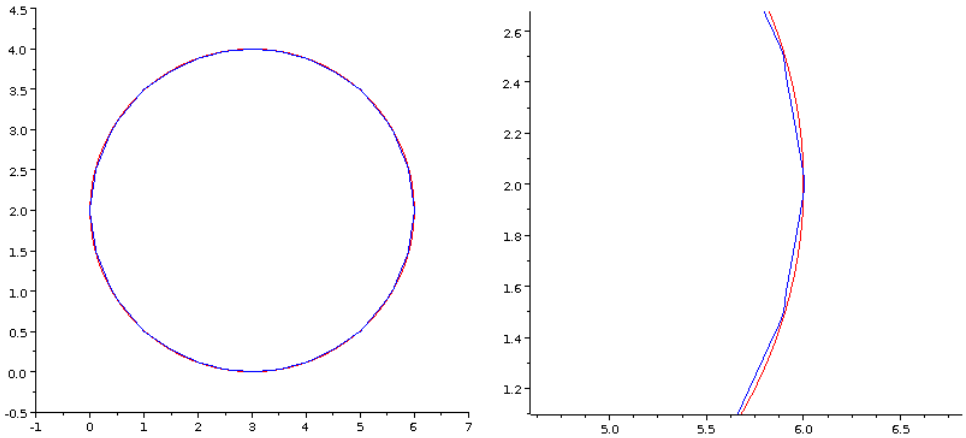


Figura 6.5: PUI construída com polinômios afins com maior refinamento, a superfície aproximada fica distintamente mais próxima da superfície original.

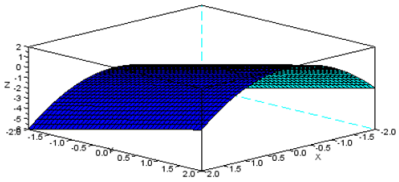


Figura 6.6: Polinômio Local

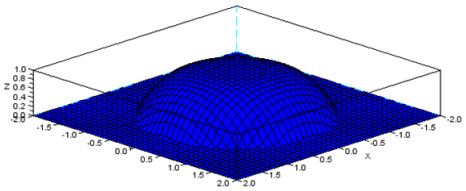


Figura 6.7: Função Peso

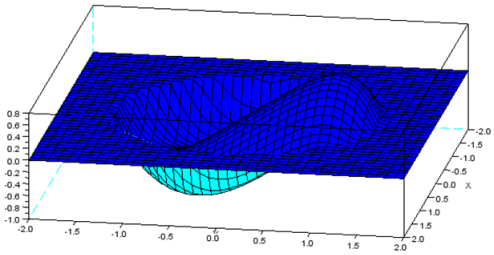


Figura 6.8: Função Peso multiplica o polinômio

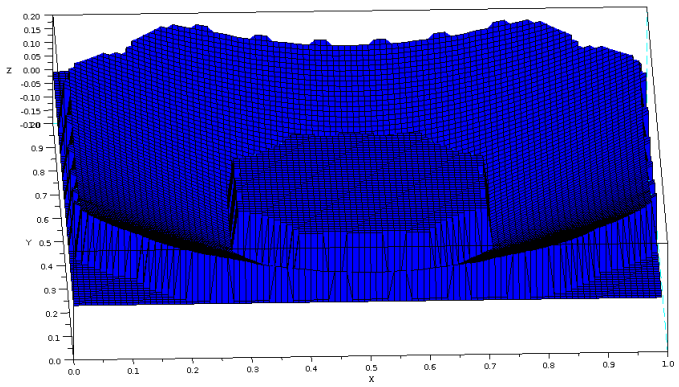


Figura 6.9: Imagem da PUI.

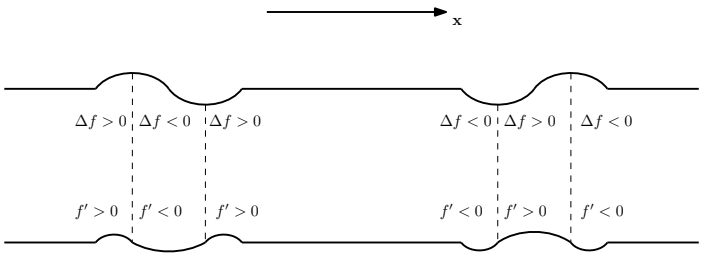


Figura 6.10: Visualizando o comportamento esperado da derivada da função.

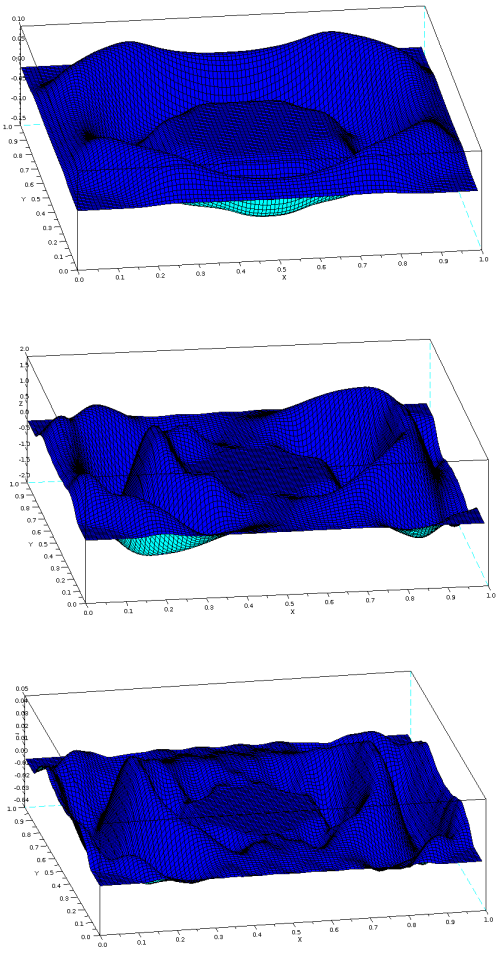


Figura 6.11: Funções g, $\partial g/\partial x$ e $\partial^2 g/\partial x^2$.

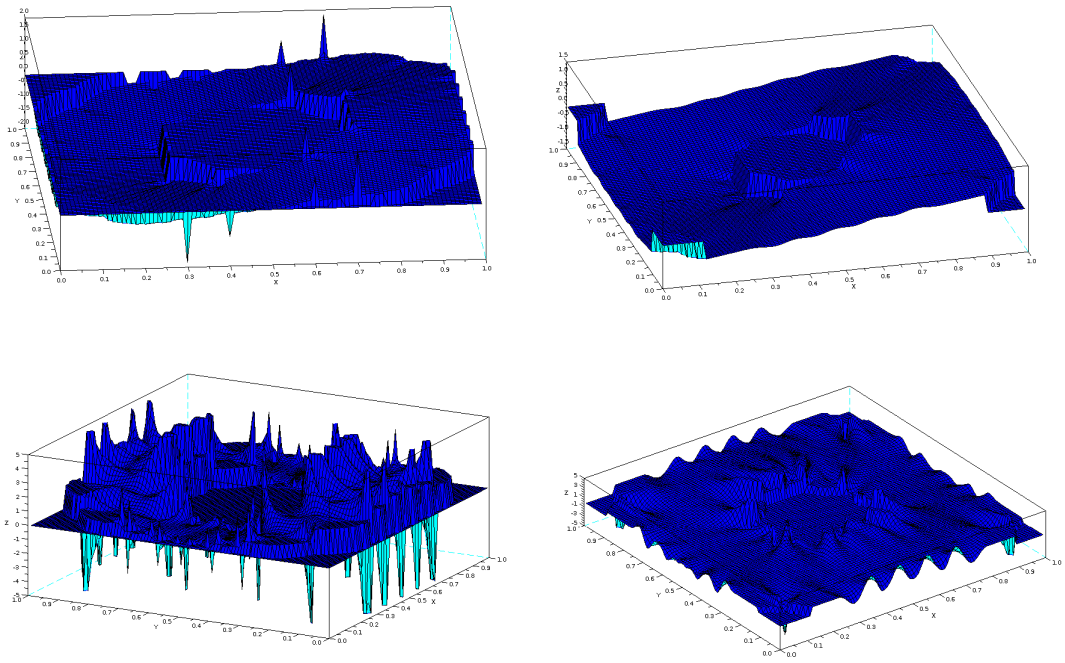


Figura 6.12: Derivadas de $P(\vec{x})$: na coluna da direita o raio de ação é pequeno demais, criando picos na derivada; na coluna da esquerda, o raio sendo maior, provoca menos oscilações na derivadas.

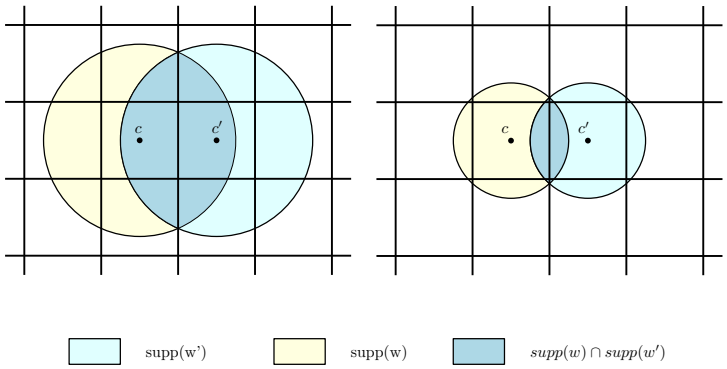


Figura 6.13: Intersecção do suporte das funções peso diminiu conforme diminuimos o raio de ação.

Tabela 6.10: Convergência da normal e curvatura para $\alpha = 0.65$.

Δ	E_8^N	E_1^N	E_2^N
1	0.051650729187	0.014776187719	0.015922973379
0.5	0.015585237688	0.003102285833	0.003317864802
0.25	0.003080020469	0.000620321871	0.000653957603
0.125	0.000947564236	0.000156214850	0.000169256241
0.0625	0.000295944162	0.000041974627	0.000045491641
0.03125	0.000097610105	0.000010652327	0.000011752843
0.015625	0.000023227184	0.000002676426	0.000002910468
Δ	E_8^C	E_1^C	E_2^C
1	0.804348254621	0.073323707424	0.113196887768
0.5	0.177576302085	0.028466762093	0.044023710967
0.25	0.084627267161	0.013239606066	0.019892192280
0.125	0.051200106222	0.006690294681	0.010519384722
0.0625	0.030580524743	0.003508585986	0.005370341167
0.03125	0.019543674829	0.001775194508	0.002827990461
0.015625	0.008937915166	0.000898181575	0.001408941608

Em seguida conferimos para este valor de raio a tabela 6.16, com polinômios de grau 3 na representação local e a tabela 6.17, com polinômios de grau 4 na representação local.

Podemos observar a curva de convergência na escala logaritma para o polinômio de grau 2 usando um raio de $r = 0.6\sqrt{\Delta x^2 + \Delta y^2}$, dispostos na figura 6.14. Os resultados apontam para uma ordem de convergência $\mathcal{O}(\Delta^3)$ para a geometria, $\mathcal{O}(\Delta^2)$ para as normais e $\mathcal{O}(\Delta)$ para a curvatura.

Considerando por último o quadrado de dimensão

$$[-0.25,0.25] \times [-0.25,0.25],$$

sob um refinamento de malha $\Delta x = \Delta y = 0.1$ e observando a convergência da PUI conforme aumentamos o número de divisão de células $k = 0,1,3,7$, podemos ver a aproximação de superfície convergindo para o quadrado na figura 6.15.

Tabela 6.11: Convergência da distância para $\alpha = 0.7$.

Δ	$d(\mathcal{S},\mathcal{S}')$
1	0.011803455725427
0.5	0.000941264450305
0.25	0.000141072234500
0.125	0.000013777870990
0.0625	0.000001829642250
0.03125	0.000000265961066
0.015625	0.000000044194626

Tabela 6.12: Convergência da normal e curvatura para $\alpha = 0.7$.

Δ	E_8^N	E_1^N	E_2^N
1	0.063136719147	0.017897740894	0.019742940015
0.5	0.015710244205	0.003629027700	0.003764326512
0.25	0.004673690706	0.000841030325	0.000926589496
0.125	0.000850321396	0.000175312140	0.000185388103
0.0625	0.000322759288	0.000044831861	0.000048470576
0.03125	0.000097882796	0.000011625666	0.000012780520
0.015625	0.000024776289	0.000002857909	0.000003130526
Δ	E_8^C	E_1^C	E_2^C
1	0.288917435054	0.070715315175	0.096507952320
0.5	0.181253660791	0.030489759951	0.046386125153
0.25	0.079637510534	0.015954163817	0.023678295285
0.125	0.034920020696	0.006840913750	0.010101853871
0.0625	0.022894078756	0.003510549869	0.005136772668
0.03125	0.016154127913	0.001804015618	0.002746812500
0.015625	0.008484924917	0.000896656433	0.001359999962

Tabela 6.13: Convergência e comparação do volume.

Δ	Half-edge	PUI
1	0.040973298263630	0.039406685334274
0.5	0.010064547884909	0.009959836699419
0.25	0.002516472932648	0.002482794422711
0.125	0.000629180857949	0.000626605148856
0.0625	0.000157307015165	0.000156997441700
0.03125	0.000039286198977	0.000039280623167
0.015625	0.000009882325635	0.000009881888538

Tabela 6.14: Convergência da distância usando funções peso de Wendland.

Δ	$d(\mathcal{S},\mathcal{S}')$
1	0.006041747307708
0.5	0.000698343684974
0.25	0.000073119580502
0.125	0.000010722417995
0.0625	0.000001235944581
0.03125	0.000000177554947
0.015625	0.000000032406948

Tabela 6.15: Convergência da normal e curvatura para o peso de Wendland.

Δ	E_8^N	E_1^N	E_2^N
1	0.078396692175	0.012393758138	0.014345146071
0.5	0.015552796809	0.002917283708	0.003198012175
0.25	0.003321934608	0.000595814730	0.000642080011
0.125	0.000990682414	0.000152146034	0.000173241507
0.0625	0.000310243864	0.000037092091	0.000041242086
0.03125	0.000106021022	0.000009570302	0.000010654306
0.015625	0.000050543995	0.000002436574	0.000002722778
Δ	E_8^C	E_1^C	E_2^C
1	2.156354802134	0.079999375116	0.170522477665
0.5	0.923041478626	0.035313771939	0.071783051044
0.25	0.088600484396	0.013582778441	0.021581713845
0.125	0.061615965949	0.007270691543	0.012368764344
0.0625	0.048486922924	0.003474489914	0.005795835572
0.03125	0.05889969652	0.001818960611	0.003298652857
0.015625	0.067597572449	0.000934785236	0.001881492549

Tabela 6.16: Tabela de erros para polinômio de grau 3.

Δ	$d(\mathcal{S}_1,\mathcal{S}_2)$	E_∞^N	E_∞^C
$k=0$	0.001541845522831	0.013239363523	0.092154011052
$k=1$	0.000080884402075	0.001447145780	0.028620616382
$k=3$	0.000004966926414	0.000174494651	0.006676620556

Tabela 6.17: Tabela de erros para polinômio de grau 4.

Δ	$d(\mathcal{S}_1,\mathcal{S}_2)$	E_∞^N	E_∞^C
$k=0$	0.000305489735454	0.002951945644	0.054911967109
$k=1$	0.000007268869337	0.000169095575	0.004746208552
$k=3$	0.000000206600676	0.000009536538	0.000510799413

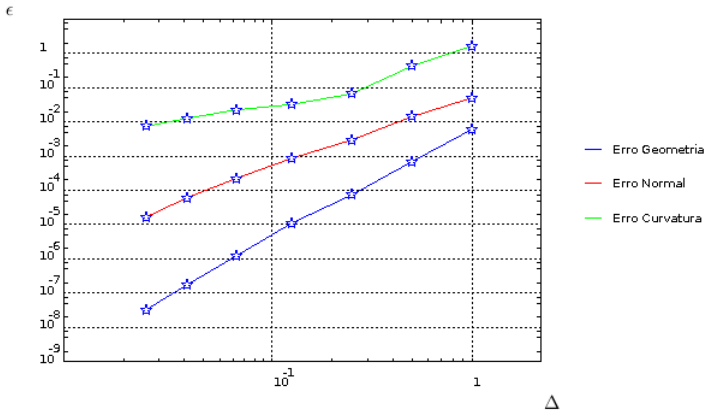


Figura 6.14: Gráfico da convergência das propriedades da esfera na aproximação com polinômio de grau 2 e raio $\alpha = 0.6$.

6.3 Movimento Rígido

Primeiro precisamos definir um campo de velocidades sobre uma malha deslocada (Cf. figura 6.16). O valor da velocidade $u_{i+\frac{1}{2},j}$ está armazenado no coeficiente de uma matriz, digamos:

```
u[i][j]
```

e seu valor deve ser $V_x(x_i + \frac{\Delta x}{2}, y_j)$. Analogamente, O valor da velocidade $v_{i,j+\frac{1}{2}}$ está armazenado no campo de uma matriz:

```
v[i][j]
```

e seu valor deve ser $V_y(x_i, y_j + \frac{\Delta y}{2})$, em que

$$\mathbf{V}(x,y) = \Big(V_x(x,y), V_y(x,y)\Big) \tag{6.23}$$

é o campo vetorial da velocidade do escoamento, definido sobre o domínio Ω.

Neste caso queremos rotacionar um quadrado imerso no domínio $\Omega = [0,1] \times [0,1]$, tendo por centro o ponto $(\frac{1}{2},\frac{1}{2})$ e de aresta $\ell = 0.5$, criado no *Modflow2d* conforme vemos na figura 6.17.

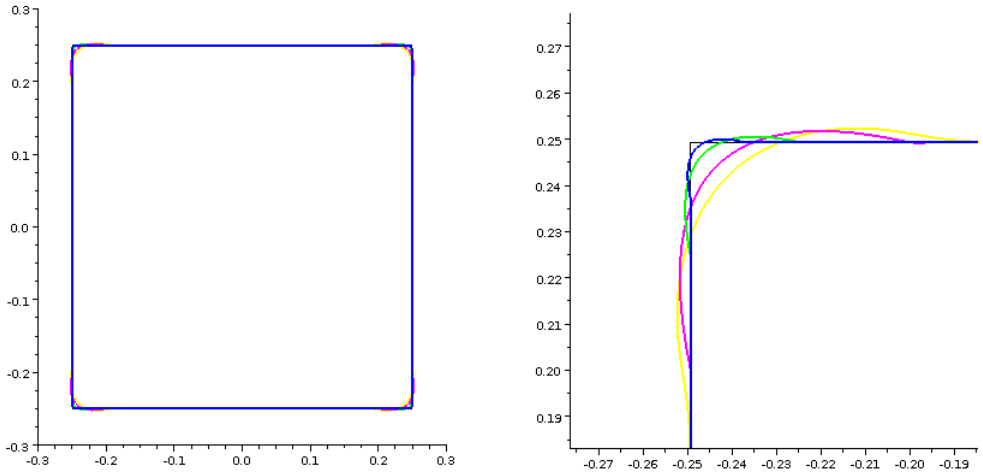


Figura 6.15: Superfície se aproxima do quadrado conforme aumentam o número de subcélulas: $k = 0$ (amarelo), $k = 1$ (roxo), $k = 3$ (verde) e $k = 7$ (azul). A superfície original está em preto.

Temos uma malha discreta Γ que aproxima o domínio Ω

$$\Gamma = \{(x_i,y_j) \mid \forall(i,j)\}, \tag{6.24}$$

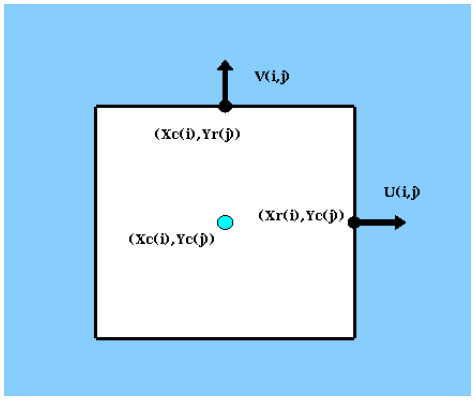


Figura 6.16: Representação visual da célula deslocada

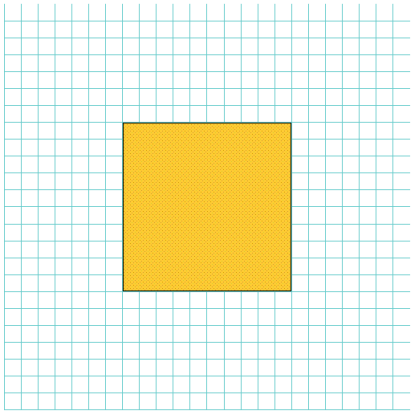


Figura 6.17: Fluido em forma de cubo no *Modflow2d*.

o vetor velocidade $\mathbf{V}(x,y)$ dado pela fórmula acima e `fx` é a matriz $[f_x^{(i,j)}]_{(i,j)}$ em que os coeficientes satisfazem

$$V_x(x_i,y_j) = f_x^{(i,j)} \tag{6.25}$$

e analogamente `fy` é a matriz $[f_y^{(i,j)}]_{(i,j)}$ cujos os coeficientes satisfazem

$$V_y(x_i,y_j) = f_y^{(i,j)} \tag{6.26}$$

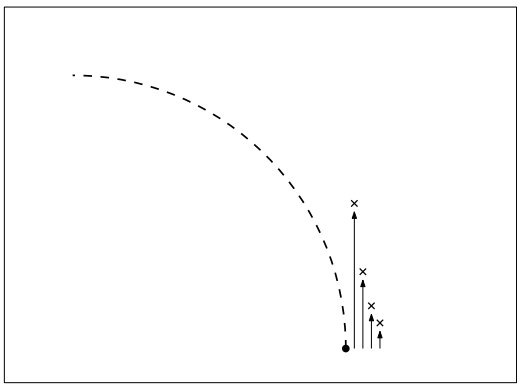


Figura 6.18: Mesmo tomando a direção correta na direção da tangente, a discretização no tempo acarreta no desvio da trajetória original.

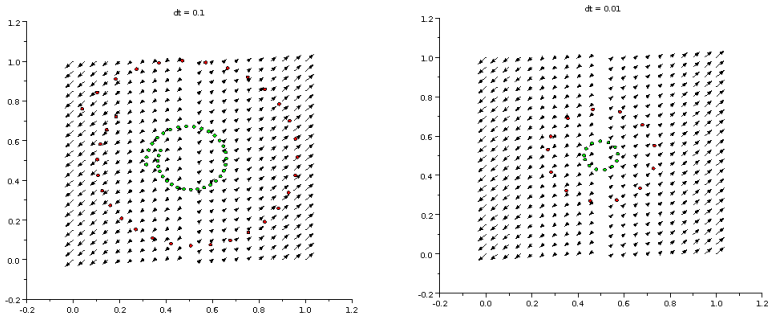


Figura 6.19: Distorção da trajetória com aumento de Δt.

Um bom candidato a campo de velocidades para rotacionar o quadrado em torno de seu centro $\vec{c}$ é

$$\mathbf{V}(\vec{x}) = \alpha \, R_{\pi/2} \times \left(\vec{x} - \vec{c}\right) \tag{6.27}$$

em que $R_{\pi/2}$ é a matriz de rotação de $\frac{\pi}{2}$ radianos e α é alguma constante.

O passo de tempo não pode ser tomado livremente pois senão acontece conforme indicam as figuras 6.19 e 6.18: o campo de velocidade atuante distoa de uma rotação poiso vetor tangente que forneceria uma velocidade angular no ponto (x,y) deixa de agir em um pequeno instante Δt e coloca o ponto em outra "órbita". O passo sobre o vetor tangente quando o tempo é contínuo tem valor infinitesimal, logo, para qualquer discretização estaremos lidando com algum ganho de volume. Na figura 6.19 visualizamos duas trajetórias de duas partículas, uma verde e outra vermelha, ambas desviam da órbita teórica por causa da discretização do tempo. Este desvio de trajetória é menos expressivo ao tomarmos um passo de tempo Δt menor. A figura 6.18 mostra como não importa a ordem de magnitude da discretização no tempo, pois caminhando na direção tangente da circunferência em um passo de tempo discreto levará a partícula à um ponto de chegada (sinalizado pela cruz) respectivo à outra órbita.

Queremos que ao cabo de um número N_i de iterações o ponto (inicial) $\vec{x} = \vec{x}_0$ seja levado à sua rotação de $\pi/2$, isto é, o ponto $x_{N_i} = R_{\pi/2} \cdot \vec{x}_0$. Assim, consideraremos o

campo de velocidade seguinte:

$$\begin{cases} \mathbf{V}: & \Omega & \longrightarrow & \mathbb{R}^2 \\ & \vec{x}=(x,y) & \mapsto & (\alpha\Delta t)\vec{\tau}(\vec{x}) \end{cases} \tag{6.28}$$

em que $\vec{\tau}(\vec{x})$ é o vetor tangente à circunferência centrada em $\vec{c}$ (o centro do quadrado) e passando por $\vec{x}$, dado explicitamente por

$$\vec{\tau}(\vec{x}) = R_{\pi/2}\cdot\Big(\vec{x}-\vec{c}\Big). \tag{6.29}$$

O caminho percorrido Δs, pela fórmula do comprimento de arco é

$$\Delta s = \frac{\pi r}{2} \tag{6.30}$$

e como o comprimento do vetor tangente $\|\vec{\tau}(\vec{x})\| = \|\vec{x}-\vec{c}\|$ é justamente o raio r, temos

$$\boxed{\alpha = \tfrac{\Delta S}{n\Delta t\|\tau\|} = \tfrac{\pi}{2n\Delta t}} \tag{6.31}$$

de tal forma que no tempo $t = n\Delta t$ (ou analogamente na n-ésima iteração), teremos percorrido

$$\begin{aligned} \textstyle\sum_{k=1}^{n}\|(\alpha\Delta t)\vec{\tau}(\vec{x}_k)\| &= \textstyle\sum_{k=1}^{n}\left(\tfrac{\pi}{2n\Delta t}\right)\Delta t\|\tau(\vec{x}_k)\| = \tfrac{\pi}{2n}\textstyle\sum_{k=1}^{n}\overbrace{\|\tau(\vec{x}_k)\|}^{r} \\ &= \tfrac{\pi nr}{2n} = \tfrac{\pi r}{2} = \Delta s \end{aligned} \tag{6.32}$$

O cálculo das componentes $u_{i+\frac{1}{2},j}$ e $v_{i,j+\frac{1}{2}}$ segue abaixo:

$$\begin{aligned}
\mathbf{V}(x_{ij}^{(r)}, y_{ij}^{(c)}) &= \alpha \begin{bmatrix} 0 & 1 \\ \\ -1 & 0 \end{bmatrix} \begin{bmatrix} -x_{ij}^{r} + c_x \\ \\ y_{ij}^{c} - c_y \end{bmatrix} \\
&= \begin{bmatrix} \alpha(y_{ij}^{(c)} - c_y) \\ \alpha(c_x - x_{ij}^{(r)}) \end{bmatrix} = \begin{bmatrix} V_x(x_{ij}^{(r)}, y_{ij}^{(c)}) \\ V_y(x_{ij}^{(r)}, y_{ij}^{(c)}) \end{bmatrix} \\
\mathbf{V}(x_{ij}^{(c)}, y_{ij}^{(r)}) &= \alpha \begin{bmatrix} 0 & 1 \\ \\ -1 & 0 \end{bmatrix} \begin{bmatrix} -x_{ij}^{c} + c_x \\ \\ y_{ij}^{r} - c_y \end{bmatrix} \\
&= \begin{bmatrix} \alpha(y_{ij}^{r} - c_y) \\ \alpha(c_x - x_{ij}^{c}) \end{bmatrix} = \begin{bmatrix} V_x(x_{ij}^{c}, y_{ij}^{r}) \\ V_y(x_{ij}^{c}, y_{ij}^{r}) \end{bmatrix}
\end{aligned} \tag{6.33}$$

De onde obtemos os valores do campo de velocidade

$$\begin{aligned}
u_{i+\frac{1}{2},j} &= V_x(x_{ij}^{(r)}, y_{ij}^{(c)}) = \alpha(y_{ij}^{(c)} - c_y) \\
v_{i,j+\frac{1}{2}} &= V_y(x_{ij}^{(c)}, y_{ij}^{(r)}) = \alpha(c_x - x_{ij}^{c})
\end{aligned} \tag{6.34}$$

A figura 6.20 mostra o resultado da rotação variando o parametro α e assim, a configuração da superfície ao cabo da simulação.

6.4 A superfície no Freeflow

Nesta seção observamos a PUI atuando em problemas clássicos referente à mecânica dos fluidos.

6.4.1 Tensão Superficial

Este problema trata de um fluido inicialmente em forma de um quadrado que sob o efeito da tensão superficial se transforma ao longo do tempo em um círculo.

As dimensões do problema são:

- Domínio $\Omega = [0,1] \times [0,1]$;
- Fluido em forma de quadrado de aresta $\ell = 0.475$;
- Tensão superficial $\gamma = 0.1$;
- Número de Reynolds $Re = 1$;
- $\Delta x = \Delta y = 2.5 \times 10^{-2}$;
- Força da gravidade nula $\vec{g} = \vec{0}$;
- Velocidade inicial do fluido nula;

Primeiro observamos na figura 6.21 o efeito da tensão superficial em uma simulação em que não há correções de ondulações. As extremidades pontudas do fluido em forma de quadrado evoluem no tempo para formar uma configuração não física do problema.

Partimos para observar a evolução da simulação em que a superfície utilizada é a definida pela partição da unidade. Conforme podemos observar na figura 6.22, já no primeiro ciclo da simulação, a superfície PUI suaviza via aproximações polinomiais locais a borda do quadrado, efeito que é cada vez mais sutil de se observar conforme aumentamos o número de subcélulas. O resultado desta simulação se assemelha a daquela em que se usa o eliminador de ondulações, conforme observamos na figura 6.23.

6.4.2 Gota em queda livre

Nesta seção, observamos o efeito de uma gota em queda livre e seu impacto ao atingir uma plataforma. As dimensões do problema são conforme:

- Domínio $\Omega = [0,1] \times [0,1]$;
- Raio da circunferência (fluido) $r = 0.125$;
- Centro da circunferência $C = [0.5, 0.7]$;
- Número de Reynolds $Re = 1$;
- Tensão superficial nula;
- Velocidade inicial do fluido nula;
- $\Delta x = \Delta y = 10^{-2}$;
- Força da gravidade somente na direção $\hat{y}$,

$$\vec{g} = \begin{bmatrix} 0 \\ \\ -g \end{bmatrix}$$

- Dimensões do conteiner:
  - `ORIGIN = 0.07`
  - `LENGTH = 0.9`
  - `WIDTH = 0.9`
  - `THICKNESS = 0.02`

A figura (6.26) mostra a configuração da simulação: acima a gota, a uma altura de 0.7, movimentada em queda livre e a figura abaixo mostra a gota momentos antes

de seu impacto. Este problema tem uma pequena assimetria, o que permite visualizar duas circunstâncias de impacto do fluido com as partes laterais do conteiner.

A primeira seguência de imagens – figuras 6.27 e 6.28 – mostram a simulação sem projeção na superfície PUI. As figuras 6.29 e 6.30 correspondem à simulação utilizando a PUI. Finalmente, a figura 6.31 compara as duas simulações no mesmo tempo. A projeção suaviza as extremidades da superfície de maneira a manifestar o comportamento de um fluido.

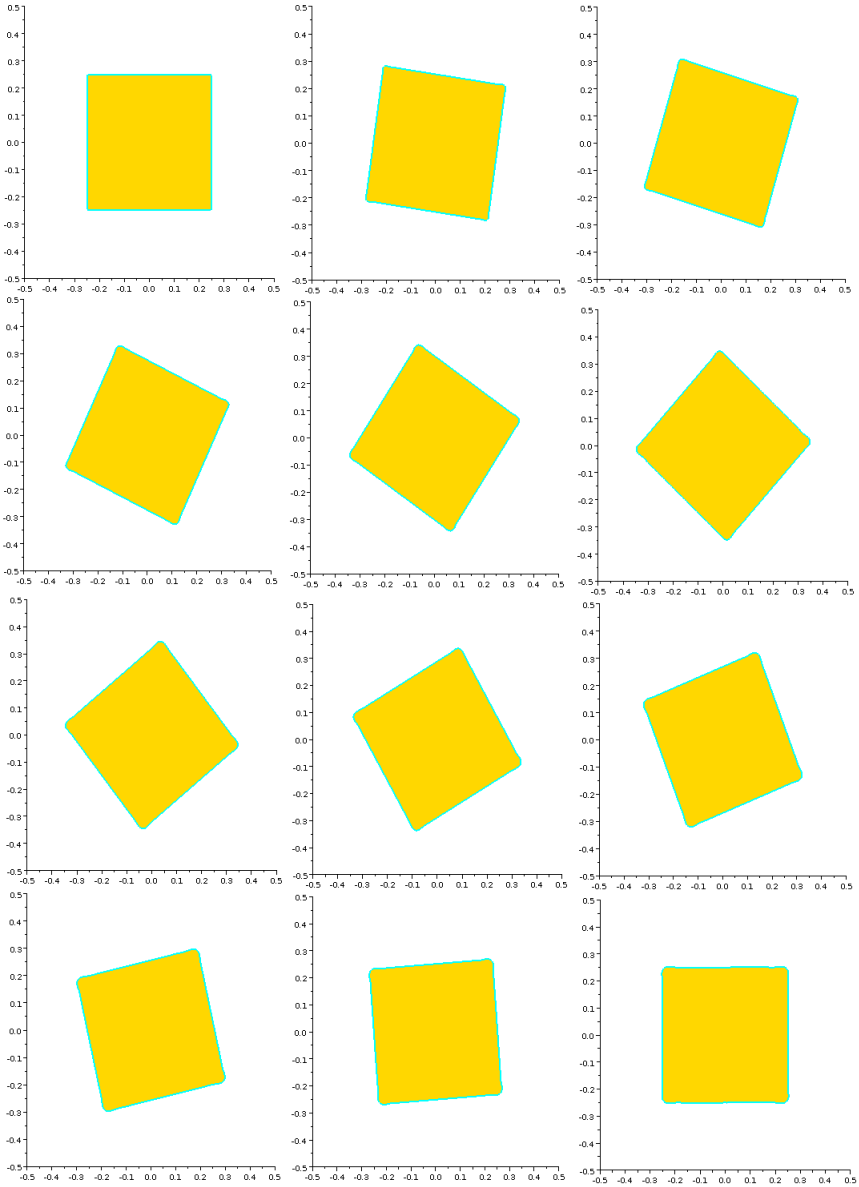


Figura 6.20: Rotação do quadrado com projeção na superfície PUI.

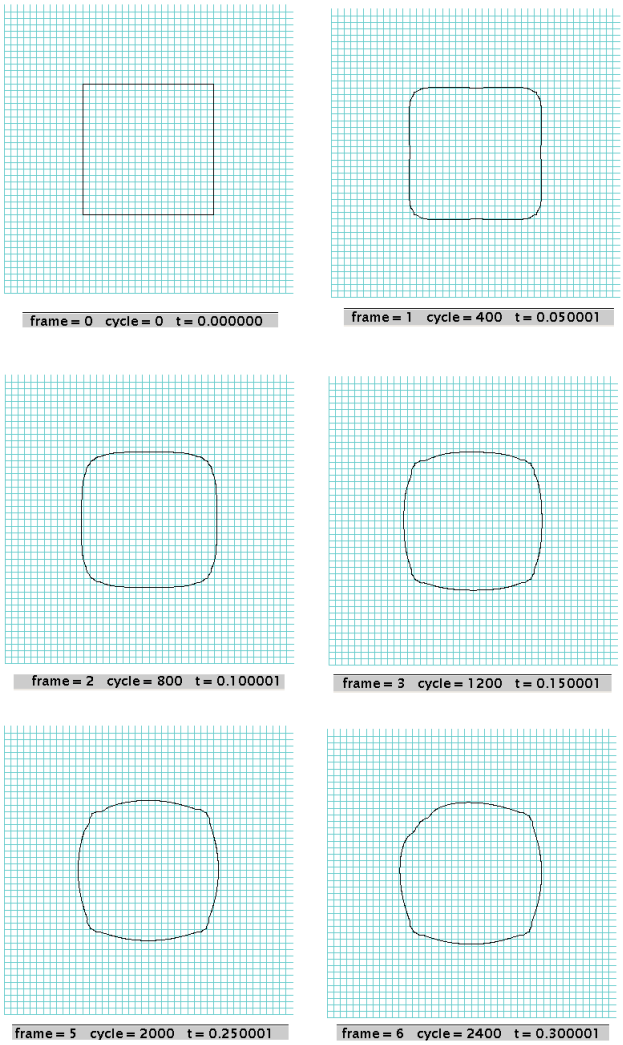


Figura 6.21: Efeito da tensão superficial do *freeflow* sem eliminação de ondulações.

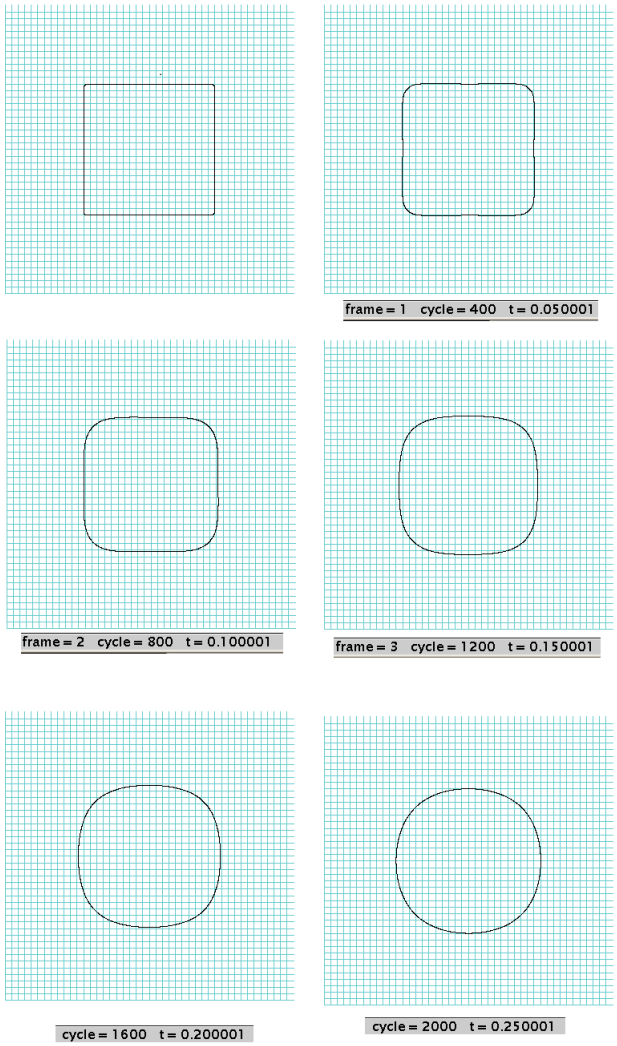


Figura 6.22: Superfície PUI atuando na tensão superficial.

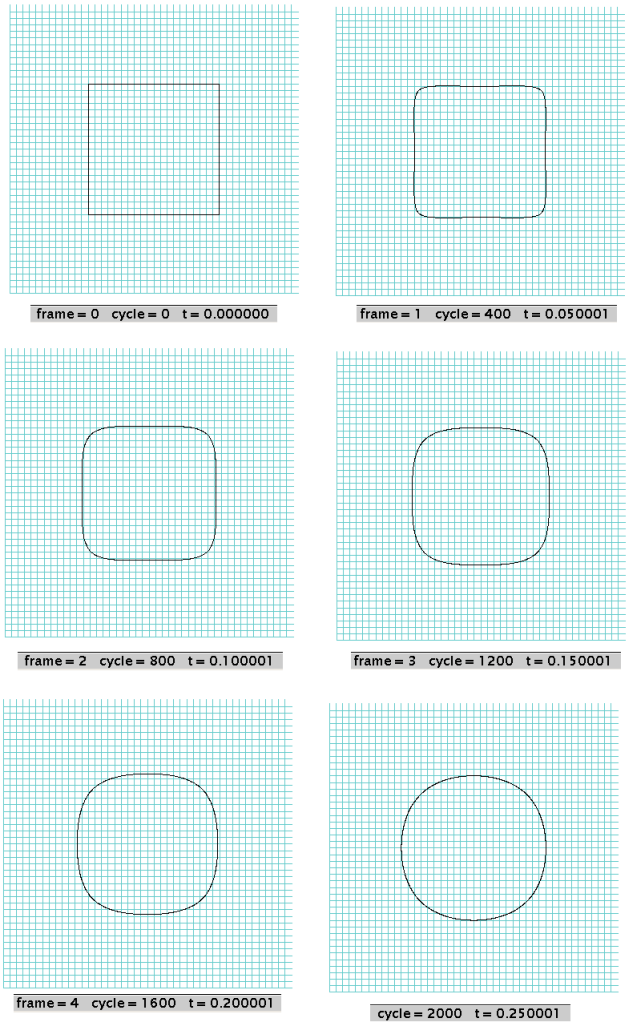


Figura 6.23: Eliminador de ondulações TSUR atuando na tensão superficial, em cada ciclo da simulação.

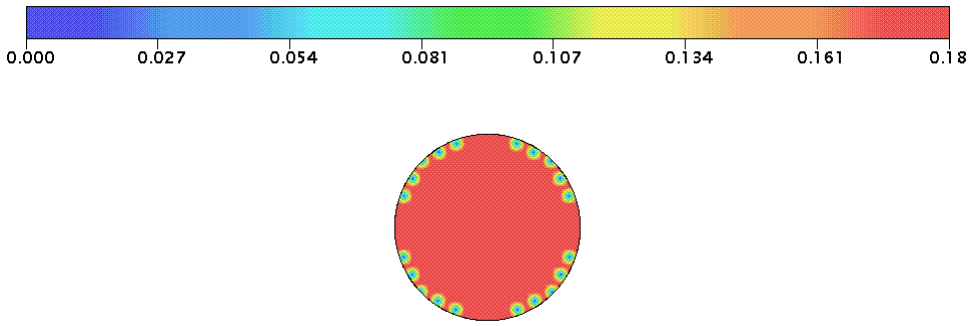


Figura 6.24: A pressão e o resultado final quando se adiciona o método `TSUR`.

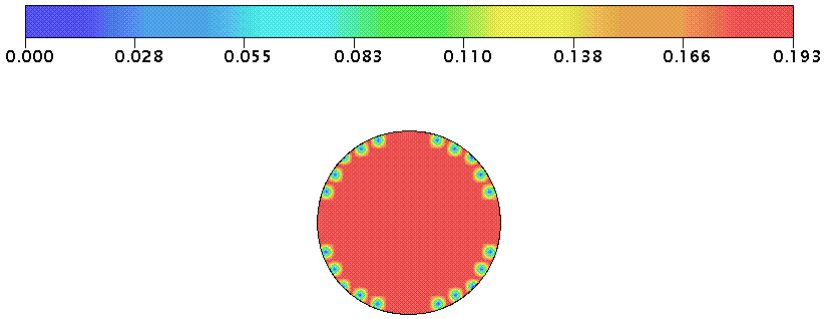


Figura 6.25: A curvatura calculada segundo a partição da unidade somada à projeção na superfície dão o efeito de eliminação de ondulação, assim como fornece um resultado análogo quando observamos a pressão.

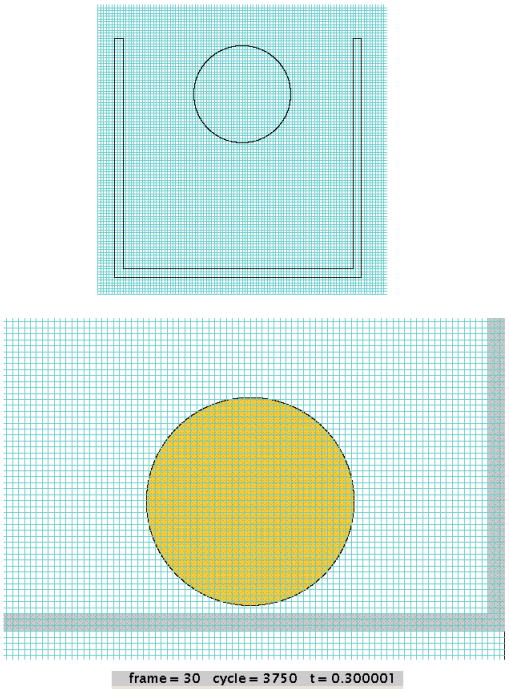


Figura 6.26: Queda livre e impacto de uma gota de fluido.

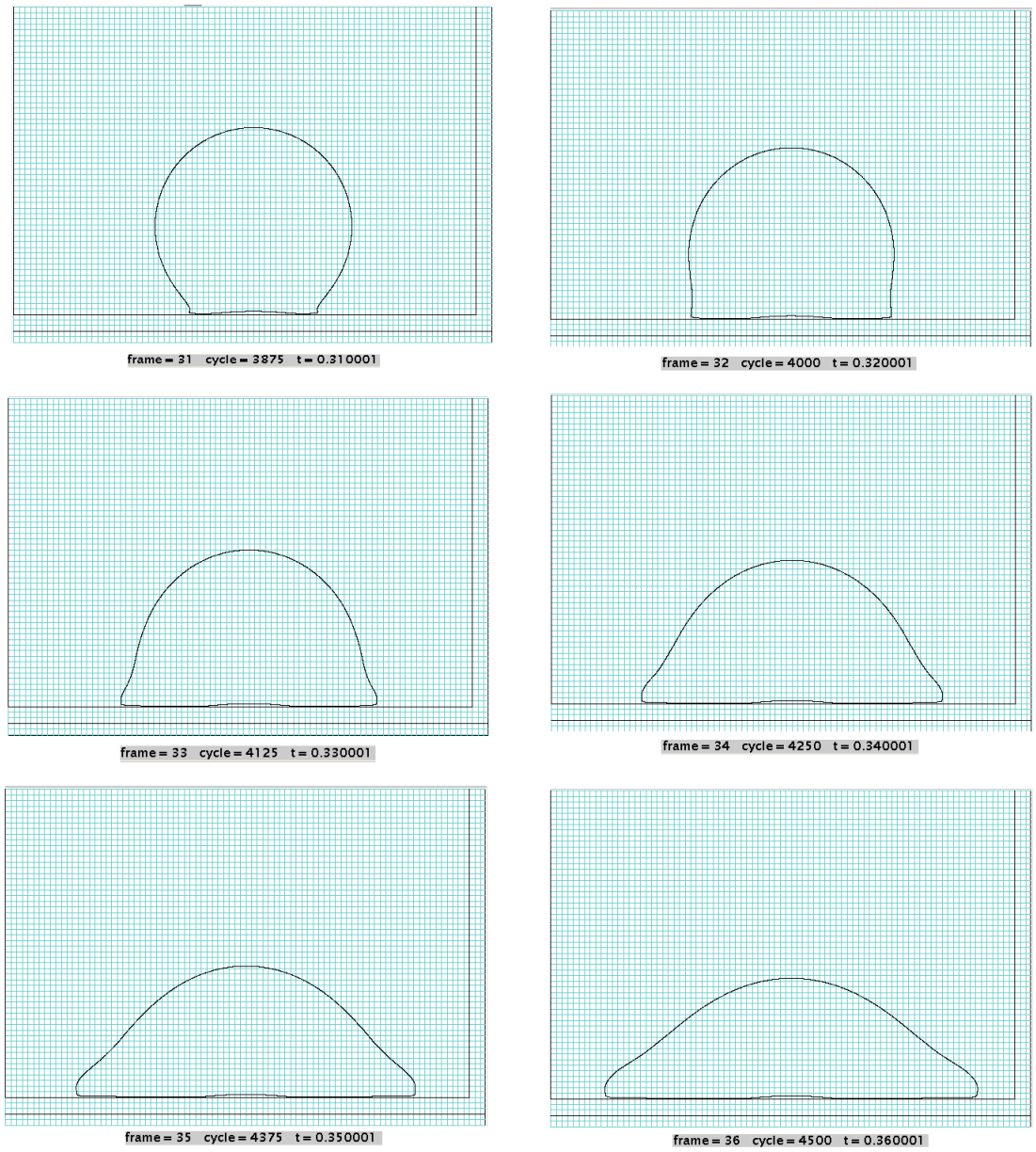


Figura 6.27: Simulação da Queda livre do *Freeflow* atual para $t \leq 0.36$.

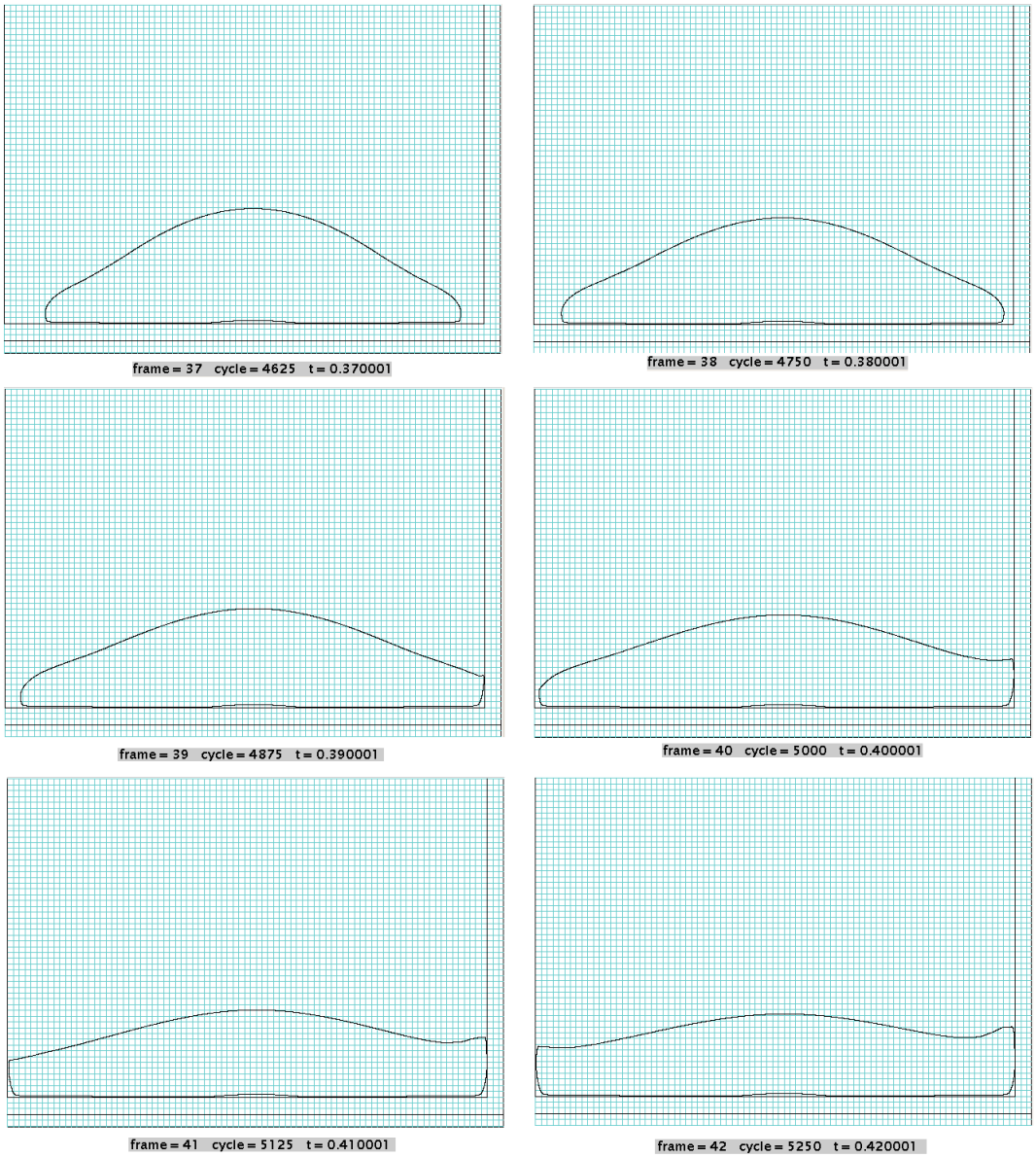


Figura 6.28: Simulação da Queda livre do *Freeflow* atual para $0.37 \leq t \leq 0.45$.

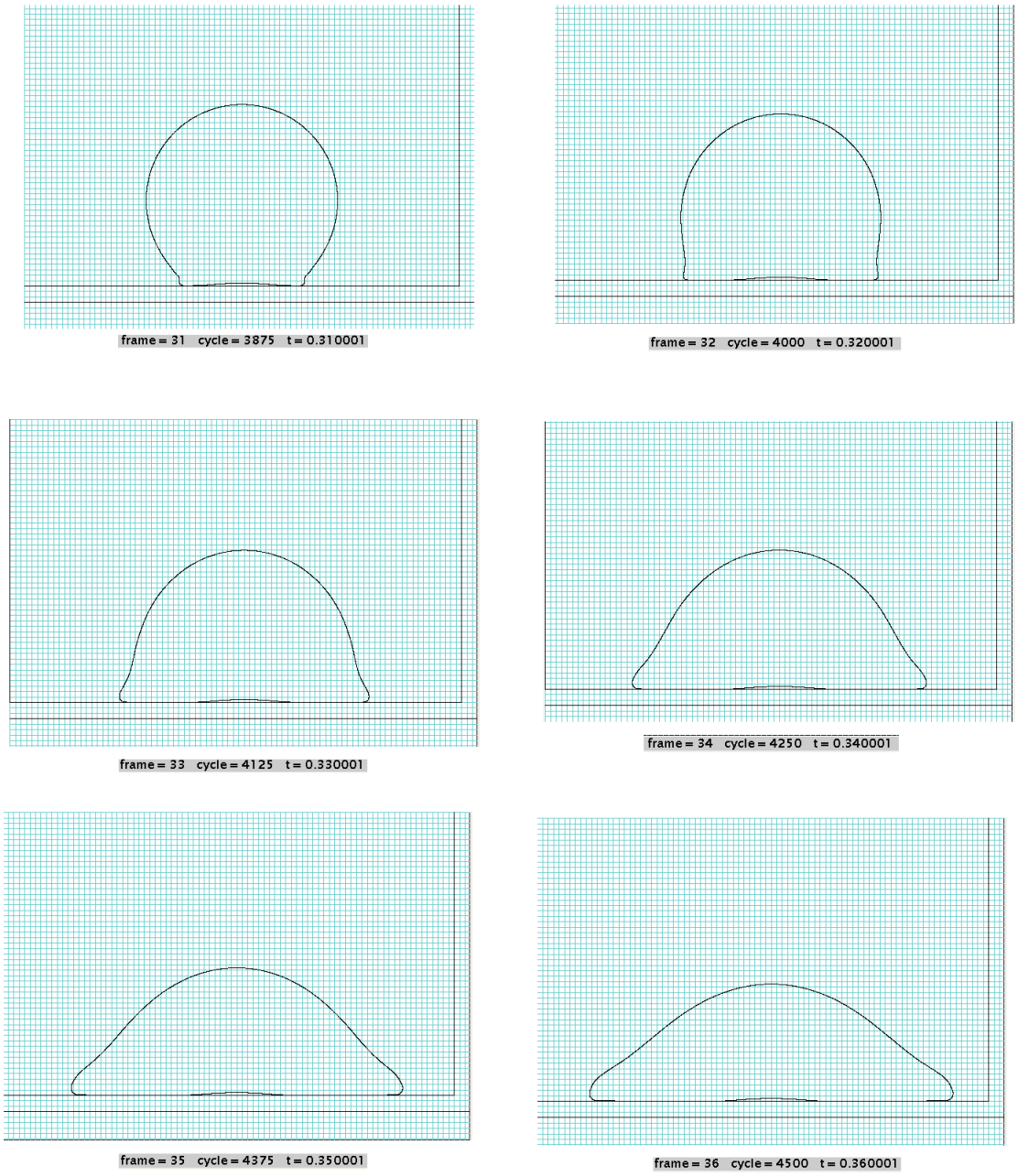


Figura 6.29: Simulação da Queda livre usando a superfície PUI para $t \leq 0.36$.

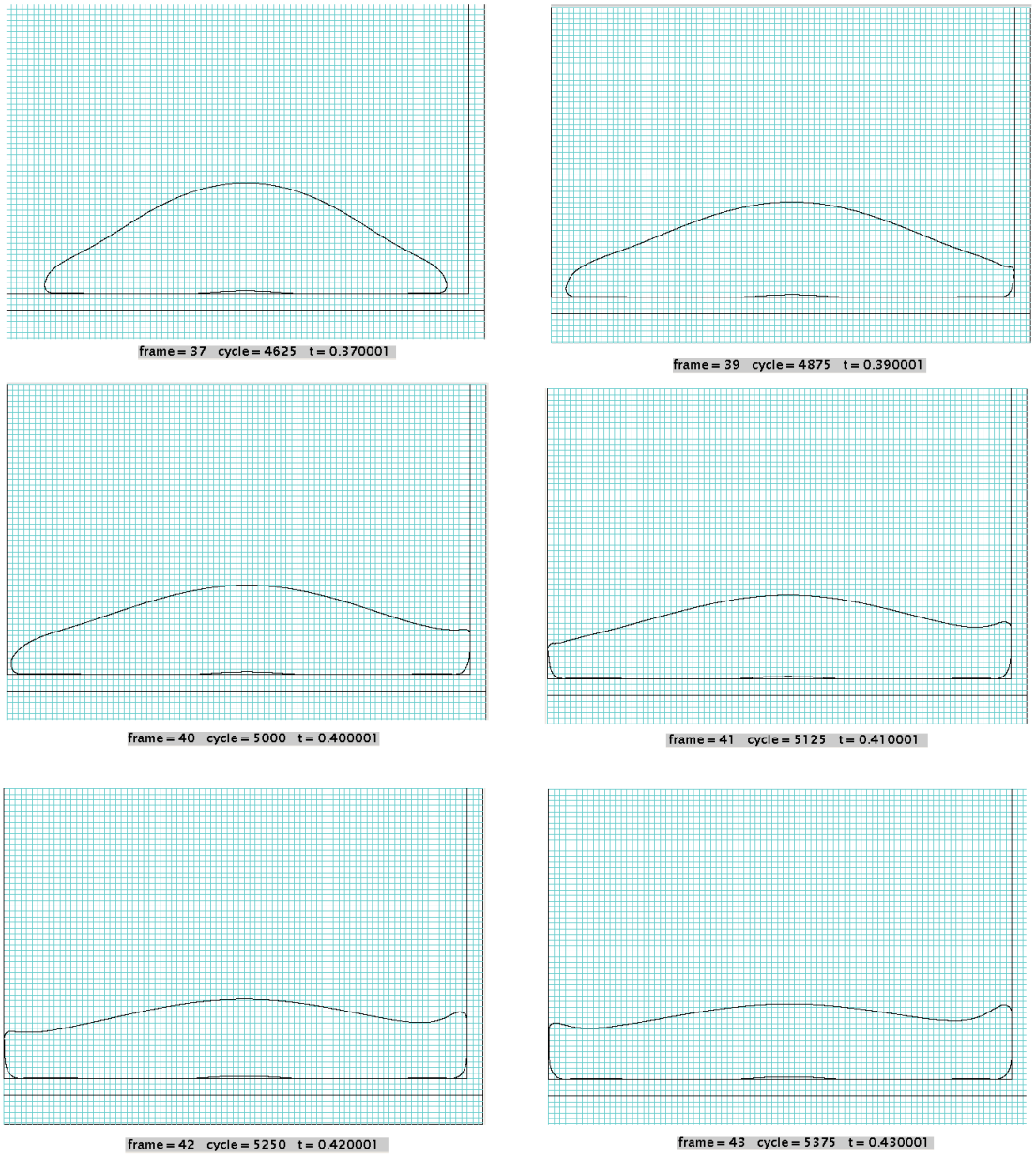


Figura 6.30: Simulação da Queda livre usando a superfície PUI para $0.37 \leq t \leq 0.45$.

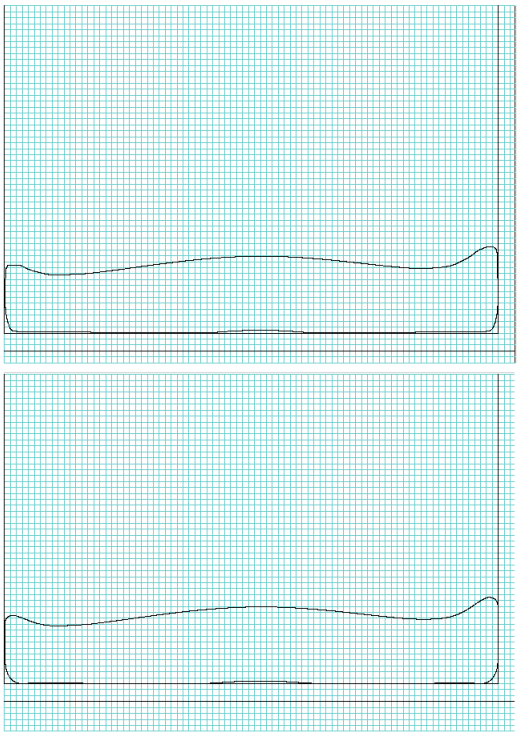


Figura 6.31: Resultado comparado das duas simulações, acima sem a técnica desenvolvida (projeção na PUI) e abaixo com projeção.

Capítulo 7

Considerações Finais

Tal qual um primeiro trabalho que estende a representação de superfícies no *Freeflow2D* este projeto de mestrado encerra com um horizonte mais largo que suas consecuções. O algoritmo herdou muitos recursos da malha lagrangiana sendo possível tornar esta dependência cada vez mais exígua, de tal maneira que o usuário tenha no futuro três possibilidades de representação da interface: i) Um algoritmo inteiramente baseado na representação por malha lagrangiana, tal qual é feito até então; ii) uma combinação de métodos implícitos com recursos da malha lagrangiana, conforme o trabalho que aqui foi desenvolvido; e iii) um método puramente baseado em partículas, em que não há nenhuma representação por malha lagrangiana;

7.1 Conclusão

Tanto a formulação teórica das representações locais – no que tange funções locais e o método dos mínimos quadrados – quanto a junção destas em uma representação global – com uma função do tipo partição da unidade – e as propriedades geométricas de que se derivam – como a projeção na superfície e cálculos de normais e curvaturas – são concepções teóricas de análogo desenvolvimento no cenário tridimensional. Em

outras palavras, tudo o que foi exposto neste documento e que foi necessário para o desenvolvimento da versão bidimensional do algoritmo é naturalmente transposto para a versão 3D do código (cf. trabalhos futuros, seção 7.2).

O ponto forte deste trabalho é uma vertente nova de tratamento da interface trazida para o *Freeflow2D*, de tal maneira que a inclusão de novas representações locais e/ou funções de tipo peso é imediata, podendo dispensar inclusive o cálculo analítico das derivadas, já que uma das possibilidades é calcular por diferenças finitas[1]. O objetivo é poder incluir vários algoritmos de representação da superfície a longo prazo, de tal maneira a poder compará-los todos estudando pontos fortes de cada um.

7.2 Trabalhos Futuros

Uma extensão imediata do trabalho é passar o algoritmo para o *Freeflow3d*, representando escoamentos tridimensionais. Além disso, algumas melhorias podem ser feitas no algoritmo para aumentar a gama de problemas que resolve, dentre as quais podemos citar

- **Adaptatividade.** O algoritmo pode provocar divisões das subcélulas em função da curvatura respectiva, aumentando a precisão localmente na medida em que se faz necessária.

- **Histórico.** Para cada segmento da superfície situada em uma célula $\mathcal{C}$ atribui-se um parâmetro $H_{\mathcal{C}}$ que o situa na vizinhança. Desta forma, para partículas de uma determinada região o cálculo dos mínimos quadrados se servirá do critério de seu histórico $H_{\mathcal{C}}$, impedindo problemas de continuidade quando seções desconexas da interface se encontram, tornando optativo mudar sua topologia, assim como utilisar estes pontos no cálculo do sistema linear que aproxima o polinômio

[1] A função peso do tipo Wendland, cujos resultados estão dispostos nas tabelas 6.14 e 6.15 foi incluída no algoritmo em questão de poucos minutos.

localmente. Alternativamente, poderá haver duas representações do polinômio, uma para cada histórico de partícula. A figura 7.2 mostra um caso em que uma figura se contorce a ponto de duas regiões distintas da mesma seção conexa se encontram em um dado momento da simulação e com o devido tratamento do *histórico* das partículas, estas serão discriminadas de acordo com a curva a que pertencem de maneira a não entrarem na vizinhança dos pontos que determinam o sistema linear da representação local.

- **Paralelismo.** Sendo os sistemas lineares das representações lineares independentes entre si e sendo que as funções do tipo peso têm suporte apenas em uma vizinhança próxima do centro das células, as informações processadas são sempre locais, ou seja, o algoritmo é paralelizável. Repartindo informações das partículas entre vários processadores, podemos tratar a superfície independentemente, tomando o cuidado de transmitir uma quantidade suficiente de dados para que se tenha a *ligadura* da superfície; assim, cada seção contígua no paralelismo terá células partilhadas na borda da região tratada, como podemos ver ilustrado na figura 7.1.

- **RAMLS.** Pretende-se também extender o trabalho para incorporar o método RAMLS como representação de superfície no *Freeflow* e métodos interpolatórios, como os de função de base radial de Hermite.

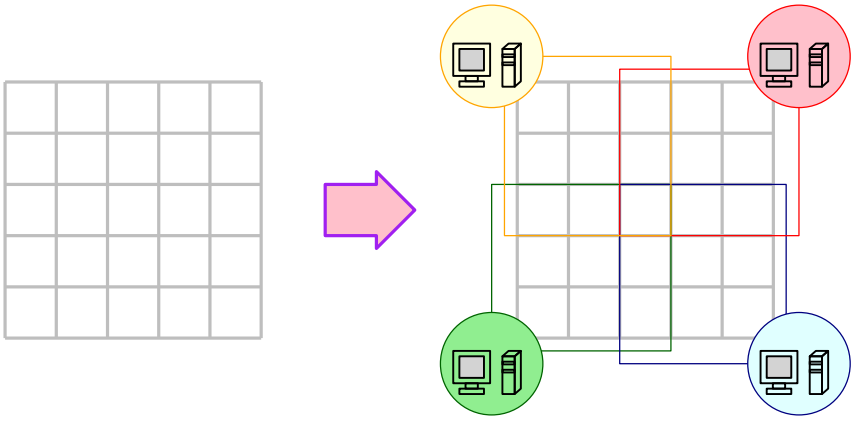


Figura 7.1: Paralelismo: visualização representativa de uma malha partilhada entre 4 processadores.

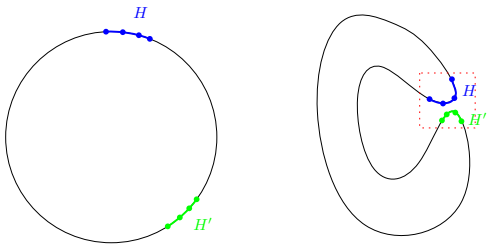


Figura 7.2: Histórico: catalogando os segmentos da curva, o algoritmo toma cuidado para não integrar informação indesejada no sistema linear que define a curva localmente; além de discriminar a origem da região dos pontos, a representação de cada parte se fará com um polinômio para cada histórico.

Referências Bibliográficas

[1] L. G. Nonato, N. Mangiavacchi, F. S. Sousa, A. Castelo, J. A. Cuminato. A Mass Conserving Smoothing Method. *XIV Congreso sobre métodos numéricos y sus aplicaciones - ENIEF, 2004,* Bariloche. Proceedings ENIEF'2004, 2004. p. 1-13.

[2] M. Mäntylä. An Introduction to Solid Modeling. *Computer Science Press*. Rockville, MD, 1988.

[3] S. McKee, M. F. Tomé, V. G. Ferreira, J. A. Cuminato, A. Castelo, F. S. Sousa e N. Mangiavacchi. The MAC method. *Computers & Fluids* , v. 37, p. 907-930, 2008.

[4] S. McKee, M. F. Tomé, J. A. Cuminato, A. Castelo e V. G. Ferreira. Recent advances in the marker and cell method. *Archives of computational methods in engineering*, volume 11, Number 2, 107-142.

[5] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin e C. T. Silva. Computing and Rendering Point Set Surface. *Computer Society*, 2003.

[6] D. Enright, R. Fedkiw, J. Ferziger e I. Mitchell. A Hybrid Particle Level Set Method for Improved Interface Capturing. *Journal of Computational Physics*, 183, 83-116(2002).

[7] G. Guennebaud e M. Gross. Algebraic Point Set Surface. *Siggraph 2007*, San-Diego, US.

[8] D.J. Torres e J.U. Brackbill. The Point-Set Method: Front-Tracking without Connectivity. *Journal of Computational Physics*, 165, 620-644 (2000).

[9] S. Shin e D. Juric. Modeling Three-Dimensional Multiphase Flow Using a Level Contour Reconstruction Method for Front Tracking without Connectivity. *Journal of Computational Physics*, 180, 427-470 (2002).

[10] G. Tryggvason, B. Bunner, A. Esmaeeli, D. Juric, N. Al-Rawahi, W. Tauber, J. Han, S. Nas e Y.-J. Jan. A Front Tracking Method for the Computations of Multiphase Flow. *Journal of Computational Physics*, Volume 169 , 708 - 759(May 2001).

[11] P. Lancaster e S. Kestutis. *Curve and surface fitting.* San Diego: Academic Press.

[12] S. Osher e R. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces.* Applied Mathematical Sciences 153, Springer.

[13] M. F. Tomé, R. A. P. Silva, C. M. Oishi e S. McKee. Numerical solution of the Upper-Convected Maxwell model for three-dimensional free surface flows. *Global Science Preprint.*

[14] A.A. Amsden e F.H. Harlow. The mac method: A numerical technique for calculating incompressible fluid flow. *Los Alamos Scientific La.* Report LA, 4370, 1970.

[15] A. Castelo, M. F. Tomé, J. A. Cuminato e S. McKee. Freeflow: An integrated simulation system for three-dimensional free surface flows. *Computing and Visualization in Science*, 2:199-210, 2000.

[16] A. B. Costacurta. Estratégias upwind e modelagem k-epsilon para simulação numérica de escoamentos com superfícies livres em altos números de Reynolds. *Dissertação de mestrado*, Universidade de São Paulo, ICMC/USP, 2005.

[17] V. G. Ferreira. Análise e implementação de esquema de convecção e modelos de turbulência para a simulação de escomentos incompressíveis envolvendo superfícies livres. *Tese de doutorado*, Universidade de São Paulo - ICMC/USP, 2005.

[18] M. Griebel, D. Thomas e Neunhoeffer. Numerical Simulation in Fluid Dynamics. A Practical Introduction. *Society for Industrial and Applied Mathematics* , 1997.

[19] L. Grossi. Solução Numérica de escoamentos axissimétricos não-newtonianos com superfícies. *Dissertação de Mestrado*. Universidade de São Paulo - São Carlos - 1997.

[20] F. H. Harlow e J. E. Welch. The mac method. *Phisics of Fluids*. 8:2182-2189, 1965.

[21] S. MacKee, M.F. Tomé, J. A. Cuminato, A. Castelo e V. G. Ferreira. Recent advances in the marker and cell method. *Archives of Computational Methods in Engeneering*. 11(2):107-142, 2004.

[22] M. Mantila. *An Introduction to Solid Moddeling*. Computer Science Press. 1988.

[23] C. M. Oishi Análise e implementação de métodos implícitos no sistema freeflow2d. *Dissertação de Mestrado*. Universidade de São Paulo - ICMC/USP, 2004.

[24] J. Oliveira. Desenvolvimento de um sistema de simulação de escoamentos axissimétricos com superfícies livres bidimensionais. *Dissertação de Mestrado*. Universidade de São Paulo - ICMC/USP, 2004.

[25] M. L. B. Oliveira. Freeflow-axi: um ambiente de simulação de escoamentos axissimétricos com superfícies livres. *Dissertação de Mestrado*. Universidade de São Paulo - ICMC/USP, 2000.

[26] L. S. Paiva. Desenvolvimento de um modelador de movimentos para o sistema freeflow3d. *Dissertação de Mestrado*. Universidade de São Paulo - ICMC/USP.

[27] W. Shyy, H. S. Udaykumar, M. M. Rao e R. W. Smith. *Computational Fluid Dynamics with Moving Boundaries.* Taylos & Francis, 1996.

[28] F. S. Sousa, N. Mangiavacchi, L. G. Nonato, A. Castelo, M. F. Tomé e V. G. Ferreira A front-tracking/front-capturing method for the simulation of 3d multi-fluid flows with free surfaces. *Journal of Computational Physics.* 198:469-499, 2004.

[29] M. Tomé, L. Grossi, A. Castelo, J. A. Cuminato, N. Manggiavacchi, V. G. Ferreira, F. S. Sousa e S. McKee. A numerical method for solving three-dimensional generalized newtonian free surface flows. *Journal of Non-Newtonian Fluid Mechanics.* 123:85-103, 2004.

[30] M. Tomé, L. Grossi, A. Castelo, J. A. Cuminato, N. Manggiavacchi, V. G. Ferreira, F. S. Sousa e S. McKee. A numerical method for solving unsteady three-dimensional free surface flows. *International International Journal for Numerical Methods in Fluids.* 37:747-796, 2001.

[31] M. F. Tomé e S. McKee. Gensmac: a computationl Marker-and-Cell method for free surface flows in general domains. *Journal of Computational Physics.* 110:171-186, 1994.

[32] J. E. Welch, F. H. Harlow, J. P. Shannon e B. J. Daly. *The mac method.* Technical report, Los Alamos Scientific Laboratory Report LA-3425, 1996

[33] I. L. Cunha. Estrutura de dados mate face e aplicações em geração e movimento de malhas. *Dissertação de Mestrado.* Universidade de São Paulo - São Carlos, 2009.

[34] J. P. Gois, A. Nakano, L. G. Nonato e G. C. Buscaglia. Front tracking with moving-least-squares surfaces. *Journal of Computational Physics.* 227 (2008) 9643-9669.

[35] J.D. Boissonnat. Geometric structures for thee-dimensional shape representation. *ACM Transactions on Graphics,* 3(4):266-286, 1984.

[36] U. Adamy, J. Giesen e M. John. Surface reconstruction using umbrella filters. *Computational Geometry Theory and Applications*, 21:63-86, 2002.

[37] N. Amenta, M. Bern e M. Kamvysselis. A new voroni-based surface reconstruction algorithm. *SiGGRAPH'98: Proceedings of the 25th annual conference on Computer Graphics and interactive techniques*, 415-421, New York, NY, USA, 1998. ACM Press.

[38] D. Attali. r-regular shape reconstruction from unorganized points. *Computational Geometry* , 10(4):239-247, July 1998.

[39] F. Bernardini e C. Bajaj. Sampling and reconstructing manifolds using alpha-shapes. *In Proc. 9th Canad. Conf. Comput. Geom*, pages 193-198, 1997.

[40] M. Bolitho, M. Kazhdan, R. Burns e H. Hoppe. Multilevel streaming for out-of-core surface reconstruction. *In SGP '07: Proceedings of the fifth Eurographics symposium on Geometry processing*, 69-78, Aire-la-Ville, Switzerland, 2007. Eurographics Association.

[41] T.K. Dey. Curve and Surface Reconstruction: Algorithms with Mathematical Analysis. *Cambridge University Press*, 1 edition, 2006.

[42] H. Bíscaro, A. Castelo, L. Nonato e M. Oliveira. A topological approach for surface reconstruction from sample points. *The visual computer*, 23(9-11):793-801, 2007.

[43] J. P. Gois. Reconstrução de Superfícices a partir de Nuvens de Pontos. *Dissertação de mestrado*. Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2004.

[44] J. P. Gois, A. C. Filho, L. G. Nonato e H. H. Bíscaro. Surface reconstruction: Classification, comparisons and applications. *Proceedings of XXY CILAMCE - Iberian Latin American Congress on Computational Methods*, 2004.

[45] H. Hope, T. DeRose, T. Duchampy, J. McDonaldz, and W. Stuetzlez. Surface reconstruction from unorganized points. *Computer Graphics*, 26(2):71-78, 1992.

[46] L. Kobbelt e M. Botsch. A survey of point-based techniques in computer graphics. *Computers and graphics*, 28(6):801-814, 2004.

[47] M. Kazjhdan, M. Bolitho e H. Hoppe. Poisson surface reconstruction. *Proceedings of Eurographics Symposium on Geometry Processing,* 2006.

[48] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk e H.-P.Seidel. Multi-Level partition of unity implicits. *ACM Trans. Graph,* 22(3):463-470, 2003.

[49] H. -K. Zhao, S. Osher e R. Fedkiw. Fast surface reconstruction using the level set method. *In Proceedings of IEEE Workshop on Variational and Level Set Methods in Computer Vision*, 194-201, 2001.

[50] H. Zhao, S. Osher, B. Merriman e M. Kang. Implicit and nonparametric shape reconstruction from unorganized data using a variational level set method. *Comput. Vis. Image Underst.*, 80(3):295-314, 2000.

[51] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. MacCallum e T. R. Evans. Reconstruction and representation of 3d objects with radial basis functions. *SIGGRAPH'01,* pages 67-76, 2001.

[52] S. Fleishman, D. Cohen-Or e C. Silva. Robust moving least-squares fitting with sharp features. *ACM Trans. Graph.,* 24(3):544-552, 2005.

[53] B. S. Morse, T. S. Yoo, P. Rheingans, D. T. Chen e K. R. Subramanian. Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions. *SIGGRAPH 2005*, p. 78, New York, NY, USA, 2005. ACM.

[54] Y. Ohtake, A. Belyaev e H.-P. Seidel. 3d scattered data approximation with adaptive compactly supported radial basis functions. *SMI'04: Proceedings of the Shape*

Modeling International 2004 , 31-39, Washington, DC, USA, 2004. IEEE Computer Society.

[55] M. F. Tomé, A. Castelo, J. A. Cuminato, N. Mangiavacchi e S. McKee. Gensmac3d: A numerical method for solving unsteady three dimensional free surface flows. *International Journal for Numerical Methods in Fluids,* 37:747-796, 2001.