COPPE
Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia UFRJ

ESCOAMENTO MULTIFÁSICO DE FLUIDOS INCOMPRESSÍVEIS EMPREGANDO LEVEL SET REGIONAL E CONTROLE DE VOLUME

Túlio Ligneul Santos

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientadores: Antonio Alberto Fernandes de Oliveira
Paulo Roma Cavalcanti

Rio de Janeiro
Março de 2013

ESCOAMENTO MULTIFÁSICO DE FLUIDOS INCOMPRESSÍVEIS EMPREGANDO LEVEL SET REGIONAL E CONTROLE DE VOLUME

Túlio Ligneul Santos

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Prof. Antonio Alberto Fernandes de Oliveira, D.Sc.

Prof. Gilson Antônio Giraldi, D.Sc.

Prof. Marcelo Bernardes Vieira, D.Sc.

Prof. Paulo Roma Cavalcanti, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
MARÇO DE 2013

Santos, Túlio Ligneul

Escoamento Multifásico de Fluidos Incompressíveis Empregando Level Set Regional e Controle de Volume/Túlio Ligneul Santos. – Rio de Janeiro: UFRJ/COPPE, 2013.

XVII, 87 p.: il.; 29, 7cm.

Orientadores: Antonio Alberto Fernandes de Oliveira
Paulo Roma Cavalcanti

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2013.

Referências Bibliográficas: p. 80 – 87.

1. simulação de fluidos. 2. simulação euleriana. 3. *level set* regional. 4. bolhas. 5. simulação multifásica. 6. controle de volume. I. Oliveira, Antonio Alberto Fernandes de *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Aos meus pais, Gisele e José Bento.

Agradecimentos

Gostaria de agradecer aos meus pais, Gisele e José Bento, pelo incentivo e orientação passados em todos os anos de minha vida. Ao meu irmão, Érick, pela união e exemplo de vida. Aos meus amigos Guilherme Martins, Erick Tavares, Gabriel Portugal, João Luiz, Thiago Xavier, Raisa Heber, Mariana Campos, Isabela Fiad e Lucas Brito pelo apoio e companheirismo recebidos ao longo dos últimos anos. Aos professores e orientadores Antonio Oliveira e Paulo Roma pelos ensinamentos durante a execução deste trabalho. Aos professores Gilson Giraldi e Marcelo Vieira pela presença na banca examinadora. Aos demais professores ao longo de toda a minha formação pela contribuição ao meu aprendizado. Ao povo brasileiro que contribuiu de forma significativa para minha formação e estada nesta Universidade. Este projeto é uma pequena forma de retribuir o investimento e confiança em mim depositados.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

ESCOAMENTO MULTIFÁSICO DE FLUIDOS INCOMPRESSÍVEIS EMPREGANDO LEVEL SET REGIONAL E CONTROLE DE VOLUME

Túlio Ligneul Santos

Março/2013

Orientadores: Antonio Alberto Fernandes de Oliveira
Paulo Roma Cavalcanti

Programa: Engenharia de Sistemas e Computação

Aborda-se, nesta dissertação, o *level set* regional como solução para o problema do acompanhamento das interfaces em simulações multifásicas de fluidos, em especial no tratamento do fino filme presente entre fases distintas. Por este método, também são rastreadas as propriedades de cada região, como o volume e a espessura do filme, inclusive quando elas se fundem ou se dividem. Ademais, propõe-se um novo método para tratar as variações de volume indesejadas ou as realizar de modo controlado. Estes objetivos são atingidos através da evolução das superfícies com base em velocidades auxiliares estimadas próximo das interfaces, e que são definidas em função do histórico do erro volumétrico. Adicionalmente, geram-se partículas a partir de fases pequenas que, em virtude de limitações da discretização adotada, teriam desaparecido.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

MULTIPHASE FLOW OF INCOMPRESSIBLE FLUIDS EMPLOYING REGIONAL LEVEL SET AND VOLUME CONTROL

Túlio Ligneul Santos

March/2013

Advisors: Antonio Alberto Fernandes de Oliveira
Paulo Roma Cavalcanti

Department: Systems Engineering and Computer Science

In this work, it is presented the approach of the regional level set to solve the problem of monitoring interfaces in multiphase fluid simulations, mainly in relation to the treatment of the thin film between distinct phases. Through this method, properties of each region, such as volume and film thickness, are tracked, including as regions merge or divide. Furthermore, it is proposed a new method for handling undesired changes in volume or inducing controlled volume changes. These objectives are achieved through the evolution of surfaces based on auxiliary velocities that are estimated near the interface and are determined from the historical data of the volumetric error. In addition, particles are generated from small phases that, because of limitations of the adopted discretization, would have disappeared.

Sumário

Lista de Figuras

Lista de Símbolos

(d_x, d_y, d_z)	Dimensão da grade MAC em cada eixo ordenado, p. 11
(i, j, k)	Para uma célula da grade MAC, suas coordenadas discretas relativas, respectivamente, aos eixos x, y e z, p. 10
(p_x, p_y, p_z)	Derivadas parciais da pressão p em relação aos respectivos eixos x, y e z, p. 51
(u, v, w)	Componentes da velocidade $\vec{u}$, respectivamente, nas direções dos eixos x, y e z, p. 9
$(x_{max}, y_{max}, z_{max})$	Coordenadas máximas de qualquer ponto pertencente à grade MAC, p. 10
$(x_{min}, y_{min}, z_{min})$	Coordenadas mínimas de qualquer ponto pertencente à grade MAC, p. 10
G	Grafo regional do *level set* regional, p. 44
$H(\Phi)$	Função Heaviside, p. 31
J	Variação (ou *jump*) de pressão através da superfície Γ, p. 50
N_{face}	Componente de $\hat{N}$ na direção perpendicular à uma determinada face, p. 60
O	Matriz com valores nulos, p. 21
V	Volume, p. 7
V_{min}	Volume mínimo que uma região pode possuir sem ser convertida em partícula, p. 47
$Vel_{entrada}$	Componente da velocidade $\vec{u}$ que aponta para dentro da massa fluida, sendo estimado em uma face nos limites da grade computacional, p. 63

Vel_{face}	Componente da velocidade $\vec{u}$ estimado no centro de uma face de célula, podendo se referir à u, v ou w, p. 59
$Vel_{saída}$	Componente da velocidade $\vec{u}$ que aponta para fora da massa fluida, sendo estimado em uma face nos limites da grade computacional, p. 63
Δt	Tamanho do passo de tempo t, i.e., tempo decorrido entre t e o passo de tempo seguinte, p. 13
Δx	Dimensão de cada célula da grade MAC, p. 11
Ω	Porção de volume, p. 9
α	Número CFL, p. 26
$\delta(\Phi)$	Derivada da função Heaviside, p. 32
η	Coeficiente de viscosidade dinâmica, p. 8
$\hat{N}$	Vetor unitário normal à superfície., p. 9
κ	Curvatura da interface, p. 27
$\mathbb{R}$	Conjunto dos números reais, p. 12
$\mathbb{R}^+$	Conjunto dos números reais não negativos, p. 36
$\mathbb{Z}$	Conjunto dos números inteiros, p. 12
ν	Coeficiente de viscosidade cinemática, p. 7
$\partial\Omega$	Superfície da porção de volume Ω, p. 9
$\phi(\vec{x})$	Função que define o valor de um *level set* na posição $\vec{x}$, p. 30
ρ	Densidade, p. 7
σ	Coeficiente de tensão superficial, p. 27
$\vec{F}$	Vetor força, p. 7
$\vec{F_a}$	Vetor da força de arraste atuante sobre uma partícula em um fluido, p. 46
$\vec{F_g}$	Vetor força da gravidade, p. 8
$\vec{F_p}$	Vetor força derivado das variações de pressão, p. 8

$\vec{F}_v$	Vetor força viscosa, p. 8
$\vec{a}$	Vetor aceleração, p. 17
$\vec{g}$	Vetor aceleração da gravidade, p. 7
$\vec{u}$	Vetor velocidade, p. 7
$\vec{u}_1$	Vetor velocidade após a etapa da advecção, p. 16
$\vec{u}_2$	Vetor velocidade após a etapa de influência de forças externas, p. 17
$\vec{x}$	Posição arbitrária no espaço, p. 9
$c_{arraste}$	Coeficiente da força de arraste atuante em uma partícula no fluido, p. 46
$f_R(\vec{x})$	Função regional avaliada em $\vec{x}$, p. 35
f_e	Componente vertical da força resultante entre forças de empuxo e gravidade atuantes em uma partícula, p. 46
h	Espessura real do filme entre duas regiões do *level set* regional, p. 39
k_c	Constante usada no método de controle de volume, relativa à contribuição do histórico do erro do volume, p. 58
k_p	Constante usada no método de controle de volume, relativa à contribuição do erro atual do volume, p. 58
l	Espessura fictícia do filme de uma região do *level set* regional, p. 40
l_0	Espessura inicial do filme fictício de uma região, p. 40
l_{min}	Menor valor que a espessura l do filme fictício de uma região pode ter sem ser rompido, p. 40
m	Massa, p. 7
p	Identificador de uma partícula, p. 46
p	Pressão, p. 7
$r(\vec{x})$	Identificador da região do *level set* regional que contém $\vec{x}$, p. 35

r_p	Raio da esfera de uma partícula p, p. 46
t	t-ésimo passo de tempo, p. 7
t_{c_p}	Define o número de correções a serem feitas no raio de uma partícula para que ela atinja seu volume alvo, p. 64
$t_{persist}$	Tempo máximo que uma região pode permanecer sem ter seu filme rompido, p. 40
S	Conjunto fechado dos pontos em uma superfície, p. 30
n	Índice de refração, p. 70

Lista de Abreviaturas

1D	Unidimensional, p. 3
2D	Bidimensional, p. 3
3D	Tridimensional, p. 32
BFECC	*Back and Forth Error Compensation and Correction*, p. 57
CFL	Courant, Friedrichs e Lewy, p. 25
CUDA	*Compute Unified Device Architecture*, p. 6
GC	Gradiente Conjugado, p. 21
LPLS	*Lagrangian Particle Level Set*, p. 29
MAC	*Marker-And-Cell*, p. 10
PLS	*Particle Level Set*, p. 29
RLS	*Regional Level Set*, p. 4
VOF	*Volume Of Fluid*, p. 30

Capítulo 1

Introdução

1.1 Tema

Na área da computação gráfica, duas abordagens principais para a simulação de fluidos são aplicadas. A primeira é a abordagem lagrangiana introduzida por MÜLLER *et al.* [3], que representa um fluido como um conjunto de partículas cujo movimento define a dinâmica do fluido como um todo. Esse tipo de tratamento ainda é bastante pesquisado, como, por exemplo, nos trabalhos de SOLENTHALER e PAJAROLA [4] e SIN *et al.* [5] sobre incompressibilidade e em pesquisas recentes que utilizam esta visão para modelar pequenas regiões (CLEARY *et al.* [6], DARLES e GHAZANFARPOUR [7]) e grandes bolhas esféricas (KÜCK *et al.* [8]).

Na segunda perspectiva, a abordagem euleriana, que tem como trabalhos iniciais aqueles de FOSTER e METAXAS [9],STAM [10] e FOSTER e FEDKIW [11], procura acompanhar o estado de um fluido - definido por propriedades como velocidade e pressão - em cada ponto do espaço ocupado por ele. Assim como a perspectiva anterior, esta também ainda é alvo de muito estudo. Entre os trabalhos recentes, podem ser indicados os de MULLEN *et al.* [12], que procura preservar a energia do sistema, de KIM *et al.* [13], sobre vorticidade com foco na superfície líquida, e de WOJTAN *et al.* [14], para híbridos de superfície lagrangiana e *level set* euleriano (OSHER e FEDKIW [15]).

Além dessas duas abordagens, também é comum a utilização de métodos híbridos onde partículas ajudam o rastreamento da superfície do fluido euleriano (ENRIGHT *et al.* [16]), ou mesmo intervêm na simulação (LOSASSO *et al.* [17], HONG *et al.* [18]). De modo geral, o objetivo é enriquecer os detalhes da simulação euleriana. Como exemplo, GREENWOOD e HOUSE [19] e GUENDELMAN *et al.* [20] usam partículas escapadas do fluido ou do ar conforme determinado pelo modelo dos *particle level sets* (ENRIGHT *et al.* [16]) para criar partículas como bolhas de ar dentro do fluido ou gotas de fluido na porção de ar. Ainda, MIHALEF *et al.* [21]

faz o mesmo, mas utilizando o modelo dos *marker level sets* (MIHALEF [22]).

1.2 Delimitação

Neste trabalho, é empregada a abordagem euleriana, mais especificamente no que se refere à simulação multifásica de fluidos incompressíveis. Nesse sentido, o foco é dado para a área da computação gráfica, de modo que se busca uma simulação que visualmente pareça real, mesmo que não possua exatamente corretude física. Assim, como exemplos de resultados da simulação que se pretende atingir, podem-se citar situações em que um óleo flutua sobre a água, bolhas de sabão voam pelo ar, massas de óleo se juntam e se fundem, ou onde filmes de água separam componentes de óleo. Note que o filme entre duas fases é uma característica muito importante para a simulação, visto que ele define se as fases permanecerão separadas ou se unirão em uma fase única em um dado instante da simulação.

Logo, é preciso acompanhar como a espessura de um filme evolui, de forma a poder detectar se são atingidas condições que levam à sua ruptura e à consequente combinação das fases adjacentes. Contudo, por causa da complexidade inerente à sua evolução, é usado um modelo simplificado, de modo que seu rompimento seja provocado apenas por seu alongamento ou pela drenagem do fluido que o compõe.

Além disso, empregam-se partículas para aumentar o detalhamento da simulação. Elas são aproveitadas especialmente para criar pequenas gotas separadas de sua região original e que podem, inclusive, voltar a se fundir com essa fase inicial.

1.3 Justificativa

A motivação inicial para o trabalho surge da premissa de que, para simular interações entre diferentes fluidos, é preciso um método eficiente que permita identificar, de forma precisa, a localização das interfaces entre eles. Note que tal ideia é naturalmente aplicada para múltiplas fases do mesmo fluido, de modo que, para fins de representação, estas poderiam bem ser de outro material. Deve-se observar, ainda, que, para trabalhar com mais de dois materiais, a técnica de acompanhamento da superfície tradicionalmente aplicada, a dos *level sets* (OSHER e FEDKIW [15]), necessita ser complementada, visto que apenas é capaz de tratar de dois meios díspares.

Outra dificuldade advém da pouca espessura do filme líquido que pode existir entre fluidos diferentes. Ela pode ser tão pequena em relação ao tamanho das fases que pode ser considerada infinitesimal, determinando que a interface possa não ser uma variedade bidimensional (2-*manifold*). Desse modo, também é imperativo modificar o método dos *level sets* para permitir a existência de interfaces compostas

por variedades 1D e 2D, ditas *multi-manifold*. Por exemplo, os trabalhos de LOSASSO *et al.* [23] e VESE e CHAN [24] utilizam variantes multifásicas do *level set* original. Contudo, tais trabalhos que suportam múltiplos materiais possuem uma complexidade que cresce com o número deles.

Entre esses dois trabalhos, o primeiro emprega um *level set* para cada material. De fato, componentes do mesmo constituinte são considerados como uma única fase, de forma que é impossível simular elementos do mesmo fluido que estão em contato mas não se fundem, tal como acontece com bolhas de ar que se tocam. Além disso, não trata de filmes mais finos do que a densidade da grade, e o número de materiais é limitado devido ao custo associado às múltiplas variáveis dos *level sets*, cuja quantidade cresce linearmente com o número de fluidos distintos. Ademais, a evolução independente de vários *level sets* pode acarretar contradições na representação da interface, assim como representado na figura 1.1. Desse modo, devem ser corrigidas as situações em que existam pontos que pertençam a nenhuma ou mais de uma região.

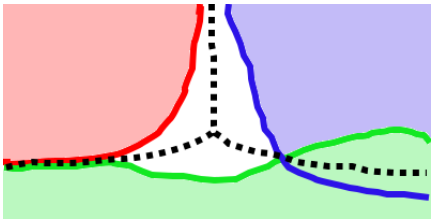

Figura 1.1: Disposição de três *level sets* após seus deslocamentos, apresentando uma região de vácuo e uma de interpenetração. As linhas pontilhas retratam a subdivisão das fases após a correção das interfaces.

Já na segunda abordagem, são empregados n *level sets* para representar até 2^n regiões. Por exemplo, dois *level sets* podem ser usados para representar até quatro regiões, classificadas através das possíveis combinações de sinal (i.e. "++", "+−", "−+"e "−−"), como ilustrado pela figura 1.2. Apesar de essa abordagem preservar a divisão das fases, apresenta artefatos indesejados, sobretudo onde mais de duas regiões intersectam.

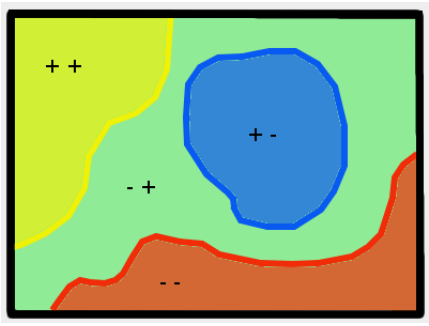

Figura 1.2: Disposição de quatro fases com base no sinal de dois *level sets*.

Outros trabalhos também já estudaram a simulação de características finas de fluidos. De fato, WOJTAN *et al.* [14] e WOJTAN *et al.* [25] utilizam modelos híbridos de grade e superfície explícita que permitem a realização de mudanças na topologia das superfícies, embora essas transformações ainda se realizem de forma bastante complexa. Já BROCHU *et al.* [26] perturba amostras para capturar as características finas. Entretanto, apesar desses métodos permitirem a inserção de características de espessura arbitrária em qualquer local, trabalham apenas com fluxos de uma ou duas fases empregando o *level set* tradicional.

Por fim, em geral, não se pode deixar de notar a ocorrência de uma perda de volume, lenta mas constante, que pode prejudicar a simulação ao longo do tempo. Ao certo, é comum em simulações de fluidos a perda de volume que pode resultar até no desaparecimento de alguma fase ou material. Então, é razoável considerar que também é necessário um esquema que garanta a manutenção do volume.

1.4 Objetivo

Este trabalho tem como objetivo desenvolver um método de evolução da superfície no contexto da simulação de fluidos incompressíveis multifásicos adotando a abordagem euleriana. Para tanto, as técnicas desenvolvidas devem ser capazes de tratar de características menores do que a resolução da grade a fim de poder representar filmes finos entre diferentes fases fluidas e possibilitar a separação e união de componentes do mesmo material. Além disso, procura-se tratar o problema da perda de volume, uma questão inerente à simulação de fluidos em geral, de modo que o volume de cada fase se mantenha constante. Adicionalmente, deseja-se permitir que o volume de cada material seja alterado de forma a se ajustar a processos como a evolução do fluxo de entrada durante a simulação, ou fazer o volume de uma bolha ser alterado de uma forma especificada.

1.5 Proposta

Tendo em vista os objetivos descritos na última seção e as dificuldades relatadas anteriormente, propõe-se a utilização da abordagem do *level set* Regional (RLS) (ZHENG *et al.* [27]) para o acompanhamento da interface. Este método, ao mesmo tempo que suporta diferentes fluidos com custo independente da multiplicidade dos materiais, define implicitamente a interface entre duas regiões do mesmo material, o que permite, por exemplo, a captura de características do fino filme líquido de bolhas de sabão. Ademais, não possui restrições quanto à existência de filmes *multi-manifold*, como o que pode ser formado na interface entre as fases quando existem três fluidos distintos. Através deste método, deseja-se identificar cada fase através

de um grafo de conectividade, como realizado por KIM [1], possibilitando que propriedades de cada fase sejam rastreadas no tempo. Assim, pode-se por exemplo, acompanhar a mudança de volume de cada fase mesmo que as elas se dividam ou se fundam a outras fases do mesmo fluido.

Em relação, especificamente, ao volume, introduz-se um novo método para controlar o tamanho de cada fase, seja para o manter constante ou o fazer variar de modo pré-definido ou de acordo com os fluxos de entrada e saída de fluido no sistema. De fato, propõe-se utilizar velocidades auxiliares exclusivamente para a movimentação das superfícies a fim de que as fases assumam seu tamanho correto. Além disso, procura-se utilizar informações do erro do volume acumulado ao longo do tempo, de modo a fazer com que, quanto mais longo for o período em que o volume de uma região é diferente do volume desejado, mais rapidamente ele deverá se aproximar do valor alvo. Adicionalmente, busca-se utilizar partículas quando o método de controle de volume não estiver conseguindo compensar o erro. Tipicamente, isto ocorre em regiões que já são muito pequenas para a discretização adotada, mas também pode ocorrer em outras situações para uma região que está sendo comprimida.

1.6 Metodologia Empregada

Para a realização deste trabalho, foi efetuado um estudo sobre métodos existentes para a simulação multifásica de fluidos incompressíveis. De maneira geral, pôde-se perceber que um ponto fundamental para este tipo de simulação é o método empregado para rastrear a interface entre fluidos distintos. Assim, tendo em vista suas vantagens, a abordagem do *level set* regional (RLS) (ZHENG *et al.* [27]) foi adotada para este trabalho. Para chegar a essa definição, foram analisados diversos artigos que empregam tal técnica ou outros métodos envolvidos.

Já no ponto de vista da simulação numérica propriamente dita, foi preciso considerar aspectos específicos do contexto multifásico euleriano. O primeiro consiste em como evoluir valores de pressão e velocidade amostrados em uma grade regular, levando em conta uma distribuição variável de densidade, referente aos diferentes fluidos. Assim, foi adaptado o método empregado por KANG *et al.* [28], que usa as técnicas desenvolvidas por LIU *et al.* [29] para resolver equações de Poisson com coeficientes variáveis.

Uma modificação realizada foi o emprego de uma técnica à parte, o método *ghost fluid* (HONG e KIM [30]), para a aplicação de forças de tensão superficial. Estas têm origem na variação de pressão através das interfaces entre fluidos diferentes, e forçam cada fase a se contrair para uma área mínima, dando, em geral, um aspecto suave às superfícies. A alteração restante foi a desconsideração do comportamento viscoso, trabalhando apenas com fluidos invíscidos.

Como outra ideia bastante presente na área de simulação de fluidos e naturalmente aplicável ao contexto multifásico, encontrou-se o controle de volume. Isto é, garantir que o volume de cada fase se mantenha constante ao longo da simulação ou que ele varie conforme desejado. Tendo isso em vista, foram estudadas técnicas e alternativas para controlar o volume durante a simulação, tendo elas servido de base para elaboração do método proposto.

Logo, optou-se por abordar o controle do volume através da evolução das interfaces. De fato, para corrigir o erro do volume de uma fase, o deslocamento de sua superfície é realizado de modo a compensar o erro volumétrico. Ainda, esse movimento está de acordo com o sentido de locomoção da fase, que não será simplesmente inflada ou desinflada omnidirecionalmente.

Para a concretização da pesquisa desenvolvida, criou-se uma aplicação utilizando a linguagem C/C++ em conjunto com a tecnologia CUDA [31]. Nesse contexto, cabe ressaltar que, com o emprego desta, apenas se procurou acelerar minimamente a simulação, de modo que não foram estudados métodos de otimizar a implementação realizada, o que é deixado como sugestão de trabalho futuro. Por fim, para fins da visualização do resultado da simulação, adotou-se o OpenGL [32] para uma visão parcial e testes, e o Blender [33] em conjunto com o *raytracer* Yafaray [34] para a renderização final.

1.7 Descrição

No capítulo 2 serão apresentadas as equações que regem o comportamento dos fluidos e são elucidados os termos que as compõem. No capítulo 3, expõe-se como é realizada a simulação numérica das equações apresentadas no capítulo anterior dentro do contexto euleriano para simulações onde há apenas uma fase. No capítulo 4 é explicado o método tradicional para acompanhar o deslocamento e a forma de superfícies, assim como aquele adotado para tratar diferentes fases. Em seguida, no capítulo 5 são tratadas as modificações que devem ser realizadas na abordagem descrita no capítulo 3 para resolver a equação dos fluidos, a fim de possibilitar seu emprego em sistemas multifásicos. Já no capítulo 6, abrange-se o assunto de controle de volume, passando-se por uma exposição de métodos existentes até a especificação da técnica proposta. No capítulo 7, são discutidos o processo e as ferramentas utilizados para a visualização do resultado da simulação. Por fim, nos capítulos 8 e 9 são apresentadas, respectivamente, conclusões do projeto realizado e propostas para trabalhos futuros.

Capítulo 2

Equações dos Fluidos

A dinâmica dos fluidos incompressíveis é regida por um par de equações diferenciais, as equações de Navier-Stokes para fluidos incompressíveis. A primeira delas é a equação de variação do momento linear:

$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} + \frac{1}{\rho} \nabla p = \vec{g} + \nu \nabla \cdot \nabla \vec{u}. \tag{2.1}$$

A outra é a equação da incompressibilidade:

$$\nabla \cdot \vec{u} = 0, \tag{2.2}$$

onde $\vec{u}$ é a velocidade, p a pressão, ρ a densidade, $\vec{g}$ aceleração da gravidade, e ν o coeficiente de viscosidade cinemática.

Neste capítulo, abordam-se as equações de Navier-Stokes de modo intuitivo para a familiarização com o contexto da dinâmica dos fluidos e para a melhor compreensão dos capítulos seguintes. Para uma explicação mais rigorosa, o livro do BRIDSON [35] pode ser consultado. Assim, iniciando pela equação 2.1, que traduz como um fluido é acelerado de acordo com as forças que atuam sobre ele, inicialmente serão analisadas as forças presentes.

De início, considera-se um sistema de partículas onde o fluido é composto de elementos de massa m, volume V, velocidade $\vec{u}$ e densidade ρ. Nesse contexto, a partir da segunda lei de Newton para sistemas onde a massa é constante, pode-se escrever:

$$m \frac{D\vec{u}}{Dt} = \vec{F}, \tag{2.3}$$

onde $\vec{F}$ é a resultante de todas as forças que atuam no fluido e o operador $\dfrac{D\vec{u}}{Dt}$ representa a derivada total em relação ao tempo.

Dentre as forças presentes, a da gravidade é a mais simples:

$$\vec{F}_g = m\vec{g}. \tag{2.4}$$

Já as demais forças estão relacionadas com o modo como as partículas interagem entre si. Ao certo, as variações de pressão dentro de um fluido fazem com que elas sejam movidas de áreas de alta pressão para àquelas de baixa pressão. Desse modo, pode-se definir a força advinda das diferenças de pressão através do negativo do gradiente da pressão, $-\nabla p$, que aponta para regiões de menor pressão. Além disso, é necessário integrar o gradiente sobre todo o volume do elemento para obter a força resultante. Assumindo que ele seja suficientemente pequeno para que se possa considerar que ∇p é constante dentro dele, apenas se multiplica o gradiente pelo próprio volume V, obtendo:

$$\vec{F}_p = -V\nabla p. \tag{2.5}$$

Como a última força presente, encontra-se a força viscosa, aquela que faz o fluido resistir a deformações localizadas. A grosso modo, pode-se definir esta força como aquela que faz com que cada partícula se mova com velocidade próxima àquelas ao seu redor. Assim, a força viscosa procura reduzir a diferença entre as velocidades de partículas próximas. Tendo isso em vista, é razoável utilizar o operador do Laplaciano, $\nabla \cdot \nabla$, para medir o quanto a velocidade de uma partícula difere da média em suas proximidades. Novamente, integra-se sobre o volume, chegando a:

$$\vec{F}_v = V\eta\nabla \cdot \nabla\vec{u}, \tag{2.6}$$

onde η é um coeficiente de viscosidade dinâmica.

Logo, juntando as equações 2.4, 2.5, 2.6 com 2.3, obtém-se a equação que define como um elemento de fluido se move:

$$m\frac{D\vec{u}}{Dt} = m\vec{g} - V\nabla p + V\eta\nabla \cdot \nabla\vec{u}. \tag{2.7}$$

Em seguida, divide-se a equação 2.7 pelo volume V, obtendo uma equação que depende da densidade e não da massa ou do volume. Então, o resultado é dividido pela densidade e rearranjado:

$$\rho\frac{D\vec{u}}{Dt} = \rho\vec{g} - \nabla p + \eta\nabla \cdot \nabla\vec{u} \longrightarrow \frac{D\vec{u}}{Dt} + \frac{1}{\rho}\nabla p = \vec{g} + \nu\nabla \cdot \nabla\vec{u}, \tag{2.8}$$

onde $\nu = \eta/\rho$ é um coeficiente de viscosidade cinemática.

O passo seguinte é calcular o termo $\frac{D\vec{u}}{Dt}$, comumente chamado de derivada material. Para uma grandeza q dependente da localização $\vec{x}$ e do tempo t, calcula-se, pela regra da cadeia

$$\frac{Dq}{Dt} = \frac{\partial q}{\partial t} + \frac{\partial q}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial q}{\partial y}\frac{\partial y}{\partial t} + \frac{\partial q}{\partial z}\frac{\partial z}{\partial t} = \frac{\partial q}{\partial t} + u\frac{\partial q}{\partial x} + v\frac{\partial q}{\partial y} + w\frac{\partial q}{\partial z} = \frac{\partial q}{\partial t} + \vec{u}\cdot\nabla q, \quad (2.9)$$

onde (u, v, w) são as coordenadas da velocidade $\vec{u}$ nas direções dos eixos de x, y e z, respectivamente.

Para o caso da velocidade, tem-se que:

$$\frac{D\vec{u}}{Dt} = \begin{bmatrix} Du/Dt \\ Dv/Dt \\ Dw/Dt \end{bmatrix} = \begin{bmatrix} \partial u/\partial t + \vec{u}\cdot\nabla u \\ \partial v/\partial t + \vec{u}\cdot\nabla v \\ \partial w/\partial t + \vec{u}\cdot\nabla w \end{bmatrix} = \frac{\partial \vec{u}}{\partial t} + \vec{u}\cdot\nabla\vec{u}. \quad (2.10)$$

Por fim, fazendo a substituição indicada em 2.10 na equação 2.8, reproduz-se a equação do momento (equação 2.1).

Em relação à equação da incompressibilidade, que garante a manutenção do volume V de qualquer porção do fluido em movimento, pode-se trabalhar com uma região arbitrária Ω com superfície $\partial\Omega$. Para ela, a variação de volume é definida como:

$$\frac{dV}{dt} = \iint_{\partial\Omega} \vec{u}\cdot\hat{N}, \quad (2.11)$$

onde $\hat{N}$ é um vetor unitário normal à superfície. Intuitivamente, essa equação pode ser entendida considerando que apenas o componente normal à superfície pode fazer com que uma esfera de fluido aumente de volume, enquanto os componentes tangenciais somente poderiam modificar sua forma.

Para o caso de fluidos incompressíveis, o volume não deve variar com o tempo de modo que:

$$\iint_{\partial\Omega} \vec{u}\cdot\hat{N} = 0. \quad (2.12)$$

Aplicando o teorema de Gauss, obtém-se que, para todo Ω, i.e., para toda região do fluido, em particular para volumes arbitrariamente pequenos, sempre deve ser verdadeiro que:

$$\iint_{\partial\Omega} \vec{u}\cdot\hat{N} = \iiint_{\Omega} \nabla\cdot\vec{u} = 0. \quad (2.13)$$

Logo, assumindo que $\nabla\cdot\vec{u}$ é uma função continua, $\nabla\cdot\vec{u} = 0$ em qualquer ponto do fluido, como indica a equação da incompressibilidade (equação 2.2).

Capítulo 3

Simulação Numérica

Após entender a composição das equações de Navier-Stokes (equações 2.1 e 2.2), passa-se a analisar como as utilizar para simular numericamente o comportamento dos fluidos. Assim, neste capítulo, são apresentadas as discretizações e os métodos numéricos envolvidos.

3.1 Discretização Espacial

De inicio, para obter uma aproximação suficientemente precisa de funções que variam continuamente, como a velocidade e a pressão, elas são avaliadas em um conjunto de pontos discretos no espaço. Para tanto, é utilizada a estrutura da grade MAC (HARLOW e WELCH [36]) com células cubicas. Esta é uma grade dita *staggered*, isto é, que possui variáveis avaliadas em locais distintos dentro de cada célula. De modo geral, componentes de variáveis vetoriais são amostrados no centro das faces de cada célula enquanto as propriedades escalares se encontram no centros das células. Para o caso da velocidade, cada componente paralelo a um dos eixos ordenados é determinado no centro das faces perpendiculares a esse eixo. No caso da pressão, cada célula tem seu valor amostrado em seu centro. Para uma melhor visualização, a figura 3.1 demonstra o posicionamento de componentes de velocidade e da pressão em uma célula da grade de índice (i, j, k), onde i, j e k são, respectivamente, as coordenadas discretas relativas aos eixos x, y e z.

A grade propriamente dita é definida ao indicar: dois pontos, $(x_{max}, y_{max}, z_{max})$ de coordenadas máximas e $(x_{min}, y_{min}, z_{min})$ de mínimas, que definem a caixa alinhada com os eixos ordenados que delimita a grade; e o número de células por dimensão. Através desta definição, é possível saber em qual célula (i, j, k) se encontra uma posição (x, y, z) no espaço: (x, y, z)Coordenadas da posição arbitrária no espaço $\vec{x}$, referentes aos eixos x, y e z

$$célula(x, y, z) = (\lfloor \frac{x - x_{min}}{\Delta x} \rfloor, \lfloor \frac{y - y_{min}}{\Delta x} \rfloor, \lfloor \frac{z - z_{min}}{\Delta x} \rfloor), \tag{3.1}$$

onde a função $\lfloor f \rfloor$ retorna o maior número inteiro menor ou igual a f e Δx é o tamanho de cada célula.

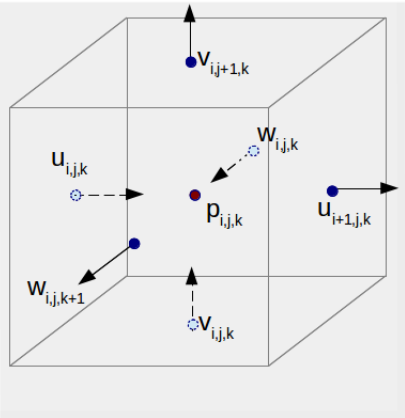

Figura 3.1: Posicionamento dos componentes da velocidade e da pressão em uma célula de índice (i, j, k) da grade MAC

Além disso, é possível calcular a posição onde algum valor é amostrado, como por exemplo, o centro da célula (i, j, k):

$$centro(i, j, k) = (x_{min} + (i + 0.5)\Delta x, y_{min} + (j + 0.5)\Delta x, z_{min} + (k + 0.5)\Delta x). \quad (3.2)$$

Para o efetivo armazenamento dos valores avaliados de funções escalares e vetoriais, são empregados diferentes *arrays*. No caso escalar, como o relativo à pressão, em uma grade tridimensional de tamanho $(d_x) \times (d_y) \times (d_z)$, é utilizado um *array* de dimensão $(d_x) \times (d_y) \times (d_z)$ cujo índice referente ao valor da célula (i, j, k) é dado por:

$$Índice(\ p_{i,j,k}\ ) = i + j(d_x) + k(d_y)(d_z). \quad (3.3)$$

Em se tratando da velocidade, usam-se três *arrays* de dimensões $(d_x + 1) \times (d_y) \times (d_z)$, $(d_x) \times (d_y + 1) \times (d_z)$ e $(d_x) \times (d_y) \times (d_z + 1)$, respectivamente para os componentes u, v e w referentes aos eixos x, y e z. A necessidade de uma posição adicional na direção dos componentes pode ser entendida ao tratar a figura 3.1 como uma grade em que $(d_x, d_y, d_z) = (1, 1, 1)$. Nesse sentido, é preciso guardar dois valores para cada dimensão. Logo, o índice de cada componente em seu respectivo *array* é obtido através de:

$$\begin{aligned} Índice(\ u_{i,j,k}\ ) &= i + j(d_x + 1) + k(d_y)(d_x + 1) \\ Índice(\ v_{i,j,k}\ ) &= i + j(d_x) + k(d_y + 1)(d_x) \\ Índice(\ w_{i,j,k}\ ) &= i + j(d_x) + k(d_y)(d_x). \end{aligned} \quad (3.4)$$

De modo geral, a grande vantagem desta discretização é permitir o cálculo do divergente da velocidade e do gradiente da pressão através de diferenças centrais

nos pontos em que eles são necessários. Ao certo, pode-se calcular o divergente da velocidade no centro de uma célula (i, j, k) como:

$$\nabla \cdot \vec{u}_{i,j,k} = \frac{u_{i+1,j,k} - u_{i,j,k}}{\Delta x} + \frac{v_{i,j+1,k} - v_{i,j,k}}{\Delta x} + \frac{w_{i,j,k+1} - w_{i,j,k}}{\Delta x}, \tag{3.5}$$

e o gradiente da pressão no centro da face entre as células (i, j, k) e $(i + 1, j, k)$, onde é amostrado o componente $u_{i+1,j,k}$, é dado por:

$$\nabla p_{(i,j,k),(i+1,j,k)} = \left(\frac{p_{i+1,j,k} - p_{i,j,k}}{\Delta x} \right). \tag{3.6}$$

Contudo, para obter a velocidade em um ponto arbitrário, é necessário realizar uma interpolação entre os componentes amostrados na faces da célula em que o ponto se encontra. Assim, para um ponto $(x_p, y_p, z_p) \in \mathbb{R}^3$ localizado em uma célula $(i, j, k) \in \mathbb{Z}^3$ de dimensões $\Delta x \times \Delta x \times \Delta x$ e cujo vértice de menores coordenadas é (x_i, y_j, z_k), pode-se obter o valor da sua velocidade através de uma interpolação linear para cada componente em suas respectivas direções:

$$\begin{aligned} \vec{u}(x_p, y_p, z_p) = (&u_{i,j,k}(1 - m_x) + u_{i+1,j,k}(m_x), \\ &v_{i,j,k}(1 - m_y) + v_{i,j+1,k}(m_y), \\ &w_{i,j,k}(1 - m_z) + w_{i,j,k+1}(m_z)), \\ m_x = \frac{x_p - x_i}{\Delta x}, &m_y = \frac{y_p - y_j}{\Delta x}, m_z = \frac{z_p - z_k}{\Delta x}. \end{aligned} \tag{3.7}$$

Esse computo pode ser simplificado quando se deseja o vetor velocidade no centro da célula ou no centro de alguma face. Nesse caso, apenas é feita uma média entre os valores calculados em pontos próximos. Por exemplo, para obter a velocidade no centro da célula, valor que futuramente será utilizado para movimentar a superfície do fluido, utiliza-se:

$$\vec{u}_{i,j,k} = \left(\frac{u_{i+1,j,k} + u_{i,j,k}}{2}, \frac{v_{i,j+1,k} + v_{i,j,k}}{2}, \frac{w_{i,j,k+1} + w_{i,j,k}}{2} \right). \tag{3.8}$$

Ainda, para calcular valores escalares como a pressão, também é feita uma interpolação para encontrar valores em posições arbitrárias, mas utilizando os oito valores obtidos nos oito centros das células mais próximas. Assim, é feita uma interpolação trilinear entre tais valores.

Por fim, além dos valores referentes a funções escalares e vetoriais, essencialmente a pressão e a velocidade, é atribuído a cada célula um identificador do material existente em seu centro. Este pode indicar se uma célula é sólida ou se ela é de determinado fluido. Ademais, são distinguidas células de entrada e saída do conjunto de células fluidas. Naturalmente, as primeiras representam por onde um fluido entra

na grade com uma velocidade controlada enquanto as seguintes denotam regiões por onde um fluido pode sair. Nesse sentido, a figura 3.2 ilustra uma possível configuração em uma grade bidimensional.

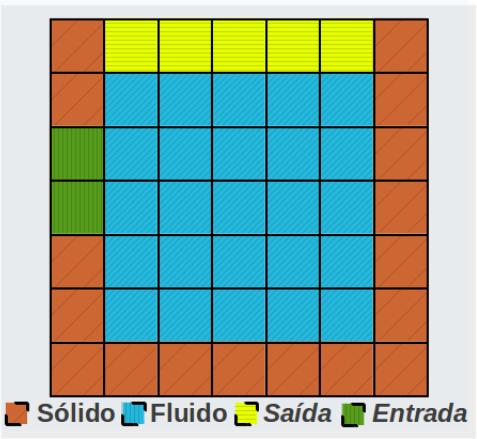

Figura 3.2: Exemplo de configuração dos tipos de célula em uma grade bidimensional.

3.2 Discretização Temporal

Após haver descrito como se amostram as propriedades do fluido em posições discretas no espaço, será analisada a discretização de tais valores no tempo. Ao certo, posições das interfaces e valores de velocidade e pressão de uma dada configuração são atualizados conforme a equação de Navier-Stokes 2.1 a cada passo de tempo t, cujo tamanho é Δt. A abordagem adotada para isso é o *splitting*, que pode ser entendido intuitivamente como uma aplicação do princípio "dividir para conquistar"(do inglês *divide and conquer*) em equações diferenciais, onde se divide a equação em outras menores que possuem métodos conhecidos de resolução. Por exemplo, ao invés de resolver diretamente a equação:

$$\frac{\partial q}{\partial t} = f(q) + g(q), \tag{3.9}$$

resolvem-se duas equações mais simples:

$$\frac{\partial q}{\partial t} = f(q^t), \tag{3.10}$$

$$\frac{\partial q}{\partial t} = g(q^*) \tag{3.11}$$

Logo, pode-se chegar ao resultado desejado através de:

$$q^* = F(\Delta t, q^t) \tag{3.12}$$

$$q^{t+1} = G(\Delta t, q^*), \tag{3.13}$$

onde F e G são algoritmos de integração, cada um específico para resolver, respectivamente, as equações 3.10 e 3.11.

Aplicando essa metodologia à equação de Navier-Stokes 2.1, ela será resolvida pela sequência:

- Difusão viscosa;

$$\frac{\partial \vec{u}}{\partial t} = \frac{1}{\rho}\nu\nabla \cdot \vec{u}, \tag{3.14}$$

onde se modifica o campo velocidade em um ponto de modo a aproximá-lo da média de sua vizinhança.

- Advecção:

$$\frac{Dq}{Dt} = 0, \tag{3.15}$$

onde se difundem uma propriedade q pelo fluido apenas em função de seu próprio movimento. Tipicamente, q se refere à velocidade, mas também poderia ser, por exemplo, a concentração de um pigmento que se espalha pelo fluido.

- Forças externas:

$$\frac{\partial \vec{u}}{\partial t} = \vec{g}, \tag{3.16}$$

onde será aplicada ao campo velocidade não apenas a força da gravidade como qualquer outra força externa ao fluido.

- Projeção:

$$\begin{gathered} \frac{\partial \vec{u}}{\partial t} + \frac{1}{\rho}\nabla p = 0, \\ \text{de modo que } \nabla \cdot \vec{u} = 0, \end{gathered} \tag{3.17}$$

onde se obtém uma nova pressão em cada célula de modo que ela garanta um divergente nulo da velocidade e, finalmente, calcula-se essa nova velocidade com divergente nulo.

Desse modo, nas seções seguintes, serão explicadas as etapas acima, além de um método para estabelecer o tamanho do passo Δt utilizado em cada iteração a fim de garantir a estabilidade numérica do sistema. Além destas etapas, é necessário acompanhar o movimento da superfície do fluido, mas como este assunto faz parte do foco principal deste trabalho, ele é discutido em um capítulo à parte (capitulo 4)

3.3 Difusão Viscosa

No começo da simulação, a equação 3.14 seria resolvida para que fosse dado comportamento viscoso ao fluido. Sua atuação faz com que cada porção do fluido se mova com uma velocidade compatível com a média ao seu redor, evitando grandes variações locais. Essa propriedade pode ser de grande importância na simulação, especialmente ao se tratar de fluidos intrinsecamente viscosos, tal como o mel.

Contudo, neste trabalho, o comportamento viscoso é desconsiderado. Trabalha-se, então, apenas com fluidos invíscidos (i.e., sem viscosidade) ou onde essa propriedade é pouco presente. Neste último caso, o investimento computacional para considerá-la não compensa a pequena perda colateral no realismo da simulação. Porém, também se deve lembrar que os erros introduzidos pela simulação numérica podem ser reinterpretados como uma viscosidade indesejada (BRIDSON [35]). Isto é, mesmo que se trabalhe com fluidos não viscosos, erros numéricos advindos do processo de simulação acabam por conferir algum comportamento viscoso aos fluidos.

Portanto, ao invés de trabalhar com a equação do momento como foi exposta na equação 2.1, trabalha-se, então, simplesmente com:

$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} + \frac{1}{\rho} \nabla p = \vec{g}, \tag{3.18}$$

que, em conjunto com a equação da incompressibilidade (equação 2.2) forma o conjunto das equações de Euler para fluidos incompressíveis.

3.4 Advecção

Dando início à simulação numérica do comportamento do fluido, resolve-se a equação 3.15 para mover as propriedades desejadas dentro do fluido. Cabe, no entanto, deixar claro que a advecção deve ser realizada sobre um campo de velocidade com divergente nulo. Desse modo, se for realizada antes da advecção alguma operação que torne o valor do divergente não nulo, ela deve ser precedida de uma etapa adicional de projeção para anular os divergentes.

Entretanto, se a difusão viscosa fosse a operação a ser executada antes, outra solução também poderia ser adotada. Poder-se-ia advectar as propriedades desejadas, inclusive o próprio campo de velocidade resultante do comportamento viscoso, utilizando as velocidades anteriores à fase da difusão viscosa. Contudo, apesar desse método ser menos custoso do que realizar uma nova etapa de projeção, pode apresentar significativa perda de qualidade.

No momento, apenas o caso da velocidade é analisado. Contudo, o método exposto é naturalmente aplicável para qualquer outra característica. No caso da ve-

locidade, como cada um de seus componentes é amostrado em uma posição distinta, advecta-se cada um deles de modo independente. Para cada componente, calcula-se, por interpolação, o vetor velocidade $\vec{u}$ em sua posição e se computa o novo valor do componente pelo método de advecção escolhido, empregando o vetor $\vec{u}$.

Para realizar a advecção, inicialmente, observa-se que ela é automática no contexto lagrangiano. De fato, uma partícula carrega consigo propriedades cujos valores não são alterados pela advecção. Assim, pode-se empregar um método que se beneficie desse cenário para resolver a equação 3.15, constituindo a advecção semi-lagrangiana (STAM [10]).

Nesse contexto, para encontrar o valor que é advectado para uma localização $\vec{x}$ no instante t, procura-se descobrir a partícula imaginária que se teria se movido para $\vec{x}$ em t e se verifica o valor da propriedade desejada em sua posição original. Logo, utilizando o esquema *forward euler* para a integração, a expressão que define a velocidade $\vec{u}_1$ após a etapa da advecção é dada por:

$$\vec{u}_1(\vec{x}) = \vec{u}^{t-1}(\vec{x} - \vec{u}^{t-1}(\vec{x})\Delta t). \tag{3.19}$$

Alternativamente, pode-se empregar um esquema do tipo Runge-Kutta, que se mostra como uma alternativa mais robusta, apresentando resultados que podem ser significativamente melhores. Contudo, como seu custo é maior do que aquele do método semi-lagrangiano, esta técnica é utilizada apenas para a evolução da superfície. Cabe lembrar que, neste trabalho, o foco é dado na visualização da dinâmica como um todo, dando mais importância à evolução das superfícies em detrimento da corretude do que acontece no interior do fluido. Assim, para uma propriedade q amostrada em $\vec{x}$ e que define implicitamente a posição da superfície, a advecção segundo um método Runge-Kutta de segunda ordem pode ser definido como:

$$\begin{aligned} \vec{x}_{meio} &= \vec{x} - \frac{1}{2}\vec{u}^{t-1}(\vec{x})\Delta t \\ \vec{x}_p &= \vec{x} - \vec{u}^{t-1}(\vec{x}_{meio})\Delta t \\ q^t(x) &= q^{t-1}(\vec{x}_p). \end{aligned} \tag{3.20}$$

De modo geral, este método utiliza a velocidade em um ponto intermediário para chegar ao ponto de origem da suposta partícula que se moveu para $\vec{x}$. Dependendo da magnitude do Δt, também é possível dividir a trajetória da partícula em mais sub-passos.

3.5 Forças Externas

Em seguida, resolve-se a equação 3.16 para determinar a influência de n forças externas, obtendo as velocidades $\vec{u}_2$ através da equação:

$$\vec{u}_2 = \vec{u}_1 + \Delta t \sum_{i=0}^{n} \vec{a}_i, \tag{3.21}$$

onde Δt é o tamanho do passo de tempo atual t, $\vec{a}_i$ é a aceleração relativa à i-ésima força externa atuante, $\vec{u}_1$ é o vetor velocidade dado após a etapa da advecção.

3.6 Projeção

Por fim, utiliza-se a velocidade $\vec{u}_2$, obtida após os passos de advecção e aplicação de forças externas, para resolver a equação 3.17 e chegar à velocidade após o passo t. Inicialmente, discretiza-se a derivada da velocidade em relação ao tempo:

$$\frac{\vec{u}^t - \vec{u}_2}{\Delta t} + \frac{1}{\rho}\nabla p^t = 0, \tag{3.22}$$

e se aplica o operador do divergente a esta última equação:

$$\nabla \cdot \left(\frac{\nabla p^t}{\rho} \right) = -\nabla \cdot \left(\frac{\vec{u}^t - \vec{u}_2}{\Delta t} \right). \tag{3.23}$$

Como se procura obter $\vec{u}^t$ tal que $\nabla \cdot \vec{u}^t = 0$, tem-se então:

$$\frac{\Delta t}{\rho}\nabla^2 p^t = \nabla \cdot \vec{u}_2 \tag{3.24}$$

Logo, calcula-se p^t resolvendo essa equação de Poisson, o que é feito usando métodos numéricos conhecidos, como o gradiente conjugado, que é exposto na seção 3.9. Finalmente, é calculada a velocidade de divergente nulo após o passo de tempo t:

$$\vec{u}^t = \vec{u}_2 - \Delta t \frac{\nabla p^t}{\rho}, \tag{3.25}$$

que será utilizada como entrada para a realização da iteração seguinte.

Note que, na implementação, como cada componente da velocidade é guardado em posições distintas, a equação acima é calculada de modo diferente para cada componente distinto. Além disso, a forma como os componentes da velocidade e os valores de pressão são guardados torna fácil o cálculo da expressão anterior para cada componente.

3.7 Condições de Borda

Uma preocupação recorrente no que refere à simulação numérica de fluidos é o tratamento das fronteiras do fluido simulado. Por exemplo, a etapa da advecção semi-lagrangiana pode requerer o cálculo de uma propriedade em uma posição que se encontre ou fora dos limites da grade, ou em outro material como o ar, o vácuo ou um sólido qualquer. Nesses casos, a abordagem padrão é extrapolar o valor da propriedade a partir da posição mais próxima dentro do fluido.

Diversos trabalhos já estudaram como realizar tal extrapolação. Ao certo, PENG *et al.* [37] utilizaram um método baseado em equações diferencias parciais para extrapolar quaisquer propriedades, ADALSTEINSSON e SETHIAN [38] criaram velocidades artificiais durante a execução do método *Fast Marching* (SETHIAN [39]), usado no cálculo de um campo de distância à superfície.

Contudo, neste trabalho utiliza-se uma abordagem mais simples em que se emprega o valor da posição mais próxima dentro da grade. Assim, para obter uma quantidade em um ponto externo à grade, projeta-se tal ponto na borda da grade e se toma o valor neste novo ponto. Porém, para o caso da velocidade em células sólidas, utilizam-se os valores da célula fluida mais próxima. Já em células de entrada, os valores dos componentes da velocidade são sempre iguais a valores previamente escolhidos.

Além desses casos gerais, existem casos específicos que também devem ser considerados. De início, a pressão fora da grade é determinada por uma condição de fronteira de Dirichlet, onde se estabelece que $p = 0$ fora da grade. Isto equivale a estabelecer que exista vácuo fora do ambiente de simulação, deixando o fluido se mover livremente para essas regiões.

Caso se deseje que o fluido não saia da grade, utilizam-se paredes formadas de células sólidas. Neste trabalho, consideram-se especialmente duas configurações das células sólidas. Uma em que o fluido é completamente cercado por paredes sólidas e outra em que apenas não existe parede na parte superior da grade, o que deixa o fluido livre para sair dos limites da simulação pela parte de cima. Ademais, esse tipo de célula também pode constituir obstáculos de conformação genérica dentro de regiões de fluido.

Ao utilizar células sólidas, se faz necessário estabelecer uma restrição para a velocidade na fronteira sólido-fluido para que se impossibilite que o fluido flua para dentro de uma célula sólida. Portanto, utiliza-se a seguinte condição de fronteira de Neumann (condição *no-stick* ou *slip*):

$$\vec{u}^t \cdot \hat{N}^t = \vec{u}^t_{sólido} \cdot \hat{N}^t = 0, \tag{3.26}$$

que é uma restrição sobre o componente da velocidade $\vec{u}^t$ no instante t paralelo à

direção normal à superfície. Segundo ela, este componente deve ser igual à projeção da velocidade do sólido na normal $\hat{N}$. Ademais, como, neste trabalho, os sólidos estão parados, na prática, apenas são zerados os componentes de velocidade estimados nas faces entre uma célula sólida e uma fluida.

Além disso, também é preciso estabelecer o valor da pressão dentro dessas células antes de resolver a equação 3.17. Tendo isso em vista, calcula-se a pressão dentro de uma célula sólida com base em uma célula de fluido vizinha. Por exemplo, supondo que a célula (i, j, k) seja fluida e que $(i + 1, j, k)$ seja sólida, a atualização devido à pressão do componente u do campo velocidade amostrada na face entre as duas células seria dada por:

$$u_{i+1}^t = u_{2,\,i+1} - \frac{\Delta t}{\rho} \frac{p_{i+1}^t - p_i^t}{\Delta x}, \tag{3.27}$$

Ainda, como pela equação 3.26 é verdadeiro que:

$$u_{i+1}^t = \vec{u}_{sólido}^t = 0, \tag{3.28}$$

é possível calcular a pressão dentro do sólido pela expressão:

$$p_{i+1}^t = p_i^t + \frac{\rho \Delta x}{\Delta t} u_{2,\,i+1} \tag{3.29}$$

Ademais, ao forçar que $u_{2,\,i+1}$ sempre seja igual a zero nas fronteiras sólido-liquido, obtém-se, então, que:

$$p_{i+1}^t = p_i^t. \tag{3.30}$$

Isto é, quando for preciso tomar o valor de pressão no centro de uma célula sólida, utiliza-se o valor da célula fluida adjacente. Adicionalmente, seguindo o exemplo anterior, mas fazendo com que a célula adjacente $(i + 1, j, k)$ fosse uma entrada também se pode inferir 3.30 a partir da equação 3.27 pois os valores u_{i+1}^t e $u_{2,\,i+1}$ são iguais à uma constante pré-determinada.

Finalmente, note que neste trabalho apenas são consideradas transições liquido-sólido nas faces das células, que são sempre quadrados. Entretanto, existem muitos estudos que consideram limites sólidos irregulares, à exemplo dos trabalhos de BATTY *et al.* [40, 41].

3.8 Sistema de Equações de Poisson

Antes de partir para a resolução da equação 3.24, cabe melhor entender o sistema a ser resolvido. De fato, se essa equação for discretizada para uma célula, obtêm-se:

$$\frac{\Delta t}{\rho \Delta x^2}\begin{pmatrix} p_{i+1,j,k} + p_{i,j+1,k} + p_{i,j,k+1} \\ +p_{i-1,j,k} + p_{i,j-1,k} + p_{i,j,k-1} \\ -6p_{i,j,k} \end{pmatrix} = \frac{1}{\Delta x}\begin{pmatrix} u_{i+1,j,k} - u_{i,j,k} + v_{i,j+1,k} \\ -v_{i,j,k} + w_{i,j,k+1} - w_{i,j,k} \end{pmatrix}. \quad (3.31)$$

Tendo discretizado a equação, inicialmente observa-se a aplicação das condições de borda, tal como apresentadas na seção 3.7. À exemplo, quando a célula (i, j, k) tem seu vizinho $(i+1, j, k)$ como uma célula sólida, de entrada ou saída, $(i, j+1, k)$ estiver fora da grade de simulação, e as demais células adjacentes forem fluidas, o primeiro termo da equação anterior seria:

$$\frac{\Delta t}{\rho \Delta x^2}\begin{pmatrix} \mathbf{p_{i,j,k}} + \mathbf{0} + p_{i,j,k+1} + p_{i-1,j,k} \\ +p_{i,j-1,k} + p_{i,j,k-1} - 6p_{i,j,k} \end{pmatrix} = \begin{pmatrix} p_{i,j,k+1} + p_{i-1,j,k} + p_{i,j-1,k} \\ +p_{i,j,k-1} - \mathbf{5}p_{i,j,k} \end{pmatrix}. \quad (3.32)$$

Logo, como regra geral, pode-se inferir que o coeficiente relativo à pressão da célula é igual a quantidade de vizinhos que não são sólidos, entradas ou saídas. Além disso, se alguma célula vizinha estiver fora da grade, pode-se apenas remover seu elemento de pressão da expressão 3.31.

Portanto, ao se considerar toda a grade de simulação, define-se um sistema de equações lineares para os valores desconhecidos de pressão no formato

$$Ap = b. \quad (3.33)$$

Neste modelo, A é a matriz dos pesos das pressões e p e b são matrizes coluna que guardam para cada célula, respectivamente, o valor a ser obtido de pressão e o divergente do campo velocidade.

Note que, para uma célula (i, j, k), sua linha em A possivelmente possui apenas sete valores diferentes de zero, referentes ao seu valor de pressão e ao valor das seis células fluidas vizinhas. Isto faz com que A seja uma matriz esparsa. Além disso, A é simétrica, de modo que apenas é necessário guardar metade dos valores não nulos da matriz. Contudo, para este trabalho, não se armazenam diretamente tais valores uma vez que eles são calculados à medida que são utilizados no método de resolução empregado.

3.9 Método do Gradiente Conjugado

Neste trabalho é aplicada a versão iterativa do método do gradiente conjugado (GC), originalmente apresentado por HESTENES e STIEFEL [42], para resolver o sistema de equações de poisson (equação 3.33). Esta técnica é uma das mais empregadas no cálculo da solução de sistemas lineares, resolvendo o sistema:

$$A_{(n\times n)}x_{(n\times 1)} = b_{(n\times 1)}, \tag{3.34}$$

onde a matriz A é simétrica ($A^T = A$) e positiva-definida ($x^T Ax > 0\,\forall x \mid x \in \mathbb{R}^n$ e $x \neq O$), e b é conhecido.

De modo geral, opta-se por utilizar métodos iterativos para trabalhar com matrizes A esparsas devido a limites de disponibilidade de memória. De fato, o que ocorre é que a utilização de métodos diretos, como a decomposição de Cholesky, usualmente destrói a esparsidade durante o processo de resolução, obrigando que mais valores sejam guardados. Para a explicação desse e outros métodos de resolução, o livro de HEATH [43] pode ser consultado.

Para entender o método do GC, cabe observar inicialmente que, para resolver a equação 3.34, basta minimizar a função:

$$f(x) = \frac{1}{2}x^T Ax - b^T x, \tag{3.35}$$

cuja derivada, se A for uma matriz simétrica, é:

$$f'(x) = Ax - b. \tag{3.36}$$

Logo, para minimizar 3.35, iguala-se 3.36 a zero, retornando ao sistema linear 3.34. Ademais, como A é positiva-definida, pode-se afirmar que x é um mínimo global de f.

Portanto, para resolver o problema de minimização, o GC utiliza um conjunto de direções ortogonais $(d_0, d_1, ..., d_{n-1})$ de modo que seja dado apenas um passo em cada uma dessas direções, terminando o algoritmo em n passos. Nesse sentido, ao fim da busca na direção d_i, é encontrada a solução do problema 3.35, restrito ao subespaço S_i gerado por $(d_0, ..., d_i)$. Por exemplo, na figura 3.3 é ilustrada a convergência do algoritmo em dois passos.

Constituindo uma solução iterativa, em cada passo é atualizado o resultado:

$$x_{i+1} = x_i + \alpha_i d_i. \tag{3.37}$$

Já para o resíduo $r_i = b - Ax_i$, que pode ser entendido como o erro transformado

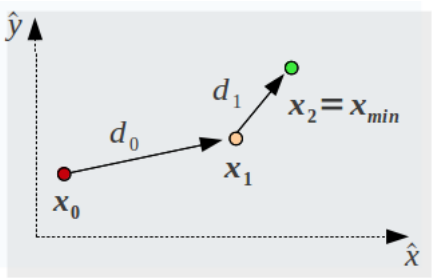

Figura 3.3: Exemplo de convergência do método do gradiente conjugado em duas iterações, onde $S = (d_0, d_1)$ é o subespaço das direções de busca e x_{min} é a posição que minimiza a função objetivo.

para o mesmo espaço que b, utiliza-se a recorrência:

$$r_{i+1} = b - A(x_{i+1}) = b - A(x_i + \alpha_i d_i) = r_i - \alpha_i A d_i. \tag{3.38}$$

Para que x_{i+1} minimize f na reta definida por $x_i + \alpha d_i$, é preciso que a direção de busca d_i seja ortogonal à direção $\nabla f(\vec{x}_{i+1}) = r_{i+1}$: Assim, α pode ser definido como:

$$\begin{aligned} d_i^T(r_{i+1}) &= 0 \\ d_i^T(r_i - \alpha_i d_i) &= 0, \\ \text{de onde: } \alpha_i &= \frac{d_i^T r_i}{d_i^t A d_i}. \end{aligned} \tag{3.39}$$

Ainda, para fazer com que x_{i+1} otimize f em todo o subespaço S_i gerado pelas dimensões $d_0, ..., d_n$, deve-se fazer com que essas direções sejam A-ortogonais. Dois vetores d_i e d_j são A-ortogonais somente se:

$$d_i^T A d_j = 0. \tag{3.40}$$

Desse modo, assumindo que x_i otimiza f em $S_{i-1} = (d_0, ..., d_{i-1})$, então, para todo $j < i$:

$$d_j^T \nabla f(x_i) = 0 \rightarrow d_j^T r_i = 0. \tag{3.41}$$

Isso acarreta que, para todo $j < i$:

$$d_j^T \nabla f(x_{i+1}) = d_j^T(r_i - \alpha_i A d_i) = d_j^T r_i - \alpha d_j^T A d_i = 0, \tag{3.42}$$

pois a primeira parcela se anula por 3.41 e a segunda por d_i e d_j serem A-ortogonais. Além disso, no caso de iterações seguidas:

$$d_i^T \nabla f(x_{i+1}) = d_i^T(r_i - \alpha_i A d_i) = d_i^T r_i - \alpha d_i^T A d_i = 0. \tag{3.43}$$

Portanto, $\nabla f(x_{i+1})$ é ortogonal a todas as direções que geram S_i e, assim, f restrita a esse subespaço será otimizada em x_{i+1}. Por fim, ao assumir $x_0 = 0$ e $S_0 = O$, pode-se começar um processo de indução que prove que, após um número de iterações igual à dimensão do espaço de busca, será atingido o mínimo da função f.

Em vista do exposto acima, é preciso conhecer como calcular um conjunto de direções A-ortogonais. Para tanto, utiliza-se o processo de Gram-Schmidt. Supondo um conjunto $u_0, u_1, ..., u_{n-1}$ de n direções linearmente independentes, constrói-se d_i removendo, de u_i, qualquer componente que não seja A-ortogonal às direções de busca anteriores, de modo que:

$$d_i = \begin{cases} u_0, & \text{se } i = 0 \\ u_i + \sum_{k=0}^{i-1} \beta_{i,k} d_k, & \text{se } i > 0 \end{cases}, \tag{3.44}$$

onde β é definido para $i > k$. Para encontrar seus valores, como as direções de busca são A-ortogonais, é possível eliminar todos os valores da equação 3.44, exceto um, ao multiplicar esta expressão por Ad_j pelo lado direito:

$$0 = d_i^T(Ad_j) = u_i^T(Ad_j) + \sum_{k=0}^{i-1} \beta_{i,k} d_k^T(Ad_j)$$

$$\beta_{i,k} d_j^T Ad_j = -u_i^T Ad_j, \text{ com } i > j$$

$$\beta_{i,j} = -\frac{u_i^T Ad_j}{d_j^T Ad_j}. \tag{3.45}$$

No caso do gradiente conjugado, as direções são construídas pela conjugação dos resíduos de modo que $u_i = r_i$. De fato, pode-se afirmar que o conjunto formado por todos os valores r_i sempre irá produzir um novo conjunto de direções de busca linearmente independente, salvo se o resíduo for zero, pois ele é ortogonal à todas as direções de busca anteriores.

Para simplificar a expressão para o cálculo de $\beta_{i,i-1}$, inicialmente se multiplica a equação 3.38 por r_i^T, obtendo:

$$r_i^T r_{j+1} = r_i^T r_j - \alpha_j r_i^T Ad_j.$$

Quando $i = j + 1$, como resíduos de iterações diferentes são ortogonais, chega-se à:

$$r_i^T Ad_{i-1} = -\frac{r_i^T r_i}{\alpha_{i-1}}. \tag{3.46}$$

Então, juntando a equação anterior com a equação 3.45 se obtém:

$$\beta_{i,i-1} = \frac{r_i^T r_i}{\alpha_{i-1} d_{i-1}^T A d_{i-1}}. \tag{3.47}$$

Utilizando a definição do α da equação 3.39:

$$\beta_{i,i-1} = \frac{r_i^T r_i}{d_{i-1}^T r_{i-1}}. \tag{3.48}$$

Para o passo final da simplificação, inicialmente se multiplica a equação 3.44 por r_j pelo lado direito:

$$d_i^T r_j = u_i^T r_j + \sum_{k=0}^{i-1} \beta_{i,k} d_k r_j. \tag{3.49}$$

Em seguida, como para $k < j$ é verdadeiro que $d_k r_j = 0$ (equação 3.41), então quando $i = j$ se observa que:

$$d_i^T r_i = u_i^T r_i, \tag{3.50}$$

permitindo que, utilizando as equações 3.50 e 3.48, seja possível calcular $\beta_{i,i-1}$ de modo simplificado:

$$\beta_{i,i-1} = \frac{r_i^T r_i}{r_{i-1}^T r_{i-1}}. \tag{3.51}$$

Ademais, a equação 3.44 se torna simplesmente:

$$d_i = r_i + \beta_{i,i-1} d_k. \tag{3.52}$$

Para a execução iterativa, é dado início com uma solução inicial que pode ser arbitrária, iterando sobre ela até atingir a acurácia desejada. Em particular, a solução inicial neste trabalho é dada por um vetor com valores nulos. Já para finalizar o algoritmo, verifica-se quando o resíduo se torna suficientemente pequeno. Ademais, como também se deseja prevenir contra erros numéricos provenientes da utilização da aritmética de ponto flutuante que pode fazer com que o algoritmo nunca convirja exatamente, limitam-se o número de iterações.

Para uma melhor visualização do funcionamento do algoritmo, um pseudo-código de sua execução pode ser visto na figura 3.4. Note que cada iteração apenas envolve as operações de multiplicação de A por um vetor, soma de vetores, multiplicação de vetores por escalares e alguns produtos escalares. Ao certo, todos são simples de implementar, inclusive no contexto paralelo.

Por fim, cabe ressaltar que, como visto, o número de iterações necessárias para que o método convirja é proporcional ao número de células da grade. Porém, além disso, o GC demora mais para convergir de acordo com o quanto a matriz A for diferente da matriz identidade I.

$r_0 = b - Ax_0$
$d_0 = r_0$
$i = 0$
Enquanto($i < i_{max}$ **e** $r_0 < r_{max}$) :
$\alpha_i = \frac{d_i^T r_i}{d_i^T A d_i}$
$x_{i+1} = x_i + \alpha_i d_i$
$r_{i+1} = r_i - \alpha_i A d_i$
$\beta_{i+1,i} = \frac{r_{i+1}^T r_{i+1}}{r_i^T r_i}$
$d_{i+1} = r_{i+1} + \beta_{i+1,i} d_i$
$i = i + 1$
Retornar x_{i+1}

Figura 3.4: Pseudocódigo da execução iterativa do método do gradiente conjugado.

Seguindo a ideia anterior, existe uma modificação do algoritmo, denominada gradiente conjugado precondicionado, que, ao invés de solucionar o sistema $Ap = b$, resolve $MAp = Mb$ para uma matriz M que faça com que MA seja próximo da matriz identidade. Entretanto, neste trabalho apenas é empregado o método do GC conforme foi apresentado pois o precondicionador M é usualmente mais complexo de ser paralelizado, sendo deixado como sugestão de trabalho futuro.

3.10 Tamanho do Passo de Tempo

Antes de iniciar qualquer iteração do algoritmo de simulação, é preciso estabelecer o tamanho Δt do t-ésimo passo de tempo a ser simulado. Antes de mais nada, para construir uma animação, não se deseja que o tempo simulado para uma quadro seja maior do que a sua duração. Assim, quando se quer visualizar a dinâmica de um fluido em tempo real, cabe restringir o valor máximo do passo do tempo ao período de exposição de cada quadro segundo a taxa de amostragem pretendida para a animação. Pode-se, ainda, usar um passo mais curto e fazer múltiplas iterações até que este tempo seja atingido.

Em adição à restrição imposta pela frequência de atualização da animação desejada, também é preciso que Δt satisfaça condições impostas por cada passo da simulação (advecção, aplicação de forças externas , etc.). Contudo, para evitar que tais exigências impliquem um passo do tempo que seja tão pequeno que torne a simulação inviável, também se permite restringir Δt à um valor mínimo. Porém, quando este requisito for utilizado, o resultado pode ser quantitativamente impreciso, mesmo que seja plausível.

As restrições advindas de cada etapa estão diretamente relacionadas com a condição CFL (COURANT *et al.* [44]), sendo necessárias para a convergência da

simulação numérica. Isto é, se tais requisitos forem atingidos e a simulação for realizada com Δt e Δx que tendam a zero, o resultado da resolução numérica deverá se aproximar da solução exata. Não se deve entender, contudo, a condição CFL como uma condição de estabilidade. Certamente, existem métodos que, ou são sempre instáveis, ou são estáveis para valores arbitrários de Δt e convergem apenas se a condição CFL for atingida.

De modo intuitivo, pode ser dado o exemplo da condição CFL para métodos semi-lagrangianos. Nesses casos, a restrição é satisfeita se a trajetória calculada para cada partícula estiver suficientemente próxima da trajetória real. Assim, se Δt for grande, a condição não será obedecida pois apenas estará sendo considerado o campo velocidade na localidade instantânea da partícula e não ao longo da trajetória durante a transição. Todavia, utilizando um Δt suficientemente pequeno, a trajetória será corretamente aproximada por um segmento de reta tangente a ela. Para ilustração, a figura 3.5 mostra a diferença entre as trajetórias real e aproximada para dois intervalos distintos.

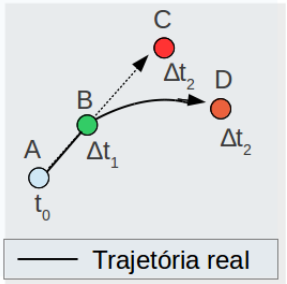

Figura 3.5: Diferença entre a trajetória real (passa por B e atinge D) da partícula inicialmente em A e sua trajetória aproximada (passa por B e atinge C), obtidas pela estimação de sua posição em momentos Δt_1 e Δt_2 distantes do tempo t_0, com $\Delta t_1 < \Delta t_2$.

Apresenta-se, então, a condição CFL como:

$$\Delta t \leq \alpha \frac{\Delta x}{c}, \tag{3.53}$$

onde: c é a velocidade máxima de propagação de informação ou da influência do valor de uma propriedade em um ponto na determinação de qualquer propriedade em outra localidade no instante seguinte. α é o denominado número CFL que, para este trabalho, tem valor igual a 1.

Portanto, se mostra necessário construir uma expressão geral para a condição CFL que seja válida para qualquer instante da simulação e que leve em conta as modificações realizadas durante a simulação no campo velocidade, em especial na velocidade máxima. Para chegar a essa expressão, utiliza-se, então, a abordagem empregada por KANG *et al.* [28]. De início, devido a etapa de advecção, que não

aumenta a velocidade máxima, tem-se que a condição CFL será atendida se:

$$\Delta t \left(\frac{|u|_{max} + |v|_{max} + |w|_{max}}{\Delta x} \right) \leq 1, \tag{3.54}$$

onde $|u|_{max}$, $|v|_{max}$, $|w|_{max}$ são as magnitudes máximas dos respectivos componentes de velocidade amostrados na grade.

Tendo a gravidade como a única força externa considerada, sua atuação impõe uma restrição sobre a magnitude máxima do componente v da velocidade no passo t:

$$|v|^t_{max} = |v|_{max} + |g|\Delta t. \tag{3.55}$$

Interpretando a equação 3.53 como $\Delta t \leq \dfrac{\Delta x}{|v|^t_{max}}$ e substituindo Δt pelo seu limitante à direita da desigualdade, é possível obter um novo limite superior para a velocidade definida pela equação 3.55:

$$|v|^t_{max} \leq |v|_{max} + \frac{g\Delta x}{|v|^t_{max}}$$

$$\left(|v|^t_{max}\right)^2 - |v|_{max}(|v|^t_{max}) - g\Delta x \leq 0$$

$$|v|^t_{max} = \frac{|v|_{max} + \sqrt{|v|^2_{max} + 4g\Delta x}}{2}. \tag{3.56}$$

Isso implica a condição:

$$\frac{\Delta t}{2} \left(\frac{|v|_{max}}{\Delta x} + \sqrt{\frac{|v|^2_{max}}{\Delta x^2} + \frac{4g}{\Delta x}} \right) \leq 1, \tag{3.57}$$

Generalizando o caso anterior para quando uma força resultante $\vec{F} = (F_x, F_y, F_z)$ atua sobre o fluido, e definindo:

$$A_{cfl} = \frac{|u|_{max} + |v|_{max} + |w|_{max}}{\Delta x}, \tag{3.58}$$

pode-se chegar a conclusão que a condição CFL é atingida ao se obter:

$$\frac{\Delta t}{2} \left(A_{cfl} + \sqrt{A^2_{cfl} + \frac{4|F_x|}{\Delta x} + \frac{4|F_y|}{\Delta y} + \frac{4|F_z|}{\Delta z}} \right) \leq 1. \tag{3.59}$$

Além da gravidade, quando se trabalha com dois meios distintos também é levada em consideração a força de tensão superficial. Como será retratado no capítulo 5, esta força tem magnitude $\dfrac{\sigma\kappa}{\rho\Delta x^2}$, onde σ é o coeficiente de tensão superficial entre os dois materiais e κ é a curvatura da interface. . Assim, considerando a tensão superficial para o caso em que a gravidade é a única força externa, o tamanho do

passo do tempo pode ser calculado através da seguinte condição CFL:

$$\Delta t \left(\frac{A_{cfl} + \sqrt{A_{cfl}^2 + 4G_{cfl}^2 + 4S_{cfl}^2}}{2} \right) \leq 1, \tag{3.60}$$

$$\text{com } G_{cfl} = \sqrt{\frac{|g|}{\Delta x}}, \text{ e } S_{cfl} = \sqrt{\frac{\sigma\kappa}{min(\rho_{fluido^1}, ..., \rho_{fluido^n})\Delta x^2}},$$

onde S_{cfl} é um termo adicional devido à consideração da tensão superficial.

Capítulo 4

Evolução da Superfície

4.1 *Level Set*

Métodos *level set*, originalmente idealizados por OSHER e SETHIAN [45], são técnicas bastante conhecidas para o acompanhamento de interfaces, sendo aplicadas em diversas áreas como em visão computacional (OSHER e PARAGIOS [46]), além de em simulações da dinâmica dos fluidos (OSHER e FEDKIW [15]). Sua ideia básica é o rastreio da interface através de uma representação implícita. Ao certo, superfícies podem ser reconstruídas a partir de um campo escalar amostrado em uma grade por metodologias como a dos *marching cubes* (LORENSEN e CLINE [47]. Como vantagem principal do conjunto de técnicas que empregam *level sets*, é possível indicar sua simplicidade de implementação e a capacidade de realizar mudanças topológicas de modo automático, além do simples rastreio do formato da superfície. Contudo, sua excessiva regularização numérica provoca arredondamento de quinas e a possível perda geral de volume. Esse e outros problemas já foram bastante estudados no meio acadêmico. Além de estratégias estritamente dentro do "padrão *level set*", algumas técnicas hibridas podem ser apontadas. No *particle level set* (PLS) (FOSTER e FEDKIW [11]), partículas lagrangianas sem massa são posicionadas em ambos os lados da interface, sendo advectadas antes de corrigirem o nível zero do campo escalar. Essas partículas carregam o valor de um raio que estima sua distância à borda. Então, é estimado em cada centro das células da grade, o valor de sua distância à borda a partir dos raios das partículas próximas. Obtém-se, assim, melhor precisão em áreas de alta curvatura.

Há ainda, o *particle level set* lagrangiano (LPLS) (HIEBER e KOUMOUTSAKOS [48]), onde partículas dispostas sobre a superfície carregam mapas gaussianos que espalham suas influências localmente. Já no *marker level set* (MIHALEF [49]), partículas são posicionadas ao longo da interface, consistindo em um modo de preservar a informação tangencial perdida pela formulação do *level set*. Ademais, para

a conservação exata da massa do sistema, SUSSMAN e PUCKETT [50] combinam o *level set* com a metodologia *volume of fluid*, que trabalha com o acompanhamento do volume do fluido em cada célula.

Apesar de esses métodos híbridos tornarem a proposta original mais robusta, como, neste trabalho, o foco está em como tratar sistemas multifásicos, optou-se por seguir a ideia de um método derivado especifico, o *level set* regional. Assim, apenas o *level set* puro será analisado, sendo ele posteriormente estendido para sua versão regional. No entanto, à principio, também seria possível modificar as técnicas hibridas de forma a ajustá-las ao contexto multifásico. Por exemplo, no caso do PLS, as partículas, além de guardar a menor distância entre sua posição e a interface, também poderiam manter um identificador de sua região inicial.

Na subseção seguinte, é introduzida a técnica do *level set* que será futuramente estendida para levar em consideração o contexto multifásico. Todavia, recomenda-se a consulta do livro de OSHER e FEDKIW [15] para um estudo mais detalhado de suas propriedades numéricas e aplicações.

4.1.1 Definição

O *level set* é definido por uma função $\phi(\vec{x})$ diferenciável em todo o domínio, salvo em pontos equidistantes da superfície. Define-se, então, a superfície pelos pontos onde $\phi(\vec{x}) = 0$. Por convenção, $\phi(\vec{x}) \leq 0$ no interior do fluido, e $\phi(\vec{x}) > 0$ no seu exterior. Tal função escalar é retratada por valores $\phi_{i,j,k}$ amostrados no centro de cada célula de índice (i, j, k) pertencente à grade. Assim, para obter seu valor nas demais posições, é feita uma interpolação trilinear entre os centros próximos.

Entretanto, eventualmente pode ser necessário obter valores de ϕ fora da grade computacional. Nesse caso, uma posição externa é projetada dentro da grade e seu valor é obtido neste ponto. Ainda, para fins de representação, o valor de ϕ é estendido das células fluidas para as sólidas de modo a evitar pequenos artefatos na fronteiras sólido-fluido.

A função $\phi(\vec{x})$ é comumente definida com base na função distância que, para o conjunto fechado S dos pontos na superfície da parte fluida, é dada por:

$$dist\hat{a}ncia_S(\vec{x}) = \{min(\| \vec{x} - \vec{p} \|) \mid \vec{p} \in S\}\,. \tag{4.1}$$

Isto é, para uma posição $\vec{x}$, o valor de $dist\hat{a}ncia_S(\vec{x})$ é a distância até o ponto mais próximo na superfície S. Ainda, como S divide o espaço nas regiões de dentro e de fora da massa fluida, pode-se definir:

$$\phi(\vec{x}) = \begin{cases} dist\hat{a}ncia_S(\vec{x}), & \text{Se } \vec{x} \text{ está fora.} \\ -dist\hat{a}ncia_S(\vec{x}), & \text{Se } \vec{x} \text{ está dentro.} \end{cases} \tag{4.2}$$

Em decorrência da utilização da função distância, algumas propriedades podem ser constatadas. Como ϕ é diferenciável fora do eixo medial da superfície S, exceto para esses pontos, é possível obter a normal à superfície por:

$$\hat{N} = \frac{\nabla\phi}{\parallel \nabla\phi \parallel}. \tag{4.3}$$

Tal propriedade pode ser entendida ao observar que o modo mais rápido de se atingir a superfície é caminhar na direção do ponto mais próximo, e sabendo que o gradiente denota a direção que um campo varia de forma mais acentuada. Além disso, deve-se observar que, no segmento de reta que une uma posição $\vec{x}$ ao ponto mais próximo de $\vec{x}$ em S ($p_S(\vec{x})$), o valor de $\hat{N}$ se mantém constante. Este último ponto também pode ser facilmente encontrado. De fato, para uma posição $\vec{x}$ qualquer:

$$p_S(\vec{x}) = \vec{x} - \phi(\vec{x})\nabla\phi(\vec{x}). \tag{4.4}$$

Ainda, a curvatura na superfície pode ser obtida através da equação:

$$\kappa = -\nabla \cdot \hat{N}. \tag{4.5}$$

Por fim, o volume e a área de uma região fluida também são propriedades que podem ser resgatadas. Com esta finalidade, é utilizada uma integração pela regra da quadratura de Gauss de 8 pontos, assim como realizado por KIM *et al.* [51]. No caso do volume, integra-se, na região desejada, a função Heaviside:

$$H(\Phi) = \begin{cases} 0, & \text{Se } \Phi \leq -\epsilon. \\ 1, & \text{Se } \Phi \geq \epsilon. \\ 0.5 + 0.75\dfrac{\Phi}{\epsilon} - 0.25\dfrac{\Phi^3}{\epsilon^3}, & \text{Se } \mid \Phi \mid < \epsilon. \end{cases}, \tag{4.6}$$

onde ϵ é uma constante pequena.

À titulo de exemplo, para uma célula (i, j, k) cujos pontos mínimo e máximo são, respectivamente (x_i, y_j, z_k) e $(x_{i+1}, y_{j+1}, z_{k+1})$, seu volume é dado por:

$$\begin{aligned} V &= \int_V H(\Phi(x,y,z))dV \\ &= \int_{z_k}^{z_{k+1}} \int_{y_j}^{y_{j+1}} \int_{x_i}^{x_{i+1}} H(\Phi(x,y,z))dx\, dy\, dz, \end{aligned} \tag{4.7}$$

onde $\Phi(x, y, z)$ é equivalente à função distância para posições dentro da região desejada, sendo, caso contrário, igual ao seu negativo.

Porém, para poder usar os valores tabelados para a quadratura de Gauss–Legendre, é preciso mudar os limites da integração para 1 e -1 através

de uma transformação linear. Nesse caso, o integrando da equação 4.7 pode ser expresso por:

$$g(x',y',z') = H\left(\Phi\left(\frac{x_{i+1}-x_i}{2}x' + \frac{x_{i+1}+x_i}{2}, \right.\right. \\ \left.\frac{y_{j+1}-y_j}{2}y' + \frac{y_{j+1}+y_j}{2} \right. \\ \left.\left.\frac{z_{k+1}-z_k}{2}z' + \frac{z_{k+1}+z_k}{2}\right)\right). \tag{4.8}$$

A expressão do volume se torna então:

$$\begin{aligned} V &= \left(\frac{z_{k+1}-z_k}{2}\frac{y_{j+1}-y_j}{2}\frac{x_{i+1}-x_i}{2}\right)\int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} g(x',y',z')dx'\,dy'\,dz' \\ &= \frac{\Delta x^3}{8}\int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} g(x',y',z')dx'\,dy'\,dz', \end{aligned} \tag{4.9}$$

de modo que o volume possa ser calculado pelo método da quadratura:

$$V = \frac{\Delta x^3}{8}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} w_i w_j w_k\, g(x_i,y_j,z_k), \tag{4.10}$$

usando pesos w_i, w_j, w_k e pontos (x_i, y_j, z_k) tabelados, cujos valores são originalmente obtidos a partir dos polinômios de Legendre. Seguindo a tabela apresentada no livro de ZWILLINGER [52], para uma quadratura de 8 pontos no espaço 3D, definindo $\alpha = 0.5773502692$, pode-se calcular o volume de uma região fluida em uma célula como:

$$V = \frac{\Delta x^3}{8}\begin{pmatrix} g(\alpha,\alpha,\alpha) + g(\alpha,\alpha,-\alpha) + \\ g(\alpha,-\alpha,-\alpha) + g(\alpha,-\alpha,\alpha) + \\ g(-\alpha,-\alpha,-\alpha) + g(-\alpha,\alpha,-\alpha) + \\ g(-\alpha,\alpha,\alpha) + g(\alpha,-\alpha,\alpha) \end{pmatrix}. \tag{4.11}$$

Analogamente, utilizando a derivada da função Heaviside (equação 4.6):

$$\delta(\Phi) = \begin{cases} 0, & \text{Se } \mid \Phi \mid \geq \epsilon \\ 0.75\left(\frac{1}{\epsilon} - \frac{\Phi^2}{\epsilon^3}\right), & \text{Se } \mid \Phi \mid < \epsilon. \end{cases}, \tag{4.12}$$

a área da superfície fluida pode ser calculada:

$$A = \frac{\Delta x^3}{8}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} w_i w_j w_k\, \delta(\phi(x_i,y_j,z_k)). \tag{4.13}$$

4.1.2 Inicialização

Como primeiro passo para o emprego do *level set*, o campo distância deve ser inicializado, isto é, todas as células devem ter seu valor igual à menor distância de seu centro à posição inicial da interface. Entretanto, futuramente ele será, por vezes, reinicializado para garantir a propriedade de ser uma distância mínima à superfície do fluido.

Logo, de início, são calculados os valores nos centros das células adjacentes à superfície. Para a primeira inicialização, a interface é localizada na face entre duas células de diferentes materiais. Um exemplo desse caso pode ser visto na figura 4.1. Já para as sucessivas reinicializações, a posição da superfície é dada pela interpolação linear entre os atuais valores de células adjacentes que estão em regiões distintas.

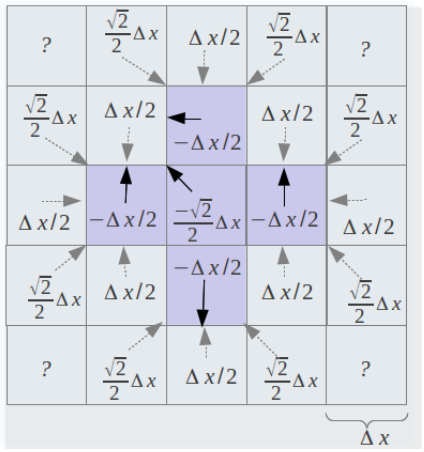

Figura 4.1: Exemplo de inicialização do campo distância com base nos diferentes materiais. Diferentes fluidos são representados por cores distintas, e os pontos relativos às distâncias mínimas exibidas são indicados por setas.

Tendo definido o valor do *level set* nas células mais próximas à interface, em seguida, a informação da distância deve ser propagada para as células mais distantes. Porém, como apenas é necessário que o campo distância esteja correto próximo à interface, pois ele a define, posições suficientemente longes dela não precisam ter seus valores recalculados com frequência, assim, nas reinicializações, apenas é atualizado o valor de células distantes até $3\Delta x$ da superfície.

Um dos métodos mais conhecidos para realizar essa tarefa é o *fast marching* (SETHIAN [39]). Contudo ele é essencialmente sequencial, utilizando uma lista de prioridades com pontos da grade que possuem distância desconhecida à borda, ordenados por seus valores estimados. Esta lista é, tipicamente, implementada como um *heap*. Assim, para cada iteração desse algoritmo, o ponto com valor estimado mínimo é retirado da lista e são atualizadas as estimativas das células vizinhas à célula cujo valor foi definido.

A fim de tomar proveito da paralelização oferecida pela tecnologia CUDA, foi

adotado outro método. Nessa solução paralela, cada célula verifica se alguma célula adjacente já possui sua distância definida. Se isto acontecer, computa-se a distância do centro da célula ao ponto da superfície mais próximo a essa célula vizinha. O valor da distância da célula à borda é então atualizado, atribuindo-se a ela a menor distância obtida pelo procedimento anterior para todas as vizinhas com distância já computada. Desse modo, em $n-1$ iterações, células distantes de até $n\Delta x$ da superfície terão seus valores definidos. À titulo de exemplo, a figura 4.2 ilustra a definição dos valores das células adjacentes à interface, com definição de seus pontos mais próximos, e a propagação da distância através de duas iterações do processo aqui descrito.

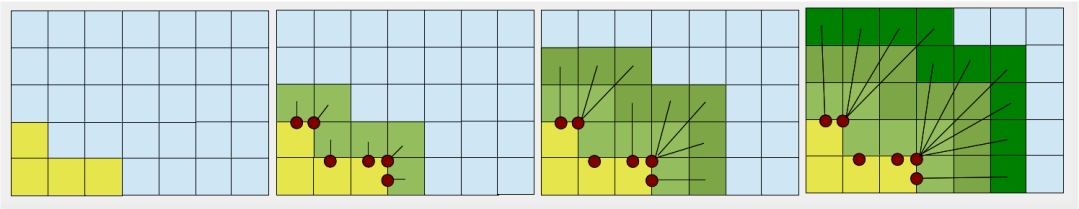

Figura 4.2: Definição dos valores do *level set* de células adjacentes à interface, com definição de seus pontos mais próximos; e duas iterações do método utilizado para a propagação das distâncias. Nas imagens, pontos indicam as posições mais próximas na interface e segmentos de reta ligam tais pontos aos centros de suas respectivas células.

4.1.3 Evolução

Após o cálculo do campo distância inicial, é necessário o atualizar, de modo que as posições do nível zero, aquelas que definem a interface, sejam alteradas de acordo com o deslocamento do fluido. Assim, o *level set* é advectado aos moldes da equação 3.15. Isto é, resolve-se a equação:

$$\frac{D\phi}{Dt} = 0, \tag{4.14}$$

Ainda, cabe ressaltar que, como a advecção necessita de um campo velocidade que possua divergente nulo em todos os pontos em que ele é definido, a advecção, assim como toda a atualização do *level set* para determinada iteração, é realizada imediatamente antes da advecção do campo velocidade.

De modo similar ao caso da velocidade, descrito na seção 3.4, a advecção do *level set* é feita pelo método semi-lagrangiano. Contudo, quando empregado no caso multifásico, a extensão Runge-Kutta é utilizada para maior precisão. Em seguida, o *level set* é reinicializado pois, geralmente, a advecção não preserva a propriedade básica de ϕ ser uma distância.

4.2 *Level Set* Regional

A ideia do *level set* regional teve origem na área de modelagem geométrica (BLOOMENTHAL e FERGUSON [53], YAMAZAKI *et al.* [54]) onde, ao invés de usar o sinal de uma função real para a regionalização binária do espaço, foram usados identificadores de regiões. Em seguida, a técnica foi estendida por ZHENG *et al.* [27] para o contexto da simulação de bolhas, onde se empregou, adicionalmente, uma função escalar. No caso, foi utilizada a função distância. Futuramente, KIM [1] empregou o método em uma simulação multifásica para rastrear propriedades de fluidos distintos.

4.2.1 Definição

O *level set* regional é definido por uma tupla obtida através da função regional:

$$f_R(\vec{x}) = \langle\, r(\vec{x}), \phi(\vec{x}) \,\rangle\,, \tag{4.15}$$

onde $\vec{x}$ é uma posição no espaço, $r(\vec{x})$ é o identificador da região que contém $\vec{x}$ e $\phi(\vec{x})$ é o valor do *level set* nesse ponto, tal como definido na seção 4.1. Contudo, para o presente caso, $\phi(\vec{x})$ pode apenas assumir valores não negativos, uma vez que o sinal já não poderia mais identificar exatamente em qual fase (ou região) um ponto se encontra.

Assim sendo, tuplas regionais obtidas pela aplicação da função anterior são guardadas no centro de cada célula. Por exemplo, a figura 4.3 demonstra as tuplas de duas células adjacentes de fluidos distintos. Nesse contexto, cabe ressaltar que, à rigor, a regionalização indica apenas uma subdivisão do espaço em regiões. Dessa forma, nada impede que existam duas regiões distintas do mesmo material ou mesmo que estas sejam adjacentes.

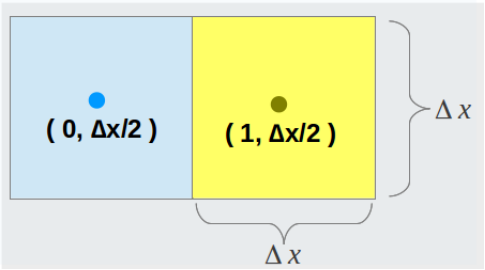

Figura 4.3: Tuplas regionais de duas células adjacentes de regiões distintas.

Para a inicialização e advecção das tuplas regionais, são empregados procedimentos análogos àqueles utilizados no *level set* tradicional. De fato, na primeira inicialização, os valores de ϕ são definidos do modo usual e são atribuídos rótulos

às regiões seguindo a ordem em que cada componente de fluido é primeiramente definido. Para as sucessivas reinicializações, referentes à recuperação de ϕ como uma função distância, apenas seus valores são modificados. Já na advecção, cada tupla é advectada pelo esquema Runge-Kutta de modo que seu novo valor seja igual à tupla na posição rastreada pela técnica escolhida.

Em relação aos limites da grade computacional que determina onde é realizada a simulação, de modo semelhante ao *level set*, para posições do espaço exteriores a ela, utiliza-se a tupla obtida no ponto projetado na borda da grade. Ademais, a região de uma célula sólida é igual a região de uma célula fluida vizinha (mas sempre a mesma célula).

Em conjunto com a função regional definida pela equação 4.15, alguns operadores (ZHENG *et al.* [27]) também devem ser definidos para que se possa trabalhar com as tuplas regionais, sejam eles:

- Soma:

$$\langle r_1, \phi_1 \rangle + \langle r_2, \phi_2 \rangle = \begin{cases} \langle r_1, \phi_1 + \phi_2 \rangle, & \text{Se } r_1 = r_2 \\ \langle r_1, \phi_1 - \phi_2 \rangle, & \text{Se } \phi_1 \leq \phi_2 \\ \langle r_2, \phi_2 - \phi_1 \rangle, & \text{Se } \phi_2 > \phi_1 \end{cases} \tag{4.16}$$

- Subtração:

$$\langle r_1, \phi_1 \rangle - \langle r_2, \phi_2 \rangle = \begin{cases} \langle r_1, \phi_1 - \phi_2 \rangle, & \text{Se } r_1 = r_2 \\ \langle r_1, \phi_1 + \phi_2 \rangle, & \text{Se } \phi_1 \leq \phi_2 \\ \langle r_2, \phi_2 + \phi_1 \rangle, & \text{Se } \phi_2 > \phi_1 \end{cases} \tag{4.17}$$

- Produto por escalar:

$$\alpha \langle r_1, \phi_1 \rangle = \langle r_1, \alpha\phi_1 \rangle, \alpha \in \mathbb{R}^+ \tag{4.18}$$

- Produto:

$$\langle r_1, \phi_1 \rangle \langle r_2, \phi_2 \rangle = \begin{cases} \langle r_1, \phi_1\phi_2 \rangle, & \text{Se } r_1 = r_2 \\ \langle r_1, -\phi_1\phi_2 \rangle, & \text{Se } r_1 \neq r_2 \end{cases} \tag{4.19}$$

- Divisão:

$$\frac{\langle r_1, \phi_1 \rangle}{\langle r_2, \phi_2 \rangle} = \begin{cases} \left\langle r_1, \dfrac{\phi_1}{\phi_2} \right\rangle, & \text{Se } r_1 = r_2 \\ \left\langle r_1, -\dfrac{\phi_1}{\phi_2} \right\rangle, & \text{Se } r_1 \neq r_2 \end{cases} \tag{4.20}$$

Essas operações são definidas de forma a permitir a atribuição de uma tupla regional a um ponto qualquer de uma célula com base nos pares definidos nos centros

adjacentes empregando uma interpolação linear em cada coordenada. Todavia, algumas propriedades usuais das operações de soma, subtração, multiplicação e divisão não são atendidas por esse conjunto de operações. Um problema já encontrado é a falta de comutatividade da operação de soma. Por exemplo, $(\langle 1,5\rangle + \langle 2,3\rangle) + \langle 3,1\rangle = \langle 1,2\rangle + \langle 3,1\rangle = \langle 1,1\rangle$, mas $\langle 1,5\rangle + (\langle 2,3\rangle + \langle 3,1\rangle) = \langle 1,5\rangle + \langle 2,2\rangle = \langle 1,3\rangle$.

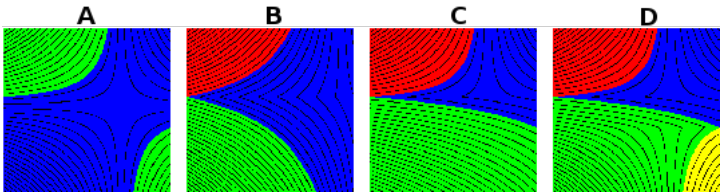

Figura 4.4: Resultado da interpolação bilinear entre quatro tuplas regionais, mas interpolando primeiro no eixo x. Imagem adaptada de KIM [1]

Como resultado, a interpolação trilinear realizada para obter uma tupla regional em uma posição qualquer deixa de ser consistente. De fato, utilizando um exemplo bidimensional, pode-se perceber que o produto obtido depende de qual eixo é interpolado primeiro. Para um ponto de coordenadas m_x e m_y dentro de uma célula unitária $[0,1] \times [0,1]$, se primeiro for realizada uma interpolação no eixo x, tem-se:

$$\langle r,\phi\rangle = [\langle r_1,\phi_1\rangle (1-m_x) + \langle r_2,\phi_2\rangle m_x](1-m_y) + [\langle r_3,\phi_3\rangle (1-m_x) + \langle r_4,\phi_4\rangle m_x] m_y, \tag{4.21}$$

obtendo, como exemplo, as quatro configurações regionais (A,B,C e D) expostas na figura 4.4.

Em contrapartida, se a primeira interpolação for dada no eixo y:

$$\langle r,\phi\rangle = [\langle r_1,\phi_1\rangle (1-m_y) + \langle r_3,\phi_3\rangle m_y](1-m_x) + [\langle r_2,\phi_2\rangle (1-m_y) + \langle r_4,\phi_4\rangle m_y] m_x, \tag{4.22}$$

alcançando, por sua vez, as distribuições regionais apresentadas na figura 4.5 para a interpolação das mesmas tuplas dos respectivos casos A,B,C e D da figura 4.4.

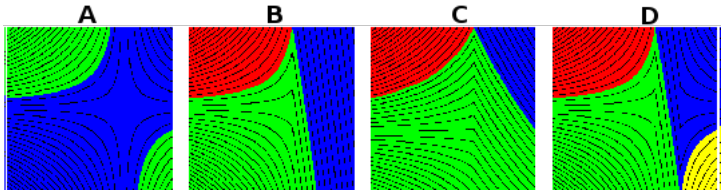

Figura 4.5: Resultado da interpolação bilinear entre quatro tuplas regionais, mas interpolando primeiro no eixo y. Imagem adaptada de KIM [1]

Para evitar que isso aconteça, KIM [1] propôs um método para realizar a soma de múltiplas tuplas enquanto mantém a propriedade comutativa. Assim, o caso do exemplo da interpolação bilinear pode ser traduzido por uma soma, que deve ser

calculada pela nova formulação:

$$\begin{aligned}\langle r, \phi \rangle &= \langle r_1, (1-m_x)(1-m_y\rangle\, \phi_1) + \langle r_2, m_x(1-m_y)\phi_2 \rangle\,, \langle r_3, (1-m_x)m_y\phi_3 \rangle\,, \langle r_4, m_x m_y \phi_4 \rangle \\ &= \sum_{i=1}^{n} \langle r_i, \phi_i' \rangle\,. \end{aligned} \tag{4.23}$$

Para a computação do produto do novo operador de soma, alguns passos devem ser seguidos. De inicio, é calculada uma soma parcial por região:

$$\Psi_k = \sum_{\forall r_i = r_k} \phi_i. \tag{4.24}$$

Em seguida, são escolhidos os dois maiores valores de Ψ. Isto é, são definidos Ψ_1 e Ψ_2, de modo que $\Psi_1 \geq \Psi_2 \geq \Psi_k$, com $k = 3, ..., m$. Então, o resultado é obtido:

$$\sum_{i=1}^{n} \langle r_i, \phi_i \rangle = \sum_{k=1}^{m} \langle r_k, \Psi_k \rangle = \langle r_1, \Psi_1 - \Psi_2 \rangle\,. \tag{4.25}$$

De modo geral, observa-se que, além de garantir uma propriedade matemática da operação de soma, o procedimento anterior gera uma regionalização mais regular. Por exemplo, os quadros da figura 4.6 foram criados a partir de configurações de tuplas semelhantes àquelas das figuras 4.4 e 4.5 que utilizaram a formulação antiga. Note que os pontos que estão na interseção entre três regiões distintas não ocorrem mais necessariamente em uma aresta do quadrado.

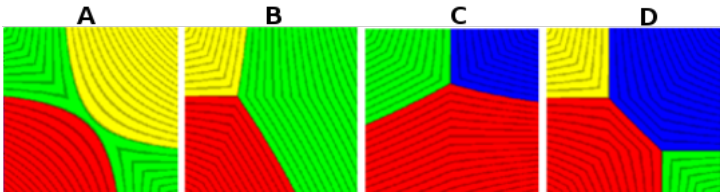

Figura 4.6: Resultado da interpolação bilinear entre quatro tuplas regionais através da formulação proposta por KIM [1].

4.2.2 Tratamento do Filme

Uma característica essencial na simulação multifásica é a existência de um fino filme que pode estar presente ou não na interface entre duas regiões (ou fases). De fato, para que possa haver duas regiões conexas do mesmo material, é preciso que exista um filme de outro fluido que as separe. Caso contrário, elas se fundiriam em uma única fase. Já no caso da adjacência de regiões de composições distintas, a presença do filme não é necessária, apesar de ela poder ocorrer.

Para que se pudesse definir se existe um filme em determinada interface entre regiões, assim como seu respectivo material, seria preciso verificar quando duas regiões passam a ser adjacentes e a composição dos fluidos próximos à nova interface criada. Além disso questões relacionadas também deveriam ser tratadas, tal como qual material escolher quando o filme puder ser formado de um de dois materiais, ou se vale a pena criar filmes através de alguma composição entre os diferentes materiais possíveis. Para evitar essas complicações adicionais, neste trabalho, é estabelecido que sempre existe um filme entre duas regiões adjacentes e que ele sempre é de um mesmo material pré-definido.

Além do material do filme, sua espessura também é uma propriedade que merece atenção especial. Sem dúvida, se o seu valor se tornar suficientemente pequeno, as forças atuantes podem fazer com que o filme seja rompido e as regiões adjacentes se unam. Assim, se mostra necessário acompanhar a espessura do filme e testar condições de ruptura.

Contudo, o filme tem comportamento muito complexo para ser simulado de modo eficiente. Por exemplo, sabe-se que o filme se torna fino devido ao alongamento, à drenagem e à evaporação. Ainda, a ruptura pode ser atrasada por variações de pressão ou pela tensão superficial que minimiza a área da interface. Desse modo, é preciso uma forma eficiente de atualizar a espessura do filme ou, ao menos, reconhecer situações em que ele deve ser rompido.

Para começar, como os filmes são bastante finos, na escala dos micrômetros, variações da espessura não são visíveis na renderização. Assim, não é necessário acompanhar a espessura do filme se existir um procedimento que detecte as situações em que ele seja rompido. Logo, pode-se estabelecer uma espessura real fixa para todos os filmes, igual a $h = 0.1\Delta x$, que será usada para efeito da visualização, enquanto é empregado um método que trata a ruptura dos filmes.

Como uma solução possível para o teste do rompimento do filme, pode-se empregar aquela proposta por KIM [1], onde é utilizada uma condição intuitiva com foco na animação. Segundo essa abordagem, um usuário define dois parâmetros para uma bolha com 1 cm de raio e 0.0004π m^2 de área. O primeiro é o tempo de vida t_c quando a ruptura dessa bolha de área padrão pode começar a acontecer, e o segundo, o tempo t_f quando essa ruptura necessariamente já terá acontecido. Desse modo, esses tempos podem ser escalados para filmes de diferentes áreas por:

$$t'_f = t_f \left(\frac{0.0004\pi}{\text{Área do Filme}} \right)^{\alpha} \tag{4.26}$$

$$t'_c = t_c \frac{0.0004\pi}{\text{Área do Filme}}, \tag{4.27}$$

onde $\alpha = 0.3$ é uma constante.

Assim, em cada iteração, um filme é rompido com probabilidade:

$$P(t) = \begin{cases} \dfrac{t - t'_c}{t'_f - t'_c}, & \text{Se } t \in [t'_c, t'_e] \\ 1, & \text{Se } t > t'_f \\ 0, & \text{Se } t < t'_c \end{cases} \tag{4.28}$$

Contudo, neste trabalho, é empregado o modelo aproximado apresentado por ZHENG *et al.* [27], por ele ser mais realista. Consoante esse método, é determinada, por região, uma espessura fictícia l para o filme, que não é a espessura da interface, mas sim um valor auxiliar para o tratamento das rupturas. Assim, os diferentes valores de l são atualizados por drenagem e deformação segundo a equação:

$$l_r^t = l_r^{t-1} - l^{t-1}\frac{(A_r^t - A_r^{t-1})}{A_r^t} - \left(\frac{l_0 - l_{min}}{t_{persist}}\right)\Delta t, \tag{4.29}$$

onde l_r^t e A_r^t são, respectivamente, a espessura e a área de uma região r no instante t, l_{min}, aqui usualmente igual a $0.01\Delta x$, é a menor espessura que uma região pode ter, $l_0 = 0.1\Delta x$ é a espessura inicial de quando uma região é criada, $t_{persist}$ é o tempo máximo que uma região pode permanecer sem ter seu filme rompido, e Δt é o tamanho do passo de tempo t.

Porém, o termo da equação anterior referente à drenagem - aquele que depende de Δt - só é levado em conta se a região em questão estiver em contato com outra do mesmo material. Caso contrário, pode não fazer sentido que o fluido do filme seja drenado. Assim, como efeito geral, pode-se ter, por exemplo, que o filme seja drenado ao longo do tempo de modo que sua espessura seja reduzida. Além disso, se a área aumentar, o filme terá sido esticado e sua espessura será reduzida.

À medida que a espessura do filme é atualizada em cada iteração, também é feito o teste do rompimento de cada filme. De fato, se a espessura média de duas regiões de mesmo material adjacentes à uma interface for menor que o limiar t_{min}, o filme é rompido e as duas regiões se fundem. Ainda, a espessura do filme da região resultante é igual ao máximo entre as espessuras do filme das regiões originais.

Adicionalmente, se a densidade do filme for menor que a das regiões adjacentes de mesmo constituinte, estas também são unidas. Essa última condição visa retratar situações em que, por exemplo, uma gota de água cai em uma grande massa de água. Como o fluido que separa as duas regiões de água é composto de ar, que é muito menos denso, a junção das duas regiões é instantânea.

4.2.3 Detecção de Regiões

Em cada iteração da simulação, é preciso conhecer quais regiões estão presentes. Nesse contexto, após a advecção, é utilizado o algoritmo de segmentação por *flood fill* (ou inundação) uma vez empregado por GREENWOOD e HOUSE [19] para detectar bolsas de ar para identificar regiões existentes em determinada iteração. Além disso, essa técnica é adaptada para também realizar a junção de regiões que, no caso do trabalho de KIM [1], é realizada por uma etapa separada. Assim, é utilizado um algoritmo modificado de segmentação por inundação, tal como ilustrado pelos pseudocódigos presentes nas Figuras 4.7 e 4.8.

Enquanto existirem células que não tiverem sido visitadas:
    Se possível, para cada fluido que não possua nenhuma célula ativa, escolher uma célula ainda não visitada e do mesmo material.
    Para cada uma dessas células:
        Criar nova região.
        Ativar célula.
    Para cada célula c_1 ativa:
        Para cada célula adjacente c_2 ainda não visitada:
            Visitar(c_1, c_2).
        Desativar c_1.
        Classificar c_1 como visitada.

Figura 4.7: Pseudocódigo do algoritmo de detecção de regiões.

função **Visitar**(c_{de}, c_{para}):
    Se$(c_{para} = sólido)$, **então**:
        $r^t(c_{para}) = r^t(c_{de})$
    Senão, **Se** $r^{t-1}(c_{de}) = r^{t-1}(c_{para})$, **então**:
        $r^t(c_{para}) = r^t(c_{de})$.
        Ativar c_{para}.
    Senão, **Se** $fluido(c_{de}) = fluido(c_{para})$, **então**:
        $l_{filme} = 0.5 * (l_{r^{t-1}(c_{de})} + l_{r^{t-1}(c_{para})})$
        romper filme $= l_{filme} < l_{min}$ **ou** $\rho_{fluido(c_{de})} > \rho_{filme})$
        Se (romper filme):
            $l_{r^t(c_{para})} = max(l_{r^t(c_{de})}, l_{r^{t-1}(c_{para})})$
            $r^t(c_{para}) = r^t(c_{de})$
            Ativar c_{para}.

Figura 4.8: Pseudocódigo da função **Visitar** do algoritmo de detecção de regiões. Este procedimento visita a célula c_{para} a partir da célula c_{de}. $r^t(c)$ é a região do centro da célula c no tempo t e $l_{r^t(c_{para})}$ é a espessura fictícia do filme dessa região.

Para o processo de detecção, consideram-se células ativas aquelas atingidas pela inundação, mas que seus vizinhos ainda não foram explorados. Além disso, durante sua execução, dois tipos de mapa são considerados: um indicando a região de cada

célula e outro com seus respectivos materiais. Para melhor explicar o funcionamento do algoritmo, ele será aplicado a um pequeno exemplo onde a configuração inicial é a retratada pela figura 4.9. À princípio, através dos dois mapas, deve-se observar que uma região foi dividida na etapa de advecção e que duas regiões do mesmo material estão em contato.

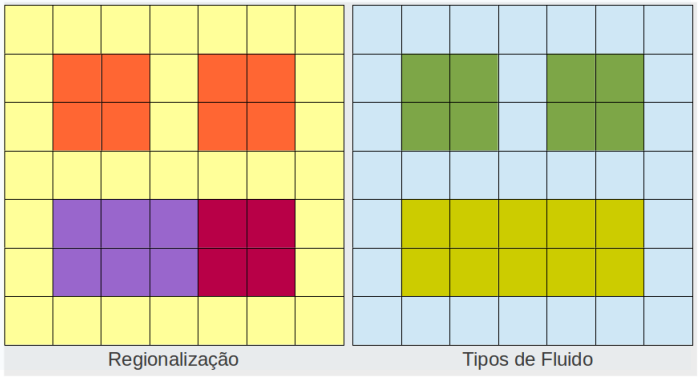

Figura 4.9: Estado inicial do exemplo do procedimento *flood fill* para a definição de regiões.

Logo, inicialmente, como nenhum fluido possui células ativas, é ativada uma célula para cada um deles, sendo criadas três novas regiões. Além disso, para essas células, apenas seus vizinhos de mesmo material são ativados, atingindo o estado ilustrado pela figura 4.10.

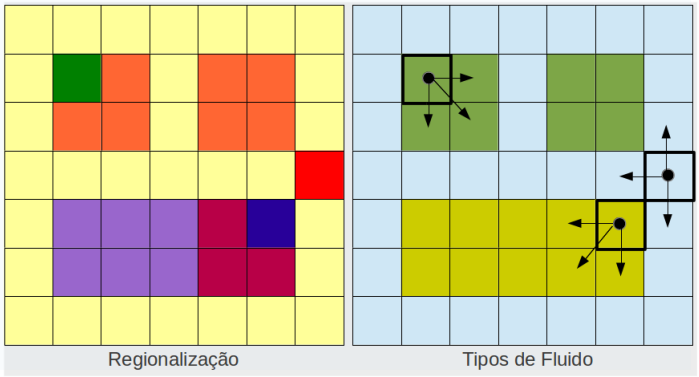

Figura 4.10: Primeira iteração do exemplo do procedimento *flood fill* para a definição de regiões.

Depois, duas situações diferentes podem ser observadas, ambas ilustradas na figura 4.11. Uma delas se refere ao esgotamento de células ativas de um material, o que resulta na ativação de uma célula desse fluido ainda não visitada e na criação de mais uma região. A outra é o atendimento da condição de ruptura de filme, indicada pelas setas maiores. Neste caso, a nova região das células C_1' e C_2' é igual àquela em C_1 e C_2, a partir de onde as primeiras foram encontradas. Além disso, a

espessura do filme da região resultante é feita ser o máximo entre aquelas das duas regiões que foram unidas.

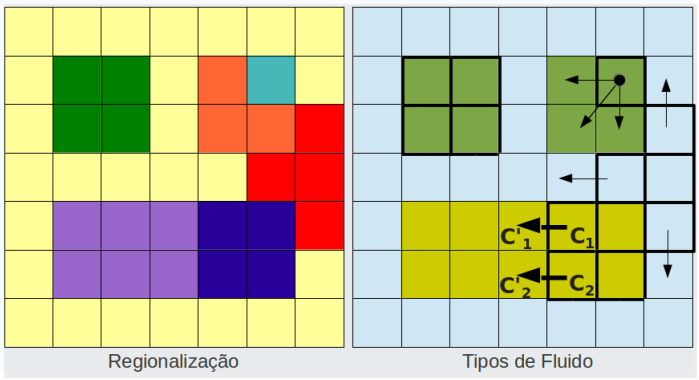

Figura 4.11: Segunda iteração do exemplo do procedimento *flood fill* para a definição de regiões.

Em seguida, a inundação prossegue normalmente, sempre fluindo entre células que possuem a mesma região antiga. Por exemplo, no caso de onde aconteceu a ruptura do filme, a nova região é propagada para as demais células da antiga região, como pode ser visto na figura 4.12.

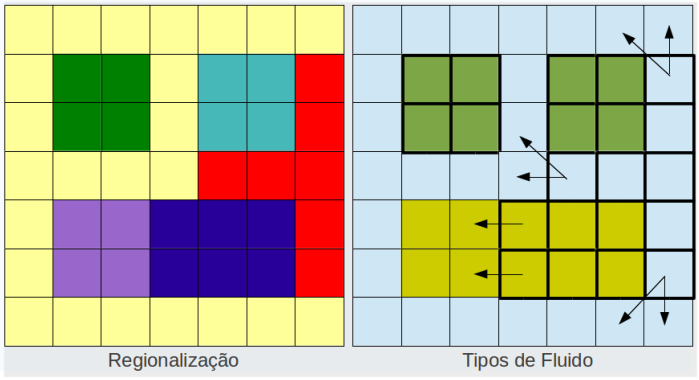

Figura 4.12: Terceira iteração do exemplo do procedimento *flood fill* para a definição de regiões.

Por fim, tem-se a configuração dada pela figura 4.13, que foi criada após a fusão de duas regiões e a divisão de outra. Note que o algoritmo é executado em paralelo para cada tipo de fluido. Assim, quanto mais fluidos distintos, mais eficiente o algoritmo pode se tornar. Além disso, pode-se perceber que os novos identificadores regionais podem ser diferentes dos antigos. Isto é, nada garante que o novo identificador de uma região seja o mesmo da etapa anterior, de modo que é preciso transmitir atributos de uma região definida em uma etapa para uma região correspondente definida no passo seguinte, como apresentado na subseção 4.2.4.

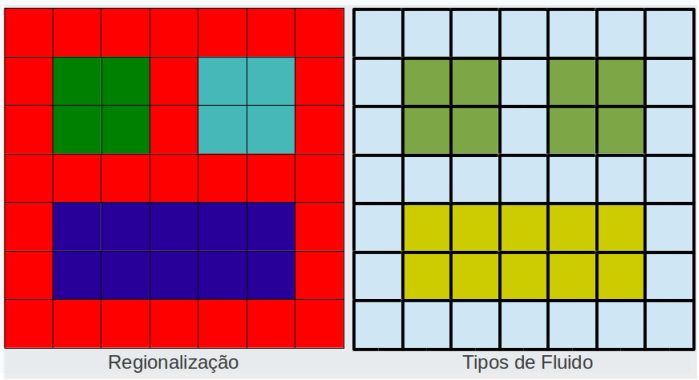

Figura 4.13: Estado final do exemplo do procedimento *flood fill* para a definição de regiões.

4.2.4 Grafo Regional

Para identificar unicamente cada filme, é definido o grafo regional G^t para a iteração (ou passo de tempo) t. Ao certo, cada um de seus nós representa uma região e, se uma aresta existe entre dois nós, suas respectivas regiões são adjacentes, existindo um filme que as separa. Assim, este grafo deve ser consultado para saber quais regiões são adjacentes, por exemplo, ao verificar se deve ser aplicada drenagem à espessura do filme.

O grafo pode ser facilmente calculado examinando a vizinhança de cada célula. De fato, se duas células vizinhas são de regiões diferentes, tais regiões são adjacentes. Desse modo, para n regiões, é determinada uma matriz triangular superior onde, para duas regiões r_i e r_j, a posição (i, j) guarda um valor booleano indicativo da adjacência ou não dessas regiões. A matriz propriamente dita é armazenada como um *array* de tamanho $\frac{(n-1)n}{2}$, indexado pela função de *hash*:

$$\acute{I}ndice_{G^t}(i,j) = i\left(\frac{2n-i-1}{2}\right) + (j-i-1), \text{ para } i < j. \tag{4.30}$$

Como apontado na subseção 4.2.3, é preciso saber como são transferidos os atributos das regiões da iteração anterior para as atuais. Para tanto, de início, os nós de um grafo regional G^{t-1} são mapeados nos nós de G^t. Isto é, cria-se o mapeamento $F : G^{t-1} \to G^t$ onde cada região r^{t-1} deve ser mapeada em pelo menos um r^t, assim como cada região r^t é inversamente mapeada em ao menos um r^{t-1}. Caso alguma região r^{t-1} não possua mapeamento, então ela foi removida durante a advecção do *level set*. Esta situação é tratada pela adição de uma partícula com as mesmas características da região removida e cuja dinâmica é detalhada na subseção 4.2.5. Além disso, através desse grafo, também é possível descobrir quais regiões sofreram ou são produto de uma união ou subdivisão.

Em seguida, analisam-se as características que devem ser rastreadas. Uma delas

é a espessura do filme regional. No caso em que uma região não tenha sofrido junção ou divisão, a propriedade é transferida diretamente pela relação F. Caso contrário, se ela é produto de uma divisão, é atribuída à sua respectiva espessura de filme, o valor mínimo l_{min}. Ainda, como abordado na subseção 4.2.3, com respeito à fusão de regiões, é conferido à região resultante o valor máximo entre as espessuras das duas regiões unidas.

Outra propriedade levada em consideração é a área da região antiga, que é utilizada para modificar a espessura do filme de uma região devido à sua distorção. Analogamente ao caso da espessura, esse valor é transmitido por F quando a região original não tiver se dividido ou se unido com outra. Todavia, nos demais casos, devido à descontinuidade que essas modificações topológicas provocam, atribui-se à área anterior o valor calculado para a área da região na iteração presente.

Por fim, a última característica rastreada é o volume alvo de uma região, isto é, o volume que a região deveria possuir e que será levado em consideração pelo método de controle apresentado na seção 6. Como regra geral, tem-se que o volume alvo de uma região deve ser conservado. Desse modo, quando duas regiões são unidas, seus volumes alvo são somados. No caso da divisão, o volume alvo de cada região resultante é uma fração do volume alvo da região original, sendo proporcional às frações de volume de cada região resultante. Assim, se, por exemplo, uma região for dividida em uma região pequena e uma grande, o volume alvo desta será maior que o daquela.

Devido à necessidade do rastreamento dessas propriedades, o algoritmo descrito na subsecção 4.2.3 é, à rigor, realizado em dois passos que possuem suas próprias etapas de transferência de características. No primeiro, é feita a simples segmentação do espaço com base no material de cada célula, visando tratar as subdivisões. Já na segunda passagem, que trata a união de regiões, também são testadas as condições de ruptura dos filmes. Esta medida se faz necessária pois, se tanto a junção como a subdivisão fossem consideradas ao mesmo tempo, em casos em que essas transformações ocorressem conjuntamente, o rastreamento das propriedades não poderia ser feito de modo tão simples. Por exemplo, seria possível que uma região se dividisse em três e que dois de seus produtos se fundissem com regiões distintas. Neste caso, não faria sentido distribuir o volume alvo da região dividida entre aquelas criadas.

4.2.5 Partículas

Para conferir maior realismo físico e visual à simulação, é empregado um sistema de partículas esféricas, cujos componentes são definidos através de seus raios (atuais e alvos), posições do centro e tipos de fluido, além de suas atuais velocidades. Entretanto, neste trabalho é feita a criação de partículas apenas com a finalidade

de substituir pequenos componentes que não podem ser simulados devido à discretização da grade euleriana. Tendo isso em vista, não é considerada a fusão de partículas, nem a interação entre elas, à exemplo do realizado no trabalho de BUSARYEV *et al.* [55], assim como não são levados em consideração outros métodos para as criar. Por exemplo, poder-se-ia tratar as partículas escapadas do *particle level set* (GREENWOOD e HOUSE [19]), que também são auxiliam a evolução das superfícies, como uma partícula do modelo aqui apresentado.

Para a movimentação (ou advecção) de cada partícula p no instante t, é usado o simples esquema de integração *forward* Euler para sua posição x_p e sua velocidade v_p:

$$\vec{u}_p^t = \vec{u}_p^{t-1} + \sum_{i=0}^{k} \vec{a}_{p,i}^t \Delta t \tag{4.31}$$

$$\vec{x}_p^t = \vec{x}_p^{t-1} + \vec{v}_p^t \Delta t, \tag{4.32}$$

onde os sobrescritos t e $t-1$ indicam o tempo em que é avaliada uma propriedade e $\sum_{i=0}^{k} \vec{a}_{p,i}^t$ é a soma de todas as acelerações $\vec{a}_{p,i}$ das forças atuantes na partícula p na iteração t.

Precisam, então, ser definidas as forças que agem em cada partícula. Seguindo os trabalhos de KIM [1], CLEARY *et al.* [6], é aplicada uma força que leva em consideração o efeito da gravidade, cuja componente vertical é:

$$f_e = V_p(\rho_p - \rho_{fluido(\vec{x}_p)})g, \tag{4.33}$$

onde V_p e ρ_p são, respectivamente, o volume e a densidade de p, e $g = -9.81 m/s^2$. Além disso, nos moldes da pesquisa de BUSARYEV *et al.* [55], também é aplicada uma força de arraste:

$$\vec{F_a} = c_{arraste} r_p^2 \parallel \vec{u}(\vec{x}_p) - \vec{u}_p \parallel (\vec{u}(\vec{x}_p) - \vec{u}_p), \tag{4.34}$$

onde $c_{arraste} \in [0.05, 0.5]\, kg/m^3$, r_p é o raio da esfera de p, e $\vec{u}(\vec{x}_p)$ é o vetor velocidade obtido na posição de p a partir da grade. Dessa forma, as acelerações necessárias para o computo da equação 4.31 podem ser obtidas dividindo cada força por $V_p \rho_p$.

Contudo, no trabalho de KIM [1], partículas com raio $r_p > 5\Delta x$ não são advectadas com base nas velocidades obtidas pela atuação das forças indicadas acima. Ao invés disso, a densidade dessas partículas é amostrada no centro das células da grade que se encontram dentro delas, de forma que essas densidades sejam consideradas durante os passos da simulação euleriana (advecção, forças externas, projeção). Então, após estas etapas, as velocidades utilizadas para movimentar partículas suficientemente grandes são obtidas a partir da grade, diretamente de suas respectivas

posições.

Após conhecer como as partículas são definidas e como elas são movimentadas, também cabe entender como elas são geradas. Ao certo, neste trabalho, elas são criadas apenas pela conversão de uma região, operação esta que sempre busca evitar que regiões desapareçam devido aos limites da discretização adotada. Essa transformação é dada quando uma região possui volume $V_p < V_{min} = 27\Delta x^3$, correspondente ao volume de uma célula e as adjacentes que têm um vértice em comum com ela.

Um exemplo da metamorfose de uma região para uma partícula pode ser visualizado na figura 4.14 onde, no tempo $t = 1$, tem-se uma região e, em $t = 2$, uma partícula. De modo intuitivo, esta aproximação por partícula pode ser justificada pois, à medida que se utiliza uma grade mais densa, regiões convertidas para partícula terão volume menor, e regiões menores tendem a ser mais esféricas do que as maiores. Adicionalmente, se for detectado que uma região foi removida na etapa da advecção dos *level sets* do fluido, também é criada uma partícula.

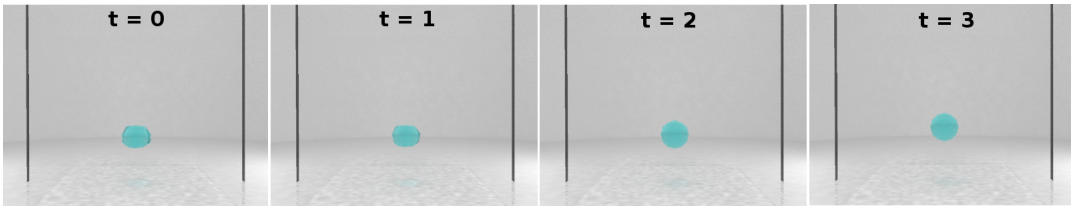

Figura 4.14: Em uma grade de dimensão 22^3, sucessivas iterações da simulação de uma pequena região que, após ter ficado suficientemente pequena é convertida em partícula entre os tempos $t = 1$ e $t = 2$.

Logo, quando uma partícula substitui uma região, ela é posicionada no centro estimado dessa região. Esta posição é definida pela média das posições dos centros de cada célula da região, ponderados por seus valores do campo distância à borda. Ainda, seu raio é determinado de modo que ela tenha o mesmo volume da região original. Para permitir certa correção de volume enquanto partícula, também é definido um raio alvo para a partícula, de modo a garantir que o seu volume alvo seja igual ao da região que a originou.

Já para a região convertida em partícula, ela não é apenas removida da grade. De fato, ela é marcada como inativa de forma a não ser mais considerada, por exemplo, na hora de testar condições de ruptura de filme. Além disso, seu volume alvo (ou volume inicial) é igualado à zero, de modo que um método de controle de volume a faria diminuir até desaparecer.

Além disso, partículas também podem ser convertidas de volta para uma região em dois casos. No primeiro, uma partícula entra em contato com uma região do mesmo material. Este teste é feito comparando o fluido da partícula com aquele da

região que contém seu centro. Se ambos forem iguais, a partícula é unida à região. Cabe ressaltar que, no trabalho de KIM [1], como as partículas também possuem um filme, também é necessário testar a condição de ruptura antes da transformação.

A outra situação é quando a partícula não estiver em contato com nenhuma região do mesmo fluido, mas for suficientemente grande, com volume $V_p >= 1.5V_{min}$. Note que existe um incremento de $0.5V_{min}$ para a conversão à região, em relação à transformação oposta. Esta discrepância pretende evitar mudanças sucessivas entre as representações.

Enfim, para efetivamente, converter uma partícula em região, é reconstruído o *level set* regional em suas proximidades, sendo também atualizado o material das células. Assim, de modo paralelo, é feito com que cada célula verifique se ela está próxima de alguma partícula a ser convertida. Estando até $r_p + \sqrt{3}\Delta x$ de distância do centro da partícula em questão, sua tupla regional e seu tipo de fluido são devidamente alterados. Note que a região e o tipo de fluido de uma célula apenas são modificados quando seu centro está dentro da partícula a ser transformada, enquanto se ele estiver do lado de fora, apenas o valor de ϕ é atualizado.

Capítulo 5

Simulação Multifásica

No capítulo anterior, foi apresentado o ferramental necessário para evoluir a superfície em simulações multifásicas. Contudo, até o momento, não foi introduzido um método numérico capaz de realizar a simulação de mais de uma fase. De fato, as técnicas apresentadas no capítulo 3 devem ser modificadas para que sejam capazes de lidar com este novo paradigma. Desse modo, neste capítulo são abordadas as modificações a serem feitas sobre o método anterior para a correta simulação de fases distintas que interajam entre si.

5.1 Tensão Superficial

Inicialmente, sabe-se que, na presença de dois meios, ocorre o efeito físico da tensão superficial na interface entre os dois materiais. Efeito esse que faz, por exemplo, com que pequenos objetos mais densos do que a água possam flutuar sobre ela e gotículas de água sobre uma superfície mantenham um formato arredondado.

Sua origem é encontrada nas forças de coesão entre as moléculas de um líquido. Dentro da massa fluida, as moléculas são puxadas igualmente em todas as direções. Porém, na superfície, a força resultante dessas forças de coesão não é nula, o que faz com que elas sejam deslocadas para dentro de um dos meios. Assim, a pressão dentro desse meio aumenta, forçando a interface a se contrair para uma área mínima, assumindo a forma mais suave possível para minimizar a energia do sistema.

Logo, também se pode afirmar que a tensão superficial está diretamente relacionada com a diferença de pressão entre os dois lados de uma interface e com a curvatura da superfície. Note que, à princípio, se nenhuma força normal atuar em uma superfície, ela permanece plana. Contudo, se a pressão de um lado for diferente daquela do outro lado, essa desigualdade resulta em uma força ortogonal à superfície. Então, para que as forças de tensão superficial consigam anular essa força normal, a superfície se torna curva.

Portanto, para conferir mais realismo à simulação, é preciso empregar um método

eficiente para levar em conta as forças de tensão superficial existentes. Nesse contexto, aborda-se o método *ghost fluid*, apresentado inicialmente por HONG e KIM [30]. Para sua explicação, utiliza-se o exemplo unidimensional retratado pela figura 5.1, onde as células i e $i+1$ estão em fluidos distintos.

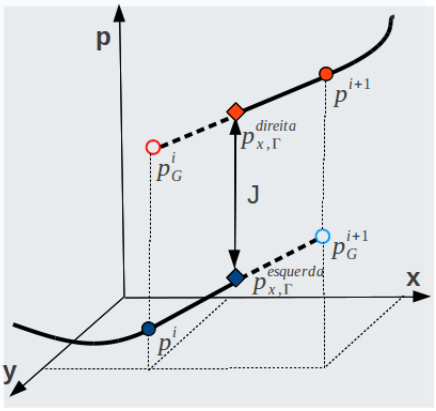

Figura 5.1: Representação da variação de pressão J através de uma interface Γ entre as célula i e $i+1$ e dos valores *ghost* de pressão p_i^G e p_{i+1}^G.

De início, assume-se que, para dois materiais diferentes, a tensão superficial cause um *jump* (ou variação) de pressão J através da superfície Γ. Tal variação possui magnitude $\sigma\kappa_\Gamma$, onde σ é o coeficiente de tensão superficial definido para os dois fluidos em questão, e κ_Γ é a curvatura - com sinal - da superfície entre os meios, sendo obtida pela interpolação das curvaturas dos *level sets* computados nos centros das duas células adjacentes. Apesar de os valores estarem em meios distintos, essa interpolação é válida assumindo que a função da curvatura do *level set* que passa por um ponto, mensurada pela equação 4.5, é contínua.

Contudo, o fato de a pressão não ser contínua através da superfície impede, em princípio, a diferenciação da pressão ao longo da interface, que é necessária para a discretização das equações de Poisson (equação 3.24) e da etapa de projeção (equação 3.25). Assim, para que seja possível calcular a derivada de p na superfície, extrapola-se o valor de pressão p_i para o outro fluido, obtendo o valor *ghost* p_i^G. Do mesmo modo, obtém-se p_{i+1}^G extrapolando p_{i+1} no sentido contrário, a partir da célula $i+1$. Como o *jump* é constante, tem-se:

$$p_i^G = p_i + J \tag{5.1}$$

$$p_{i+1}^G = p_{i+1} - J \tag{5.2}$$

Logo, derivadas laterais da pressão podem ser calculadas como:

$$p_{x,\Gamma}^{esquerda} = \frac{p_{i+1}^{G} - p_i}{\Delta x} = \frac{(p_{i+1} - J) - p_i}{\Delta x} \tag{5.3}$$

$$p_{x,\Gamma}^{direita} = \frac{p_{i+1} - p_i^{G}}{\Delta x} = \frac{p_{i+1} - (p_i + J)}{\Delta x}, \tag{5.4}$$

onde $p_{x,\Gamma}^{esquerda}$ e $p_{x,\Gamma}^{direita}$ são derivadas para o mesmo ponto, mas em relação, respectivamente, aos meios da célula i e $i+1$.

Desse modo, a equação de Poisson (equação 3.24) pode ser discretizada para a célula i como:

$$\begin{aligned}
\nabla^2 p_i = p_{xx,i} = \frac{p_{x,\Gamma}^{esquerda} - p_{x,i-1/2}}{\Delta x} = \frac{\dfrac{p_{i+1}^{G} - p_i}{\Delta x} - \dfrac{p_i - p_{i-1}}{\Delta x}}{\Delta x} = \\
\frac{\dfrac{p_{i+1} - J - p_i}{\Delta x} - \dfrac{p_i - p_{i-1}}{\Delta x}}{\Delta x} = \frac{\rho}{\Delta t} \nabla \cdot \vec{u}_i \\
\frac{\Delta t}{\rho} \left(\frac{p_{i+1} + p_{i-1} - 2p_i}{\Delta x^2} \right) = \nabla \cdot \vec{u}_i + \frac{\Delta t}{\rho} \frac{J}{\Delta x^2} \\
\frac{\Delta t}{\Delta x^2} \left(\frac{\nabla^2 p_i}{\rho} \right) = \nabla \cdot \vec{u}_i + \frac{\Delta t}{\rho} \frac{J}{\Delta x^2}.
\end{aligned} \tag{5.5}$$

Analogamente, para a célula $i+1$ tem-se que:

$$\begin{aligned}
\nabla^2 p_{i+1} = p_{xx,i+1} = \frac{p_{x,i+3/2} - p_{x,\Gamma}^{direita}}{\Delta x} = \frac{\dfrac{p_{i+2} - p_{i+1}}{\Delta x} - \dfrac{p_{i+1} - p_i^{G}}{\Delta x}}{\Delta x} = \\
\frac{\dfrac{p_{i+2} - p_{i+1}}{\Delta x} - \dfrac{p_{i+1} - (p_i + J)}{\Delta x}}{\Delta x} = \frac{\rho}{\Delta t} \nabla \cdot \vec{u}_{i+1} \\
\frac{\Delta t}{\rho} \left(\frac{p_{i+2} + p_i - 2p_{i+1}}{\Delta x^2} \right) = \nabla \cdot \vec{u}_{i+1} - \frac{\Delta t}{\rho} \frac{J}{\Delta x^2} \\
\frac{\Delta t}{\Delta x^2} \left(\frac{\nabla^2 p_{i+1}}{\rho} \right) = \nabla \cdot \vec{u}_{i+1} - \frac{\Delta t}{\rho} \frac{J}{\Delta x^2}.
\end{aligned} \tag{5.6}$$

Note que a diferença básica entre as equações referentes às duas células é o sinal do termo com o J. De modo geral, para uma célula, se seu vizinho à direita for de outro material, soma-se um termo ao lado direito da equação e, se a célula adjacente à esquerda estiver em outro meio, é feita uma subtração.

Portanto, como pode ser visto nas equações 5.5 e 5.6, é possível aplicar forças de tensão superficial com apenas uma pequena modificação no lado direito da equação de Poisson (equação 3.24). Assim, pode-se utilizar o procedimento usual para a resolver. Esta técnica pode ser naturalmente estendida para o caso tridimensional, em que um novo termo poderá ser incluído para cada sentido de cada coordenada.

Contudo, cabe ressaltar que, para trabalhar com um sistema que comporte fluidos distintos, i.e, que possua densidade variável, o cálculo de $\frac{\Delta t}{\rho}\frac{J}{\Delta x^2}$ deve ser realizado usando a densidade amostrada na face entre as células em questão. Desse modo, como a densidade, à principio, não é contínua ao longo da interface, é preciso que seu valor seja computado com base nos fluidos adjacentes, o que é tratado na seção seguinte.

5.2 Densidades de Face

Como visto anteriormente, é preciso que sejam conhecidos os valores de densidade amostrados no centro das faces de cada célula. Assim, neste momento contempla-se como calcular essas densidades.

De inicio, observa-se que, na abordagem de KIM [1], a densidade do filme entre duas regiões é distribuída para essas regiões adjacentes. De fato, seguindo o modelo do autor, ao considerar duas células i e $i+1$ de regiões distintas r_i e r_{i+1}, o valor da densidade no centro da célula i devido à célula $i+1$ pode ser definido como:

$$\begin{aligned} \rho_{i_{i+1}} &= \rho_{r_i}(1-\alpha) + \rho_{filme}(\alpha) \\ \alpha &= \frac{h}{\Delta x}\frac{\phi_i}{\phi_i + \phi_{i+1}}, \end{aligned} \tag{5.7}$$

onde h é a espessura real do filme, ρ_{r_i} é o valor da densidade da região r_i

Logo, para cada vizinho que a célula i possua que esteja em uma região distinta, calcula-se um valor de densidade. Assim, a densidade da célula i será a média das densidades calculadas desse modo. Porém, se nenhuma célula próxima estiver em outra região, a densidade da célula será a própria densidade de sua região. Por fim, para obter a densidade em cada face, é computada a média das densidades das células adjacentes.

Contudo, neste trabalho é empregada a técnica utilizada por KIM *et al.* [51] e KANG *et al.* [28] onde a densidade no centro da face entre duas células de regiões diferentes é dado pela interpolação das densidades de suas regiões, de modo que ela seja contínua mesmo na interface. Por exemplo, para duas células i e $i+1$, inicialmente é calculada a tupla regional $\langle r_{face}, \phi_{face}\rangle$ no centro da face. Então, é aplicada a função *Heaviside* (equação 4.6), com ϵ igual à metade da espessura do filme h, para calcular:

$$\rho_{i+1/2,j,k} = \rho_{fora} + (\rho_{dentro} - \rho_{fora})H(\phi_{face}), \tag{5.8}$$

onde $\rho_{dentro} = \rho_{face}$ e ρ_{fora} a região do outro lado do filme.

5.3 Sistema de Equações de Poisson

Tendo em vista que, para uma simulação multifásica, como é preciso resolver a equação de Poisson (equação 3.24) no caso em que a densidade ρ é variável, a discretização dada pela equação 3.31 não é mais válida. Assim, apresenta-se uma nova discretização que pode ser igualmente resolvida utilizando o método do gradiente conjugado apresentado na seção 3.9.

A técnica utilizada se baseia no trabalho de LIU *et al.* [29] onde é abordado um método que permite resolver, para domínios cartesianos 3D, equações de Poisson do formato:

$$(\beta q_x)_x + (\beta q_y)_y + (\beta q_z)_z = f(\vec{x}), \tag{5.9}$$

onde os sub-índices x, y e z denotam a derivada parcial em relação aos seus respectivos componentes, β é um fator variável e q é a quantidade para a qual se deseja resolver a equação, estando sujeita à especificação do *jump* através da interface Γ que deve atender às seguintes equações:

$$[q]_\Gamma = a(\vec{x}_\Gamma) \tag{5.10}$$

$$[\beta q_n]_\Gamma = b(\vec{x}_\Gamma) \tag{5.11}$$

$$\text{com } q_n = (\nabla q) \cdot \hat{N},$$

onde o operador $[A]_\Gamma = A_{fora} - A_{dentro}$ é o *jump* (ou variação) de A através da superfície Γ.

Assim sendo, para resolver numericamente um sistema de equações de Poisson, LIU *et al.* [29] o discretiza para cada célula (i, j, k):

$$\begin{aligned}
&\frac{\beta_{i+1/2,j,k}\left(\frac{q_{i+1,j,k}-q_{i,j,k}}{\Delta x}\right) - \beta_{i-1/2,j,k}\left(\frac{q_{i,j,k}-q_{i-1,j,k}}{\Delta x}\right)}{\Delta x} + \\
&\frac{\beta_{i,j+1/2,k}\left(\frac{q_{i,j+1,k}-q_{i,j,k}}{\Delta x}\right) - \beta_{i,j-1/2,k}\left(\frac{q_{i,j,k}-q_{i,j-1,k}}{\Delta x}\right)}{\Delta x} + \\
&\frac{\beta_{i,j,k+1/2}\left(\frac{q_{i,j,k+1}-q_{i,j,k}}{\Delta x}\right) - \beta_{i,j,k-1/2}\left(\frac{q_{i,j,k}-q_{i,j,k-1}}{\Delta x}\right)}{\Delta x} = f_{i,j,k} + F^x + F^y + F^z,
\end{aligned} \tag{5.12}$$

onde F^x, F^y e F^z dependem diretamente das condições de *jump* $a(\vec{x}_\Gamma)$ e $b(\vec{x}_\Gamma)$, e os valores de β são definidos nos centros das faces de cada célula. Nesse sentido, os incrementos $+1/2$ e $-1/2$ nos índices de um $\beta_{i,j,k}$ denotam a face da célula (i, j, k) onde ele é estimado. Por exemplo $\beta_{i+1/2,j,k}$ está localizado na face entre as células (i, j, k) e $(i + 1, j, k)$.

Aplicando 5.12 para o caso da resolução da equação 3.24, pode-se atribuir

$q_{i,j,k} = p_{i,j,k}$, $\beta_{i,j,k} = \dfrac{\Delta t}{\rho_{i,j,k}}$ e $f_{i,j,k} = \nabla \cdot \vec{u}_{i,j,k}$. Além disso, neste trabalho a condição 5.10 para a variação da pressão através da interface é tratada através da aplicação de forças de tensão superficial, conforme descrito na seção 5.1. Logo, é possível estabelecer que $a(\vec{x}_\Gamma) = 0$ considerando, nesse caso, que $f_{i,j,k}$ já incorpora a contribuição da tensão superficial.

Outrossim, segundo KANG *et al.* [28], pode-se considerar $\left[\dfrac{p_x}{\rho}\right]_\Gamma = \left[\dfrac{p_y}{\rho}\right]_\Gamma = \left[\dfrac{p_z}{\rho}\right]_\Gamma = 0$ ao resolver a equação de Poisson. Um modo intuitivo de chegar a essa conclusão decorre de duas observações. Primeiramente, a densidade é feita contínua, mesmo nas interfaces, como discutido na seção 5.2. Segundo, a aproximação de uma derivada parcial da pressão na interface, por exemplo p_x quando $p_{i,j,k}$ e $p_{i+1,j,k}$ estão em meios distintos é realizada por meio das expressões 5.3, para o meio contendo $p_{i,j,k}$, e 5.4, para o outro que detém $p_{i+1,j,k}$. Como o valor do *jump* acrescentado em um caso e subtraído no outro é o mesmo, essas equações são idênticas e, assim, $[p_x] = 0$. Logo, tendo em vista a restrição 5.11, confere-se $b(\vec{x}_\Gamma) = 0$.

Portanto, como $a(\vec{x}_\Gamma) = b(\vec{x}_\Gamma) = 0$, os termos F^x, F^y e F^z também são nulos, o que permite modificar a equação 5.12 para:

$$\begin{aligned}\frac{\Delta t}{\Delta x^2}\Bigg[&\left(\frac{p_{i+1,j,k} - p_{i,j,k}}{\rho_{i+1/2,j,k}}\right) + \left(\frac{p_{i-1,j,k} - p_{i,j,k}}{\rho_{i-1/2,j,k}}\right) + \\ &\left(\frac{p_{i,j+1,k} - p_{i,j,k}}{\rho_{i,j+1/2,k}}\right) + \left(\frac{p_{i,j-1,k} - p_{i,j,k}}{\rho_{i,j-1/2,k}}\right) + \\ &\left(\frac{p_{i,j,k} - p_{i,j,k+1}}{\rho_{i,j,k+1/2}}\right) + \left(\frac{p_{i,j,k-1} - p_{i,j,k}}{\rho_{i,j,k-1/2}}\right)\Bigg] = \nabla \cdot \vec{u}_{i,j,k},\end{aligned} \tag{5.13}$$

que também pode ser resolvida pelo método usual. Além do mais, se a densidade ρ fosse contínua, a equação recairia na expressão 3.31.

Finalmente, os valores de pressão obtidos a partir da equação anterior devem ser utilizados para atualizar os componentes da velocidade na grade, segundo a equação 3.25. Ainda, lembrando que a densidade é variável dentro da massa fluida, é imperativo que, ao calcular as atualizações de cada componente da velocidade, seja utilizada a densidade computada no centro da face entre as células em questão. Assim, componentes $u_{i,j,k}$, $v_{i,j,k}$ e $w_{i,j,k}$ são atualizadas conforme as equações:

$$u^t_{i,j,k} = u_{2,\,i,j,k} - \frac{\Delta t}{\Delta x}\frac{p^t_{i,j,k} - p^t_{i-1,j,k}}{\rho_{i-1/2,j,k}}, \tag{5.14}$$

$$v^t_{i,j,k} = v_{2,\,i,j,k} - \frac{\Delta t}{\Delta x}\frac{p^t_{i,j,k} - p^t_{i,j-1,k}}{\rho_{i,j-1/2,k}}, \text{ e} \tag{5.15}$$

$$w^t_{i,j,k} = w_{2,\,i,j,k} - \frac{\Delta t}{\Delta x}\frac{p^t_{i,j,k} - p^t_{i,j,k-1}}{\rho_{i,j,k-1/2}}, \tag{5.16}$$

onde o índice subscrito 2 em um componente denota seu valor após as etapas de advecção e aplicação de forças externas.

Capítulo 6

Controle de Volume

Sabe-se que o problema da perda de volume costuma ser uma constante na área da simulação de fluidos, sendo geralmente resultado de erros numéricos ou de limitações da discretização empregada. Pode, inclusive, estar ligado ao próprio método pelo qual se faz a massa fluida evoluir. Assim sendo, em simulações multifásicas onde há vários volumes a serem considerados, e variações topológicas ocorrem mais comumente, esse problema tende a ter maior relevância.

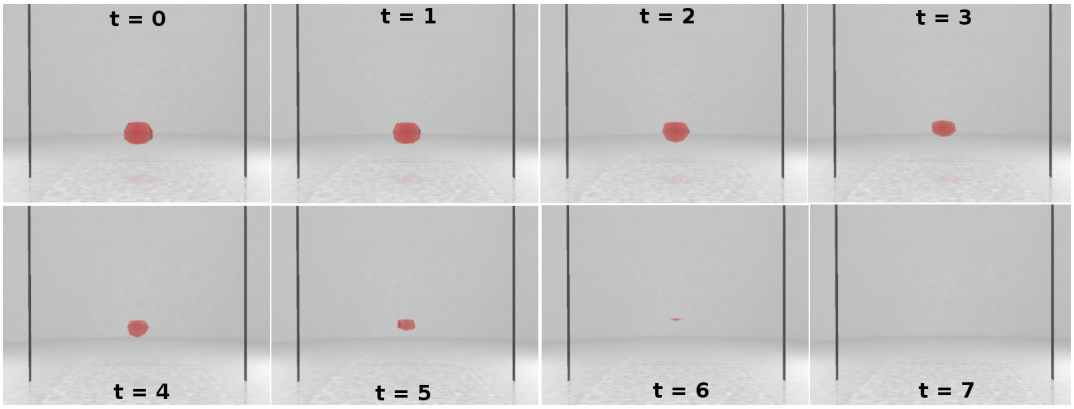

Figura 6.1: Iterações sucessivas da simulação em que uma bolha de ar envolta por água perde volume até desaparecer enquanto ascende sob efeito de forças de empuxo.

Certamente, pode ocorrer, por exemplo, que uma fase perca volume ao longo das iterações, podendo chegar até a desaparecer. Por exemplo, a figura 6.1 ilustra o caso em que uma fase relativa a uma bolha de ar envolta por água desaparece completamente da simulação. Em outro caso, uma fina lâmina de um fluido que escorre para dentro de um copo pode ter seu deslocamento impedido, o que causa que o fluido pare de encher o recipiente. Tendo isso em vista, é preciso um método eficiente que garanta a conservação do volume de cada fluido ao longo da simulação.

Para começar, é possível reduzir a perda causada pela etapa da advecção ao utilizar métodos de ordem superior, apesar de, ao longo do tempo, a perda ainda poder se tornar substancial. Como exemplos de técnicas que podem ser empregadas,

apontam-se o *Back and Forth Error Compensation and Correction* (BFECC) (DUPONT e LIU [56], KIM *et al.* [57]), o MacCormack modificado (SELLE *et al.* [58]), e a utilização de polinômios cúbicos para as interpolações (SONG *et al.* [59], TAKAHASHI *et al.* [60]).

Além disso, também é razoável usar a abordagem híbrida do *particle level set* (PLS) (ENRIGHT *et al.* [16]), que reduz a perda de volume à uma pequena quantidade, mesmo que este valor possa se acumular até se tornar visível (LOSASSO *et al.* [23], GOKTEKIN *et al.* [61]). Utilizando esse método, o detalhamento das superfícies poderia ser aumentado ao se renderizar partículas escapadas de seu meio original (GUENDELMAN *et al.* [20]). Por fim, é possível combinar o *level set* com métodos *volume of fluid* (VOF), como no trabalho de SUSSMAN [62]. Nesse caso, realmente não se tem perda de volume pois se faz um acompanhamento do volume presente em cada célula.

Todavia, entre as possíveis soluções para controlar o volume da simulação, optou-se por analisar o método proposto por KIM *et al.* [51], que também emprega o *level set* regional. De início, observa-se que ele pode ser usado em conjunto com os métodos anteriores para melhorar o resultado ou possibilitar mudanças controladas de volume. Ademais, não se preocupa com as fases de líquido, tratando apenas do volume de bolhas de ar. Nesse sentido, diferentes fases líquidas são sempre fundidas ao se tocarem. Além disso, não apresenta solução para regiões menores que um determinado limiar, de modo que pequenas gotas continuam desaparecendo.

Sua proposta é forçar um divergente não nulo para o campo velocidade em cada região, de modo a inflar ou desinflar bolhas de ar, mantendo os volumes desejados. A noção de controlar um processo, no caso a variação de volume, através do divergente da velocidade, denominada originalmente de *divergence sourcing*, foi adaptada a partir de outros contextos. Ao certo, FELDMAN *et al.* [63] trata partículas como fontes de divergência para simular explosões em sistemas de partículas. Ainda, LOSASSO *et al.* [64] utiliza o divergente para modelar a expansão de um combustível sólido ao estado gasoso.

Para a efetiva execução da técnica, antes de calcular o divergente desejado para cada região, inicialmente é calculado o erro e_r^t do volume da região r no instante t, dado por:

$$e_r^t = \frac{V_r^t - V_r^{t_0}}{V_r^{t_0}}, \tag{6.1}$$

onde $V_r^{t_0}$ é o volume inicial (ou volume-alvo) da região r. Em sequência, o erro é acumulado ao longo das iterações na variável obtida por:

$$b_r^t = b_r^{t-1} + e_r^t \Delta t. \tag{6.2}$$

Depois, com base nos erros atual e acumulado, o divergente c_r^t é computado por:

$$c_r^t = \frac{1}{e_r^t + 1}(-k_p e_r^t - k_l b_r^t), \tag{6.3}$$

$$\text{com } k_p = \frac{2.3}{25\Delta t} \text{ e } k_l = \frac{k_p^2 \Delta t}{16} \tag{6.4}$$

Por fim, é garantido que $\nabla \cdot \vec{u}_r^{t+1} = c_r^t$ ao resolver, na etapa da projeção:

$$\frac{\Delta t}{\rho}\nabla^2 p^t = \nabla \cdot \vec{u}^t + c_r^t. \tag{6.5}$$

Ou seja, ao empregar a discretização da equação 5.13 para uma célula (i, j, k) cujo centro está contido em uma região r, é acumulado c_r^t em seu lado direito, modificando o divergente $\nabla \cdot \vec{u}^t$. Dessa forma, quando os componentes de velocidade forem alterados pela pressão através das equações 5.14, 5.15 e 5.16, ao invés do campo velocidade em uma região se tornar nulo, ele se igualará ao seu respectivo c_r^t.

6.1 Método Proposto

Com base no método de KIM *et al.* [51], surgiu a abordagem proposta, cuja ideia inicial era garantir, de modo aproximado, o divergente da equação 6.3 através da alteração das velocidades próximas à superfície de cada região. Contudo, as correções de volume acabaram por serem efetivamente realizadas através de um campo auxiliar utilizado exclusivamente para a evolução do *level set* regional e não através de alterações diretas no campo de velocidade dos fluidos. Adicionalmente, procura-se preservar a direção do deslocamento de cada região, de forma que a variação do volume não seja omnidirecional como na técnica anterior.

Para tanto, ao invés de forçar um determinado valor para o divergente do campo velocidade em uma região através da equação 6.5, é empregado um conjunto de sub-iterações de advecção do *level set* regional, adicionais à execução do esquema de advecção original. Essas sub-iterações são realizas em sub-passos t' da t-ésima iteração da simulação, abrangendo, cada um, uma duração $\Delta t'$, que é uma fração do tamanho do passo de tempo Δt. Além disso, em cada um desses passos, é utilizado um campo de velocidades distinto, definido com base no erro do volume de cada região, recalculado em cada passo.

Por conseguinte, os valores desses campos são definidos de modo a apenas não serem nulos próximo às interfaces. Devem fazer, ainda, com que cada região se aproxime do seu volume-alvo (ou volume inicial). De modo geral, pode-se considerar que essa abordagem é menos sujeita a erros numéricos, pois os valores são calculados diretamente, e mais previsível, pois o resultado não depende de um complexo processo

de otimização.

Para a criação do campo auxiliar, definem-se, em primeiro lugar, componentes de velocidade Vel^t_{face} - que podem se referir a u, v ou w - localizados no centros de faces cujos centros das células adjacentes estão em regiões distintas. Assim, dentro do campo auxiliar de velocidade, apenas componentes relativos a esses valores do campo velocidade original serão não nulos . Em seguida, é obtida a região r nas respectivas posições em que eles são amostrados. Determinam-se, então, seus respectivos valores $Vel^{t'}_{face}$ no campo auxiliar, de modo que cada um propicie, pela advecção, um acréscimo ou decréscimo estimado no volume de sua respectiva região r de:

$$(Vel^{t'}_{face}\Delta t')\Delta x^2. \tag{6.6}$$

6.1.1 Método das Correções Egoístas

Desenvolveu-se um método que procura corrigir o erro individual de cada região, realizando correções de volume locais através do campo de velocidades auxiliar. Logo, de início, cabe ressaltar que, como essas correções apenas objetivam corrigir o volume individual de cada região r que contém um respectivo componente Vel_{face}, sem se preocupar com o erro das regiões ao seu redor, pode acontecer que, por exemplo, ao se incrementar localmente o volume de uma região, seja reduzido desnecessariamente o volume de outra adjacente. Por isso, pode ser que o erro do volume não seja corrigido de forma ótima para todas as regiões.

Para garantir a correção do volume, seria necessário um modelo global que levasse em consideração o erro do volume de cada região e indicasse as velocidades auxiliares que garantissem correções de volume de modo que o erro global fosse mínimo. Nesse contexto, poder-se-ia definir, por exemplo, o quanto do volume de uma região deve ser transferido para uma adjacente através de uma interface e resolver um problema de otimização para obter a melhor configuração desse fluxo de volume. Entretanto, restringe-se este trabalho, ainda, ao modelo egoísta, assim denominado por se preocupar apenas com o erro individual de cada região. Dessa forma, o modelo mais robusto é deixado como uma sugestão de trabalho futuro.

Segundo o método egoísta, através de cada componente de velocidade auxiliar, procura-se forçar que a variação de volume que ele gera seja uma fração da correção total feita no volume de sua região r. Dessa forma, pode-se definir seu valor a partir do erro do volume dessa região e da razão entre Vel^t_{face} e a soma de todos os componentes que são da região r e estão em faces cujas células adjacentes são um de r e outro de uma região diferente:

$$\mid V_r^{t_0} - V_r^{t'} \mid \frac{\mid Vel^t_{face} \mid}{\sum_{face \in r} \mid Vel^t_{face} \mid}. \tag{6.7}$$

Logo, igualando as equações 6.6 e 6.7, é definido o valor $Vel^{t'}_{face}$ como:

$$(Vel^{t'}_{face}\Delta t')\Delta x^2 = \mid V_r^{t_0} - V_r^{t'} \mid \frac{\mid Vel^{t}_{face} \mid}{\sum_{face \in r} \mid Vel^{t}_{face} \mid}$$

$$Vel^{t'}_{face} = \frac{\mid V_r^{t_0} - V_r^{t'} \mid}{\Delta t' \Delta x^2} \frac{\mid Vel^{t}_{face} \mid}{\sum_{face \in r} \mid Vel^{t}_{face} \mid}. \tag{6.8}$$

Todavia, ainda é preciso estabelecer o sinal de $Vel^{t'}_{face}$. A ideia por traz dessa escolha está em fazer com que o movimento da interface seja acelerado ou freado em porções distintas da superfície, de modo a garantir a correção volumétrica desejada. Para ajudar nessa definição, considera-se N_{face} como o componente na direção perpendicular à face em questão do vetor normal $\hat{N}$, que é estimado no centro dessa face e que aponta para dentro da região que está sendo tratada.

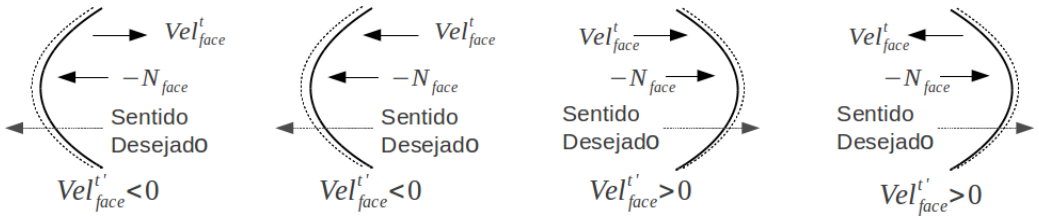

Figura 6.2: Diferentes configurações que definem o sinal do componente auxiliar $Vel^{t'}_{face}$ quando $(V_r^{t_0} - V_r^{t'}) > 0$.

Por exemplo, na figura 6.2 são exibidos diferentes configurações da direção normal e da velocidade de um componente quando $(V_r^{t_0} - V_r^{t'}) > 0$, i.e., quando se quer aumentar o volume de uma região. Nesse contexto, no primeiro caso à esquerda, o incremento é definido para fazer com que movimentação da interface seja freada. Já no segundo, o deslocamento é favorecido. Ademais, analogamente às situações da figura 6.2, a figura 6.3 exibe os casos relativos a quando $(V_r^{t_0} - V_r^{t'}) < 0$.

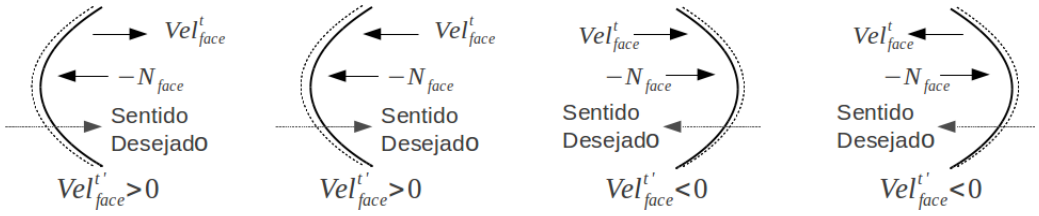

Figura 6.3: Diferentes configurações que definem o sinal do componente de velocidade auxiliar $Vel^{t'}_{face}$ quando $(V_r^{t_0} - V_r^{t'}) < 0$.

Como regra geral, pode-se deduzir que o sinal do componente $Vel^{t'}_{face}$ é dado por:

$$sinal(Vel^{t'}_{face}) = sinal(-N_{face} \cdot (V_r^{t_0} - V_r^{t'})). \tag{6.9}$$

Além da simples utilização do erro atual para corrigir o volume, também é levado

em conta o histórico do erro próximo à interface. São acumulados os valores $Vel^{t'}_{face}$ a partir de quando um componente se torna próximo à interface. Porém, esse valor acumulado é zerado quando ele se afasta novamente. Assim, assumindo que $Vel^{t'}_{face}$ está suficientemente próximo de uma interface em t', o erro atual e o acumulado são somados segundo uma ponderação baseada no trabalho de KIM *et al.* [51]:

$$Vel^{t'}_{face} = \frac{1}{\Delta x^2}\left\{ k_p \left[\mid V_r^{t_0} - V_r^{t'} \mid \frac{\mid Vel^t_{face} \mid}{\sum_{face \in r} \mid Vel^t_{face} \mid} \right] + k_l \left[\sum_{\tau'=t_0^{face}}^{t'} \left(\mid V_{r^{\tau'}}^{t_0} - V_{r^{\tau'}}^{\tau'} \mid \frac{\mid Vel^{\tau}_{face} \mid \Delta\tau'}{\sum_{face \in r^{\tau'}} \mid Vel^{\tau}_{face} \mid} \right) \right] \right\}, \quad (6.10)$$

onde $k_p = \frac{1}{\Delta t}$, $k_l = \frac{k_p^2 \Delta t'}{16} = \frac{1}{16\Delta t'}$. Ainda, t_0^{face} indica quando um componente Vel_{face} passou a estar em uma face cujas células adjacentes são de regiões distintas, e $r^{\tau'}$ é a região que contém a posição do componente Vel^t_{face} no sub-passo τ' de tamanho $\Delta\tau'$ da iteração τ.

Por fim, note que, seguindo a formulação anterior e sem considerar os dados do histórico do erro, uma velocidade auxiliar $Vel^{t'}_{face}$ é sempre diretamente proporcional ao seu respectivo Vel^t_{face}. Desse modo, pode-se dizer que a correção do volume é dada na mesma direção do movimento do fluido. Todavia, essa ideia ainda pode ser melhor aplicada se, ao invés de utilizar $\mid Vel^t_{face} \mid$ para calcular a proporção do erro volumétrico a ser corrigido, fosse empregada a projeção da velocidade $\vec{u}^t$ na normal $\hat{N}^{t'}$, ambas estimadas na posição do componente em questão.

Logo, quanto menor for o angulo entre $\vec{u}^t$ e $\hat{N}^{t'}$, maior será a contribuição do respectivo componente na correção do erro volumétrico. Essa abordagem pode ser justificada ao se pensar que apenas velocidades normais à superfície podem fazer com que o volume de uma fase seja alterado. Assim sendo, a equação 6.10 pode ser reescrita como:

$$Vel^{t'}_{face} = \frac{1}{\Delta x^2}\left\{ k_p \left[\mid V_r^{t_0} - V_r^{t'} \mid \frac{\mid \vec{u}^t \cdot \hat{N}^{t'} \mid}{\sum_{face \in r} \mid \vec{u}^t \cdot \hat{N}^{t'} \mid} \right] + k_l \left[\sum_{\tau'=t_0^{face}}^{t'} \left(\mid V_{r^{\tau'}}^{t_0} - V_{r^{\tau'}}^{\tau'} \mid \frac{\mid \vec{u}^{\tau} \cdot \hat{N}^{\tau'} \mid \Delta\tau'}{\sum_{face \in r^{\tau'}} \mid \vec{u}^{\tau} \cdot \hat{N}^{\tau'} \mid} \right) \right] \right\} \quad (6.11)$$

Como um exemplo de resultado da aplicação dessa técnica, a figura 6.4 ilustra as mesmas iterações da simulação do movimento de uma bolha de ar em uma massa de água que são exibidas na figura 6.1, mas com o diferencial da aplicação do método de controle de volume aqui descrito. Além disso, exibe-se no gráfico da figura 6.5 a variação do volume da bolha durante a simulação com e sem o controle de volume.

Ao analisar esse gráfico, cabe manter em mente que ele foi criado para uma grade de baixa resolução e que, à medida que a resolução fosse aumentada, mais preciso seria o cálculo do volume, além do método proposto se tornar mais eficiente.

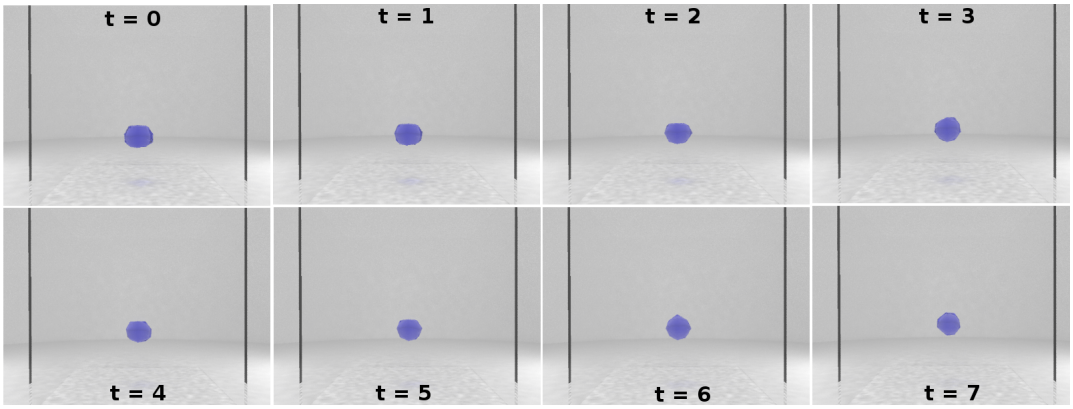

Figura 6.4: Iterações sucessivas da simulação em que uma bolha de ar envolta por água ascende sob efeito de forças de empuxo, enquanto tem seu volume controlado pelo método proposto.

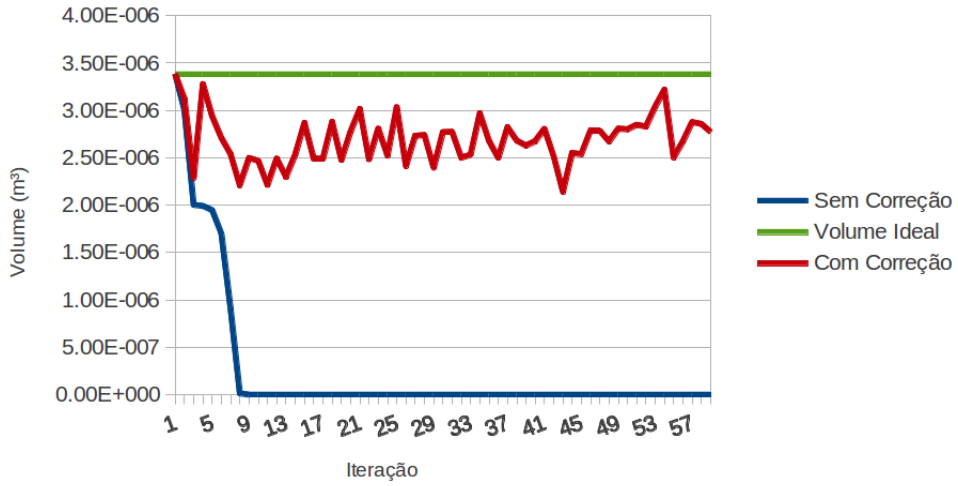

Figura 6.5: Gráfico da variação do volume de uma bolha de ar que ascende sob efeito de forças de empuxo em uma simulação realizada em uma grade de volume $0.1^3 m^3$ com 22^3 células.

6.1.2 Fluxos de Entrada e Saída

Tendo o volume inicial (ou volume-alvo) como fonte da correção, é preciso tratar de situações em que fluidos entram ou saem da grade computacional, ocasionando a alteração de seus respectivos volumes-alvo. De fato, deve-se aumentar o volume-alvo de regiões relativas aos fluidos que entram e o diminuir para aqueles que saem. Para tratar desses casos, inicialmente são definidas as células de entrada e saída e suas respectivas velocidades as quais determinam a alteração do volume de uma região.

Por definição, uma face é de saída (ou *output*) se ela se encontra em um dos limites da grade e o componente de velocidade $Vel_{saída}$ nessa face aponta para o lado de fora. Por outro lado, se esse componente apontar para dentro da massa fluida, a face é considerada de entrada (ou *input*), e o componente passa a ser indicado por $Vel_{entrada}$. Estes podem ser definidos pela simulação em si, serem fixos ou variarem de forma controlada.

Logo, para cada face de entrada, deve-se adicionar:

$$(Vel_{entrada}\Delta t)\Delta x^2 \tag{6.12}$$

ao volume inicial da região da célula a que pertence, equivalente ao volume que passa pela face no intervalo Δt. Assim, o total adicionado é igual a:

$$\sum_{entradas} (Vel_{entrada}\Delta t)\Delta x^2. \tag{6.13}$$

De forma similar, o volume total que sai pelas faces de saída é:

$$\sum_{saídas} (Vel_{saída}\Delta t)\Delta x^2. \tag{6.14}$$

Como os fluidos em questão são incompressíveis, então, considerando todos os fluidos que ocupam o volume onde se faz a simulação, tem-se que:

$$\sum_{entradas} (Vel_{entrada}\Delta t)\Delta x^2 = \sum_{saídas} (Vel_{saída}\Delta t)\Delta x^2, \tag{6.15}$$

ou seja, deve sair o mesmo volume total que entrou. Entretanto, essa equação pode não ser satisfeita em um dado momento da simulação devido às mesmas razões que determinaram a perda de volume. Nesse caso, é preciso restabelecer a condição de incompressibilidade para toda a massa fluida a fim de que todo o volume que entre também saia da grade. Assim, para cada face de saída, é retirado do volume inicial da região que engloba seu centro, uma fração do volume total de entrada, determinada por:

$$\frac{Vel_{saída}}{\sum_{saídas} Vel_{saída}} \sum_{entradas} (Vel_{entrada}\Delta t\Delta x^2). \tag{6.16}$$

6.1.3 Partículas

De modo adicional, pode-se controlar o volume de partículas ao estabelecer um raio-alvo a partir do volume que cada uma deveria abranger. Cabe lembrar que o volume inicial (ou volume-alvo) $V_p^{t_0}$ de uma partícula p é igual àquele da região que

a originou. Assim, define-se:

$$V_p^{t_0} = \frac{4}{3}\pi(r_p^{t_0})^3 \rightarrow r^{t_0} = \sqrt[3]{\frac{3}{4}\frac{V_p^{t_0}}{\pi}}, \tag{6.17}$$

de modo que o raio da partícula possa ser atualizado pela equação:

$$r_p^{t+1} = r_p^t + \left(\frac{r_p^{t_0} - r_p^t}{t_{c_p}}\right), \tag{6.18}$$

onde $t_{c_p} = 20$ regula o número de iterações necessárias para que uma partícula tenha seu volume corrigido. Logo, se acontecer de alguma região perder tanto volume ao ponto de ser transformada em partícula, esta pode recuperar as dimensões antes possuídas, podendo, inclusive, voltar a se tornar uma região em decorrência do seu volume ser maior do que o limiar determinado.

Ainda em relação às partículas, é preciso saber como atualizar os volumes-alvo das regiões quando uma partícula é criada ou convertida para uma região. De fato, no primeiro caso, uma região é convertida em partícula, de modo que ela é removida da grade, tendo seu volume ocupado pelas demais regiões. Assim, uma quantidade de volume-alvo é retirada da grade e, se deixado sem ajuste, o controle de volume poderá se tornar instável, pois nunca serão atingidos os volumes-alvo.

Portanto, quando uma região é convertida em partícula, seu volume-alvo é distribuído entre as regiões restantes de acordo com suas proporções de volume-alvo em relação ao volume fluido total. Analogamente, quando é realizada a transição de uma partícula para uma região, o volume-alvo de cada região que já existia é subtraído de uma parte do volume-alvo da nova região, sendo esta divisão ponderada pela fração do volume-alvo total que a região já existente ocupa. Além disso, devido às diferenças entre as representações de partícula e região, o volume-alvo da região criada não é obtido diretamente daquele da partícula que a originou. Em vez disso, ele é igual ao volume ocupado pela região após a reconstrução do *level set* nas proximidades da partícula.

Capítulo 7

Visualização

Após ter sido apresentada cada etapa da simulação, resta descrever como o resultado é visualizado. Primeiramente, foram criados modos de visualização utilizando a biblioteca do OpenGL [32] para a linguagem C++. Essas diferentes visões permitem que a simulação possa ser visualizada de quatro maneiras distintas à medida que ela é realizada. Além disso, permite-se pausar os cômputos para navegar livremente nas representações disponíveis, analisando uma iteração específica.

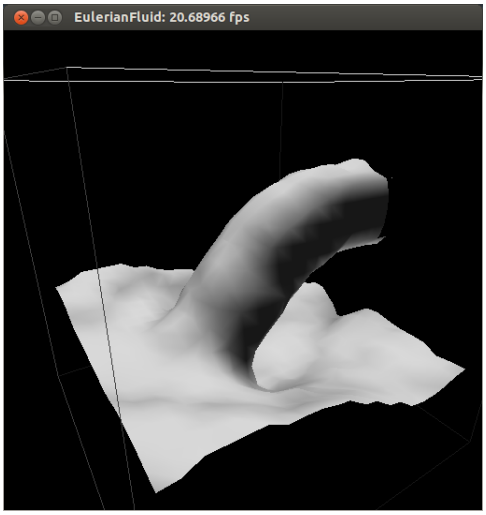

Figura 7.1: Exemplo da visualização da malha de triângulos extraída do *level set* regional em uma grade de 22^3 células.

A primeira forma de exibição é constituída por uma malha poligonal extraída da isosuperfície de nível zero definida pelo campo escalar do *level set*. Ainda, no caso das partículas, são construídas esferas aproximadas a partir de sucessivas subdivisões das faces de um icosaedro regular. Um exemplo deste caso pode ser visto na figura 7.1, onde um fluido é impulsionado para dentro da grade. À principio, essa triangulação da isosuperfície pode ser obtida simplesmente através do algoritmo do *marching cubes* (LORENSEN e CLINE [47]). Contudo, como ela vai ser futuramente exportada para a geração de uma visualização mais realista, alguns detalhes

ainda serão elucidados. Já as demais visões somente são empregadas para testes, permitindo visualizar certas propriedades dos fluidos em cortes paralelos aos eixos ordenados.

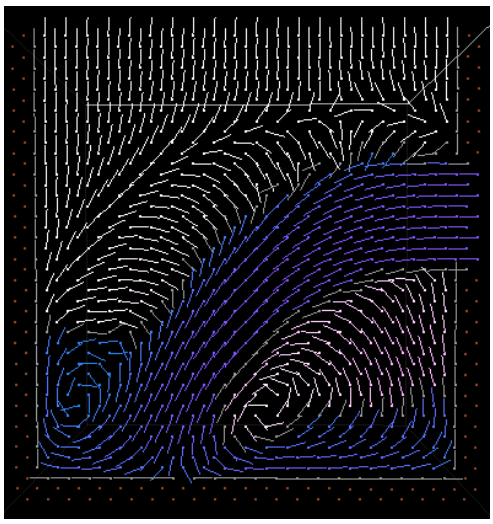

Figura 7.2: Exemplo da visualização do campo velocidade em um plano transversal ao eixo z em uma grade de 22^3 células.

Nesse contexto, o segundo tipo expõe o campo velocidade estimado nas posições referentes à cada componente de velocidade. De fato, em cada uma dessas localidades, é obtido um vetor velocidade através da interpolação dos valores dos componentes próximos e é desenhado um segmento de reta de tamanho fixo na direção da projeção dessa velocidade no plano de visualização. Além disso, sua cor base depende da cor do fluido na posição desejada, podendo sofrer variações em função do quanto a magnitude de sua velocidade está próxima do valor atual máximo para toda a grade. Assim, pode-se verificar onde o campo possui valores altos ou baixos, ou mesmo localizar máximos e mínimos. À exemplo, a ilustração 7.2 mostra essa representação aplicada a uma das possíveis secções que podem ser obtidas no mesmo instante da imagem anterior (figura 7.1).

Para o terceiro arranjo, os segmentos exibidos estão relacionados ao valor do campo escalar do *level set*. Através deles, representam-se os negativos dos gradientes da função distância à borda, de modo que cada um aponta para seu respectivo ponto mais próximo em uma superfície. Vale lembrar que tais valores somente estão necessariamente corretos nos pontos próximos à superfície, visto que posições longes dela não têm seus valores reinicializados em cada iteração. Desse modo, é possível averiguar onde cada interface está localizada segundo o *level set*, podendo identificar situações indesejadas. Ademais, a coloração dos segmentos de reta permite distinguir qual fluido está em cada ponto. Considerando o mesmo estado do sistema da figura 7.2 e usando o mesmo plano de corte, retrata-se na figura 7.3 o campo distância tal como aqui descrito.

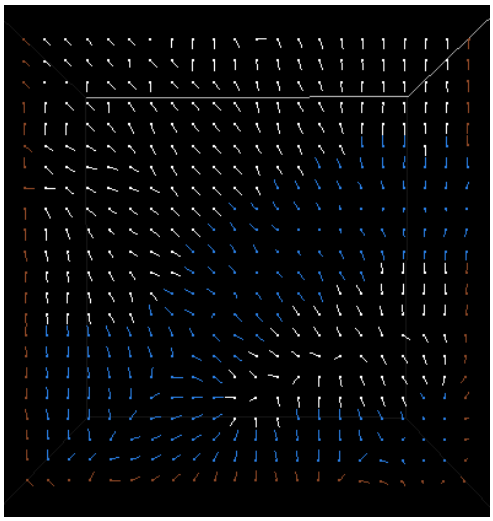

Figura 7.3: Exemplo da visualização do campo distância do *level set* regional em um plano transversal ao eixo z em uma grade de 22^3 células.

Por fim, a última configuração retrata a regionalização do *level set* regional. À vista disso, o quadrado definido pela interseção de uma célula e o plano de visualização é colorido com a cor referente à região no centro da célula. Apesar de esta maneira não permitir a visualização exata da regionalização, que depende da interpolação das tuplas regionais de células adjacentes, constitui uma boa forma de visualizar a distribuição espacial das regiões e buscar possíveis erros. Por exemplo, a figura 7.4 denota a configuração regional equivalente à secção das imagens anteriores (figuras 7.2 e 7.3).

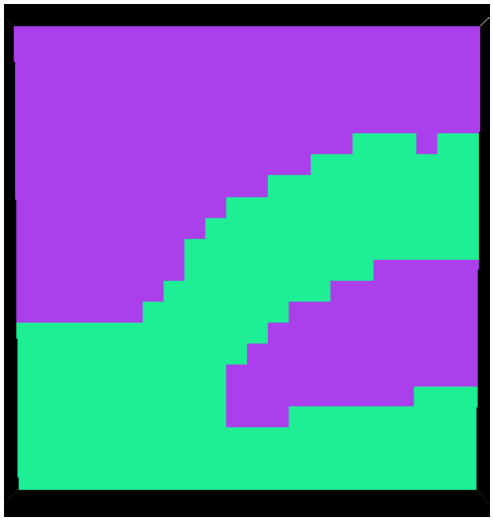

Figura 7.4: Exemplo da visualização da distribuição regional em um plano transversal ao eixo z em uma grade de 22^3 células.

Para a visualização final, é utilizado o *raytracer* Yafaray [34]. Exemplos desse tipo de visão podem ser encontrados nas figuras 7.5 e 7.6. Para que elas sejam criadas, como se trabalha com superfícies e não com volumes, é preciso definir pro-

priedades para cada interface onde ocorre uma mudança de meio. Logo, ao invés da simples criação de uma única malha para o sistema usando o *marching cubes*, é criada uma para cada interface, atribuindo a ela suas devidas propriedades. Além disso, como é considerado um fino filme entre cada fase (ou região), na verdade, cada interface do *level set* equivale a duas interfaces reais: uma que separa o primeiro meio do filme e outra que dissocia este do segundo meio. Desse modo, o filme entre duas regiões é criado pelo deslocamento da interface do *level set* para dentro de cada região de modo que duas superfícies sejam criadas. Contudo, no caso das partículas, como elas não possuem filme, apenas a superfície da esfera é considerada.

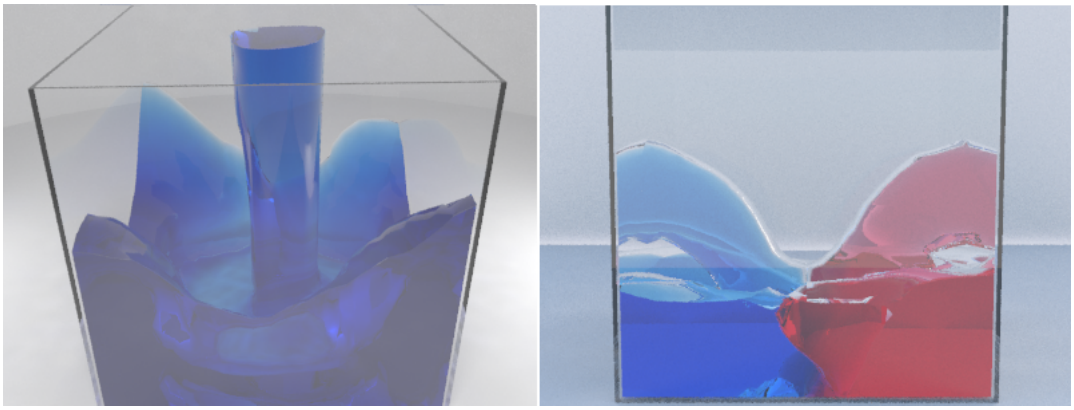

Figura 7.5: Exemplo de duas visualizações finais, onde a primeira apresenta a entrada de um fluido em uma região ocupada por ar enquanto a segunda retrata a introdução de dois fluidos distintos.

Portanto, para cada região r_i, são construídas suas interfaces com cada uma das regiões adjacentes r_j utilizando o *marching cubes*, mas deslocando tal superfície em um valor $\epsilon = 0.1 * \frac{\Delta x}{2}$ para dentro de r_i, de modo que o filme gerado possua espessura $h = 0.1 * \Delta x$. Este deslocamento é realizado ao se avaliar o *level set* regional em um ponto k de tupla regional $\langle r_k, \phi_k \rangle$. Se $r_k = r_i$, então é subtraído ϵ de ϕ_k e, caso contrário, ϵ é somado ao mesmo. Devido à subtração, o valor de ϕ_k pode se tornar negativo. Nesse caso, atribui-se $phi_k = 0$ para evitar inconsistências na representação da interface, apesar de isso fazer com que ela não tenha a espessura desejada próximo ao ponto k.

Já para a criação das interfaces de uma região fluida com um sólido ou com os limites da grade computacional, emprega-se o método do *marching squares* [65], pois, no caso deste trabalho, estas são sempre superfícies bidimensionais. Nessas situações as interfaces também são deslocadas para dentro de cada região, aos moldes do procedimento descrito anteriormente. Por fim, para toda e qualquer malha criada, é calculado um índice de refração de acordo com os materiais adjacentes à interface e levando em conta o material do filme, como descrito na seção 7.1, e se estabelece sua cor difusa de acordo com aquela do fluido em r_i.

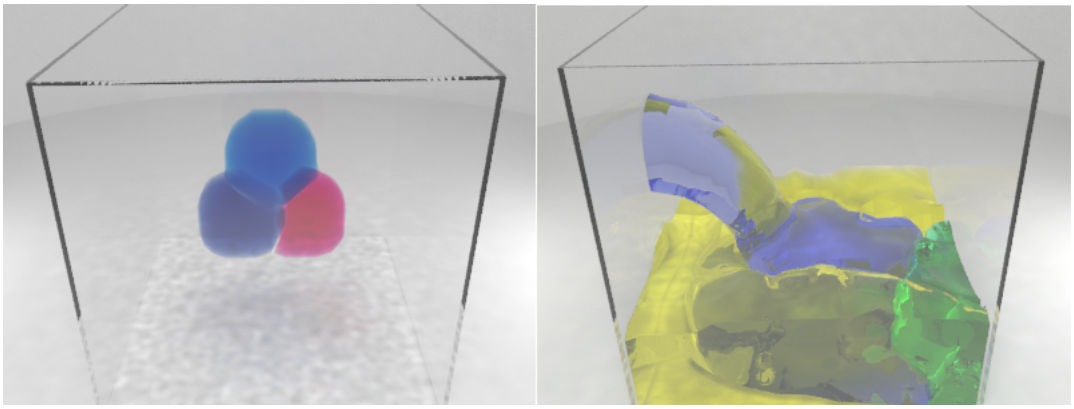

Figura 7.6: Exemplo de duas visualizações finais, onde a primeira retrata três regiões aproximadamente esféricas que se tocam. Já a segunda mostra um fluido azul entrando em um recipiente que já continha um fluido amarelo e um verde.

Para a efetiva geração de um video da animação, inicialmente é guardado um arquivo de texto contendo o número de quadros por segundo que a animação deve ter, o centro e as dimensões da grade. Depois, respeitando a taxa de atualização de quadros definida, a aplicação do simulador exporta, para cada quadro, um modelo no formato '.obj' cujos objetos são as interfaces. Adicionalmente, é salvo um arquivo '.txt' com a cor e o índice de refração de cada item. Para futura distinção, cada arquivo é identificado pelo número do quadro que ele constitui no contexto da animação.

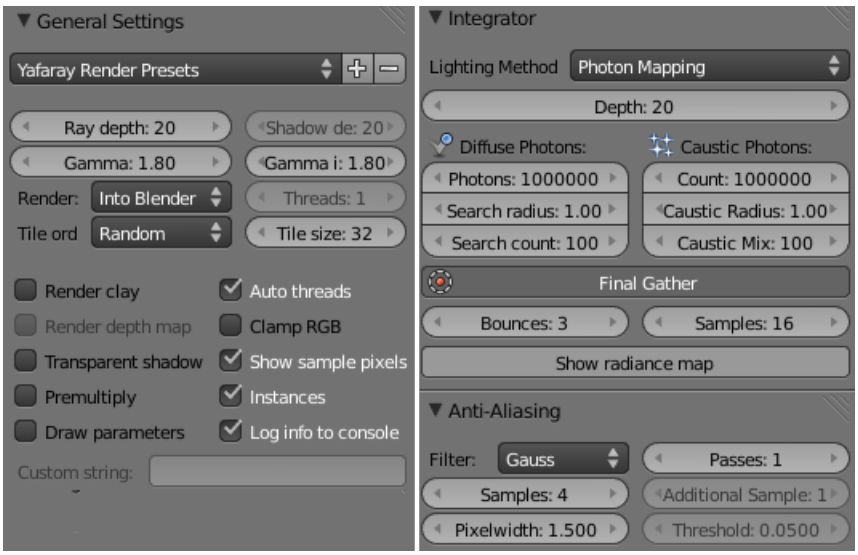

Figura 7.7: Configuração geral do Yafaray pela interface do Blender.

Em seguida, é utilizado um *script* na linguagem Python [66] no Blender [33], onde já haviam sido previamente configurados a iluminação, os materiais padrão, a câmera e os parâmetros do renderizador, para carregar os sucessivos modelos e os renderizar com o Yafaray. Uma possível configuração geral do Yafaray é exibido na

figura 7.7, tendo ela sido empregada para a geração das imagens 7.5 e 7.6. Para fins de comparação, o tempo médio gasto para a renderização de cada quadro com essa configuração do *render* para uma malha gerada a partir de uma grade de dimensão 22^3 é igual a 31 minutos e 53.0 segundos. A escolha da utilização do ambiente do Blender foi realizada justamente pela facilidade com que tais parâmetros podem ser configurados tirando proveito de sua interface gráfica, e pela fácil integração com o Yafaray, que também é bastante conhecido. Por fim, as múltiplas imagens criadas são agrupadas automaticamente em um filme com a taxa de amostragem desejada.

7.1 Índices de Refração

Tendo em vista que, neste trabalho, procura-se simular fluidos que, por vezes, são transparentes, a definição dos índices de refração é imperativa para uma visualização realista. Conforme visto previamente, devido ao *raytracer* escolhido, ao invés de definir índices para cada meio, são definidos valores para cada interface. Assim sendo, é preciso conhecer como eles devem ser definidos. Logo, como essas definições podem gerar certa confusão, neste momento, optou-se por explicar, de forma geral, como calcular tais propriedades.

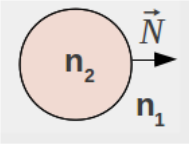

Figura 7.8: Configuração básica para formulação da regra do cálculo dos índices de refração da interface que separa meios com índices n_1 e n_2.

Como regra básica, para uma interface entre dois meios que possuem índices de refração n_1 e n_2 e onde a normal à superfície $\hat{N}$ aponta para o meio de n_1, vide a ilustração 7.8, o valor referente à interface Γ pode ser calculado como:

$$(\text{Índice de Refração})_\Gamma = \frac{n_2}{n_1}. \tag{7.1}$$

Note que o sentido da normal é de fundamental importância para o cálculo do índice de refração. Não obstante, para entender mais profundamente seu papel, cabe entender como o Yafaray utiliza a informação desta constante. Assim, se, no exemplo ilustrado na figura 7.9, um raio atingir o conjunto de regiões pelo lado direito, ele iria encontrar quatro transições. Em cada um desses pontos, o índice de refração utilizado depende do sentido da normal e da direção do raio. Se eles tiverem sentidos opostos, isto é, se o raio atingir a região pelo lado de fora, é utilizado o valor conforme definido pela equação 7.1. Caso contrário, é utilizado o inverso deste

valor.

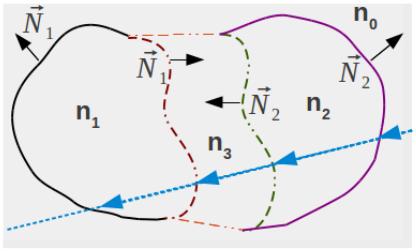

Figura 7.9: Raio atravessando sistema composto por duas regiões envoltas em um material distinto e cujo filme entre elas também possui constituição diferente.

Por fim, também é possível conhecer o índice de refração de cada meio ao se compor as transições de índice às quais um raio é submetido. Por exemplo, considerando todas as transições no trajeto do raio da figura 7.9, obtém-se

$$\left\{\left[\left((n_0)\cdot\frac{n_2}{n_0}\right)\cdot\frac{n_3}{n_2}\right]\cdot\frac{n_1}{n_3}\right\}\cdot\frac{n_0}{n_1}=n_0. \tag{7.2}$$

Isto é, inicialmente o raio se encontra no meio com n_0 passando seguidamente pelos meios de n_2, n_3 e n_1 e, por fim, retornando ao meio de índice n_0.

7.2 *Marching Cubes*

O *marching cubes*, apresentado originalmente por LORENSEN e CLINE [47], é capaz de construir uma malha poligonal à partir de um campo escalar tridimensional, como o campo distância utilizado no *level set*. De fato, esta técnica cria a malha poligonal da superfície de nível que separa a região que possui valores maiores que aquele do nível desejado, daquela que tem valores menores. No caso deste trabalho, sempre é criada a superfície de nível 0.

Logo, para uma célula cúbica, o algoritmo verifica os valores do campo em cada um de seus vértices, os classifica em dentro ou fora da região desejada e determina, a partir de uma tabela com 256 possíveis configurações de polígonos, aquela que representa a parte da superfície dentro do cubo. Tais alternativas são originalmente formadas pela rotação dos 15 casos únicos exibidos na figura 7.10.

Em seguida, tendo definido o aspecto dos triângulos, as posições de seus vértices são definidas em arestas do cubo, precisamente onde a interpolação dos valores escalares de vértices ligados possuir interpolante nulo. Ainda, a normal em cada vértice dos triângulos também podem ser, analogamente, obtidas, visto que elas sempre podem ser estimadas através do gradiente do campo escalar. Assim sendo, como toda a computação depende apenas de informações referentes à cada célula, a execução desta técnica pode ser naturalmente paralelizada.

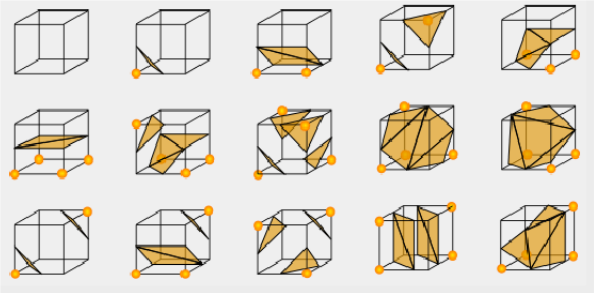

Figura 7.10: Possíveis configurações de triângulos do *marching cubes*. Retirado da WIKIPEDIA [2]

Tendo isso em vista, neste projeto, o *marching cubes* é empregado para construir as malhas de triângulos relativas às superfícies definidas nas interfaces entre cada região do *level set* regional. Por exemplo, para a célula da figura 7.11, onde os vértices superiores e inferiores são de regiões diferentes, o resultado deve gerar uma superfície para cada região, criando um espaço entre elas de distância ϵ referente à espessura do filme. Desse modo, índices de refração e cores podem ser atribuídos a essas superfícies, retratando as transições $fluido - filme - fluido$.

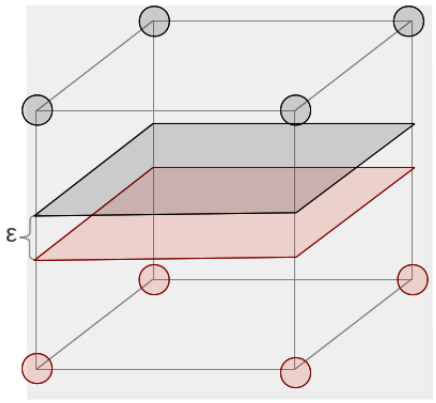

Figura 7.11: Malhas geradas pelo *marching cubes* para uma célula onde os vértices superiores e inferiores estão em regiões distintas.

Portanto, a fim de garantir uma clara divisão regional, para cada par de regiões adjacentes, devem ser criados triângulos em cada célula que possua vértices que indiquem a adjacência entre tais regiões. Além disso, deve-se evitar que o mesmo triângulo seja criado repetidas vezes. Para tanto, apenas se permite que sejam criados triângulos para a região exterior à região em questão que seja, dentre as regiões que possuem um vértice adjacente da região desejada, aquela que possua menor índice. Tendo isso em mente, o pseudocódigo presente na figura 7.12 elucida os passos gerais do algoritmo. Não se deve, contudo, esquecer que os valores das distâncias utilizadas para as interpolações são sempre deslocadas para dentro da região cuja superfície se deseja obter.

Para cada região r_i:
Para cada região r_j adjacente à r_i:
Para cada célula:
Se existir ao menos um vértice v de região r_j que tenha um vértice adjacente em r_i e, além disso, $j \leq k$ para qualquer região $r_k \neq r_i$ de um vértice da célula que tenha pelo menos um vizinho em r_i, **então**:
Classificar cada vértice v da célula em *dentro* ou $fora$:
Se $região(v) = r_i$, **então** v está *dentro*.
Senão v está $fora$
Identificar uma das possíveis configurações de triângulos,
Posicionar cada vértice dos polígonos ao longo da devida aresta da célula no local onde a interpolação linear entre os dois valores que são conectados pela aresta possui interpolante nulo.

Figura 7.12: Pseudocódigo da execução do *marching cubes* para criar as malhas das interfaces inter-regionais.

7.3 *Marching Squares*

O *marching squares* [65] é um método bidimensional análogo ao algoritmo do *marching cubes*. Isto é, através dele também é extraída uma malha poligonal de um campo escalar. Contudo, o resultado é essencialmente bidimensional, de modo que apenas é necessário trabalhar com vértices dispostos em um quadrado e não em um cubo. Além disso, ao contrário do seu parente tridimensional, constitui um método direto, não necessitando de uma tabela para a verificação da configuração dos triângulos a serem criados.

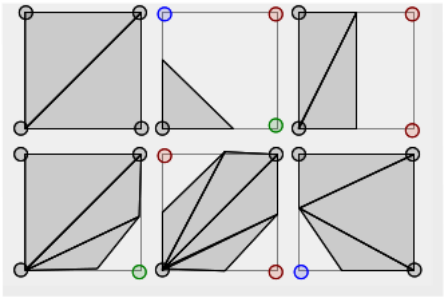

Figura 7.13: Exemplos de triangulação do algoritmo do *marching squares*

A simplicidade do algoritmo permite que se possa visitar os vértices em uma ordem pré-determinada para obter os sucessivos vértices dos triângulos. Durante este trajeto, um vértice de triangulo é criado em um vértice do quadrado se ele for da região desejada, ou em uma aresta se, entre os vértice ligados, um for da região em questão e o outro de uma região diferente. Neste caso, a posição do novo vértice é dada pelo ponto onde o interpolante dos valores do campo distância dos vértices

ligados é nulo.

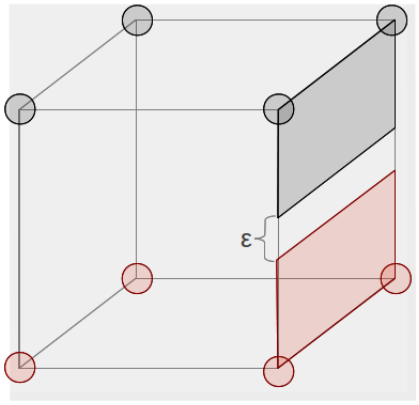

Figura 7.14: Exemplo do produto da aplicação do algoritmo do *marching squares* para uma célula, onde os vértices superiores e inferiores são de regiões diferentes, e a célula à direita contém material sólido ou simplesmente não existe.

Logo, os primeiros três vértices descobertos formam um triângulo. Em seguida, cada novo vértice constituirá outro triângulo em conjunto com os vértices que formaram a última aresta do triangulo recém definido, resultando em uma estrutura em leque. Exemplos de estruturas que podem ser reproduzidas são exibidas na figura 7.13.

Para cada região r_i:
  Para cada célula ao lado de uma célula sólida ou nos limites da grade:
    Para cada face em fronteiras fluido-sólido ou nos limites da grade:
      Percorrer vértices pelo sentido horário partindo daquele com as menores coordenadas (x, y, z).
      Criar vértices de triângulo em vértices da face que sejam de r_i ou em arestas onde, entre os vértices que ela liga, um é de r_i e outro é de uma região r_j qualquer. $(i \neq j)$
      Formar um triangulo a cada 3 vértices de triângulo definidos, guardando aqueles da última aresta criada para o triangulo seguinte.

Figura 7.15: Pseudocódigo da execução do *marching squares* para criar as malhas das interfaces inter-regionais em fronteiras fluido-sólido ou nos limites da grade.

Portanto, neste trabalho o *marching squares* é empregado para criar superfícies em faces de células situadas em locais específicos. A figura 7.14 ilustra o resultado que se deseja obter. De fato, são criados polígonos que separam o interior da região fluida de um exterior sólido ou fora da grade computacional, mas ainda separando fielmente as diferentes regiões. Inclusive, é mantida a espessura ϵ do filme, uma vez que ainda é feito o mesmo deslocamento da interface para dentro de cada região que é realizado para o *marching cubes*. Por fim, exibe-se, na figura 7.15, um pseudocódigo que ilustra o funcionamento do método.

Capítulo 8

Conclusões

Em linhas gerais, pode-se dizer que, neste trabalho, foi implementada a simulação multifásica de fluidos incompressíveis empregando o *level set* regional para representar implicitamente diferentes fases (ou regiões) com precisão sub-grade e custo independente da multiplicidade dos meios envolvidos. Através desta técnica, foi possível acompanhar o deslocamento e as mudanças topológicas das diferentes fases e do filme presente entre meios distintos. Nesse sentido, trataram-se mudanças relativas à junção de duas regiões ou à subdivisão de uma fase. Ainda, fez-se possível que fases adjacentes do mesmo material não fossem necessariamente fundidas, possibilitando, por exemplo, o acumulo local de bolhas de ar.

Contudo, apesar da precisão no tratamento do filme, apenas foi tratado o caso em que ele é composto de um único material enquanto, teoricamente, tal filme poderia ser o produto da composição de múltiplos fluidos. Além disso, o filme implícito foi considerado como necessariamente uma interface entre duas regiões. Assim, mesmo que ele possa ter dimensões sub-grade, não é possível retratar características das regiões com a mesma precisão. À exemplo, uma região que se torne extremamente fina irá desaparecer da grade.

Adicionalmente, partículas foram empregadas para conferir aparência mais realista à simulação e auxiliar a conservação do volume. Ao certo, converteram-se regiões muito pequenas para partículas do mesmo volume, evitando que as primeiras fossem removidas em decorrência da advecção. Enquanto representadas por partículas, fases foram movimentadas pelo efeito da gravidade e do arraste gerado pela massa fluida. Ademais, seus raios foram ajustados para que elas retomassem o volume original de suas respectivas regiões. Assim, se, por exemplo, uma fase se tornasse muito pequena por erros numéricos, ela se tornaria uma partícula que aumentaria até possuir o volume original. Por fim, partículas foram convertidas de volta para uma região ao atingirem um volume máximo ou se fundirem a uma fase do mesmo fluido, tendo os valores do *level set* regional em posições dentro ou próximas da partícula atualizados para representarem a nova região formada.

Já no requisito da simulação propriamente dita, o âmbito multifásico requereu o uso de um ferramental adicional àquele comumente usado no tratamento de uma única fase. De fato, o sistema das equações de Poisson passou a considerar valores de densidade variáveis usando técnicas descritas por LIU *et al.* [29]. Além disso, foi empregado o método *ghost fluid* (HONG e KIM [30]) para levar em consideração a atuação das forças de tensão superficial nas interfaces entre dois meios. Vale lembrar que cada um desses métodos introduziu apenas pequenas alterações no sistema de equações original, de modo que ainda se pôde utilizar o método tradicional de resolução; no caso deste trabalho, o gradiente conjugado (HESTENES e STIEFEL [42]).

Para evitar que o volume de cada fase variasse de modo não desejado devido a erros numéricos, foi desenvolvido um novo método para o controle do volume das fases. Por esta técnica, um campo de velocidades auxiliar com valores não nulos próximo às interfaces foi utilizado para realizar passos adicionais de advecção do *level set* regional. Ainda, diferentemente dos outros métodos estudados, o processo apresentado corrige o volume preservando o sentido de deslocamento do fluido e não apenas fazendo com que as fases inflem ou murchem. Ademais, permite que modificações no volume das regiões sejam realizadas de modo controlado.

No que se refere à visualização, enquanto a geração da malha de triângulos é realizada pela própria aplicação, foi implementada uma solução genérica para a renderização final de fluidos utilizando ferramentas amplamente conhecidas como o Blender [33], o Yafaray [34] e o padrão '.obj'. Ao certo, o processo de renderização pode ser facilmente adaptado para outras aplicações que apenas precisam fornecer as entradas necessárias. Em suma, precisa-se, pra cada frame, um modelo '.obj' com um objeto para cada interface, um arquivo '.txt' com as cores e índices de refração de cada objeto, além de um arquivo de texto com parâmetros diversos como o número de quadros por segundo.

Neste trabalho, todo o processo de simulação e renderização foi executado em um computador com as seguintes especificações: processador Intel® Core™ i7-2630QM 2.00GHz, memória de 8090MB, placa de video GeForce GT 540M, rodando o sistema operacional Ubuntu 12.04.1 LTS. Assim sendo, usando o caso retratado pela imagem mais à direita da figura 7.6, onde foi empregada uma grade de 22^3 células, o tempo médio da execução de cada iteração foi de 1.07433 segundos. Já para a simulação de um quadro de uma animação exibida na taxa de 30 quadros por segundo, despendeu-se 7.15506 segundos. Vale lembrar que é criado um quadro sempre que a soma do tempo simulado de iterações sucessivas for equivalente a duração de um quadro; nesse caso 1/30 segundo. Finalmente, para a renderização final, gastou-se, em média, 31 minutos e 53.0 segundos por quadro.

Como contribuição propriamente dita deste trabalho para o meio acadêmico,

pode-se citar, além da simples integração de diferentes técnicas já existentes para o atingimento dos objetivos definidos na seção 1.4, alguns outros fatores. Em primeiro lugar, destaca-se o método das correções egoístas empregado para tratar o erro do volume de cada fase. Em conjunto com esta técnica, foi elaborada uma forma de atualizar os volumes alvo de cada fase de acordo com os fluxos de entrada e saída de fluido na grade da simulação. Adicionalmente, foi definido como corrigir o volume de partículas com base no volume alvo de suas respectivas regiões de origem.

Além disso, foi estabelecido como obter a espessura fictícia da região resultante da fusão entre duas regiões distintas, e se desenvolveu o algoritmo de inundação (ou *flood fill*) para a detecção de regiões que realiza, inclusive, fusões entre fases adjacentes. Por fim, também se pode indicar como contribuição o procedimento de geração das malhas de triângulos através da iteração das interfaces presentes, gerando as devidas superfícies com seus respectivos parâmetros para o *render*.

Capítulo 9

Trabalhos Futuros

Neste trabalho, restringiu-se o filme a ser composto de apenas um material e a sempre existir entre duas regiões distintas. Assim, como uma sequência natural ao trabalho desenvolvido, sugere-se o emprego de filmes de diferentes materiais que podem, inclusive, serem formados pela composição de fluidos distintos, assim como a consideração da possibilidade de existência ou não de filmes entre regiões conexas.

Em termos da robustez do método de controle de volume proposto, acredita-se que sua implementação em conjunto com outras técnicas tais como o *Particle Level Set* (PLS) (ENRIGHT *et al.* [16]) e o *Back and Forth Error Compensation and Correction*(BFECC) (DUPONT e LIU [56], KIM *et al.* [57]) o tornariam mais eficaz. Além disso, no caso do PLS, este poderia ser utilizado também para a geração de gotículas a partir de partículas escapadas do seu meio, incrementando o realismo da simulação.

Outra ideia que também poderia ser concretizada é a correção de volume através de um modelo global que considere o erro volumétrico de cada região. Para tanto, sugere-se a resolução de um problema de fluxo em redes em que se buscaria descobrir a quantidade de volume a ser transferida entre cada par de regiões adjacentes através da interface que as separa, de modo a minimizar o erro volumétrico total do sistema. Assim, uma vez calculado esse fluxo por interface, a definição de cada componente de velocidade auxiliar seria feita com base em uma interface específica e não mais na superfície de uma região inteira.

Além disso, a metodologia empregada para o cômputo dos volumes não garante que a soma dos volumes das regiões que intersectam uma célula reproduza o próprio volume desta. Isso, por si só, já determina a necessidade de correção do volume, em especial no contexto de grades de baixa resolução. Para minimizar esse problema, a resolução precisaria ser mais alta. Tendo isso em mente, uma fórmula para o cômputo do volume das interseções entre fases e células que assegure que a soma desses valores seja igual ao volume das células, se vinculada ao modo como o *level set* regional reparte a célula, seria, sem dúvida, de utilidade.

Neste trabalho utilizou-se a linguagem CUDA [31] apenas como tentativa de acelerar minimamente a execução da simulação ao aplicá-lo a situações naturalmente paralelas, como a advecção semi-lagrangiana, ou adaptando algoritmos conforme necessário. Desse modo, não foram aplicados métodos de otimização no contexto paralelo específico da programação CUDA como, por exemplo, a melhor alocação que pouparia recursos de memória e reduziria o número de acessos à ela. Assim, deixa-se como trabalho futuro a análise e melhoria das técnicas abordadas tendo em vista o ambiente CUDA.

Por fim, propõe-se a implementação de métodos numéricos mais robustos para resolver a equação de Poisson, a exemplo do Gradient Conjugado Precondicionado. Além disso, deixa-se indicada a possibilidade de passar a considerar a propriedade da viscosidade durante a simulação numérica euleriana para a efetiva simulação de fluidos viscosos.

Referências Bibliográficas

[1] KIM, B. "Multi-phase fluid simulations using regional level sets", *ACM Trans. Graph.*, v. 29, n. 6, pp. 175:1–175:8, dez. 2010. ISSN: 0730-0301. doi: 10.1145/1882261.1866197. Disponível em: <http://doi.acm.org/10.1145/1882261.1866197>.

[2] WIKIPEDIA. "Marching cubes - Wikipedia, the free encyclopedia". 2012. Disponível em: <http://en.wikipedia.org/wiki/Marching_cubes>. [Online; acessado 13-Novembro-2012].

[3] MÜLLER, M., CHARYPAR, D., GROSS, M. "Particle-based fluid simulation for interactive applications". In: *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '03, pp. 154–159, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association. ISBN: 1-58113-659-5. Disponível em: <http://dl.acm.org/citation.cfm?id=846276.846298>.

[4] SOLENTHALER, B., PAJAROLA, R. "Predictive-corrective incompressible SPH". In: *ACM SIGGRAPH 2009 papers*, SIGGRAPH '09, pp. 40:1–40:6, New York, NY, USA, 2009. ACM. ISBN: 978-1-60558-726-4. doi: 10.1145/1576246.1531346. Disponível em: <http://doi.acm.org/10.1145/1576246.1531346>.

[5] SIN, F., BARGTEIL, A. W., HODGINS, J. K. "A point-based method for animating incompressible flow". In: *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '09, pp. 247–255, New York, NY, USA, 2009. ACM. ISBN: 978-1-60558-610-6. doi: 10.1145/1599470.1599502. Disponível em: <http://doi.acm.org/10.1145/1599470.1599502>.

[6] CLEARY, P. W., PYO, S. H., PRAKASH, M., et al. "Bubbling and frothing liquids". In: *ACM SIGGRAPH 2007 papers*, SIGGRAPH '07, New York, NY, USA, 2007. ACM. doi: 10.1145/1275808.1276499. Disponível em: <http://doi.acm.org/10.1145/1275808.1276499>.

[7] DARLES, E., C. B., GHAZANFARPOUR, D. "Accelerating and enhancing rendering of realistic ocean scenes". In: *Proceedings of WSCG'2007*, WSCG'2007, 2007.

[8] KÜCK, H., VOGELGSANG, C., GREINER, G. "Simulation and Rendering of Liquid Foams". In: *In Proc. Graphics Interface '02 (2002*, pp. 81–88, 2002.

[9] FOSTER, N., METAXAS, D. "Realistic animation of liquids". In: *Proceedings of the conference on Graphics interface '96*, GI '96, pp. 204–212, Toronto, Ont., Canada, Canada, 1996. Canadian Information Processing Society. ISBN: 0-9695338-5-3. Disponível em: <http://dl.acm.org/citation.cfm?id=241020.241077>.

[10] STAM, J. "Stable fluids". In: *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '99, pp. 121–128, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co. ISBN: 0-201-48560-5. doi: 10.1145/311535.311548. Disponível em: <http://dx.doi.org/10.1145/311535.311548>.

[11] FOSTER, N., FEDKIW, R. "Practical animation of liquids". In: *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pp. 23–30, New York, NY, USA, 2001. ACM. ISBN: 1-58113-374-X. doi: 10.1145/383259.383261. Disponível em: <http://doi.acm.org/10.1145/383259.383261>.

[12] MULLEN, P., CRANE, K., PAVLOV, D., et al. "Energy-preserving integrators for fluid animation", *ACM Trans. Graph.*, v. 28, n. 3, pp. 38:1–38:8, jul. 2009. ISSN: 0730-0301. doi: 10.1145/1531326.1531344. Disponível em: <http://doi.acm.org/10.1145/1531326.1531344>.

[13] KIM, D., SONG, O.-Y., KO, H.-S. "Stretching and wiggling liquids". In: *ACM SIGGRAPH Asia 2009 papers*, SIGGRAPH Asia '09, pp. 120:1–120:7, New York, NY, USA, 2009. ACM. ISBN: 978-1-60558-858-2. doi: 10.1145/1661412.1618466. Disponível em: <http://doi.acm.org/10.1145/1661412.1618466>.

[14] WOJTAN, C., THÜREY, N., GROSS, M., et al. "Physics-inspired topology changes for thin fluid features". In: *ACM SIGGRAPH 2010 papers*, SIGGRAPH '10, pp. 50:1–50:8, New York, NY, USA, 2010. ACM. ISBN: 978-1-4503-0210-4. doi: 10.1145/1833349.1778787. Disponível em: <http://doi.acm.org/10.1145/1833349.1778787>.

[15] OSHER, S., FEDKIW, R. *Level set methods and dynamic implicit surfaces*. Springer Verlag, 2003.

[16] ENRIGHT, D., MARSCHNER, S., FEDKIW, R. "Animation and rendering of complex water surfaces". In: *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '02, pp. 736–744, New York, NY, USA, 2002. ACM. ISBN: 1-58113-521-1. doi: 10.1145/566570.566645. Disponível em: <http://doi.acm.org/10.1145/566570.566645>.

[17] LOSASSO, F., TALTON, J., KWATRA, N., et al. "Two-Way Coupled SPH and Particle Level Set Fluid Simulation", *IEEE Transactions on Visualization and Computer Graphics*, v. 14, n. 4, pp. 797–804, jul. 2008. ISSN: 1077-2626. doi: 10.1109/TVCG.2008.37. Disponível em: <http://dx.doi.org/10.1109/TVCG.2008.37>.

[18] HONG, J.-M., LEE, H.-Y., YOON, J.-C., et al. "Bubbles alive". In: *ACM SIGGRAPH 2008 papers*, SIGGRAPH '08, pp. 48:1–48:4, New York, NY, USA, 2008. ACM. ISBN: 978-1-4503-0112-1. doi: 10.1145/1399504.1360647. Disponível em: <http://doi.acm.org/10.1145/1399504.1360647>.

[19] GREENWOOD, S. T., HOUSE, D. H. "Better with bubbles: enhancing the visual realism of simulated fluid". In: *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '04, pp. 287–296, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association. ISBN: 3-905673-14-2. doi: 10.1145/1028523.1028562. Disponível em: <http://dx.doi.org/10.1145/1028523.1028562>.

[20] GUENDELMAN, E., SELLE, A., LOSASSO, F., et al. "Coupling water and smoke to thin deformable and rigid shells", *ACM Trans. Graph.*, v. 24, n. 3, pp. 973–981, jul. 2005. ISSN: 0730-0301. doi: 10.1145/1073204.1073299. Disponível em: <http://doi.acm.org/10.1145/1073204.1073299>.

[21] MIHALEF, V., METAXAS, D. N., SUSSMAN, M. "Simulation of two-phase flow with sub-scale droplet and bubble effects", *Comput. Graph. Forum*, v. 28, n. 2, pp. 229–238, 2009. doi: 10.1111/j.1467-8659.2009.01362.x. Disponível em: <http://dblp.uni-trier.de/db/journals/cgf/cgf28.html#MihalefMS09>.

[22] MIHALEF, V. *The marker level set method: applications to simulation of liquids*. Tese de Doutorado, New Brunswick, NJ, USA, 2007. AAI3319673.

[23] LOSASSO, F., SHINAR, T., SELLE, A., et al. "Multiple interacting liquids". In: *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, pp. 812–819, New York, NY, USA, 2006. ACM. ISBN: 1-59593-364-6. doi: 10.1145/1179352.1141960. Disponível em: <http://doi.acm.org/10.1145/1179352.1141960>.

[24] VESE, L. A., CHAN, T. F. "A Multiphase Level Set Framework for Image Segmentation Using the Mumford and Shah Model", *Int. J. Comput. Vision*, v. 50, n. 3, pp. 271–293, dez. 2002. ISSN: 0920-5691. doi: 10.1023/A:1020874308076. Disponível em: <http://dx.doi.org/10.1023/A:1020874308076>.

[25] WOJTAN, C., THÜREY, N., GROSS, M., et al. "Deforming meshes that split and merge", *ACM Trans. Graph.*, v. 28, n. 3, pp. 76:1–76:10, jul. 2009. ISSN: 0730-0301. doi: 10.1145/1531326.1531382. Disponível em: <http://doi.acm.org/10.1145/1531326.1531382>.

[26] BROCHU, T., BATTY, C., BRIDSON, R. "Matching fluid simulation elements to surface geometry and topology", *ACM Trans. Graph.*, v. 29, n. 4, pp. 47:1–47:9, jul. 2010. ISSN: 0730-0301. doi: 10.1145/1778765.1778784. Disponível em: <http://doi.acm.org/10.1145/1778765.1778784>.

[27] ZHENG, W., YONG, J.-H., PAUL, J.-C. "Simulation of bubbles", *Graph. Models*, v. 71, n. 6, pp. 229–239, nov. 2009. ISSN: 1524-0703. doi: 10.1016/j.gmod.2009.08.001. Disponível em: <http://dx.doi.org/10.1016/j.gmod.2009.08.001>.

[28] KANG, M., FEDKIW, R. P., LIU, X.-D. "A Boundary Condition Capturing Method for Multiphase Incompressible Flow", *J. Sci. Comput.*, v. 15, n. 3, pp. 323–360, set. 2000. ISSN: 0885-7474. doi: 10.1023/A:1011178417620. Disponível em: <http://dx.doi.org/10.1023/A:1011178417620>.

[29] LIU, X.-D., FEDKIW, R. P., KANG, M. "A boundary condition capturing method for Poisson's equation on irregular domains", *J. Comput. Phys.*, v. 160, n. 1, pp. 151–178, maio 2000. ISSN: 0021-9991. doi: 10.1006/jcph.2000.6444. Disponível em: <http://dx.doi.org/10.1006/jcph.2000.6444>.

[30] HONG, J.-M., KIM, C.-H. "Discontinuous fluids", *ACM Trans. Graph.*, v. 24, n. 3, pp. 915–920, jul. 2005. ISSN: 0730-0301. doi: 10.1145/1073204.1073283. Disponível em: <http://doi.acm.org/10.1145/1073204.1073283>.

[31] NVIDIA. "Parallel Programming and Computing Platform — CUDA — NVIDIA". 2012. Disponível em: <http://www.nvidia.com/object/cuda_home_new.html>. [Online; acessado 13-Novembro-2012].

[32] OPENGL. "OpenGL - The Industry Standard for High Performance Graphics". 2012. Disponível em: <http://www.opengl.org/>. [Online; acessado 13-Novembro-2012].

[33] BLENDER. "Blender.org – Home". 2012. Disponível em: <http://www.blender.org/>. [Online; acessado 13-Novembro-2012].

[34] YAFARAY. "Home — Yafaray". 2012. Disponível em: <http://www.yafaray.org/>. [Online; acessado 13-Novembro-2012].

[35] BRIDSON, R. *Fluid Simulation for Computer Graphics*. A K Peters/CRC Press, set. 2008. ISBN: 1568813260. Disponível em: <http://www.worldcat.org/isbn/1568813260>.

[36] HARLOW, F. H., WELCH, J. E. "Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface", *Physics of Fluids*, v. 8, n. 12, pp. 2182–2189, 1965. doi: 10.1063/1.1761178. Disponível em: <http://dx.doi.org/10.1063/1.1761178>.

[37] PENG, D., MERRIMAN, B., OSHER, S., et al. "A PDE-Based Fast Local Level Set Method", *Journal of Computational Physics*, v. 155, 1999.

[38] ADALSTEINSSON, D., SETHIAN, J. A. "The Fast Construction of Extension Velocities in Level Set Methods", *Journal of Computational Physics*, v. 148, pp. 2–22, 1997.

[39] SETHIAN, J. A. "A Fast Marching Level Set Method for Monotonically Advancing Fronts", *Proceedings of the National Academy of Sciences of the United States of America*, v. 93, n. 4, pp. 1591–1595, 1996.

[40] BATTY, C., BERTAILS, F., BRIDSON, R. "A fast variational framework for accurate solid-fluid coupling", *ACM Trans. Graph.*, v. 26, n. 3, pp. 100, 2007.

[41] BATTY, C., XENOS, S., HOUSTON, B. "Tetrahedral Embedded Boundary Methods for Accurate and Flexible Adaptive Fluids". In: *Proceedings of Eurographics*, 2010.

[42] HESTENES, M. R., STIEFEL, E. "Methods of conjugate gradients for solving linear systems", *Journal of research of the National Bureau of Standards*, v. 49, pp. 409–436, 1952.

[43] HEATH, M. T. *Scientific Computing: An Introductory Survey*. McGraw-Hill Higher Education, 1996. ISBN: 0070276846.

[44] COURANT, R., FRIEDRICHS, K., LEWY, H. "Über die partiellen differenzengleichungen der mathematischen physik", *Mathematische Annalen*, v. 100, n. 1, pp. 32–74, dez. 1928. ISSN: 0025-5831. doi: 10.1007/BF01448839.

[45] OSHER, S., SETHIAN, J. A. "Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations", *J. Comput. Phys.*, v. 79, n. 1, pp. 12–49, nov. 1988. ISSN: 0021-9991. doi: 10.1016/0021-9991(88)90002-2. Disponível em: <http://dx.doi.org/10.1016/0021-9991(88)90002-2>.

[46] OSHER, S., PARAGIOS, N. *Geometric Level Set Methods in Imaging,Vision,and Graphics*. Secaucus, NJ, USA, Springer-Verlag New York, Inc., 2003. ISBN: 0387954880.

[47] LORENSEN, W. E., CLINE, H. E. "Seminal graphics". ACM, cap. Marching cubes: a high resolution 3D surface construction algorithm, pp. 347–353, New York, NY, USA, 1998. ISBN: 1-58113-052-X. doi: 10.1145/280811.281026. Disponível em: <http://doi.acm.org/10.1145/280811.281026>.

[48] HIEBER, S. E., KOUMOUTSAKOS, P. "A Lagrangian particle level set method", *J. Comput. Phys.*, v. 210, n. 1, pp. 342–367, nov. 2005. ISSN: 0021-9991. doi: 10.1016/j.jcp.2005.04.013. Disponível em: <http://dx.doi.org/10.1016/j.jcp.2005.04.013>.

[49] MIHALEF, V. *The marker level set method: applications to simulation of liquids*. Tese de Doutorado, New Brunswick, NJ, USA, 2007. AAI3319673.

[50] SUSSMAN, M., PUCKETT, E. G. "A coupled level set and volume-of-fluid method for computing 3D and axisymmetric incompressible two-phase flows", *J. Comp. Phys.*, v. 162, n. 2, ago. 2000.

[51] KIM, B., LIU, Y., LLAMAS, I., et al. "Simulation of bubbles in foam with the volume control method", *ACM Trans. Graph.*, v. 26, n. 3, jul. 2007. ISSN: 0730-0301. doi: 10.1145/1276377.1276500. Disponível em: <http://doi.acm.org/10.1145/1276377.1276500>.

[52] ZWILLINGER, D. "CRC Standard Mathematical Tables and Formulae". 32nd ed., p. 642, Taylor & Francis Group, 2011. ISBN: 9781439835487.

[53] BLOOMENTHAL, J., FERGUSON, K. "Polygonization of non-manifold implicit surfaces". In: *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '95, pp. 309–316, New York, NY, USA, 1995. ACM. ISBN: 0-89791-701-4. doi: 10.1145/218380.218462. Disponível em: <http://doi.acm.org/10.1145/218380.218462>.

[54] YAMAZAKI, S., KASE, K., IKEUCHI, K. "Non-Manifold Implicit Surfaces Based on Discontinuous Implicitization and Polygonization". In: *Proceedings of the Geometric Modeling and Processing — Theory and Applications (GMP'02)*, GMP '02, pp. 138–, Washington, DC, USA, 2002. IEEE Computer Society. ISBN: 0-7695-1674-2. Disponível em: <http://dl.acm.org/citation.cfm?id=795681.797476>.

[55] BUSARYEV, O., DEY, T. K., WANG, H., et al. "Animating bubble interactions in a liquid foam", *ACM Trans. Graph.*, v. 31, n. 4, pp. 63:1–63:8, jul. 2012. ISSN: 0730-0301. doi: 10.1145/2185520.2185559. Disponível em: <http://doi.acm.org/10.1145/2185520.2185559>.

[56] DUPONT, T. F., LIU, Y. "Back and forth error compensation and correction methods for removing errors induced by uneven gradients of the level set function", *J. Comput. Phys.*, v. 190, n. 1, pp. 311–324, set. 2003. ISSN: 0021-9991. doi: 10.1016/S0021-9991(03)00276-6. Disponível em: <http://dx.doi.org/10.1016/S0021-9991(03)00276-6>.

[57] KIM, B., LIU, Y., LLAMAS, I., et al. "FlowFixer: using BFECC for fluid simulation". In: *Proceedings of the First Eurographics conference on Natural Phenomena*, NPH'05, pp. 51–56, Aire-la-Ville, Switzerland, Switzerland, 2005. Eurographics Association. ISBN: 3-905673-29-0. doi: 10.2312/NPH/NPH05/051-056. Disponível em: <http://dx.doi.org/10.2312/NPH/NPH05/051-056>.

[58] SELLE, A., FEDKIW, R., KIM, B., et al. "An Unconditionally Stable MacCormack Method", *J. Sci. Comput.*, v. 35, n. 2-3, pp. 350–371, jun. 2008. ISSN: 0885-7474. doi: 10.1007/s10915-007-9166-4. Disponível em: <http://dx.doi.org/10.1007/s10915-007-9166-4>.

[59] SONG, O.-Y., SHIN, H., KO, H.-S. "Stable but nondissipative water", *ACM Trans. Graph.*, v. 24, n. 1, pp. 81–97, jan. 2005. ISSN: 0730-0301. doi: 10.1145/1037957.1037962. Disponível em: <http://doi.acm.org/10.1145/1037957.1037962>.

[60] TAKAHASHI, T., FUJII, H., KUNIMATSU, A., et al. "Realistic Animation of Fluid with Splash and Foam", *Computer Graphics Forum*, v. 22, n. 3, pp. 391–400, 2003. doi: 10.1111/1467-8659.00686. Disponível em: <http://dx.doi.org/10.1111/1467-8659.00686>.

[61] GOKTEKIN, T. G., BARGTEIL, A. W., O'BRIEN, J. F. "A method for animating viscoelastic fluids". In: *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, pp. 463–468, New York, NY, USA, 2004. ACM. doi: 10.1145/1186562.1015746. Disponível em: <http://doi.acm.org/10.1145/1186562.1015746>.

[62] SUSSMAN, M. "A second order coupled level set and volume-of-fluid method for computing growth and collapse of vapor bubbles", *J. Comput. Phys.*, v. 187, n. 1, pp. 110–136, maio 2003. ISSN: 0021-9991. doi: 10.1016/S0021-9991(03)00087-1. Disponível em: <http://dx.doi.org/10.1016/S0021-9991(03)00087-1>.

[63] FELDMAN, B. E., O'BRIEN, J. F., ARIKAN, O. "Animating suspended particle explosions", *ACM Trans. Graph.*, v. 22, n. 3, pp. 708–715, jul. 2003. ISSN: 0730-0301. doi: 10.1145/882262.882336. Disponível em: <http://doi.acm.org/10.1145/882262.882336>.

[64] LOSASSO, F., IRVING, G., GUENDELMAN, E., et al. "Melting and Burning Solids into Liquids and Gases", *IEEE Transactions on Visualization and Computer Graphics*, v. 12, n. 3, pp. 343–352, maio 2006. ISSN: 1077-2626. doi: 10.1109/TVCG.2006.51. Disponível em: <http://dx.doi.org/10.1109/TVCG.2006.51>.

[65] WIKIPEDIA. "Marching squares - Wikipedia, the free encyclopedia". 2012. Disponível em: <http://en.wikipedia.org/wiki/Marching_squares>. [Online; acessado 13-Novembro-2012].

[66] PYTHON. "Python Programming Language - Official Website". 2012. Disponível em: <http://www.python.org/>. [Online; acessado 27-Dezembro-2012].