Animação computacional de escoamento de fluidos utilizando o método SPH

Tiago Etiene Queiroz

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

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

USP – São Carlos
Junho/2008

Animação computacional de escoamento de fluidos utilizando o método SPH

Tiago Etiene Queiroz

À minha querida e amada família.

Agradecimentos

Primeiramente, agradeço ao Deus inefável, o *nunca bastante*.

À minha família chamada LCAD, local onde, durante anos, longas horas dos meus dias foram investidas. Em particular, aos professores Luis Gustavo Nonato, Antonio Castelo Filho e José Alberto Cuminato, os amigos que me apresentaram o mundo da pesquisa.

Agradeço aos professores Fabrício Simeoni de Souza e Marcelo Siqueira pelas valiosas dicas durante minha qualificação do mestrado.

Um agradecimento especial aos meus amigos João Paulo Gois e Valdecir Polizelli Junior, pessoas de competência e habilidade ímpar. Eles me ensinaram muito durante agradáveis noites de trabalho nas vésperas do SIBGRAPI.

Aos meus amigos, pelo forte apoio, em especial Guilherme Ulliana e Caio Carélo que acompanharam de perto o fim de uma fase em minha vida trazendo sempre alegria em suas palavras.

Agradeço à FAPESP pelo apoio financeiro.

Finalmente, agradeço à minha esposa, Karina, que, com carinho e paciência, tem acompanhado minha caminhada pela vida acadêmica.

Resumo

Desde a década de 70, há um crescente interesse em simulações em computador de fenômenos físicos visto sua diversidade de aplicações. Dentre esses fenômenos, podem ser destacados a interação entre corpos rígidos, elásticos, plásticos, quebráveis e também fluidos. Neste trabalho realizamos a simulação de um desses fenômenos, o escoamento de fluidos, por um método conhecido como *Smoothed Particles Hydrodynamics*, uma abordagem lagrangeana baseada em partículas para resolução das equações que modelam o movimento do fluido. Várias são as vantagens de métodos lagrangeanos usando partículas sobre os que usam malhas, por exemplo, as propriedades do material transladam com as partículas como função do tempo, além da capacidade de lidar com grandes deformações. Dentre as desvantagem, destacamos uma deficiência relacionada ao ganho de energia total do sistema e estabilidade das partículas. Para lidar com isso, utilizamos uma abordagem baseada na lei da conservação da energia: em um sistema isolado a energia total se mantém constante e ela não pode ser criada ou destruida. Dessa forma, alterando o integrador temporal nós restringimos o aumento arbitrário de energia, tornando a simulação mais tolerante às condições iniciais.

Abstract

Since the late 70's, there is a growing interest in physically-based simulations due to its increasing range of application. Among these simulations, we may highlight interaction between rigid, elastic, plastic and breakable bodies and also fluids. In this work, one of these phenomena, fluid flow, is simulated using a technique known as Smoothed Particle Hydrodynamics, a meshless lagrangean method that solves the equations of the flow behavior of fluids. There are several advantages of meshless methods over mesh-based methods, for instance, the material properties are translated along with particles as a function of time and the ability to handle arbitrary deformations. Among the disadvantages, we may highlight a problem related to the gain of energy by the system and stability issues. In order to handle this, we used an approach based on the law of conservation of energy: in an isolated system the total energy remains constant and cannot be created or destroyed. Based on this, we used a technique that bounds the total energy and the simulation becomes less sensitive to initial conditions.

Sumário

Lista de Figuras

Siglas

CC	Condições de Contorno
CFD	*Computational Fluid Dynamics*
CI	Condições Iniciais
DFC	Dinâmica de Fluidos Computacional
EDO	Equação Diferencial Ordinária
EDP	Equação Diferencial Parcial
MDF	Método das Diferenças Finitas
MEF	Método dos Elementos Finitos
MM	*Meshfree Methods*
MPU	*Multi-level Partition of Unity Implicits*
MVF	Método dos Volumes Finitos
RSPH	*Regularized Smoothed Particles Hydrodynamics*
SPH	*Smoothed Particles Hydrodynamics*

1 Introdução

Animação de fenômenos naturais por computador tem chamado a atenção de muitos pesquisadores por aproximadamente três décadas. Fenômenos naturais são vastos começando com a simulação dos fótons que incidem em uma superfície translúcida e de aspecto rugoso até a simulação de grandes detonações e mesmo explosões nucleares. Podemos corretamente supor que os investimentos recebidos nessa área, tanto de cunho intelectual quanto financeiro, trouxeram notável progresso à qualidade das imagens vistas no dias atuais. Para a felicidade de muitos pesquisadores, não são raras as vezes em que somos enganados pela própria "natureza artificial" que criamos. Para ilustrar esse comentário, nada mais adequado que o sítio eletrônico *Fake or Photo?*[1], lugar onde somos desafiados a perceber as nuances entre o que é real e artificial.

Esta dissertação lida com a simulação de um fenômeno físico que está presente em toda parte: o escoamento de fluido. Este capítulo contextualiza o problema e apresenta as razões de estudo desse tópico tão intrigante e instigante.

1.1 Contextualização e Motivação

Fenômenos como o vento que sopra pela copa de uma árvore, a fumaça que sai de uma vela ou mesmo a água que escoa de uma torneira são exemplos de fluidos com diferentes características e comportamentos. Apesar de parecerem simples em um primeiro momento, esses fenômenos ocultam grande complexidade e são difíceis de reproduzir computacionalmente. Antes do advento da computação, os estudos e análises dos fluidos eram feitos de maneira teórica, aprimorando o modelo e o conceito envolvido, ou experimental, avaliando o comportamento do fluido em ambiente controlado. Com o surgimento da Dinâmica de Fluidos Computacional (DFC) (do inglês *Computational Fluid Dynamics* - CFD) nasceu uma nova forma de análise do comportamento dos fluidos por meio de simulações em computador.

As simulações de DFC podem realizadas por meio da resolução das *equações de Navier-*

[1]Endereço eletrônico: $http://www.autodesk.com/eng/etc/fake\_or\_foto$

Stokes. Essas equações representam o modelo matemático do movimento do fluido. Como a solução analítica desse conjunto de equações é ainda um problema em aberto (elas possuem tal solução apenas em casos particulares), a simulação computacional torna-se um processo importante. Conseqüentemente, a resolução deve ser feita por meio de métodos numéricos como Diferenças Finitas (MDF), Volumes Finitos (MVF) ou Elementos Finitos (MEF) ou outro método de discretização das equações de Navier-Stokes.

As aplicações dos conceitos e idéias da DFC são numerosas e atraem pesquisadores de diversas áreas como engenharia mecânica, aeronáutica, medicina, meteorologia, entre outras. Essas aplicações são vantajosas em situações em que a modelagem experimental do problema em laboratório é difícil (ou possui custo financeiro elevado) ou para complementar resultados teóricos e experimentais. Exemplos variam desde simulação do fluxo sanguíneo nas artérias de um paciente até escoamento de fluidos em torno de um novo perfil de veículo automotor. Dessa forma, é natural que essas áreas requeiram bom grau de precisão do escoamento e, como conseqüência, as simulações têm custo computacional elevado, levando horas, dias ou até semanas para finalização de um ciclo de teste.

Existem ainda outras áreas em que a simulação de escoamento de fluidos tem papel importante. Pesquisadores da área de computação gráfica têm dado grande importância a esse tópico visto suas possíveis aplicações. Na indústria de efeitos especiais, esses fenômenos são muito estudados devido à demanda por comportamentos e aparências convincentes. Filmes como Mar em Fúria[2] ou O Dia Depois de Amanhã[3] possuem tomadas feitas inteiramente no computador, exigindo simulações realistas e conseqüentemente maior esforço computacional. Além dessas, existem aplicações em que é desejável que a simulação seja tão realista quanto possível e em tempo-real, por exemplo, ambientes de realidade virtual (realidade aumentada, cirurgia virtual, etc.) e jogos eletrônicos. Nesse caso, a precisão dos resultados pode ser sacrificada em prol do desempenho.

A simulação computacional de escoamento de fluidos que apresentem um comportamento realista em tempo real é um grande desafio em computação gráfica. Em geral, são feitas simplificações nas equações de Navier-Stokes para que o ganho de desempenho computacional seja considerável. Tais simplificações podem atuar de diferentes formas como, por exemplo, restringindo o tipo de movimento que o fluido executa, desprezando termos viscosos das equações, entre outros. Essas simplificações podem introduzir erros no processo de simulação, tornando-a inaceitável do ponto de vista científico. Entretanto, aplicações para computação gráfica não requerem um alto grau de precisão. De fato, o desempenho computacional vem

[2] *The Perfect Storm*. Produção da *WarnerBros*: http://www.warnerbros.com/
[3] *The Day After Tomorrow*. Produção da *20th Century Fox*: http://www.fox.com/

em primeiro lugar, de modo que a precisão é bem-vinda, mas não é o objetivo das simulações. Animações para computação gráfica geralmente objetivam replicar eventos naturais já conhecidos ao invés de simular fenômenos ainda desconhecidos, como o escoamento de fluido em torno de um modelo experimental de perfil de asa.

Profissionais da área de computação gráfica, como animadores, durante muito tempo fizeram uso de animação procedural para criação de fluidos, simulando o efeito do fluido. Com a gama de ferramentas disponíveis atualmente, esses profissionais conseguem o mesmo efeito (ou melhor) de maneira mais simples. Este projeto de mestrado tem como objetivo o *estudo e implementação de um simulador interativo de escoamento de fluidos baseado em partículas* que pode ser aplicado em simulações e treinamentos, realidade aumentada, jogos eletrônicos, enfim, qualquer área em que o ambiente necessite de fluidos que se comportem de maneira realista. Esse simulador é baseado em uma técnica que utiliza partículas para simulação de fluidos conhecida como *Smoothed Particles Hydrodynamics* (SPH). De forma geral, o desenvolvimento de um ambiente que permite a simulação de escoamento de fluidos pode ser dividido em: *simulador*, porção que lida com a resolução do problema; *representação da superfície*, porção que lida com a apresentação dos resultados ao usuário.

Os pontos de destaque dessa dissertação são:

- *Desenvolvimento de um integrador temporal que restringe o ganho de energia para esse cenário baseado em partículas, aumentando a estabilidade do método*. Durante o desenvolvimento do simulador, o método SPH se mostrou uma ferramenta sensível aos parâmetros iniciais. Uma simulação de escoamento de fluidos pode facilmente tornar-se uma explosão gasosa devido ao ganho de energia cinética e potencial do fluido. Assim, baseado na lei de conservação da energia, adaptamos uma solução que restringe o ganho de energia para esse cenário baseado em partículas. Como resultado, obtivemos um ganho na estabilidade do método (Seção 3.4);

- *Utilização do método da partição da unidade multinível robusta para geração de quadros para animação*. Como o SPH é um método baseado em partículas, a superfície do fluido deve ser extraida a partir de uma nuvem de pontos. A representação da superfície baseada em nuvem de pontos é uma área que chama a atenção dos pesquisadores. Nesse trabalho, utilizamos o método da partição da unidade multinível implícita robusta, uma técnica poderosa para geração de superfícies implícitas a partir de nuvem de pontos. Apesar dos resultados apresentarem fraca coerência entre quadros, essa técnica ainda pode ser explorada para fornecer melhores resultados para animação (Seção 2.7);

- *Implementação de uma interface entre o código desenvolvido para simulação de esco-*

amento de fluido e o software livre Blender 3D. Para tornar fácil e rápido processo de criação de um escoamento de fluido, foi desenvolvida uma interface para interação entre o código desenvolvido em linguagem C e o *Blender 3D*(Seção 4.5).

1.2 Organização do texto

Esse texto é dividido em quatro capítulos. O primeiro contém a introdução e motivação desse projeto. O segundo traz os conceitos fundamentais para o entendimento da DFC: o que é uma simulação e os diferentes tipos de discretizações do domínio usado na simulação, conceitos básicos de dinâmica de fluidos, conceitos básicos de visualização de pontos no espaço, além de uma revisão de importantes trabalhos na área de simulação de escoamento de fluidos para computação gráfica. O terceiro capítulo disserta sobre os conceitos de estabilidade, consistência e convergência além de apresentar uma técnica baseada na conservação da energia adaptada ao método *Smoothed Particles Hydrodynamics*. O quarto mostra a implementação do projeto, incluindo os resultados obtidos e as limitações encontradas. O último capítulo apresenta a conclusão e possíveis trabalhos futuros.

2 Fundamentos

Nesse capítulo, serão dados os primeiros passos para compreensão dos elementos envolvidos no escoamento, além de sua visualização. Primeiramente, são mostradas as convenções adotadas no restante do trabalho (Seção 2.1). Logo após, é feita uma explicação a respeito das simulações numéricas (Seção 2.2) e as discretização do domínio envolvido na simulação (Seção 2.3) e uma breve introdução à Dinâmica de Fluidos Computacional (Seção 2.4). Em seguida, são apresentados os trabalhos relacionados a este (Seção 2.5) assim como uma descrição detalhada do método SPH (Seção 2.5.2), foco dessa monografia. O capítulo é finalizado com uma breve discussão a respeito da representação computacional do fluido (Seção 2.7).

2.1 Convenções do texto

Nesse texto, letras minúsculas em negrito, como em $\mathbf{v}$, são usadas para diferenciar um vetor de um escalar, denotado por v. Um ponto acima de uma letra, como em $\dot{w}$ denota diferenciabilidade. A notação $\nabla\phi$ denota o gradiente de uma função $\phi = \phi(x,y,z)$. No espaço euclidiano é definido como:

$$\nabla\phi = \left(\frac{\partial\phi}{\partial x}, \frac{\partial\phi}{\partial y}, \frac{\partial\phi}{\partial z}\right)^T$$

$\nabla \cdot \mathbf{w}$, denota o divergente de um campo vetorial $\mathbf{w} = (w_x, w_y, w_z)^T$. No espaço euclidiano é definido como:

$$\nabla \cdot \mathbf{w} = \frac{\partial w_x}{\partial x} + \frac{\partial w_y}{\partial y} + \frac{\partial w_z}{\partial z}$$

$\nabla^2\phi$, denota o laplaciano de uma função ϕ. No espaço euclidiano é definido como:

$$\nabla^2\phi = \frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2} + \frac{\partial^2\phi}{\partial z^2}$$

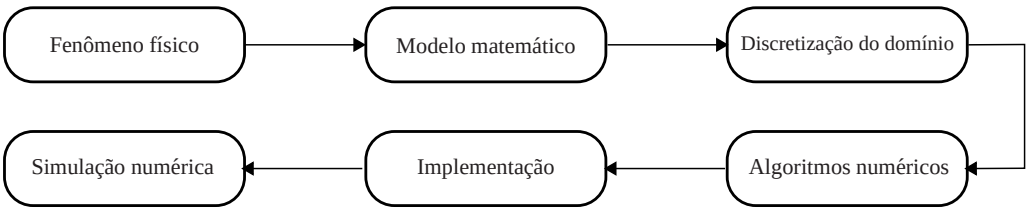


Figura 2.1: Seqüência de passos necessária para criação de uma solução numérica para um fenômeno físico. Figura adaptada de [30].

Hf, denota a hessiana de uma função $f(x,y,z)$. No plano cartesiano é definida como:

$$Hf = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial x \partial z} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} & \frac{\partial^2 f}{\partial y \partial z} \\ \frac{\partial^2 f}{\partial z \partial x} & \frac{\partial^2 f}{\partial z \partial y} & \frac{\partial^2 f}{\partial z^2} \end{bmatrix},$$

e também $Hf\mathbf{u}$, $u = (u_x, u_y, u_z)^T$:

$$Hf\mathbf{u} = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial x \partial z} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} & \frac{\partial^2 f}{\partial y \partial z} \\ \frac{\partial^2 f}{\partial z \partial x} & \frac{\partial^2 f}{\partial z \partial y} & \frac{\partial^2 f}{\partial z^2} \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix}.$$

A mudança da fonte denota um trecho de código ou pseudocódigo, como em:

```
var = (exp) ?  a :  b;
var = var & 0xFF88AA00;
```

2.2 Simulações numéricas

Nas simulações numéricas, um fenômeno contínuo é discretizado em formas matemáticas, recriando uma situação real em um ambiente virtual. Com o crescimento do poder computacional, é possível ter uma representação mais fiel do comportamento dos fenômenos em menor tempo. Conseqüentemente, os resultados teóricos previstos, resultados experimentais observados e numéricos podem ser confrontados e avaliados. Nesse cenário, a simulação numérica atua como uma ponte entre teoria e experimento, levando a melhoria de ambos, além da própria simulação.

A Figura 2.1 mostra os passos necessários para construção da simulação numérica de um fenômeno natural. Inicialmente um *fenômeno físico* é observado e simplificado. Em seguida, é extraído um *modelo matemático*, chamado de equações governantes, que descreve esse fenômeno. Em geral, na natureza, essas equações são escritas na forma de um con-

junto Equações Diferenciais Ordinárias (EDOs) ou Equações Diferenciais Parciais (EDPs). O próximo passo é a *discretização do domínio* ou *discretização espacial*. Para encontrar a solução das equações governantes é necessário dividir seu domínio de atuação em partes discretas. Da mesma forma, esse processo de discretização estende-se para as equações, já que são dadas na forma contínua. Em geral a discretização é feita usando *malhas* ou *grades* de pontos (veja Figura 2.2 para discretização unidimensional do eixo x). São nessas malhas de pontos que as variáveis de campos do problema (velocidade, pressão, força, entre outras) são calculadas.

O próximo passo é a construção dos *algoritmos numéricos*. Para modelar esses algoritmos, é necessária a discretização das equações governantes (EDO, EDP, etc.) além da configuração das condições iniciais (CI) e/ou condições de contorno (CC) do problema. Essas informações resultam em um conjunto de equações que são resolvidas pelos algoritmos numéricos existentes na literatura [5]. A *implementação* é a etapa de codificação dos algoritmos numéricos para a simulação numérica. Nessa etapa deve ser levada em conta a precisão computacional (erros por arredondamento da máquina), velocidade de processamento, capacidade de armazenamento, paralelismo, entre outras variáveis. Com todas essas etapas concluídas, a simulação numérica de um fenômeno natural é realizada e estudos aprofundados podem ser feitos, criando diversos cenários de simulação e obtendo os resultados.

Essa abordagem para criação de uma solução numérica para as equações governantes pode parecer desnecessária já que é feita apenas para obter a solução das equações governantes. A dificuldade é que, exceto em algumas situações particulares, equações governantes na forma de EDP não possuem solução analítica. Por isso é preciso criar uma aproximação da solução utilizando a simulação numérica. A Figura 2.2 mostra o resultado de uma simulação nos pontos discretos.

2.3 Discretização espacial

O próximo passo após a definição do modelo matemático é a discretização espacial do domínio de simulação. O domínio do problema pode ser entendido como o ambiente onde a simulação ocorre. Por exemplo, um problema clássico em dinâmica dos fluidos é o *shear driven cavity*. Ele consiste em uma caixa cujo interior está repleto por apenas um tipo de fluido. A tampa superior dessa caixa movimenta-se na horizontal com velocidade constante e assim o fluido em seu interior começa a se mexer. Nesse exemplo, o domínio do problema é a caixa onde o fluido está contido. Outro problema clássico é o *tubo de choque*, onde dois fluidos distintos estão separados no interior de um tubo por uma película. Quando essa película é retirada esses fluidos entram em choque. Nesse caso, o tubo é o domínio do problema. Ambos

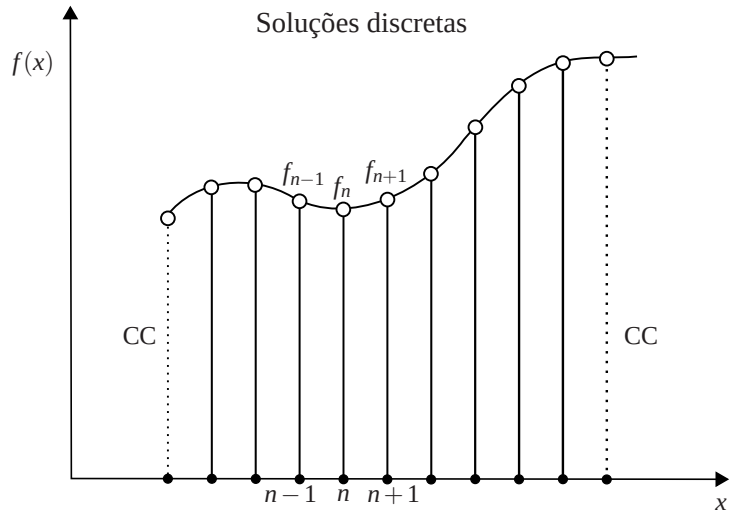


Figura 2.2: Aproximação discreta de uma função juntamente com suas condições de contorno. O eixo x é discretizado em uma malha de pontos unidimensional e os valores de f são calculados apenas nesses pontos. Figura adaptada de [30].

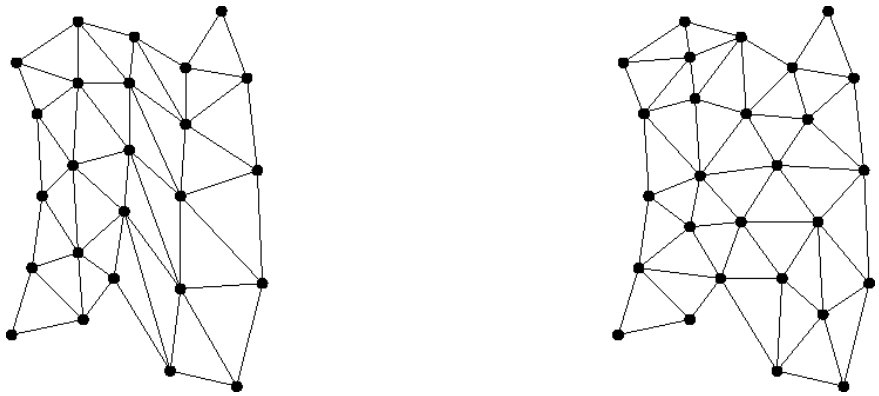


Figura 2.3: Uma malha estruturada (esquerda) e uma malha não-estruturada (direita). Figura extraída de [54].

os domínios apresentados anteriormente podem ser discretizado, por exemplo, criando-se uma malha de pontos no seu interior. Nesta seção serão apresentadas duas maneiras de realizar a discretização do domínio de acordo com o tipo de equação governante que rege o fenômeno de interesse.

2.3.1 Métodos baseados em malhas

Existem basicamente duas abordagens para descrição das equações governantes de fenômenos físicos [30]: a abordagem Euleriana e Lagrangeana. As equações de movimento resultantes das abordagens são diferentes já que utilizam formas distintas para descrever o movimento de um corpo. A abordagem Euleriana descreve o movimento em relação ao espaço ao redor do material, já a abordagem Lagrangeana descreve o movimento por meio do próprio

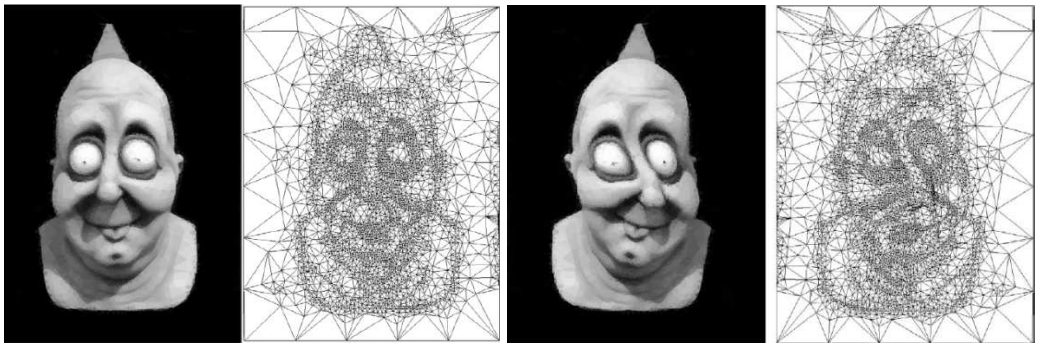


Figura 2.4: Os triângulos da malha que compõem o domínio se movem junto com as partículas numa simulação elástica. À esquerda estão as duas imagens antes da deformação e a direita após a deformação. A malha da imagem acima foi gerada com o algoritmo *Imesh* [8].

material que se move.

Métodos baseados em malhas (ou grades) são estudados desde a década de 50. Eles baseiam-se na discretização de um domínio de interesse em células (primitivas geométricas como triângulos, quadriláteros e outros no caso bidimensional ou tetraedros, prismas e outros no caso tridimensional). Essas discretizações têm influência no método de resolução das equações governantes. Dependendo do tipo de equação de movimento que é resolvida, o espaço (equação Euleriana) ou o material (equação Lagrangeana) deve ser discretizado.

As malhas podem ser de dois tipos [54]: malhas estruturadas ou não-estruturadas. As malhas estruturadas não precisam manter explicitamente uma relação de ordem entre suas células. Assim, dada uma célula, todas as células vizinhas são conhecidas e podem ser acessadas facilmente utilizando uma simples função. Uma grade cartesiana é uma malha estruturada onde as células são alinhadas com os eixos cartesianos (Figura 2.5). Toda malha estruturada tem a mesma topologia de uma grade cartesiana. Malhas não-estruturadas precisam manter uma relação explícita de ordem entre suas células, de maneira que não é possível a partir de uma célula qualquer, conhecer quaisquer de seus vizinhos sem uma estrutura de dados auxiliar. Nesse caso essa estrutura é obrigada a armazenar para cada célula seus vizinhos utilizando, por exemplo, uma lista encadeada. Ambos os tipos de malhas são ilustrados na Figura 2.3.

Discretização Lagrangeana

Nesse método, o *objeto* em questão é discretizado em uma malha. Nesse caso, o domínio em questão se move inteiramente com o material onde ocorre a simulação. A Figura 2.4 ilustra o movimento de uma malha lagrangeana.

O uso de malhas lagrangeana apresenta algumas vantagens [30]:

1. A malha se move com o domínio em questão. Dessa forma, não existem termos convectivos (Seção 2.4.1) relacionados às equações diferenciais e, portanto o código-fonte é conceitualmente mais simples e mais rápido. Nesse tipo de método, não é necessário um esforço extra para codificação desses termos convectivos além de um ganho computacional;

2. Em métodos lagrangeanos, a configuração da malha se adapta ao escoamento;

3. Malhas de pontos são necessárias apenas no material onde ocorre a simulação. Não é necessária informação além do domínio do problema.

Uma desvantagem no uso de malhas lagrangeanas é a dificuldade em lidar com casos de grandes deformações no domínio. Essas distorções podem comprometer o resultado final da simulação e mesmo interromper a execução natural do sistema. Uma possível solução é realizar uma *remalhagem*, processo que consiste em transformar uma malha em outra que seja mais adequada para o problema [56], no domínio em regiões necessárias. Contudo o processo pode ser computacionalmente caro, inviabilizando seu uso em aplicações interativas.

Discretização Euleriana

Contrariamente às malhas lagrangeana, nesse método o *domínio espacial* de simulação é discretizado, mantendo-se fixo. Nessas simulações, o objeto em questão se move *sobre* a grade fazendo que cada uma de suas células seja preenchida de acordo com os resultados das simulações. O Método das Diferenças Finitas (MDF) está entre os muitos métodos que utilizam esse tipo de grade. A Figura 2.5 ilustra uma malha euleriana.

Uma vantagem de malhas eulerianas é que ela suporta grandes deformações do objeto da simulação. Os pontos da grade permanecem fixos durante o movimento do objeto e assim não gera problemas numéricos devido às grandes deformações como ocorre nas malhas lagrangeanas. Essa é uma das razões que fazem com que os métodos eulerianos sejam preferidos em simulações de fluidos. Contudo, há algumas desvantagens que devem ser mencionadas.

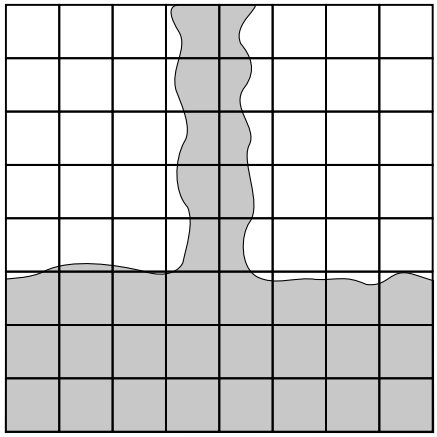


Figura 2.5: Usando uma malha estruturada é possível representar o domínio inteiro onde ocorre a simulação.

1. É difícil trabalhar com simulações que envolvam geometrias complexas. Em geral, é preciso fazer um mapeamento para uma grade regular quando é encontrado esse tipo de situação, o que pode tornar o processo custoso do ponto de vista computacional;

2. Métodos eulerianos precisam de uma grade de pontos que envolva toda a região possível que o objeto possa estar presente. Simulações com maior precisão requerem uma grade mais fina e, portanto computacionalmente mais cara;

3. A posição da superfície livre, interface entre dois fluidos distintos, é difícil de ser calculado;

Existem ainda combinações de formulações euleriana e lagrangeana para resolução de equações governantes, chamadas de métodos ALE (*Arbitrary Lagrangian-Eulerian*) [22]. Um exemplo comum dessa abordagem é quando uma malha se move a uma velocidade diferente do fluido em que está imersa para o melhoramento da malha utilizada.

2.3.2 Métodos baseados em partículas

Os métodos lagrangeanos baseados em partículas (ou pontos), conhecidos como *Meshfree Methods* (MM), são uma alternativa para discretização das equações governantes. Eles tentam superar os problemas oriundos do uso de formulações baseadas em malha, tornando-se uma alternativa atraente na simulação de diversos tipos de fenômenos. Algumas vantagens dos MM são [29]:

- *Grandes deformações*. Eles conseguem lidar com grandes deformações já que a conectividade entre os pontos envolvidos faz parte da computação do resultado da simulação;

- *Fácil representação*. A estrutura de dados para representação do conjunto de partículas pode ser uma simples lista;
- *Mudanças topológicas*. Os métodos são projetados para se adaptarem às mudanças topológicas de um corpo contínuo, inclusive fratura, explosões, deformações, etc.;
- *Refinamento adaptativo*. Eles conseguem lidar de maneira robusta com refinamento adaptativo para controle da precisão computacional dos resultados. Isso porque é possível inserir facilmente pontos onde o refinamento é necessário, sem nenhum problema com informações topológicas;
- *Representação de objetos*. Métodos baseados em pontos podem representar objetos geométricos com boa precisão.

Dentre as desvantagens dos métodos baseados em partículas destacam-se [16]:

- *Cálculo da vizinhança*. As grandes deformações suportadas por esses métodos vêm com o custo extra de calcular, a cada iteração, a vizinhança das partículas por meio de estrutura de dados adequadas;
- *Consistência*. Apesar dos resultados numéricos apontarem que métodos livres de malha podem convergir mais rapidamente para a mesma ordem de consistência, a teoria para métodos de alta ordem de consistência deixa a desejar;
- *Estado inicial das partículas*. O posicionamento inicial das partículas tem forte influência no resultado numérico, dessa forma, uma tarefa não trivial passa a ser a sua configuração inicial;
- *Esforço computacional*. O esforço computacional dos métodos livre de malha pode, em alguns casos, ser maior do que os baseados em malhas. Isso vem do número de vizinhos necessários para se obter uma solução suficientemente precisa em métodos baseados em partículas.

As vantagens e desvantagens acima citadas são relativas ao aspecto numérico para aplicações científicas ou voltadas para engenharia. Do ponto de vista da computação gráfica, algumas dessas têm menor peso ou são negligenciáveis, cedendo lugar a outras variáveis, como estabilidade numérica.

Smoothed Particles Hydrodynamics

O método *Smoothed Particles Hydrodynamics* (SPH) foi desenvolvido por Lucy [34] e Gingold e Monaghan [19] simultaneamente para problemas astrofísicos. A técnica baseia-se no conceito de aproximação local de uma função em torno de um ponto por meio de uma integral. No SPH a discretização do domínio é feita por meio de partículas e as propriedades relevantes como pressão, velocidade, etc., são calculadas nessas partículas.

Outros métodos

Existem diversos tipos de MM baseados em partículas tais como *Molecular Dynamics*, Monte Carlo, *Particle-In-Cell*, MPS, entre outros. Esses métodos compartilham as mesmas vantagens do MM com diferenças na discretização do domínio e formulação do método, sendo que alguns assumem inclusive formulações lagrangeana-euleriana. Fires e Matties [16] fazem uma revisão detalhada sobre os diversos tipos de MM. Dentre os MM baseados em partículas, o SPH tem se destacado na simulação de escoamento de fluidos sendo amplamente estudado e desenvolvido pela comunidade científica. Além disso, o SPH permite a realização das simulações em tempo linear, o que é altamente desejável para aplicações em tempo-real [39].

Representação das partículas

Em MM, não há necessidade de armazenar as informações de conectividade. As simulações requerem somente uma distribuição inicial de pontos. Como visto anteriormente, a distribuição de pontos não é uma tarefa trivial dependendo do problema em questão. No entanto, para computação gráfica, a questão do posicionamento inicial é menos severa e por isso, nesse texto, vamos adotar uma postura simplificada em relação a isso. Assim, há diversas maneiras de gerar uma distribuição inicial no domínio, por exemplo, a partir de uma malha que o representa e escolhendo como partículas os centróides dos elementos que compõem a malha[1]. Outra maneira é a geração aleatória de partículas no domínio ou por meio da criação de uma fonte de partículas. A Figura 2.6 ilustra esses exemplos.

[1]Construir uma malha para o domínio em questão para gerar uma distribuição inicial de partículas advém do fato que atualmente existem geradores de malha 2D e 3D. Contudo, não é necessário a geração de uma malha.

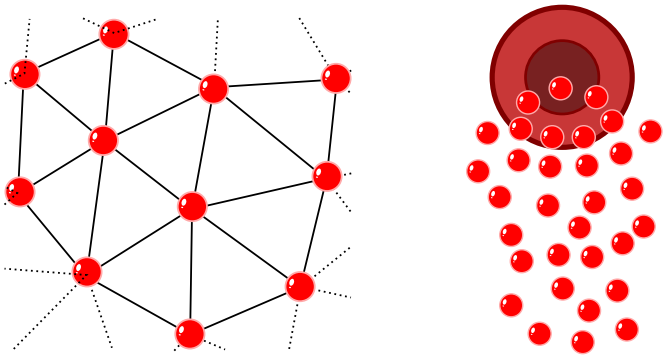


Figura 2.6: Diferentes métodos podem ser usados para geração de partículas. A imagem à esquerda ilustra a geração de partículas a partir de uma malha, colocando cada partícula no centróide dos triângulos. A imagem à direita ilustra a geração de partículas de maneira aleatória em torno de uma fonte, o círculo central.

2.4 Dinâmica dos Fluidos

A Dinâmica dos Fluidos Computacional (DFC) surgiu há algumas décadas para complementar o estudo teórico e experimental de dinâmica de fluidos. Desde o seu surgimento houve intenso investimento em métodos baseados em malhas para resolução das equações governantes das simulações de fluidos, as equações de Navier-Stokes.

A dificuldade da simulação de escoamento de fluidos advém da natureza complexa do comportamento do fluido que é resultado da interação entre vários fenômenos como convecção, difusão, tensão superficial e turbulência.

O comportamento do escoamento do fluido pode classificado de duas maneiras [9]:

- *Estacionário ou laminar*: o fluido escoa como lâminas, sem variações bruscas no comportamento. Dentre as simulações de fluidos, esse é o tipo de escoamento mais simples de ser simulado;
- *Turbulento*: nesse regime o comportamento do fluido é imprevisível, podendo assumir qualquer forma ou direção. Esse é o tipo de escoamento mais difícil de ser simulado;

Existe ainda uma fase de transição entre o escoamento laminar e o turbulento. Esse escoamento é denominado *transicional*. Ele pode ser observado com freqüência no cotidiano, por exemplo, em uma chama de cigarro, como ilustrado na Figura 2.7.

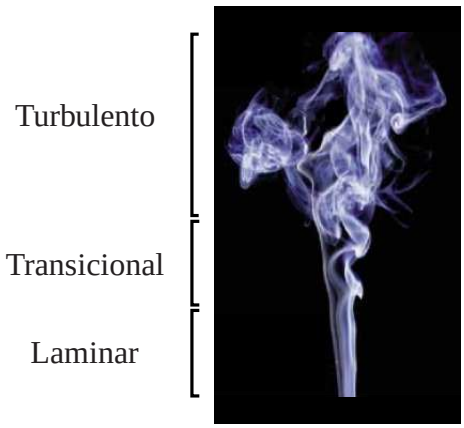


Figura 2.7: Diferentes tipos de escoamento. A fumaça que sai do cigarro inicialmente possui um escoamento laminar. Logo em seguida, passa por um estágio de transição até se tornar turbulento. Figura adaptada do endereço http://boojum.as.arizona.edu/∼jill/.

2.4.1 Equações de Navier-Stokes

As equações de Navier-Stokes formam o conjunto de equações que modelam o comportamento do fluido. As equações foram descobertas independentemente por Claude Navier e George Stokes na primeira metade do século XIX e são consideradas o melhor modelo matemático que descreve o movimento do fluido.

Há diferentes formas de representação das equações dependendo do que é assumido. Abaixo é mostrada uma versão compacta das equações para fluidos newtonianos e compressíveis:

$$\frac{\partial \rho}{\partial t} = -\rho \nabla \cdot \mathbf{v} \tag{2.1}$$

$$\frac{\partial \mathbf{v}}{\partial t} = -(\mathbf{v}\cdot\nabla)\mathbf{v} + \nu\nabla^2\mathbf{v} - \frac{1}{\rho}\nabla p + \mathbf{f} \tag{2.2}$$

$$\begin{aligned}\rho\frac{\partial e}{\partial t} = {} & -(\mathbf{v}\cdot\nabla)e - p\nabla\cdot\mathbf{v} + \\ & \mu\varepsilon_{xx}\frac{\partial v_x}{\partial x} + \mu\varepsilon_{xy}\frac{\partial v_x}{\partial y} + \mu\varepsilon_{xz}\frac{\partial v_x}{\partial z} + \\ & \mu\varepsilon_{yx}\frac{\partial v_y}{\partial x} + \mu\varepsilon_{yy}\frac{\partial v_y}{\partial y} + \mu\varepsilon_{yz}\frac{\partial v_y}{\partial z} + \\ & \mu\varepsilon_{zx}\frac{\partial v_z}{\partial x} + \mu\varepsilon_{zy}\frac{\partial v_z}{\partial y} + \mu\varepsilon_{zz}\frac{\partial v_z}{\partial z},\end{aligned} \tag{2.3}$$

onde $\varepsilon_{\alpha\beta}$, $\alpha,\beta \in \{x,y,z\}$ advém da hipótese de Stokes:

$$\varepsilon_{\alpha\beta} = \frac{\partial v_\beta}{\partial \alpha} + \frac{\partial v_\alpha}{\partial \beta} - \frac{2}{3}(\nabla\cdot\mathbf{v})\delta^{\alpha\beta} \tag{2.4}$$

com

$$\delta^{\alpha\beta} = \begin{cases} 1 & \text{se } \alpha = \beta \\ 0 & \text{caso contrário} \end{cases}$$

A Equação 2.1 é chamada *equação da continuidade* ou *conservação da massa* e impõe que, na ausência de fontes de massa ou sorvedouros, toda massa que entra no sistema deve sair ou acumular. Nessa equação $\mathbf{v}$ é a velocidade.

A Equação 2.2 é chamada de *equação da quantidade de movimento* ou *equação de balanço do momento*, resultado direto da segunda lei de Newton. Ela mostra a evolução temporal da velocidade de uma partícula de fluido. Nessa equação, t representa o tempo, $\mathbf{v}$ a velocidade de um elemento de fluido, ν representa a viscosidade cinemática do fluido dada por $\nu = \frac{\mu}{\rho}$, μ é a viscosidade do fluido, ρ é a densidade do fluido, p é a pressão exercida e $\mathbf{f}$ representa as forças externas atuantes.

O termo $-(\mathbf{v}\cdot\nabla)\mathbf{v}$ é chamado termo convectivo e representa o deslocamento dos elementos de fluido com velocidade $\mathbf{v}$. O segundo termo $\nu\nabla^2\mathbf{v}$ é chamado de termo viscoso e representa a perda de energia devido à interação entre as partículas. O termo ∇p, chamado termo difusivo, tende a espalhar as partículas de fluidos. O último termo representa ação de forças externas.

A Equação 2.3 é a equação de *conservação da energia* onde e é a energia interna de um elemento de fluido. Como será visto mais adiante (Capítulo 3), em um sistema isolado deve haver conservação da energia e essa equação fornecerá ferramentas para o cálculo da energia interna envolvida em uma simulação de fluidos (Seção 3.4). Uma dedução detalhada a respeito das equações pode ser encontrada em [9].

2.5 Trabalhos Relacionados

Existe na literatura um grande número de trabalhos dedicados a simulação de escoamento de fluidos utilizando abordagens eulerianas, lagrangeanas ou mistas. Esta seção apresenta alguns dos trabalhos clássicos em computação gráfica e na seqüência trabalhos utilizando o método SPH para simulação do escoamento de fluido e outros fenômenos.

2.5.1 Computação Gráfica

Um dos trabalhos clássicos na área de fluidos para computação gráfica é o de Kass et. al. [24]. Os autores introduziram para comunidade científica um método rápido e estável para animação de água baseado na solução das equações de águas rasas, simplificações das

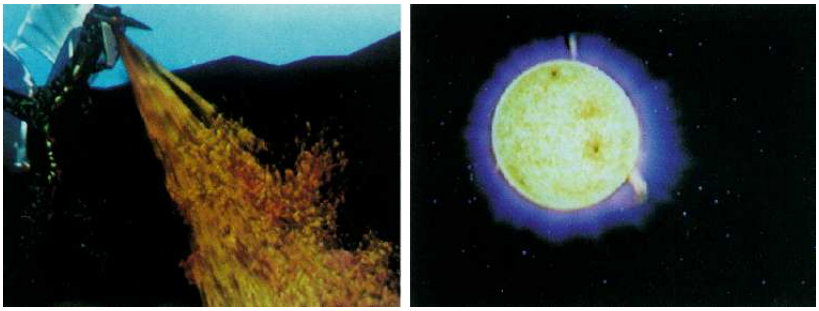


Figura 2.8: Geração de imagens usando simulações por partícula [55]. A esquerda uma baforada de um dragão e a direita uma nebulosa.

equações de Navier-Stokes. O método proposto lineariza as equações de águas rasas tornando-as equações da onda. Na abordagem proposta, a superfície do líquido é representada por uma função altura e a velocidade do fluido é uniforme nas colunas de líquido verticais. Kass utiliza uma abordagem Euleriana para resolução das equações de águas rasas. Um malha cartesiana é gerada e as simulações do escoamento são realizadas nessa malha. Esse processo implica na solução de um sistema linear tridiagonal facilmente resolvível em tempo linear [48]. A integração temporal é feita pelo método de Euler implícito a fim de obter uma simulação numericamente estável. Kass, no entanto, devido às limitações das equações de onda, se concentra na simulação de escoamentos não turbulentos e assim deixa de lado um fenômeno interessante do comportamento do fluido.

No início da década de 90, Sims [55] desenvolveu um sistema de partículas usado para modelagem e animação de quedas d'água, além de fogo, neve, fogos de artifício, grama, entre outros. Ele utilizou a segunda Lei de Newton, $\mathbf{f} = m\mathbf{a}$, para movimentação das partículas, onde a força $\mathbf{f}$ aplicada poderia ser de vários tipos, dependendo do comportamento desejado (centrípeta, espiral, constante, etc.). O *hardware* usado por Sims para a simulação era composto por um computador altamente paralelo para acelerar o processo de integração numérica. Além disso, foi usado um mecanismo de processadores virtuais para aumentar no número de processadores disponíveis. Dessa forma, cada processador representa uma partícula e efetua as operações relevantes sobre ela, aumentando o desempenho da simulação. A Figura 2.8 ilustra os resultados alcançados. Contudo, Sims não utiliza as equações de Navier-Stokes para simulação dos fluidos. Isso restringe severamente os tipos de fenômenos que podem ocorrer já que é preciso acertar previamente os tipos de forças envolvidas.

O'Brien e Hodgins [44] estendem a técnica apresentada por Kass [24] propondo um método para animação de fluidos submetidos à ação de forças externas, como por exemplo, uma pedra que atinge um lago ou objetos que flutuam na superfície da água. O'Brien e Hodgins incluíram nas equações de Kass um termo para interação com forças devido à ação de outros corpos. São

simuladas as ondas resultantes bem como o espirro de água causado por uma força agindo na superfície livre. Para isso, foi utilizado um sistema de partículas para simulação do borrifo d'água além da malha de pontos utilizada para simulação das ondas causadas por impacto com objetos. As partículas são criadas quando a velocidade vertical de uma porção da superfície excede um determinado limite, caracterizando o espirro d'água. Apesar de estender o método apresentado por Kass, ainda possui algumas de suas limitações, por exemplo, a impossibilidade de simulação de quebra de onda.

O trabalho de Stam [57] pode ser considerado um divisor de águas no âmbito da computação gráfica. O autor foca na simulação de fenômenos com gases. Stam mostra uma maneira inovadora para simular fluidos usando a equação de Navier-Stokes em três dimensões de maneira estável e utilizando uma abordagem semi-lagrangeana. Ele utiliza uma grade de pontos cartesiana para o cálculo da velocidade e pressão em cada ponto para resolução das equações de Navier-Stokes. Stam utiliza uma variação de uma técnica para resolução de equações diferenciais parciais, chamada *método das características*, para resolver o termo não-linear de maneira estável. Stam adiciona cada termo da equação de Navier-Stokes, finalizando com uma projeção em um espaço livre de divergente. O método resultante é incondicionalmente estável, uma qualidade fortemente desejável do ponto de vista da computação gráfica. A Figura 2.9 mostra alguns resultados da simulação de fumaça. Esse trabalho deu um passo importante em direção à simulação interativa de fluidos.

Figura 2.9: Simulação de escoamento de fluidos usando a técnica de fluidos estáveis [57].

Mais tarde, outro trabalho de Stam [58] mostra como simular fenômenos de fluidos utilizando a equação da variação da densidade para jogos eletrônicos de maneira incondicionalmente estável. Como no trabalho anterior, ele utiliza uma malha de pontos cartesiana para resolução das equações da densidade. Essa equação tem a seguinte forma:

$$\frac{\partial \rho}{\partial t} = -(\mathbf{v} \cdot \nabla)\rho + \mu \nabla^2 \rho - \frac{1}{\rho}\nabla p + s.$$

A forma dessa equação é muito semelhante à equação de Navier Stokes (Equação 2.2) e os parâmetros são os mesmo apresentados na Seção 2.4.1 e s é um termo-fonte. Uma das vantagens

Figura 2.10: Água caindo em um copo utilizando a técnica apresentada em [39].

imediatas de se utilizar essa equação é que não possui o termo não-linear existente nas equações de Navier-Stokes, tornando-a mais simples.

Para geração de imagens, Stam visualiza a densidade ao invés do campo de velocidade. A visualização do campo de velocidade é interessante quando objetos são movidos por esse campo, por exemplo, na simulação de partículas de fumaça. Contudo, nesse caso, o desempenho da aplicação seria comprometido devido ao grande número de partículas necessárias para tornar a simulação visualmente atrativa. No caso da visualização da densidade, Stam utilizou uma malha cartesiana de pontos e em cada célula atribuiu um valor de densidade. A atribuição da tonalidade de cada célula passa a ser proporcional a densidade daquela célula, um processo simples e com bons resultados. Stam ainda apresenta nesse trabalho um tutorial para resolução das equações acima resultando em um programa com pouco mais de 100 linhas de código em linguagem C que realiza em tempo real simulação de fenômenos de gases de maneira estável.

Outro trabalho que merece destaque é o de Foster e Fedkiw [17] onde os autores se baseiam na técnica de Stam [57] para criar um método realista para simulação de fluidos viscosos, variando de água até lama. Os autores utilizam uma combinação de partículas e uma função *level-set* para rastrear a evolução do volume/superfície do fluido, onde a evolução temporal das partículas é dada de forma lagrangeana enquanto a função *level-set* evolui de forma euleriana. Para obter a vantagem de ambos as técnicas elas são combinadas de forma a preservar regiões de grande curvatura, locais ricos em detalhes onde ocorrem borrifos d'água ou *splashes*, e suavizar as de baixa curvatura, superfície plana do fluido. Os resultados obtidos pelos autores são de fato impressionantes (o filme *Shrek*[2] utiliza essa técnica para a cena do banho de lama), com grande nível de realismo por meio dos detalhes. Uma das desvantagens da técnica é o grande consumo de memória principal e processamento da CPU para atingir bons níveis de detalhe do fluido. Na mesma linha

Em 2003, Müller, Charypar e Gross [39] introduzem para a comunidade de computação

[2]Endereço eletrônico: $http://www.shrek.com/$

gráfica uma abordagem baseada em partículas, o SPH (Seção 2.5.2), para a simulação de escoamento de fluidos, como a água, para aplicações interativas. Os autores resolvem cada termo da equação de Navier-Stokes de maneira independente, aproximando as propriedades de cada partícula, como pressão, velocidade, etc., por meio de funções núcleo suaves (*smooth kernels*). Diferente do trabalho de Stam [57], o método desenvolvido por Müller et. al. não é incondicionalmente estável. O processo de visualização conta com uma etapa para identificação das partículas que estão na superfície do fluido e então a geração de uma malha utilizando o método de *marching cubes*. O resultado são animações convincentes de 2200 partículas executadas a uma taxa que varia de 4 a 25 quadros por segundos dependendo da técnica utilizada para visualização do fluido (Figura 2.10). Uma dificuldade para métodos baseados em partículas é a geração de uma malha de pontos em volta do conjunto de pontos de maneira eficiente. Soluções para o problema de geração de uma superfície a partir de um conjunto de pontos geralmente possui alto custo computacional, o que pode comprometer o desempenho da aplicação.

O recente trabalho de Irving et. al. [23] mostra uma solução para o problema de simulação de grandes volumes de água de maneira eficiente. Técnicas tradicionais que utilizam grades cartesianas uniformes produzem bons resultados na simulação de escoamento de fluidos, contudo um alto preço computacional é pago se o volume de água é muito grande. Isso ocorre devido ao refinamento uniforme da grade cartesiana. Há ainda alguns métodos que procuram criar grades não uniformes, de maneira que o número de células diminui na medida em que se afasta da superfície do fluido utilizando uma estrutura de dados como *octree*. Contudo, as células que representam o fundo do contêiner passam a ser grandes e, portanto, fazem com que a superfície não tenha nenhuma informação a respeito dos eventos que ocorrem no fundo, comprometendo a simulação. Para solucionar esse problema, ao invés de refinar a grade cartesiana, Irving et. al. detalha apenas as regiões próximas à superfície do fluido e nas profundezas do local onde o fluido se encontra. Outras regiões são representadas por células mais largas, o que reduz consideravelmente o custo da simulação. A Figura 2.11 mostra alguns resultados da simulação de grandes volumes de água. Nesse trabalho, o enfoque na qualidade final das imagens resulta em um alto custo computacional, o que torna a simulação inviável para aplicações interativas.

Yuksel, House e Keyser [61] desenvolveram um novo método incondicionalmente estável para simular, em tempo-real, ondas em oceanos e em piscinas baseado em função altura. O método proposto introduz o conceito de *partículas de onda* que transportam a altura do fluido. As partículas se movem sobre um plano bidimensional (plano *xy*) variando a altura da onda (eixo *z*) de acordo com as suas posições. Para uma simulação realista de ondas, os autores utilizam uma função de colagem (*blending*) para criação de frentes de ondas. Além disso, o trabalho conta com a simulação de expansão e contração de ondas além de interação com outros objetos.

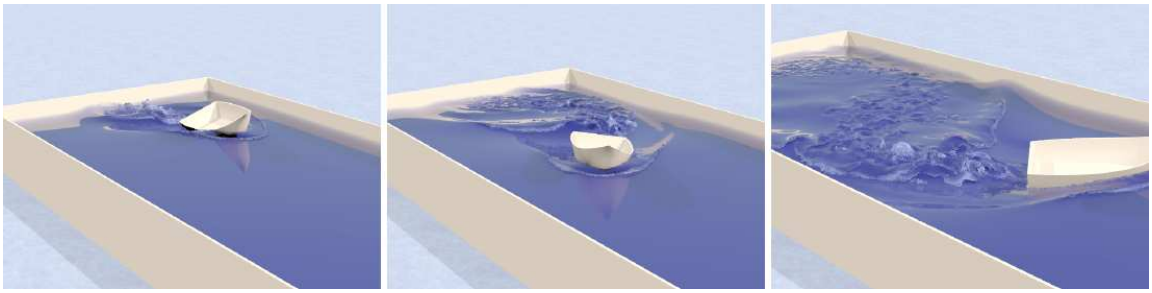


Figura 2.11: Simulação de grandes volumes de água [23].

Como uma desvantagem, a técnica é capaz de lidar apenas com fluidos que se comportam como uma função altura, ou seja, quebra de ondas, borrifos de água e outros fenômenos interessantes estão fora do escopo.

Além dos trabalhos apresentados, existem muitos outros que focam em tanto em técnicas para redução do consumo de poder computacional [62, 32] quanto no aperfeiçoamento dos resultados visuais [13, 50]. Há também outras técnicas de discretização das equações de Navier-Stokes para animação de fluidos. A próxima seção apresenta alguns dos trabalhos que utilizam o método SPH para simulação de escoamento de fluidos.

2.5.2 *Smoothed Particles Hydrodynamics*

Na computação gráfica, o trabalho de Desbrun e Gascuel [10] introduziu o método SPH para comunidade de computação gráfica para animação de corpos deformáveis. Anos mais tarde, Müller, Charypar e Gross [39] pela primeira vez resolvem as equações de Navier-Stokes utilizando o método SPH. Desde então, vários trabalhos contribuíram para o avanço da técnica como [43] onde o método é usado para modelar a interação entre diferentes fluidos. Paiva et. al. [46] fazem a simulação de objetos com comportamento plástico e que mudam de fase, do solido para o fluido. A idéia é modelar os objetos como fluidos não-newtonianos onde a transição de estado desses fluidos ocorre de alta para baixa viscosidade. A equação do calor é usada para difundir o calor pela superfície do objeto. Á medida que a temperatura varia, a viscosidade do fluido é alterada por um parâmetro denominado *jump number*, um parâmetro reológico que combina um conjunto de outros parâmetros e o objeto muda de fase. Adams et. al. [1] desenvolveram uma técnica adaptativa para o método SPH, onde regiões podem ser refinadas ou compactadas de acordo com o número de partículas necessárias para representar detalhes do fluido. Para isso, os autores utilizam critérios geométricos: regiões onde não há obstáculos complexos ou no fundo de um contêiner possuem poucas partículas ao passo que outras regiões são bem amostradas para preservar a forma do fluido.

O trabalho de Cleary et. al. [7] utiliza o método SPH para simulação realista de líquidos gasosos e que formam espumas. Os autores modelam a formação de bolhas no interior do fluido considerando que cada partícula carrega como propriedade uma quantidade de gás dissolvida. A medida que essas partículas interagem a quantidade de gás reunido pode ultrapassar um limiar formando uma bolha. O resultado é uma bela seqüência de imagens do enchimento de uma caneca de chope. Losasso et. al. [33] criam um novo método SPH para simulação de borrifos d'água e espuma, características que são difíceis de serem capturadas por métodos tradicionais baseados em grades. Esse método é usado para simulação de fluidos *densos* e *difusos*. Fluidos *densos* são grandes volumes de água onde o uso da equação de Navier-Stokes incompressível é mais adequado ao passo que fluidos considerados *difusos*, como os efeitos de espuma, *sprays* e bolhas, são fenômenos incompressíveis e utilizam a versão incompressível das equações de Navier-Stokes.

2.6 Fundamentos do método SPH

O método SPH baseia-se na aproximação local de uma função. Uma função pode possuir diferentes representações em torno de um ponto dado. Essas representações podem ser usadas para aproximação da função em torno daquele ponto. De acordo com [30] a teoria de aproximação de funções usadas nos MM pode ser classificada em métodos de representação diferencial, representação por séries de funções de base polinomial e aproximação por integral.

2.6.1 Aproximação pela função núcleo de uma função

O conceito de representação integral de uma função f usado no método SPH começa com a seguinte identidade:

$$f(\mathbf{x}) = \int_{\Omega} f(\mathbf{x}')\delta(\mathbf{x}-\mathbf{x}')d\mathbf{x}', \quad \forall \mathbf{x} \in \Omega \tag{2.5}$$

onde $\delta(x)$ é chamada função delta de Dirac (o Apêndice A possui maiores detalhes sobre tal função) definida como:

$$\delta(x) = \lim_{\varepsilon \to 0} \begin{cases} 0 & \text{se } x < -\frac{\varepsilon}{2} \\ \frac{1}{\varepsilon} & \text{se } -\frac{\varepsilon}{2} < x < \frac{\varepsilon}{2} \\ 0 & \text{se } x > \frac{\varepsilon}{2} \end{cases} \tag{2.6}$$

e ainda:

$$\int_{-\infty}^{+\infty} \delta(x)dx = 1 \tag{2.7}$$

Como é usada a função delta de Dirac, a representação integral de Equação 2.5 é exata desde que a função $f(\mathbf{x})$ seja contínua no domínio Ω.

Apesar de ser denominada *função*, a delta de Dirac $\delta(x)$ na verdade é uma *função generalizada* ou uma *distribuição*. Isso implica que a integral da Equação 2.5 não é computável e, portanto, é necessário encontrar outro modo para calcular $f(\mathbf{x})$. Para isso, $\delta(x)$ deve ser substituída por uma função que, sob alguma condição, se comporte como a função delta de Dirac. Tal função é chamada *função núcleo* (do inglês *smooth kernel function*) e é representada por $W(\mathbf{x}-\mathbf{x}',h)$ onde h é chamado de *comprimento de suavização* (*smooth lenght*). Reescrevendo a Equação 2.5:

$$< f(\mathbf{x}) >= \int_{\Omega} f(\mathbf{x}')W(\mathbf{x}-\mathbf{x}',h)d\mathbf{x}', \quad \forall \mathbf{x} \in \Omega. \tag{2.8}$$

onde $< \cdot >$ é uma aproximação para uma propriedade sendo que h define o raio de influência dessa aproximação.

Existem três importantes propriedades que a função núcleo deve possuir. São elas:

1. $\int_{\Omega} W(\mathbf{x}-\mathbf{x}',h)d\mathbf{x}' = 1$
2. $\lim_{h\to 0} W(\mathbf{x}-\mathbf{x}',h) = \delta(\mathbf{x}-\mathbf{x}')$
3. $W(\mathbf{x}-\mathbf{x}',h) = 0$ quando $|\mathbf{x}-\mathbf{x}'| > \kappa h$, onde κ é uma constante relacionada ao comprimento de suavização h.

A primeira propriedade é equivalente à propriedade mostrada na Equação 2.7 da função delta de Dirac. A segunda propriedade diz que o comportamento de W se aproxima do comportamento de δ na medida em que h tende a zero. Isso é importante para garantir consistência com a aproximação integral mostrada na Equação 2.5. A terceira propriedade define a área de atuação da função W, chamada de *suporte compacto*. Essa propriedade declara que fora do suporte compacto W vale zero.

2.6.2 Aproximação pela função núcleo da derivada de uma função

A integral da derivada de uma função $f(\mathbf{x})$, denotada pelo divergente $\nabla \cdot f(\mathbf{x})$, é obtida substituindo $f(\mathbf{x})$ por $\nabla \cdot f(\mathbf{x})$ na Equação 2.8:

$$< \nabla \cdot f(\mathbf{x}) >= \int_{\Omega} [\nabla \cdot f(\mathbf{x}')]W(\mathbf{x}-\mathbf{x}',h)d\mathbf{x}', \quad \forall \mathbf{x} \in \Omega. \tag{2.9}$$

Pela regra da cadeia temos que:

$$\begin{aligned} \nabla \cdot [f(\mathbf{x}')W(\mathbf{x}-\mathbf{x}',h)] &= [\nabla \cdot f(\mathbf{x}')]W(\mathbf{x}-\mathbf{x}',h) + f(\mathbf{x}') \cdot [\nabla W(\mathbf{x}-\mathbf{x}',h)] \\ [\nabla \cdot f(\mathbf{x}')]W(\mathbf{x}-\mathbf{x}',h) &= \nabla \cdot [f(\mathbf{x}')W(\mathbf{x}-\mathbf{x}',h)] - f(\mathbf{x}') \cdot [\nabla W(\mathbf{x}-\mathbf{x}',h)]. \end{aligned} \tag{2.10}$$

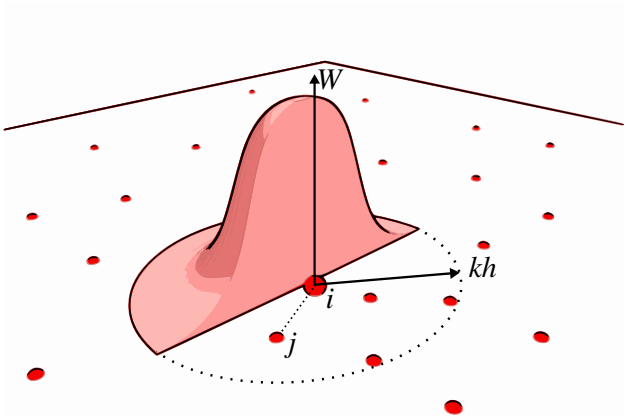


Figura 2.12: O domínio é representado por um conjunto de partículas. No centro é mostrada a partícula i com sua respectiva zona de influência com raio κh. As partículas j formam o conjunto de partículas no interior da zona de influência.

Substituindo a equação anterior na Equação 2.9:

$$< \nabla \cdot f(\mathbf{x}') >= \int_{\Omega} \nabla \cdot [f(\mathbf{x}')W(\mathbf{x}-\mathbf{x}',h)]d\mathbf{x}' - \int_{\Omega} f(\mathbf{x}') \cdot [\nabla W(\mathbf{x}-\mathbf{x}',h)]d\mathbf{x}'. \tag{2.11}$$

Pelo Teorema da Divergência, a integral mais a esquerda do lado direito da equação acima pode ser representada em termos de uma integral de superfície:

$$< \nabla \cdot f(\mathbf{x}) >= \int_{S} f(\mathbf{x}')W(\mathbf{x}-\mathbf{x}',h)\mathbf{n}d\mathbf{x}' - \int_{\Omega} f(\mathbf{x}') \cdot [\nabla W(\mathbf{x}-\mathbf{x}',h)]d\mathbf{x}', \tag{2.12}$$

onde $\mathbf{n}$ é um vetor normal à superfície S. Como W possui suporte compacto, W vale zero na superfície do domínio de integração. Dessa forma, quando o domínio de integração está no interior do domínio do problema, a integral de superfície presente na Equação 2.12 vale zero. Assim:

$$< \nabla \cdot f(\mathbf{x}) >= -\int_{\Omega} f(\mathbf{x}') \cdot [\nabla W(\mathbf{x}-\mathbf{x}',h)]d\mathbf{x}'. \tag{2.13}$$

A equação acima mostra que a aproximação para a derivada de f pode ser escrita em termos da derivada da função núcleo W.

2.6.3 Aproximação por partículas

As integrais presentes nas Equações 2.8 e 2.13 são aproximações de variáveis de campo para resolução de EDPs. Contudo, o domínio do problema é composto por um conjunto discreto de partículas requerendo, portanto a discretização das equações acima por meio de um somatório nas partículas. A Figura 2.12 ilustra o domínio discretizado juntamente com a função núcleo W e o raio de suporte κh.

A integral da distância infinitesimal $d\mathbf{x}'$ representa um pequeno volume de controle de cada

partícula. Escrevendo o volume de cada partícula j:

$$\rho_j = \frac{m_j}{\Delta V_j} \Rightarrow m_j = \Delta V_j \rho_j,$$

onde ρ_j é a densidade da partícula, m_j é sua massa e ΔV_j seu volume e $j = 1, 2, ..., N$, onde N é o número de partículas no interior do domínio de suporte da partícula j. Segue que a Equação 2.8 pode ser discretizada da seguinte maneira:

$$\begin{aligned} < f(\mathbf{x}) > &= \int_\Omega f(\mathbf{x}')W(\mathbf{x}-\mathbf{x}', h)d\mathbf{x}' \\ &= \sum_{j=1}^{N} f(\mathbf{x}_j)W(\mathbf{x}-\mathbf{x}_j, h)\Delta V_j \\ &= \sum_{j=1}^{N} f(\mathbf{x}_j)W(\mathbf{x}-\mathbf{x}_j, h)\frac{m_j}{\rho_j} \\ &= \sum_{j=1}^{N} \frac{m_j}{\rho_j} f(\mathbf{x}_j)W(\mathbf{x}-\mathbf{x}_j, h). \end{aligned}$$

A equação anterior assume a forma:

$$< f(\mathbf{x}) >= \sum_{j=1}^{N} \frac{m_j}{\rho_j} f(\mathbf{x}_j)W(\mathbf{x}-\mathbf{x}_j, h). \tag{2.14}$$

Note que f pode ser qualquer variável de campo, inclusive densidade. A aproximação da densidade ρ de uma partícula pode ser calculada da seguinte maneira utilizando a Equação 2.14:

$$\begin{aligned} < \rho(\mathbf{x}) > &= \sum_{j=1}^{N} \frac{m_j}{\rho_j} \rho(\mathbf{x}_j)W(\mathbf{x}-\mathbf{x}_j, h) \\ &= \sum_{j=1}^{N} \frac{m_j}{\rho_j} \rho_j W(\mathbf{x}-\mathbf{x}_j, h) \\ &= \sum_{j=1}^{N} m_j W(\mathbf{x}-\mathbf{x}_j, h), \end{aligned}$$

que é a soma das massas das partículas vizinhas ponderada por W.

De maneira similar, a Equação 2.13 pode ser discretizada da seguinte maneira:

$$\begin{aligned} < \nabla \cdot f(\mathbf{x}) > &= -\int_\Omega f(\mathbf{x})[\nabla W(\mathbf{x}-\mathbf{x}', h)]d\mathbf{x}' \\ &= -\sum_{j=1}^{N} \frac{m_j}{\rho_j} f(\mathbf{x}_j)\nabla W(\mathbf{x}-\mathbf{x}_j, h), \end{aligned} \tag{2.15}$$

onde o gradiente ∇W na equação refere-se à partícula j. Sabendo que, $\nabla W_{ij} = \frac{\partial W_{ij}}{\partial \mathbf{r}} \frac{\mathbf{r}_{ij}}{\|\mathbf{r}\|}$, onde $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$. As Equações 2.14 e 2.15 são as equações básicas para aproximação de uma

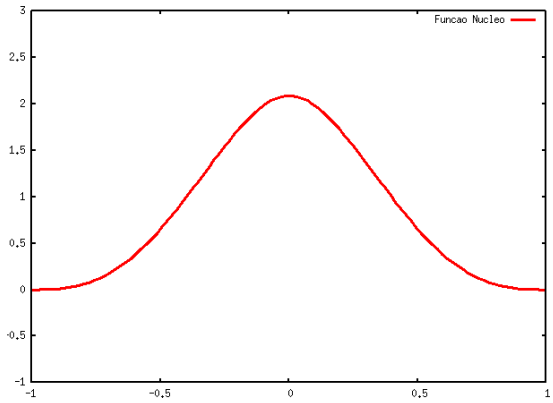


Figura 2.13: Gráfico da função núcleo em formato de sino usada por [34].

função e sua primeira derivada utilizando SPH. Essas equações serão utilizadas para resolução dos termos das Equações de Navier-Stokes.

2.6.4 Função Núcleo

Há diversos tipos de função núcleo na literatura. No artigo de Lucy [34] que introduziu o SPH foi usada a seguinte função:

$$W(|\mathbf{x'}-\mathbf{x}|,h) = W(r,h) = \alpha \begin{cases} (1+3R)(1-R)^3 & \text{se } R \leq 1 \\ 0 & \text{se } R > 1 \end{cases} \tag{2.16}$$

Na equação acima, α vale $\frac{5}{4h}$, $\frac{5}{\pi h^2}$ e $\frac{105}{16\pi h^3}$ se estiver em uma, duas ou três dimensões e $R = \frac{r}{h}$ é a distância relativa entre os pontos. A Figura 2.13 mostra o comportamento dessa função. Monaghan sugere que para encontrar a interpretação física das equações usando SPH é melhor assumir uma função núcleo Gaussiana [30], considerada a Regra de Ouro do SPH. Uma função núcleo como a Gaussiana tem a vantagem de ser suave mesmo para derivadas de alta ordem, característica importante para aplicações que fazem análise de derivadas de alta ordem. Em contrapartida, a Gaussiana não é realmente compacta já que teoricamente nunca atinge zero. No entanto, o decaimento da Gaussiana é suficientemente rápido, aproximando-se de zero rapidamente e assim pode ser considerada compacta.

Existem ainda outros tipos de funções núcleos baseadas em *splines* e polinômios de alta ordem. Cada uma dessas funções núcleos tenta se ajustar a uma classe de problemas para tornar o resultado preciso com desempenho satisfatório. Outro exemplo é a *spline* cúbica com

a seguinte forma:

$$W(|\mathbf{x'}-\mathbf{x}|,h) = W(r,h) = \alpha \begin{cases} \frac{2}{3} - R^2 + \frac{1}{2}R^3 & 0 \leq R < 1 \\ \frac{1}{6}(2-R)^3 & 1 \leq R < 2 \\ 0 & R \geq 2 \end{cases} \tag{2.17}$$

Na equação acima, α vale $\frac{1}{h}$, $\frac{15}{7\pi h^2}$ e $\frac{3}{2\pi h^3}$ se estiver em uma, duas ou três dimensões e $R = \frac{r}{h}$ é a distância relativa entre os pontos. Os valores de α são escolhidos de forma a integral no domínio seja tenha valor um. Essa *spline* cúbica (Figura 2.13) tem sido a função mais usada na literatura SPH uma vez que seu comportamento é semelhante ao da Gaussiana e ainda tem a vantagem de ser realmente compacta.

2.6.5 XSPH

Em sua forma tradicional, o SPH permite o livre movimento das partículas pelo espaço, em outras palavras, não existe colisão entre elas. Isso permite que duas partículas se aproximem de maneira arbitrária em determinados cenários, o que pode levar a inconsistências físicas (partículas com a mesma posição podem possuir diferentes valores das propriedades que carregam, por exemplo, velocidade ou temperatura) ou ainda divisões por zero quando avaliamos a Equação 2.15. O XSPH foi desenvolvido por Monaghan [36] para reduzir o problema de interpenetração de partículas no SPH. A técnica consiste em ponderar a velocidade da *i*-ésima partícula pelas velocidades de seus vizinhos:

$$\mathbf{v}_i = \mathbf{v}_i - \varepsilon \sum_j \frac{m_j}{\rho_j}(\mathbf{v}_i - \mathbf{v}_j)W_{ij}, \tag{2.18}$$

onde ε é uma constante no intervalo $0 \leq \varepsilon \leq 1$. Utilizando o XSPH as partículas tendem a se mover de forma mais organizada, com velocidade mais próxima à velocidade média dos quadros. Essa técnica simples mostrou-se eficiente no combate à interpenetração das partículas [36].

2.7 Representação da superfície

A representação final do escoamento de fluidos é crucial para o realismo da simulação. Nesse ponto, como no caso da simulação física, dois são os caminhos que podem ser trilhados: o primeiro leva a imagens de qualidade excepcional que são amplamente usadas pela indústria de cinema e que, no entanto, requerem muito tempo para ser geradas (*off-line rendering*); o

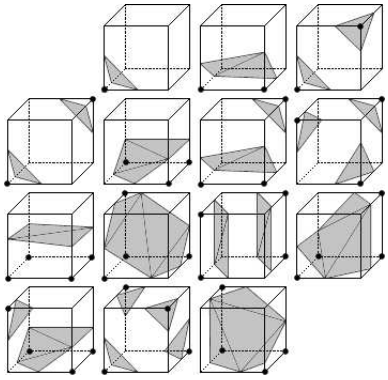


Figura 2.14: A figura acima ilustra os 14 casos possíveis para as combinações de cortes.

outro leva a imagens qualidade gráfica inferior (mas não pobres), muito usada pela indústria de jogos e em realidade virtual e que, por outro lado, são geradas em tempo-real. Obviamente, a escolha entre essas opções depende da necessidade da aplicação e do poder computacional disponível. Contudo, independentemente do caminho, o poder de processamento das placas de vídeo tem aumentado substancialmente nos últimos anos e hoje uma placa como a *NVidia GeForce* 8800 GTX[3] possui, por exemplo, um *texture fill rate* de 33,6 bilhões de *pixels* texturizados por segundo. A evolução das placas aliadas a novos algoritmos e técnicas de *rendering* tem cada vez mais aproximado os dois caminhos outrora tão distantes [60].

A visualização de partículas para aplicações interativas por meio poligonalização da superfície tem atingido bons resultados em termos de desempenho computacional [52, 42]. Nesta seção duas técnicas serão apontadas. A primeira, denominada *marching cubes*, dentre as técnicas que realizam a poligonalização de isosuperfícies, é a abordagem mais utilizada devido a sua natureza simples de implementação e bom desempenho [52]. A outra técnica baseada na técnica de *Multi-level Partition of Unity Implicits* permite a geração de superfícies de forma robusta a partir de grandes nuvens de pontos equipada com normais.

2.7.1 *Marching Cubes*

O *marching cubes* é um método clássico para extração de superfície proposto por Lorensen e Cline [31]. Inicialmente, ele foi concebido para reconstrução de superfícies a partir de um conjunto de fatias de imagens médicas, mas é direta sua extensão para representação a partir de partículas ou nuvem de pontos. O algoritmo tem 3 passos (veja Figura 2.16 para um exemplo bidimensional): 1) primeiramente uma grade é definida em torno do domínio Ω; 2) em seguida, são avaliados todos os pontos $\mathbf{x} \in \Omega$ de maneira a definir os pontos que estão no interior, sobre ou exterior da superfície S de interesse; 3) com base nos pontos selecionados no cubo é possível

[3] *http://www.nvidia.com*

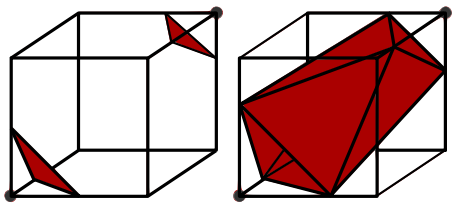


Figura 2.15: A disposição dos vértices do cubo sugere duas possíveis formas para a poligonalização desse cubo.

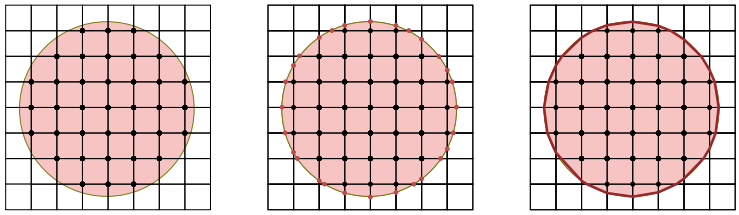


Figura 2.16: Na figura à esquerda, dada uma grade de pontos, é possível encontrar todos os pontos que estão dentro do domínio (pontos pretos). Em seguida, utilizando uma tabela de reconstrução, é possível encontrar a região do cubo que será cortada (pontos vermelhos da figura central). A figura à direita mostra a superfície reconstruída.

reconstruir a superfície utilizando uma tabela de reconstrução e interpolações lineares nas arestas dos cubos. Essa tabela de reconstrução mostra todos os casos possíveis que uma superfície pode cortar o cubo. Em princípio, existem $2^8 = 256$ combinações de vértices possíveis a serem considerados. No entanto, devido à simetria e operações de rotações, é possível reduzir os 256 casos para apenas 14, como mostrado na Figura 2.14.

O *marching cubes* destaca-se pela sua simplicidade e desempenho [52]. A implementação é direta e garante bons resultados em taxas interativas sendo amplamente aceito e utilizado pela comunidade de computação gráfica. Contudo, duas desvantagens da técnica são a ambigüidade presente no método (Figura 2.15) e também a falta de adaptatividade para refinamento de regiões críticas.

Nesse texto, o *marching cubes* fará sempre uso de uma função implícita baseada no suporte compacto h do método SPH. Matematicamente temos:

$$f(\mathbf{x}) = \min_i(||\mathbf{x} - \mathbf{p}_i|| - h) \tag{2.19}$$

2.7.2 Partição da Unidade Multinível

Ohtake et. al. [45] introduziram para a comunidade reconstrução de superfícies a partir de nuvens de pontos um método conhecido como *Multi-level Partition of Unity Implicits* (MPU). O método consiste, como outros métodos implícitos, em encontrar o nível zero de uma função

F definida implicitamente e tem como hipótese uma nuvem de pontos equipadas com normais consistentes. O trabalho se baseia no conceito de *partição da unidade* (explicado logo abaixo), onde aproximações locais $\mathbb{F} = \{f_1, ..., f_n\}$ de uma superfície S são combinadas a fim de definir uma função global F que represente S no domínio Ω. Para realizar a combinação do conjunto $\mathbb{F}$ é preciso primeiramente definir um conjunto de pesos $\Phi = \{w_1, ..., w_n\}$, onde w_i é não-negativos e possui suporte compacto. Um conjunto Φ é dito uma partição da unidade se satisfaz:

$$\sum_{i=0}^{n} w(\mathbf{x}) \equiv 1, \mathbf{x} \in \Omega \tag{2.20}$$

Assim, a função global F definida implicitamente é escrita como:

$$F(\mathbf{x}) = \sum_{i=0}^{n} f_i(\mathbf{x}) w(\mathbf{x}), \;\; \mathbf{x} \in \Omega \tag{2.21}$$

As aproximações locais $\mathbb{F}$ são obtidas em regiões oriundas de subdivisões espaciais sucessivas do domínio de interesse, utilizando uma *octree*. Tais subdivisões ocorrem enquanto determinado critério (qualidade dessas aproximações) não seja satisfeito. Uma das vantagens do método proposto em [45] é a rapidez da geração da superfície implícita mesmo quando grande volume de dados é usado, contudo, o método pode gerar artefatos e superfícies espúrias em suas aproximações locais devido à distribuição dos pontos no interior do domínio local.

Gois et. al. [21] propuseram uma técnica adaptativa em duas vias baseado em [45], contornando suas desvantagens e aumentando significativamente a robustez das aproximações locais. Esse trabalho faz parte do trabalho de doutorado de João Paulo Gois [20] e de mestrado de Valdecir Polizelli-Junior [47], onde são encontradas descrições detalhadas do método além de extensões. As principais contribuições são [21]:

- **Adaptatividade das aproximações locais.** Utilizando polinômios ortogonais, é possível realizar sucessivas aproximações locais sem um alto custo computacional. Polinômios ortogonais de alta ordem podem ser construídos a partir de polinômios de baixa ordem [3, 4].

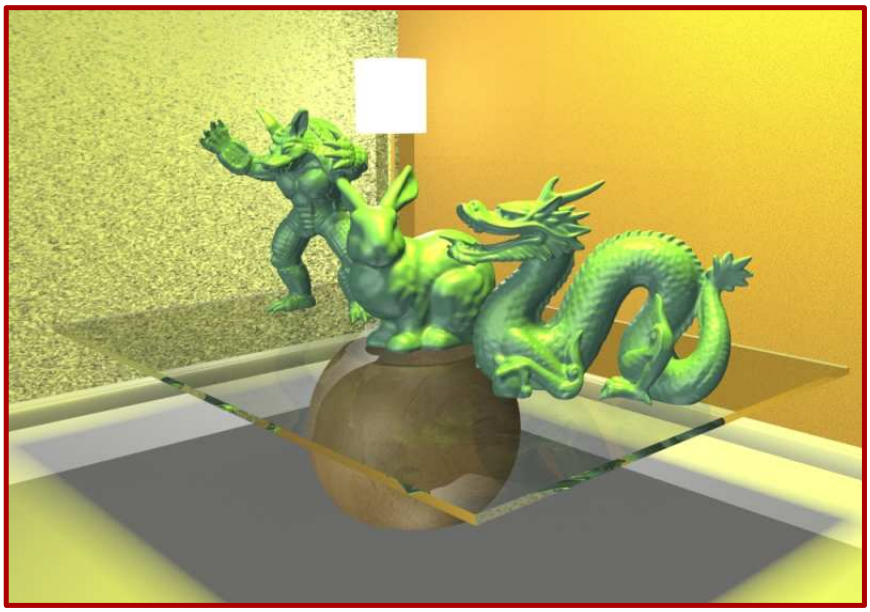
- **Redução das superfícies espúrias**. Por meio de heurísticas, os autores descartam aproximações que oscilam consideravelmente localmente e que geram como conseqüência superfícies espúrias e artefatos.

- **Reconstrução da superfície com garantias topológicas**. O método proposto pelos autores utiliza uma estrutura de dados algébrica denominada J_a^1 [6]. Diferentemente de outros métodos, a triangulação J_a^1 é usada tanto na decomposição espacial para aproximação da superfície implícita quanto para extração da isosuperfície.

A triangulação J_a^1 é uma estrutura de dados algébrica desenvolvida por Castelo et. al. [6] que pode ser estendida para qualquer dimensão e lidar com refinamento arbitrário. Um ponto forte dessa estrutura é a ausência de relações topológicas comumente encontradas em malhas não estruturadas. A navegação pela estrutura é definida por regras algébricas, salvando espaço em memória principal. Uma explicação detalhada da implementação pode ser encontrada em [6, 47].

A Figura 2.17 mostra os resultados da técnica de Gois et. al. para os exemplos clássicos utilizados pela comunidade de computação gráfica. Foram geradas imagens dos modelos de *Stanford* em dois ambientes distintos: uma mesa de vidro com apoio de madeira no canto de uma sala com a presença de um abajur; em uma moldura similar às encontradas em igrejas ou santuários.

O método MPU tem como hipótese uma nuvem de pontos equipadas com normais consistentes. Essa é uma restrição forte quando estamos lidando com um método como o SPH. A nuvem de pontos oriunda dessas simulações tem duas características que vão de encontro ao MPU: os pontos formam um *volume* no espaço, não uma superfície; as normais não são consistentes. Utilizando o SPH é possível escolher somente as partículas da superfície e estimar as normais de cada uma dessas partículas para serem usadas pelo MPU (Seção 4.1.6). Contudo, isso é apenas uma estimativa, o que pode levar a escolha de pontos no interior e/ou normais poucos consistentes e, como resultado, temos uma variação brusca entre quadros consecutivos.



(a) Mesa decorada com os modelos de *Stanford*

(b) O anjo *Lucy* em uma moldura eclesial.

Figura 2.17: Os cenários acima foram modelados e gerados utilizam o *software* livre *Blender*. Os modelos apresentados nas imagens, *Stanford Armadillo Man*, *Stanford Bunny*, *Stanford Dragon* e *Stanford Lucy* foram gerados a partir de uma nuvem de pontos utilizando a técnica apresentada em [21].

3 *Estabilidade numérica*

Sendo uma técnica lagrangeana baseada em partículas, o método SPH fornece bons resultados a um baixo custo computacional. No entanto, o seu uso para a discretização das equações de Navier-Stokes associado com métodos explícitos vem com uma sensibilidade às condições iniciais do problema. Isso se manifesta pelo aumento excessivo das velocidades das partículas durante a simulação, ou seja, aumento da energia envolvida no sistema. Para contornar esse problema, apresentamos uma solução baseada na lei de conservação da energia que restringe o ganho excessivo de energia no sistema.

Neste capítulo a Seção 3.1 mostra a importância do conceito da estabilidade em computação gráfica. As Seções 3.2 e 3.3 lidam com o conceito de consistência, estabilidade e convergência para o clássico método das diferenças finitas (MDF) e algumas considerações são feitas para o método SPH. A Seção 3.4 apresenta o abordagem baseada na restrição do ganho de energia cinética para o método SPH.

3.1 Estabilidade e computação gráfica

Em uma simulação numérica de escoamento de fluido em torno de uma aeronave, o esquema numérico deve ser convergente para que as soluções produzidas sejam confiáveis e apóiem tomadas de decisão. Contudo, quando o alvo é a computação gráfica, uma qualidade altamente desejável de um esquema numérico é a estabilidade. Um esquema incondicionalmente estável proporciona ao usuário da técnica liberdade de utilização, sem preocupações quanto à “explosões” numéricas ou instabilidades que levam a um comportamento não físico, ambos oriundos de parâmetros iniciais mal planejados. O ponto mais importante é que, muitas vezes em uma simulação de fluido, o usuário da técnica, por exemplo, um artista gráfico, espera que o resultado final se comporte com um fluido, independentemente dos parâmetros utilizados.

Um bom exemplo disso é o trabalho de Stam [57]. O autor desenvolveu um novo método para simulação de gases utilizando uma abordagem semi-langrangeana que possui a boa propriedade de ser incondicionalmente estável. O método não é capaz de oferecer precisão numérica

suficiente para aplicações científicas devido à perda de energia, no entanto, essa característica proporciona aos usuários da técnica um bom controle da evolução do fenômeno. Outros trabalhos como [40, 41, 61] tem como destaque a simulação numérica de comportamentos físicos de maneira estável.

O método SPH tradicional é sensível às condições iniciais de forma que, variando apenas o passo de tempo Δt, por exemplo, podemos simular desde um fluido até gases quando usado métodos explícitos. Isso foi observado também com os outros parâmetros do SPH. Métodos implícitos [37, 27] tendem a ser computacionalmente mais caros contudo permitem uma maior liberdade para o passo de tempo. A próxima seção apresenta uma formulação baseada na limitação do ganho de energia que aumenta a estabilidade do sistema de partículas sem comprometer o desempenho.

3.2 Consistência, estabilidade e convergência no MDF

Os conceitos de consistência, estabilidade e convergência são de grande importância em DFC. Ao resolver uma EDP, as soluções apresentadas utilizando MM, MDF ou outro método devem ser compatíveis com a EDP original. Assim, precisamos determinar sob quais condições a solução obtida pela discretização é representativa do problema original. Como veremos, a consistência das equações, estabilidade e convergência do método empregado são fundamentais para determinar essas condições [9].

3.2.1 Consistência

A *consistência* de um método numérico relaciona-se à discretização das EDPs. Ao realizar a discretização de um conjunto de EDPs é preciso ter garantias de que as equações sendo resolvidas pelo modelo discretizado possuem soluções equivalentes às das equações originais. Para ilustrar esse conceito tomemos a equação do calor em um domínio unidimensional $\frac{\partial T}{\partial t} - \alpha \frac{\partial^2 T}{\partial x^2} = 0$. Essa equação pode ser discretizada utilizando MDF como [53]:

$$\frac{T_i^{t+1} - T_i^t}{\Delta t} - \alpha \frac{T_{i+1}^t - 2T_i^t + T_{i-1}^t}{\Delta x^2} = 0. \tag{3.1}$$

Uma forma utilizada para verificar a consistência das equações é realizando a expansão dos seus termos em série de Taylor:

$$T_{i\pm1}^{t} = T_i^t \pm \frac{\partial T^t}{\partial x}\Delta x + \frac{1}{2!}\frac{\partial^2 T^t}{\partial x^2}\Delta x^2 \pm \frac{1}{3!}\frac{\partial^3 T}{\partial x^3}\Delta x^3 + O(\Delta x^4) \quad (3.2)$$

$$T_i^{t+1} = T_i^t + \frac{\partial T^t}{\partial t}\Delta t + \frac{1}{2!}\frac{\partial^2 T^t}{\partial t^2}\Delta t^2 + \frac{1}{3!}\frac{\partial^3 T^t}{\partial x^3}\Delta t^3 + O(\Delta t^4), \quad (3.3)$$

e substituindo 3.2 e 3.3 em 3.1:

$$\frac{\partial T}{\partial t} - \alpha\frac{\partial^2 T}{\partial x^2} = -\frac{1}{2}\frac{\partial^2 T}{\partial t^2}\Delta t + \alpha\frac{1}{12}\frac{\partial^4 T}{\partial x^4}\Delta x^2 + O(\Delta t^2) + O(\Delta x^4). \quad (3.4)$$

Ao lado esquerdo da Equação 3.4 é a equação do calor original enquanto o lado direito representa a diferença entre as equações originais e a discretização, ou seja, o erro oriundo do MDF. Portanto, analisando o lado direito de 3.4, vemos que a discretização é de primeira ordem no tempo e segunda ordem no espaço. Além disso, podemos concluir que à medida que o $\Delta t \to 0$ e $\Delta x \to 0$, a Equação3.4 aproxima-se da EDP original, ou seja, o método é consistente.

3.2.2 Estabilidade

O conceito de *estabilidade* está relacionado ao crescimento dos diversos tipos de erros inerentes à simulação numérica (arredondamento, discretização, erros da iteração). No entanto, apesar de estarem presentes, esses erros não representam à física do problema e devem ser controlados.

Em problemas *transientes*, ou seja, problemas que evoluem com o tempo, a estabilidade de um método garante que a solução é *limitada*. Em contrapartida, estabilidade de métodos iterativos de solução de sistema linear garante que a solução encontrada em iterações sucessivas se aproxima cada vez mais da solução real problema.

Um algoritmo é considerado *incondicionalmente estável* se os erros envolvidos não crescem de maneira ilimitada independentemente das condições iniciais. Caso não seja incondicionalmente estável, se houver um conjunto de condições iniciais para os quais os erros não crescem sem limites o algoritmo é dito *condicionalmente estável*. Caso contrário o algoritmo é dito *instável*.

Algoritmos que são condicionalmente estáveis possuem condições de estabilidade. Para problemas transientes isso pode significar um limite superior para Δt enquanto para problemas de solução de sistema lineares isso pode requerer que a matriz possua alguma propriedade, por exemplo, $\| A \|_\infty \leq 1$. Essas condições são atingidas por meio de uma análise de estabilidade do método. A análise de estabilidade de Von Neumann [49] é uma das técnicas mais usadas para

analise de estabilidade de métodos numéricos. Ela se baseia na expansão das discretizações em séries de Fourier e posterior análise do crescimento dos termos e conseqüentemente dos erros envolvidos.

3.2.3 Convergência

Finalmente, o conceito de *convergência* deriva de consistência e estabilidade. O teorema da equivalência de Lax [51] mostra que dada condições iniciais apropriadas e assumindo que o esquema numérico é consistente, estabilidade é condição necessária e suficiente para a convergência do método. Em um esquema muito simples é possível escrever:

$$\textit{consistência} + \textit{estabilidade} = \textit{convergência}$$

3.3 Consistência utilizando o método SPH

Atualmente, na literatura sobre SPH há poucos trabalhos que fazem uma análise aprofundada sobre sua consistência, estabilidade e convergência. Dos trabalhos que promovem uma análise mais detalhada sob essa óptica, freqüentemente eles o fazem em cenários simplificados e unidimensional [35, 18, 11, 59, 38].

Recentemente o uso da técnica em outras aplicações não relacionadas à astrofísica propiciou um aumento na comunidade de pesquisadores contribuindo para seu amadurecimento. Apesar disso, ainda existem várias formulações e adaptações feitas nas implementações, além do elevado número de parâmetros envolvidos que contam com intuição e experiência dos pesquisadores, tornando a técnica subjetiva em vários aspectos [2]. O desenvolvimento de um arcabouço matemático para o aprofundamento da técnica traz uma maneira formal de avaliar e evoluir a técnica, como ocorre, por exemplo, em técnicas tradicionais como MDF, onde esse arcabouço é bastante evoluído e bem aceito pela comunidade. Esta seção versa sobre o conceito de consistência para o SPH.

3.3.1 Consistência

O método SPH é oriundo de dois tipos de aproximações: *aproximação da função núcleo* e *aproximação de partículas*. A primeira lida com a aproximação da função δ por uma função núcleo W enquanto a última lida com a discretização do domínio. Em uma análise de consistência, ambas as aproximações têm influência nos erros envolvidos e devem ser levadas em consideração.

3.3.2 Ordem polinomial máxima

[30] apresenta o conceito de consistência ligado à representação exata de um polinômio de grau p. Segundo esse raciocínio, para possuir consistência de ordem 0, a aproximação pela função núcleo deve satisfazer $f(\mathbf{x}) = c$:

$$f(\mathbf{x}) = \int_{\Omega} cW(\mathbf{x}-\mathbf{x}',h)d\mathbf{x}' = c. \tag{3.5}$$

Da mesma maneira, para obter a aproximação de primeira ordem:

$$\begin{aligned} f(\mathbf{x}) &= \int_{\Omega} (c_0 - c_1\mathbf{x})W(\mathbf{x}-\mathbf{x}',h)d\mathbf{x}' && (3.6)\\ &= c_0 + \int_{\Omega} c_1\mathbf{x}W(\mathbf{x}-\mathbf{x}',h)d\mathbf{x}', && (3.7)\end{aligned}$$

ou seja, a aproximação de primeira ordem mostrada anteriormente será válida se:

$$\int_{\Omega} c_1\mathbf{x}W(\mathbf{x}-\mathbf{x}',h)d\mathbf{x}' = c_1\mathbf{x}, \tag{3.8}$$

ou

$$\int_{\Omega} \mathbf{x}W(\mathbf{x}-\mathbf{x}',h)d\mathbf{x}' = \mathbf{x}. \tag{3.9}$$

Generalizando para o grau p:

$$\int_{\Omega} \mathbf{x}^k W(\mathbf{x}-\mathbf{x}',h)d\mathbf{x}' = \mathbf{x}^k, \text{ para } 0 \leq k \leq p, \tag{3.10}$$

e então temos uma condição para a consistência da aproximação pelo kernel.

A consistência da aproximação por partícula é uma extensão direta do método contínuo, dada por

$$\sum_j \mathbf{x}^k W(\mathbf{x}-\mathbf{x}',h)V_j = \mathbf{x}^k, \text{ para } 0 \leq k \leq p, \tag{3.11}$$

onde $V_j = m_j/\rho_j$. Uma dificuldade para manter a consistência é a deficiência de partículas na borda do domínio [38] já que em suas proximidades é natural haver mais partículas concentradas na região interior de Ω. A distribuição não uniforme e a função núcleo também afetam a consistência do SPH.

3.3.3 Ordem do erro da discretização

Apesar da Equação 3.11 fornecer as condições necessárias para se ter consistência para um polinômio de grau p, ela não deixa de forma explícita os erros envolvidos na aproximação. No parágrafo seguinte é mostrada uma forma alternativa para verificar a consistência do SPH baseada na tese de [18].

Definição 1 (*Consistência*)**.** *[18] Seja $f(\boldsymbol{x})$ uma função suficientemente suave em um domínio D. Seja também $\tau(f) = g(f) - g^h(f)$, onde g é uma EDP e g^h sua discretização espacial ($\Delta \boldsymbol{x}_\alpha$, α denota x, y ou z para problemas tridimensionais) e temporal (Δt). Então a discretização é consistente se e somente se:*

$$\| \tau(f) \| \rightarrow 0 \quad se \quad \Delta \boldsymbol{x}_\alpha, \Delta t \rightarrow 0. \tag{3.12}$$

Aproximação da função núcleo

Realizando a expansão de Taylor na Equação 2.8 e fazendo $\mathbf{u} = \mathbf{x}' - \mathbf{x}$:

$$\begin{aligned} < f(\mathbf{x}) > &= \int_\Omega f(\mathbf{x}+\mathbf{u}) W(-\mathbf{u}, h) d\mathbf{u} && (3.13) \\ &= \int_\Omega [f(\mathbf{x}) + \nabla f(\mathbf{x})^T \mathbf{u} + \frac{1}{2} \mathbf{u}^T H f(\xi_0) \mathbf{u}] W(-\mathbf{u}, h) d\mathbf{u}, && (3.14) \end{aligned}$$

onde $Hf(\xi)$ é a hessiana de f. A integral acima pode ser reescrita como:

$$\begin{aligned} < f(\mathbf{x}) > &= f(\mathbf{x}) \int_\Omega W(-\mathbf{u}, h) d\mathbf{u} + \nabla f(\mathbf{x})^T \int_\Omega \mathbf{u} W(-\mathbf{u}, h) d\mathbf{u} + \\ &\quad + \frac{1}{2} \int_\Omega \mathbf{u}^T H f(\xi_0) \mathbf{u} W(-\mathbf{u}, h) d\mathbf{u}. && (3.15) \end{aligned}$$

Lembrando que $\int_\Omega W(-\mathbf{u}, h) d\mathbf{u} = 1$ e que $\int_\Omega \mathbf{u} W(-\mathbf{u}, h) d\mathbf{u} = 0$, já que W é uma função par, podemos reescrever a equação anterior como:

$$\begin{aligned} < f(\mathbf{x}) > &= f(\mathbf{x}) + \frac{1}{2} \int_\Omega \mathbf{u}^T H f(\xi_0) \mathbf{u} W(-\mathbf{u}, h) d\mathbf{u} && (3.16) \\ < f(\mathbf{x}) > &= f(\mathbf{x}) + E_k(f, \mathbf{x}). && (3.17) \end{aligned}$$

O objetivo agora é encontrar um limitante superior para $|E_k|$ e dessa forma limitar $< f(\mathbf{x}) >$. Para isso, vamos definir:

$$D^2 = \sum_{i=1}^{d} \sum_{j=1}^{d} \frac{\partial^2}{\partial x_i \partial x_j}, \tag{3.18}$$

onde d é a dimensão do problema e:

$$e_k = \frac{\kappa^2}{2} \sup_{\xi \in \Omega} |D^2 f(\xi)|. \tag{3.19}$$

Com as definições anteriores podemos escrever

$$|\mathbf{u}^T H f(\xi)\mathbf{u}| \leq (\kappa h)^2 |D^2 f(\xi_0)| \leq (\kappa h)^2 \frac{2e_k}{\kappa^2} = 2h^2 e_k. \tag{3.20}$$

Substituindo 3.20 em $|E_k|$:

$$\begin{aligned} |E_k(f,\mathbf{x})| &= |\frac{1}{2}\int_\Omega \mathbf{u}^T H f(\xi_0)\mathbf{u} W(-\mathbf{u},h)d\mathbf{u}| && (3.21)\\ &\leq \frac{1}{2}\int_\Omega |\mathbf{u}^T H f(\xi_0)\mathbf{u}||W(-\mathbf{u},h)|d\mathbf{u} && (3.22)\\ &\leq \frac{1}{2}\int_\Omega |2h^2 e_k||W(-\mathbf{u},h)|d\mathbf{u} && (3.23)\\ &\leq |h^2 e_k|\int_\Omega |W(-\mathbf{u},h)|d\mathbf{u} = h^2 e_k, && (3.24) \end{aligned}$$

onde $\int_\Omega |W(-\mathbf{u},h)|d\mathbf{u} = 1$ já que $W(-\mathbf{u},h) \geq 0$ para todos os pontos dentro do suporte κh.

Aproximação de partículas

Veremos agora o erro causado pela aproximação por partículas utilizando o SPH. O raciocínio desenvolvido é valido em duas dimensões mas pode ser estendido para dimensões maiores. Para o cálculo desse erro vamos utilizar o método mais simples para integração numérica, a regra do retângulo (caso unidimensional):

$$\int_a^b f(x)dx = f(a)(b-c), + \frac{1}{2}(b-c)^2 f'(\xi_i).$$

Para o caso bidimensional, devemos considerar o processo de integração em um plano *xy*, como mostrado a seguir. Dado uma partícula p_i e suas partículas vizinhas p_j, crie uma *triangulação de Delaunay* [54] cujos vértices são as essas partículas. Crie também pontos no centróide de cada triângulo formando o *diagrama de Voronoi*. Para cada partícula, crie um polígono convexo em torno dessa partícula ligando os pontos do diagrama de Voronoi da partícula dada. A Figura 3.1 ilustra o procedimento.

A integração de uma função *g* sobre o domínio é dada por:

$$\iint_\Omega g(x,y)dxdy = \sum_j^{N_k} \iint_{P_j} g(x,y)dxdy, \tag{3.25}$$

onde N_k é o número de polígonos formados na malha. Para um polígono P_j podemos escrever a integral sobre sua superfície como:

$$\iint_{P_j} g(x,y)dxdy = \sum_k^{N_t} \iint_{T_k} g(x,y)dxdy, \tag{3.26}$$

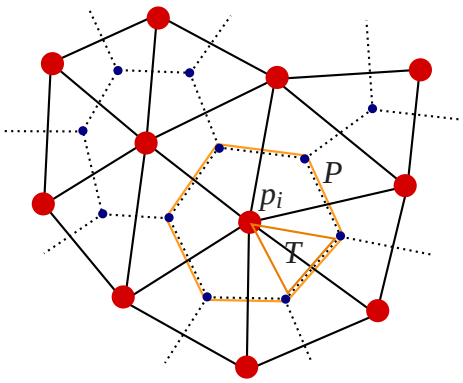


Figura 3.1: O polígono convexo P é destacado na figura assim como um triangulo T. O polígono é formado pelas arestas de Voronoi enquanto cada triângulo pela ligação entre a partícula p_i e os vértices de Voronoi.

onde N_t é o número de triângulos T presente em um polígono. Agora resta então calcular o valor da aproximação de g sobre um triângulo T_k:

$$\iint_{T_k} g(x,y)dxdy = g(x_j,y_j)a_k + e_k, \tag{3.27}$$

onde a_k é a área de T_k e e_k é o erro cometido na aproximação e é definido por $e_k = [\frac{\partial g}{\partial x}a_k\delta x + \frac{\partial g}{\partial y}a_k\delta y] = a_k\nabla g\cdot\delta\mathbf{x}$, onde $\delta\mathbf{x} = (\delta x,\delta y)$, e_k nada mais é do que o erro oriundo da regra do trapézio para o caso tridimensional. Substituindo a equação anterior em 3.26:

$$\iint_{P_i} g(x,y)dxdy = \sum_j^{N_t}[g(x_j,y_j)a_k + e_k] = g(x_j,y_j)\sum_j^{N_t}(a_k+e_k) = g(x_i,y_i)A_i + E_i. \tag{3.28}$$

Finalmente, substituindo 3.28 em 3.25 temos:

$$\iint_{\Omega} g(x,y)dxdy = \sum_j^{N_k}\iint_{P_j} g(x,y)dxdy = \sum_j^{N_k} A_j[g(x_j,y_j)+E_j]. \tag{3.29}$$

Suponha que $g(x,y) = f(x,y)W(\parallel(x_i,y_i)-(x,y)\parallel,h)$ e lembrando que $A_j = \frac{m_j}{\rho_j}$:

$$\begin{aligned}\iint_{\Omega} g(x,y)dxdy &= \sum_j^{N_k}\frac{m_j}{\rho_j}f(x_j,y_j)W(\parallel(x_i,y_i)-(x_j,y_j)\parallel,h)+E_p &(3.30)\\ &= <f(\mathbf{x}_i)> + E_p(f,\mathbf{x}_i). &(3.31)\end{aligned}$$

Queremos agora encontrar um limitante superior para o erro $|E_p|$:

$$|E_i| \leq N_k \max_j |N_{t_j}\max_k |e_k||. \tag{3.32}$$

Assumindo $N_k N_{t_j} a_k \sim \pi(\kappa h)^2 \leq (2\kappa h)^2$ e que $e_p = 4\kappa^2 sup_{\xi \in T} \nabla g(\xi)$.

$$|E_p| \leq \max_j |N_k N_{t_j} a_k \parallel \nabla g \parallel \parallel \delta \mathbf{x} \parallel | \leq \parallel \delta \mathbf{x} \parallel h^2 e_p \tag{3.33}$$

A ordem de convergência das aproximações pela função núcleo e por partículas são usados para verificar a consistência do SPH. Primeiramente é preciso definir:

$$\text{Operador identidade} \quad : \quad If = f \tag{3.34}$$

$$\text{Operador derivada} \quad : \quad Pf = \frac{\partial f}{\partial x} \tag{3.35}$$

$$\text{Aprox. função núcleo} \quad : \quad Kf(\mathbf{x}) = \int_\Omega f(\mathbf{x}') W(\mathbf{x} - \mathbf{x}', h) d\mathbf{x}' \tag{3.36}$$

$$\text{Aprox. por partícula} \quad : \quad Sf(\mathbf{x}) = \sum_{j=1}^{N} \frac{m_j}{\rho_j} f(\mathbf{x}_j) W(\mathbf{x} - \mathbf{x}_j, h) \tag{3.37}$$

Para que uma discretização seja consistente com a equação original é preciso que a distância entre ambas seja reduzida a zero quando no limite do refinamento do domínio discretizado (Definição 1). Portanto:

$$\parallel If - Sf \parallel_\infty = \parallel If - Kf + Kf - Su \parallel_\infty \leq \parallel If - Kf \parallel_\infty + \parallel Kf - Su \parallel_\infty \tag{3.38}$$

Resolvendo o lado direito da inequação:

$$\parallel If - Kf \parallel_\infty \quad = \quad \parallel f - f - E_k \parallel_\infty = \parallel E_k \parallel_\infty \leq h^2 e_k \tag{3.39}$$

$$\parallel Kf - Sf \parallel_\infty \quad = \quad \parallel Kf - Kf - E_p \parallel_\infty = \parallel E_p \parallel_\infty \leq \parallel \delta \mathbf{x} \parallel h^2 e_p \tag{3.40}$$

e:

$$\parallel If - Sf \parallel_\infty \leq h^2 e_k + \parallel \delta \mathbf{x} \parallel h^2 e_p \tag{3.41}$$

Vemos que $\parallel If - Sf \parallel_\infty \rightarrow 0$ quando $h \rightarrow 0$, portanto o método é consistente.

3.4 Restrição do ganho de energia

A lei da conservação da energia enuncia que a energia total em um sistema isolado não sofre alterações. Dessa forma, não é possível criar/aumentar a energia total de um sistema isolado, apesar de existir a possibilidade de transformar um tipo de energia em outras formas de energia por meio de trabalho. Se considerarmos a ação de forças não-conservativas, como o atrito, então a tendência da energia total desse sistema diminuir com o tempo devido à ação dessa força.

Em uma simulação numérica, o aumento indiscriminado do campo de velocidade leva a um

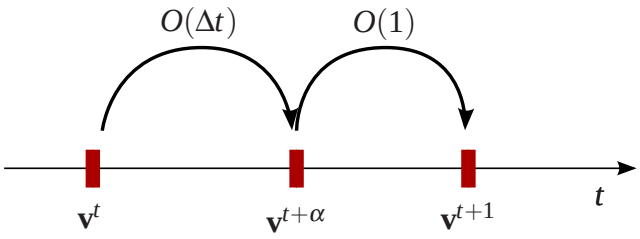


Figura 3.2: No método de dois passos, o primeiro passo possui ordem $O(\Delta t)$ enquanto o segundo é uma interpolação constante de ordem $O(1)$ do vetor velocidade.

grande ganho de energia no sistema. Uma maneira simples de evitar que esse campo cresça além das expectativas é limitando o ganho de energia. Müller et. al. [41] mostram que para um simples sistema massa-mola envolvendo duas partículas em 1D a deformação da mola a cada passo de tempo pode aumentar a energia envolvida no sistema se um método explícito como o método de Euler Explícito for utilizado. Isso ocorre porque o avanço temporal é feito de forma a não considerar os limites máximos de deformação da mola em um instante de tempo, ou seja, ela pode ser deformada além de seu deslocamento inicial o que fornece energia ao sistema.

No SPH, a liberdade de movimento das partículas associada à integração numérica pode levar a um ganho significativo de energia no sistema. Isso se reflete no movimento desordenado das partículas e no crescimento da energia total do sistema. Nesta seção é apresentado um método que faz controle da energia total para dar maior estabilidade às simulações numéricas envolvendo o SPH.

3.4.1 Método de Euler de dois passos

O modelo de restrição do ganho de energia baseia-se em uma pequena variação do método de Euler explícito para integração do campo de velocidade. Considere:

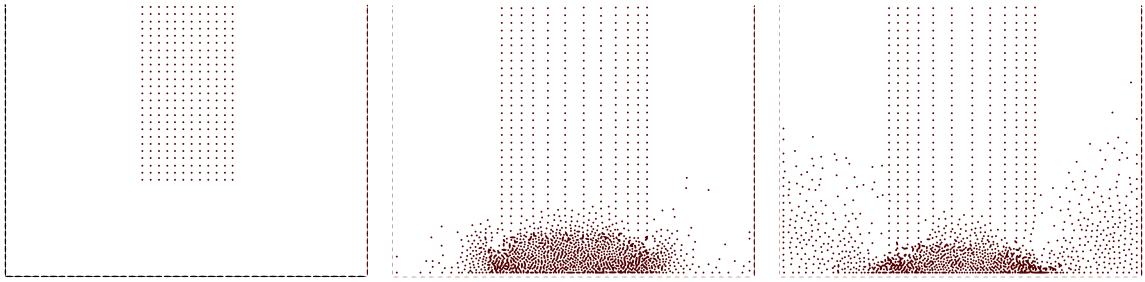
$$\mathbf{v}(t+\alpha\Delta t) = \mathbf{v}(t) + \alpha\Delta t\frac{d\mathbf{v}}{dt} + O((\alpha\Delta t)^2), \tag{3.42}$$

onde $0 \leq \alpha \leq 1$. Temos que uma aproximação para $\frac{d\mathbf{v}}{dt}$ pode ser expressa por:

$$\mathbf{f} = \frac{d\mathbf{v}}{dt} = \frac{\mathbf{v}(t+\alpha\Delta t) - \mathbf{v}(t)}{\alpha\Delta t} + O(\alpha\Delta t), \tag{3.43}$$

onde $\mathbf{f}$ é o vetor aceleração. A equação anterior é uma aproximação de primeira ordem para aceleração $\mathbf{f}$ agindo sobre um objeto. Portanto, para encontrar o vetor velocidade no tempo $\mathbf{v}(t+\alpha\Delta t) = \mathbf{v}^{t+\alpha}$ a partir de uma força $\mathbf{f}(t) = \mathbf{f}^t$ que age no tempo t fazemos:

$$\mathbf{v}^{t+\alpha} = \mathbf{v}^t + \alpha\Delta t\mathbf{f}^t + O(\alpha\Delta t). \tag{3.44}$$



(a) Coluna de fluido inicial (t_0) (b) Sem a preservação da energia (c) Com a preservação da energia

Figura 3.3: A figura central e à direita mostram um instante t de uma simulação com mesmos parâmetros iniciais. A simulação mostrada ao centro não distribui a energia perdida da interação entre partículas enquanto a simulação à direita preserva mente a energia.

Aplicações que evoluem no tempo podem fazê-lo em intervalos fixos ou variados. Isso é feito para controlar a estabilidade do método numérico empregado. Uma implementação típica do SPH utiliza passos de tempos fixos com um integrador temporal como o *leap-frog* (Seção 4.4.2). Contudo, implementações com passo de tempo variável [36] são corriqueiras na literatura. Algoritmos de integração numérica utilizam avanço temporal adaptativo para lidar com o crescimento da força, viscosidade e condições CFL e assim obter estabilidade numérica. Em geral, o passo de tempo é escolhido como sendo o valor mais restritivo encontrado, ou seja, o menor valor possível. Uma desvantagem dessa abordagem é a utilização do mesmo passo de tempo para todas as partículas, comprometendo o desempenho da simulação como um todo.

Nesse trabalho o avanço temporal é feito com um passo de tempo fixo e igual a Δt. Portanto, a integração temporal apresentada em 3.44 está incompleta. Para corrigir esse problema a velocidade $\mathbf{v}$ precisa ser avançada em $(1-\alpha)\Delta t$. Ao invés de utilizar um método de resolução de EDOs para calcular $\mathbf{v}^{t+1}$ a partir de $\mathbf{v}^{t+\alpha}$ optamos nesse texto por realizar uma interpolação constante $O(1)$, ou seja, simplesmente adotar $\mathbf{v}^{t+1} = \mathbf{v}^{t+\alpha} + O(1)$ como ilustrado na Figura 3.2. Do ponto de vista de precisão numérica o erro cometido por essa abordagem é muito alto, no entanto, como será apresentado próxima seção, isso permitirá alcançar uma estabilidade maior do SPH. A posição $\mathbf{x}^{t+1}$ das partículas evolui de maneira tradicional:

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \Delta t \mathbf{v}^{t+1}. \tag{3.45}$$

3.4.2 Evitando a ganho de energia

Durante uma simulação, as partículas estão sujeitas às energias *potencial gravitacional* (E_p), *cinética* (E_k) e *interna* (e), ou seja, controlando o crescimento dessas formas de energia é possível tornar a simulação mais estável. A primeira relaciona-se à energia "armazenada" em

um sistema que pode a qualquer momento ser transformada em outra forma de energia por meio de um trabalho. A segunda é a oriunda de uma velocidade atuante em uma partícula. A terceira energia é a energia resultante da interação entre partículas e também permanece "armazenada" no sistema.

Em um sistema de partículas, a energia potencial pode ser considerada como a energia potencial gravitacional, energia que é resultado das forças gravitacionais atuantes no sistema e definida como:

$$E_p = \| \mathbf{p} \| h = m \| \mathbf{g} \| h = mgh, \tag{3.46}$$

onde m é a massa de cada partícula, g é a norma do vetor gravidade $\mathbf{g}$ e h é a altura da partícula. A energia cinética é definida como:

$$E_k = \frac{1}{2} m \| \mathbf{v} \|^2, \tag{3.47}$$

onde $\mathbf{v}$ é a velocidade da partícula.

O objetivo é evitar que a resolução da EDO envolvida na simulação não faça um escalonamento "às cegas" da força, aumentando a energia do sistema. Portanto, podemos escrever a energia total do sistema como

$$E_p^{t+1} + E_k^{t+1} \leq E_p^t + E_k^t = E_{total}^t \tag{3.48}$$

ou seja, a energia total do sistema diminui (devido à ação de forças viscosas, atrito, etc.) ou se mantém. Na verdade, isso pode ser feito para cada partícula i individualmente:

$$E_{p,i}^{t+1} + E_{k,i}^{t+1} \leq E_{p,i}^t + E_{k,i}^t = E_{total,i}^t \tag{3.49}$$

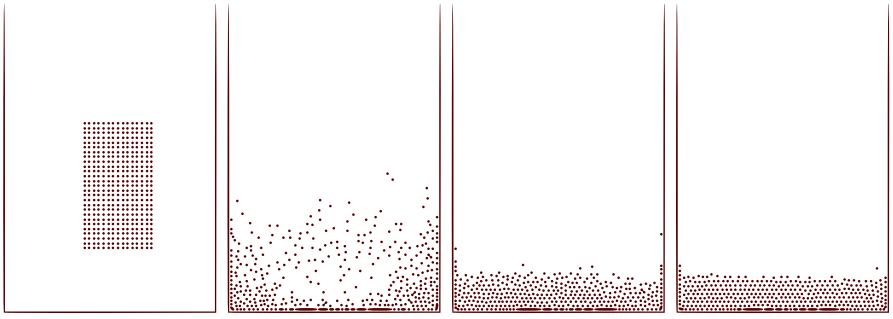
$$m_i g h_i^{t+1} + \frac{1}{2} m_i \| \mathbf{v}_i^{t+1} \|^2 \leq E_{total,i}^t \tag{3.50}$$

Neste ponto, vamos utilizar o método de Euler de dois passos para a velocidade apresentado na Seção 3.4.1:

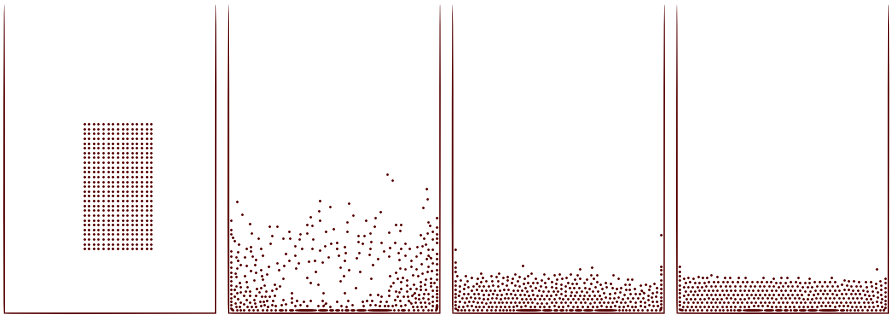
$$\mathbf{v}_i^{t+1} = \mathbf{v}_i^t + \Delta t \frac{\alpha \mathbf{f}_i^t}{\rho_i} + \Delta t \mathbf{f}_{ext,i} \tag{3.51}$$

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \Delta t v_i^{t+1}, \tag{3.52}$$

onde $\mathbf{f}_i$ é a força agindo na partícula i e $\mathbf{f}_{ext}$ são as forças externas atuantes. O parâmetro α, restrito à $0 \leq \alpha \leq 1$, será usado para o correto escalonamento do vetor aceleração de forma que



(a) Sem limitação de energia



(b) Com limitação de energia

Figura 3.4: As figuras mostram diferentes instantes de tempo e com mesmos parâmetros para a simulação de um bloco de fluido. Os quadros gerados são os de número 1, 200, 400 e 600. Nesse cenário, os parâmetros são adequados e portanto os valores de $\alpha \rightarrow 1$ e por assim as simulações são idênticas.

a energia total em uma partícula não extrapole. Substituindo 3.51 e 3.52 em 3.50:

$$m_i \overline{\mathbf{g}}(\mathbf{x}_i^{t+1} - \mathbf{x'}_i^{t+1}) + \frac{1}{2} m_i \overline{\mathbf{v}}_i^{t+1} \mathbf{v}_i^{t+1} \quad \leq \quad E_{total,i}^{t}, \tag{3.53}$$

onde o vetor $\overline{\mathbf{u}}$ é transposto de $\mathbf{u}$ e $\mathbf{x'}_i$ é a projeção da posição $\mathbf{x}_i$ da partícula i na superfície onde é considerada o "chão" da cena. A Equação 3.53 é na verdade uma simples equação de segundo grau em α que limita o crescimento da energia interna do sistema. Resolvendo-a temos o valor de α para que não se crie energia quando 3.51 e 3.52 forem resolvidas. Note que a Equação 3.53 não leva em conta a energia interna e, discutida mais à frente.

3.4.3 Valores de α

A Equação 3.53 nada mais é do que uma equação de segundo grau em α. Sendo uma equação desse tipo, e lembrando que os valores de interesse de α variam no intervalo $[0,1]$, temos as seguintes combinações:

- $\alpha > 1$. Assumimos o valor de $\alpha = 1$ e a simulação segue normalmente.
- $0 \leq \alpha \leq 1$. Para qualquer passo de tempo maior que $\alpha \Delta t$ o sistema ganha energia.

- $\alpha < 0$. Nesse cenário, é preciso retroceder no tempo para evitar o ganho de energia. Assumimos $\alpha = 0$ e o sistema ganha energia.
- $\nexists\alpha$. Nesse caso o sistema necessariamente ganha energia.

A equação de segundo grau nos fornece duas soluções, se elas existem. A preferência é dada sempre para solução cujo valor é mais próximo de 1.

3.4.4 Preservando efetivamente energia

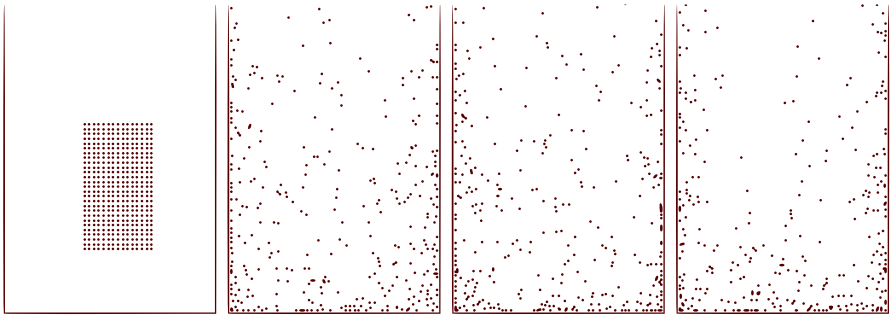
Mesmo utilizando o mecanismo anterior, ainda se perde muita energia durante a simulação. Uma razão para esse fenômeno é a interação entre uma partícula *i* com outras em sua vizinhança, além da natural perda de energia devido às colisões com as paredes do domínio. Para que a energia total do sistema não se reduza drasticamente durante a interação entre as partículas, a Equação 3.53 também é usada para computar a "sobra" de energia de uma partícula, transferindo-a para uma próxima. Isso diminui a taxa de decaimento da energia total do sistema fazendo com que a simulação tenha aspecto mais realista. A Figura 3.3 mostra um instante de uma mesma simulação transferindo energia entre partículas e sem sua transferência.

A transferência é dada da seguinte maneira: quando a integração de uma partícula *i* permitir um valor de $\alpha \geq 1.0$ então houve uma sobra de energia, ou seja, a partícula poderia ter atingido velocidade e/ou altura maior (ganho de energia potencial ou cinética). Essa energia restante é passada integralmente para a próxima partícula para que ela se movimente o máximo possível na tentativa de fazer com que $\alpha \rightarrow 1$.

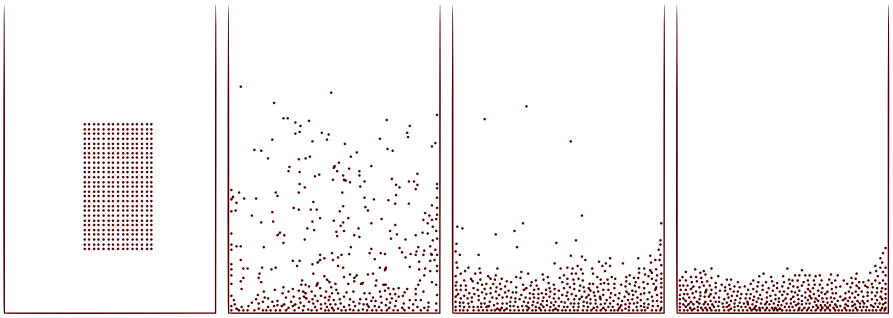
3.4.5 Resultados e vantagens

Os resultados são mostrados de uma forma qualitativa, avaliando a estabilidade dos resultados dos quadros intermediários, e quantitativa, calculando-se a energia envolvida no sistema durante a simulação. As simulações a seguir fazem uso do XSPH (Seção 2.6.5) para suavizar o problema de interpenetração de partículas com $\varepsilon = 0,4$. O sistema perde energia por meio de colisões com as paredes do recipiente (99% da energia cinética da partícula é perdida durante uma colisão).

Estabilidade e convergência O ponto forte dessa abordagem é a aumentar a robustez do método, favorecendo a convergência por meio da dissipação da energia. A Figura 3.4 mostra o resultado de duas simulações bem sucedidas com parâmetros idênticos utilizando o método



(a) Sem limitação de energia

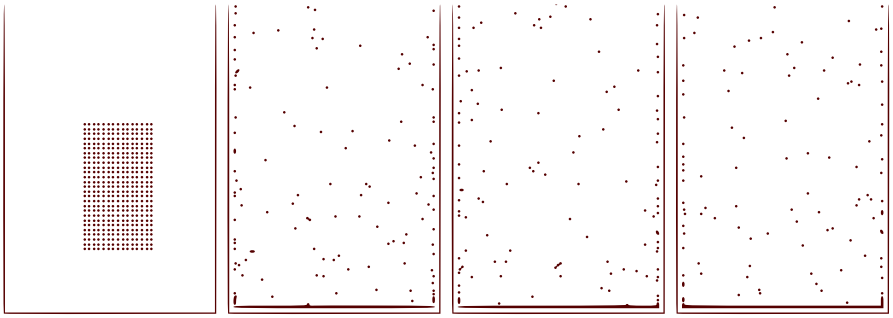


(b) Com limitação de energia

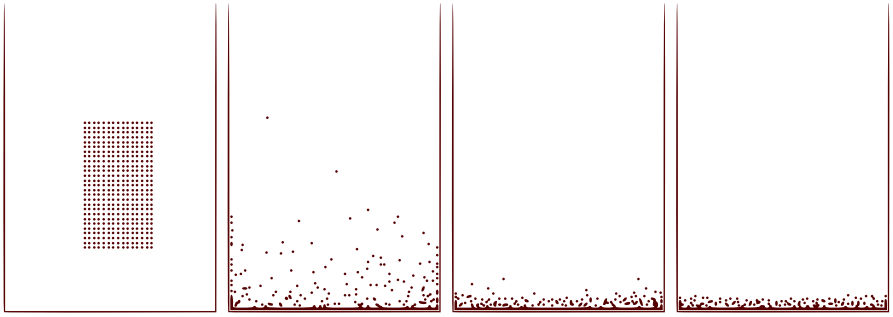
Figura 3.5: As figuras mostram diferentes instantes de tempo e com mesmos parâmetros para a simulação de um bloco de fluido. Os quadros gerados são os de número 1, 200, 400 e 600. O parâmetro $m = 1,5 \times 10^{-3}$ foi levado a uma condição de estresse.

de Euler e Euler de 2 passos. Os parâmetros para esse problema são: $h = 3,2 \times 10^{-3}$, $dt = 4 \times 10^{-3}$, $\rho_0 = 0$, $m = 1,125$, $\mu = 1$ e $w = 1,2$, onde w representa o comprimento da caixa onde o bloco de fluido se encontra. O número de partículas no sistema é 400. Podemos ver pela Figura 3.4 que quando os parâmetros são adequados, os valores de α tendem sempre a ser iguais a 1 e portanto as simulações ficam idênticas.

A Figura 3.5(a) mostra o resultado da simulação anterior (Figura 3.4) levando a massa m a uma condição extrema. O valor da massa foi alterado de maneira brusca levando a um crescimento não natural da energia quando simulado da maneira tradicional. A Figura 3.5(b) mostra o resultado quando é realizada a limitação da energia. A Figura 3.6 ilustra a mesma idéia levando o parâmetro Δt a uma variação brusca. Neste caso, repare que a coluna inicial de fluido é convertida numa fina camada. O tamanho do passo de tempo associado à perda de energia da colisão é responsável por esse fenômeno. Aumentando o valor de Δt, a gravidade gera uma velocidade suficientemente forte para colidir, a cada iteração, as partículas com o fundo do recipiente. Se o coeficiente de restituição for pequeno (nos testes usamos 1%), isso faz com que as partículas “colem” plano ao fundo.



(a) Sem limitação de energia

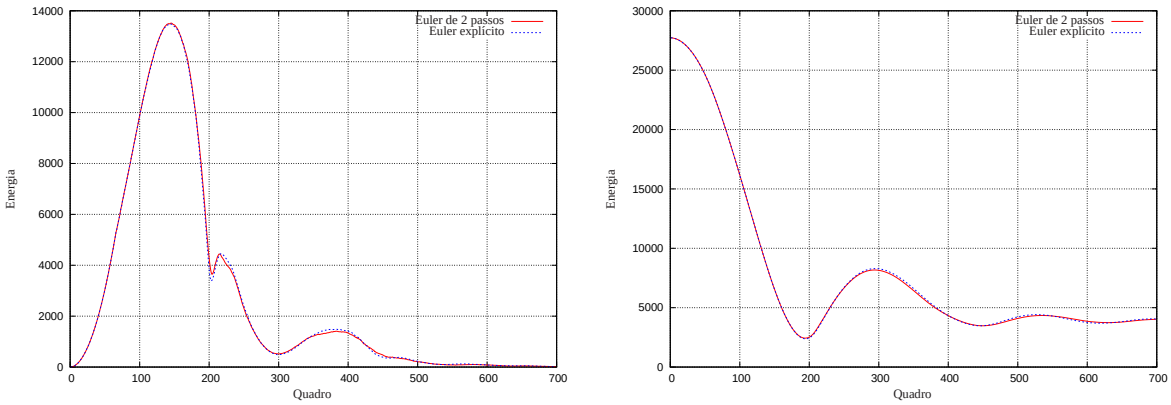


(b) Com limitação de energia

Figura 3.6: As figuras mostram diferentes instantes de tempo para a simulação de um bloco de fluido. Os quadros gerados são os de número 1, 100, 200 e 300. Os parâmetros são todos semelhantes. O parâmetro $t = 0,02$ foi levado a uma condição de estresse.

Evolução da energia Como visto nas figuras anteriores, um integrador temporal explícito como o método de Euler promove, sob determinadas condições, acréscimo de energia ao sistema que posteriormente é convertida em velocidade. A Figura 3.7 mostra a evolução da energia para o cenário ilustrado pela Figura 3.4 (exceto pelo número de partículas: 1400). Segundo ela, os parâmetros iniciais utilizados são suficientemente adequados e não fornecem energia ao sistema sendo essa a razão dos gráficos para ambos os métodos, Euler e Euler de 2 passos, apresentarem evoluções muito semelhantes ao longo do tempo. Nesse caso, durante boa parte do tempo de simulação, incluindo o choque com o fundo do recipiente, o valor de α encontrado é muito próximo de 1. Durante a outra metade, os valores permanecem acima de 0.5, na média.

A Figura 3.8 mostra a evolução da energia para o caso ilustrado pela Figura 3.5 (novamente, o único parâmetro que difere é o número de partículas: 1400). Esse caso em que um parâmetro é levado a uma condição de estresse mostra claramente a quantidade excessiva de energia envolvida quando utilizado o método de Euler explícito. A Figura 3.9 apresenta o caso ilustrado pela Figura 3.6. Nesse último exemplo, em que o bloco de fluido foi reduzido a uma fina camada, a energia cinética das partículas é reduzida para aproximadamente $\frac{1}{4}$ da energia inicial nos primeiros 100 quadros. Novamente, isso ocorre pelos constantes choques com o fundo do recipiente, causando perda de energia. A Figura 3.9 mostra a evolução da energia cinética e potencial para o caso em que um pequeno bloco de fluido é solto sobre uma piscina de fluidos. Em



(a) Energia cinética (b) Energia Potencial

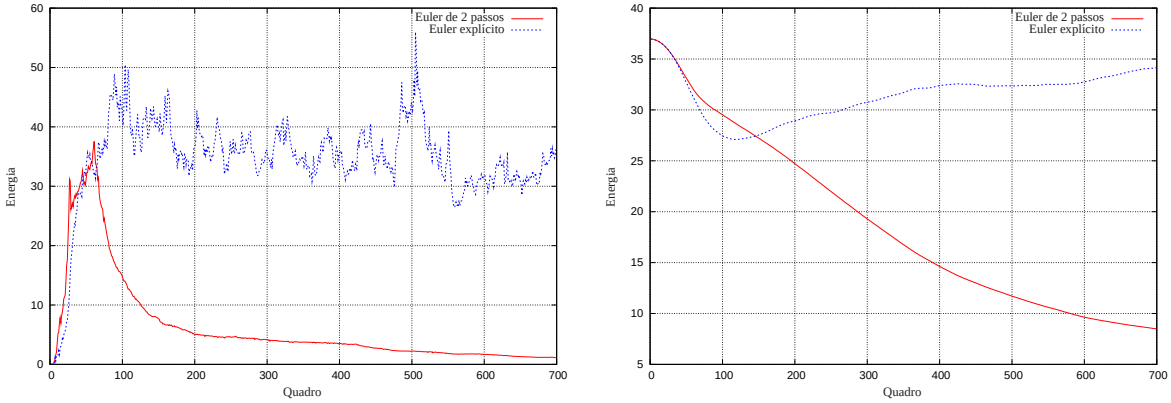
Figura 3.7: Evolução da energia cinética (esquerda) e potencial (direita) durante um intervalo de tempo (medido pelos quadros da simulação) para a simulação ilustrada pela Figura 3.4 (exceto pelo número de partículas: 1400). As condições iniciais do problema não geram energia durante a simulação e por isso a curva para ambos os métodos, Euler e Euler de 2 passos, são semelhantes.

todos as figuras é possível observar o forte decaimento da energia e a estabilização do processo.

Os resultados mostram que, para a técnica tradicional, o crescimento da energia cinética total é grande, mas não ilimitado. Em [59] os autores verificam comportamento semelhante das partículas quando elas são submetidas a um regime de tensão. Um *stress* positivo significa uma variação negativa do volume de partículas o que causaria instabilidade. O autor justifica que, nesse caso, as partículas tendem a se agrupar e não causar uma explosão numérica.

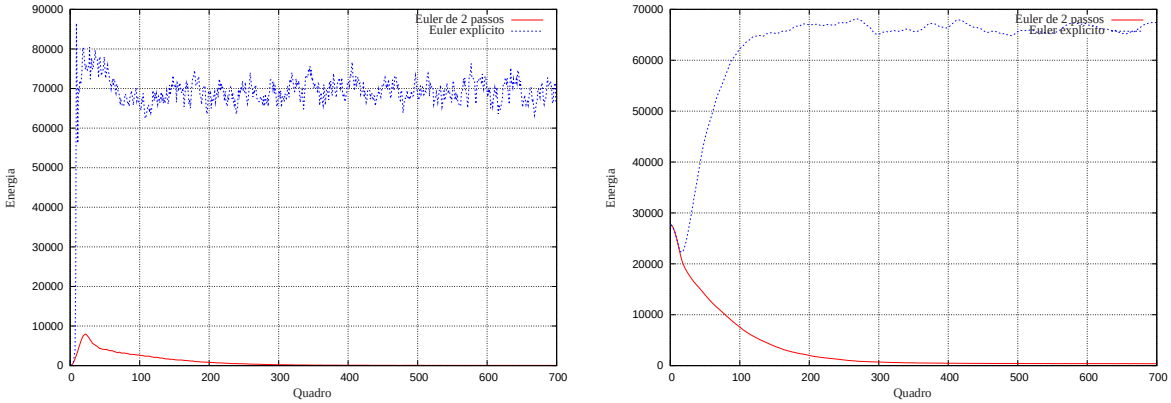
Evolução dos valores de alfa A Figura 3.11 ilustra a variação dos valores dos valores de α no tempo para os casos apresentados nas Figura 3.4 e 3.6. Os valores de α para a Figura 3.11(a) permanecem próximos de 1 mesmo após o início das colisões. A partir daí, algumas partículas adquirem velocidade muito alta e então os valores de α diminuem e a média adquire valor próximo de 0.5. Quando um pequeno passo de tempo é utilizado, esperamos que os valores médios de α sejam muito próximos de 1 e de fato isso ocorre. Quando o passo de tempo cresce substancialmente, Figura 3.11(b), esses valores diminuem drasticamente na tentativa de preservar a energia total.

Implementação A idéia apresentada é simples suficiente para ser facilmente inserida no em códigos já existentes de simuladores de fluidos. Outros integradores temporais podem ser utilizados sem grandes complicações para evitar a criação de energia no sistema.



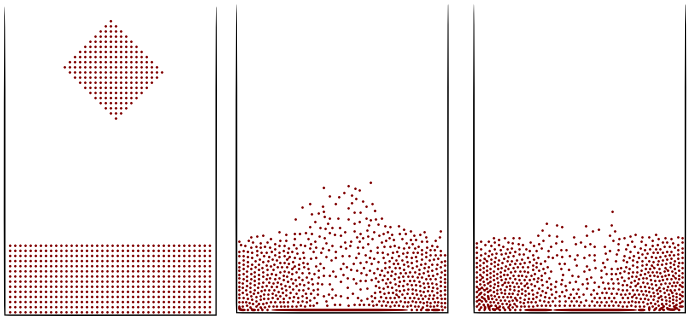
(a) Energia cinética (b) Energia Potencial

Figura 3.8: Evolução da energia cinética (esquerda) e potencial (direita) durante um intervalo de tempo (medido pelos quadros da simulação) para a simulação ilustrada pela Figura 3.5 (exceto pelo número de partículas: 1400). O ganho de energia é significativo sob essas condições.

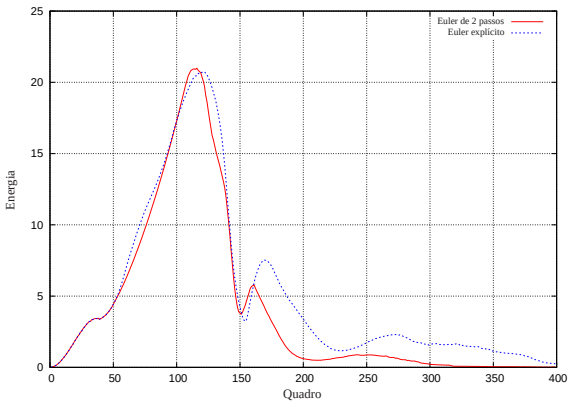


(a) Energia cinética (b) Energia Potencial

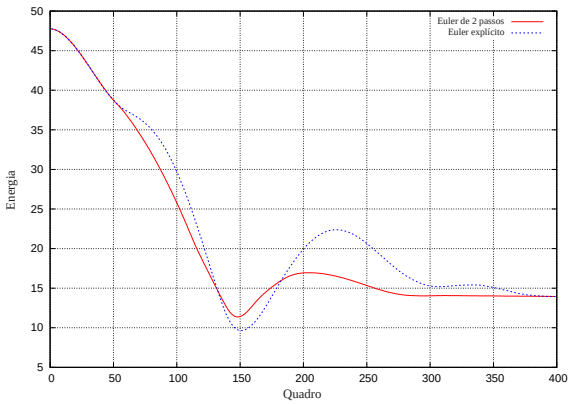
Figura 3.9: Evolução da energia cinética (esquerda) e potencial (direita) durante um intervalo de tempo (medido pelos quadros da simulação) para a simulação ilustrada pela Figura 3.6 (exceto pelo número de partículas: 1400). O ganho de energia é significativo sob essas condições.



(a) Estado inicial (b) Sem preservação (c) Com preservação

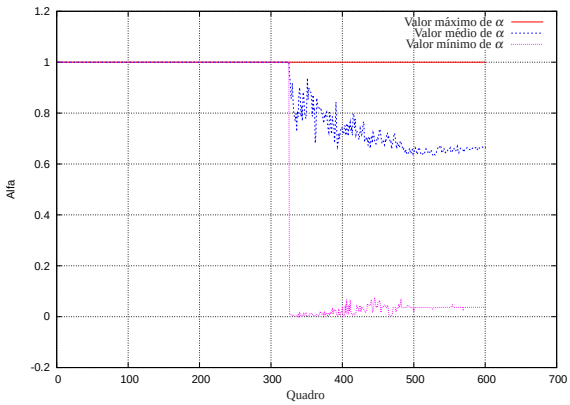


(d) Energia cinética

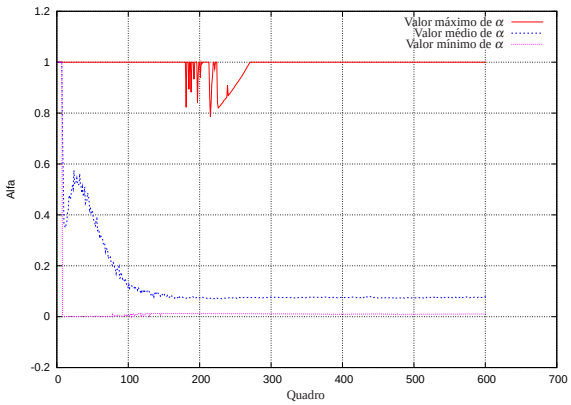


(e) Energia Potencial

Figura 3.10: Evolução da energia para cenário diferente. (a) configuração inicial do sistema. (b) Resultado da simulação utilizando método de Euler explícito. (c) Resultado da simulação utilizando o método de Euler de 2 passos. (d) e (e) Gráfico da evolução das energias cinética e potencial para ambos os integradores temporais.



(a) Passo de tempo pequeno



(b) Passo de tempo grande

Figura 3.11: (a) Caso ilustrado pela Figura 3.4. (b) Caso ilustrado pela Figura 3.6. Quando o passo de tempo é suficientemente pequeno, o valor médio de α é maior que $0,5$ (a), mostrando que pouco precisa ser feito para evitar o ganho de energia. Caso contrário, seu valor pode ficar abaixo de $0,2$.

Custo computacional Não há aumento significativo do custo computacional. Esse método de 2 passos tem como custo extra por iteração apenas o esforço para encontrar raízes de uma equação de segundo grau, custo muito baixo comparado com operações de busca, por exemplo. Como conseqüência, a complexidade computacional do método SPH não se altera.

Independência da função núcleo A restrição da energia independe dos tipos de função núcleo sendo utilizadas. Apenas o resultado final da simulação contabiliza para o cálculo da velocidade de posição.

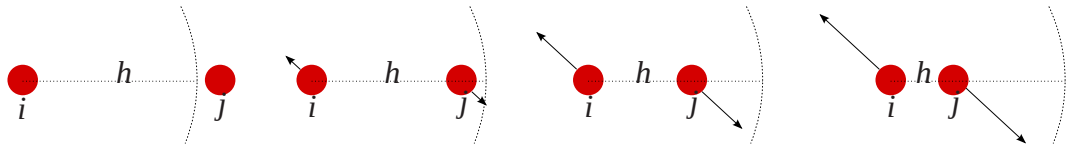
3.4.6 Desvantagens

A abordagem que envolve a controle da energia total envolvida no sistema permite que o sistema convirja mais facilmente. No entanto, novos problemas e desvantagens surgem e precisam ser levados em consideração para o bom uso da técnica.

integração numérica O integrador temporal utilizado (Seção 3.4.1) possui no pior caso uma aproximação de ordem $O(1)$. Para a computação gráfica, um integrador temporal que produza resultados visualmente plausíveis, mesmo de ordem $O(1)$, é suficiente. No entanto, técnicas que possuem alta ordem de convergência sem sacrificar o desempenho são bem-vindas [25]. Essa é uma das razões pela qual o método *leap-frog* (Seção 4.4.2) é bastante utilizado na literatura de SPH para animação por computador. O *leap-frog* possui ordem de convergência quadrática $O(\Delta t^2)$ sendo eficiente do ponto de vista do desempenho porque a avaliação da força é realizada somente uma vez.

Colisão O processo de colisão descrito na Seção 4.3 pode aumentar a energia total do sistema. Isso ocorre porque o processo de colisão não leva em consideração um possível ganho de energia potencial para realizar a reflexão do vetor velocidade. Assim, a resposta de colisão deve ter um tratamento especial para lidar com o ganho de energia. Um outra solução é utilizar métodos de colisão baseado em forças de interação com os limites do domínio e não relacionados a critérios puramente geométricos.

Energia interna A restrição da energia gera problemas ao movimento das partículas. Primeiramente, note que a posição inicial das partículas tem grande influência no comportamento do fluido. A Figura 3.12(a) mostra duas partículas em repouso, com j fora do raio de suporte h de i. Nesse cenário, a energia das partículas se resume à energia potencial gravitacional E_p. Já as



(a) $e_0 = 0$ (b) $e_1 > e_0$ (c) $e_2 > e_1$ (d) $e_3 > e_2$

Figura 3.12: Possíveis configurações iniciais para o sistema de partículas. À medida que as partículas se aproximam a energia interna e_k aumenta e se manifesta na forma de uma força agindo sobre elas.

Figuras 3.12(b) a 3.12(d) mostram duas partículas em repouso (estado inicial) que tem mútua influência. Nesse novo cenário, além da energia potencial gravitacional inerente ao sistema, ainda existe uma energia interna causada pela proximidade das partículas.

O modelo descrito na Seção 3.4 não leva em conta essa energia interna. Como resultado, podemos observar um decaimento rápido da energia (a Figura 3.4 ilustra a diferença de movimento entre a abordagem tradicional e a apresentada). Nessas condições o fluido simulado tem forte aspecto viscoso, mesmo na ausência de viscosidade. Contudo, é possível amenizar esse problema levando em conta a energia interna do sistema por meio da equação de conservação de energia (Equação 2.3). Ela pode ser discretizada juntamente com as outras equações obtendo o valor da energia interna e_i para cada partícula i em cada instante de tempo acrescentando-a ao lado direito de 3.53.

O processo de Euler de 2 passos move as partículas de acordo com a força aplicada para sua posição preocupando-se apenas com o ganho de energia potencial gravitacional e cinética. Assim que as partículas se aproximam a energia interna pode aumentar de tal forma a criar energia no sistema, levando o sistema à instabilidade. Novamente, é preciso um tratamento especial para lidar com o aumento da energia interna de tal forma que o ganho pela ela não ultrapasse a energia inicial no sistema.

4 Desenvolvimento

Apesar da simplicidade e rapidez de implementação do SPH, aqueles que se aventuram pelo seu caminho acidentado freqüentemente tropeçam na sensibilidade aos parâmetros iniciais. Durante a implementação do método há uma estranha sensação de que existe algo errado com as simulações já que, ao invés de fluidos, os resultados lembram uma explosão. Os mais insistentes percebem que a técnica é sensível a esses parâmetros e que o seu ajuste deve ser feito paulatinamente, nenhuma novidade do ponto de vista da pesquisa científica. Em contrapartida, outros tipos de usuários gostariam que a técnica fosse mais robusta, permitindo um intervalo de escolha maior para esses parâmetros. Os resultados apresentados nesta seção fazem uso da metodologia apresentada na Seção 3.4 para aumentar a robustez do método deixando-a menos sensível às condições iniciais.

Este capítulo lida com as questões de implementação de um simulador de escoamento de fluidos baseado no método SPH. Para tanto, ele abrange a discretização das equações que modelam o comportamento do fluido (Seção 4.1), um algoritmo eficiente para busca espacial (Seção 4.2), detecção e resposta de colisão (Seção 4.3), avanço temporal (Seção 4.4) e finalmente os resultados obtidos (Seção 4.6).

4.1 Discretização das Equações de Navier-Stokes

A discretização usada nesse trabalho segue as idéias de [39]. Inicialmente, em prol do desempenho, são realizadas algumas simplificações nas equações apresentadas na Seção 2.4.1. O primeiro ponto a ser observado é que a equação da conservação de massa, Equação 2.1, é automaticamente satisfeita utilizando-se um método baseado em partículas que não cria ou remove partículas dinamicamente, ou seja, o número de partículas permanece constante e, portanto a massa do sistema não varia. Logo essa equação pode ser completamente omitida na simulação. Já na equação da quantidade de movimento, Equação 2.2, pode ser feita uma simplificação relevante. Primeiramente devemos observar que a equação da conservação de momento pode ser

reescrita da seguinte maneira:

$$\frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v}\cdot\nabla)\mathbf{v} = \nu\nabla^2\mathbf{v} - \frac{1}{\rho}\nabla p + \mathbf{f} \tag{4.1}$$

$$\rho\frac{D\mathbf{v}}{Dt} = \mu\nabla^2\mathbf{v} - \nabla p + \rho\mathbf{f} \tag{4.2}$$

Derivadas do tipo $\frac{D\phi}{Dt} = \frac{\partial\phi}{\partial t} + (\mathbf{v}\cdot\nabla)\phi$, são chamadas *derivada total* e possuem significado físico importante. Ela mede a taxa de variação de uma propriedade ϕ qualquer (no caso das Equações de Navier-Stokes $\phi = \mathbf{v}$) em relação ao tempo, termo $\frac{\partial\phi}{\partial t}$, e ao espaço, termo $(\mathbf{v}\cdot\nabla)\phi$. Na simulação utilizando o SPH, as partículas se movem junto com o domínio e a velocidade resultante é calculada dependendo da posição da partícula. Dessa maneira o termo não-linear $(\mathbf{v}\cdot\nabla)\mathbf{v}$ não precisa ser calculado e a simulação requer apenas que a evolução da derivada total no tempo.

Dessa maneira, para calcular a nova posição $\mathbf{x}$ da partícula é necessário encontrar sua velocidade. A velocidade para cada partícula é dada pelo lado esquerdo Equação 4.2. O lado direito dessa equação é o campo densidade de forças resultante agindo sobre uma partícula, $\mathbf{f}_R = \mu\nabla^2\mathbf{v} - \nabla p + \rho\mathbf{f}$. Uma vez encontrada a força resultante, segue que:

$$\begin{aligned} \mathbf{a} &= \frac{d\mathbf{v}}{dt} = \ddot{\mathbf{r}} = \frac{\mathbf{f}_R}{\rho} \\ \mathbf{v} &= \frac{d\mathbf{r}}{dt} = \dot{\mathbf{r}} \end{aligned}$$

onde $\mathbf{a}$ é a aceleração do fluido e $\mathbf{r}$ é a posição da partícula. Para cada partícula deve ser calculado o valor de $\mathbf{f}_R$ e após a velocidade $\mathbf{v}$ e finalmente a nova posição $\mathbf{r}$. Resta agora calcular cada termo força do lado direito da Equação 4.2 para encontrar a resultante $\mathbf{f}_R$. A discretização das equações será feitas conforme as Equações 2.14 e 2.15.

4.1.1 Funções núcleo

As funções núcleo utilizadas são [39]:

$$W_{poly6}(\mathbf{r},h) = \begin{cases} (h^2 - ||\mathbf{r}||^2)^3 & \text{se } 0 \le ||\mathbf{r}|| \le h \\ 0 & \text{se } ||\mathbf{r}|| > h \end{cases} \tag{4.3}$$

$$W_{spiky}(\mathbf{r},h) = \frac{15}{\pi h^6}\begin{cases} (h - ||\mathbf{r}||)^3 & \text{se } 0 \le ||\mathbf{r}|| \le h \\ 0 & \text{se } ||\mathbf{r}|| > h \end{cases} \tag{4.4}$$

$$W_{viscosity}(\mathbf{r},h) = \frac{15}{2\pi h^3}\begin{cases} \frac{-||\mathbf{r}||^3}{2h^3} + \frac{||\mathbf{r}||^2}{h^2} + \frac{h}{2||r||} - 1 & \text{se } 0 \leq ||\mathbf{r}|| \leq h \\ 0 & \text{se } ||\mathbf{r}|| > h \end{cases} \tag{4.5}$$

A função núcleo W_{poly6} é usada para todas as aproximações excetuando a aproximação da pressão, que utiliza W_{spiky}, e da viscosidade, que utiliza $W_{viscosity}$.

4.1.2 Forças devido à ação externa

As forças externas são as mais simples de serem adicionadas ao sistema. Em geral tais forças são consideradas *forças de campo*, ou seja, são forças que agem sobre todas as partículas com mesma intensidade. Nesse caso, nenhuma discretização é necessária para o calculo da força. Portanto:

$$\mathbf{f}_i^{externa} = \rho_i \mathbf{f} \tag{4.6}$$

4.1.3 Forças devido à viscosidade

A força oriunda da viscosidade vem do termo $\mu\nabla^2\mathbf{v}$. Nesse caso, o campo vetorial que precisa ser discretizado é dado por $\nabla^2\mathbf{v}$. Assim, para cada partícula a força resultante é dada por:

$$f_i^{viscosidade} = \mu\nabla^2\mathbf{v}(\mathbf{r}_i) = -\mu\sum_j \frac{m_j}{\rho_j}(\mathbf{v}_j - \mathbf{v}_i)\nabla^2 W(\mathbf{r}_i - \mathbf{r}_j, h)$$

4.1.4 Forças devido à pressão

A força oriunda da pressão vem do termo $-\nabla p$. Nesse caso, o campo vetorial que precisa ser discretizado é dado por ∇p. Assim, para cada partícula, a força resultante é dada por:

$$f_i^{pressao} = -\nabla p(\mathbf{r}_i) = -\mu\sum_j \frac{m_j}{\rho_j}\frac{1}{2}(p_j + p_i)\nabla W(\mathbf{r}_i - \mathbf{r}_j, h)$$

O cálculo da pressão envolve um procedimento adicional. Em geral, não são fornecidas condições iniciais a respeito da pressão em torno de cada partícula e apenas a densidade. Para contornar esse problema [10] trata o fluido como um gás perfeito, portanto é possível extrair uma relação entre a pressão e a densidade em cada partícula. Essa relação vem da equação dos gases perfeitos e é dada por:

$$p = k(\rho - \rho_0) \tag{4.7}$$

onde p é a pressão, k é a uma constante do gás, ρ a densidade e ρ_0 uma densidade de referência. Dessa forma é possível calcular os valores da força devido à pressão em todas as partículas.

4.1.5 Energia interna

A evolução da energia interna e é utilizada nesse trabalho para evitar o crescimento da velocidade além das expectativas. Para isso a Equação 2.3 é discretizada da seguinte forma [30]:

$$\begin{aligned}\frac{\partial e_i}{\partial t} = \frac{1}{2}\sum_j m_j \frac{(p_i + p_j)}{\rho_i \rho_j}(\mathbf{v}_i - \mathbf{v}_j)\nabla W_{ij} + \frac{\mu}{2\rho_i}(\quad & \varepsilon_{xx,i}\varepsilon_{xx,i} + \varepsilon_{xy,i}\varepsilon_{xy,i} + \varepsilon_{xz,i}\varepsilon_{xz,i} + \\ + \quad & \varepsilon_{yx,i}\varepsilon_{yx,i} + \varepsilon_{yy,i}\varepsilon_{yy,i} + \varepsilon_{yz,i}\varepsilon_{yz,i} + \\ + \quad & \varepsilon_{zx,i}\varepsilon_{zx,i} + \varepsilon_{zy,i}\varepsilon_{zy,i} + \varepsilon_{zz,i}\varepsilon_{zz,i}), \end{aligned} \tag{4.8}$$

onde e_i é a energia interna e $\varepsilon_{\alpha\beta}$, $\alpha, \beta \in \{x, y, z\}$, é dado por:

$$\varepsilon_{\alpha\beta} = \sum_j \frac{m_j}{\rho_j}(v_{\beta,j} - v_{\beta,i})\frac{\partial W_{ij}}{\partial \alpha} + \sum_j \frac{m_j}{\rho_j}(v_{\alpha,j} - v_{\alpha,i})\frac{\partial W_{ij}}{\partial \beta} - \frac{2}{3}(\sum_j \frac{m_j}{\rho_j}(\mathbf{v}_j - \mathbf{v}_i)\cdot\nabla W_{ij})\delta^{\alpha\beta}, \tag{4.9}$$

onde $\mathbf{v}_i = (v_{x,i}, v_{y,i}, v_{z,i}) = (v_x, v_y, v_z)_i$ e

$$\delta^{\alpha\beta} = \begin{cases} 1 & \text{se } \alpha = \beta \\ 0 & \text{caso contrário} \end{cases}$$

A resolução de 4.8 pode ser feita por qualquer um dos tradicionais métodos de integração numérica. Nessa dissertação, Euler explícito é utilizado.

4.1.6 Partículas da superfície e normais

Podemos estimar as partículas que estão na superfície, assim como suas normais, utilizando um campo de cor c (*color field*) que vale 1 na posição da partícula e 0 em todo resto [39]. A aproximação de c em uma partícula i é dada por:

$$c_i = \sum_j m_j \frac{1}{\rho_j} W_{ij} \tag{4.10}$$

O gradiente da função cor gera as normais na superfície:

$$\mathbf{n}_i = \nabla c_i \approx \sum_j m_j \frac{1}{\rho_j} \nabla W_{ij} \tag{4.11}$$

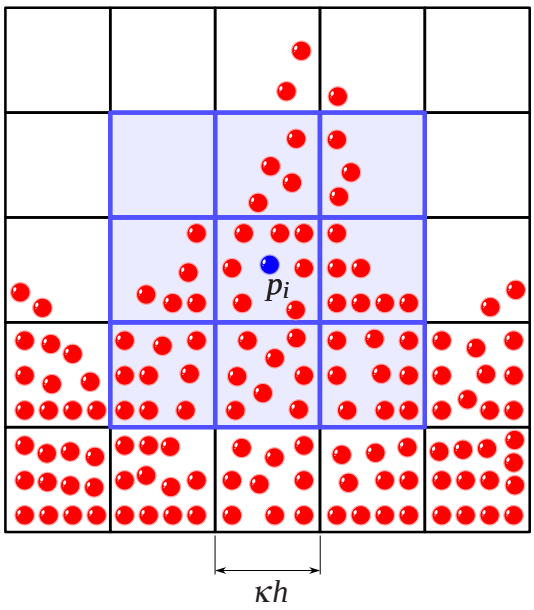


Figura 4.1: Uma grade cobrindo o domínio do fluido. A busca pelos vizinhos de uma partícula p_i (azul na figura) pode ser realizada facilmente encontrando o bloco B que a partícula pertence e iterando pelos blocos que compartilham um vértice com (m,n) (região azul-claro).

Uma partícula i é considerada de superfície se a magnitude do seu vetor normal excede um limite predeterminado.

4.2 Busca espacial

Uma das operações mais repetidas durante a simulação utilizando o método SPH é a busca pelas partículas mais próximas de acordo com o suporte da função núcleo κh. Se esse suporte não é transiente, a busca pode ser melhorada se uma grade regular de espaçamento kh recobrindo todo o domínio for utilizada. A Figura 4.1 ilustra a estrutura. Cada bloco B na estrutura é indexado por meio de tuplas (m,n) (2D) ou (m,n,o) (3D). A complexidade de busca a essa estrutura é $O(kN)$, onde N é o número de partículas envolvidas na simulação e k o número médio de partículas por bloco. Uma estrutura de dados eficiente é fundamental para a implementação de um simulador interativo de fluidos ou mesmo em tempo-real.

4.3 Detecção e resposta de colisão

As forças externas podem também ser aplicadas individualmente nas partículas devido à colisão com outros corpos ou por meio de interação com o usuário. Nesse último caso, a força é aplicada diretamente na partícula ou no conjunto de partículas selecionado. No primeiro caso,

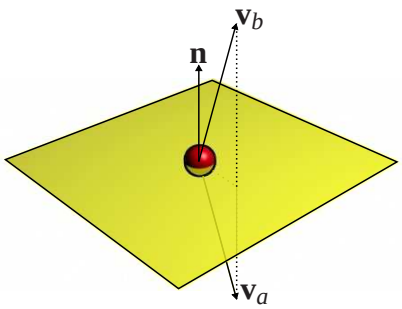


Figura 4.2: Colisão de uma partícula com um plano. No momento em que a partícula tenta penetrar no plano com velocidade $\mathbf{v}_a$ ela é refletida de acordo com a normal à superfície $\mathbf{n}$ e adquire uma nova velocidade $\mathbf{v}_b$.

a colisão manipula diretamente a velocidade ao invés da força. Quando ocorre a colisão entre partícula e objeto, a velocidade da partícula é simplesmente refletida de acordo com a normal à superfície de colisão. Isso faz com que a partícula não ultrapasse os limites do objeto. A Figura 4.2 ilustra isso. Uma partícula com velocidade $\mathbf{v}_a$ tenta penetrar em uma superfície e sua velocidade é refletida de maneira que agora ela tende a se afastar do objeto com velocidade $\mathbf{v}_b$. A colisão pode ou não ser perfeitamente elástica e é dada por:

$$\mathbf{v}_b = \mathbf{v}_a + 2\lambda(\mathbf{v}_a \cdot \mathbf{n})\mathbf{n} \tag{4.12}$$

onde $\lambda \in [0,1]$ representa o coeficiente de restituição das superfícies em contato. Se $\lambda = 1$ então a colisão é perfeitamente elástica e a velocidade mantém sua magnitude.

4.4 Avanço temporal

A seção anterior descreve o cálculo das forças envolvidas em cada partícula no processo de simulação numérica. O próximo passo é realizar o avanço temporal, ou seja, encontrar a próxima posição das partículas dada uma condição inicial. As equações de movimento a uma partícula de massa m posicionada em $\mathbf{r}$, velocidade $\mathbf{v} = \dot{\mathbf{r}}$ e aceleração $\mathbf{a} = \dot{\mathbf{v}} = \ddot{\mathbf{r}}$ estão na forma da segunda Lei de Newton. A equação abaixo é uma EDO de segunda ordem:

$$\mathbf{f} = m\ddot{\mathbf{r}} \tag{4.13}$$

A fim de encontrar $\mathbf{r}$, a nova posição a partícula, é necessário resolver esta EDO de segunda ordem. Contudo é possível transformar esta equação de segunda ordem em uma equação de primeira ordem, adquirindo as vantagens dos métodos de resolução de equações diferenciais de

primeira ordem. Ademais, o problema é escrito da seguinte maneira:

$$\dot{\mathbf{r}} = \mathbf{v},$$

$$\dot{\mathbf{v}} = \mathbf{f}/m$$

As equações acima ainda podem ser reescritas da seguinte forma:

$$\begin{bmatrix} \dot{\mathbf{r}} \\ \dot{\mathbf{v}} \end{bmatrix} = \begin{bmatrix} \mathbf{v} \\ \mathbf{f}/m \end{bmatrix} \tag{4.14}$$

O vetor $\mathbf{S}(t) = [\mathbf{r} \quad \mathbf{v}]^T$ é chamado de *state vector*. O sistema de equações diferenciais será então:

$$\frac{d\mathbf{S}}{dt} = \frac{d}{dt}\begin{bmatrix} \mathbf{r} \\ \mathbf{v} \end{bmatrix} = \begin{bmatrix} \dot{\mathbf{r}} \\ \dot{\mathbf{v}} \end{bmatrix} = \begin{bmatrix} \mathbf{v} \\ \mathbf{f}/m \end{bmatrix} \tag{4.15}$$

A simulação física é questão de atualizar o *state vector* do sistema.

Para realizar a atualização das partículas é necessário algum método a integração temporal. Textos de análise numérica possuem descrição detalhada a respeito dos diversos métodos existentes [5, 48]. Nesse projeto, foram implementados os seguintes métodos: Euler, Euler modificado, Runge-Kutta de quarta ordem, Ponto médio e *leap-frog*. Desses algoritmos, vale a pena destacar o método *leap-frog*, muito utilizado na literatura de animação por computador.

4.4.1 Método de integração de Euler

O método de Euler é talvez o mais simples dos métodos de integração. Sua baixa ordem de convergência ($O(\Delta t)$) o torna inviável do ponto de vista prático a simulações científicas, contudo, sua simplicidade faz dele uma excelente ferramenta didática em cursos como análise numérica além de ser adequado a animação por computador. O avanço temporal utilizando o método de Euler modificado é dado por:

$$\mathbf{v}^{t+1} = \mathbf{v}^t + \Delta t \mathbf{f}^t \tag{4.16}$$

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \Delta t \mathbf{v}^{t+1}. \tag{4.17}$$

Nessa dissertação, esse método serviu como base a o integrador temporal mostrado na Seção 3.4.1.

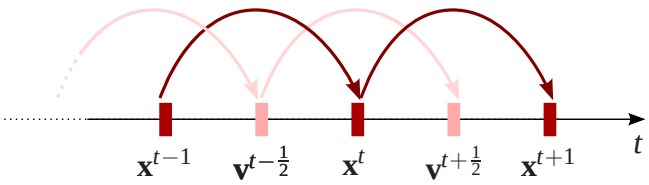


Figura 4.3: O método *leap-frog* utiliza a velocidade no tempo $\mathbf{v}^{t+\frac{1}{2}}$ a calcular a posição em $\mathbf{x}^{t+1}$.

4.4.2 Método de integração *Leap-frog*

O *leap-frog* [12] é um integrador temporal atraente por algumas razões: implementação simples e rápida (tão simples como o método de Euler); possui ordem de convergência $O(\Delta t^2)$, onde Δt é o avanço temporal; realiza apenas uma avaliação da força $\mathbf{f}^t$ no tempo t; baixo custo de armazenamento em memória principal quando comparado a métodos de alta ordem.

A Figura 4.3 ilustra o comportamento do método. Sabendo que $\mathbf{u}(t+\frac{1}{2}\Delta t) = \mathbf{u}^{t+\frac{1}{2}}$, a velocidade é calculado no tempo $\mathbf{v}^{t+\frac{1}{2}}$ e a posição da partícula em $\mathbf{x}^{t+1}$:

$$\mathbf{v}^{t+\frac{1}{2}} = \mathbf{v}^{t-\frac{1}{2}} + \Delta t \mathbf{f}^t \tag{4.18}$$

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \Delta t \mathbf{v}^{t+\frac{1}{2}} \tag{4.19}$$

A velocidade no tempo t é dada por uma média das velocidades anteriormente:

$$\mathbf{v}^t = \frac{1}{2}(\mathbf{v}^{t+\frac{1}{2}} + \mathbf{v}^{t-\frac{1}{2}}) \tag{4.20}$$

Para ser iniciado, o *leap-frog* exige que a velocidade exista no tempo $\mathbf{v}^{-\frac{1}{2}}$ por conta da forma como ele faz o avanço temporal. Essa velocidade pode ser facilmente obtida usando o método de Euler para retroceder no tempo.

4.5 Interface com usuário

Para facilitar o processo de criação de animações computacionais de escoamento de fluidos, uma interface para o *software* livre *Blender 3D* foi desenvolvida (Figura 4.4). Essa interface deixa a técnica mais atraente de forma que os usuarios tem liberdade para experimentá-la. Além disso, o desenvolvedor se beneficia para agilizar a depuração e testes do código. O *Blender 3D* foi escolhido para servir como plataforma para criação dessa interface por algumas razões: é um *software* muito maduro para criação, animação e pós-produção; possui uma API em *Python* que permite uma integração fácil com outros códigos; por último, é um *software* livre. Os passoas para se utilizar o *plugin* desenvolvido se resumem a:

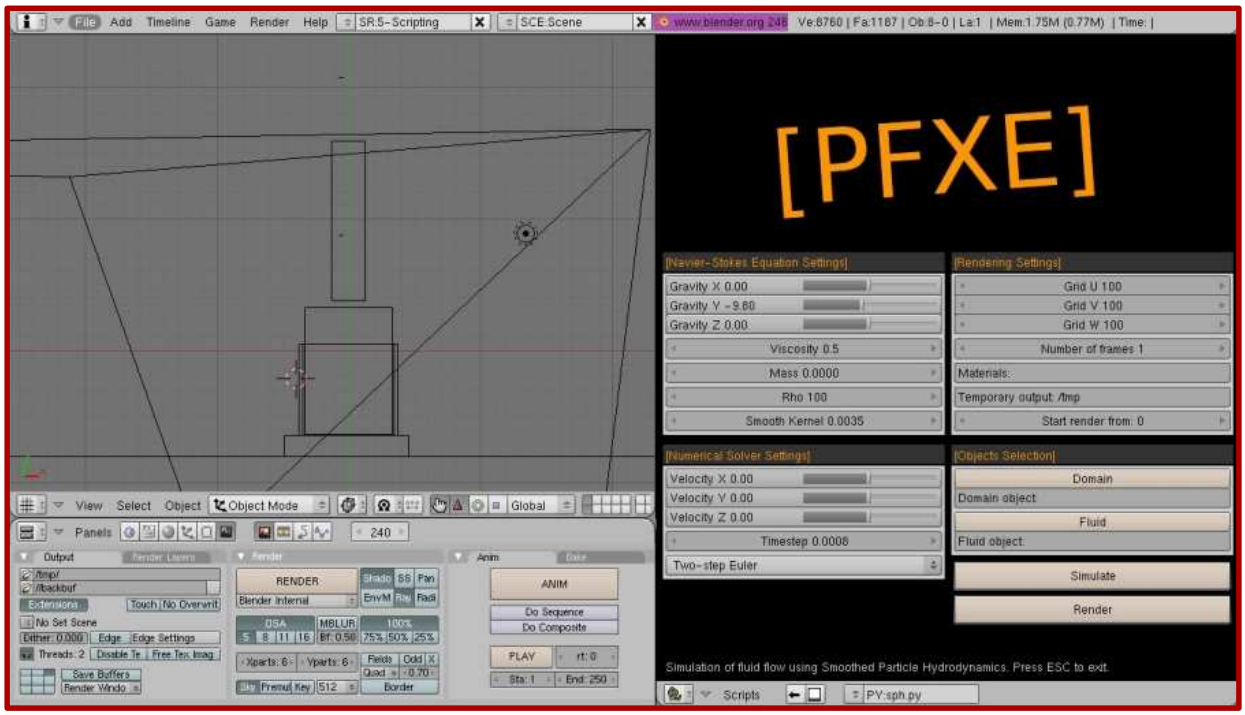


Figura 4.4: A figura acima ilustra um exemplo de utilização da interface desenvolvida para o *Blender 3D*. A interface desenvolvida está à direita na figura acima.

1. Escolha dos parâmetros de simulação por meio dos botões da interface (*gravity*, *viscosity*, *smooth kernel*, etc);
2. Escolha do objeto que representa o domínio (clicando em *Domain*);
3. Escolha do objeto que representa o fluid (clicando em *Fluid*);
4. *Simulate*, gera cada etapa da simulação;
5. *Render*, gera as imagens criadas na etapa anterior.

Utilizando-se o *Blender 3D*, o resultado final fica com aspecto muito mais profissional e realista (Ver Figura 4.9 e Figura 4.10). Além disso, o *Blender 3D* permite que as simulações sejam integradas com a cena corrente, tornando ainda mais fácil o processo de criação interativa.

4.6 Resultados

Os resultados das simulações obtidos podem ser divididos em dois tipos: *off-line* e em taxas interativas. Os quadros gerados *off-line* utilizam ferramentas auxiliares para visualização do código como o *Blender 3D*[1], *POV-Ray*[2] e o método MPU (Seção 2.7.2). Os quadros gerados em

[1]Endereço eletrônico: *http://www.blender3d.org*
[2]Endereço eletrônico: *http://www.povray.org/*

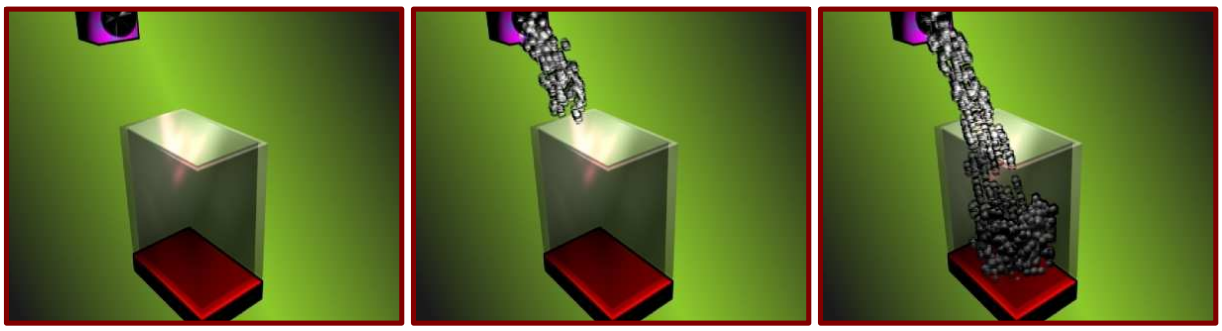


Figura 4.5: Seqüencia de imagem que mostra o preenchimento de um tanque com partículas.

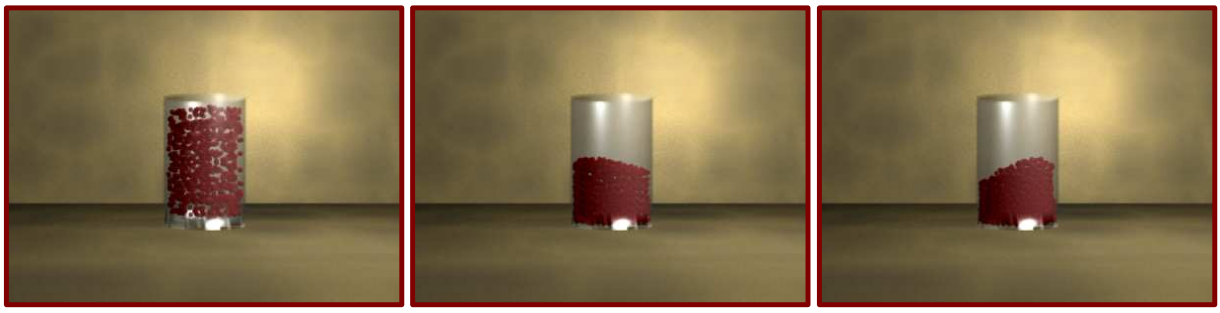


Figura 4.6: Seqüencia de imagem que mostra o movimento do fluido em um copo.

taxas interativas foram implementados em OpenGL[3] utilizando o *marching cubes* [28] (Seção 2.7.1).

As Figuras 4.5 e 4.6 ilustram uma seqüência de imagens das primeiras simulação realizada utilizando o método SPH. Em ambas, nenhum método é usado a representação da superfície, portanto as partículas podem ser observadas nas figuras. Além disso, não é feito nenhuma restrição quanto ao ganho de energia, caracterizando a técnica tradicional do SPH. Essas imagens foram obtidas com o *Blender 3D*.

A Figura 4.8 mostra o resultado de um bloco de água imerso em um ambiente natural. A superfície foi gerada pelo MPU e o quadro produzido pelo *POV-Ray* e utilizando a restrição da energia mostrada anteriormente. Utilizando o traçador de raios, o corpo do fluido ganha um

[3]Endereço eletrônico: *http://www.opengl.org/*

Figura 4.7: Quadros gerado de um bloco viscoso que cai sobre uma superfície de mármore.

Figura 4.8: Quadro gerado de um bloco de água num ambiente natural.

aspecto mais realista, incorporando-se à cena. A Figura 4.7 tem o mesmo objetivo. Podemos ver pequena seqüência de imagens de uma massa viscosa caindo sobre a mesa de uma cozinha. Novamente, o traçador de raios dá a cena um toque de realismo que faz com que o usuário fique imerso na cena. Nessas figuras apresentadas, cada quadro demorava em média 200*s* a ser gerado.

Ao contrário dos resultados anteriores, as animações obtidas utilizando *marching cubes* são gerados mais rapidamente, o que permite acompanhá-la em taxas interativas. Os quadros da Figura 4.11 mostram um bloco de fluido que teve sua superfície gerada com o *OpenGL*. A baixa resolução da grade inicial somado à ausência do traçador de raios (a criação de sombras e transparências) cria uma superfície com aspecto menos realista. As Figuras 4.9 e 4.10 foram geradas usando-se a interface desenvolvida em conjunto com o *Blender 3D*e utilizando como poligonalizador da superfície foi feita com o *Marching Cubes*. Com um pouco de habilidade no Blender, é possivel criar animações realistas e de grande apelo visual.

4.6.1 Limitações

A primeira grande limitação da técnica é a perda de energia. Mesmo utilizando os mecanismos apresentados na Seção 3.4, o aspecto do fluido é mais viscoso do que realmente deveria ser. Se levada em conta a energia interna do fluido a qualidade da simulação aumenta, no entanto um novo problema surge. Em função da desordem das partículas, a energia interna pode crescer

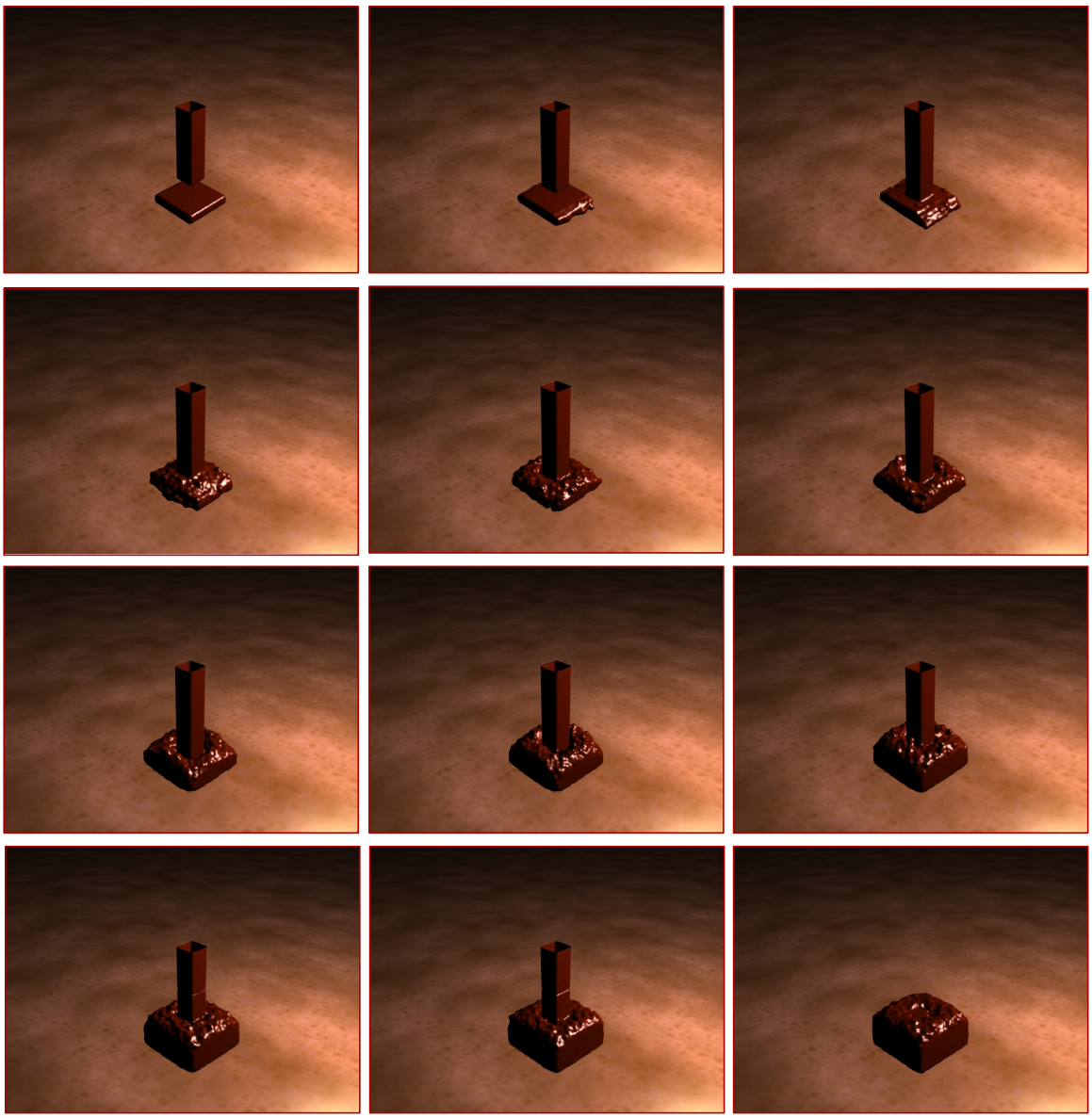


Figura 4.9: A seqüência de imagens mostra a queda de chocolate em um recipiente invisível.

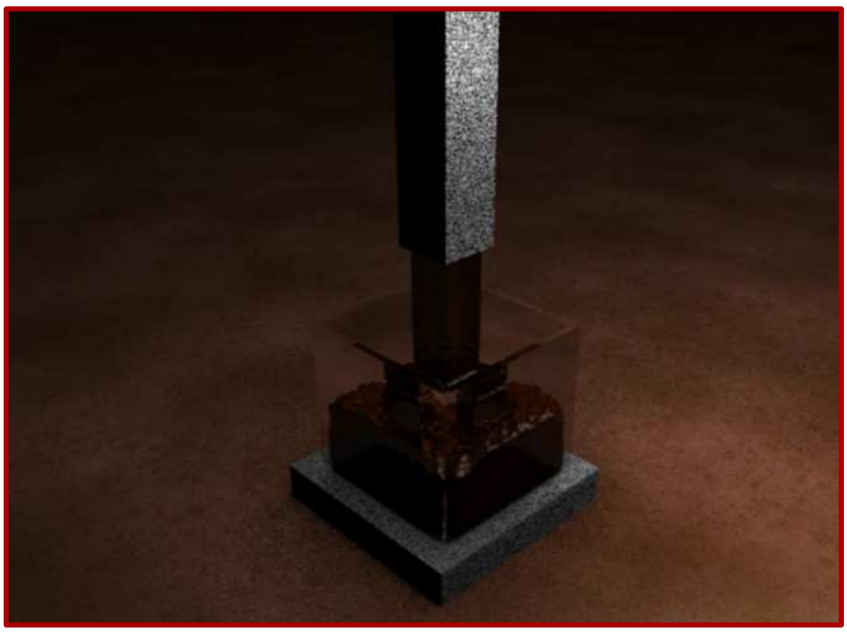


Figura 4.10: Imagem final da seqüência de chocolate gerada no *Blender 3D*.

de tal forma a superar a energia total do sistema. Uma possível solução limitar o crescimento de energia interna realizando um truncamento em um valor máximo permitido. Outra solução é incluir a energia interna no lado esquerdo da Equação 3.53 o que levaria a um sistema não linear nas variáveis α_i.

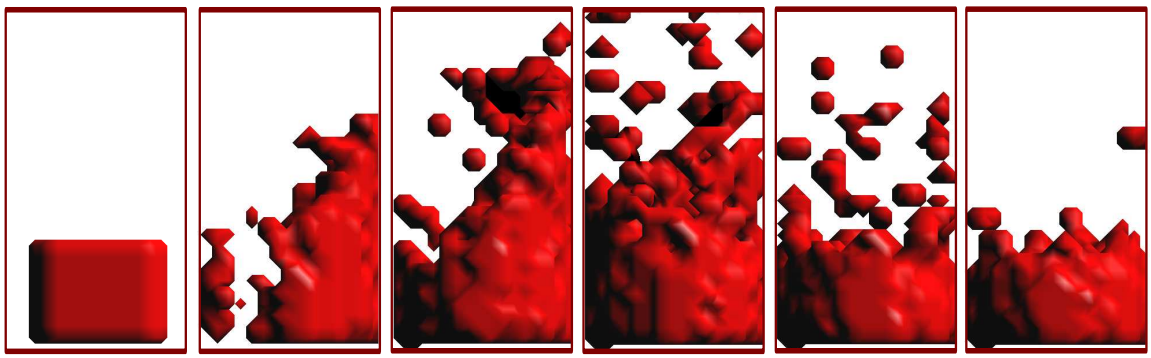


Figura 4.11: O bloco inicial de fluido (esquerda) sofre uma força horizontal no sentido da esquerda para a direita. Os quadros seguintes mostram as resoluções intermediárias do fluido. A grade de pontos utilizada tem dimensões $40 \times 80 \times 40$.

Podemos destacar aqui uma grande vantagem do *marching cubes* que permite a criação de animações: boa coerência entre quadros. A coerência entre quadros é um fator importante, pois ela não cria ou remove partes das superfícies de maneira brusca, uma das desvantagens no uso do MPU. Como explicado anteriormente, o MPU exige que os pontos sejam equipados com *normais consistentes* (contrariamente ao *marching cubes*). Essas normais são aproximadas pelo SPH utilizando a Equação 4.11 e como resultado elas não possuem forte consistência. Além disso, pontos do interior do fluido podem ser considerados de superfície se a magnitude do vetor

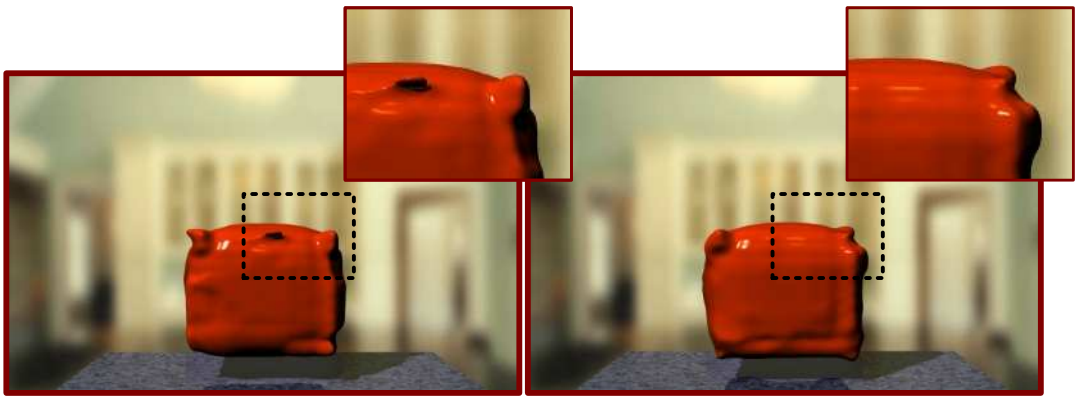


Figura 4.12: As imagens acima mostram a falta de coerência entre quadros consecutivos. A forma da superfície implícita muda drasticamente criando ou removendo partes do objeto.

normal no seu interior ultrapassar o limiar especificado. Em outras palavras, o MPU não fornece uma seqüência de imagens adequada para animação. A Figura 4.12 ilustra esse fato onde dois quadros consecutivos possuem características bem diferentes devido à função implícita gerada.

5 *Conclusão*

Animações auxiliadas por computador se tornaram fundamental em diversas aplicações, desde jogos eletrônicos e entretenimento digital até cirurgia virtual e simuladores de fobias, passando inclusive pela bilionária indústria de cinema. Nesse ponto, o sentimento daqueles que tentam simular em computador fenômenos naturais é de que a natureza brinca e os desafia, sempre deixando uma incógnita no ar, como a solução analítica para as equações de Navier-Stokes. Muitos aceitaram o desafio e trabalham dia e noite para prover uma representação cada vez mais fiel desses fenômenos. Nesse sentido, esta dissertação visa ser uma introdução à área de fluidos em computação gráfica, em especial uma técnica conhecida como *Smoothed Particles Hydrodynamics*, apontando uma de suas deficiências e uma possível solução. Este capítulo encerra a presente dissertação com as conclusões do trabalho realizado além de destacar pontos que carecem de especial atenção.

5.1 Computação gráfica e SPH

Na literatura de animação por computador, três tipos de abordagens modelam o problema de fluidos: euleriana, lagrangeana e mista. As abordagens eulerianas, já bastante estudadas e presentes em ferramentas proprietárias como Autodesk Maya[1] ou mesmo em *software* livre como o *Blender 3D*, produzem resultados surpreendentes a ponto de muitas vezes confundir o que é real com o criado computacionalmente. Abordagens eulerianas-lagrangeanas com malhas começam a ter maior expressividade em computação gráfica [15, 14, 26] com trabalhos cujo realismo prende a atenção do observador. O uso de abordagens lagrangeanas sem malha tem recebido especial atenção nos últimos anos para suprir as deficiências dos métodos baseados em malhas. Apesar do realismo proporcionado ainda não ser compatível com as técnicas tradicionais [14], os recentes avanços movem na direção de diminuir a distância de qualidade entre ambos [1, 33].

Dentre as técnicas lagrangeanas livre de malhas, o SPH tem se mostrado promissor e re-

[1] Endereço eletrônico: *http://www.autodesk.com*

cebido muito investimento da comunidade científica. Uma das dificuldades em se utilizar o SPH com métodos explícitos de integração é a quantidade de parâmetros a serem configurados (numa simulação tradicional temos: suporte compacto κh, massa m, densidade de repouso ρ_0, viscosidade μ, além da escolha das funções núcleo W, posicionamento inicial das partículas e por fim o passo de tempo Δt) e a sensibilidade às condições iniciais. O passo de tempo Δt precisa obedecer a determinadas condições para que a simulação seja estável, chamadas condições CFL. Na literatura, existe uma definição para as condições CFL adaptada ao SPH e que traz bons resultados ao preço de usar passos de tempo variados (o que dificulta a sincronização de quadros) e com pequenos passos de tempo que se aplicam uniformemente a todas as partículas.

Nesse sentido, o presente trabalho apresentou uma solução simples, já aplicada em outros contextos, para simulação de fluidos baseada em partículas. Oriunda da lei de conservação de energia, a idéia é restringir o ganho arbitrário de energia cinética e potencial por conta da integração numérica. O sistema não deve ganhar energia enquanto ele a perde de duas formas: colisões com a fronteira ou interação entre as partículas. Os resultados mostram que a evolução da energia é efetivamente controlada, resultando em um algoritmo menos suscetível às condições iniciais. Por outro lado, dessa forma a perda de energia é valorizada e o fluido ganha um aspecto mais viscoso, mesmo na ausência de viscosidade. Para contornar esse problema, a equação da energia presente na equação de Navier-Stoke é usada para controlar a energia interna do sistema. Novamente, esse ganho pode ser arbitrariamente elevado, portanto deve ser limitado de maneira que a soma das energias não ultrapasse o valor inicial.

5.2 Trabalhos futuros

Apesar de fornecer boas imagens, as superfícies geradas pelo MPU têm uma deficiência devido à qualidade das normais. Em um mesmo quadro, as normais aproximadas podem causar a oscilação das aproximações, criando superfícies mais irregulares. Para a coerência entre quadros, uma possível melhoria para o processo de animação da imagem é interpolar as normais referentes à mesma partícula nos diferentes instantes de tempo e utilizar o valor interpolado em cada quadro, fazendo-a variar de maneira mais suave.

Outro ponto de melhoria é uma formulação da equação de restrição de energia (Equação 3.53) que envolva a energia interna. Fazendo isso, o aspecto do fluido tende a ser mais realista sem a necessidade de criar novos limites, como feito atualmente. Contudo, simplesmente inserindo a Equação da energia interna 3.53, um sistema não-linear é criado nas incógnitas α_i, aumentando a robustez do método ao custo da solução do sistema linear.

Outro aspecto não abordado nessa dissertação e que tem forte influência na estabilidade do método é o tamanho do suporte compacto κh. O crescimento do suporte compacto tende a suavizar a solução das EDPs, o que evita "explosões" numéricas. As funções núcleos também podem ser consideradas para aumentar a estabilidade do método.

Por fim, fica ainda o desafio da criação de um algoritmo incondicionalmente estável utilizando o SPH para simulação de escoamento fluidos.

APÊNDICE A – Delta de Dirac

A função delta de Dirac é denotada por $\delta(x)$ e definida como:

$$\delta(x) = \lim_{\varepsilon \to 0} \begin{cases} 0 & \text{se } x < -\frac{\varepsilon}{2} \\ \frac{1}{\varepsilon} & \text{se } -\frac{\varepsilon}{2} < x < \frac{\varepsilon}{2} \\ 0 & \text{se } x > \frac{\varepsilon}{2} \end{cases} \tag{A.1}$$

A.1 Propriedades

Calculando a sua integral no intervalo $(-\infty, +\infty)$:

$$\begin{aligned} \int_{-\infty}^{+\infty} \delta(x)dx &= \int_{-\infty}^{-\frac{\varepsilon}{2}} \delta(x)dx + \int_{-\frac{\varepsilon}{2}}^{\frac{\varepsilon}{2}} \delta(x)dx + \int_{\frac{\varepsilon}{2}}^{+\infty} \delta(x)dx \\ &= 0 + \int_{-\frac{\varepsilon}{2}}^{\frac{\varepsilon}{2}} \delta(x)dx + 0 \\ &= \lim_{\varepsilon \to 0} \int_{-\frac{\varepsilon}{2}}^{\frac{\varepsilon}{2}} \frac{1}{\varepsilon} dx \\ &= 1 \end{aligned}$$

Suponha $f(x)$ uma função de classe C^{∞}. Integrando $f(x)\delta(x)$:

$$\begin{aligned} \int_{-\infty}^{+\infty} f(x)\delta(x)dx &= \int_{-\infty}^{-\frac{\varepsilon}{2}} f(x)\delta(x)dx + \int_{-\frac{\varepsilon}{2}}^{\frac{\varepsilon}{2}} f(x)\delta(x)dx + \int_{\frac{\varepsilon}{2}}^{+\infty} f(x)\delta(x)dx \\ &= 0 + \int_{-\frac{\varepsilon}{2}}^{\frac{\varepsilon}{2}} f(x)\delta(x)dx + 0 \\ &= \int_{-\frac{\varepsilon}{2}}^{\frac{\varepsilon}{2}} f(x)\frac{1}{\varepsilon}dx, \\ &= \lim_{\varepsilon \to 0} \frac{f(\frac{\varepsilon}{2}) + f(-\frac{\varepsilon}{2})}{2} \\ &= f(0) \end{aligned} \tag{A.2}$$

Novamente, suponha $f(x)$ uma função de classe C^∞. Integrando $f(x)\delta(x-x_0)$:

$$\begin{aligned}
\int_{-\infty}^{+\infty} f(x)\delta(x-x_0)dx &= \int_{-\infty}^{+\infty} f(u+x_0)\delta(u)du \quad \text{assumindo que } u = x-x_0 \\
&= \int_{-\infty}^{-\frac{\varepsilon}{2}} f(u+x_0)\delta(u)du + \int_{-\frac{\varepsilon}{2}}^{\frac{\varepsilon}{2}} f(u+x_0)\delta(u)du + \\
&\quad + \int_{\frac{\varepsilon}{2}}^{+\infty} f(u+x_0)\delta(u)du \\
&= 0 + \int_{-\frac{\varepsilon}{2}}^{\frac{\varepsilon}{2}} f(u+x_0)\delta(u)du + 0 \\
&= \int_{-\frac{\varepsilon}{2}}^{\frac{\varepsilon}{2}} f(u+x_0)\delta(u)du \\
&= \lim_{\varepsilon\to 0} \frac{f(\frac{\varepsilon}{2}+x_0) + f(-\frac{\varepsilon}{2}+x_0)}{2} \\
&= f(x_0) \qquad \text{(A.3)}
\end{aligned}$$

Referências Bibliográficas

[1] ADAMS, B., PAULY, M., KEISER, R., AND GUIBAS, L. J. Adaptively sampled particle fluids. In *ACM SIGGRAPH 2007* (2007).

[2] BALSARA, D. S. von neumann stability analysis of smoothed particle hydrodynamics—suggestions for optimal algorithms. *J. Comput. Phys. 121*, 2 (1995), 357–372.

[3] BARTELS, R. H., AND JEZIORANSKI, J. J. Constr and eval: routines for fitting multinomials in a least-squares sense. *ACM Trans. Math. Softw. 11* (1985), 218–228.

[4] BARTELS, R. H., AND JEZIORANSKI, J. J. Least-squares fitting using orthogonal multinomials. *ACM Trans. Math. Softw. 11*, 3 (1985), 201–217.

[5] BURDEN, R. L., AND FAIRES, J. D. *Análise Numérica*. Thompson, 2003.

[6] CASTELO, A., NONATO, L. G., SIQUEIRA, M., AND MINGHIM, R. The j1a triangulation: An adaptive triangulation in any dimension. *Computer & Graphics 30*, 5 (2006), 737–753.

[7] CLEARY, P. W., PYO, S. H., PRAKASH, M., AND KOO, B. K. Bubbling and frothing liquids. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers* (2007), p. 97.

[8] CUADROS-VARGAS, A., NONATO, L. G., MINGHIM, R., AND ETIENE, T. Imesh: An image based quality mesh generation technique. In *SIBGRAPI - Brazilian Symposium on Computer Graphics and Image Processing* (July 2005).

[9] DE OLIVEIRA FORTUNA, A. *Técnicas Computacionais para Dinâmica dos Fluidos*. Edusp - Editora da Universidade de São Paulo, 2000.

[10] DESBRUN, M., AND GASCUEL, M.-P. Smoothed particles : A new paradigm for animating highly deformable bodies. In *Computer Animation and Simulation '96* (1996), pp. 61–76.

[11] DYKAA, C. T., AND INGEL, R. P. An approach for tension instability in smoothed particle hydrodynamics (sph). *Computers & Structures 57* (1995), 573–580.

[12] EBERLY, D. H. *Game Physics*. Morgan Kaufmann Publishers, 2004.

[13] ENRIGHT, D., MARSCHNER, S., AND FEDKIW, R. Animation and rendering of complex water surfaces. *ACM Trans. Graph. 21*, 3 (2002), 736–744.

[14] FELDMAN, B. E., O'BRIEN, J. F., AND KLINGNER, B. M. Animating gases with hybrid meshes. In *Proceedings of ACM SIGGRAPH 2003* (2005).

[15] FELDMAN, B. E., O'BRIEN, J. F., KLINGNER, B. M., AND GOKTEKIN, T. G. Fluids in deforming meshes. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation 2005* (2005).

[16] FIRES, T.-P., AND MATTHIES, H.-G. Classification and overview of meshfree methods. Tech. rep., Technical University Braunschweig, Brunswick, Alemanhã, Julio 2004.

[17] FOSTER, N., AND FEDKIW, R. Practical animation of liquids. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), ACM, pp. 23–30.

[18] FULK, D. A. *A numerical analysis of smoothed particle hydrodynamics*. PhD thesis, Air University, 1994.

[19] GINGOLD, R. A., AND MONAGHAN, J. J. Smoothed particle hydrodynamics - theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society 181* (Novembro 1977), 375–389.

[20] GOIS, J. P. *Mínimos-quadrados e aproximação de superfície de pontos: novas perspectivas e aplicações*. PhD thesis, USP/ICMC, 2008.

[21] GOIS, J. P., POLIZELLI-JUNIOR, V., ETIENE, T., TEJADA, E., FILHO, A. C., NONATO, L. G., AND ERTL, T. Robust and adaptive surface reconstruction using partition of unity implicits. In *20th Brazilian Symposium on Computer Graphics and Image Processing* (October 2007).

[22] HIRT, C. W., AMSDEN, A. A., AND COOK, J. L. An arbitrary lagrangian-eulerian computing method for all flow speeds. *J. Comput. Phys. 135* (1997), 203–216.

[23] IRVING, G., GUENDELMAN, E., LOSASSO, F., AND FEDKIW, R. Efficient simulation of large bodies of water by coupling two and three dimensional techniques. In *Proceedings of ACM SIGGRAPH 2006* (2006).

[24] KASS, M., AND MILLER, G. Rapid, stable fluid dynamics for computer graphics. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1990), ACM Press, pp. 49–57.

[25] KHAREVYCH, L., YANG, W., TONG, Y., KANSO, E., MARSDEN, J. E., SCHRÖDER, P., AND DESBRUN, M. Geometric, variational integrators for computer animation. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2006), Eurographics Association, pp. 43–51.

[26] KLINGNER, B. M., FELDMAN, B. E., CHENTANEZ, N., AND O'BRIEN, J. F. Fluid animation with dynamic meshes. *ACM Trans. Graph. 25*, 3 (2006), 820–825.

[27] KNAPP, C. E. *An Implicit Smooth Particle Hydrodynamics Code*. PhD thesis, University of New Mexico, 2000.

[28] LEWINER, T., LOPES, H., VIEIRA, A. W., AND TAVARES, G. Efficient implementation of marching cubes' cases with topological guarantees. *journal of graphics tools 8*, 2 (2003), 1–15.

[29] LI, S., AND LIU, W. K. *Meshfree Particle Methods*. Springer, 2004.

[30] LIU, G. R., AND LIU, M. B. *Smoothed Particle Hydrodynamics*. World Scientific, 2003.

[31] LORENSEN, W. E., AND CLINE, H. E. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1987), ACM Press, pp. 163–169.

[32] LOSASSO, F., GIBOU, F., AND FEDKIW, R. Simulating water and smoke with an octree data structure. *ACM Trans. Graph. 23*, 3 (2004), 457–462.

[33] LOSASSO, F., TALTON, J., KWATRA, N., AND FEDKIW, R. Two-way coupled sph and particle level set fluid simulation. In *IEEE Transactions on Visualization and Computer Graphics* (Fevereiro 2008), I. C. S. D. Library, Ed., IEEE Computer Society.

[34] LUCY, L. B. A numerical approach to the testing of the fission hypothesis. *Astronomical Journal 82* (Dec. 1977), 1013–1024.

[35] MONAGHAN, J. J. Why particle methods work. *SIAM Journal on Scientific and Statistical Computing 3*, 4 (1982), 422–433.

[36] MONAGHAN, J. J. Smoothed particle hydrodynamics. *ARAA 30* (1992), 543–574.

[37] MONAGHAN, J. J. Implicit sph drag and dusty gas dynamics. *J. Comput. Phys. 138* (1997), 801–820.

[38] MORRIS, J. P. *Analysis of Smoothed Particles Hydrodynamics with Applications*. PhD thesis, Monash University, 1996.

[39] MULLER, M., CHARYPAR, D., AND GROSS, M. Particle-based fluid simulation for interactive applications. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2003), Eurographics Association, pp. 154–159.

[40] MÜLLER, M., DORSEY, J., MCMILLAN, L., JAGNOW, R., AND CUTLER, B. Stable real-time deformations. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2002).

[41] MÜLLER, M., HEIDELBERGER, B., TESCHNER, M., AND GROSS, M. Meshless deformations based on shape matching. *ACM Trans. Graph. 24*, 3 (2005), 471–478.

[42] MÜLLER, M., SCHIRM, S., AND DUTHALE, S. Screen space meshes. In *Proceedings ACM SIGGRAPH / EUROGRAPHICS Symposium on Computer Animation (SCA)* (2007).

[43] MÜLLER, M., SOLENTHALER, B., KEISER, R., AND GROSS, M. Particle-based fluid-fluid interaction. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2005).

[44] O'BRIEN, J. F., AND HODGINS, J. K. Dynamic simulation of splashing fluids. In *CA '95: Proceedings of the Computer Animation* (Washington, DC, USA, 1995), IEEE Computer Society, p. 198.

[45] OHTAKE, Y., BELYAEV, A., ALEXA, M., TURK, G., AND SEIDEL, H.-P. Multi-level partition of unity implicits. *ACM Trans. Graph. 22*, 3 (2003), 463–470.

[46] PAIVA, A., PETRONETTO, F., LEWINER, T., AND TAVARES, G. Particle-based non-newtonian fluid animation for melting objects. In *19th Brazilian Symposium on Computer Graphics and Image Processing, 2006. SIBGRAPI '06.* (2006).

[47] POLIZELLI-JUNIOR, V. Métodos implícitos para a reconstrućão de superfícies a partir de nuvens de pontos. Master's thesis, USP/ICMC, 2008.

[48] PRESS, W. H., TEUKOLSKI, S. A., VETTERLING, W. T., AND FLANNERY, B. P. *Numerical Recipes in C*, 2 ed. Cambridge University Press, Outrubro 1992.

[49] QUARTERONI, A., SACCO, R., AND SALERI, F. *Numerical Mathematics*. Springer, 2000.

[50] RASMUSSEN, N., ENRIGHT, D., NGUYEN, D., MARINO, S., SUMNER, N., GEIGER, W., HOON, S., AND FEDKIW, R. Directable photorealistic liquids. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2004), Eurographics Association, pp. 193–202.

[51] RITCHMYER, R. D., AND MORTON, K. W. *Difference methods for initial-value problem*. Krieger publishing company, 1957.

[52] ROSENBERG, I. D., AND BIRDWELL, K. Real-time particle isosurface extraction. In *SI3D '08: Proceedings of the 2008 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2008), ACM, pp. 35–43.

[53] ROY, C. J. Review of code and solution verification procedures for computational simulation. *J. Comput. Phys. 205*, 1 (2005), 131–156.

[54] SHEWCHUK, J. R. *Delaunay Refinement Mesh Generation*. PhD thesis, Carnegie Mellon University, Maio 1997.

[55] SIMS, K. Particle animation and rendering using data parallel computation. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1990), ACM Press, pp. 405–413.

[56] SOARES, I. P. *Movimento de malhas e remalhamento de malhas superficiais*. PhD thesis, USP/ICMC, 2007.

[57] STAM, J. Stable fluids. In *Computer Graphics (SIGGRAPH 99)* (Agosto 1999), ACM, pp. 121–128.

[58] STAM, J. Real-time fluid dynamics for games. In *Proceedings of the Game Developer Conference* (Marh 2003).

[59] SWEGLE, J. W., HICKS, D. L., AND ATTAWAY, S. W. Smoothed particle hydrodynamics stability analysis. *J. Comput. Phys. 116*, 1 (1995), 123–134.

[60] WALD, I., BENTHIN, C., WAGNER, M., AND SLUSALLEK, P. Interactive rendering with coherent ray tracing. In *Computer Graphics Forum (Proceedings of EUROGRAPHICS 2001* (2001), A. Chalmers and T.-M. Rhyne, Eds., vol. 20, Blackwell Publishers, Oxford, pp. 153–164.

[61] YUKSEL, C., HOUSE, D. H., AND KEYSER, J. Wave particles. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007) 26*, 3 (2007), 99.

[62] ZHAO, H.-K., OSHER, S., AND FEDKIW, R. Fast surface reconstruction using the level set method. In *VLSM '01: Proceedings of the IEEE Workshop on Variational and Level Set Methods (VLSM'01)* (Washington, DC, USA, 2001), IEEE Computer Society, p. 194.