Pontifícia Universidade Católica
do Rio de Janeiro

Vitor Barata Ribeiro Blanco Barroso

Simulação eficiente de fluidos no espaço paramétrico de malhas estruturadas tridimensionais

Tese de Doutorado

Tese apresentada ao Programa de Pós–graduação em Informática do Departamento de Informática da PUC-Rio como requisito parcial para obtenção do título de Doutor em Informática.

Orientador: Prof. Waldemar Celes Filho

Rio de Janeiro
Fevereiro de 2014

Pontifícia Universidade Católica
do Rio de Janeiro

Vitor Barata Ribeiro Blanco Barroso

Simulação eficiente de fluidos no espaço paramétrico de malhas estruturadas tridimensionais

Tese apresentada ao Programa de Pós–graduação em Informática do Departamento de Informática do Centro Técnico Científico da PUC-Rio como requisito parcial para obtenção do título de Doutor em Informática. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Waldemar Celes Filho
Orientador
Departamento de Informática — PUC-Rio

Prof. Marcelo Gattass
PUC-Rio

Prof. Marcio da Silveira Carvalho
PUC-Rio

Prof. Manuel Menezes de Oliveira Neto
UFRGS

Prof. Claudio Esperança
UFRJ

Prof. José Eugenio Leal
Coordenador Setorial do Centro Técnico Científico — PUC-Rio

Rio de Janeiro, 19 de fevereiro de 2014

Vitor Barata Ribeiro Blanco Barroso

Graduou-se em Engenharia de Controle e Automação pela Universidade de Brasília. Concluiu o Mestrado em Informática pela Pontifícia Universidade Católica do Rio de Janeiro, especializando-se na área de Computação Gráfica. Trabalhou junto ao Instituto Tecgraf no desenvolvimento de algoritmos de simulação física, visualização científica e renderização em tempo real, com aplicações nas indústrias do petróleo e do entretenimento. Desenvolveu, junto com seu orientador durante o Doutorado, ferramentas para a simulação Euleriana de fluidos em grades estruturadas explorando o processamento paralelo de placas gráficas. Tem especial interesse em algoritmos paralelos para a simulação de fenômenos físicos e habilidades humanas.

Ficha Catalográfica

Barroso, Vitor Barata Ribeiro Blanco

Simulação eficiente de fluidos no espaço paramétrico de malhas estruturadas tridimensionais / Vitor Barata Ribeiro Blanco Barroso; orientador: Waldemar Celes Filho. — Rio de Janeiro : PUC-Rio, Departamento de Informática, 2014.

85 f: il. (color.); 29,7 cm

Tese (doutorado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2014.

Inclui bibliografia.

1. Informática – Teses. 2. Dinâmica dos fluidos computacional. 3. Simulação Euleriana. 4. Parametrização de domínios. 5. Transformação de coordenadas. 6. Computação paralela. I. Celes Filho, Waldemar. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Para meus pais, Paulo e Claudia, e minha futura esposa Beatriz, por todo o amor e confiança e por sempre me incentivarem a manter o trabalho fluindo.

Agradecimentos

Ao meu orientador Professor Waldemar Celes, pelo incentivo e apoio constantes durante toda a realização do trabalho.

Ao CNPq, à FAPERJ, à PUC-Rio e ao Tecgraf, pelos auxílios concedidos, sem os quais este trabalho não poderia ter sido realizado.

Aos meus pais, avós, irmãos e primos por todo o estímulo, apoio e confiança, e por todas as conversas revigorantes ao telefone ou nos almoços de domingo.

À minha futura esposa Beatriz pelo apoio e carinho incondicionais nos melhores e piores momentos, pela paciência nos momentos de crise, e pelo constante incentivo à conclusão deste trabalho.

Aos meus colegas da PUC-Rio e especialmente do Tecgraf pela amizade, brincadeiras e jogatinas, bem como pela cooperação, pelo aprendizado mútuo e pelo apoio ao longo de todo esse tempo.

A todos os professores e funcionários do Departamento de Informática e do Tecgraf pelos ensinamentos e pela ajuda nos problemas e tarefas do dia-a-dia acadêmico.

Resumo

Barroso, Vitor Barata Ribeiro Blanco; Celes Filho, Waldemar. **Simulação eficiente de fluidos no espaço paramétrico de malhas estruturadas tridimensionais**. Rio de Janeiro, 2014. 85p. Tese de Doutorado — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Fluidos são extremamente comuns em nosso mundo e têm papel central em muitos fenômenos naturais. A compreensão de seu comportamento tem importância fundamental em uma vasta gama de aplicações e diversas áreas de pesquisa, da análise de fluxo sanguíneo até o transporte de petróleo, da exploração do fluxo de um rio até a previsão de maremotos, tempestades e furacões. Na simulação de fluidos, a abordagem conhecida como Euleriana é capaz de gerar resultados bastante corretos e precisos, mas as computações envolvidas podem se tornar excessivamente custosas quando há a necessidade de tratar fronteiras curvas e obstáculos com formas complexas. Este trabalho aborda esse problema e apresenta uma técnica Euleriana rápida e direta para simular o escoamento de fluidos em grades estruturadas parametrizadas tridimensionais. O principal objetivo do método é tratar de forma correta e eficiente as interações de fluidos com fronteiras curvas, incluindo paredes externas e obstáculos internos. Para isso, são utilizadas matrizes Jacobianas por célula para relacionar as derivadas de campos escalares e vetoriais nos espaços do mundo e paramétrico, o que permite a resolução das equações de Navier-Stokes diretamente no segundo, onde a discretização do domínio torna-se simplesmente uma grade uniforme. O trabalho parte de um simulador baseado em grades regulares e descreve como adaptá-lo com a aplicação das matrizes Jacobianas em cada passo, incluindo a resolução de equações de Poisson e dos sistemas lineares esparsos associados, utilizando tanto iterações de Jacobi quanto o método do Gradiente Biconjugado Estabilizado. A técnica é implementada na linguagem de programação CUDA e procura explorar ao máximo a arquitetura massivamente paralela das placas gráficas atuais.

Palavras–chave

Dinâmica dos fluidos computacional; simulação Euleriana; parametrização de domínios; transformação de coordenadas; computação paralela;

Abstract

Barroso, Vitor Barata Ribeiro Blanco; Celes Filho, Waldemar (Advisor). **Efficient fluid simulation in the parametric space of three-dimensional structured grids**. Rio de Janeiro, 2014. 85p. PhD Thesis — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Fluids are extremely common in our world and play a central role in many natural phenomena. Understanding their behavior is of great importance to a broad range of applications and several areas of research, from blood flow analysis to oil transportation, from the exploitation of river flows to the prediction of tidal waves, storms and hurricanes. When simulating fluids, the so-called Eulerian approach can generate quite correct and precise results, but the computations involved can become excessively expensive when curved boundaries and obstacles with complex shapes need to be taken into account. This work addresses this problem and presents a fast and straightforward Eulerian technique to simulate fluid flows in three-dimensional parameterized structured grids. The method's primary design goal is the correct and efficient handling of fluid interactions with curved boundary walls and internal obstacles. This is accomplished by the use of per-cell Jacobian matrices to relate field derivatives in the world and parameter spaces, which allows the Navier-Stokes equations to be solved directly in the latter, where the domain discretization becomes a simple uniform grid. The work builds on a regular-grid-based simulator and describes how to apply Jacobian matrices to each step, including the solution of Poisson equations and the related sparse linear systems using both Jacobi iterations and a Biconjugate Gradient Stabilized solver. The technique is implemented efficiently in the CUDA programming language and strives to take full advantage of the massively parallel architecture of today's graphics cards.

Keywords

Computational fluid dynamics; Eulerian simulation; domain parameterization; coordinate transformation; parallel computing;

Sumário

Lista de figuras

Lista de tabelas

L'étude approfondie de la nature est la source la plus féconde des découvertes mathématiques. Non seulement cette étude, en offrant aux recherches un but déterminé, a l'avantage d'exclure les questions vagues et les calculs sans issue; elle est encore un moyen assuré de former l'analyse elle-même, et d'en découvrir les éléments qu'il nous importe le plus de connaitre, et que cette science doit toujours conserver: ces éléments fondamentaux sont ceux qui se reproduisent dans tous les effets naturels.

Jean Baptiste Joseph Fourier, *Théorie Analytique de la Chaleur.*

1
Introdução

Os fluidos estão fortemente presentes em nossa vida e desempenham um papel fundamental em muitos fenômenos naturais importantes. Exemplos de fluidos incluem rios, oceanos, o ar atmosférico, muitos combustíveis e nosso próprio sangue, entre incontáveis outros. O movimento dessas substâncias, também conhecido como escoamento, pode ser observado em todas as escalas, desde a difusão de sangue por capilares microscópicos até fenômenos poderosos como as correntes de água e de vento, as ondas do mar e os furacões.

A compreensão do comportamento dos fluidos e escoamentos é de grande interesse em muitas áreas de pesquisa. Além de todos os fenômenos naturais e biológicos, alguns objetos de estudo frequentes são a aerodinâmica de um carro, a sustentação de um avião, a propulsão gerada por uma hélice, a energia aproveitada por uma turbina, a difusão de poluentes em mares e bacias hidrográficas, e o transporte de água, gás, petróleo e outras substâncias. Devido a essa vasta gama de aplicações, por muitos anos um grande esforço de pesquisa tem sido investido na modelagem, simulação e visualização de fluidos e escoamentos.

O comportamento dinâmico de um fluido é modelado pelas equações diferenciais parciais de Navier-Stokes, cuja solução numérica com precisão se revela bastante difícil e apresenta elevado custo computacional. Por essa razão, simulações científicas muitas vezes precisam rodar por horas até que seus resultados possam ser analisados. Isso motiva o desenvolvimento de técnicas mais eficientes, tanto para a simulação completa quanto para se obter uma boa primeira aproximação do comportamento de escoamentos por meio de um modelo físico simplificado. Esse tipo de técnica também é especialmente útil para aplicações interativas como jogos eletrônicos, ferramentas de treinamento e software de visualização científica. Além de trabalharem com resoluções reduzidas de tempo e espaço, tais aplicações tipicamente utilizam implementações eficientes de modelos físicos simplificados para obterem o desempenho desejado.

Atualmente, duas abordagens principais são amplamente utilizadas na representação e simulação de fluidos: técnicas Lagrangianas, que representam

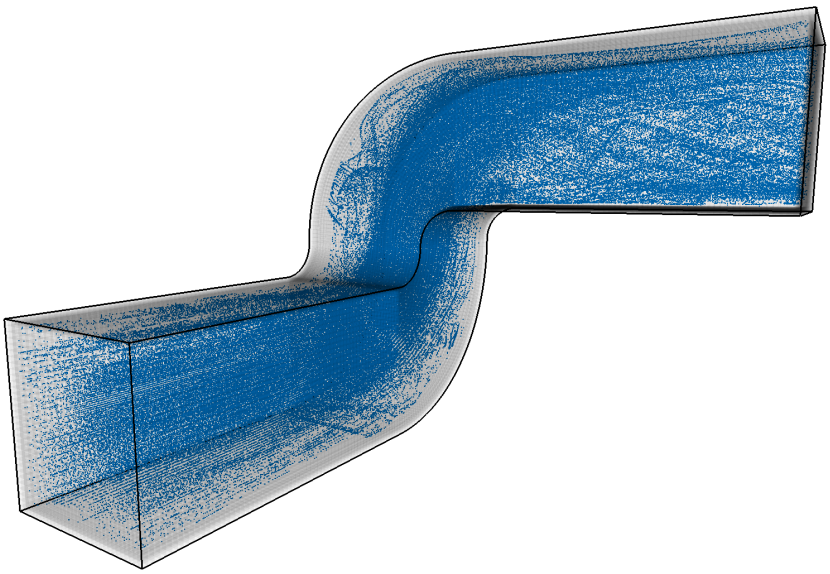


Figura 1.1 – Exemplo de escoamento em uma grade estruturada tubular tridimensional. A imagem mostra partículas sem massa transportadas pelo fluido.

um fluido por um conjunto de partículas móveis; e técnicas Eulerianas, que consideram uma amostragem das propriedades do fluido em cada elemento de uma subdivisão estacionária do espaço e integram essas propriedades no tempo a fim de se gerar um movimento implícito. Exemplos importantes de implementações eficientes de cada abordagem são a Hidrodinâmica Suavizada de Partículas (SPH) [1] e o *Stable Fluids* [2], respectivamente.

O escopo deste trabalho limita-se à abordagem Euleriana, e seu foco principal é a geração de boas aproximações do comportamento de fluidos para computação gráfica e aplicações científicas interativas. Nosso objetivo é simular eficientemente escoamentos em malhas estruturadas parametrizadas tridimensionais com fronteiras de formato arbitrário, como o caminho tubular mostrado na Figura 1.1. Obtemos desempenho elevado por meio do emprego de um modelo físico simplificado e de uma implementação projetada para tirar proveito máximo do grande poder de processamento paralelo disponível em *hardware* gráfico atual. Simplificamos e aceleramos a simulação trabalhando numa grade uniforme no espaço paramétrico, cujas derivadas espaciais se relacionam com as do espaço do mundo por meio de matrizes Jacobianas.

Nossa pesquisa toma por base o importante trabalho de Stam conhecido como *Stable Fluids* [2], que tem como resultado uma simulação Euleriana de fluidos muito eficiente e incondicionalmente estável. Em sua abordagem, as equações de Navier-Stokes são integradas em uma série de passos sequenciais, cada um utilizando os resultados parciais obtidos no anterior e apresentando adaptações específicas para se garantir estabilidade numérica. Essa técnica, implementada em GPU por Harris [3], tornou-se e se mantém como a base da

maioria dos algoritmos Eulerianos modernos de alto desempenho.

Uma das principais limitações do algoritmo *Stable Fluids* original, bem como de outros métodos Eulerianos baseados em malhas regulares, é a representação das fronteiras externas do domínio e dos obstáculos internos nele presentes. Na abordagem mais simples, essas entidades são voxelizadas na grade regular, cujas células são simplesmente marcadas como parte ou não do fluido [4]. Infelizmente, isso requer que a malha seja discretizada com uma resolução muito elevada para que se possam capturar formas curvas complexas. Além disso, mesmo com uma discretização muito refinada, o comportamento resultante apresenta artefatos visíveis sempre que a orientação de uma fronteira ou obstáculo não está perfeitamente alinhada com a da grade. Uma das principais metas da técnica proposta nesta tese é a aplicação de um tratamento mais robusto e adequado a essa questão, possibilitando simulações eficientes e livres de artefatos em malhas estruturadas parametrizadas tridimensionais com fronteiras de formato arbitrário.

Deve-se notar que parte das técnicas propostas nesta tese apresentam algumas semelhanças com um trabalho de generalização posterior do próprio Stam [5]. No entanto, enquanto sua abordagem é restrita a superfícies paramétricas tridimensionais de Catmull-Clark, nossa proposta é válida para superfícies planares e volumes tridimensionais, desde que sem buracos em sua topologia. Além disso, nosso método apresenta menor complexidade matemática, não necessitando de nenhuma informação topológica explícita ou de funções de transição entre diferentes trechos de malha. Para calcular as matrizes Jacobianas que relacionam derivadas e grandezas vetoriais entre os espaços paramétrico e do mundo, precisamos inicialmente apenas das posições globais dos nós da malha. A partir de então, podemos integrar as equações de Navier-Stokes eficientemente no espaço paramétrico, considerando curvaturas e deformações de forma natural, evitando assim o custo elevado de se trabalhar com malhas simpliciais ou de se refinar excessivamente o domínio junto às fronteiras.

Finalmente, podemos dizer que a principal contribuição desta tese é a apresentação de um algoritmo de alta eficiência para a geração de boas aproximações do comportamento de fluidos em malhas curvilíneas paramétricas tridimensionais. O método, voltado para aplicações interativas de computação gráfica, é capaz de gerar resultados aproximados em tempo real e com maior qualidade do que é possível conseguir com a utilização de malhas regulares.

1.1 Trabalhos relacionados

No passado, a geração de animações interativas de escoamentos só era viável com a especificação prévia manual das possíveis características do movimento por um animador, já que não havia poder computacional suficiente para se realizar simulações físicas realistas em tempo real. Um dos primeiros trabalhos a conseguir desempenho interativo com um esquema de integração completa das equações de Navier-Stokes foi apresentado por Foster e Metaxas [4], que utilizaram uma discretização do domínio de simulação em malha regular e integraram as equações do escoamento explicitamente. Apesar de simples e rápido, sua solução sofria de problemas de instabilidade, o que impunha limites significativos no passo de integração, na resolução da malha e na viscosidade do fluido.

Poucos anos mais tarde, Stam [2] desenvolveu uma técnica robusta capaz de realizar uma simulação Euleriana de fluidos incondicionalmente estável para resoluções de malha e passos de tempo arbitrários. Em sua abordagem, as equações de Navier-Stokes são integradas em uma série de passos sequenciais, cada um utilizando os resultados parciais obtidos no anterior. A fim de se garantir estabilidade numérica, a etapa de advecção é realizada com um esquema semi-Lagrangiano implícito, enquanto os termos de difusão e de pressão são integrados separadamente por meio da resolução de equações de Poisson também implícitas. Essa técnica, implementada em GPU por Harris [3], rapidamente se tornou uma referência essencial na área de simulação de fluidos para computação gráfica.

Mais recentemente, muitos autores exploraram uma variedade de aspectos da simulação de escoamentos e fenômenos relacionados. Alguns trabalhos procuraram modelar superfícies livres [6], mudanças de fase [7] e interações entre tipos diferentes de fluidos [8, 9] ou entre um fluido e um sólido rígido ou deformável [10–12]. Outros utilizaram a formulação de fluidos para simular fenômenos especiais, como o fogo [13]. Além disso, são objetos de investigação frequentes as limitações apresentadas pelo método *Stable Fluids*, como o problema de dissipação numérica inerente ao esquema de integração estável [14, 15] e a questão da representação da geometria de fronteiras e obstáculos internos, cuja voxelização simples em uma malha regular não leva a resultados satisfatórios.

Numa tentativa de lidar apropriadamente com formatos arbitrários de fronteiras e obstáculos internos, alguns métodos tentam capturar sua geometria explicitamente modificando os cálculos realizados nas células de borda [16–18]. Entretanto, eles geralmente não conseguem eliminar completamente os

artefatos ou têm dificuldade em avaliar as propriedades do fluido nessas células de forma robusta [11]. Outros métodos tentam capturar detalhes refinados do movimento dos fluidos somente onde necessário por meio de *octrees* [19, 20], células com altura elevada [21] ou outras técnicas de refinamento adaptativo. Porém, apesar de seu valor em termos de aceleração, esse tipo de método ainda está preso a uma representação das fronteiras baseada em blocos alinhados com os eixos. Uma abordagem diferente é a utilização de particionamentos em malhas simpliciais com informações topológicas explícitas [15, 22–24], mas o custo associado geralmente é alto demais para aplicações interativas. Finalmente, também existem abordagens completamente diferentes como a proposta por Batty [11]: em seu trabalho, o passo de projeção é reformulado como um problema de minimização de energia, resultando na eliminação completa de artefatos de alinhamento mesmo com discretizações regulares relativamente grosseiras. Como ponto negativo, a técnica ainda requer manter informações sobre o domínio completo, incluindo muitas células que podem estar marcadas como externas ao fluido, como num caminho em forma da letra "U".

Além de toda essa variedade de métodos, alguns trabalhos se aproximam mais de nossa abordagem ao tentar generalizar a representação do domínio a superfícies tridimensionais com topologia arbitrária [5, 25]. No entanto, eles requerem ou uma discretização em malha de triângulos [25] ou uma implementação de superfícies de subdivisão de Catmull-Clark [26] e sua avaliação exata para quaisquer valores de parâmetros [5, 27]. Ambos casos dependem de conhecimento sobre a topologia da malha. Além disso, a abordagem baseada em superfícies de subdivisão não pode ser facilmente estendida para domínios volumétricos, além de ter que lidar com a sobreposição de regiões vizinhas introduzindo condições de fronteira especiais, regras de atualização e funções de transição para converter quantidades vetoriais entre os espaços paramétricos de diferentes regiões.

Em contraste com as técnicas anteriores, o método proposto neste trabalho é capaz de lidar com superfícies planares ou volumes em geral representados por malhas estruturadas de geometria arbitrária em duas ou três dimensões. Não há necessidade de informações topológicas explícitas nem de funções de transição entre diferentes regiões da malha. Na verdade, precisamos apenas das posições dos nós da malha no espaço do mundo, a partir das quais podemos computar matrizes Jacobianas que relacionam derivadas e grandezas vetoriais nos sistemas de coordenadas do mundo e paramétrico. Isso nos permite integrar as equações de Navier-Stokes de forma eficiente diretamente no espaço paramétrico, levando em conta curvas e deformações

de forma natural, inclusive no caso de obstáculos internos voxelizados. Assim, evitamos também o custo de se trabalhar com malhas simpliciais ou de se refinar excessivamente a malha nas proximidades de fronteiras curvas.

Além das técnicas citadas anteriormente, foi recentemente apresentado por Azevedo [28, 29] um trabalho desenvolvido em paralelo, cuja abordagem é semelhante e de certa forma complementar à utilizada nesta tese. Seu método, assim como o nosso, procura adaptar o algoritmo *Stable Fluids* para trabalhar diretamente no espaço paramétrico de malhas curvilíneas. No entanto, seu tratamento ignora o passo de viscosidade e, por outro lado, tenta ser um pouco mais rigoroso matematicamente com a etapa de projeção das velocidades. Além disso, sua abordagem utiliza uma malha curvilínea ajustada a um obstáculo e sobreposta a uma grade regular de fundo, o que requer um passo de compatibilização de grandezas entre os dois domínios computacionais. Nesta tese, evitamos essa complexidade adicional para conseguirmos uma maior eficiência.

1.2 Organização da tese

O restante deste trabalho está organizado da seguinte forma: No Capítulo 2, revisamos a modelagem teórica do problema de simulação de escoamentos e algumas maneiras relevantes de se discretizar o domínio computacional e de se escolher pontos de amostragem, além de apresentarmos nossa estratégia para converter grandezas entre o espaço do mundo e um sistema de coordenadas paramétricas. No Capítulo 3, relembramos e discutimos o método *Stable Fluids* [2], utilizado como base para o desenvolvimento de nosso trabalho. No Capítulo 4, descrevemos nossa abordagem e como ela modifica cada passo desses algoritmos de modo a levar a maior parte do trabalho para o espaço paramétrico. Isso inclui adaptações a dois solucionadores de equações de Poisson conhecidos: as iterações de Jacobi e o método do Gradiente Biconjugado Estabilizado (BiCGStab) com precondicionador *Multigrid* Algébrico (AM) [30]. No Capítulo 5, apresentamos detalhes de nossa implementação e mostramos como tirar proveito da arquitetura altamente paralela disponível em *hardware* gráfico. No Capítulo 6, apresentamos nossos casos de teste e discutimos os resultados obtidos, que demonstram a corretude e a eficiência de nosso método. Finalmente, no Capítulo 7, concluímos o trabalho e apontamos direções para pesquisas futuras.

2
Modelagem do problema

Iniciamos este capítulo com uma revisão de alguns tópicos fundamentais para o embasamento de nosso trabalho: as equações reduzidas de Navier-Stokes para fluidos incompressíveis [31], a discretização Euleriana do domínio de simulação em uma malha computacional e as diferentes formas de amostragem que podem ser utilizadas para a aplicação de diferenças finitas.

Ao final do capítulo, introduzimos o tipo de domínio curvilíneo em que focamos nossa abordagem e mostramos como calcular e aplicar Jacobianos para relacionar o espaço paramétrico com o do mundo. Nosso objetivo é sermos capazes de converter propriedades do fluido, derivadas posicionais e equações físicas entre os dois sistemas de coordenadas, de modo que a integração possa ser feita diretamente no espaço paramétrico.

2.1
Equações de Navier-Stokes

Um fluido pode ser descrito em três dimensões por um campo vetorial de velocidade $\vec{u} = [u, v, w]^T$ e um campo escalar de pressão q (para distinguir da terceira coordenada do sistema (s, t, p) do espaço paramétrico). Também podemos considerar propriedades como temperatura (T), massa específica (ρ), viscosidade cinemática (ν) e forças externas atuantes (representadas normalmente por sua aceleração resultante $\vec{a}$), constantes ou não ao longo do fluido.

Considerando um fluido incompressível, a evolução dos campos de velocidade e pressão ao longo do tempo é dada pelas equações reduzidas de Navier-Stokes [31, 32]:

$$\nabla \cdot \vec{u} = 0 \tag{2.1}$$

$$\frac{\partial \vec{u}}{\partial t} = -(\vec{u} \cdot \nabla)\,\vec{u} - \frac{1}{\rho}\nabla q + \nu \nabla^2 \vec{u} + \vec{a} \tag{2.2}$$

A expressão (2.1) é uma equação de continuidade associada à conservação de massa. Conhecida também como condição de incompressibilidade, ela indica que o campo de velocidade do fluido é livre de divergência, o que significa que

não pode haver concentração ou dissipação de massa em nenhum ponto. A expressão (2.2), por outro lado, é associada com a conservação de momento. Ela pode ser melhor entendida examinando-se cada um de seus termos: o primeiro, advecção ou convecção, modela como o escoamento carrega qualquer propriedade do fluido consigo, incluindo sua própria velocidade; o segundo, gradiente de pressão, mostra como o fluido tende a ocupar regiões de menor pressão; o terceiro, difusão ou viscosidade, modela o atrito interno, ou seja, como o fluido resiste a mudanças em sua velocidade devido à viscosidade; finalmente, o último termo corresponde à aceleração causada por forças externas, como a gravidade. Note que a equação (2.2) é na verdade vetorial, correspondendo a duas equações em 2D e três em 3D. Além disso, convém relembrar a expansão dos operadores diferenciais relevantes, onde $\vec{u} = [u, v, w]^T$ é um vetor e r um escalar qualquer:

$$\nabla r = \begin{bmatrix} \frac{\partial r}{\partial x} & \frac{\partial r}{\partial y} & \frac{\partial r}{\partial z} \end{bmatrix}^T \qquad \nabla \cdot \vec{u} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}$$

$$\nabla^2 r = \frac{\partial^2 r}{\partial x^2} + \frac{\partial^2 r}{\partial y^2} + \frac{\partial^2 r}{\partial z^2} \qquad (\vec{u} \cdot \nabla) r = \frac{\partial r}{\partial x} u + \frac{\partial r}{\partial y} v + \frac{\partial r}{\partial z} w$$

Infelizmente, as equações (2.1) e (2.2) não possuem solução analítica para o caso geral, e sua integração por métodos numéricos revela-se uma tarefa bastante complexa, apresentando problemas de precisão e estabilidade difíceis de superar [4]. Por isso, diversas técnicas já foram propostas com abordagens diferentes para todas as fases do processo, incluindo a discretização do domínio computacional, a localização das amostras das propriedades do fluido, métodos de interpolação e integração e estratégias para se tentar garantir características desejáveis específicas. Além disso, como estamos lidando com fluidos incompressíveis, as equações apresentam em geral um comportamento matemático do tipo elíptico, em que a solução em qualquer ponto do domínio depende do estado presente e passado de todos os outros pontos [33]. Isso significa que qualquer algoritmo deve incluir um passo de elevado custo computacional, normalmente a resolução de um grande sistema com número de incógnitas proporcional à quantidade de células na malha discretizada. Assim, o desenvolvimento de solucionadores rápidos e robustos ainda é considerado um problema em aberto.

2.2 Discretização do domínio

Na simulação de fluidos pela abordagem Euleriana, o primeiro passo é discretizar o domínio de simulação em uma malha computacional para a apli-

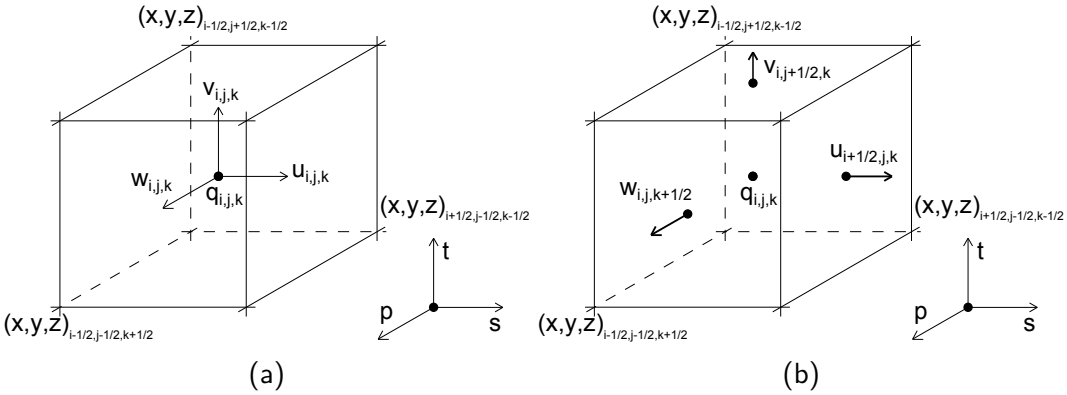


Figura 2.1 – Amostragem de velocidade e pressão para a célula (i, j, k) em um domínio tridimensional: (a) malha colocalizada nos centros; (b) malha deslocada nas faces.

cação de algum método numérico. Neste trabalho, consideramos inicialmente uma grade regular: em três dimensões, cada célula é identificada por um índice $[i, j, k]^T$, e todas têm o mesmo tamanho $[\delta_x, \delta_y, \delta_z]^T$.

Queremos aproximar as derivadas espaciais das propriedades do fluido por diferenças finitas. Para isso, podemos tomar amostras dessas propriedades em diferentes locais, como nos centros ou nas faces das células. Primeiramente, consideramos uma amostragem colocalizada, em que todas as grandezas envolvidas são avaliadas nos centros $\vec{x} = [x, y, z]^T$ das células, como mostrado na Figura 2.1a. Tomando diferenças centrais de segunda ordem, temos as seguintes derivadas discretas para um campo escalar r na direção x:

$$\frac{\partial r}{\partial x} = \frac{1}{2\delta_x}\left[r\left(x+\delta_x\right) - r\left(x-\delta_x\right)\right] + O\left(\delta_x^2\right)$$
$$\frac{\partial^2 r}{\partial x^2} = \frac{1}{\left(2\delta_x\right)^2}\left[r\left(x+2\delta_x\right) - 2r\left(x\right) + r\left(x-2\delta_x\right)\right] + O\left(\delta_x^2\right)$$

Descartando os termos de alta ordem e adotando uma notação mais compacta baseada em índices, temos para a célula (i, j, k):

$$\left.\frac{\partial r}{\partial x}\right|_{i,j,k} \approx \frac{1}{2\delta_x}\left[r_{i+1,j,k} - r_{i-1,j,k}\right] \tag{2.3}$$
$$\left.\frac{\partial^2 r}{\partial x^2}\right|_{i,j,k} \approx \frac{1}{\left(2\delta_x\right)^2}\left[r_{i+2,j,k} - 2r_{i,j,k} + r_{i-2,j,k}\right] \tag{2.4}$$

Note que, na equação (2.3), a derivada de r na célula (i, j, k) não leva em conta o valor da grandeza na própria célula. Isso significa que, se $r_{i+1,j,k} = r_{i-1,j,k}$, por exemplo, teremos $\partial r/\partial x = 0$ mesmo que $r_{i,j,k}$ seja diferente. Assim, variações espaciais em r com período $2\delta_x$ são completamente ignoradas. Numa simulação de escoamento, ao lidarmos com operadores como a divergência do campo de velocidade e o gradiente de pressão, aplicar

essa equação pode resultar em uma resposta com componentes oscilatórias indesejáveis que podem ter grande influência sobre o comportamento do fluido.

Observe agora a equação (2.4). Pode-se perceber que o estêncil dessa segunda derivada ignora as amostras vizinhas à da célula central. Aplicando esse tipo de operador a todas as células na discretização de uma equação de Poisson $\nabla^2 r = f$ em três dimensões, chegaríamos a um sistema com oito conjuntos totalmente desacoplados de equações, que resultariam então em oito soluções distintas: os valores de r em qualquer grupo $(2 \times 2 \times 2)$ de células seriam totalmente independentes entre si. Como utilizaremos uma equação de Poisson desse tipo para encontrar as pressões do fluido, essa característica é inaceitável. Uma maneira simples de evitarmos esse problema e obtermos uma solução única é reduzir artificialmente o tamanho do estêncil da equação (2.4) de modo a utilizar a célula central e suas duas vizinhas diretas. No entanto, pode-se mostrar que a expressão resultante corresponde a tomarmos diferenças progressivas e atrasadas de primeira ordem para a pressão na equação (2.2), exigindo a utilização de malhas muito refinadas para a obtenção de resultados com precisão comparável às diferenças centrais de segunda ordem [34].

Todos os problemas descritos acima para a malha colocalizada podem ser evitados com a utilização de uma amostragem *deslocada*[1], em que as componentes de grandezas vetoriais são armazenadas nas faces ortogonais correspondentes. Esse esquema, ilustrado na Figura 2.1b, foi originalmente proposto como parte do método *Marker-and-Cell* (MAC) para o tratamento de superfícies livres [35], mas é aplicável a simulações eulerianas em geral. A amostragem dá origem às seguintes equações para a divergência da velocidade (ou de outro campo vetorial) avaliada no centro de cada célula:

$$\begin{aligned}
\left.\frac{\partial u}{\partial x}\right|_{i,j,k} &\approx \frac{1}{\delta_x}\left[u_{i+1/2,j,k} - u_{i-1/2,j,k}\right] \\
\left.\frac{\partial v}{\partial y}\right|_{i,j,k} &\approx \frac{1}{\delta_y}\left[v_{i,j+1/2,k} - v_{i,j-1/2,k}\right] \\
\left.\frac{\partial w}{\partial z}\right|_{i,j,k} &\approx \frac{1}{\delta_z}\left[w_{i,j,k+1/2} - w_{i,j,k-1/2}\right]
\end{aligned} \tag{2.5}$$

De forma semelhante, a utilização da malha deslocada resulta nas seguintes equações para o gradiente de um campo escalar r com cada componente

[1] *Staggered* em inglês. Também conhecida como malha *desencontrada* ou *diferenciada* em português.

avaliada na face ortogonal correspondente:

$$\begin{aligned}
\left.\frac{\partial r}{\partial x}\right|_{i+1/2,j,k} &\approx \frac{1}{\delta_x}\left[r_{i+1,j,k} - r_{i,j,k}\right] \\
\left.\frac{\partial r}{\partial y}\right|_{i,j+1/2,k} &\approx \frac{1}{\delta_y}\left[r_{i,j+1,k} - r_{i,j,k}\right] \\
\left.\frac{\partial r}{\partial z}\right|_{i,j,k+1/2} &\approx \frac{1}{\delta_z}\left[r_{i,j,k+1} - r_{i,j,k}\right]
\end{aligned} \tag{2.6}$$

Finalmente, temos ainda a seguinte expressão para a derivada segunda de um campo escalar r na direção x, avaliada no centro de cada célula:

$$\left.\frac{\partial^2 r}{\partial x^2}\right|_{i,j,k} \approx \frac{1}{\delta_x^2}\left[r_{i+1,j,k} - 2r_{i,j,k} + r_{i-1,j,k}\right] \tag{2.7}$$

As equações (2.5) a (2.7) serão usadas em nosso algoritmo como parte do passo de projeção das velocidades, responsável por garantir a conservação de massa e a incompressibilidade do escoamento. Note que foram utilizadas diferenças centrais de segunda ordem em todos os casos, e que a segunda derivada do campo escalar não ignora as células vizinhas à central, de modo que será possível encontrar uma solução única, de segunda ordem e não oscilatória para o campo de pressão.

Note também que, na amostragem deslocada, as componentes (u, v, w) da velocidade são armazenadas em diferentes posições da malha. Isso significa que, se precisarmos da velocidade completa em qualquer ponto, precisaremos sempre fazer interpolações, o que aumenta o custo computacional do simulador.

Neste trabalho, consideramos tanto uma malha colocalizada quanto uma deslocada. No Capítulo 6, comparamos a eficiência e a qualidade dos resultados obtidos com as duas abordagens. Como veremos, a malha colocalizada pode ser capaz de gerar uma aproximação suficientemente boa para aplicações interativas, mas os resultados mais robustos da malha deslocada podem ser desejados quando o aumento do custo computacional for aceitável.

2.3 Separação de operadores

Na resolução numérica de equações diferenciais complexas como as de Navier-Stokes, é comum a utilização de uma estratégia do tipo "dividir e conquistar" por meio da aplicação de técnicas conhecidas como "separação de operadores" (*operator splitting* em inglês). A idéia é dividir o problema complexo inicial em uma sequência de subproblemas mais simples, cujas soluções podem ser obtidas de maneira mais fácil e rápida. Ao longo do

processo, as contribuições de cada solução intermediária são somadas ao resultado final segundo alguma regra, o que implica em um erro numérico que pode ser devidamente estimado.

Normalmente, esse tipo de abordagem é utilizado de duas formas principais: primeiramente, pode-se tentar separar um operador diferencial de modo que cada subproblema envolva derivadas parciais em termos de apenas um dos eixos coordenados (separação de coordenadas ou *coordinate splitting* em inglês). Alternativamente, pode-se quebrar uma equação diferencial de maneira que cada subproblema represente um único fenômeno físico isolado, como por exemplo advecção ou difusão (integração por passos fracionados ou *fractional step method* em inglês).

De qualquer maneira, a estratégia é capaz de gerar métodos altamente eficientes, já que cada parte do operador original pode ser tratada independentemente com a utilização de algoritmos especializados. É possível, por exemplo, empregar integradores de diferentes ordens ou passos de tempo, trabalhar com diferentes discretizações espaciais, ou tratar alguns subproblemas de forma explícita e outros de forma implícita. É interessante notar que, apesar de apresentar um grande potencial acelerador, até recentemente esse tipo de técnica ainda era relativamente pouco explorado na área de computação gráfica interativa [32].

Para ilustrarmos a separação de operadores, considere a seguinte equação, onde $r(t)$ é um campo escalar que varia com o tempo t, A e B são funções ou operadores diferenciais lineares, e índices sobrescritos indicam passos interativos em vez de expoentes:

$$\frac{\partial r(t)}{\partial t} = A(r(t)) + B(r(t)), \text{ com } t \in [0, T], r(0) = r^0 \tag{2.8}$$

A notação a seguir [36] representa a resolução direta do problema de valor inicial acima, separando-se o intervalo de tempo $[0, T]$ em uma sequência de subintervalos $[t^n, t^{n+1}]$, com $n = 0, 1, ..., N-1$, $t^0 = 0$ e $t^N = T$:

$$\frac{\partial r(t)}{\partial t} = A(r(t)) + B(r(t)), \text{ com } t \in [t^n, t^{n+1}], r(t^n) = r^n \tag{2.9}$$

Utilizando essa mesma notação, podemos aplicar uma separação de operadores aditiva como a seguir, onde os índices subscritos identificam campos escalares auxiliares:

$$\begin{aligned} \frac{\partial r_A(t)}{\partial t} &= A(r_A(t)), \text{ com } t \in [t^n, t^{n+1}], r_A(t^n) = r^n \\ \frac{\partial r_B(t)}{\partial t} &= B(r_B(t)), \text{ com } t \in [t^n, t^{n+1}], r_B(t^n) = r^n \\ r^{n+1} &= r_A(t^{n+1}) + r_B(t^{n+1}) - r^n \end{aligned} \tag{2.10}$$

Note que, na equação(2.10), ambos operadores são aplicados com as mesmas condições iniciais. Alternativamente, podemos utilizar um esquema do tipo *Lie-Trotter* em que a condição inicial utilizada por cada operador é dada pelo resultado parcial obtido pelo anterior:

$$\frac{\partial r_A(t)}{\partial t} = A(r_A(t)), \text{ com } t \in [t^n, t^{n+1}], r_A(t^n) = r^n \tag{2.11}$$

$$\frac{\partial r_B(t)}{\partial t} = B(r_B(t)), \text{ com } t \in [t^n, t^{n+1}], r_B(t^n) = r_A(t^{n+1}) \tag{2.12}$$

$$r^{n+1} = r_B(t^{n+1}) \tag{2.13}$$

Pode-se mostrar [36] que o resultado obtido por (2.10) e (2.11) é exato quando os operadores A e B são comutativos, mas que apresenta acurácia limitada à primeira ordem caso contrário, mesmo que sejam utilizados integradores de maior ordem na resolução de cada subproblema. Por outro lado, é possível construir esquemas de separação de operadores de segunda ordem ou maior, mas tal nível de acurácia não é necessário para os resultados aproximados em que estamos interessados.

2.4 Matriz Jacobiana

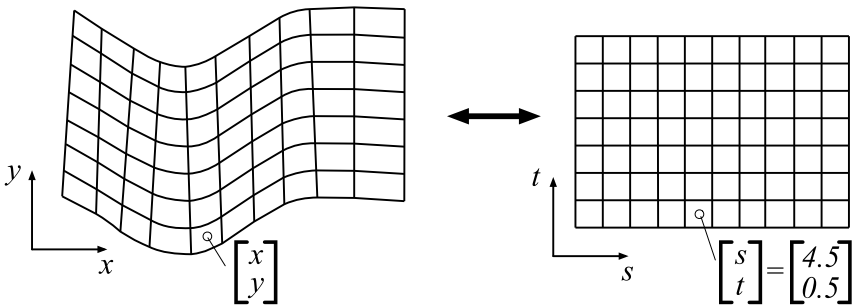


Figura 2.2 – Relacionando coordenadas nos espaços do mundo (x, y) e paramétrico (s, t) para o caso bidimensional.

Como mencionado anteriormente, temos como objetivo simular escoamentos de fluidos em malhas estruturadas paramétricas com formato suave arbitrário. Até agora, no entanto, consideramos apenas discretizações em grades regulares. O ponto fundamental a se perceber é que, na verdade, existe uma estreita relação entre esses dois casos. Como ilustrado na Figura 2.2, qualquer malha estruturada paramétrica no espaço (x, y, z) do mundo pode ser diretamente associada a uma grade uniforme simples no espaço (s, t, p) paramétrico. Isso sugere que a integração das equações reduzidas de Navier-Stokes deve poder ser realizada de forma bastante simplificada, desde que trabalhemos diretamente no espaço paramétrico.

Voltando nossa atenção para (2.1) e (2.2), podemos perceber que o comportamento de um fluido é descrito apenas em termos de velocidades e derivadas espaciais de campos escalares, não apresentando nenhuma dependência direta de dados posicionais. Assim, para transformar as equações do espaço do mundo para o paramétrico, basta relacionarmos essas velocidades e derivadas nos dois sistemas de coordenadas. Considerando que o mapeamento entre os espaços seja uma bijeção contínua no interior do domínio de simulação, podemos estabelecer as seguintes relações, onde f_x denota a derivada do campo escalar f na direção x e $\dot{s}$ denota a velocidade do fluido ou outra grandeza vetorial tomada na direção s:

$$\begin{aligned} f_s &= f_x x_s + f_y y_s + f_z z_s & \dot{s} &= s_x\dot{x} + s_y\dot{y} + s_z\dot{z} \\ f_t &= f_x x_t + f_y y_t + f_z z_t & \dot{t} &= t_x\dot{x} + t_y\dot{y} + t_z\dot{z} \\ f_p &= f_x x_p + f_y y_p + f_z z_p & \dot{p} &= p_x\dot{x} + p_y\dot{y} + p_z\dot{z} \end{aligned} \tag{2.14}$$

De maneira similar, também podemos estabelecer as seguintes relações inversas, que nos permitem transformar campos vetoriais e derivadas de campos escalares do espaço paramétrico de volta para o espaço do mundo:

$$\begin{aligned} f_x &= f_s s_x + f_t t_x + f_p p_x & \dot{x} &= x_s\dot{s} + x_t\dot{t} + x_p\dot{p} \\ f_y &= f_s s_y + f_t t_y + f_p p_y & \dot{y} &= y_s\dot{s} + y_t\dot{t} + y_p\dot{p} \\ f_z &= f_s s_z + f_t t_z + f_p p_z & \dot{z} &= z_s\dot{s} + z_t\dot{t} + z_p\dot{p} \end{aligned} \tag{2.15}$$

As equações (2.14) e (2.15) podem ser reescritas de forma mais compacta utilizando a seguinte notação matricial, onde $\vec{v}_{(s,t,p)}$ denota o valor de um campo vetorial $\vec{v}$ no sistema de coordenadas (s,t,p), e J^{-T} representa de forma compacta a inversa transposta de J:

$$[\nabla f]_{(s,t,p)} = J\,[\nabla f]_{(x,y,z)} \qquad \vec{v}_{(s,t,p)} = J^{-T}\vec{v}_{(x,y,z)} \tag{2.16}$$

$$[\nabla f]_{(x,y,z)} = J^{-1}\,[\nabla f]_{(s,t,p)} \qquad \vec{v}_{(x,y,z)} = J^{T}\vec{v}_{(s,t,p)} \tag{2.17}$$

Nas equações anteriores, a matriz J representa o Jacobiano do mapeamento entre os espaços do mundo e paramétrico, definido a seguir:

$$J = \begin{pmatrix} x_s & y_s & z_s \\ x_t & y_t & z_t \\ x_p & y_p & z_p \end{pmatrix} \qquad J^{-1} = \begin{pmatrix} s_x & t_x & p_x \\ s_y & t_y & p_y \\ s_z & t_z & p_z \end{pmatrix} \tag{2.18}$$

Os elementos da matriz Jacobiana e de sua inversa, também conhecidos como métricas, dependem apenas das geometrias do domínio e da malha com-

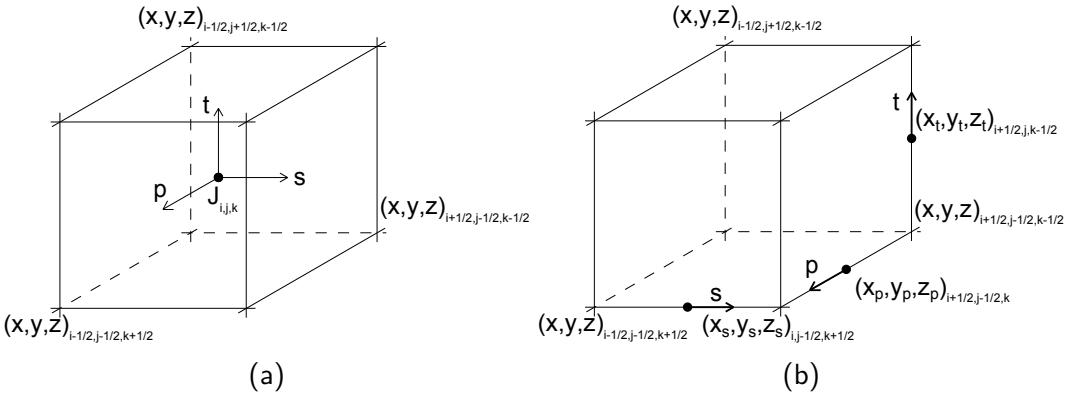


Figura 2.3 – Amostragem do Jacobiano para a célula (i, j, k) em um domínio tridimensional: (a) malha colocalizada no centro; (b) malha deslocada nas arestas.

putacional no mundo físico. Assim, considerando que a discretização espacial se mantém sempre inalterada, esses elementos podem ser pré-computados uma única vez pelo solucionador e consultados quando necessário. Por outro lado, é importante perceber que o Jacobiano representa uma geometria curva e varia continuamente no espaço. Por isso, as métricas devem ser amostradas ao longo da malha e interpoladas sempre que seus valores não forem conhecidos no ponto requisitado.

Neste trabalho, investigamos duas maneiras de se amostrar e calcular as métricas do Jacobiano por diferenças finitas ao longo do domínio, tendo como dados de entrada as posições dos nós da discretização. Primeiramente, consideramos uma malha colocalizada com amostras nos centros das células, em que todas as métricas são calculadas nesse mesmo ponto, conforme ilustrado na Figura 2.3a. As equações a seguir mostram como obter cada linha da matriz J:

$$\begin{aligned} \vec{x}_s|_{i,j,k} &= \vec{x}_{i+1/2,j,k} - \vec{x}_{i-1/2,j,k} \\ \vec{x}_t|_{i,j,k} &= \vec{x}_{i,j+1/2,k} - \vec{x}_{i,j-1/2,k} \\ \vec{x}_p|_{i,j,k} &= \vec{x}_{i,j,k+1/2} - \vec{x}_{i,j,k-1/2} \end{aligned} \tag{2.19}$$

Note que, em (2.19), estamos utilizando os centros das faces das células para calcular o Jacobiano. Poderíamos também escolher os centros das células vizinhas em cada direção, o que resultaria em variações mais suaves das métricas, uma característica que pode ser desejável em algumas discretizações.

Seguindo uma estratégia distinta, consideramos também uma malha deslocada com amostras nas arestas das células, como mostra a Figura 2.3b. Para entender essa abordagem, basta perceber que o vetor definido por uma aresta da malha corresponde precisamente às variações das componentes de $\vec{x}$ ao longo de uma única coordenada paramétrica. Tais variações, por sua

vez, podem ser associadas diretamente às derivadas que aparecem em uma das linhas da matriz J em (2.18). Assim, parece natural escolher o ponto médio de cada aresta da malha para amostrar as métricas correspondentes. Note que, dessa maneira, cada linha de J é calculada num ponto diferente do domínio, e em qualquer ponto será sempre necessário interpolar alguns ou todos os termos do Jacobiano. As equações seguintes formalizam o método utilizado:

$$\vec{x}_s|_{i,j+1/2,k+1/2} = \vec{x}_{i+1/2,j+1/2,k+1/2} - \vec{x}_{i-1/2,j+1/2,k+1/2}$$
$$\vec{x}_t|_{i+1/2,j,k+1/2} = \vec{x}_{i+1/2,j+1/2,k+1/2} - \vec{x}_{i+1/2,j-1/2,k+1/2}$$
$$\vec{x}_p|_{i+1/2,j+1/2,k} = \vec{x}_{i+1/2,j+1/2,k+1/2} - \vec{x}_{i+1/2,j+1/2,k-1/2}$$

É importante ressaltar que, apesar dos diferentes padrões de amostragem propostos para o Jacobiano, utilizamos sempre os centros das células para calcular e armazenar as métricas da inversa J^{-1}. Essa escolha é justificada pelo fato de que a inversão de J só é possível onde a matriz está completamente determinada, o que só ocorre a partir dos centros das células de fronteira do domínio. De forma semelhante, os termos de segunda ordem que serão necessários nas Seções 4.2 e 4.4 são igualmente pré-calculados por diferenças finitas numa malha centrada.

3
Método numérico em malha regular

A integração correta das equações de Navier-Stokes (2.1) e (2.2) revela-se uma tarefa bastante complexa, apresentando elevado custo computacional e problemas de precisão e estabilidade difíceis de contornar [4]. Apesar dessas dificuldades, o método *Stable Fluids* [2] foi a primeira abordagem capaz de gerar um comportamento de escoamento aproximadamente correto de maneira altamente eficiente e incondicionalmente estável para resoluções de malha e passos de tempo arbitrários.

As características de eficiência e estabilidade do *Stable Fluids* são fundamentais para aplicações interativas e de tempo real. Por isso, o método numérico que propomos para nosso integrador neste trabalho é baseado em uma versão moderna desse algoritmo, que inclui aprimoramentos recentes de diversos trabalhos, reunidos em [32].

Neste capítulo, apresentamos o método *Stable Fluids* moderno, considerando sua aplicação original para malhas regulares. Ao longo do texto, discutimos algumas formulações e características do algoritmo que serão aproveitadas, exploradas e adaptadas quando abordarmos malhas curvilíneas. Uma explicação mais detalhada de técnicas Eulerianas modernas de alto desempenho para computação gráfica e aplicações interativas em geral pode ser encontrada em [32].

3.1
Stable Fluids

O método *Stable Fluids* emprega um modelo físico simplificado em que as equações de Navier-Stokes são resolvidas por meio da aplicação sequencial de quatro operadores: advecção (A), difusão (D), forças externas (F) e projeção (P). Cada passo utiliza as velocidades intermediárias computadas no anterior:

$$\vec{u}_{i,j,k}^{n+1} = P \circ F \circ D \circ A \left(\vec{u}_{i,j,k}^{n} \right) \tag{3.1}$$

Note que a abordagem adotada pelo método corresponde a uma separação de operadores do tipo *Lie-Trotter*, descrita na Seção 2.3. Mais precisa-

mente, queremos resolver os seguintes subproblemas de forma independente:

$$\frac{\partial \vec{u}}{\partial t} = A(\vec{u}) = -\left(\vec{u}\cdot \nabla\right)\vec{u} \tag{3.2}$$

$$\frac{\partial \vec{u}}{\partial t} = D(\vec{u}) = \nu\nabla^2\vec{u} \tag{3.3}$$

$$\frac{\partial \vec{u}}{\partial t} = F(\vec{u}) = \vec{a} \tag{3.4}$$

$$\frac{\partial \vec{u}}{\partial t} = P(\vec{u}) = -\frac{1}{\rho}\nabla q, \text{ com } \nabla\cdot\vec{u} = 0 \tag{3.5}$$

É importante perceber que, ao separarmos os operadores conforme descrito acima, os campos de velocidade intermediários obtidos após cada um dos primeiros passos não serão necessariamente livres de divergência. Como veremos na Seção 3.5, o operador de projeção P será aplicado de forma corretiva, com o cálculo de pressões que garantam que a condição de incompressibilidade (2.1) seja satisfeita ao final de cada ciclo.

Perceba também que a ordem em que os operadores são aplicados é relevante para o algoritmo. Em especial, para que o fluido conserve massa e volume corretamente, é de fundamental importância que a advecção seja aplicada sempre sobre um campo de velocidades livre de divergência, o que é garantido pela ordem descrita acima.

Quanto à acurácia do método, salienta-se que a separação de operadores proposta é de primeira ordem, como vimos na Seção 2.3, o que limita o ganho de qualidade que podemos obter com melhorias isoladas na resolução de cada etapa. Por outro lado, a separação permite que o algoritmo apresente a valiosa característica de estabilidade incondicional, desde que formulações implícitas e outras técnicas estáveis sejam utilizadas em todas as etapas. Isso quer dizer que, mesmo para malhas pouco refinadas e passos de tempo grandes, o comportamento resultante do escoamento pode se tornar pouco correto fisicamente, mas a simulação não deve "explodir" numericamente.

A seguir, descrevemos brevemente como os operadores em (3.1) são aplicados a grades regulares. Nos referiremos a esses operadores quando apresentarmos nossas adaptações para trabalharmos no espaço paramétrico no Capítulo 4. Na equação (3.1) e no restante deste capítulo, consideramos sempre um passo de tempo h e utilizamos as notações r^n e r' para designar o valor de uma grandeza r no passo de tempo n e após a aplicação de cada operador, respectivamente.

3.2 Advecção

Relembre a equação (3.2) do operador de advecção, reescrita para um escalar qualquer r, que pode representar até mesmo cada uma das componentes da velocidade $\vec{u}$ do fluido:

$$\frac{\partial r}{\partial t} = -(\vec{u} \cdot \nabla) r$$

Expandindo o operador diferencial no espaço:

$$\frac{\partial r}{\partial t} = -\left(\frac{\partial r}{\partial x}u + \frac{\partial r}{\partial y}v + \frac{\partial r}{\partial z}w\right) \tag{3.6}$$

Normalmente, poderíamos tentar resolver a equação acima utilizando diferenças finitas centrais no espaço e um integrador da ordem desejada no tempo. Infelizmente, porém, essa abordagem apresenta alguns problemas [32]: primeiro, pode-se mostrar que um integrador de Euler explícito é incondicionalmente *instável* para essa equação. Por outro lado, mesmo que aplicássemos um integrador exato para o tempo, as diferenças centrais tomadas no espaço acarretam o mesmo problema que nos levou a definir a malha deslocada na Seção 2.2: variações espaciais de alta frequência na grandeza sendo integrada não são capturadas pelas diferenças. Assim, a advecção acaba separando artificialmente os componentes de baixa e alta frequência presentes no campo, gerando artefatos e comportamentos indesejados na simulação.

Para contornar os problemas descritos acima, o algoritmo *Stable Fluids* utiliza uma abordagem diferente, mais simples, conhecida como semi-Lagrangiana. Em um método puramente Lagrangiano, o fluido seria simulado como um sistema de partículas em movimento, cada uma situada numa posição $\vec{x}_P$ e possuindo um conjunto de propriedades, como sua velocidade $\vec{u}_P$ e alguns valores escalares. Ao longo de um passo de advecção, simplesmente determinaríamos a trajetória de todas as partículas e assumiríamos que cada uma levaria consigo todas as suas propriedades. Utilizando esse raciocínio, podemos evitar lidar diretamente com a equação (3.6). Primeiro, imaginamos uma partícula que, ao final do passo de tempo atual, se encontraria exatamente sobre uma amostra da malha. Em seguida, assumimos que os novos valores das propriedades do fluido nesse local devem ser aqueles trazidos pela partícula a partir de sua posição anterior.

Para entender melhor o método semi-Lagrangiano, observe a Figura 3.1 para o caso 2D. Considere uma propriedade r do fluido amostrada no ponto A com valor r_A. Salienta-se que essa propriedade pode ser, por exemplo, uma das componentes da própria velocidade do fluido. Queremos determinar o valor r'_A após um passo de tempo h. Imagine agora uma partícula sem massa na

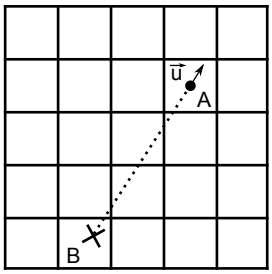


Figura 3.1 – Advecção semi-Lagrangiana para um ponto amostral A onde a velocidade do fluido é $\vec{u}$. Considerando uma partícula que termina o passo de tempo exatamente em A, geramos uma trajetória para trás (linha tracejada) até sua posição inicial B. As propriedades em B são então interpoladas linearmente e copiadas para A.

posição $\vec{x}_A$ da amostra, onde a velocidade do fluido é $\vec{u}_A$. Numa aproximação de primeira ordem, se essa partícula fosse carregada pelo escoamento, ela chegaria a uma posição $(\vec{x}_A + \vec{u}_A h)$, levando consigo a propriedade r. De forma semelhante, podemos pensar que, se uma partícula chegará exatamente até a posição $\vec{x}_A$, ela deve estar vindo de $\vec{x}_B = (\vec{x}_A - \vec{u}_A h)$ no passo de tempo atual. Note que utilizamos a velocidade final da partícula para caminharmos para trás ao longo de sua trajetória. Terminado esse processo, podemos consultar a propriedade r_B no local encontrado e atualizar a amostra original com seu valor:

$$r'_A = r_B = r\left(\vec{x}_B\right) = r\left(\vec{x}_A - \vec{u}_A h\right) \tag{3.7}$$

Alternativamente, de forma um pouco mais genérica:

$$r\left(\vec{x}_A, t + h\right) = r\left(\vec{x}_A - \vec{u}\left(\vec{x}_A, t\right) h, t\right)$$

É importante perceber que, apesar de termos considerado uma integração de primeira ordem para a trajetória das partículas, podemos igualmente utilizar técnicas mais acuradas nesse processo. Em geral, é desejável a aplicação de um método Runge-Kutta de segunda ordem para que as trajetórias sigam corretamente o campo de velocidades. Métodos de maior ordem também podem ajudar, mas deve-se lembrar que seu benefício é limitado devido ao erro decorrente da separação de operadores (Seção 2.3).

De qualquer maneira, uma vez encontrada a posição de origem de uma partícula, é necessário fazer uma interpolação da propriedade de interesse utilizando as amostras da malha mais próximas. Em princípio, uma interpolação trilinear é suficiente para a obtenção de um comportamento aproximadamente correto. Essa abordagem tem duas vantagens interessantes: primeiro, apresenta um custo de computação muito baixo, especialmente quando utilizando *hardware* gráfico especializado. Segundo, o resultado é limitado por construção, mantendo-se obrigatoriamente dentro do intervalo de valores já presentes na simulação. Com isso, garante-se a estabilidade incondicional do método.

Infelizmente, a interpolação trilinear simples também apresenta uma desvantagem importante: como seu resultado é sempre uma média de valores da malha, tende-se a suavizar componentes de alta frequência do campo. No caso das velocidades do fluido, verifica-se um comportamento semelhante à atuação de uma viscosidade, que com o tempo pode eliminar pequenos vórtices e outras características interessantes do escoamento. Por isso, esse fenômeno é conhecido como dissipação numérica.

Existem diferentes técnicas para se tentar reduzir o problema de dissipação numérica. Merece menção uma técnica simples e razoavelmente eficaz: a substituição da interpolação linear pela de Catmull-Rom [14]. A idéia é construir um polinômio cúbico que satisfaça simultaneamente os valores do campo e suas derivadas nos extremos do intervalo considerado. Assim, em uma dimensão, para obtermos o valor de $r(x)$ no ponto $x \in [x_i, x_{i+1}]$, impomos:

$$\begin{aligned} r\left(x_i\right) &= r_i & r\left(x_{i+1}\right) &= r_{i+1} \\ \frac{dr}{dx}\left(x_i\right) &= \frac{r_{i+1} - r_{i-1}}{2\delta_x} & \frac{dr}{dx}\left(x_{i+1}\right) &= \frac{r_{i+2} - r_i}{2\delta_x} \end{aligned}$$

Note que as derivadas são estimadas por diferenças centrais utilizando os próprios valores do campo. Pode-se verificar que polinômio resultante tem a seguinte forma, onde $\bar{x} = (x - x_i)/\delta_x$:

$$\begin{aligned} r(x) = r_{i-1}\left(-\frac{1}{2}\bar{x} + \bar{x}^2 - \frac{1}{2}\bar{x}^3\right) + r_i\left(1 - \frac{5}{2}\bar{x}^2 + \frac{3}{2}\bar{x}^3\right) \\ + r_{i+1}\left(\frac{1}{2}\bar{x} + 2\bar{x}^2 - \frac{3}{2}\bar{x}^3\right) + r_{i+2}\left(-\frac{1}{2}\bar{x}^2 + \frac{1}{2}\bar{x}^3\right) \end{aligned} \tag{3.8}$$

O polinômio acima tem precisão de segunda ordem, oferecendo geralmente melhores resultados do que uma interpolação linear. Por outro lado, seu valor não está restrito ao intervalo de valores pré-existentes em r. Assim, para se garantir a estabilidade do método, torna-se preciso limitar artificialmente o resultado.

Note também que, ao trabalharmos com três dimensões, a expressão acima deve ser utilizada múltiplas vezes cobrindo todas as direções, de forma análoga a uma interpolação trilinear. Isso quer dizer que são necessárias 64 amostragens do campo r ao longo do processo, o que pode prejudicar significativamente o desempenho de um algoritmo.

3.3 Difusão

Relembre a equação (3.3) do operador de difusão, onde ν é uma viscosidade cinemática considerada constante ao longo de todo o fluido:

$$\frac{\partial \vec{u}}{\partial t} = \nu \nabla^2 \vec{u}$$

Deve-se ressaltar que o *Stable Fluids* trata a difusão de forma bastante simplificada. A equação acima, além de exigir um fluido de viscosidade constante, só é válida quando aplicada sobre um campo de velocidade livre de divergência [32]. Em princípio isso não deveria ser um problema, já que estamos considerando apenas fluidos incompressíveis. No entanto, devemos lembrar que, com a separação de operadores que estamos utilizando, os campos de velocidade intermediários ao longo de um passo de tempo não garantem essa propriedade. Isso gera um conflito de requisitos, pois tanto a advecção quanto a difusão precisariam ser realizados imediatamente após o passo de projeção. Essa questão é ignorada no *Stable Fluids* original.

Para se tratar adequadamente o problema, é comum na literatura o estudo da chamada equação de Stokes, que considera simultaneamente os termos de viscosidade e pressão. Na área de computação gráfica e algoritmos interativos, porém, essa complexidade costuma ser evitada. Em vez disso, adota-se um procedimento bastante aproximado: aplica-se a advecção após a difusão, mas ainda considerando as velocidades livres de divergência resultantes do último passo de projeção. Os valores pós-difusão são utilizados apenas no momento de se atribuir a uma amostra o valor de velocidade encontrado na posição de origem de uma partícula Lagrangiana [32]. Neste trabalho, adotamos essa abordagem.

Voltando nossa atenção à resolução do operador de difusão, o método *Stable Fluids* utiliza uma formulação implícita para garantir a estabilidade [2]:

$$\left(I - \nu h \nabla^2\right) \vec{u}'_{i,j,k} = \vec{u}_{i,j,k} \tag{3.9}$$

Em (3.9), temos uma equação de Poisson, em que cada componente de velocidade na verdade representa um sistema linear esparso simétrico com uma incógnita para cada célula da malha. Se discretizarmos a equação por diferenças centrais numa malha colocalizada, podemos encontrar a expressão abaixo para $\vec{u}$. Note que a versão apresentada aqui é mais geral do que a encontrada em [2], que assume $\delta_x = \delta_y = \delta_z$:

$$\begin{aligned} \vec{u}'_{i,j,k}\left[\beta + 2\left(k_{xx} + k_{yy} + k_{zz}\right)\right] = \alpha f + \\ \left(\vec{u}'_{i+1,j,k} + \vec{u}'_{i-1,j,k}\right) k_{xx} + \\ \left(\vec{u}'_{i,j+1,k} + \vec{u}'_{i,j-1,k}\right) k_{yy} + \\ \left(\vec{u}'_{i,j,k+1} + \vec{u}'_{i,j,k-1}\right) k_{zz} \end{aligned} \tag{3.10}$$

onde:

$$k_{xx} = 1/\delta_x^2 \qquad \alpha = 1/\nu h$$
$$k_{yy} = 1/\delta_y^2 \qquad \beta = 1/\nu h$$
$$k_{zz} = 1/\delta_z^2 \qquad f = \vec{u}_{i,j,k}$$

Note que o sistema descrito pela equação (3.10) pode ser aplicado de forma independente a cada uma das componentes da velocidade. Além disso, se estivermos considerando uma malha deslocada, a expressão resultante é análoga, bastando-se ajustar os índices de forma apropriada em cada dimensão.

Um sistema como (3.10) pode ser resolvido por uma grande variedade de algoritmos, mas abordagens modernas tendem a dar preferência ao método do Gradiente Conjugado com precondicionador de Cholesky Incompleto ou *Multigrid* [2, 32, 37], que apresentam excelentes características de convergência. Além disso, para estabelecermos uma base de comparação para testes de precisão, estabilidade e desempenho, também consideraremos o método de iterações de Jacobi descrito em [3], que é facilmente paralelizável mas apresenta convergência bastante lenta.

A técnica de iterações de Jacobi relaxa a equação (3.10) substituindo todas as ocorrências das incógnitas $\vec{u}'$ pelos valores já conhecidos $\vec{u}$, exceto no caso da amostra central em (i, j, k). A expressão resultante é então utilizada diretamente como uma regra de atualização para $\vec{u}'_{i,j,k}$ até que o resultado deixe de variar significativamente ou até que um número máximo de iterações seja alcançado.

3.4 Forças externas

Observe novamente a equação (3.4) do operador de forças externas:

$$\frac{\partial \vec{u}}{\partial t} = \vec{a}$$

Perceba que esse operador extremamente simples não depende do campo de velocidades. Assim, se as acelerações acrescentadas ao sistema forem sempre conhecidas e limitadas, a integração será garantidamente estável, de modo que podemos utilizar com segurança um método explícito de Euler. Considerando uma malha colocalizada, temos:

$$\vec{u}'_{i,j,k} = \vec{u}_{i,j,k} + h\vec{a}_{i,j,k} \tag{3.11}$$

No caso de uma malha deslocada, as acelerações a serem acrescentadas também devem ser definidas nas faces das células, bastando então ajustar os

índices de cada componente na equação acima.

A simplicidade do operador se reflete também no fato de não ser necessária nenhuma informação de vizinhança em (3.11). Assim, não precisamos de nenhum tipo de condição de fronteira em sua resolução.

Apesar da simplicidade desse operador, há uma pequena ressalva a ser feita: como as acelerações são completamente arbitrárias, essa etapa pode fazer que as velocidades intermediárias obtidas violem fortemente a condição de incompressibilidade. Por isso, recomenda-se que ela seja sempre seguida diretamente pelo operador de projeção que veremos na Seção 3.5.

3.5 Projeção

Ao aplicarmos a separação de operadores descrita na Seção 3.1, permitimos que as velocidades intermediárias obtidas pelas resoluções de advecção, difusão e forças externas violem livremente a condição de estabilidade (2.1). Nesta última etapa, calculamos o campo de pressão que se faz necessário para restaurar a validade dessa condição.

Seguindo o raciocínio do método *Stable Fluids* [2], começamos utilizando uma propriedade matemática conhecida como decomposição de Helmholtz-Hodge. De acordo com ela, qualquer campo vetorial suave pode ser expresso de forma única como a soma de um campo com rotacional zero e outro com divergência zero. A parte irrotacional, por sua vez, pode ser considerada um campo potencial escalar e expressa como o gradiente de um campo escalar:

$$\vec{u} = \vec{u}' + \nabla q \tag{3.12}$$

Aplicando o operador divergência aos dois lados dessa equação e lembrando que $\nabla \cdot \vec{u}' = 0$, temos:

$$\nabla \cdot \vec{u} = \nabla^2 q \tag{3.13}$$

A equação (3.13) pode ser usada para determinar o campo escalar q, associado às pressões do fluido. Uma vez que ele seja encontrado, podemos impor a condição de incompressibilidade fazendo:

$$\vec{u}' = \vec{u} - \nabla q \tag{3.14}$$

É interessante notar que, se repetirmos todo esse procedimento mais de uma vez, a velocidade resultante deve se manter inalterada, pois já é livre de divergência. Esse fato motiva a descrição do método como uma projeção.

Retornemos agora à equação (3.13). Para aplicá-la, é preciso primeiramente calcular a divergência do campo de velocidades intermediárias $\vec{u}$. Con-

siderando uma discretização espacial em malha colocalizada e utilizando diferenças centrais de segunda ordem, temos:

$$\begin{aligned}[\nabla\cdot\vec{u}]_{i,j,k} &= (u_{i+1,j,k} - u_{i-1,j,k}) \,/\, (2\delta_x) \\ &+ (v_{i,j+1,k} - v_{i,j-1,k}) \,/\, (2\delta_y) \\ &+ (w_{i,j,k+1} - w_{i,j,k-1}) \,/\, (2\delta_z)\end{aligned} \tag{3.15}$$

Se estivermos trabalhando com uma malha deslocada:

$$\begin{aligned}[\nabla\cdot\vec{u}]_{i,j,k} &= \left(u_{i+1/2,j,k} - u_{i-1/2,j,k}\right) /\delta_x \\ &+ \left(v_{i,j+1/2,k} - v_{i,j-1/2,k}\right) /\delta_y \\ &+ \left(w_{i,j,k+1/2} - w_{i,j,k-1/2}\right) /\delta_z\end{aligned} \tag{3.16}$$

Uma vez encontrada a divergência, precisamos lidar com a equação (3.13) propriamente dita. Trata-se novamente de uma equação de Poisson, que representa um sistema linear esparso simétrico com uma incógnita para cada célula da malha. A resolução desse sistema pode ser feita com auxílio dos mesmos métodos citados na Seção 3.3. Além disso, sua regra de construção tem a mesma forma de (3.10), bastando substituir $\vec{u}$ e $\vec{u}'$ por q e q', respectivamente, e utilizando valores diferentes para os parâmetros α, β e f, conforme se pode ver abaixo:

$$\begin{aligned}q'_{i,j,k}\left[\beta + 2\left(k_{xx} + k_{yy} + k_{zz}\right)\right] = \alpha f + \\ \left(q'_{i+1,j,k} + q'_{i-1,j,k}\right) k_{xx} + \\ \left(q'_{i,j+1,k} + q'_{i,j-1,k}\right) k_{yy} + \\ \left(q'_{i,j,k+1} + q'_{i,j,k-1}\right) k_{zz}\end{aligned} \tag{3.17}$$

onde:

$$\begin{aligned} k_{xx} &= 1/\delta_x^2 & \alpha &= -1 \\ k_{yy} &= 1/\delta_y^2 & \beta &= 0 \\ k_{zz} &= 1/\delta_z^2 & f &= (\nabla\cdot\vec{u})_{i,j,k}\end{aligned}$$

De posse do campo de pressões, é fácil utilizar (3.14) para projetar as velocidades em um campo livre de divergência. Considerando diferenças centrais numa malha colocalizada:

$$\begin{aligned} u'_{i,j,k} &= u_{i,j,k} - (q_{i+1,j,k} - q_{i-1,j,k}) \,/\, (2\delta_x) \\ v'_{i,j,k} &= v_{i,j,k} - (q_{i,j+1,k} - q_{i,j-1,k}) \,/\, (2\delta_y) \\ w'_{i,j,k} &= w_{i,j,k} - (q_{i,j,k+1} - q_{i,j,k-1}) \,/\, (2\delta_z)\end{aligned} \tag{3.18}$$

De forma semelhante, numa malha deslocada, temos:

$$\begin{aligned} u'_{i+1/2,j,k} &= u_{i+1/2,j,k} - \left(q_{i+1,j,k} - q_{i,j,k}\right)/\delta_x \\ v'_{i,j+1/2,k} &= v_{i,j+1/2,k} - \left(q_{i,j+1,k} - q_{i,j,k}\right)/\delta_y \\ w'_{i,j,k+1/2} &= w_{i,j,k+1/2} - \left(q_{i,j,k+1} - q_{i,j,k}\right)/\delta_z \end{aligned} \tag{3.19}$$

3.6 Transporte de escalares e partículas

Finalizada a integração do campo de velocidade, podemos utilizá-lo para simular grandezas escalares sendo transportadas pelo fluido, como densidades de tinta ou outros materiais diluídos. A equação a seguir apresenta uma modelagem simples para esse tipo de fenômeno de transporte [2]:

$$\frac{\partial r}{\partial t} = -\left(\vec{u}\cdot\nabla\right) r + \kappa_d \nabla^2 r - \kappa_e r + S \tag{3.20}$$

Nesta equação, κ_d e κ_e são coeficientes configuráveis que controlam a dissipação e a extinção da grandeza escalar, respectivamente. Perceba também que (2.2) e (3.20) são notavelmente semelhantes. Na verdade, a mesma abordagem utilizada para as velocidades do fluido também pode ser repetida aqui, com a aplicação sequencial de operadores para convecção ou advecção (A), difusão (D), fonte externa (F) e extinção (E):

$$r^{n+1}_{i,j,k} = E \circ F \circ D \circ A\left(r^n_{i,j,k}\right) \tag{3.21}$$

Na equação acima, não existe um operador de projeção, já que a grandeza escalar não está sujeita à condição de incompressibilidade (2.1). Por outro lado, temos um novo termo de extinção, que modela uma perda gradual da grandeza para o ambiente externo. Os operadores podem ser expressados como se segue:

$$\frac{\partial r}{\partial t} = A(r) = -\left(\vec{u}\cdot\nabla\right) r \tag{3.22}$$

$$\frac{\partial r}{\partial t} = D(r) = \kappa_d \nabla^2 r \tag{3.23}$$

$$\frac{\partial r}{\partial t} = F(r) = S \tag{3.24}$$

$$\frac{\partial r}{\partial t} = E(r) = -\kappa_e r \tag{3.25}$$

A aplicação dos operadores de advecção, difusão e fonte externa é análoga à de seus correspondentes na integração do campo de velocidade do fluido. O novo operador de extinção, por sua vez, não apresenta problemas de estabilidade e pode ser integrado diretamente:

$$r'_{i,j,k} = \left(1 + \kappa_e h\right)^{-1} r_{i,j,k} \tag{3.26}$$

Finalmente, também já de posse do campo de velocidade do fluido, podemos calcular a trajetória de partículas sem massa sendo transportadas por ele, o que é útil para a visualização do escoamento e a geração de linhas de fluxo. A integração da posição de cada partícula é análoga à advecção semi-Lagrangiana, mas pode ser feita explicitamente com o passo de tempo positivo. Considerando um integrador de primeira ordem, por exemplo, temos:

$$\vec{x}'_{particula} = \vec{x}_{particula} + h\vec{u}\left(\vec{x}_{particula}\right) \tag{3.27}$$

Assim como no caso da advecção, recomenda-se a utilização de Runge-Kutta de pelo menos segunda ordem para essa etapa.

3.7 Condições de contorno

A advecção de partículas Lagrangianas e a aproximação de derivadas por diferenças finitas requerem frequentemente os valores de vizinhos diretos de cada ponto amostral. Quando um desses vizinhos recai sobre uma região fora do domínio de simulação do fluido, precisamos decidir os valores a serem utilizados de maneira a respeitar a natureza física do problema. Essa imposição de condições de contorno deve ser feita sempre que necessário para cada um dos operadores utilizados.

O método *Stable Fluids* utiliza um conjunto de condições de contorno bastante simples, porém adequado para a obtenção de simulações aproximadas e genérico o suficiente para permitir a representação de diferentes tipos de comportamentos. A abordagem, proposta por [4], consiste em utilizar uma camada extra de células de fronteira ao redor do domínio computacional, como ilustrado na Figura 3.2. As propriedades do fluido nessas células são então manipuladas de forma a satisfazer restrições físicas e a impor características desejáveis ao escoamento. Podemos, por exemplo, considerar que há uma parede lisa (livre de atrito) ou rugosa ao redor do domínio, com alguns pontos em que o fluido é obrigado a entrar ou sair com uma velocidade determinada.

A Tabela 3.1 mostra como controlar a velocidade do fluido nas células de fronteira de modo a se obter os diferentes comportamentos mostrados na Figura 3.2.

Quanto à pressão atribuída às células de borda, utiliza-se em todos os casos a seguinte condição de Neumann, onde $\hat{n}$ significa a direção normal à fronteira:

$$\frac{\partial q}{\partial \hat{n}} = 0 \Rightarrow q_{borda} = q_{interna} \tag{3.28}$$

A aplicação das condições de contorno descritas acima é bastante direta. Primeiramente, note que as células de borda na Figura 3.2 não fazem parte

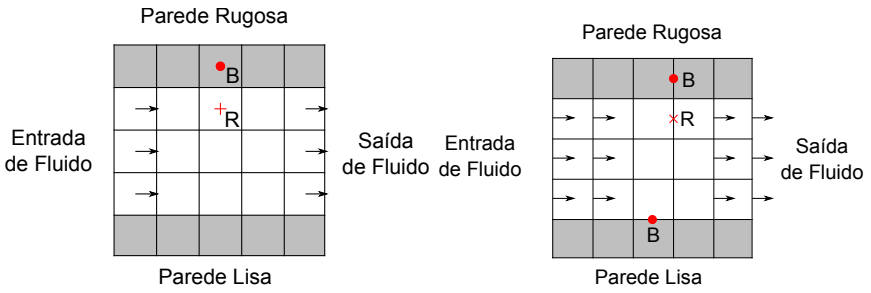


Figura 3.2 – Exemplos de condições de contorno para a velocidade de um fluido em uma malha regular bidimensional: (a) malha colocalizada; (b) malha deslocada.

Tabela 3.1 – Especificação de velocidades nas amostras de fronteira para a obtenção de diferentes comportamentos de escoamento. Os subscritos $\parallel$ e $\perp$ indicam as componentes tangencial e normal da velocidade em relação à fronteira, respectivamente. Os índices B e R se referem à amostra de borda e sua vizinha direta no interior do domínio, respectivamente.

Efeito	Malha colocalizada		Malha deslocada	
	$u_{B\parallel}$	$u_{B\perp}$	$u_{B\parallel}$	$u_{\perp}$
Passagem livre	$u_{R\parallel}$	$u_{R\perp}$	$u_{R\parallel}$	$u_{R\perp}$
Parede lisa	$u_{R\parallel}$	$-u_{R\perp}$	$u_{R\parallel}$	0
Parede rugosa	$-u_{R\parallel}$	$-u_{R\perp}$	$-u_{R\parallel}$	0
Entrada de fluido	0	u_{in}	0	u_{in}
Saída de fluido	0	u_{out}	0	u_{out}

do domínio computacional: suas propriedades são controladas e atualizadas explicitamente pela simulação sempre que necessário.

No passo de advecção, a posição de origem de uma partícula Lagrangiana pode resultar fora do domínio. Nesse caso, consideramos o ponto mais próximo sobre a fronteira e interpolamos nesse local uma nova velocidade. Perceba que, no caso de uma parede lisa ou rugosa, a componente normal resultante será sempre zero, como desejado.

Nos demais passos da simulação, utilizamos diretamente as amostras necessárias das células de fronteira. No caso do passo de projeção, note que as divergências do campo de velocidade são calculadas já considerando-se os valores de fronteira esperados após a aplicação do operador. Por isso, o gradiente de pressão ao longo da borda não deve exercer influência sobre esses valores, o que torna razoável a condição de Neumann especificada.

4
Método numérico em malha curvilínea

Neste capítulo, descrevemos como aplicar as matrizes Jacobianas ao integrador *Stable Fluids* de modo que cada um de seus passos seja realizado, tanto quanto possível, diretamente no espaço paramétrico. Na Seção 4.6, damos atenção especial também à estipulação de condições de contorno.

Ao longo do capítulo, consideramos quatro configurações distintas para a amostragem e a representação das velocidades e pressões do fluido. Quanto ao esquema de amostragem, consideramos grades colocalizadas com amostras centrais e malhas deslocadas com as velocidades nas faces das células. Quanto à representação, consideramos as velocidades do fluido expressas tanto no espaço do mundo quanto no sistema de coordenadas paramétricas local. A Figura 4.1 ilustra esses arranjos para o caso bidimensional.

À primeira vista, pode parecer estranho trabalhar com velocidades no espaço do mundo em uma malha deslocada, pois a componente de velocidade armazenada em uma face pode apresentar uma contribuição pequena ou nula para o fluxo passando por ela quando a grade está rotacionada em cerca de noventa graus (Figura 4.1c). No entanto, como nota Azevedo [28], essa característica só pode causar problemas quando a grade está rotacionada em exatamente noventa graus numa região com células de dimensões constantes (ou seja, com uma matriz Jacobiana constante).

4.1
Advecção

No passo de advecção, utilizamos a mesma abordagem semi-Lagrangiana descrita na Seção 3.2 para o método *Stable Fluids*. Inicialmente, consultamos a velocidade do fluido na posição de cada amostra, interpolando as componentes faltantes no caso de uma malha deslocada. Em seguida, consideramos uma partícula sem massa nesse local e integramos sua trajetória para trás ao longo de um passo de tempo. Finalmente, interpolamos a velocidade do fluido no ponto alcançado e atribuímos seu valor à amostra original, descartando componentes desnecessárias quando na malha deslocada. Note que, devido à obrigatoriedade de trabalharmos sempre com vetores completos de velocidade,

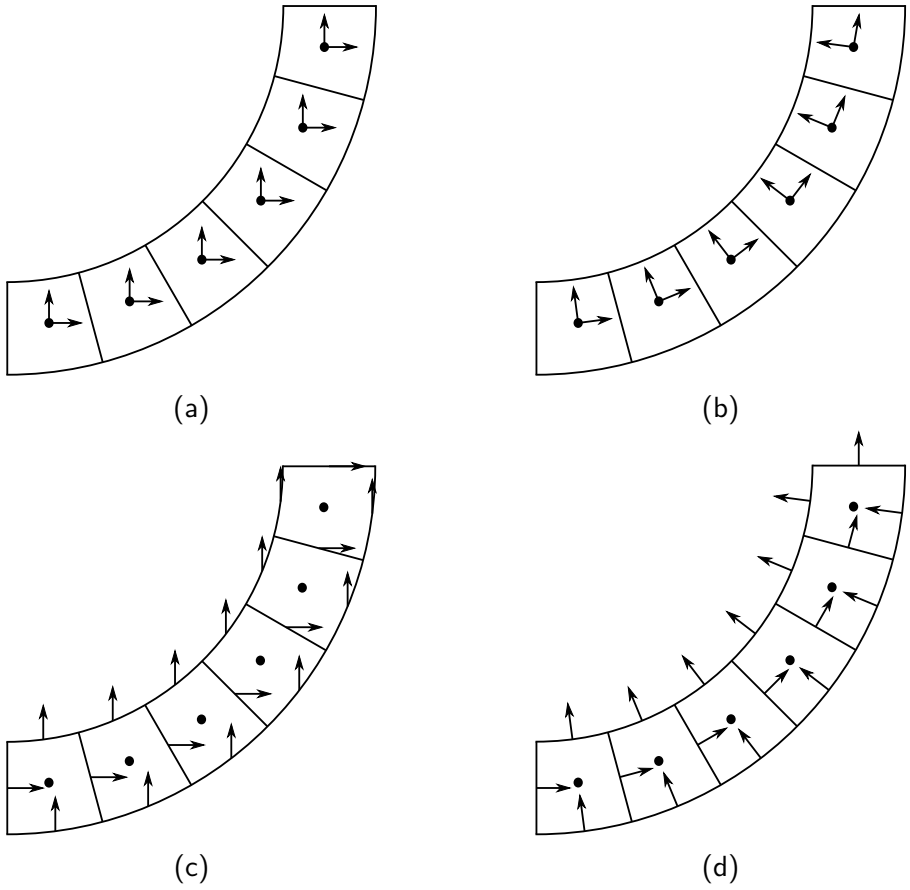


Figura 4.1 – Arranjos considerados para a amostragem e a representação de velocidades (setas) e pressões (pontos) no caso bidimensional: (a) velocidades globais em malha colocalizada; (b) velocidades paramétricas em malha colocalizada; (c) velocidades globais em malha deslocada; (d) velocidades paramétricas em malha deslocada.

o custo computacional associado a cada componente de uma malha deslocada é comparável ao custo total de todas as componentes numa amostragem colocalizada, o que torna essa última mais eficiente nesta etapa.

É importante salientar que, como tanto a posição original quanto a que queremos calcular estão em coordenadas paramétricas, é conveniente e correto realizarmos a integração diretamente nesse sistema. Isso quer dizer que quaisquer velocidades consultadas e calculadas pelo integrador no espaço do mundo podem ser sempre convertidas para o paramétrico utilizando a equação (2.16) e o Jacobiano local no ponto de amostragem. Da mesma maneira, ao utilizarmos um integrador de ordem não linear, a velocidade final pode ser obtida por uma soma ponderada de velocidades em coordenadas paramétricas, mesmo que tenham sido obtidas em pontos com Jacobianos diferentes. Isso permite que o integrador leve em conta efetivamente as variações tanto das velocidades no espaço do mundo quanto dos Jacobianos ao longo da trajetória.

Para compreender melhor a diferença entre realizar a integração no

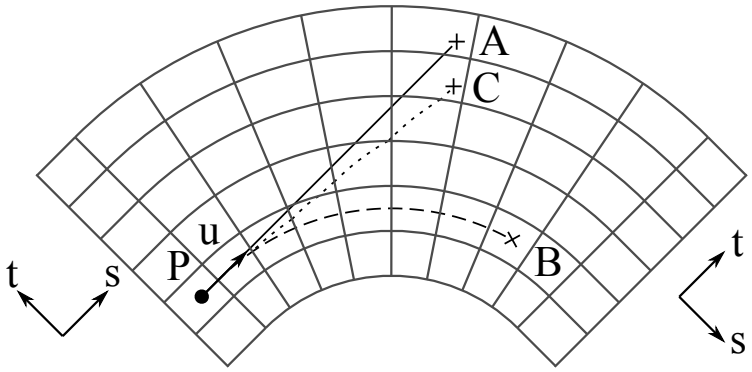


Figura 4.2 – Transportando uma partícula P com velocidade $\vec{u}$. A) Linha sólida: trajetória obtida por um integrador linear no espaço do mundo, correta para $\vec{u}$ constante em (x, y). B) Linha tracejada: trajetória obtida por um integrador linear no espaço paramétrico, correta para $\vec{u}$ constante em (s, t). C) Linha pontilhada: exemplo de aproximação conseguida para $\vec{u}$ constante em (x, y) por um integrador linear no espaço paramétrico com passo adaptativo.

espaço do mundo e no paramétrico, observe a Figura 4.2, que mostra uma partícula P com velocidade $\vec{u}$ em certa região de um domínio curvilíneo. Considere primeiramente que o campo de velocidade é constante no espaço do mundo ao longo de toda a região. Um integrador linear atuando no espaço do mundo seria suficiente para gerar a trajetória A, que corresponde exatamente ao resultado teórico esperado. Por outro lado, o mesmo tipo de integrador atuando em coordenadas paramétricas levaria a partícula à posição B, possivelmente muito distante da desejada. Invertendo o raciocínio, considere agora que o campo de velocidade é constante no espaço paramétrico, ou seja, que o escoamento local está seguindo a forma da malha na região. Nesse caso, o resultado B dado por um integrador linear em coordenadas paramétricas passa a ser exato, enquanto a trajetória A do mesmo integrador no espaço do mundo passa a apresentar um erro possivelmente grosseiro. Isso faz sentido, pois como o Jacobiano está variando ao longo da região, um campo de velocidade constante ou simples em um dos sistemas corresponde a um campo variável ou mais complexo no outro. Assim, podemos concluir que a precisão de qualquer integrador está diretamente ligada ao comportamento do campo de velocidade no mesmo espaço em que a integração é realizada.

Para tornarmos nosso integrador mais robusto, é importante tentarmos garantir uma boa precisão em campos de velocidade simples no espaço do mundo, mesmo que a integração seja realizada em coordenadas paramétricas e que os Jacobianos variem ao longo do domínio. Voltando ao exemplo da Figura 4.2 e considerando um escoamento com velocidade constante no mundo, gostaríamos de obter uma trajetória próxima a A, se possível mesmo com um integrador linear no espaço paramétrico. Para conseguirmos isso,

subdividimos o passo de integração em subpassos adaptativos, que se reduzem automaticamente em regiões de grande variação de velocidade no espaço considerado.

Repare que, em nossa modelagem, variações de velocidade e de Jacobiano só podem ser capturadas até o nível de resolução da malha discretizada escolhida para o domínio. Assim, numa grade bem projetada, podemos assumir como pequenas as variações ocorridas dentro de cada célula. Por outro lado, mesmo distâncias curtas e pequenos números de células podem levar a grandes variações de velocidade em algumas situações, como em regiões centrais de vórtices ou próximo a quinas e curvas acentuadas do domínio. Essas características nos fornecem uma heurística simples para o controle de nosso passo adaptativo: basta limitarmos seu tamanho de forma a impedir que uma partícula atravesse mais de uma célula por vez. Podemos implementar essa estratégia acompanhando a partícula ao longo de seu caminho à medida que percorre cada célula da grade: sempre que ela atravessar a fronteira entre duas células, podemos reamostrar ou recalcular a velocidade restante no espaço paramétrico, o que resulta na trajetória pontilhada (C) na Figura 4.2. Alternativamente, uma abordagem mais rápida e igualmente eficaz é ignorar as fronteiras entre as células e reamostrar ou recalcular a velocidade restante sempre que a partícula percorrer uma distância unitária no espaço paramétrico. Isso evita trabalho desnecessário quando a partícula atravessa fronteiras em duas ou três dimensões de uma vez, enquanto mantém a propriedade útil de considerar o tamanho local da malha, já que a reamostragem é realizada com maior frequência quando atravessando células pequenas.

É importante perceber que toda a discussão anterior diz respeito apenas ao espaço em que pode ocorrer a advecção das partículas Lagrangianas. Após esse processo, na atribuição final da velocidade encontrada à amostra originalmente considerada, é essencial que a operação seja feita com base no espaço do mundo. Assim, se estivermos trabalhando com velocidades paramétricas, torna-se necessário realizar uma conversão dupla: primeiro para o sistema global utilizando-se o Jacobiano no ponto encontrado, e depois para o espaço local da amostra aplicando-se a transformação correspondente.

4.2 Difusão

A equação discretizada para o passo de difusão no espaço paramétrico precisa ser derivada desde o começo. Partimos de (3.9), a equação implícita de Poisson utilizada no método *Stable Fluids* [2]. A expressão pode ser reescrita como a seguir, onde $\vec{u}'$ representa o valor de $\vec{u}$ ao final do passo de tempo

corrente:

$$\frac{\vec{u}' - \vec{u}}{\nu h} = (\vec{u}'_{xx} + \vec{u}'_{yy} + \vec{u}'_{zz}) \tag{4.1}$$

Para convertermos as derivadas direcionais do operador Laplaciano para o espaço paramétrico, começamos utilizando (2.15):

$$\begin{aligned} = &\left(\vec{u}'_s s_x + \vec{u}'_t t_x + \vec{u}'_p p_x\right)_x + \\ &\left(\vec{u}'_s s_y + \vec{u}'_t t_y + \vec{u}'_p p_y\right)_y + \\ &\left(\vec{u}'_s s_z + \vec{u}'_t t_z + \vec{u}'_p p_z\right)_z \end{aligned}$$

Aplicando agora a regra da cadeia, temos:

$$\begin{aligned} = &\vec{u}'_{sx} s_x + \vec{u}'_s s_{xx} + \vec{u}'_{tx} t_x + \vec{u}'_t t_{xx} + \vec{u}'_{px} p_x + \vec{u}'_p p_{xx} + \\ &\vec{u}'_{sy} s_y + \vec{u}'_s s_{yy} + \vec{u}'_{ty} t_y + \vec{u}'_t t_{yy} + \vec{u}'_{py} p_y + \vec{u}'_p p_{yy} + \\ &\vec{u}'_{sz} s_z + \vec{u}'_s s_{zz} + \vec{u}'_{tz} t_z + \vec{u}'_t t_{zz} + \vec{u}'_{pz} p_z + \vec{u}'_p p_{zz} \end{aligned}$$

Utilizando novamente (2.15), chegamos a uma expressão em que as derivadas direcionais das velocidades estão todas em relação aos eixos paramétricos:

$$\begin{aligned} = &\left(\vec{u}'_{ss} s_x + \vec{u}'_{st} t_x + \vec{u}'_{sp} p_x\right) s_x + \vec{u}'_s s_{xx} + \\ &\left(\vec{u}'_{ts} s_x + \vec{u}'_{tt} t_x + \vec{u}'_{tp} p_x\right) t_x + \vec{u}'_t t_{xx} + \\ &\left(\vec{u}'_{ps} s_x + \vec{u}'_{pt} t_x + \vec{u}'_{pp} p_x\right) p_x + \vec{u}'_p p_{xx} + \\ &\left(\vec{u}'_{ss} s_y + \vec{u}'_{st} t_y + \vec{u}'_{sp} p_y\right) s_y + \vec{u}'_s s_{yy} + \\ &\left(\vec{u}'_{ts} s_y + \vec{u}'_{tt} t_y + \vec{u}'_{tp} p_y\right) t_y + \vec{u}'_t t_{yy} + \\ &\left(\vec{u}'_{ps} s_y + \vec{u}'_{pt} t_y + \vec{u}'_{pp} p_y\right) p_y + \vec{u}'_p p_{yy} + \\ &\left(\vec{u}'_{ss} s_z + \vec{u}'_{st} t_z + \vec{u}'_{sp} p_z\right) s_z + \vec{u}'_s s_{zz} + \\ &\left(\vec{u}'_{ts} s_z + \vec{u}'_{tt} t_z + \vec{u}'_{tp} p_z\right) t_z + \vec{u}'_t t_{zz} + \\ &\left(\vec{u}'_{ps} s_z + \vec{u}'_{pt} t_z + \vec{u}'_{pp} p_z\right) p_z + \vec{u}'_p p_{zz} \end{aligned}$$

Podemos simplificar a equação acima combinando alguns termos:

$$\begin{aligned} = &\vec{u}'_s \left(s_{xx} + s_{yy} + s_{zz}\right) + \vec{u}'_{ss} \left(s_x^2 + s_y^2 + s_z^2\right) + \\ &\vec{u}'_t \left(t_{xx} + t_{yy} + t_{zz}\right) + \vec{u}'_{tt} \left(t_x^2 + t_y^2 + t_z^2\right) + \\ &\vec{u}'_p \left(p_{xx} + p_{yy} + p_{zz}\right) + \vec{u}'_{pp} \left(p_x^2 + p_y^2 + p_z^2\right) + \\ &2\vec{u}'_{st} \left(s_x t_x + s_y t_y + s_z t_z\right) + \\ &2\vec{u}'_{tp} \left(t_x p_x + t_y p_y + t_z p_z\right) + \\ &2\vec{u}'_{ps} \left(p_x s_x + p_y s_y + p_z s_z\right) \end{aligned}$$

Finalmente, considerando uma malha colocalizada com velocidades ar-

mazenadas no espaço do mundo, podemos aproximar essa equação por diferenças centrais na grade uniforme subjacente, onde $\delta_s = \delta_t = \delta_p = 1$. Assim, incluindo de volta também o lado esquerdo de (4.1), obtemos a seguinte expressão discreta para $\vec{u}$:

$$\begin{aligned}
&\vec{u}'_{i,j,k}\left[\beta + 2\left(k_{ss} + k_{tt} + k_{pp}\right)\right] = \alpha f + \\
&\left(\vec{u}'_{i+1,j,k} + \vec{u}'_{i-1,j,k}\right) k_{ss} + \left(\vec{u}'_{i+1,j,k} - \vec{u}'_{i-1,j,k}\right) k_s + \\
&\left(\vec{u}'_{i,j+1,k} + \vec{u}'_{i,j-1,k}\right) k_{tt} + \left(\vec{u}'_{i,j+1,k} - \vec{u}'_{i,j-1,k}\right) k_t + \\
&\left(\vec{u}'_{i,j,k+1} + \vec{u}'_{i,j,k-1}\right) k_{pp} + \left(\vec{u}'_{i,j,k+1} - \vec{u}'_{i,j,k-1}\right) k_p + \\
&\left(\vec{u}'_{i+1,j+1,k} + \vec{u}'_{i-1,j-1,k} - \vec{u}'_{i+1,j-1,k} - \vec{u}'_{i-1,j+1,k}\right) k_{st} + \\
&\left(\vec{u}'_{i,j+1,k+1} + \vec{u}'_{i,j-1,k-1} - \vec{u}'_{i,j+1,k-1} - \vec{u}'_{i,j-1,k+1}\right) k_{tp} + \\
&\left(\vec{u}'_{i+1,j,k+1} + \vec{u}'_{i-1,j,k-1} - \vec{u}'_{i-1,j,k+1} - \vec{u}'_{i+1,j,k-1}\right) k_{ps}
\end{aligned} \tag{4.2}$$

onde:

$$\begin{aligned}
k_{ss} &= s_x^2 + s_y^2 + s_z^2 & k_s &= \left(s_{xx} + s_{yy} + s_{zz}\right)/2 \\
k_{tt} &= t_x^2 + t_y^2 + t_z^2 & k_t &= \left(t_{xx} + t_{yy} + t_{zz}\right)/2 \\
k_{pp} &= p_x^2 + p_y^2 + p_z^2 & k_p &= \left(p_{xx} + p_{yy} + p_{zz}\right)/2 \\
k_{st} &= \left(s_x t_x + s_y t_y + s_z t_z\right)/2 & \alpha &= 1/\nu h \\
k_{tp} &= \left(t_x p_x + t_y p_y + t_z p_z\right)/2 & \beta &= 1/\nu h \\
k_{ps} &= \left(p_x s_x + p_y s_y + p_z s_z\right)/2 & f &= \vec{u}_{i,j,k}
\end{aligned}$$

Assim como no caso de uma malha regular, a equação (4.2) deve ser aplicada de maneira independente a cada componente da velocidade do fluido. Além disso, a utilização de uma amostragem deslocada é também simples, bastando-se ajustar os índices de $\vec{u}$ e $\vec{u}'$ apropriadamente em cada dimensão.

No caso de termos velocidades armazenadas no espaço paramétrico, devemos aplicar (2.17) a cada $\vec{u}$ e $\vec{u}'$ em (4.2). Idealmente, cada velocidade deveria ser convertida pelo Jacobiano local em seu ponto de amostragem. No entanto, seguindo esse procedimento, a expressão resultante teria uma complexidade muito elevada, com as componentes (s, t, p) acopladas entre si. Em vez disso, assumimos como uma simplificação que as amostras estão suficientemente próximas ao ponto central para que todas possam ser convertidas pela mesma matriz J^T. Com isso, a mesma transformação aparece em ambos lados da equação, o que torna seu efeito efetivamente nulo sobre o resultado. Assim, podemos utilizar (4.2) diretamente também para cada coordenada paramétrica.

Note que, como mencionado na Seção 2.4, as métricas k_s, k_t, k_p, k_{ss},

k_{tt}, k_{pp}, k_{st}, k_{tp} e k_{ps} em (4.2) podem ser todas precalculadas em cada célula da malha (9 constantes em 3D, 5 em 2D). Entretanto, é importante perceber que, com a generalização de (3.10) para (4.2), o sistema de equações do solucionador de Poisson passa a apresentar diversas novas super e subdiagonais em sua matriz, de modo que o valor de cada incógnita passa a ser dependente de não menos que dezoito vizinhas (oito em 2D). Além disso, o sistema deixa de ser simétrico, de forma que o método do Gradiente Conjugado precisa ser substituído por um solucionador iterativo mais complexo, como o Gradiente Biconjugado Estabilizado (BiCGStab). Quanto ao precondicionador, escolhemos agora uma versão de *Multigrid* Algébrico, já que Cholesky Incompleto também deixa de ser válido.

Como mencionado anteriormente, utilizaremos também iterações de Jacobi simples como uma base de comparação para testes de desempenho e convergência. A regra de atualização de Jacobi pode ser obtida a partir de (4.2) simplesmente substituindo-se cada valor desconhecido de velocidade $\vec{u}'$, com exceção do valor da amostra central $\vec{u}'_{i,j,k}$, pelo correspondente obtido na iteração anterior.

Como observação final, ressalta-se que, na área de computação gráfica, muitas vezes o termo de difusão ou viscosidade é completamente ignorado [29, 32]. Isso é razoável ao se buscar apenas uma aproximação do comportamento de um fluido, sobretudo devido à difusão numérica associada ao processo de advecção semi-Lagrangiana visto nas Seções 3.2 e 4.1. Pode-se mostrar [32] que essa difusão numérica resulta numa suavização do campo de velocidade do fluido muito semelhante com a causada pela viscosidade.

4.3 Forças externas

O operador de forças externas não precisa sofrer nenhuma adaptação para trabalhar com malhas curvilíneas paramétricas. No entanto, quando consideramos velocidades armazenadas no espaço local da malha, quaisquer acelerações conhecidas no sistema global precisam ser primeiramente convertidas para ele por (2.16). De forma semelhante, acelerações dadas no espaço paramétrico precisam ser convertidas por (2.17) antes de serem aplicadas a um campo de velocidade no sistema global.

Ressalta-se que é preciso tomar cuidado ao se realizar transformações desse tipo em meio a uma malha deslocada. Como as componentes vetoriais são armazenadas em diferentes pontos com Jacobianos possivelmente distintos, pode ser impossível recuperar o campo vetorial original por meio de interpolações. Isso é especialmente relevante ao se lidar explicitamente com acelerações

assumidas constantes em um dos espaços. Após a conversão, acelerações desse tipo podem não ser mais vistas como constantes pelo simulador.

4.4 Projeção

O passo de projeção calcula inicialmente a divergência do campo de velocidade para que possa ser usada em (3.13). A equação (2.15) pode ser usada para computar as derivadas no espaço paramétrico:

$$\begin{aligned}\nabla\cdot\vec{u} =& \dot{x}_x + \dot{y}_y + \dot{z}_z \\ =& \dot{x}_s s_x + \dot{x}_t t_x + \dot{x}_p p_x + \\ & \dot{y}_s s_y + \dot{y}_t t_y + \dot{y}_p p_y + \\ & \dot{z}_s s_z + \dot{z}_t t_z + \dot{z}_p p_z\end{aligned} \tag{4.3}$$

A equação (4.3) é válida tanto para uma malha colocalizada quanto para uma deslocada. Ela assume, porém, que as velocidades estão armazenadas no espaço do mundo ou que devem ser convertidas para ele. Se estivermos trabalhando com velocidades em coordenadas paramétricas, podemos obter uma forma mais apropriada para a divergência aplicando (2.15) primeiramente às derivadas temporais:

$$\begin{aligned}\nabla\cdot\vec{u} =& \dot{x}_x + \dot{y}_y + \dot{z}_z \\ =& \left(x_s\dot{s} + x_t\dot{t} + x_p\dot{p}\right)_x + \\ & \left(y_s\dot{s} + y_t\dot{t} + y_p\dot{p}\right)_y + \\ & \left(z_s\dot{s} + z_t\dot{t} + z_p\dot{p}\right)_z\end{aligned}$$

Aplicando agora a regra da cadeia:

$$\begin{aligned}\nabla\cdot\vec{u} = & x_{sx}\dot{s} + x_s\dot{s}_x + x_{tx}\dot{t} + x_t\dot{t}_x + x_{px}\dot{p} + x_p\dot{p}_x + \\ & y_{sy}\dot{s} + y_s\dot{s}_y + y_{ty}\dot{t} + y_t\dot{t}_y + y_{py}\dot{p} + y_p\dot{p}_y + \\ & z_{sz}\dot{s} + z_s\dot{s}_z + z_{tz}\dot{t} + z_t\dot{t}_z + z_{pz}\dot{p} + z_p\dot{p}_z\end{aligned}$$

A expressão acima pode ser simplificada se reagruparmos os termos como a seguir:

$$\begin{aligned}\nabla\cdot\vec{u} = &\left(\dot{s}_x x_s + \dot{s}_y y_s + \dot{s}_z z_s\right) + \\ &\left(\dot{t}_x x_t + \dot{t}_y y_t + \dot{t}_z z_t\right) + \\ &\left(\dot{p}_x x_p + \dot{p}_y y_p + \dot{p}_z z_p\right) + \\ &\dot{s}\left(x_{sx} + y_{sy} + z_{sz}\right) + \\ &\dot{t}\left(x_{tx} + y_{ty} + z_{tz}\right) + \\ &\dot{p}\left(x_{px} + y_{py} + z_{pz}\right)\end{aligned}$$

Aplicando (2.14) e a definição do operador divergência, temos:

$$\begin{aligned}\nabla\cdot\vec{u} = &\left(\dot{s}_s + \dot{t}_t + \dot{p}_p\right) + \\ &\dot{s}\left(\nabla\cdot\left(x_s, y_s, z_s\right)\right) + \\ &\dot{t}\left(\nabla\cdot\left(x_t, y_t, z_t\right)\right) + \\ &\dot{p}\left(\nabla\cdot\left(x_p, y_p, z_p\right)\right)\end{aligned} \tag{4.4}$$

A equação (4.4) mostra que a divergência no espaço do mundo é igual à divergência das velocidades paramétricas somada com a divergência do campo formado por cada uma das bases do espaço paramétrico. Note ainda que o valor dos operadores nas últimas três linhas (conhecidos como Símbolos de Christoffel) depende apenas da malha, podendo então ser precalculado e armazenado. Assim, a avaliação da expressão requer menos computações e a leitura de apenas 12 valores da memória por célula, contra 27 no caso de (4.3).

Uma vez calculada a divergência, utilizamos seu valor na resolução do sistema linear esparso referente às pressões do fluido, cuja regra de atualização para iterações de Jacobi simples deve ser derivada de maneira análoga ao passo de difusão. Similarmente ao algoritmo *Stable Fluids* original, (4.2) pode ser aplicada diretamente substituindo-se $\vec{u}$ e $\vec{u}'$ por q e q', respectivamente, e alterando-se os valores dos parâmetros α, β e f conforme abaixo:

$$\begin{aligned}&q'_{i,j,k}\left[\beta + 2\left(k_{ss} + k_{tt} + k_{pp}\right)\right] = \alpha f + \\ &\left(q'_{i+1,j,k} + q'_{i-1,j,k}\right)k_{ss} + \left(q'_{i+1,j,k} - q'_{i-1,j,k}\right)k_s + \\ &\left(q'_{i,j+1,k} + q'_{i,j-1,k}\right)k_{tt} + \left(q'_{i,j+1,k} - q'_{i,j-1,k}\right)k_t + \\ &\left(q'_{i,j,k+1} + q'_{i,j,k-1}\right)k_{pp} + \left(q'_{i,j,k+1} - q'_{i,j,k-1}\right)k_p + \\ &\left(q'_{i+1,j+1,k} + q'_{i-1,j-1,k} - q'_{i+1,j-1,k} - q'_{i-1,j+1,k}\right)k_{st} + \\ &\left(q'_{i,j+1,k+1} + q'_{i,j-1,k-1} - q'_{i,j+1,k-1} - q'_{i,j-1,k+1}\right)k_{tp} + \\ &\left(q'_{i+1,j,k+1} + q'_{i-1,j,k-1} - q'_{i-1,j,k+1} - q'_{i+1,j,k-1}\right)k_{ps}\end{aligned} \tag{4.5}$$

onde:

$$
\begin{aligned}
k_{ss} &= s_x^2 + s_y^2 + s_z^2 & k_s &= (s_{xx} + s_{yy} + s_{zz})/2 \\
k_{tt} &= t_x^2 + t_y^2 + t_z^2 & k_t &= (t_{xx} + t_{yy} + t_{zz})/2 \\
k_{pp} &= p_x^2 + p_y^2 + p_z^2 & k_p &= (p_{xx} + p_{yy} + p_{zz})/2 \\
k_{st} &= (s_x t_x + s_y t_y + s_z t_z)/2 & \alpha &= -1 \\
k_{tp} &= (t_x p_x + t_y p_y + t_z p_z)/2 & \beta &= 0 \\
k_{ps} &= (p_x s_x + p_y s_y + p_z s_z)/2 & f &= (\nabla \cdot \vec{u})_{i,j,k}
\end{aligned}
$$

De posse dos valores de pressão, podemos finalmente projetar as velocidades em um campo livre de divergência. Para isso, precisamos utilizar (2.17) e transformar a equação (3.14) de modo que o gradiente de pressão seja calculado no espaço paramétrico:

$$\vec{u}'_{(x,y,z)} = \vec{u}_{(x,y,z)} - J^{-1} \left[\nabla q\right]_{(s,t,p)} \tag{4.6}$$

A equação (4.6) assume, mais uma vez, que as velocidades estão armazenadas no espaço do mundo. Se estivermos trabalhando com velocidades em coordenadas paramétricas, podemos aplicar a versão correspondente de (2.17) também a elas:

$$
\begin{aligned}
& J^T \vec{u}'_{(s,t,p)} = J^T \vec{u}_{(s,t,p)} - J^{-1} \left[\nabla q\right]_{(s,t,p)} \\
& \Rightarrow \vec{u}'_{(s,t,p)} = \vec{u}_{(s,t,p)} - J^{-T} J^{-1} \left[\nabla q\right]_{(s,t,p)}
\end{aligned} \tag{4.7}
$$

onde:

$$J^{-T} J^{-1} = J_2 = \begin{pmatrix} k_{ss} & 2k_{st} & 2k_{ps} \\ 2k_{st} & k_{tt} & 2k_{tp} \\ 2k_{ps} & 2k_{tp} & k_{pp} \end{pmatrix}$$

As métricas k_* que aparecem em (4.7) são as mesmas que foram definidas anteriormente em (4.2) e (4.5). Como vimos, podem ser todas precalculadas se desejado.

Note que, se estivermos trabalhando com uma amostragem deslocada da velocidades, o gradiente de pressão ∇q que aparece nas equações (4.6) e (4.7) utiliza valores que não estão na grade. No caso do gradiente utilizado para atualizar a primeira coordenada, temos:

$$
\begin{aligned}
\left[\nabla q\right]_{i+1/2,j,k} = & \left(q_{i+1,j,k} - q_{i,j,k}\right) s_x + \\
& \left(q_{i+1/2,j+1/2,k} - q_{i+1/2,j-1/2,k}\right) t_x + \\
& \left(q_{i+1/2,j,k+1/2} - q_{i+1/2,j,k-1/2}\right) p_x
\end{aligned} \tag{4.8}
$$

Assim, as pressões com índices fracionários em (4.8) recaem em arestas da

malha e precisam ser interpoladas com base nos quatro valores mais próximos.

4.5 Transporte de escalares e partículas

Da mesma forma que no *Stable Fluids* original, podemos simular o transporte de grandezas escalares pelo fluido numa malha curvilínea aplicando a mesma sequência de operadores descrita anteriormente na Seção 3.6. Novamente, as adaptações aos operadores de advecção e difusão são análogas às apresentadas para a integração do campo de velocidade do fluido. Quanto às etapas de fonte externa e de extinção, podem ser realizadas de maneira idêntica ao tratamento de malhas regulares, pois os operadores e grandezas envolvidos não dependem da parametrização do domínio.

Quanto ao transporte de partículas sem massa pelo fluido, podemos também manter a estratégia do *Stable Fluids* original e realizar a integração explícita de suas posições no tempo. Numa aproximação de primeira ordem, por exemplo:

$$\vec{x}'_{particula} = \vec{x}_{particula} + h\vec{u}\left(\vec{x}_{particula}\right) \tag{4.9}$$

O processo de integração segue as mesmas regras e adaptações que foram discutidas na Seção 4.1 para o acompanhamento das partículas Lagrangianas para trás durante a etapa de advecção.

É importante observar que é mais conveniente armazenar as posições das partículas diretamente no espaço paramétrico, já que suas localizações no espaço do mundo podem ser facilmente aproximadas por meio da interpolação das posições dos nós da malha em suas coordenadas. Por outro lado, a conversão inversa de uma posição do espaço do mundo para o paramétrico exigiria a custosa resolução de um pequeno sistema não linear.

4.6 Condições de contorno

É importante ressaltar que, como o cálculo de derivadas por diferenças finitas na malha exige que acessemos os vizinhos diretos de cada ponto amostral, precisamos de uma camada extra de células ao redor do domínio de simulação. Os valores das propriedades do fluido nessa camada extra podem então ser utilizados para impor condições de contorno em cada etapa da integração [4, 32]. Podemos, por exemplo, considerar que há uma parede lisa (livre de atrito) ou rugosa ao redor do domínio, com alguns pontos em que o fluido é obrigado a entrar ou sair com uma velocidade determinada, como ilustra a Figura 4.3.

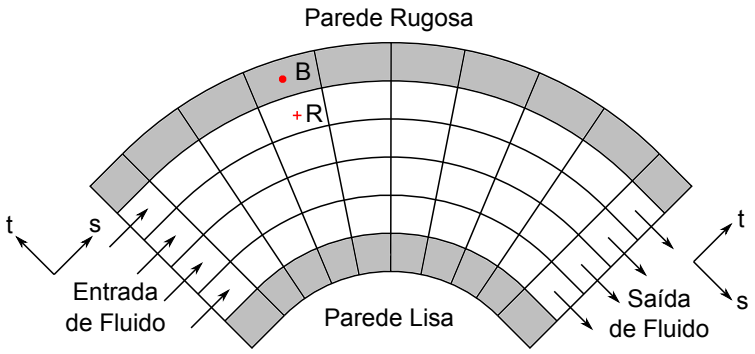


Figura 4.3 – Exemplos de condições de contorno para a velocidade de um fluido em duas dimensões. A amostra de referência R pode ser encontrada adicionando-se $(0, -1)$ às coordenadas paramétricas (s, t) da amostra de borda B.

Para que possamos atribuir valores à pressão q_B e à velocidade $\vec{u}_B$ de uma amostra de borda B como a da Figura 4.3, precisamos primeiramente encontrar uma amostra vizinha R interna ao domínio, que será utilizada como referência. No caso geral tridimensional com suporte a obstáculos internos, podemos encontrar R adicionando *offsets* conhecidos (s, t, p) às coordenadas paramétricas de B. Uma vez que tenhamos R, podemos calcular as condições de contorno como se segue, onde $\parallel$ e $\perp$ se referem às componentes paralela e ortogonal à fronteira, respectivamente.

$$q_B = q_R \tag{4.10}$$

$$\vec{u}_{B\parallel} = s_\parallel \cdot \vec{u}_{R\parallel} \tag{4.11}$$

$$\vec{u}_{B\perp} = s_\perp \cdot \vec{u}_{R\perp} + o_\perp \cdot \hat{\sigma}^*_{B\perp} \tag{4.12}$$

Nas equações (4.11) e (4.12), $s_\parallel$, $s_\perp$ e $o_\perp$ são parâmetros arbitrários de escala e *offset*. O vetor $\hat{\sigma}^*_{B\perp}$, por sua vez, corresponde à direção ortogonal à fronteira local. Esse vetor pode ser obtido pela conversão de um dos vetores unitários de base do espaço paramétrico ($\hat{s}$, $\hat{t}$ ou $\hat{p}$, que chamaremos genericamente de $\hat{\sigma}_{B\perp}$) para as coordenadas do mundo, seguida de uma renormalização:

$$\hat{\sigma}^*_{B\perp} = \frac{J^T(\hat{\sigma}_{B\perp})}{\|J^T(\hat{\sigma}_{B\perp})\|} \tag{4.13}$$

A Tabela 4.1 fornece alguns exemplos de como os parâmetros das condições de contorno podem ser utilizados para se obter os diferentes comportamentos de fluido mostrados na Figura 4.3 nas fronteiras do domínio de simulação.

É importante perceber que as equações (4.11) e (4.12) estão expressas no espaço do mundo, de modo que seu efeito seja independente da geometria

Tabela 4.1 – Parâmetros das condições de contorno para a velocidade de um fluido em uma malha estruturada paramétrica de modo a se obterem diferentes comportamentos de escoamento.

Efeito	Malha colocalizada			Malha deslocada		
	$s_\parallel$	$s_\perp$	$o_\perp$	$s_\parallel$	$s_\perp$	$o_\perp$
Passagem livre	1	1	0	1	1	0
Parede lisa	1	−1	0	1	0	0
Parede rugosa	−1	−1	0	0	0	0
Entrada de fluido	0	0	u_{fixa}	0	0	u_{fixa}
Saída de fluido	0	0	u_{fixa}	0	0	u_{fixa}

local das células da malha. Por exemplo, se escolhemos $s_\parallel = s_\perp = 1$ e $o_\perp = 0$, desejamos que a velocidade resultante em B seja igual à de R no espaço do mundo e não em seus espaços paramétricos, que podem ter eixos distintos. De forma semelhante, se fazemos $s_\parallel = s_\perp = 0$ e $o_\perp = u_{fixa}$, queremos que essa velocidade de entrada ou saída do fluido seja imposta igualmente em qualquer ponto, independente do tamanho local das células da malha.

Apesar da observação acima, note que as componentes de velocidade $\vec{u}_{B\parallel}$ e $\vec{u}_{B\perp}$ em (4.11) e (4.12) estão sempre alinhadas com as coordenadas paramétricas locais em B. Assim, parece conveniente transformá-las para esse espaço. Além disso, podemos levar em conta também o padrão de amostragem das velocidades e o sistema de coordenadas em que elas são armazenadas, conforme analisamos a seguir.

Considerando amostras de velocidade numa malha colocalizada e armazenadas no espaço do mundo, precisamos utilizar tanto a matriz Jacobiana quanto sua inversa ao longo do processo: devemos converter os vetores para as coordenadas paramétricas de B (não de R), aplicar as escalas das condições de contorno e trazer os resultados de volta para o espaço do mundo. O *offset* ortogonal $o_\perp \cdot \hat{\sigma}^*_{B\perp}$ pode ser somado diretamente em coordenadas do mundo. Podemos expressar essas operações como se segue:

$$\vec{u}_{B\parallel} = J_B^T \left[s_\parallel \cdot \left(J_B^{-T} \vec{u}_R \right)_\parallel \right] \tag{4.14}$$

$$\vec{u}_{B\perp} = J_B^T \left[s_\perp \cdot \left(J_B^{-T} \vec{u}_R \right)_\perp \right] + o_\perp \hat{\sigma}^*_{B\perp} \tag{4.15}$$

Considerando agora amostras de velocidade numa malha deslocada e armazenadas no espaço paramétrico, a rigor precisaríamos primeiramente converter as velocidades de referência do espaço paramétrico de R para o espaço do mundo, para então seguir o mesmo procedimento descrito acima. No entanto, isso nos obrigaria a interpolar algumas componentes de velocidade em R e a descartar outras em B, o que significaria certo aumento de custo e

perda de informações. Alternativamente, podemos obter um resultado também aproximado utilizando um procedimento mais simples e eficiente: já que, por definição, cada amostra corresponde exatamente a uma componente ortogonal ou paralela à fronteira, trabalhamos apenas com os módulos das velocidades, como mostrado a seguir:

$$u_{B\parallel} = \frac{s_\parallel \left(u_{R\parallel}\sigma^*_{R\parallel}\right)}{\sigma^*_{B\parallel}} \qquad u_{B\perp} = \frac{s_\perp \left(u_{R\perp}\sigma^*_{R\perp}\right) + o_\perp}{\sigma^*_{B\perp}} \tag{4.16}$$

Nas equações (4.16), cada um dos valores σ^* corresponde ao módulo, no espaço do mundo, de um dos vetores $\hat{s}$, $\hat{t}$ ou $\hat{p}$, bases unitárias do espaço paramétrico em B ou R. Por exemplo, se estivermos considerando uma amostra sobre uma face ortogonal a $\hat{s}$, teremos $\sigma^* = \|(x_s, y_s, z_s)\|$, o que vale tanto para $\hat{s}$ paralelo quanto perpendicular à fronteira do domínio. Note que, nas equações, os módulos $u_\parallel$ e $u_\perp$ das componentes de velocidade são primeiramente convertidos do espaço paramétrico de R para coordenadas do mundo pela multiplicação por σ^*_R, e ao final são transformados para o espaço paramétrico de B pela divisão por σ^*_B. Isso é suficiente para garantir que o tamanho local das duas células não influencie as condições de contorno, mas é uma aproximação pelo fato de desconsiderar a possível rotação existente entre os eixos $\hat{\sigma}^*_B$ e $\hat{\sigma}^*_R$.

5
Implementação

Nosso simulador de fluidos em malhas estruturadas parametrizadas tridimensionais foi implementado em C++ padrão, com uso extenso da linguagem de programação CUDA [38] para arquiteturas paralelas de placas gráficas NVidia. Adicionalmente, como ferramentas de auxílio para a validação dos resultados, foram implementados algoritmos básicos de visualização utilizando-se a biblioteca gráfica OpenGL e a linguagem GLSL.

5.1
Dados da simulação

Antes de dar início à simulação de um fluido, definimos a geometria do domínio a partir de um vetor com as coordenadas dos nós da malha no espaço do mundo. A partir desses dados, podemos calcular e armazenar as posições dos centros de células, faces e arestas, utilizando coordenadas globais ou paramétricas conforme a necessidade. Calculamos também os termos das matrizes Jacobianas e suas inversas de acordo com (2.18), bem como todas as métricas k_* em (4.2), (4.5) e (4.7) e os símbolos de Christoffel em (4.4). Todas essas grandezas são consideradas propriedades da grade e armazenadas por célula ou aresta em texturas (*textures*) e vetores (*arrays*) CUDA tridimensionais com acesso somente de leitura, que oferecem a interpolação linear rápida e automática típica de placas gráficas. Convém lembrar que os padrões de amostragem dos Jacobianos e dos campos de propriedades do fluido foram discutidos anteriormente nas Seções 2.2 e 2.4.

Propriedades do fluido como velocidade e pressão, assim como divergência, densidades de tinta e outras grandezas simuladas, requerem acesso de leitura e escrita na memória da placa gráfica a partir de *kernels* CUDA. Por isso, dependendo da capacidade de computação CUDA disponível no *hardware* cliente, armazenamos seus valores ou em memória linear alinhada 2D (*linear pitched memory*) ou em vetores CUDA 3D. O armazenamento em três dimensões requer suporte especial do *hardware* para escrita via superfícies (*surfaces*) CUDA, mas oferece interpolação linear automática em todas as três dimensões, o que pode proporcionar uma melhora significativa de desempenho. Ressalta-

se ainda que, se uma propriedade precisar ser consultada e atualizada por um único *kernel* em diferentes coordenadas, devemos utilizar duas áreas de memória distintas e alternar entre os ponteiros para elas após cada iteração.

Influências externas, como forças de campo e fontes de tinta, também são armazenadas em vetores CUDA 3D, nos mesmos pontos amostrais que as grandezas afetadas (velocidades do fluido e densidades de tinta, respectivamente, para os exemplos citados). Se alguma atualização se faz necessária, utilizamos uma área temporária de memória em página bloqueada (*page-locked*) para transferir os novos dados assincronamente sem comprometer o desempenho da simulação.

Vetores de dados usados diretamente pelos algoritmos de visualização são armazenados como recursos OpenGL: *pixel buffer objects* (PBOs), *vertex buffer objects* (VBOs) ou texturas. Como um exemplo, as posições de partículas sem massa em coordenadas paramétricas são mantidas em um VBO e acessadas em CUDA por meio de um ponteiro para *cudaGraphicsResource*.

Finalmente, informações relacionadas às condições de contorno são armazenadas e atualizadas da mesma maneira que influências externas. Para cada amostra de borda B, armazenamos o *offset* em coordenadas paramétricas necessário para se chegar à sua amostra de referência R, sua vizinha interna mais próxima (veja a Figura 4.3). Esse dado também assume valores especiais para indicar que uma amostra é interna ao fluido ou a um obstáculo. Além disso, armazenamos ainda os parâmetros $s_{\parallel}$, $s_{\perp}$ e $o_{\perp}$, que controlam como as componentes tangencial e ortogonal da velocidade em cada amostra de borda dependem de seus valores na amostra de referência, conforme explicado na Seção 4.6.

É interessante notar que algumas propriedades, como a pressão e a divergência das velocidades do fluido, podem ser sempre acessadas de maneira coalescida, especialmente em *hardware* mais moderno com *cache* de memória global. Nesses casos, a utilização de vetores CUDA se torna opcional, já que um simples acesso direto à memória apresenta a mesma eficiência. Entretanto, mesmo nesses casos, deve-se ter em mente que a interpolação automática do *hardware* é um motivo forte para a utilização de texturas ou superfícies.

Deve-se ressaltar que CUDA não oferece suporte nativo a texturas ou vetores de tipos com três componentes. A linguagem expande automaticamente esse tipo de dado para quatro componentes, o que aumenta desnecessariamente o consumo de memória e pode reduzir significativamente o desempenho da simulação, já que a eficiência de nosso algoritmo é fortemente limitada pela largura de banda disponível para acessos e transferências. Assim, ao lidarmos com grandezas tridimensionais, como coordenadas (x, y, z) ou (s, t, p),

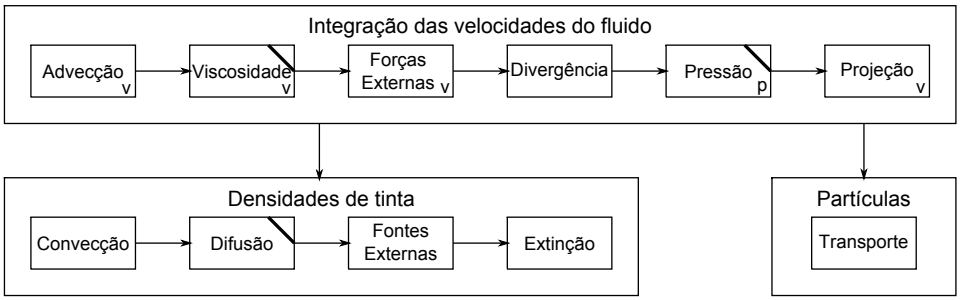


Figura 5.1 – Diagrama de blocos do simulador. As faixas diagonais representam solucionadores de sistemas lineares esparsos. As pequenas letras "v" e "p" indicam a necessidade de se atualizar as velocidades e pressões de fronteira, respectivamente.

preferimos sempre utilizar cada componente separada individualmente em sua própria textura ou vetor. O mesmo ocorre com as métricas da malha e símbolos de Chistoffel, presentes na simulação sempre em grupos de três ou nove valores.

Outro ponto importante a se mencionar é que nossa implementação utiliza valores em ponto flutuante de precisão simples para todos os tipos reais da simulação. Um desempenho ainda mais elevado poderia ser conseguido com a utilização de meia precisão, comum em algoritmos gráficos, para todas as grandezas exceto a pressão [39]. Por outro lado, caso se desejem resultados melhores para aplicações científicas, podem-se utilizar valores com precisão dupla, especialmente para a pressão. Um tratamento diferenciado para essa grandeza pode ser justificado pela provável melhora na convergência do algoritmo numérico utilizado em sua determinação e pela grande influência que esse campo exerce sobre o comportamento geral do escoamento. Note que placas gráficas modernas conseguem lidar normalmente com operações de precisão dupla, muitas delas apenas duplicando o esforço computacional associado a cada operação matemática e acesso de memória.

5.2 Passos da simulação

A Figura 5.1 apresenta um diagrama de blocos contendo todas as operações de alto nível realizadas por nosso simulador. A linha superior mostra os passos responsáveis pela integração das velocidades do fluido. A linha inferior, por outro lado, apresenta a integração de densidades de tinta e o transporte de partículas sem massa pelo escoamento, operações necessárias para se gerar uma visualização que permita avaliar e interpretar qualitativamente os resultados.

Com exceção dos blocos marcados com uma faixa diagonal na Figura 5.1, cada passo da simulação é realizado por um *kernel* CUDA especializado. A

implementação é bastante direta: não há necessidade de se utilizar a memória compartilhada da placa gráfica, já que todos os acessos à memória global são naturalmente coalescidos ou apresentam boa localidade espacial para a utilização de texturas e superfícies CUDA. Cada disparo de *kernel* utiliza um bloco de $(32 \times 2 \times 2)$ ou $(32 \times 4 \times 4)$ *threads*, dependendo de limitações impostas à utilização de registradores.

É importante notar que, quando um *kernel* altera as velocidades ou pressões no interior do fluido, torna-se necessário atualizar também os valores de fronteira dessas grandezas. Nesse caso, disparamos um *kernel* adicional para realizar essa operação, o que está representado com pequenas letras "v" e "p" na Figura 5.1. A atualização das pressões de borda, no entanto, só é utilizada em conjunto com o método de iterações de Jacobi. Quando utilizamos um solucionador de sistemas esparsos mais avançado, os valores corretos de pressão já são obtidos diretamente como parte da solução.

Os passos mais custosos do algoritmo são as resoluções das equações de Poisson, destacadas por faixas diagonais na Figura 5.1, em especial a inevitável resolução das pressões. Investigamos duas técnicas diferentes para lidar com esse problema: as iterações de Jacobi simples e o método do Gradiente Biconjugado Estabilizado (BiCGStab). A primeira técnica, simples e facilmente paralelizável, foi implementada como um laço iterativo fazendo repetidas chamadas a um pequeno *kernel*. Para o BiCGStab, por outro lado, optamos por utilizar uma rotina de alto desempenho disponível na biblioteca CUSP [30], que faz uso de um precondicionador *Multigrid* Algébrico baseado em Agregação Suavizada. Deve-se lembrar que o método do Gradiente Conjugado básico e o precondicionador de Cholesky Incompleto não podem ser aplicados, já que as matrizes dos sistemas não são simétricas quando consideramos os Jacobianos.

Voltamos agora nossa atenção para a etapa de transporte de partículas sem massa pelo fluido. É importante notar que não existe nenhuma garantia de ordenação espacial relativa entre essas partículas. Na verdade, conforme a simulação avança, há uma tendência de embaralhamento entre elas. Se executarmos o *kernel* de transporte nessas condições, o acesso à textura de velocidades do fluido tende a se tornar cada vez mais próximo de aleatório, o que pode comprometer o desempenho do algoritmo, já que o *hardware* gráfico é otimizado para acessos com alto grau de localidade. Assim, torna-se necessário realizar uma ordenação espacial no vetor de posições das partículas como primeiro passo da etapa, o que é feito com a utilização da biblioteca *thrust* que faz parte do ambiente CUDA.

Finalmente, para a visualização dos resultados, as densidades de tinta e

partículas sem massa transportadas pelo fluido são desenhadas utilizando-se *shaders* GLSL simples. As densidades de tinta são visualizadas por meio de hexaedros semitransparentes com opacidade proporcional a seu valor, enquanto cada partícula é representada apenas por um ponto. Ressalta-se que técnicas de visualização avançadas estão fora do escopo deste trabalho.

6
Resultados e discussão

Nesta seção, apresentamos alguns exemplos de domínios, descrevemos experimentos de validação, e discutimos a corretude e o desempenho de nossos resultados.

6.1
Grades retangulares distorcidas

Como uma forma de validação básica inicial, aplicamos nosso solucionador a uma grade regular tridimensional com células de tamanho $[\delta_x, \delta_y, \delta_z]^T$, como mostrado na Figura 6.1a. Nessas condições, as equações do Capítulo 4 se reduzem às suas versões originais do Capítulo 3. Verificamos que o comportamento resultante do fluido foi idêntico ao gerado pelo algoritmo *Stable Fluids* padrão.

Testamos em seguida algumas discretizações não regulares do mesmo domínio em forma de paralelepípedo, de modo a avaliarmos a sensibilidade do nosso algoritmo a variações de tamanho e geometria das células. Primeiramente, a malha representada na Figura 6.1b apresenta uma região central com amostras gradativamente mais condensadas, uma característica frequentemente utilizada para a obtenção de uma simulação mais detalhada em regiões específicas. Como condições de contorno de nosso experimento, utilizamos paredes rugosas ao redor de toda a região (Tabela 4.1), com a imposição extra de entrada de fluido pelo centro da face esquerda do domínio. A velocidade de entrada foi mantida constante com valor por segundo igual à metade do comprimento da região. Utilizamos uma discretização de tamanho $(256 \times 32 \times 32)$, passo de tempo de $0,01s$, viscosidade zero e advecção com Runge-Kutta de segunda ordem.

Para estabelecermos um *ground truth* para nosso experimento, observamos primeiramente o seu efeito na malha regular, com a utilização de 100 iterações de Jacobi para a resolução das pressões. O resultado pode ser visto na Figura 6.1c, onde as componentes RGB de cor representam cada uma das componentes XYZ de velocidade (as componentes de cor estão restritas ao intervalo $[0, 1]$, de modo que velocidades negativas não produzem cor e módu-

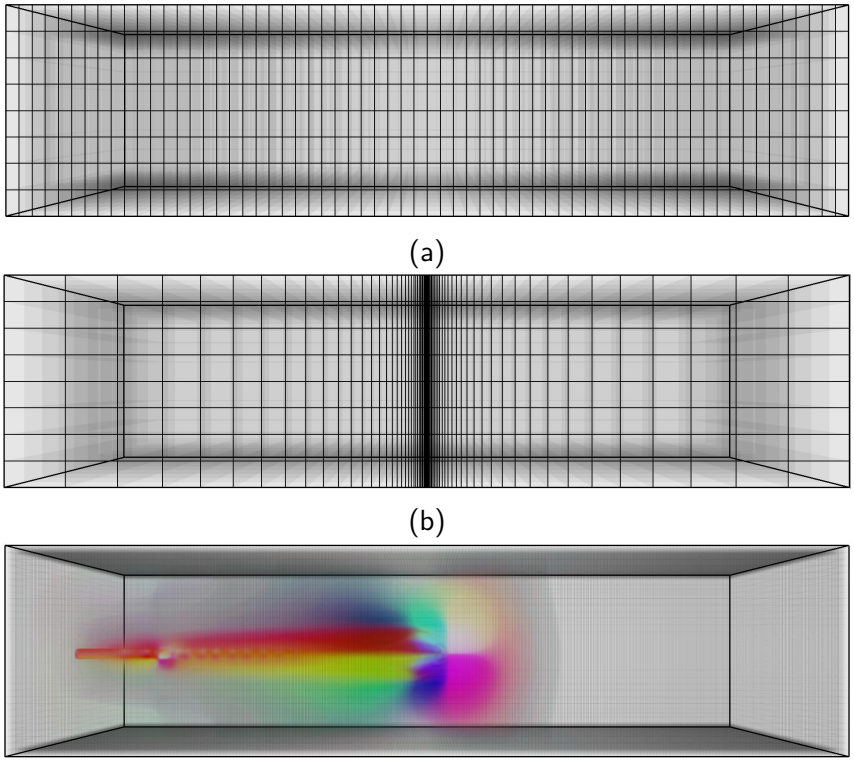


Figura 6.1 – Testes envolvendo um domínio retangular de tamanho $(256 \times 32 \times 32)$ com uma concentração de células na região central ao longo do eixo longitudinal. (a) Geometria da malha regular básica. (b) Geometria da malha paramétrica condensada. (c) Resultado da aplicação de uma força ao longo do eixo na malha regular, com as componentes de cor RGB codificando as velocidades do fluido nos eixos XYZ, respectivamente.

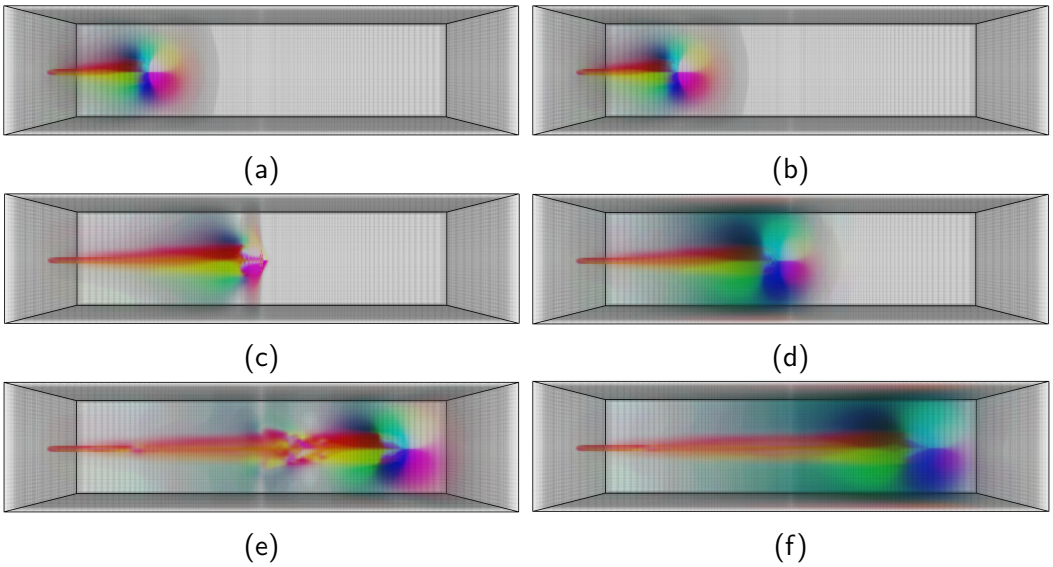


Figura 6.2 – Evolução do vórtice descrito na Figura 6.1 quando na malha paramétrica condensada. Lado esquerdo: resolução de pressão com 100 iterações de Jacobi. Lado direito: resolução de pressão com 10.000 iterações de Jacobi.

los superiores à unidade resultam em cores saturadas). Como se pode notar na figura, ocorreu o surgimento e a propagação de um vórtice atravessando o domínio de um lado ao outro. Verificamos que o movimento se deu com velocidade aproximadamente constante.

Passando então para a malha condensada, repetimos o procedimento acima com as mesmas configurações, o que resultou na sequência ilustrada do lado esquerdo da Figura 6.2. Percebemos que o vórtice diminuiu de tamanho e velocidade ao passar pela região com elevada concentração de amostras, recuperando-se em seguida ao ultrapassá-la, porém com o aparecimento de alguns artefatos nas velocidades ao longo de sua trajetória. Verificamos que esse tipo de comportamento indesejado se deve à resolução das pressões do fluido com precisão insuficiente: em regiões densamente amostradas, o efeito de uma célula com divergência diferente de zero precisa se propagar para um número maior de amostras de pressão para que o vórtice resultante cubra toda sua área de influência. Assim, para efeito de comparação, elevamos para 10.000 o número de iterações de Jacobi utilizadas na determinação das pressões. O resultado, ilustrado na sequência do lado direito da Figura 6.2, apresentou qualidade visivelmente superior, sem que o vórtice sofresse qualquer tipo de efeito adverso ao passar pela região de amostras concentradas.

Dando prosseguimento a nossos experimentos, avaliamos a sensibilidade de nosso algoritmo a variações de geometria das células de uma discretização. A malha representada na Figura 6.3a foi distorcida ao longo dos eixos transversais com a utilização de uma função senoidal, resultando assim a geometria mostrada na Figura 6.3b. Além disso, ela foi envolta por uma camada de células de fronteira não distorcidas, de modo que o fluido permanecesse confinado num domínio retangular análogo aos anteriores.

Para este teste, utilizamos uma discretização de tamanho $(256 \times 32 \times 32)$, passo de tempo de $0,01s$, viscosidade zero, resolução de pressão com 100 iterações de Jacobi, e advecção com Runge-Kutta de quarta ordem para capturar melhor variações senoidais. Como condições de contorno, ativamos a entrada de fluido pela face esquerda do domínio e sua saída pela face direita, mantendo as demais fronteiras como paredes rugosas (Tabela 4.1). Após algumas leves oscilações iniciais, verificamos a estabilização do escoamento com vetores velocidade razoavelmente horizontais, como mostra a Figura 6.3c. Note, na figura, que a disposição senoidal dos vetores está associada a suas origens, localizadas nos centros das células. Suas direções, por outro lado, correspondem a cada uma das pequenas linhas, todas próximas da horizontal. Essa característica correta nos indica que a curvatura da discretização não está afetando negativamente o algoritmo.

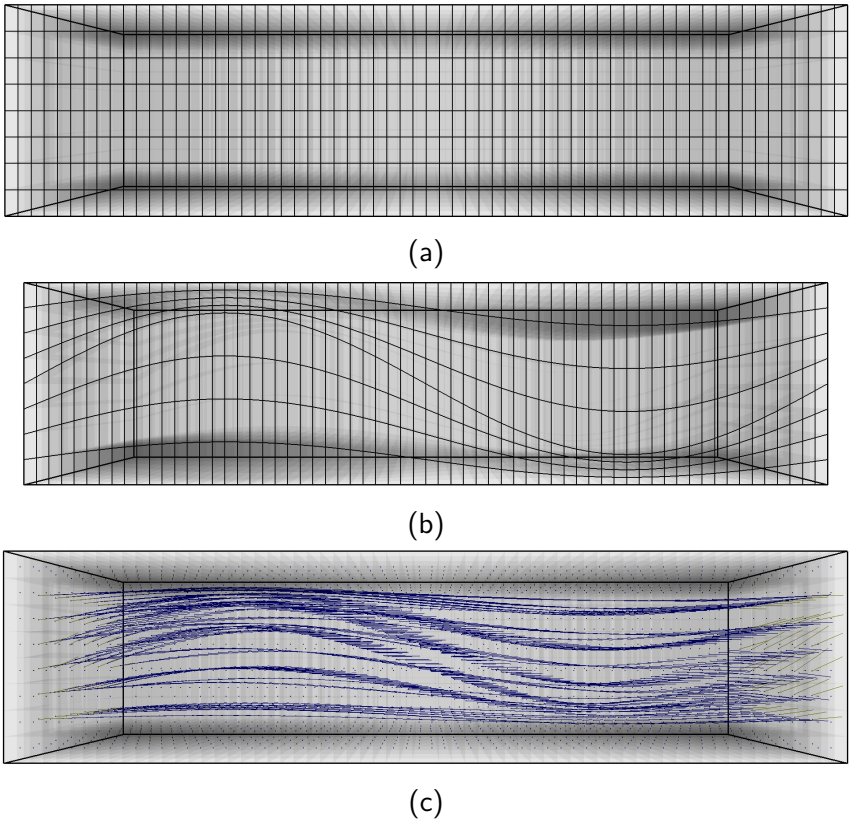


Figura 6.3 – Testes envolvendo um domínio retangular de tamanho $(256 \times 32 \times 32)$ com uma discretização interna distorcida por funções senoidais. (a) Geometria da malha regular básica. (b) Geometria da malha paramétrica com distorção senoidal. (c) Representação simples dos vetores velocidade ao longo do domínio como linhas.

Os experimentos anteriores foram repetidos para amostragens colocalizadas e deslocadas, com velocidades cartesianas e paramétricas. Os resultados *menos* satisfatórios foram encontrados com a utilização de velocidades locais no domínio com distorção trigonométrica. Aparentemente, essa configuração exige maior cuidado na preparação da malha (na discretização utilizada, a camada de ligação entre a parte deformada e a fronteira regular inclui células com baixa qualidade).

6.2 Padrões de amostragem

Para observarmos com clareza uma das vantagens da malha deslocada em relação à colocalizada, realizamos um teste simples envolvendo um domínio regular rotacionado nos três eixos e discretizado em $(50 \times 50 \times 50)$ células. Nessa simulação, deixamos ativos apenas os operadores de força externa e projeção. Em seguida, aplicamos uma força na região central do domínio, alinhada com o eixo da coordenada paramétrica s, e analisamos o resultado após algum tempo de execução.

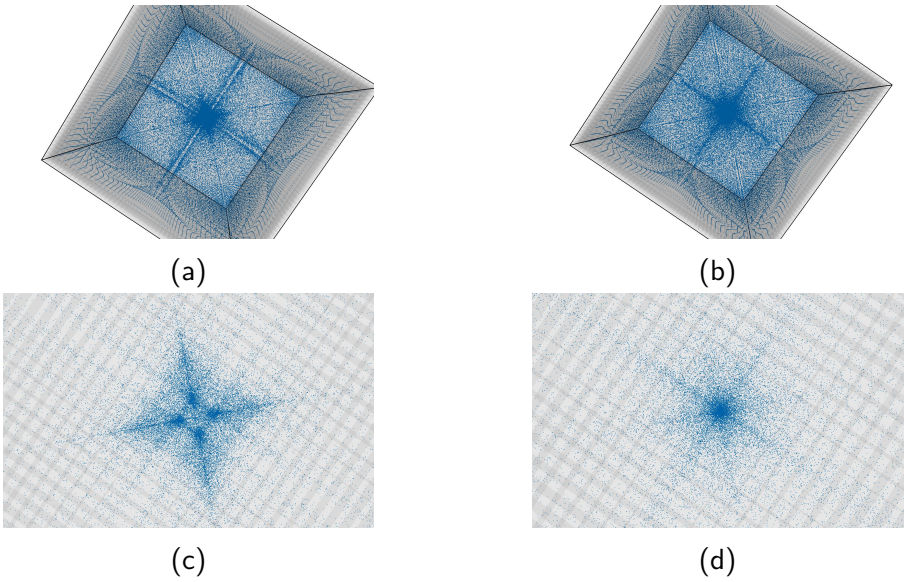


Figura 6.4 – Testes da projeção de velocidades de um fluido em um domínio regular rotacionado com diferentes padrões de amostragem: (a) malha colocalizada; (b) malha deslocada; (c) detalhe da malha colocalizada mostrando a parede atingida pelo fluxo de partículas; (d) detalhe análogo para a malha deslocada.

A Figura 6.4 mostra o resultado obtido, com a direção da força aplicada entrando na página. Observando a linha superior com cuidado, podemos perceber que em ambos casos a projeção das velocidades causou o aparecimento de um grande vórtice, com formato toroidal e eixo coincidente com a força aplicada, alcançando todo o domínio. No entanto, comparando as Figuras 6.4a e 6.4c com as Figuras 6.4b e 6.4d, é fácil detectar padrões de comportamento incorretos de alta frequência nos resultados obtidos com a malha colocalizada. Em particular, na Figura 6.4a, observamos linhas alternadamente mais claras e escuras, com apenas uma célula de espessura. De forma semelhante, na Figura 6.4c, existem células de fronteira alternadamente marcadas com mais e menos força. Essas diferenças de coloração são reflexo de diferentes concentrações de partículas, que por sua vez estão sendo carregadas por um campo de velocidade com valores se alternando nessa mesma frequência, um comportamento fisicamente improvável. Assim, fica clara a superioridade da malha deslocada.

Por outro lado, quanto ao padrão de amostragem deslocada que propusemos para as métricas do Jacobiano na Seção 2.4, a princípio não fomos capazes de observar nenhuma vantagem ou desvantagem na estratégia. Em nossos experimentos, não foi possível detectar nenhuma mudança significativa no comportamento do escoamento ao se alternar entre essas duas formas de amostragem. Na verdade, nos parece que o ideal é que o Jacobiano simplesmente varie o mais suavemente possível.

6.3
Degrau invertido

Como próximo conjunto de testes, escolhemos o cenário clássico de um escoamento passando por uma descida em degrau. Utilizamos nesse caso uma discretização de tamanho $(256 \times 32 \times 32)$, uma amostragem deslocada com velocidades no espaço global, um passo de tempo de $0,1s$, viscosidade zero e advecção com Runge-Kutta de segunda ordem. Para a resolução de pressão, utilizamos o Gradiente Biconjugado Estabilizado. Consideramos o fluido entrando pela face esquerda do domínio e saindo pela direita, mantendo as demais fronteiras e o degrau como paredes rugosas.

Começamos com um grid regular simples como o da Figura 6.1a, com as células internas ao degrau marcadas como obstáculo. A Figura 6.5 mostra

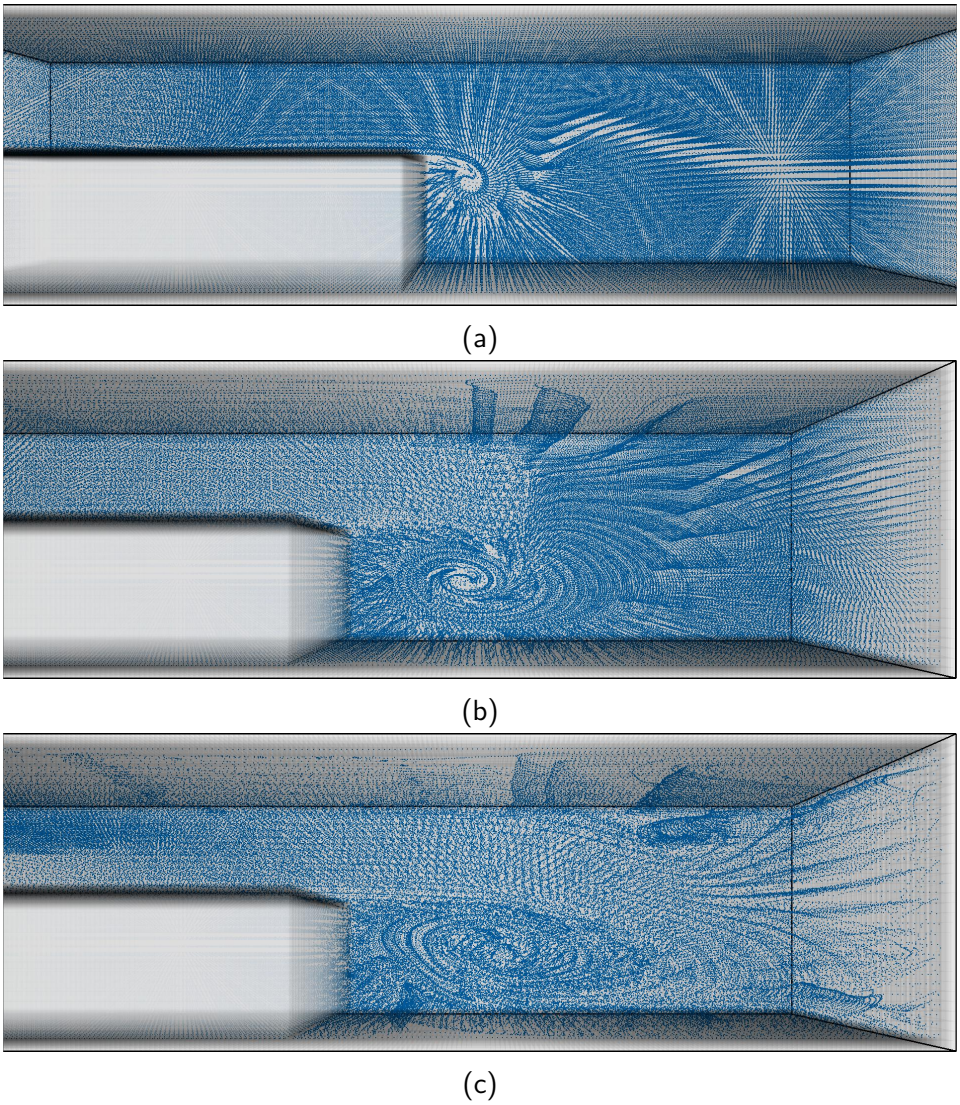


Figura 6.5 – Testes envolvendo um degrau voltado para trás em uma grade regular de tamanho $(256 \times 32 \times 32)$: (a) Surgimento de um vórtice; (b) Evolução do vórtice; (c) Formação de uma região de recirculação.

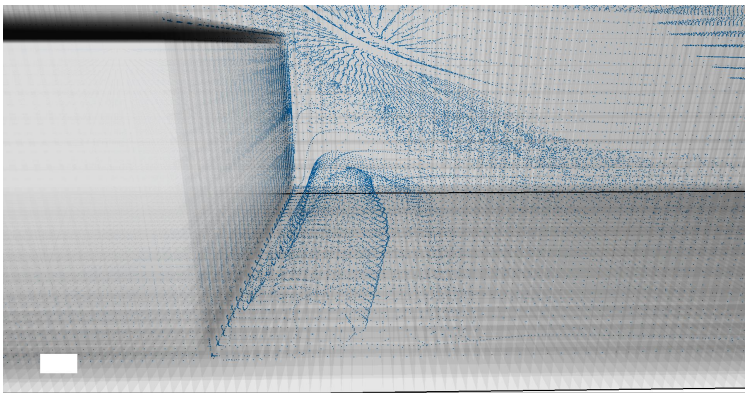


Figura 6.6 – Vórtice de Moffatt que surge junto à concavidade do degrau invertido.

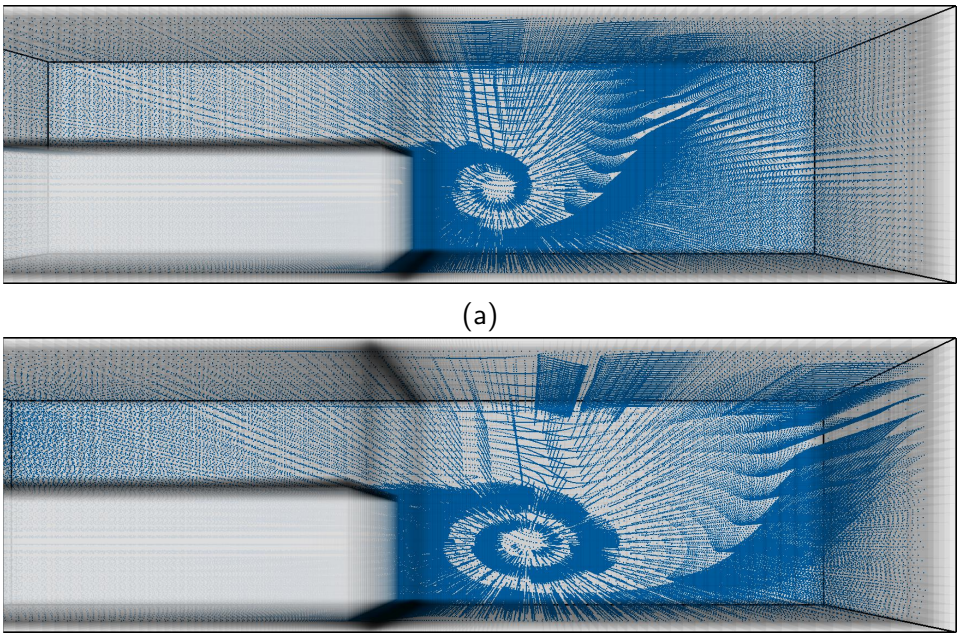
a origem e evolução de um vórtice, com a criação de uma grande zona de recirculação bem definida atrás do degrau. O vórtice surge a partir da aresta onde há o encontro de volumes de fluido com diferentes velocidades. A partir desse ponto, segue se deslocando para a frente e para baixo enquanto aumenta sua área de influência. Forma-se assim uma zona de recirculação junto ao degrau e abaixo do fluxo principal, com uma delimitação bem clara entre as regiões. Por alguns instantes, chega a se formar um vórtice de Moffatt [40] junto à concavidade, como se pode ver na Figura 6.6. Finalmente, após algum tempo o vórtice se eleva e é arrastado pelo escoamento, como resultado da diferença de pressão entre o fluxo principal acima e a região de estagnação abaixo. Essas observações mostram que o comportamento obtido está razoavelmente de acordo com [40] e outros resultados da literatura.

Continuando o experimento, substituímos a grade regular por uma malha condensada como a da Figura 6.1b. A região de maior concentração de células foi posicionada na zona logo após o degrau, onde se esperava o surgimento do vórtice descrito acima, para que ele pudesse ser simulado com maior detalhe.

A Figura 6.7 ilustra o resultado obtido. O vórtice se forma na região esperada e apresenta uma forma inicialmente bem mais arredondada, concordando com nossas expectativas. Logo se pode perceber sua influência sobre toda a área ao redor. Curiosamente, nesse experimento não foi verificado o aparecimento de um vórtice de Moffatt, pois o principal se destacou rapidamente do degrau, antes mesmo da formação completa de uma zona de recirculação.

6.4 Caminho por cotovelos

Passamos agora a testes envolvendo domínios com fronteiras curvas. Nesta e nas próximas seções, todos os fluidos foram simulados como percorrendo um duto da esquerda para a direita. Exceto quando especificado explicitamente, as simulações foram realizadas com um passo de tempo de



(a)

(b)

Figura 6.7 – Testes envolvendo um degrau voltado para trás em uma malha condensada. (a) Surgimento de um vórtice; (b) Evolução do vórtice.

$0,05s$, uma densidade de $1,0g/cm^3$, bordas rugosas, advecção e transporte por Runge-Kutta de segunda ordem e 100 iterações de Jacobi para a resolução das pressões e, quando considerada a viscosidade do fluido, 20 iterações de Jacobi para a resolução da difusão.

Como um primeiro caso simples, o caminho mostrado na Figura 6.8a foi projetado para testar o comportamento do fluido ao fazer curvas e fluir ao longo de cada eixo coordenado.

Observando a figura Figura 6.8b, pode-se notar que o fluido reage corretamente a cada uma das curvas ao longo do duto, com a formação de vórtices após cada dobra. Isso indica que os Jacobianos estão sendo corretamente aplicados, pois caso contrário o fluido se moveria como se estivesse num segmento completamente linear.

Os detalhes das Figuras 6.8c e 6.8d mostram vistas internas da última curva do duto e olhando para o extremo onde o fluido deixa o domínio. Pode-se perceber que um vórtice originário da primeira curva se propaga até o final, naturalmente induzindo o fluido a girar numa espiral.

É interessante notar a diferença de qualidade do vórtice original dependendo do tipo de amostragem utilizado na simulação. As Figuras 6.9a e 6.9b mostram o surgimento do vórtice quando se utiliza uma amostragem colocalizada e deslocada, respectivamente. É fácil perceber a presença de componentes de alta frequência influenciando o resultado no primeiro caso, contrastando com a maior suavidade do segundo.

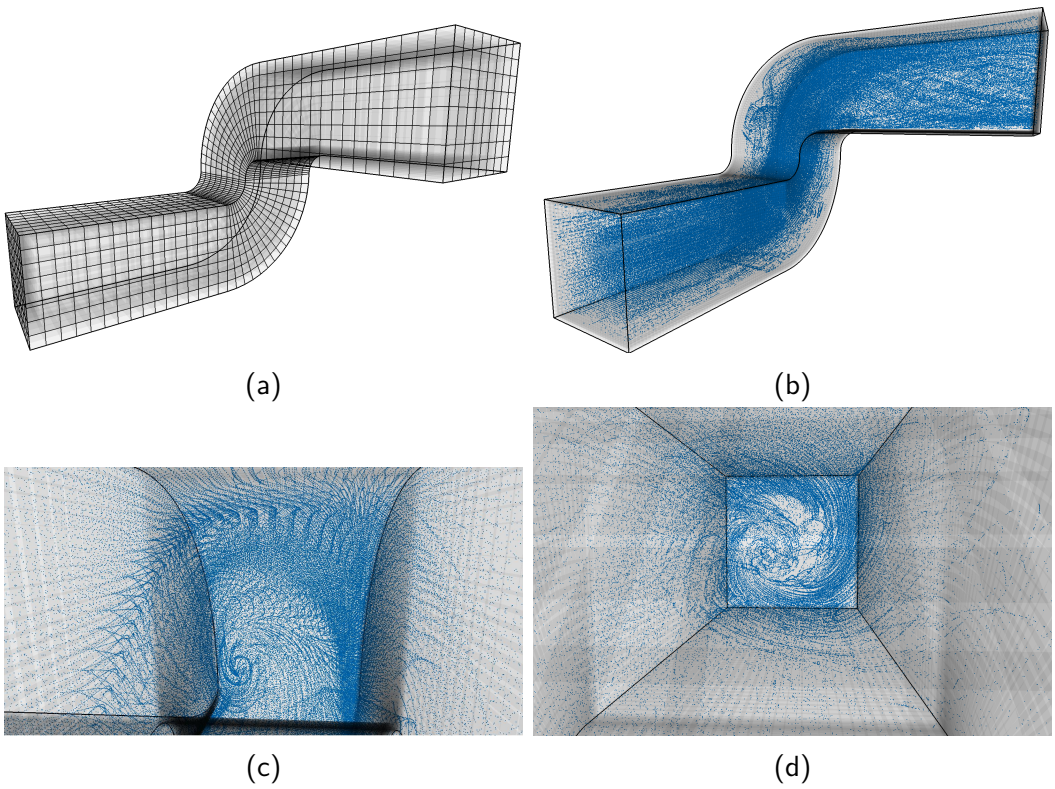


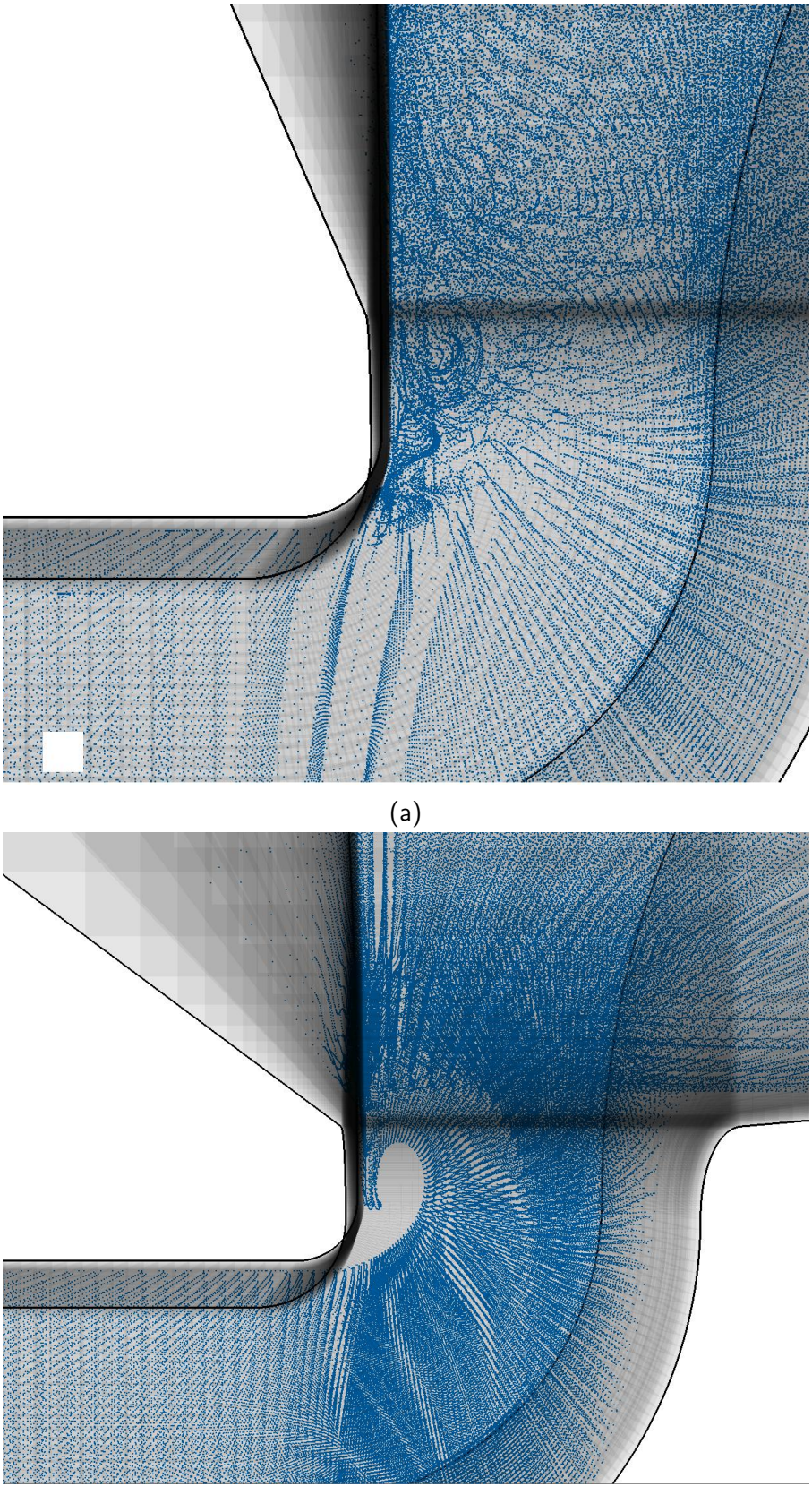
Figura 6.8 – Testes envolvendo o caminho com cotovelos em uma malha de tamanho $(256 \times 32 \times 32)$. (a) Geometria da malha com resolução reduzida para $(64 \times 8 \times 8)$ para melhor clareza. (b) Partículas sem massa transportadas pelo escoamento. (c) Detalhe mostrando a propagação de um vórtice desde a primeira curva até o último segmento. (d) Detalhe mostrando partículas se movendo em uma espiral induzida pela sequência de curvas.

6.5 Caminho com constrição

O caminho na Figura 6.10a foi projetado para testar a corretude do comportamento do fluido ao atravessar uma constrição como a de um tubo de Venturi.

Para um escoamento laminar incompressível (obtido com uma viscosidade de $0,1 Pa \cdot s$), a velocidade no interior da região constrita deve ser maior que no restante do duto, de maneira a manter constante o fluxo de massa por área. Isso pode ser verificado na Figura 6.11, que mostra a velocidade média do escoamento na direção x ao longo do caminho, considerando uma região constrita com um quarto da área de seção transversal normal. O gráfico mostra que, conforme aumentamos a precisão da resolução das pressões ou utilizamos solucionadores de melhor convergência, a velocidade no interior da região constrita tende a se estabilizar em quatro vezes a velocidade no exterior, conforme esperado.

Por outro lado, as Figuras 6.10b e 6.10c revelam um comportamento di-



(a)

(b)

Figura 6.9 – Comparativo da qualidade do vórtice gerado pela primeira curva do caminho: (a) Resultado em malha colocalizada. (b) Resultado em malha deslocada.

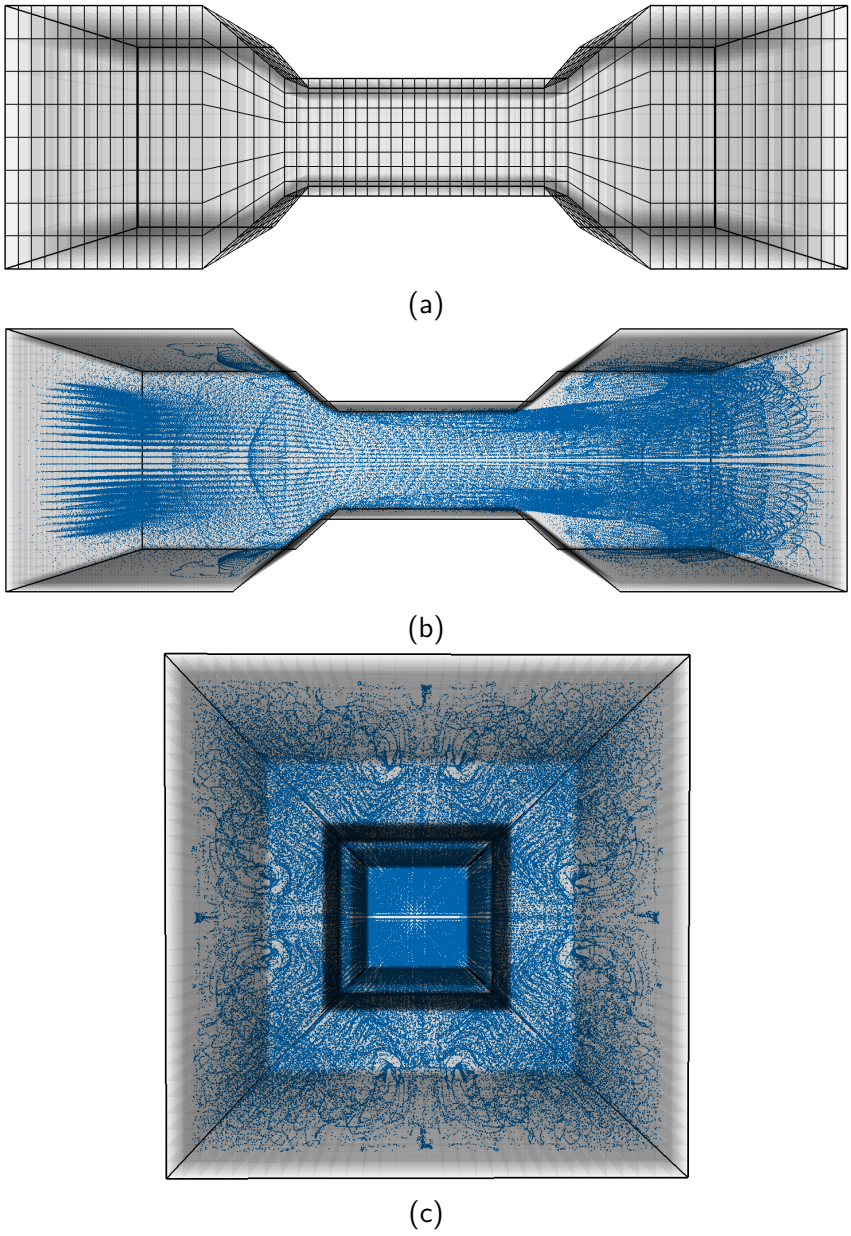


Figura 6.10 – Resultados obtidos para o caminho com constrição em uma malha de tamanho $(256 \times 32 \times 32)$. (a) Geometria da malha com resolução reduzida para $(64 \times 8 \times 8)$ para melhor clareza. (b) Partículas sem massa transportadas pelo escoamento. (c) Detalhe mostrando partículas presas em vórtices após a abertura da constrição.

ferente. Considerando um fluido sem nenhuma viscosidade além da dissipação numérica causada pela advecção semi-Lagrangiana, as características inerciais do escoamento tendem a se tornar dominantes. Como resultado, o movimento se torna turbulento, com o aparecimento de grandes vórtices na saída da constrição.

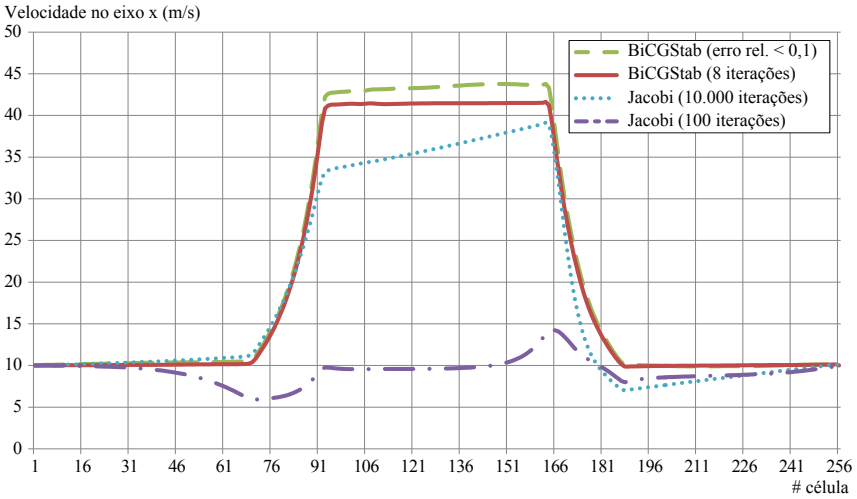


Figura 6.11 – Gráfico mostrando a velocidade média do escoamento na direção x através de seções ortogonais tomadas em células com índice i crescente ao longo do caminho com constrição. A área de seção transversal na região constrita é de um quarto da área em outros locais. O fluido entra no domínio com uma velocidade de $10m/s$ e se torna quatro vezes mais rápido no interior da região constrita. Cada curva foi obtida com um solucionador de Poisson e uma condição de parada diferente, conforme indicado.

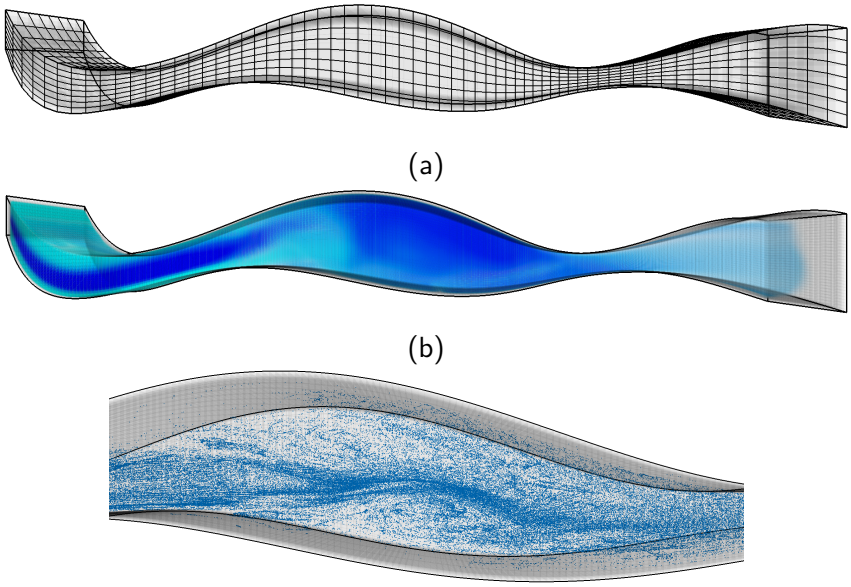


Figura 6.12 – Resultados obtidos para o caminho curvo suave em uma malha de tamanho $(256 \times 32 \times 32)$. (a) Geometria da malha com resolução reduzida para $(64 \times 8 \times 8)$ para melhor clareza. (b) Tinta azul transportada pelo fluido. (c) Detalhe mostrando partículas sem massa transportadas pelo escoamento.

6.6 Caminho curvo suave com obstáculo interno

A Figura 6.12a ilustra um caminho curvo suave, usado como preparação para o caso de teste final a seguir. Modelos suaves como esse são especialmente apropriados para nossa técnica, já que não ocorrem variações bruscas no

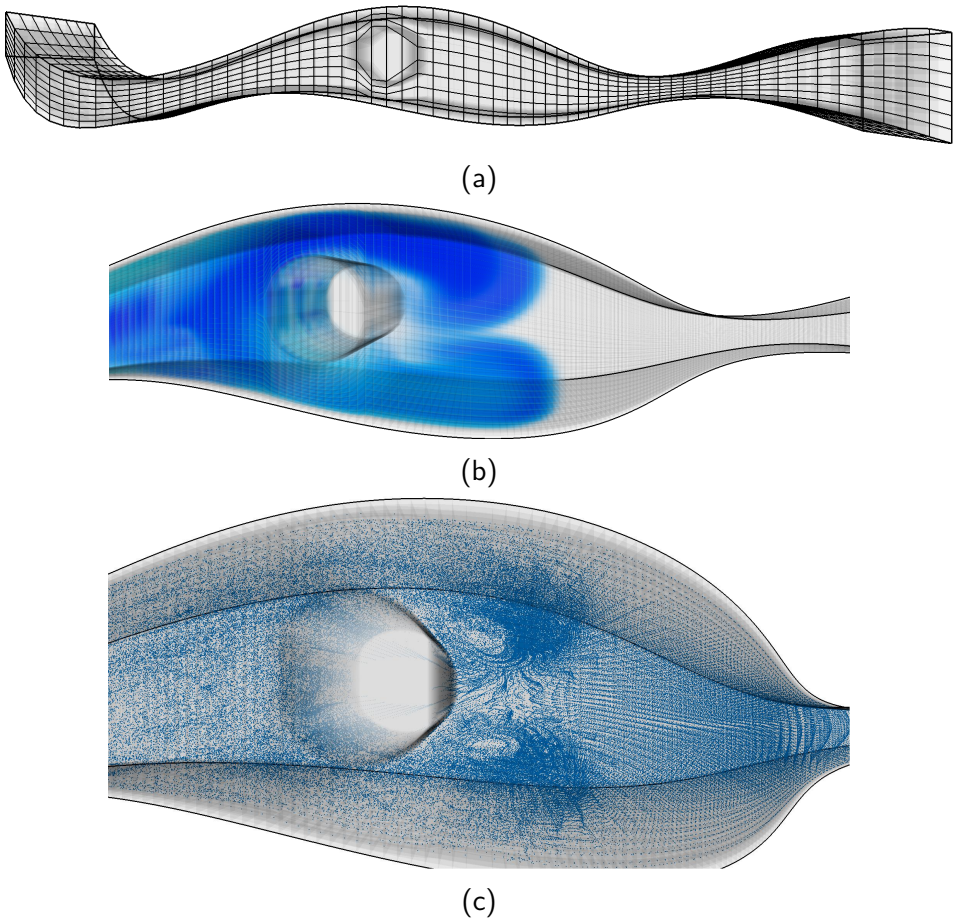


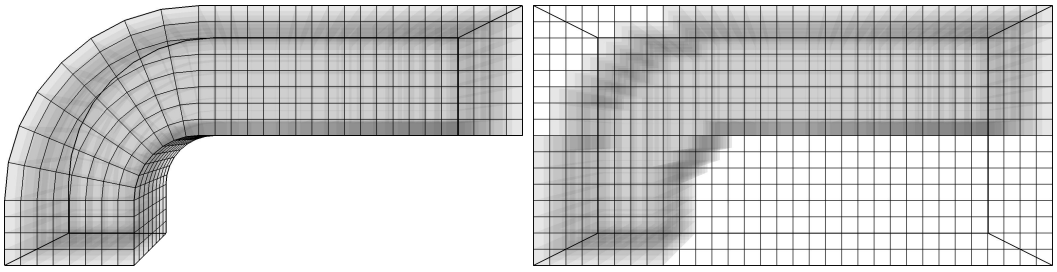
Figura 6.13 – Resultados obtidos para o caminho curvo suave com um obstáculo interno em uma malha de tamanho $(256 \times 32 \times 32)$. (a) Geometria da malha com resolução reduzida para $(64 \times 8 \times 8)$ para melhor clareza. (b) Tinta azul transportada pelo fluido. (c) Detalhe mostrando partículas sem massa transportadas pelo escoamento.

Jacobiano.

Note que, apesar da suavidade do caminho, grandes vórtices ainda podem aparecer em regiões que não estão na trajetória natural do escoamento, como consequência da diferença de pressão que surge entre áreas com diferentes velocidades do fluido (assim como após os cotovelos e na saída da região constrita nos exemplos anteriores). Esse efeito pode ser visto nas Figuras 6.12b e 6.12c.

Vamos agora incluir um obstáculo interno em meio ao caminho anterior, como ilustrado na Figura 6.13a. Apesar de nossa modelagem não dar suporte a buracos de fato na topologia da grade, podemos manipular uma região de modo que suas células aproximem o formato do obstáculo pretendido e marcá-las como paredes sólidas.

O efeito do obstáculo no escoamento pode ser identificado claramente nas Figuras 6.13b e 6.13c. Atrás do obstáculo, forma-se uma zona de turbulência com vórtices alternadamente para cima e para baixo, conforme esperado.



(a) (b)

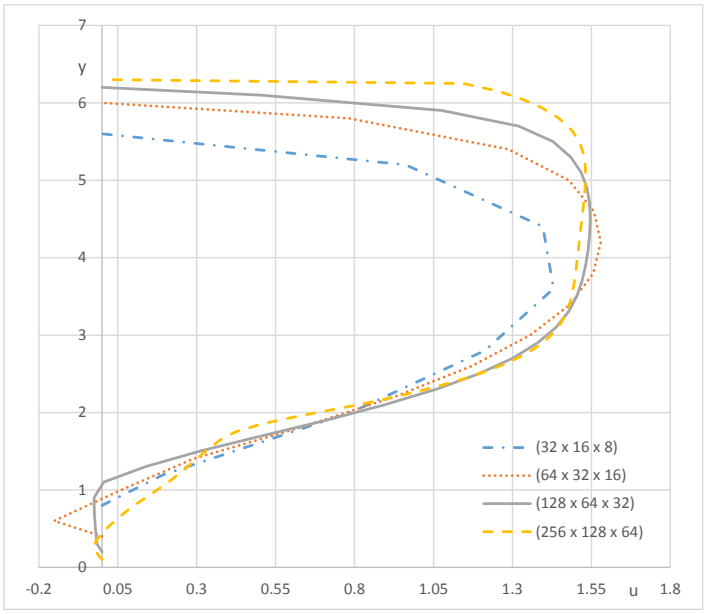
Figura 6.14 – Malhas de teste utilizadas na comparação qualitativa entre resultados obtidos com malhas curvilíneas e regulares. (a) Malha curvilínea de tamanho $(32 \times 8 \times 8)$. (b) Malha regular equivalente de tamanho $(32 \times 16 \times 8)$. Cada célula do grid regular tem 0,8m de tamanho, e a região curva tem raio médio igual à espessura do domínio.

6.7 Comparação qualitativa entre malhas regulares e curvilíneas

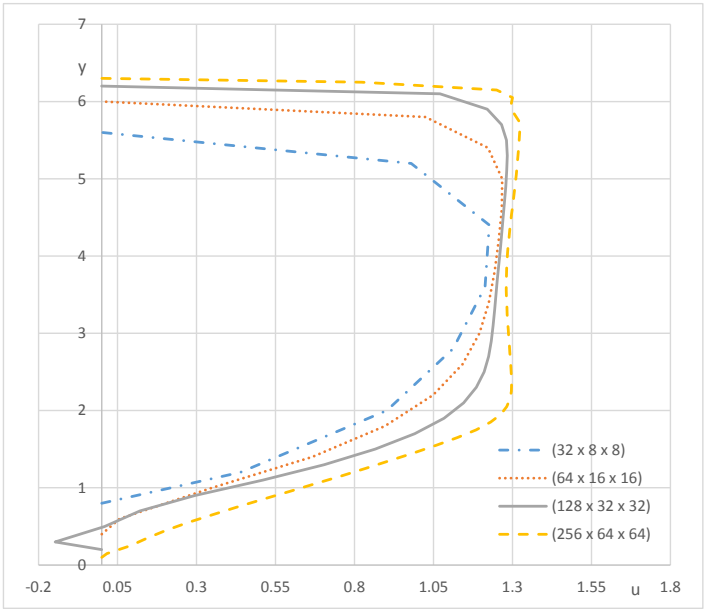
Para demonstrar a validade e a qualidade de nossos resultados com a utilização de malhas curvilíneas, realizamos alguns testes comparando o comportamento do escoamento obtido com nossa abordagem e com uma discretização regular. A Figura 6.14 mostra o domínio escolhido e versões em baixa resolução das duas discretizações estudadas. Salienta-se que essas subdivisões espaciais foram projetadas de forma que as células dos trechos retilíneos coincidam perfeitamente, garantindo que qualquer diferença entre os resultados se deva exclusivamente à simulação na região curva. Além disso, as duas malhas possuem exatamente o mesmo número de células em comprimento e profundidade, diferindo apenas no eixo vertical: a discretização regular precisa de resolução dobrada nessa direção para cobrir toda a área válida de simulação, e as células externas são marcadas como paredes.

Nestes testes, utilizamos um passo de tempo de $0,1s$, resolução de pressão com 100 iterações de Jacobi, e advecção com Runge-Kutta de segunda ordem, além de desconsiderarmos a viscosidade. Como condições de contorno, utilizamos paredes rugosas ao redor do domínio (Tabela 4.1), com entrada de fluido pela face inferior e saída pela face direita com velocidade de $1m/s$. Em ambas discretizações, consideramos as características usuais do *Stable Fluids*: amostragem colocalizada nos centros das células e representação de velocidades no espaço do mundo.

Para realizarmos uma análise qualitativa dos resultados, consideramos cada malha da Figura 6.14 em quatro níveis de resolução diferentes, começando com as indicadas na figura e dobrando o número de células em todas as dimensões a cada teste. No caso mais refinado da grade regular, também utilizamos um passo de tempo reduzido de $0,01s$ e resolução de pressão por BiCGStab para estabelecermos uma referência mais confiável. Uma vez que o



(a)



(b)

Figura 6.15 – Gráficos mostrando o perfil da componente u das velocidades do escoamento ao longo de uma linha vertical passando pelo centro da região horizontal próxima à saída do fluido. (a) Malhas regulares de tamanho indicado. (b) Malhas curvilíneas de tamanho indicado.

sistema atingisse regime permanente, medimos a componente u das velocidades do fluido ao longo de uma linha paralela ao eixo y. Tal linha, além de contida no plano de simetria do domínio na direção z, foi escolhida partindo-se do plano de saída do fluido e caminhando-se para a esquerda 1/4 do número de células da grade nessa dimensão.

As Figuras 6.15 e 6.15b mostram os perfis de velocidade obtidos com

o procedimento descrito acima para malhas regulares e curvilíneas, respectivamente. Em ambos casos, pode-se perceber que o aumento de resolução da discretização leva as velocidades a subirem mais rapidamente até seu valor máximo a partir da parede superior do domínio. Além disso, uma vez atingido esse valor, ele é mantido cada vez mais estável e ao longo de uma região maior dos gráficos antes de voltar a se reduzir. Esse comportamento indica que, conforme melhoramos a precisão da simulação, tendemos a obter escoamentos mais turbulentos, o que é esperado pelo fato de estarmos desconsiderando a viscosidade do fluido.

Comparando novamente as Figuras 6.15 e 6.15b, pode-se notar que o comportamento descrito acima é mais pronunciado quando utilizamos a malha curvilínea. De fato, uma malha curvilínea de tamanho $(64 \times 16 \times 16)$ já produz um perfil de velocidades razoavelmente semelhante à referência obtida com a grade regular mais refinada, de tamanho $(256 \times 128 \times 64)$, com passo de tempo reduzido e resolução de pressão mais precisa. Isso demonstra a superioridade da malha curvilínea para a obtenção de simulações de qualidade.

6.8 Desempenho, convergência e estabilidade

A fim de avaliarmos o desempenho e a estabilidade de nosso algoritmo, realizamos uma série de testes e medimos o tempo de execução em uma máquina com processador Intel Core i5 750 2.67GHz, 4GB de memória RAM e uma placa gráfica NVidia GeForce GTX 550 Ti.

O passo de maior custo computacional de qualquer simulador Euleriano de fluidos é a resolução das equações de Poisson, especialmente o cálculo das pressões, que impõe um limite à eficiência do algoritmo. Por isso, primeiramente investigamos o desempenho e a estabilidade de dois solucionadores de equações de Poisson: o método de iterações de Jacobi, simples e facilmente paralelizável, e o método do Gradiente Biconjugado Estabilizado (BiCGStab) com precondicionador *Multigrid* Algébrico, capaz de convergir em um número muito reduzido de iterações.

A Tabela 6.1 mostra o tempo de execução de um pequeno número de iterações do integrador BiCGStab, bem como o número de iterações de Jacobi que podem ser concluídas no mesmo período. Essas últimas têm um custo por iteração significativamente menor, mas nossos experimentos mostraram que não são capazes de alcançar resultados de qualidade comparável: muitas vezes, como mostra a última linha da Tabela 6.1, nem mesmo 200 iterações de Jacobi são capazes de alcançar um erro residual relativo tão pequeno quanto apenas 3 rodadas do BiCGStab. A diferença na convergência dos métodos

Tabela 6.1 – Número de iterações até que um limite de tempo seja atingido ou até que o erro residual relativo se torne menor do que um limiar.

Condição de Parada	Iterações BiCGStab	Iterações de Jacobi
t = 23 ms	1	40
t = 37 ms	2	65
t = 51 ms	3	90
Erro relativo < 0,1	3	>200

também pode ser vista claramente na Figura 6.11, que mostra como mesmo dez mil iterações de Jacobi não resultam num comportamento tão correto quanto algumas rodadas do BiCGStab.

Apesar da vantagem de desempenho do método BiCGStab, percebemos que sua estabilidade não é garantida, de modo que em alguns testes não houve convergência para uma solução apropriada, resultando num comportamento claramente incorreto do fluido. As iterações de Jacobi, por outro lado, forneceram resultados mais consistentes e nunca falharam em convergir. Por isso, consideramos essa técnica ainda útil para aplicações em que essa característica seja fundamental.

Outro ponto importante a se considerar é a estimativa inicial usada para os valores de pressão na resolução do sistema correspondente. Percebemos que os algoritmos geralmente convergiam mais rapidamente quando utilizamos a solução anterior como valor inicial em vez de partirmos de zero. Assim, incluímos essa estratégia como uma opção em nosso simulador e a utilizamos nos testes por padrão.

Nos testes seguintes, comparamos nosso desempenho com o de um simulador *Stable Fluids* tradicional. A Tabela 6.2 mostra os tempos obtidos para fluidos em grades regulares tridimensionais utilizando o *Stable Fluids* original e nosso método. O primeiro par de medidas de tempo em cada linha considera a simulação completa do fluido, incluindo a advecção de densidades de tinta e partículas sem massa, utilizando sempre 20 iterações de Jacobi para os passos de viscosidade e difusão. Já no segundo par de valores, os passos de viscosidade e difusão foram desligados. Em todos os casos, tempos de desenho da visualização foram desconsiderados.

Como se pode ver na Tabela 6.2, nosso método duplicou a triplicou o custo computacional do simulador *Stable Fluids* original. No entanto, a parametrização pode evitar artefatos e eliminar a necessidade de superamostrar o domínio próximo às fronteiras. Assim, se o número total de células necessárias na grade parametrizada for menor que um terço das usadas em uma grade regular equivalente, podemos obter desempenho e qualidade bastante

Tabela 6.2 – Comparação de desempenho entre o Stable Fluids e nosso método em uma grade de tamanho $(64 \times 64 \times 32)$, totalizando $128k$ células

Iterações de pressão	Simulação completa		Sem difusão	
	Stable Fluids	Método Proposto	Stable Fluids	Método Proposto
1 BiCGStab	18 ms	45 ms	11 ms	16 ms
2 BiCGStab	22 ms	53 ms	15 ms	25 ms
3 BiCGStab	26 ms	63 ms	19 ms	33 ms
50 Jacobi	14 ms	48 ms	07 ms	18 ms
75 Jacobi	16 ms	53 ms	10 ms	25 ms
100 Jacobi	19 ms	59 ms	12 ms	31 ms

Tabela 6.3 – Medidas de desempenho para cada operador e configuração de nosso simulador numa malha suave de tamanho $(64 \times 32 \times 32)$, totalizando $64k$ células

Operadores	Malha colocalizada		Malha deslocada	
	Velocidades Globais	Velocidades Paramétricas	Velocidades Globais	Velocidades Paramétricas
Advecção$_{(x,y,z)RK2}$	0.07 ms	0.20 ms	0.22 ms	0.90 ms
Advecção$_{(s,t,p)RK2}$	0.10 ms	0.10 ms	0.34 ms	0.28 ms
Difusão$_{(20\times Jacobi)}$	6.0 ms	6.0 ms	10.0 ms	9.0 ms
Força Externa	0.04 ms	0.04 ms	0.05 ms	0.05 ms
Divergência	0.07 ms	0.04 ms	0.08 ms	0.04 ms
Pressão$_{(100\times Jacobi)}$	11.5 ms	11.2 ms	13.4 ms	14.0 ms
Projeção	0.07 ms	0.06 ms	0.10 ms	0.25 ms
Total sem difusão	12.4 ms	11.8 ms	15.0 ms	15.3 ms
Total	18.4 ms	17.8 ms	25.0 ms	24.3 ms

superiores ao algoritmo tradicional.

Vamos agora observar melhor o custo associado a cada operador e configuração de nosso simulador. As Tabelas 6.3 a 6.5 mostram tomadas de tempo para cara operador do algoritmo, operando em cada uma das quatro configurações principais, em malhas curvilíneas suaves com $64k$, $256k$ e $1M$ células. O operador de projeção foi separado ainda em suas três partes componentes. Diferentemente dos testes anteriores, essas tomadas de tempo foram feitas numa máquina com processador Intel Core i7 2.4GHz, 12GB de memória RAM e placa gráfica NVidia GeForce 770M, que oferece capacidade de computação CUDA 3.0. Note que a resolução de sistemas lineares esparsos foi feita sempre com o método de Jacobi, mas que a Tabela 6.2 oferece uma base de comparação com o BiCGStab.

Em todas as tabelas, podemos perceber o alto custo associado à resolução de sistemas lineares esparsos, que domina fortemente a simulação. Também

Tabela 6.4 – Medidas de desempenho para cada operador e configuração de nosso simulador numa malha suave de tamanho $(256 \times 32 \times 32)$, totalizando $256k$ células

Operadores	Malha colocalizada		Malha deslocada	
	Velocidades Globais	Velocidades Paramétricas	Velocidades Globais	Velocidades Paramétricas
Advecção$_{(x,y,z)RK2}$	0.24 ms	0.77 ms	0.91 ms	4.2 ms
Advecção$_{(s,t,p)RK2}$	0.34 ms	0.36 ms	1.55 ms	1.31 ms
Difusão$_{(20\times Jacobi)}$	19.0 ms	19.0 ms	25.0 ms	23.5 ms
Força Externa	0.13 ms	0.13 ms	0.15 ms	0.15 ms
Divergência	0.25 ms	0.12 ms	0.25 ms	0.11 ms
Pressão$_{(100\times Jacobi)}$	32.0 ms	32.1 ms	32.6 ms	32.4 ms
Projeção	0.26 ms	0.21 ms	0.51 ms	0.33 ms
Total sem difusão	35.5 ms	33.8 ms	37.6 ms	35.7 ms
Total	54.5 ms	52.8 ms	62.6 ms	59.2 ms

Tabela 6.5 – Medidas de desempenho para cada operador e configuração de nosso simulador numa malha suave de tamanho $(256 \times 64 \times 64)$, totalizando $1M$ células

Operadores	Malha colocalizada		Malha deslocada	
	Velocidades Globais	Velocidades Paramétricas	Velocidades Globais	Velocidades Paramétricas
Advecção$_{(x,y,z)RK2}$	1.0 ms	3.2 ms	3.7 ms	17.6 ms
Advecção$_{(s,t,p)RK2}$	1.4 ms	1.4 ms	6.8 ms	5.3 ms
Difusão$_{(20\times Jacobi)}$	73.2 ms	72.5 ms	88.2 ms	84.5 ms
Força Externa	0.5 ms	0.5 ms	0.7 ms	0.6 ms
Divergência	1.0 ms	0.5 ms	1.0 ms	0.4 ms
Pressão$_{(100\times Jacobi)}$	117 ms	117 ms	117 ms	118 ms
Projeção	1.0 ms	0.8 ms	1.5 ms	1.5 ms
Total sem difusão	126 ms	124 ms	136 ms	131 ms
Total	199 ms	196 ms	224 ms	215 ms

podemos ver que o custo dos operadores de força externa, divergência e projeção é comparativamente negligenciável, e que a advecção é mais rápida quando realizada no mesmo espaço em que as velocidades estão representadas. Na verdade, o custo associado à advecção pode variar bastante dependendo do campo de velocidade atual do fluido e da utilização ou não de passos adaptativos (desligados nas tabelas).

Observe agora as últimas linhas das tabelas. Na Tabela 6.3, vemos que nosso algoritmo é capaz de rodar em tempo real para $64k$ células, mesmo com a difusão habilitada e um possível aumento de custo da advecção, bastando para isso a utilização de um passo de tempo correspondente à taxa de 30 quadros por segundo. Afinal, ao menos com a utilização do método de Jacobi,

o solucionador é incondicionalmente estável. Observando agora as últimas linhas da Tabela 6.4, podemos perceber que o algoritmo é capaz de se manter com desempenho altamente interativo mesmo para $256k$ células, desde que dispensando a etapa de difusão. Quanto à Tabela 6.4, ela mostra como o método se mantém escalável até ao menos $1M$ células.

7
Conclusão e trabalhos futuros

Nesta tese, propomos uma técnica rápida baseada num modelo físico simplificado para simular fluidos em domínios tridimensionais com fronteiras de forma arbitrária. Empregamos uma grade uniforme para guiar a simulação em uma discretização estruturada desses domínios no espaço paramétrico. Para isso, as matrizes Jacobianas que relacionam os espaços do mundo e paramétrico foram derivadas por diferenças finitas, e cada passo do integrador *Stable Fluids* de Stam [2] foi adaptado para acomodar as transformações de coordenadas. Como resultado, nossa abordagem foi capaz de gerar simulações de fluido eficientes em domínios complexos, como demostrado nos exemplos apresentados de caminhos tridimensionais com curvas, constrições e obstáculos internos.

Apesar de ter um custo por célula maior que o algoritmo *Stable Fluids* original, o método proposto pode produzir resultados convincentes para domínios de fronteiras curvas mesmo com um número relativamente pequeno de células, o que compensa o custo extra e permite simulações com desempenho interativo. Além disso, nossa abordagem evita completamente artefatos causados por fronteiras desalinhadas, que são difíceis de eliminar com a utilização apenas de grades regulares, mesmo com um grande número de células ou esquemas de refinamento adaptativo.

Nosso simulador foi implementado em CUDA, o que permitiu que aproveitássemos completamente a arquitetura massivamente paralela das placas gráficas atuais. Para os solucionadores de equações de Poisson, utilizamos uma versão paralela do método do Gradiente Biconjugado Estabilizado com precondicionador *Multigrid* Algébrico.

Como trabalhos futuros, pretendemos acrescentar ao nosso método o tratamento da aceleração da gravidade e o suporte à simulação da superfície livre do fluido, de modo a permitir a simulação de ondas do mar e outros fenômenos.

Referências Bibliográficas

[1] STAM, J.; FIUME, E.. **Depicting fire and other gaseous phenomena using diffusion processes**. In: PROC. 22ND ANNUAL CONF. COMPUT. GRAPH. INTERACT. TECH., SIGGRAPH '95, p. 129 – 136, 1995.

[2] STAM, J.. **Stable fluids**. In: PROC. 26TH ANNUAL CONF. COMPUT. GRAPH. INTERACT. TECH., SIGGRAPH '99, p. 121 – 128, 1999.

[3] HARRIS, M. J.. **Fast fluid dynamics simulation on the gpu**. In: Nguyuen, H., editor, GPU GEMS, chapter 38. Pearson Higher Education, 2004.

[4] FOSTER, N.; METAXAS, D.. **Modeling the motion of a hot, turbulent gas**. In: PROC. 24TH ANNUAL CONF. COMPUT. GRAPH. INTERACT. TECH., SIGGRAPH '97, p. 181 – 188, 1997.

[5] STAM, J.. **Flows on surfaces of arbitrary topology**. In: ACM SIGGRAPH 2003 PAP., SIGGRAPH '03, p. 724 – 731, 2003.

[6] ENRIGHT, D.; MARSCHNER, S. ; FEDKIW, R.. **Animation and rendering of complex water surfaces**. In: PROC. 29TH ANNUAL CONF. COMPUT. GRAPH. INTERACT. TECH., SIGGRAPH '02, p. 736 – 744. ACM, 2002.

[7] LOSASSO, F.; IRVING, G.; GUENDELMAN, E. ; FEDKIW, R.. **Melting and burning solids into liquids and gases**. IEEE Trans. Vis. Comput. Graph., 12:343 – 352, 2006.

[8] HONG, J.-M.; KIM, C.-H.. **Discontinuous fluids**. In: ACM SIGGRAPH 2005 PAP., SIGGRAPH '05, p. 915 – 920. ACM, 2005.

[9] LOSASSO, F.; SHINAR, T.; SELLE, A. ; FEDKIW, R.. **Multiple interacting liquids**. In: ACM SIGGRAPH 2006 PAP., SIGGRAPH '06, p. 812 – 819. ACM, 2006.

[10] GUENDELMAN, E.; SELLE, A.; LOSASSO, F. ; FEDKIW, R.. **Coupling water and smoke to thin deformable and rigid shells**. In: ACM SIGGRAPH 2005 PAP., SIGGRAPH '05, p. 973 – 981. ACM, 2005.

[11] BATTY, C.; BERTAILS, F. ; BRIDSON, R.. **A fast variational framework for accurate solid-fluid coupling**. In: ACM SIGGRAPH 2007 PAP., SIGGRAPH '07. ACM, 2007.

[12] ROBINSON-MOSHER, A.; SHINAR, T.; GRETARSSON, J.; SU, J. ; FEDKIW, R.. **Two-way coupling of fluids to rigid and deformable solids and shells**. In: ACM SIGGRAPH 2008 PAP., SIGGRAPH '08, p. 46:1 – 46:9. ACM, 2008.

[13] HORVATH, C.; GEIGER, W.. **Directable, high-resolution simulation of fire on the gpu**. In: ACM SIGGRAPH 2009 PAP., SIGGRAPH '09, p. 41:1 – 41:8. ACM, 2009.

[14] FEDKIW, R.; STAM, J. ; JENSEN, H. W.. **Visual simulation of smoke**. In: PROC. 28TH ANNUAL CONF. COMPUT. GRAPH. INTERACT. TECH., SIGGRAPH '01, p. 15 – 22. ACM, 2001.

[15] MULLEN, P.; CRANE, K.; PAVLOV, D.; TONG, Y. ; DESBRUN, M.. **Energy-preserving integrators for fluid animation**. In: ACM SIGGRAPH 2009 PAP., SIGGRAPH '09, p. 38:1 – 38:8. ACM, 2009.

[16] FOSTER, N.; FEDKIW, R.. **Practical animation of liquids**. In: PROC. 28TH ANNUAL CONF. COMPUT. GRAPH. INTERACT. TECH., SIGGRAPH '01, p. 23 – 30. ACM, 2001.

[17] JOHANSEN, H.; COLELLA, P.. **A cartesian grid embedded boundary method for poisson's equation on irregular domains**. J. Comput. Phys., 147(1):60 – 85, 1998.

[18] ROBLE, D.; BIN ZAFAR, N. ; FALT, H.. **Cartesian grid fluid simulation with irregular boundary voxels**. In: ACM SIGGRAPH 2005 SKETCHES, SIGGRAPH '05. ACM, 2005.

[19] SHI, L.; YU, Y.. **Visual smoke simulation with adaptive octree refinement**. Technical report, University of Illinois, 2002.

[20] LOSASSO, F.; GIBOU, F. ; FEDKIW, R.. **Simulating water and smoke with an octree data structure**. In: ACM SIGGRAPH 2004 PAP., SIGGRAPH '04, p. 457 – 462. ACM, 2004.

[21] CHENTANEZ, N.; MÜLLER, M.. **Real-time eulerian water simulation using a restricted tall cell grid**. In: ACM SIGGRAPH 2011 PAP., SIGGRAPH '11, p. 82:1 – 82:10, New York, NY, USA, 2011. ACM.

[22] FELDMAN, B. E.; O'BRIEN, J. F. ; KLINGNER, B. M.. **Animating gases with hybrid meshes**. In: ACM SIGGRAPH 2005 PAP., SIGGRAPH '05, p. 904 – 909. ACM, 2005.

[23] KLINGNER, B. M.; FELDMAN, B. E.; CHENTANEZ, N. ; O'BRIEN, J. F.. **Fluid animation with dynamic meshes**. In: ACM SIGGRAPH 2006 PAP., SIGGRAPH '06, p. 820 – 825. ACM, 2006.

[24] CHENTANEZ, N.; FELDMAN, B. E.; LABELLE, F.; O'BRIEN, J. F. ; SHEWCHUK, J. R.. **Liquid simulation on lattice-based tetrahedral meshes**. In: PROC. 2007 ACM SIGGRAPH/EUROGRAPHICS SYMP. COMPUT. ANIMAT., SCA '07, p. 219 – 228. Eurographics Association, 2007.

[25] SHI, L.; YU, Y.. **Inviscid and incompressible fluid simulation on triangle meshes**. Comput. Animat. Virtual Worlds, 15(3-4):173 – 181, 2004.

[26] CATMULL, E.; CLARK, J.. **Recursively generated b-spline surfaces on arbitrary topological meshes**. Comput. Aided Des., 10(6):350 – 355, 1978.

[27] STAM, J.. **Exact evaluation of catmull-clark subdivision surfaces at arbitrary parameter values**. In: PROC. 25TH ANNUAL CONF. COMPUT. GRAPH. INTERACT. TECH., SIGGRAPH '98, p. 395 – 404, 1998.

[28] AZEVEDO, V. C.. **Efficient smoke simulation on curvilinear grids**. Master's thesis, UFRGS, 2012.

[29] AZEVEDO, V. C.; OLIVEIRA, M. M.. **Efficient smoke simulation on curvilinear grids**. Computer Graphics Forum, 32(7):235–244, 2013.

[30] BELL, N.; GARLAND, M.. **Cusp: Generic parallel algorithms for sparse matrix and graph computations**, 2012. Version 0.3.0.

[31] CHORIN, A. J.; MARSDEN, J. E.. **A mathematical introduction to fluid mechanics**. Springer-Verlag, 1993.

[32] BRIDSON, R.. **Fluid Simulation For Computer Graphics**. Ak Peters Series. A K Peters, 2008.

[33] ANDERSON, J.. **Computational Fluid Dynamics**. Computational Fluid Dynamics: The Basics with Applications. McGraw-Hill Education, 1995.

[34] DE OLIVEIRA FORTUNA, A.. **Técnicas computacionais para dinâminca dos fluidos: conceitos básicos e aplicações**. Edusp, 2000.

[35] HARLOW, F. H.; WELCH, J. E.. **Numerical calculation of time-dependent viscous incompressible flow of fluid with a free surface**. In: THE PHYSICS OF FLUIDS 8, p. 2182 – 2189, 1965.

[36] YAZICI, Y.. **Operator splitting methods for differential equations**. Master's thesis, Izmir Institute of Technology, 2010.

[37] HACKBUSCH, W.. **Multi-grid methods and applications**. Springer series in computational mathematics. Springer, 1985.

[38] CORPORATION, N.. **NVIDIA CUDA C Programming Guide Version 4.2**. NVIDIA Corporation, 2012.

[39] CRANE, K.; LLAMAS, I. ; TARIQ, S.. **Real-time simulation and rendering of 3d fluids**. In: Nguyuen, H., editor, GPU GEMS 3, chapter 30. Addison Wesley Professional, 2008.

[40] BISWAS, G.; BREUER, M. ; DURST, F.. **Backward-facing step flows for various expansion ratios at low and moderate reynolds numbers**. Journal of Fluids Engineering Transactions, 126:362–374, 2004.