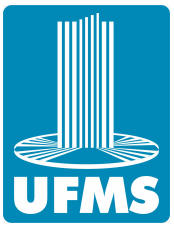
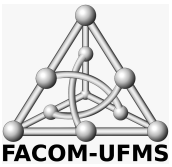


Universidade Federal de Mato Grosso do Sul
Faculdade de Computação



Bianca Namie Sakiyama

Simulação de fluidos com PIC usando RBF-FD e grades adaptativas balanceadas

Campo Grande – MS
2023

Bianca Namie Sakiyama

Simulação de fluidos com PIC usando RBF-FD e grades adaptativas balanceadas

Dissertação apresentada à Faculdade de Computação da Universidade Federal de Mato Grosso do Sul como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Paulo Aristarco Pagliosa

Campo Grande – MS
2023

Agradecimentos

Ao longo dos últimos anos de trabalho nesta pós-graduação muitos desafios foram enfrentados, entre eles o período de isolamento decorrente da pandemia e as adversidades derivadas de uma crise de saúde, e a conclusão deste trabalho só foi possível com a ajuda e apoio de familiares, amigos e colegas.

Agradeço aos meus pais, Nelson e Marilene, pelo apoio e paciência incondicionais ao longo de mais uma etapa na minha vida; à minha madrasta e padrasto, Selma e Sérgio, pelo incentivo e carinho desde que entraram na minha vida; ao meu irmão por sempre ser um excelente ouvinte e estendo esses agradecimentos ao restante da minha família que sempre esteve ao meu lado. Agradeço ainda meu parceiro, Cássio, que dividiu minhas angústias, ansiedades e também felicidades nesses últimos anos.

Minha eterna gratidão aos meus amigos e colegas, que em meio à correria do cotidiano ainda se fizeram presentes em cada passo dessa jornada. Em especial, agradeço ao meu amigo de longa data, Gabriel Sanches, que além da parceria em projetos que pré datam este, desempenhou um papel fundamental para conclusão desta pós-graduação e me acompanhou em todos estes anos, oferecendo conselhos e *insigths* que considero essenciais. Agradeço também ao Rafael Nakanishi e ao Filipe "Pirata" Nascimento, que não mediram esforços para sanar as dúvidas que surgiram durante o processo de estudo do método proposto pelo trabalho deles.

Não poderia deixar de agradecer também ao Prof. Paulo Pagliosa, que além de um ótimo orientador e professor, também foi um grande amigo e quase um segundo pai nesta caminhada, sempre a uma mensagem de distância para sanar todas as minhas dúvidas e discutir os próximos passos do trabalho. Com toda certeza, sem o seu apoio, paciência e orientação, este trabalho não seria concluído.

Por fim, agradeço aos demais professores da FACOM e também de outras instituições que me agraciaram com seu conhecimento e que continuam me motivando a aprofundar meus conhecimentos através dos estudos e pesquisa. Agradeço também a oportunidade de ter uma formação pública e gratuita de excelência.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001.

Resumo

Sakiyama, B. N. *Simulação de fluidos com PIC usando RBF-FD e grades adaptativas balanceadas*. Campo Grande, 2023. Dissertação (Mestrado) – Faculdade de Computação, Universidade Federal de Mato Grosso do Sul.

Em animações baseadas em física, a simulação de fluido por métodos híbridos, ou seja, empregando tanto uma discretização material quanto espacial, é normalmente baseada na subdivisão do espaço por uma grade regular. Essa abordagem, a depender da resolução adotada, pode gerar um grande número de células que não contém fluido e não precisam ser processadas, mas que ainda assim serão visitadas. Com o intuito de contornar esse problema, grades adaptativas foram introduzidas para eliminar células vazias. Nas grades regulares, o cálculo dos operadores diferenciais, que é necessário para a resolver a simulação, pode ser feito pelo método das diferenças finitas. O método aproxima o valor da derivada em um ponto com base em seus vizinhos — que devem estar alinhados ao ponto em relação aos eixos do domínio e a uma distância do ponto sendo avaliado — e essa vizinhança é chamada de estêncil. Como nas grades adaptativas os tamanhos das células podem ser diferentes, os estênceis gerados não são aptos para o uso do método das diferenças finitas como na grade regular. Uma alternativa para calcular os operadores diferenciais em uma grade adaptativa é o uso do método de diferenças finitas baseada em funções de base radial (RBF-FD), mas esse método é mais complexo e computacionalmente extensivo que o usado nas grades regulares. A fim de manter os benefícios de uma grade adaptativa e acelerar o cálculo dos operadores diferenciais com RBF-FD, este trabalho propõe o uso de grades adaptativas balanceadas, ou seja, grades em que a diferença de nível entre duas células vizinhas não seja maior que um. O objetivo é, uma vez identificado os estênceis gerados com tal condição, definir um dicionário cujas entradas contém funções que permitem o cálculo acelerado do RBF-FD para cada um desses estênceis possíveis.

Palavras-chave: *simulação de fluidos*, *RBF-FD*, *grades adaptativas balanceadas*.

Abstract

Sakiyama, B. N. *PIC fluid simulation using RBF-FD and adaptive graded grids.* Campo Grande, 2023. Thesis (Master's program) – Faculty of Computing, Federal University of Mato Grosso do Sul.

In physics based animations, hybrid fluid simulations — simulations that rely on both material and spatial discretization — are usually based on a regular subdivision of the domain by using a regular Cartesian grid. Depending on the adopted resolution, such an approach can create a large number of cells that do not contain fluid but will still be processed. To avoid this issue, adaptive grids have been introduced to reduce the number of non-fluid cells. With a Cartesian regular grid, the differential operators needed for a numerical fluid solver can be computed using the finite difference method (FDM). FDM gives an approximation of the partial derivatives of a quantity at a grid point based on the quantities of its neighboring grid points. The configuration formed by the target point and its neighbors is called a stencil. However, a stencil formed by the points of an adaptive grid cannot use FDM for computing the differential operators since such points do not form vectors aligned to the domain Cartesian grid. In this case, an alternative is to use the finite difference based on radial basis functions, which enable stencils in generic configurations. To maintain the benefits of using adaptive grids and accelerate the computing of the differential operator with RBF-FD, this thesis proposes a PIC-based fluid solver that employs a graded adaptive grid, that is, one in which the level difference between two neighboring cells is not greater than one. The proposed approach relies on maps whose entries store the RBF weights for a given differential operator and stencil. The map entries are indexed by a key that encodes the cell level and the relative position of the stencil target and its neighbors. The weights are computed only once and reused for the same stencils throughout the simulation.

Keywords: *fluid simulation*, *RBF-FD*, *graded adaptive grids.*

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

1.1 Motivação e justificativa

As animações baseadas em física são amplamente exploradas em uma variedade de aplicações, tanto nas ciências e engenharias quanto na indústria de entretenimento, como em efeitos especiais na cinematografia e jogos digitais. Essas aplicações se diferenciam na precisão de seus cálculos, precisão visual e eficiência computacional, uma vez que cada tipo de aplicação valoriza mais um aspecto diferente. As aplicações para ciências e engenharias, por exemplo, requerem cálculos precisos para testar produtos e teorias, e assim predizer, o mais exatamente possível, o comportamento do objeto de estudo na vida real em uma simulação de custo menor quando comparado com um teste físico. Já aplicações focadas em efeitos especiais têm mais ênfase na precisão visual, ou seja, o resultado produzido na simulação deve ser o mais real possível aos olhos humanos, tolerando-se um erro numérico maior do que em aplicações destinadas para ciências e engenharias. Por fim, as aplicações voltadas para jogos digitais requerem um processamento em tempo real, ou seja, a precisão numérica e visual são sacrificadas para se obter resultados mais rápidos e proporcionar a experiência fluida exigida pelos jogadores.

Independente da finalidade da aplicação, as simulações empregam modelos dos objetos sendo simulados, que podem ser geométrico — representação exata ou aproximada da forma e dimensões de um objeto — ou matemático — descrição do comportamento de um objeto, geralmente em termos de equações diferenciais parciais e suas condições iniciais e de contorno. Nesses casos, a solução de um modelo matemático envolve o emprego de um método numérico que fornece uma solução aproximada em uma versão discreta do objeto sendo simulado.

Um dos tipos de simulação mais empregada em animação baseada em física é a dinâmica de corpos rígidos. Um corpo rígido é um meio contínuo sólido no qual duas partículas quaisquer mantêm a mesma distância entre si, independentemente das forças atuantes no corpo. Geometricamente, um corpo rígido pode ser representado por uma conjunto de formas básicas

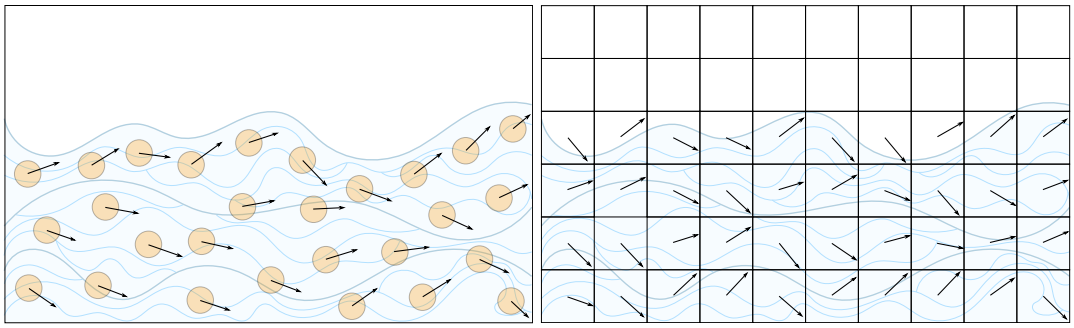
tais como blocos, cápsulas e cilindros, ou com superfície dada por uma malha de polígonos. Alternativamente, o modelo geométrico de um corpo rígido pode ser definido por um sistema de partículas, em que cada partícula concentra uma parte da massa do objeto. O comportamento de um corpo rígido é descrito matematicamente por uma equação de movimento derivada das leis de Newton, que relaciona grandezas tais como o estado do corpo (posição de seu centro de massa, orientação de seus eixos principais de inércia, velocidade linear e velocidade angular), sua inércia (massa e momento de inércia) e esforços externos (forças e torques) que nele atuam. Simular um corpo rígido significa determinar, a cada instante de tempo, qual o estado do corpo em função do seu estado corrente e dos esforços externos atuantes. Embora seja um dos casos mais simples de simulação em animação baseada em física, o problema é não trivial, pois a solução depende de restrições derivadas de colisões, com ou sem atrito, e junções entre corpos.

Outro tipo de simulação é a de sólidos deformáveis. Um corpo deformável pode ser elástico e/ou plástico. O material de um corpo perfeitamente elástico é tal que este recupera sua configuração inicial, ou de repouso, uma vez retiradas as forças externas causadoras das deformações. Em um corpo plástico, por outro lado, parte das deformações são permanentes, mesmo cessadas as ações externas. Comumente, um material é elástico somente até certo limite da razão entre força e deformação, a partir do qual o corpo plastifica. Geometricamente, um corpo deformável pode ser representado do mesmo modo que um corpo rígido, por exemplo, através de malhas de polígonos ou sistemas de partículas. O modelo matemático é definido por equações diferenciais derivadas da mecânica do contínuo [34], as quais relacionam deslocamentos, deformações, tensões e forças de superfície e de volume aplicadas ao corpo. Soluções para tais equações só podem ser obtidas, geralmente, através de métodos numéricos, sendo o mais amplamente empregado o método dos elementos finitos (MEF) [46]. Este baseia-se na discretização do domínio do corpo em uma nuvem de pontos, ou nós, conectados através de elementos chamados elementos finitos, por exemplo, tetraedros ou hexaedros no caso tridimensional. Como resultado de tal discretização, as equações diferenciais do modelo matemático são transformadas em um sistema de equações algébricas cuja solução fornece os deslocamentos (que simula as deformações) em cada ponto nodal. Os deslocamentos nos pontos interiores de um elemento finito podem ser obtidos por interpolação dos deslocamentos dos nós nos quais o elemento incide, com pesos definidos pela chamada função de forma do elemento.

Além de corpos rígidos e deformáveis, a simulação de fluidos, foco deste trabalho, tem sido bastante empregada em aplicações de engenharia, efeitos especiais e jogos digitais. Um fluido é um meio contínuo que oferece pouca resistência a efeitos cortantes, ou seja, é uma substância que se deforma quando submetida a forças aplicadas em sentidos iguais ou opostos e com intensidades diferentes, por menor que sejam tais forças, a exemplo da água ou fumaça. Assim como com corpos rígidos e deformáveis, o comportamento de um fluido é regido por equações diferenciais derivadas da teoria da mecânica do contínuo, e a geometria de um

fluido pode ser representada por malhas de polígonos ou por um sistema de partículas. Neste trabalho, consideram-se somente fluidos incompressíveis, isto é, que não perdem ou ganham volume ao serem submetidos à ação de uma força, e que têm seu comportamento governado pelas chamadas equações de Navier-Stokes, apresentadas no Capítulo 2.

Há três tipos de abordagem para a solução numérica das equações que modelam o comportamento de um fluido, sendo elas: Lagrangeana, Euleriana e híbrida. Na abordagem Lagrangeana, ou material, o fluido é dividido em elementos de volume e o fluxo destes são observados. Uma forma de fazer esta subdivisão é utilizando um sistema de partículas, em que cada partícula guarda propriedades como posição, velocidade, pressão e força externa. Nessa abordagem, simular o fluido significa, a cada passo de tempo, determinar a posição e a velocidade de cada partícula do sistema, em termos das forças que atuam em cada uma. Um exemplo clássico de método baseado em partículas é o *smoothed particle hydrodynamics* (SPH) [31, 25]. A abordagem Euleriana, ou espacial, emprega um método de subdivisão do espaço de simulação em células que guardam as informações, já mencionadas, do fluido passando por elas e a cada passo de tempo essas informações são atualizadas. O trabalho *Stable Fluids* de Stam [41] é um exemplo de método que utiliza a abordagem Euleriana. A Figura 1.1 ilustra duas simulações, uma baseada em grade e outra em partículas.



(a) (b)

Figura 1.1: Exemplos de uma simulação Lagrangeana e Euleriana. (b) Uma simulação baseada em partículas. (a) Uma simulação baseada em grade. (Adaptado de Liu et al. [32].)

Nas abordagens híbridas, principal foco deste trabalho, busca-se mesclar a abordagem Lagrangeana e a abordagem Euleriana para amenizar os pontos negativos de cada uma, pontos estes que são comentados no Capítulo 2. Entre os métodos híbridos, há propostas como o *particle level set* — que utiliza um campo de distâncias com sinal para definir a geometria do fluido [13, 27] — e o *particle-in-cell* (PIC) [23], que segue o fluido de forma Lagrangeana, ou seja, utilizando partículas e realizando alguns cálculos nas partículas e outros na grade. Outro método também híbrido, que é uma extensão do PIC, é o *fluid-implicit-particle* (FLIP) [7, 45, 14, 38], que com poucas mudanças busca diminuir a introdução de viscosidade artificial, ou seja, diminuir o aumento da viscosidade do fluido ao longo da simulação que aparece no PIC. Os modelos híbridos são abordados no Capítulo 2.

Na resolução dos métodos Eulerianos e híbridos, é frequentemente utilizada uma grade regular para subdividir o espaço, ou seja, o domínio espacial que pode ser ocupado por

um fluido é dividido em células quadrilaterais iguais — no caso bidimensional — ou em hexaedrais iguais — no caso tridimensional. Um problema apresentado por esta subdivisão é uma possível distribuição irregular do volume do fluido através da grade, uma vez que, em decorrência do fluxo do fluido ao longo da simulação, pode haver muito espaço com pouco ou nenhum fluido e uma maior concentração de fluido em espaços menores. Por este motivo, quanto mais refinada a grade, maior o número de células que podem não conter fluido mas que, ainda assim, precisam ser processadas a cada passo de tempo da simulação, enquanto que um maior número de cálculos é requerido naquelas células que interceptam a superfície livre ou que contêm maiores porções de fluido. Para contornar o problema criado pelas grades regulares, pode-se empregar as grades adaptativas, as quais subdividem o espaço de acordo com a concentração do material sendo simulado, ou seja, com mais células onde houver maior concentração do fluido e menos onde não houver, sendo a árvore quaternária, ou *quadtree* — em ambientes bidimensionais — e a árvore octária, ou *octree* — em ambientes tridimensionais — as estruturas mais comuns.

Parte essencial dos métodos híbridos (e Eulerianos) é a computação de operadores diferenciais sobre propriedades do fluido, sendo eles o gradiente, divergente e Laplaciano, todos definidos em termos de derivadas parciais de primeira e segunda ordem. Para resolver estes operadores diferenciais em espaços discretizados por grades regulares, comumente utiliza-se o método das diferenças finitas [30]. Para tal, é necessário se considerar a célula contendo o ponto no qual se deseja avaliar o operador, que pode ser o centro da célula ou o centro de alguma de suas faces, bem como pontos sobre as células vizinhas. A configuração de tais pontos define o que se costuma chamar de estêncil, conforme ilustrado na Figura 1.2. Em uma grade regular, o estêncil associado ao centroide de uma célula possui sempre a configuração mostrada na Figura 1.2. Como consequência, pode-se obter uma fórmula fechada para o cálculo de um operador diferencial em cada célula (mais especificamente em seu centroide, nesse caso), expressa em termos dos valores da função sobre a qual se deseja aplicar o operador, avaliadas em todos os pontos do estêncil. Tanto os operadores diferenciais quanto o método das diferenças finitas são abordados no Capítulo 2.

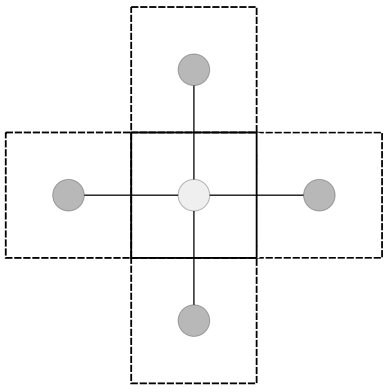
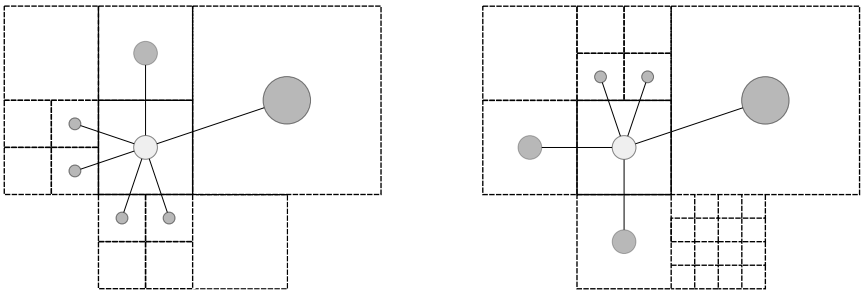


Figura 1.2: Estêncil em uma grade regular.

Em uma grade adaptativa, uma célula pode ter como vizinhas células com diferentes

níveis de profundidade na árvore. Com isso, pode-se ter, para cada caso de vizinhança, uma configuração de estêncil diferente, como ilustrado na Figura 1.3. Nesses casos, não se tem uma expressão pré-definida para o cálculo de cada operador diferencial, o qual pode ser efetuado com o emprego de diferenças finitas baseadas em funções de base radial (RBF-FD) [16, 17, 5, 12], introduzida na Seção 2.3.



(a) (b)

Figura 1.3: Exemplos de estênceis em uma grade adaptativa. (a) Grade adaptativa balanceada, ou seja, em que os células vizinhas possuem no máximo um nível de diferença. (b) Grade adaptativa desbalanceada, ou seja, em que não há limite para a diferença de níveis entre células vizinhas.

1.2 Objetivos

Em 2020, o trabalho de Nakanishi et al. [37] propôs um método híbrido de simulação de fluidos que utiliza grades adaptativas, justamente para contornar as desvantagens apresentadas pelas grades regulares, e RBF-FD para o cálculo dos operadores diferenciais e interpolações em estênceis não regulares. Como apontado no artigo, os cálculos de RBF-FD, que devem ser feitos para cada configuração diferente de estêncil, são intensivos computacionalmente, e para contornar esse problema, este trabalho propõe o uso de uma grade adaptativa balanceada, ou seja, que limita a diferença de níveis entre células vizinhas da árvore. Essa limitação na diferença entre os níveis limita também a quantidade de configurações de estênceis distintas e espera-se que a quantidade fixa de configurações possibilite a implementação de uma estrutura que guarda os pesos já calculados, acelerando os cálculos de RBF-FD.

A partir do exposto anteriormente, este trabalho visa propor um método híbrido de simulação de fluidos que utilize RBF-FD para o cálculo dos operadores diferenciais em uma grade adaptativa balanceada, explicada com mais detalhes no Capítulo 3. Os objetivos específicos traçados são:

O1 Propor um método híbrido de simulação de fluidos baseado no PIC que utilize grades adaptativas balanceadas e RBF-FD com uma estrutura de otimização.

O2 Implementar um solucionador híbrido que utilize o método proposto.

O3 Estudar a eficiência e robustez do método proposto, comparando simulações geradas pelo solucionador que utiliza uma estrutura de otimização e um que não utiliza.

1.3 Organização do texto

O restante do documento é organizado como segue. O Capítulo 2 apresenta conceitos e trabalhos relacionados ao trabalho proposto, detalhando os métodos de solução de simulações de fluidos — baseados em partículas, grades e híbridos — e suas diferenças, além de abordar as estruturas adaptativas e RBF-FD presentes no trabalho de Nakanishi et al. [37] . O Capítulo 3 descreve o solucionador que foi desenvolvido, detalhando seu *pipeline*, as classes que modelam a grade adaptativa balanceada, operadores diferenciais e estênceis, ademais, os dicionários testados para otimizar o cálculo dos operadores diferenciais também são comentados. O Capítulo 4 apresenta os resultados obtidos pelo solucionador descrito no Capítulo 3, comparando seu desempenho com e sem a estrutura de otimização. Por fim, o Capítulo 5 conclui este documento repassando pelos principais pontos de cada capítulo, além de propor trabalhos que continuem a desenvolver o simulador aqui proposto.

Capítulo 2

Conceitos e trabalhos relacionados

2.1 Considerações iniciais

Este capítulo detalha os conceitos teóricos utilizados neste trabalho e resume os principais trabalhos relacionados descritos na literatura. São apresentados com mais detalhes os já mencionados métodos de simulação de fluidos baseada em partículas, em grades e métodos híbridos, com foco no último tipo, além da base teórica que os suporta — como as equações de Navier-Stokes e operadores diferenciais. Ademais, as grades adaptativas e suas diferenças em relação às grades regulares também são exploradas, juntamente com as funções RBF-FD utilizadas para resolver os operadores diferenciais nas grades adaptativas.

2.2 Simulações de fluidos

Um fluido, como já definido antes, é um meio contínuo que oferece pouca resistência a efeitos cortantes, e podem ser divididos em líquidos e gases. O líquido, quando em repouso, apresenta uma superfície estacionária não determinada pelo recipiente que o contém e o gás tem como uma propriedade marcante a sua expansão livre quando não contido por um obstáculo. Além disso, outra forma de classificar os fluidos é de acordo com sua compressibilidade, dividindo-os em compressíveis ou incompressíveis. Os compressíveis são os que podem sofrer redução ou expansão de volume quando sob influência de uma força externa e os incompressíveis são exatamente o contrário, ou seja, é um fluido que resiste à compressão independentemente da força aplicada.

As propriedades dos fluidos podem ser divididas em três grupos: cinemáticas, termodinâmicas e outras propriedades. Pode-se destacar especificamente a velocidade, densidade, pressão e viscosidade. A velocidade descreve o movimento de um corpo contínuo ao longo do tempo, em outras palavras, é a grandeza que relaciona uma quantia de tempo ao espaço percorrido por uma massa. A pressão é a grandeza que mede a razão entre uma ou mais

forças e uma determinada área e a densidade é a razão entre a massa de um fluido pelo seu volume, e pode sofrer alterações se houver alterações na velocidade ou na pressão, se o fluido for compressível. Por fim, a viscosidade de um fluido mede a resistência deste a efeitos cortantes, sendo assim, fluidos com maior viscosidade se assemelham a uma pasta grossa ou a um xarope.

2.2.1 Equações de Navier-Stokes

O comportamento de fluidos incompressíveis é modelado pelas chamadas equações de Navier-Stokes [34]:

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} + \frac{\nabla p}{\rho} = \mathbf{g} + \nu \nabla \cdot \nabla \mathbf{u}, \tag{2.1}$$

$$\nabla \cdot \mathbf{u} = 0. \tag{2.2}$$

A Equação (2.1) é a equação de momento, isto é, a Segunda lei de Newton reescrita para fluidos incompressíveis, a qual prescreve que a diferença entre as forças externas e internas atuando sobre o fluido é igual ao produto entre a massa e a aceleração. Nessa equação, bem como na Equação (2.2), $\mathbf{u}$ representa a velocidade e t representa o tempo, e $\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u}$ é a derivada material, ou seja, a taxa de variação de uma grandeza ao longo do tempo para uma porção de material que se move a uma velocidade $\mathbf{u}$. O termo $\frac{1}{\rho}\nabla p$, em que ∇p representa o gradiente da pressão e ρ é a densidade, é a razão que quantifica a força da pressão de uma porção do fluido. No lado direito, $\mathbf{g}$ é a aceleração da gravidade e $\nu \nabla \cdot \nabla \mathbf{u}$ — em que ν é a viscosidade cinemática — representa a força viscosa, que é a força do fluido que se opõe ao movimento e tenta fazer uma porção do fluido assumir uma velocidade média das porções vizinhas.

A Equação (2.2) é a condição de incompressibilidade, que garante que o fluido não tem alterações no seu volume, condição essa definida simplesmente pela nulidade do divergente da velocidade. A unicidade da solução das equações de Navier-Stokes depende das condições de contorno de Dirichlet (as quais restringem os valores de uma função, por exemplo, a velocidade, em pontos o contorno do fluido) e de Neumann (que restringem os valores da derivada direcional de uma função, por exemplo, fluxo, na direção das normais do contorno). Maiores detalhes podem ser obtidos em [8, 28].

Neste trabalho, a simulação será restrita a fluidos incompressíveis invíscidos, ou seja, a viscosidade será ignorada. Logo, a Equação (2.1), com os termos rearranjados, torna-se:

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = \mathbf{g} - \frac{\nabla p}{\rho}. \tag{2.3}$$

Ao acompanhar o fluido ao longo do tempo, sua velocidade modifica-se de acordo com as forças externas ($\mathbf{g}$) e da pressão ($-\nabla p/\rho$). Tanto a gravidade quanto a densidade são constantes e como a densidade e a pressão estão relacionadas, resta obter a pressão que mantém a

densidade constante. Para tal, considera-se a aceleração causada pela pressão $a_p = -\nabla p/\rho$ e o cálculo da velocidade em um novo passo de tempo δt como $\mathbf{u}^{n+1} = \mathbf{u}^n - a\delta t$ — em que a é uma aceleração —, dos quais tem-se:

$$\mathbf{u}^{n+1} = \mathbf{u}^n - \delta t \frac{\nabla p}{\rho}. \tag{2.4}$$

Como a condição para manter a densidade constante, ou seja, o fluido incompressível, é a Equação (2.2), aplica-se o divergente em ambos os lados da Equação (2.4):

$$\nabla \cdot \mathbf{u}^{n+1} = \nabla \cdot \mathbf{u}^n - \delta t \frac{\nabla^2 p}{\rho}. \tag{2.5}$$

Em seguida substitui-se $\nabla \cdot \mathbf{u}^{n+1} = 0$ e isola-se $\nabla^2 p$:

$$\nabla^2 p = \rho \frac{\nabla \mathbf{u}^n}{\delta t}. \tag{2.6}$$

A Equação (2.6), da qual obtém-se o valor da pressão, é uma equação de Poisson, por vezes chamado de equação de Poisson para a pressão ou apenas de equação de Poisson, em contextos de simulação de fluidos.

Nas equações de Navier-Stokes e na Equação (2.6), observa-se a presença do símbolo ∇, definido como:

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

Usando-se ∇, o gradiente de um campo escalar $f(\mathbf{x})$, o qual mede a taxa e direção da variação do campo escalar, é dado por:

$$\nabla f(\mathbf{x}) = \left(\frac{\partial f(\mathbf{x})}{\partial x}, \frac{\partial f(\mathbf{x})}{\partial y}, \frac{\partial f(\mathbf{x})}{\partial z} \right). \tag{2.7}$$

Já o divergente, que opera sobre um campo vetorial $\mathbf{F}(\mathbf{x})$, mede se o fluxo do campo vetorial converge ou diverge dos pontos no campo e é dado por:

$$\nabla \cdot \mathbf{F}(\mathbf{x}) = \frac{\partial F_x}{\partial x} + \frac{\partial F_y}{\partial y} + \frac{\partial F_z}{\partial z}, \tag{2.8}$$

em que F_x, F_y e F_z representam $\mathbf{F}(\mathbf{x})$ na coordenada x, y e z respectivamente. Desses dois operadores, pode-se obter o Laplaciano, responsável por calcular o quão côncavas são as áreas do campo, escrito como:

$$\nabla^2 f(\mathbf{x}) = \frac{\partial^2 f(\mathbf{x})}{\partial x^2} + \frac{\partial^2 f(\mathbf{x})}{\partial y^2} + \frac{\partial^2 f(\mathbf{x})}{\partial z^2}. \tag{2.9}$$

2.2.2 Simulações baseadas em partículas

Os métodos Lagrangeanos, como o SPH [36] e o *Predictive-Corrective Incompressible SPH* (PCISPH) [40], que é uma melhoria do SPH, discretizam o volume do fluido em partículas. Em ambos os modelos, dados como a posição, velocidade e força são armazenados nas partículas e a cada passo de tempo de simulação atualiza-se o estado de cada partícula, ou seja, posição e velocidade, em função das forças e do tempo decorrido na simulação. Para atualizar o estado das partículas e avançar na simulação o solucionador acumula as forças externas — como a gravidade e forças de arrasto — e as forças do gradiente da pressão e da viscosidade; em seguida, um passo de integração temporal, que utiliza o passo de tempo da simulação, as forças acumuladas e a massa das partículas, é executado para obter as novas velocidades e posições; por fim, as possíveis colisões das partículas são tratadas.

Ao atualizar as posições das partículas no final de cada passo de tempo de simulação, a advecção — transporte de matéria e grandezas físicas ao longo do fluido — é resolvida naturalmente, uma vez que a própria partícula armazena as grandezas físicas. Isso está entre os pontos positivos do método baseado em partículas, já que as grandezas físicas são conservadas durante a atualização das posições das partículas. Além disso, como as partículas não possuem posições fixas, o método se adapta com facilidade a qualquer formato de superfície, porém, elas também podem se amontoar em algumas áreas, introduzindo erros na simulação. Mais um ponto fraco do SPH é o tamanho limitado que pode ser assumido para o passo de tempo, derivado da equação de estado que correlaciona a densidade e a pressão usada no cálculo do gradiente da pressão. Mais informações sobre sistemas de partículas e SPH podem ser encontradas no trabalho de Xi et al. [44].

2.2.3 Simulações baseadas em grades regulares

Os métodos Eulerianos ou baseados em subdivisão espacial [24, 41] discretizam o espaço, ou domínio da simulação, em sub-regiões às quais são atribuídas propriedades do fluido. O tipo mais comum de estrutura de subdivisão é a grade regular, sendo as sub-regiões células da grade. Nesse caso, as propriedades do fluido podem ser associadas aos centros ou vértices das células, ou ainda, aos centros das faces das células. Nas grades regulares, os pontos equidistantes um dos outros nas direções de cada eixo que define o domínio torna possível o uso do método de diferenças finitas para o cálculo dos operadores diferenciais, em que as derivadas parciais são aproximadas para a razão entre a diferença entre as grandezas avaliadas nos pontos e a distância entre os pontos. Mais detalhes sobre a estrutura de dados das grades e o cálculo dos operadores diferenciais sobre elas serão descritos na Seção 2.2.4.

Assim como nos métodos baseados em partículas, as simulações baseadas em grades resolvem as equações de Navier-Stokes, descrita no Seção 2.2.1, sendo assim, é necessário calcular o passo da pressão, viscosidade e gravidade. Além destes, o passo da advecção deve ser calculado explicitamente, uma vez que a estrutura da grade não resolve a transferência de

matéria ao longo do fluxo naturalmente, como acontece na discretização utilizando partículas. Uma descrição mais detalhada de métodos Eulerianos foi feita por Tan e Yang [42].

2.2.4 Métodos híbridos

Métodos híbridos são atraentes por se aproveitarem dos pontos positivos tanto dos modelos Lagrangeanos quanto Eulerianos e por amenizarem seus pontos negativos — como o passo de tempo de tamanho limitado como no SPH ou a dissipação numérica no passo da advecção em simulações baseadas em grades. O PIC [23] e o FLIP [7, 45] realizam transferências da velocidade entre as partículas e pontos na grade para se aproveitar do passo da advecção resolvido naturalmente nos métodos baseados em partículas, assim como do cálculo dos operadores diferenciais bem definidos pela estrutura da grade. O método segue o fluido de maneira Lagrangeana e as partículas neste modelo são utilizadas para marcar em quais células há fluido, além de guardar também a velocidade.

De modo geral, a cada passo de tempo, tanto o PIC quanto o FLIP resolve os seguintes passos, explicados em mais detalhes a seguir:

P1 Transferir a velocidade de cada partícula para a grade;

P2 Computar gravidade, pressão e incompressibilidade na grade;

P3 Transferir a velocidade da grade para as partículas;

P4 Atualizar a posição das partículas.

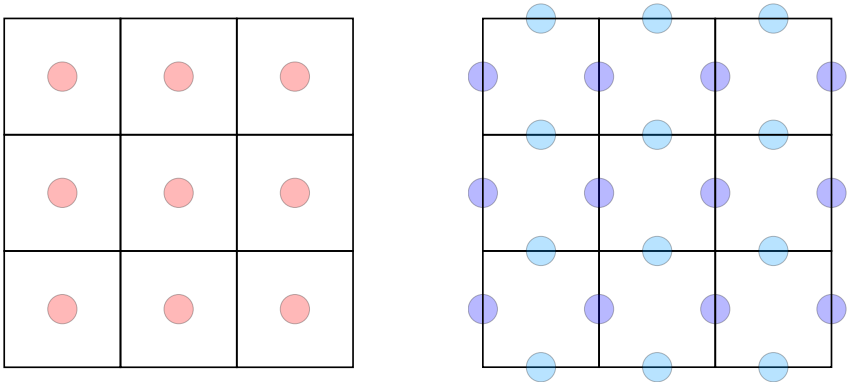
Estes métodos inspiram outros trabalhos [6, 43, 20], como é o caso do *Affine Particle-In-Cell* (APIC) proposto por Jiang et al. [26] que busca manter a estabilidade do PIC sem sofrer da dissipação numérica, discutida ao longo deste capítulo, ao utilizar uma matriz para representar o fluxo do fluido nas partículas. Os métodos PIC e FLIP são descritos em mais detalhes a seguir.

Partículas e grades

As partículas no PIC são utilizadas no passo da advecção e para marcar em que células há fluido. Logo, a grandeza que será armazenada nas partículas é a velocidade. Além disso, como o passo **P2** é feito na grade, não há a necessidade de definir o cálculo dos operadores diferencias nas partículas e nem tratar a colisão entre elas.

As grades podem ser divididas quanto ao tipo de dado que guardam — valores escalares ou vetoriais — e quanto ao local de armazenamento — nos centros das faces, das células ou nos vértices — e mais de um tipo de grade pode ser utilizado em um mesmo solucionador, com as especificidades implementadas em apenas uma estrutura de dados ou em estruturas separadas. Os métodos híbridos e baseados em grades estudados avaliam a pressão em grades

centradas nas células e operam sobre a velocidade em grades conhecidas como *marker-and-cell* (MAC) [24], que são centradas nas faces. As grades centradas nas células são grades colocadas (do inglês *collocated*), ou seja, as componentes x, y e z do vetor são guardas no mesmo local (no centro da célula); já as grades centradas nas faces são grades deslocadas (do inglês *staggered*), com cada componente do vetor sendo guardada em uma face diferente. Ambas as grades são ilustradas na Figura 2.1.



(a) (b)

Figura 2.1: Grades com propriedades centradas nas células e nas faces. (a) Grades centradas na célula armazenam a pressão, representada pelos círculos vermelhos. (b) Grades centradas na face das células armazenam a velocidade, que é representada por componente, sendo (para o caso bidimensional) os círculos azuis escuro a componente x e azuis claro a componente y.

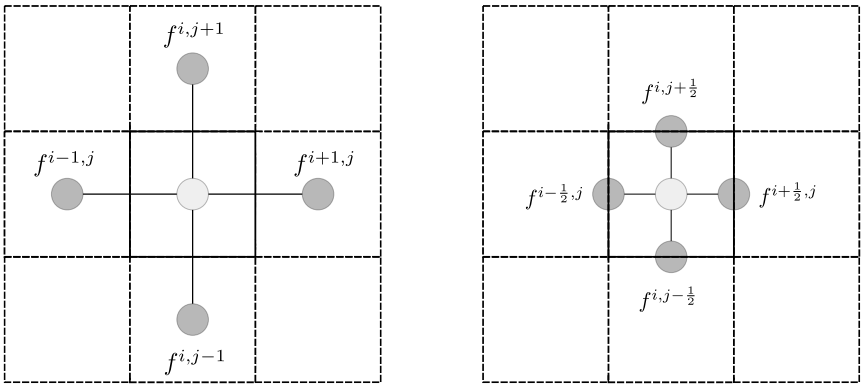
Dada a discretização dos campos sobre os quais os cálculos serão realizados, pode-se obter os estênceis, que são as configurações formadas entre um ponto da célula sendo avaliada e os pontos em suas células vizinhas, pontos estes que armazenam as grandezas sobre as quais os cálculos serão realizados. O cômputo dos operadores diferenciais depende das derivadas parciais e uma abordagem para o seu cálculo é o uso do método das diferenças finitas. Supondo que há uma função $f(x, y, z)$ e que se tem o valor desta função em dois pontos da grade consecutivos e também o espaçamento entre as células da grade, o cálculo da derivada parcial de f em função de x pode ser dada por:

$$\frac{\partial f}{\partial x} \approx \frac{f^{i+1,j,k} - f^{i-1,j,k}}{2\Delta x}, \tag{2.10}$$

em que i, j e k representam os índices dos pontos da grade e Δx representa o espaçamento entre as células, que em uma grade regular é sempre igual na mesma direção. A Equação (2.10) é conhecida como diferença finita centrada e é utilizada quando se opera sobre a pressão, grandeza armazenada nos centros das células. Além desta, há também a diferença regressiva e a progressiva. No caso da grandeza estar armazenada no centro da face e não no centro da célula, ou seja, operações sobre a velocidade, o cálculo é ajustado para:

$$\frac{\partial f}{\partial x} \approx \frac{f^{i+\frac{1}{2},j,k} - f^{i-\frac{1}{2},j,k}}{\Delta x}. \tag{2.11}$$

Ambos os casos são ilustrados na Figura 2.2.



(a) (b)

Figura 2.2: Exemplos da diferença finita em grade regulares, operando sobre os centros das células e centros das faces. (a) Pontos utilizados no método das diferenças finitas na grade centrada nas células. (b) Pontos utilizados no método das diferenças finitas na grade centrada na face das células. (Adaptado de Kim [28].)

Com um método para calcular as derivadas parciais, o cálculo dos operadores podem ser feitos de forma direta, apenas aplicando as diferenças finitas (nos exemplos a seguir, para grandezas centradas no centro das células) nas equações expostas na Seção 2.2.1. Sendo assim, para o gradiente de um escalar temos:

$$\nabla f(\mathbf{x}) = \left(\frac{f^{i+1,j,k} - f^{i-1,j,k}}{2\Delta x}, \frac{f^{i,j+1,k} - f^{i,j-1,k}}{2\Delta y}, \frac{f^{i,j,k+1} - f^{i,j,k-1}}{2\Delta z} \right). \tag{2.12}$$

O divergente de um vetor é dado por:

$$\nabla \cdot \mathbf{F}(\mathbf{x}) = \frac{F_x^{i+1,j,k} - F_x^{i-1,j,k}}{2\Delta x} + \frac{F_y^{i,j+1,k} - F_y^{i,j-1,k}}{2\Delta y} + \frac{F_z^{i,j,k+1} - F_z^{i,j,k-1}}{2\Delta z}, \tag{2.13}$$

e o Laplaciano de um escalar é dado por:

$$\begin{aligned} \nabla^2 f(\mathbf{x}) = & \frac{f^{i+1,j,k} - 2f^{i,j,k} + f^{i-1,j,k}}{\Delta x^2} + \\ & \frac{f^{i,j+1,k} - 2f^{i,j,k} + f^{i,j-1,k}}{\Delta y^2} + \\ & \frac{f^{i,j,k+1} - 2f^{i,j,k} + f^{i,j,k-1}}{\Delta z^2}. \end{aligned} \tag{2.14}$$

Usando o mesmo processo, podem ser encontradas as expressões para grandezas localizadas no centro da face.

Particle-in-Cell

A transferência de velocidade de uma partícula para a grade regular é realizada distribuindo a velocidade para os 4 (caso 2D) ou 8 (caso 3D) pontos — centro das faces — mais próximos na grade, sendo essa distribuição separada por componente do vetor velocidade, uma vez que a velocidade é armazenada em uma grade alternada. A distribuição é feita utilizando pesos que podem ser obtidos quando a interpolação bilinear, no caso 2D, ou trilinear,

no caso 3D, é realizada e uma distribuição é efetuada neste passo. Ou seja, a distribuição é feita, no caso bidimensional, com base nas quatro áreas formadas quando é traçado uma linha horizontal e outra vertical, ambas paralelas aos seus respectivos eixos da grade, sobre o ponto a ser interpolado, como demonstrado na Figura 2.3 por A, B, C e D; no caso tridimensional a distribuição é baseada nos oito volumes obtidos da mesma subdivisão da célula a partir do ponto a ser distribuído.

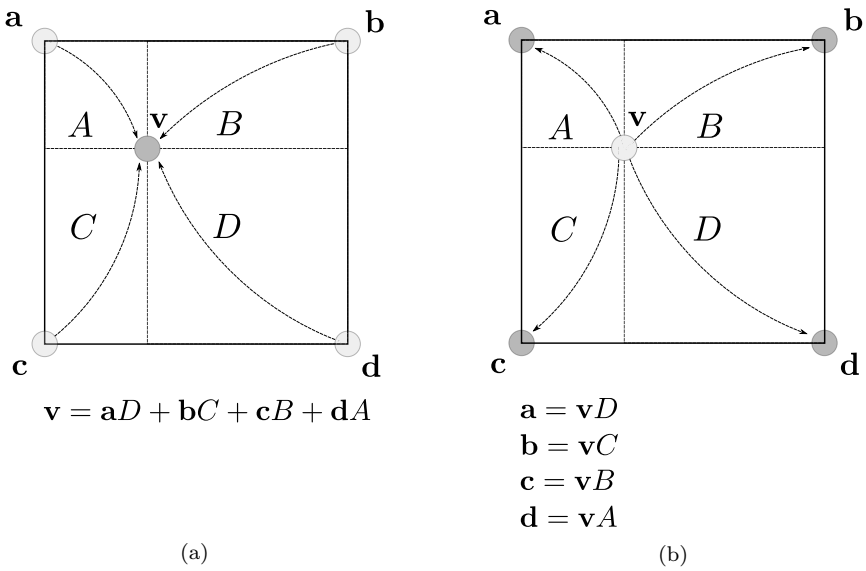
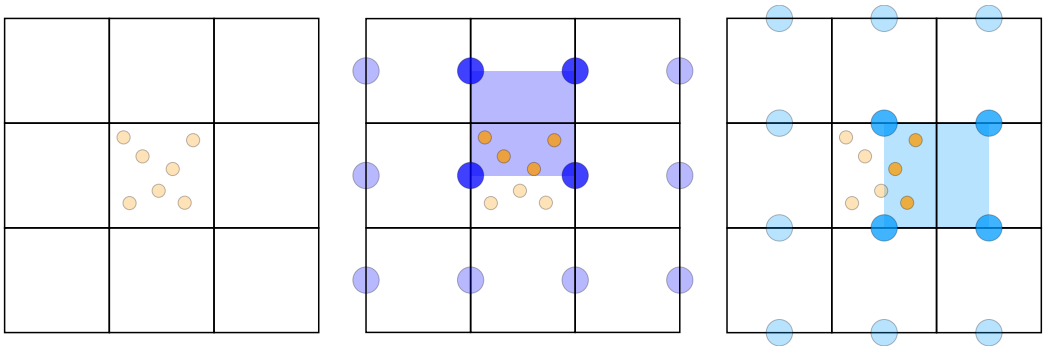


Figura 2.3: Ilustração da interpolação e distribuição da velocidade entre pontos. (a) Pontos **a**, **b**, **c** e **d** sendo interpolado para o ponto **v** utilizando os pesos A, B, C e D. (b) Ponto **v** sendo distribuído para os pontos **a**, **b**, **c** e **d**, no processo inverso de (a). (Adaptado de Kim [28].)

Para efetivamente distribuir a velocidade das partículas para a grade, a cada passo da simulação, a velocidade na grade e os pesos devem ser zerados. Em seguida, para cada partícula, acumulam-se os pesos e velocidade para os pontos mais próximos da grade e após a soma destes atributos para todas as partículas, a velocidade é normalizada e os pontos da grade que guardam velocidade são sinalizados, isto é, mantém-se uma marcação dos pontos em que há a presença do fluido na simulação. A distribuição da velocidade por componente é ilustrada na Figura 2.4.

A gravidade e pressão são calculadas como na abordagem Euleriana, com cada etapa modificando o campo da velocidade. A gravidade é computada apenas somando seus termos diferentes de zero ao campo da velocidade, sendo assim, a nova velocidade é dada por $\mathbf{u}^{n+1} = \mathbf{u}^n + \delta t \mathbf{g}$. A Equação (2.6), quando aplicada à grade, resulta em um sistema linear que é resolvido para obter o campo da pressão. Com o novo valor da pressão, é possível calcular o gradiente da pressão e computar a velocidade final, dada por $\mathbf{u}^{n+1} = \mathbf{u}^n - \delta t \rho^{-1} \nabla p$, que será transferida da grade para as partículas, para o cálculo do passo da advecção.

A transferência da velocidade das células da grade para as partículas é feita por componente através de uma interpolação bilinear ou trilinear, a depender das dimensões da simulação. Logo, de forma similar à distribuição, as partículas irão receber as velocidades dos pontos que formam o quadrado (simulação 2D) ou cubo (simulação 3D) em que estão



(a) (b) (c)

Figura 2.4: Ilustração da transferência da velocidade entre partículas e a grade em uma simulação bidimensional. (a) Posição inicial das partículas. (b) A componente x das partículas destacadas são distribuídas para os pontos destacados em azul escuro (a componente x dos pontos da grade são interpolados para as partículas destacadas). (c) A componente y é distribuída (ou interpolada).

contidas por componente da velocidade. Com a velocidade atualizada, é necessário atualizar a posição de cada partícula. Para isso a regra do ponto médio é utilizada: ao invés de se utilizar apenas uma interpolação linear para encontrar a posição final, utiliza-se uma interpolação com uma velocidade intermediária — calculada em termos da velocidade em pontos das células vizinhas daquela contendo a partícula — e depois com a velocidade final transferida da grade. O método PIC, apesar de apresentar vantagens quando comparado com um modelo exclusivamente Euleriano, como a conservação de massa, ainda tem uma desvantagem: a adição de dissipação numérica, e por consequência viscosidade artificial, decorrida da transferência da velocidade das células da grade para as partículas e vice-versa. Para evitar estes erros de aproximação, o método *fluid-implicit-particle* (FLIP) pode ser utilizado.

Fluid-Implicit-Particle

O FLIP é uma extensão do PIC que altera os passos da transferência da velocidade das partículas para as células da grade e vice-e-versa. Como a interpolação realizada para transferir a velocidade dos pontos da grade para as partículas adiciona erro numérico à simulação, uma alternativa é interpolar a diferença da velocidade inicial para a final e somar esta diferença na velocidade inicial presente nas partículas. Uma vez que a diferença entre as velocidades é menor do que a velocidade final, a interpolação gera um erro menor. Esta mudança em relação ao PIC é o suficiente para tornar o FLIP uma alternativa mais atrativa para a simulação de fluidos menos viscosos, como a água.

2.3 Grades adaptativas e RBF-FD

Como comentado no Capítulo 1, os solucionadores implementados com base em grades regulares ou partículas podem realizar processamento desnecessário que afeta o desempenho da aplicação, o que pode tornar o uso de métodos adaptativos atraente. Dentre os métodos

propostos, há os que não utilizam estruturas de divisão regular do espaço e optam por outras alternativas, como é o caso dos trabalhos de Batty et al. [4] e Ando et al. [2] que subdividem o espaço do domínio em tetraedros. Há também adaptações da grade regular, como é o caso do trabalho de Chentanez e Müller [9], que transforma células consecutivas na direção vertical — presentes na parte inferior do fluido sendo simulado — em uma única célula alta (do inglês *tall cell*) para diminuir a quantidade de cálculos. Entre as propostas adaptativas, há aquelas que apresentam a adaptatividade em outros aspectos que não na grade usada para discretizar o domínio da simulação, como é o caso do solucionador proposto por Edwards e Bridson [11], que utiliza uma malha de triângulos adaptativa para determinar a superfície e o método de Galerkin descontínuo como alternativa para o método dos elementos finitos. Ainda, no solucionador proposto por Ferstl e outros [15], uma adaptação do FLIP usa partículas em uma faixa menor próxima à superfície do líquido para diminuir o número de partículas, enquanto simula o restante do volume utilizando uma grade.

O foco neste trabalho são as grades adaptativas que realizam a divisão regular do espaço: *quadtree*, no caso bidimensional, e *octree*, no caso tridimensional. Em tais grades, uma determinada célula pode ter como vizinhas células que estão em níveis distintos da árvore. Os diferentes tamanhos das células são decorrentes de um processo de subdivisão que leva em conta a distribuição de fluido no espaço, de forma que não haja uma discrepância na quantidade de fluido em cada célula. Sendo assim, quanto mais células menores estão próximas, maior é a concentração de fluido no local. O mesmo ocorre na superfície livre do fluido, isto é, na interface entre o fluido e o ar.

A cada passo de tempo da simulação híbrida, a grade pode ser alterada para acomodar o novo estado do fluido e pode haver a necessidade de ajustar o número de partículas por célula (processo conhecido como *particle reseeding*). Além disso, a fim de computar os termos das equações que modelam o comportamento do fluido em cada passo de tempo da simulação, torna-se necessário, conforme introduzido na Seção 2.2, o cálculo de operadores diferenciais sobre as várias propriedades do fluido. Nos métodos híbridos, tais cálculos usualmente são baseados no método numérico de diferenças finitas. Em grades adaptativas, os estênceis de diferenças finitas não são fixos como na grade regular, como consequência, não é possível utilizar o método das diferenças finitas para o cálculo de tais operadores, os quais devem ser avaliados caso a caso.

O uso de grades adaptativas tem sido objeto de pesquisa no campo de simulação de fluidos como vem sendo demonstrado pelos trabalhos encontrados na literatura. O trabalho de Losasso et al. [33] utilizou *octrees* para simular líquidos e gases, propondo também um novo método para a solução da equação de Poisson — presente na solução da pressão — que inspirou outros trabalhos baseados em *octrees*. Lentine et al. [29] adotaram o uso de *octrees* (assim como grades regulares e grades com múltiplos níveis de grades regulares) para refinar a malha regular e obter uma grade adaptativa, com o objetivo de resolver a equação de Poisson em uma grade mais grossa regular, se opondo ao restante dos cálculos feitos em

uma grade adaptativa. Goldade et al. [21] resolve os operadores diferenciais ao interpolar nós fantasmas, que possibilitam o uso do método das diferenças finitas; já Ando e Batty [1] utilizam operadores diferenciais discretizados para resolver os operadores nas grades não regulares.

Uma outra alternativa para o método das diferenças finitas é utilizar diferenças finitas com funções de base radial (RBF-FD). Este método já é utilizado em trabalhos de geociência computacional que simulam as equações de Navier-Stokes [10, 18, 3, 39]. A técnica baseia-se em diferenças finitas generalizadas e na interpolação de funções de base radial (RBF) [19]. Dado uma nuvem de n pontos $\mathbf{x}_j$, para $j = 1, ..., n$ — também chamada de vizinhança $\mathcal{N}_c$ — que satisfaz a equação:

$$y_j = s(\mathbf{x}_j), \tag{2.15}$$

a interpolação de $\mathcal{N}_c$ para um alvo $\mathbf{x}_c$ qualquer pode ser aproximada por:

$$s(\mathbf{x}_c) \approx \sum_{j=1}^{n} \lambda_j \phi(\|\mathbf{x}_c - \mathbf{x}_j\|) + \sum_{k=1}^{m} \beta_k p_k(\mathbf{x}_c). \tag{2.16}$$

O primeiro termo do membro direito da Equação (2.16) é a combinação linear de $\phi(\|\mathbf{x}_c - \mathbf{x}_j\|)$ que resulta na aproximação da interpolação e o segundo termo é a regularização da aproximação; nestes, $\mathbf{x}_c$ é o alvo, λ_j representa os pesos associados a cada $\phi(\|\mathbf{x}_c - \mathbf{x}_j\|)$, ϕ é uma função de base radial, $\mathbf{x}_j$ são os pontos vizinhos ao alvo e β_k são os pesos de cada $p_k(\mathbf{x}_c)$, que formam uma base de espaço de polinômios multivariados. Para determinar os pesos λ_j e β_k, as seguintes restrições são assumidas:

$$\sum_{j=1}^{n} \lambda_j p_k(\mathbf{x}_j) = 0, \quad k = 1, ..., m. \tag{2.17}$$

Resolvendo a Equação (2.16) para toda a vizinhança $\mathcal{N}_c$ conhecida e da Equação (2.15) e Equação (2.17), tem-se:

$$\begin{bmatrix} A & P \\ P^T & O \end{bmatrix} \begin{bmatrix} \lambda \\ \beta \end{bmatrix} = \begin{bmatrix} y \\ 0 \end{bmatrix}, \tag{2.18}$$

que é o sistema linear de ordem $n + m$ que resulta nos pesos λ e β para o cálculo da interpolação, em que A é a matriz de ordem n das funções $\phi(\|\mathbf{x}_c - \mathbf{x}_j\|)$ para $c, j = 1, .., n$; P é a matriz $n \times m$ de $p_k(\mathbf{x}_j)$ para $k = 1, ..., m$ e $j = 1, ..., n$; e O é uma matriz nula $m \times n$. No sistema linear (2.18) λ, β e y representam, respectivamente, os pesos λ_j e β_k e os y_i para $j = 1, ..., n$ e $k = 1, ..., m$.

Da Equação (2.16) é possível chegar no sistema para interpolar um operador diferencial L. Como um operador diferencial pode ser aproximado por:

$$Ls(\mathbf{x}_c) \approx \sum_{i=1}^{n} \lambda_i s(\mathbf{x}_i), \tag{2.19}$$

em que $s(\mathbf{x})$ pode ser substituída por uma RBF ϕ, como na Equação (2.16), resultando em:

$$L\phi(\|\mathbf{x}_c - \mathbf{x}_j\|) \approx \sum_{i=1}^{n} \lambda_i \phi(\|\mathbf{x}_i - \mathbf{x}_j\|), \quad j = 1, ..., n. \tag{2.20}$$

Na Equação (2.20), $\mathbf{x}_i$ também representa os pontos da vizinhança $\mathcal{N}_c$ como $\mathbf{x}_j$. Ainda, pode-se adicionar o termo de regularização utilizado na Equação (2.16) à Equação (2.20), obtendo assim o sistema linear:

$$\left[\begin{array}{ccc|ccc} \phi_{11} & \cdots & \phi_{1n} & 1 & p_{11} & \cdots & p_{m1} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \phi_{n1} & \cdots & \phi_{nn} & 1 & p_{1n} & \cdots & p_{mn} \\ \hline 1 & \cdots & 1 & 0 & 0 & \cdots & 0 \\ p_{11} & \cdots & p_{1n} & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & 0 & \vdots & \ddots & \vdots \\ p_{m1} & \cdots & p_{mn} & 0 & 0 & \cdots & 0 \end{array}\right] \left[\begin{array}{c} \lambda_1 \\ \vdots \\ \lambda_n \\ \hline \beta_1 \\ \beta_2 \\ \vdots \\ \beta_m \end{array}\right] = \left[\begin{array}{c} L\phi_{c1} \\ \vdots \\ L\phi_{cn} \\ \hline L1|_{\mathbf{x}=\mathbf{x}_c} \\ Lp_1|_{\mathbf{x}=\mathbf{x}_c} \\ \vdots \\ Lp_m|_{\mathbf{x}=\mathbf{x}_c} \end{array}\right], \tag{2.21}$$

em que a notação ϕ_{ij} representa $\phi(\|\mathbf{x}_i - \mathbf{x}_j\|)$. Por exemplo, se a base de polinômios multivariados for de grau 1, o sistema resultante será:

$$\left[\begin{array}{ccc|ccc} \phi_{11} & \cdots & \phi_{1n} & 1 & x_1 & y_1 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ \phi_{n1} & \cdots & \phi_{nn} & 1 & x_n & y_n \\ \hline 1 & \cdots & 1 & 0 & 0 & 0 \\ x_1 & \cdots & x_n & 0 & 0 & 0 \\ y_1 & \cdots & y_n & 0 & 0 & 0 \end{array}\right] \left[\begin{array}{c} \lambda_1 \\ \vdots \\ \lambda_n \\ \hline \beta_1 \\ \beta_2 \\ \beta_3 \end{array}\right] = \left[\begin{array}{c} L\phi_{c1} \\ \vdots \\ L\phi_{cn} \\ \hline L1|_{\mathbf{x}=\mathbf{x}_c} \\ Lx|_{\mathbf{x}=\mathbf{x}_c} \\ Ly|_{\mathbf{x}=\mathbf{x}_c} \end{array}\right]. \tag{2.22}$$

No sistema linear (2.22), x_i e y_i são os polinômios avaliados em cada ponto i da vizinhança de $\mathbf{x}_c$. Esse sistema linear calcula os pesos λ_i para estênceis bidimensionais; para estênceis tridimensionais, o polinômio z_i será adicionado à matriz do lado esquerdo do sistema, $Lz|_{\mathbf{x}=\mathbf{x}_c}$ ao lado direito e a submatriz quadrada de zeros terá ordem 4. Portanto, a determinação dos pesos de um determinado estêncil de diferenças finitas utilizando RBF-FD é obtida pela resolução do sistema linear (2.22), da mesma forma que a obtenção dos pesos para o cálculo da interpolação no sistema linear (2.18).

Recentemente, Nakanishi et al. [37] propuseram um solucionador híbrido que alia uma grade adaptativa e RBF para calcular interpolações e operadores diferenciais. A grade adaptativa adotada — *Cellgraph* — é baseada na hierarquia e subdivisão das *quadtrees* e *octrees*, mas é implementada como um grafo, em que os vértices são associados às folhas da árvore e guardam a caixa delimitadora mínima de eixo alinhado (AABB) e as arestas ligam as células que são vizinhas. Ainda, a *Cellgraph* permite que a grade ajuste a sua resolução dinamicamente, através da subdivisão ou remoção de células de níveis mais baixos, sem a necessidade

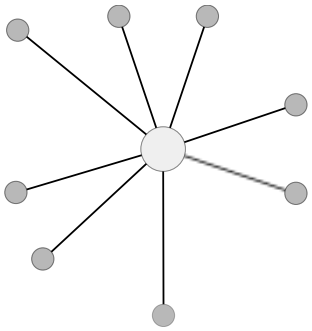
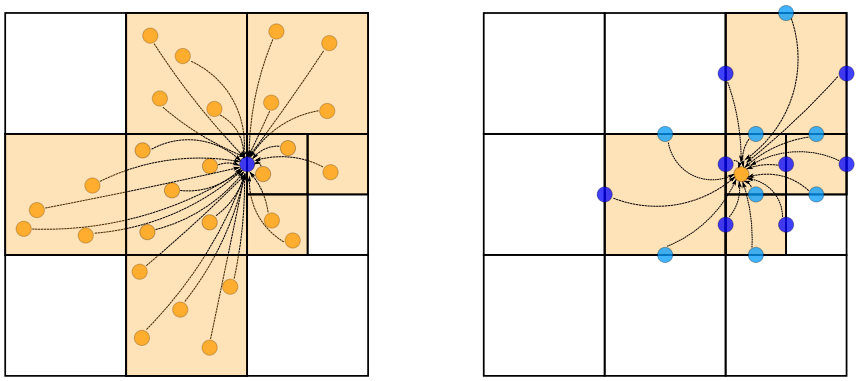


Figura 2.5: Nuvem de pontos formando um estêncil.

de reconstruir a estrutura a cada passo de tempo. Quando o número de partículas dentro das novas células for insuficiente ou excessivo, as partículas são excluídas e novas 2^d, sendo d a dimensão da simulação, são geradas, num processo chamado *reseeding*. Com a possibilidade de células de tamanhos diferentes existirem na grade, a transferência da velocidade entre partículas e grade sofre alteração. Como ilustrado na Figura 2.6, na transferência das partículas para grade, o ponto na grade recebe a interpolação das velocidades das partículas das células adjacentes à face do ponto e das vizinhas face a face dessas células; já na transferência da grade para as partículas, a partícula recebe a interpolação de todas as velocidades das células vizinhas à célula que contém a partícula.



(a) (b)

Figura 2.6: Exemplo da transferência da velocidade entre pontos da grade e partículas para grades não regulares. (a) Transferência da velocidade das partículas para o ponto na célula. (b) Passo inverso, em que a velocidade é transferida de pontos da célula para a partícula.

Os estênceis gerados pela grade são divididos pela operação diferencial que será aproximada, como ilustrado na Figura 2.7. Dada uma célula, o Laplaciano da pressão e o divergente da velocidade são calculados sobre o centro da célula, sendo que o primeiro toma como pontos vizinhos os centros das células vizinhas e outro toma os centros das faces vizinhas. Já o alvo do gradiente da pressão é o centro das faces da célula e a vizinhança é composta pelos centros das células vizinhas. As células vizinhas, das quais são retiradas os pontos que formam a vizinhança para o cálculo dos pesos RBF, são obtidas por meio de critérios diferentes para cada estêncil. Para o cálculo do Laplaciano, as células vizinhas são as células que possuem no mínimo uma face ou um vértice em comum com a célula que contém o alvo; para o estêncil

do divergente, as células da vizinhança são aquelas que possuem uma face em comum com a célula que contém o alvo; e por fim, as células que compõem a vizinhança do gradiente são as células que possuem uma face em comum a uma das duas células que contém a face alvo.

Estudos numéricos foram feitos para avaliar o uso de RBF para o cálculo de interpolações e operadores diferenciais em grades adaptativas, sem limitações na diferença entre níveis de células vizinhas, que apresentam resultados satisfatórios quando comparados com outras técnicas da literatura. Além disso, a transferência entre partículas e grades com RBF apresentou um erro quadrático médio menor que a técnica *Moving Least Square* e, quando comparado a outros métodos, preserva mais energia cinética das partículas. Nas simulações, o solucionador preserva mais vorticidade que o PIC ou FLIP e se iguala ao APIC, além de ser mais resistente à dissipação artificial sem sacrificar a estabilidade da simulação.

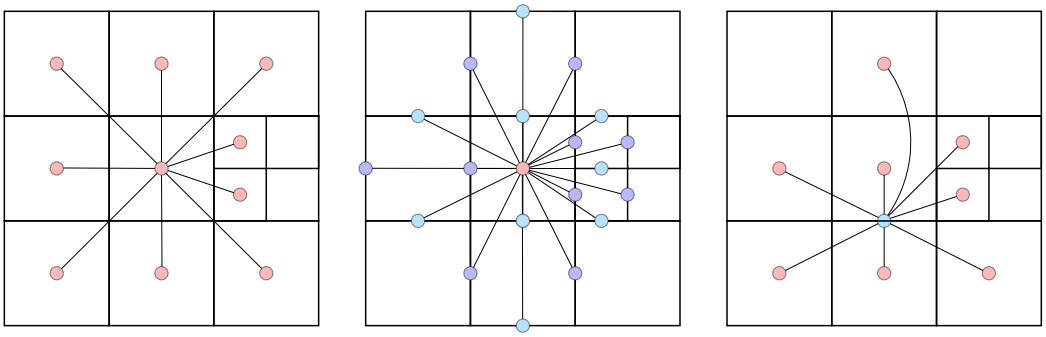


Figura 2.7: Exemplos de estênceis para o cálculo dos operadores diferenciais em uma grade adaptativa. (a) Estêncil para o cálculo do Laplaciano da pressão. (b) Estêncil para o cálculo do divergente da velocidade. (c) Estêncil para o cálculo do gradiente da pressão. (Adaptado de Nakanishi et al. [37].)

Em contrapartida, a sua maior limitação é o cálculo dos pesos RBF, uma vez que para cada célula em que deseja-se calcular um operador diferencial, é necessário a resolução de um sistema linear diferente, cada um dependente da vizinhança do ponto sendo avaliado, como mostrado na Equação (2.20). Considerando ainda que células da grade podem ser subdivididas e reagrupadas em passo de tempo distintos da simulação de acordo com a distribuição do fluido no espaço, pode ser necessário resolver um novo sistema linear de RBF-FD a cada passo de tempo. Assim, apesar de resolver o problema da quantidade de volume de fluido em cada célula com uma grade adaptativa, RBF-FD introduz a necessidade de mais cálculos para resolver os operadores diferencias ao longo da simulação, já que cada configuração de estêncil distinta gera um sistema linear novo.

2.4 Considerações finais

Nesse capítulo três abordagens para simulação de fluidos baseada em física foram descritas, sendo elas: método baseado em partículas, em grade e híbrido. Como os três métodos utilizam as equações de Navier-Stokes para simular fluidos incompressíveis, todos eles seguem

passos similares, sendo, para fluidos invíscidos, o cálculo do gradiente da pressão e força da gravidade, além do passo da advecção, para obter o estado do fluido no próximo passo de tempo da simulação. Além disso, as particularidades, bem como limitações de cada método foram abordadas e discutidas para justificar a escolha dos métodos híbridos como foco deste trabalho.

Grades adaptativas foram apresentadas como alternativa às grade regulares e o método de diferenças finitas baseadas em funções de bases radiais (RBF-FD) foi introduzido como alternativa ao método das diferenças finitas para o cálculo dos operadores diferenciais nas grades adaptativas, dado as novas configurações de estênceis decorrentes da discretização espacial.

Capítulo 3

PIC com RBF-FD e grades adaptativas balanceadas

3.1 Considerações iniciais

Este capítulo apresenta o solucionador proposto para contornar os gargalos apontados na Seção 2.3, o PIC utilizando RBF-FD e grades adaptativas balanceadas, e descreve as diferenças entre o solucionador proposto por Nakanishi et al. [37] (Furoo) e o proposto nesta tese, além de detalhar seus aspectos de implementação. Serão especificadas as classes paramétricas empregadas para representar a grade e as partículas, base da simulação de fluidos, além das que representam os estênceis dos operadores diferenciais e as funções de base radial. Ainda, a estrutura e funcionamento do dicionário de estênceis, proposto para acelerar o cálculo dos operadores diferenciais, será descrita. Para evitar que o texto torne-se verboso, neste capítulo o Laplaciano da pressão, o divergente da velocidade e gradiente da pressão serão citados apenas como Laplaciano, divergente e gradiente, respectivamente.

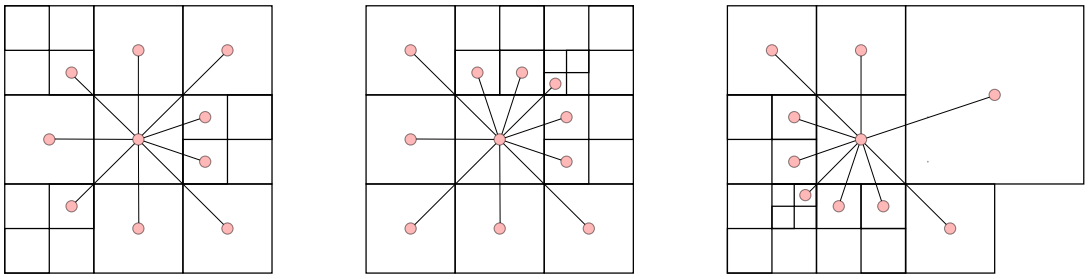
3.2 Solucionador PIC

A implementação do solucionador proposto neste trabalho baseia-se nos trabalhos de Nakanishi et al. [37] e Kim [28]; o método proposto pelos primeiros emprega grades adaptativas e RBF-FD, como detalhado na Seção 2.3 e o segundo traz os passos que os solucionadores PIC e FLIP tomam para resolver as equações de Navier-Stokes. Como na implementação do Furoo, adota-se o uso de grades adaptativas, mais especificamente nesta implementação, as *quadtrees*, que diferente de grades regulares, podem ser refinadas em regiões que contém mais fluido. A diferença entre os níveis de células vizinhas que a adaptatividade gera impossibilita o uso de diferenças finitas como nas grades regulares para o cálculo de interpolações e operadores diferenciais, ambos necessários para a implementação de uma simulação de fluidos.

Como alternativa, optou-se por usar funções de bases radiais, que viabiliza operações sobre uma nuvem de pontos em oposição aos pontos alinhados necessários para realizar cálculos com diferenças finitas.

A grade implementada no Furoo não possui limitação quanto à diferença de níveis entre células vizinhas, como ilustrado na Figura 1.3(b), ou seja, a cada passo de tempo da simulação novos estênceis podem ser gerados quando a grade é atualizada, aumentando ou diminuindo a vizinhança necessária para montar os sistemas lineares RBF-FD. Como apontado pelos autores, o cálculo dos pesos dos sistemas RBF-FD e da interpolação utilizando RBF-FD repetida vezes é uma limitação do método. Para contornar essa limitação, uma grade adaptativa balanceada — em que a diferença de nível entre células vizinhas é menor ou igual a um — é proposta, com o intuito de reduzir a um número fixo as diferentes configurações de estênceis. Apesar do balanceamento da estrutura introduzir células que poderiam ser menos refinadas, ainda pode resultar em menor quantidade quando comparado com uma estrutura regular.

Estas células vazias são justificadas pela redução das configurações dos estênceis a um número fixo, que torna viável a implementação de uma função dedicada à resolução de cada sistema possível, acessível através de um dicionário que mapeia a chave de cada estêncil a uma entrada no dicionário. A grade adaptativa balanceada gera, para o caso bidimensional, 133 estênceis para o cálculo do Laplaciano, dos quais alguns são ilustrados na Figura 3.1. Na Figura 3.2 e na Figura 3.3 há exemplos dos estênceis do divergente e gradiente, respectivamente, que também totalizam mais de 100 estênceis.

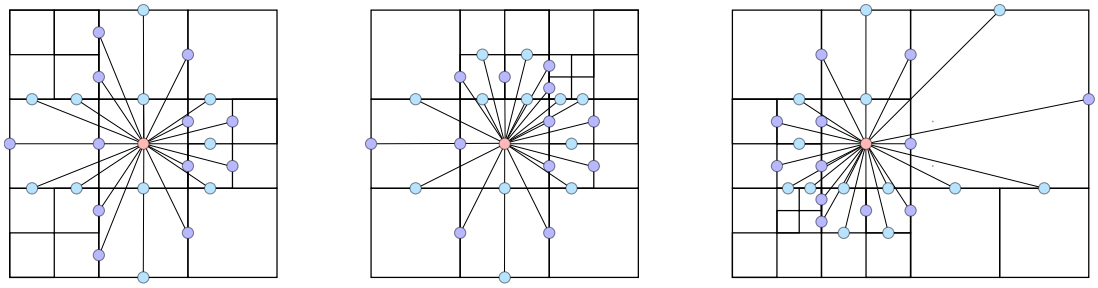


(a) (b) (c)

Figura 3.1: Exemplos de estênceis para o cálculo do Laplaciano em uma *quadtree* balanceada.

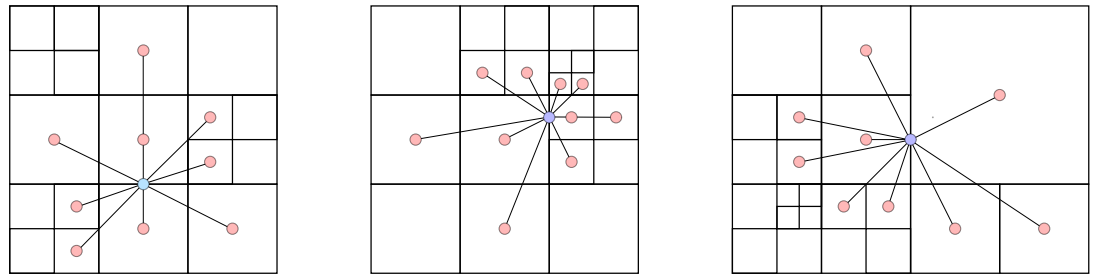
3.2.1 *Pipeline* do solucionador

Entre as principais diferenças entre o Furoo e o solucionador proposto neste trabalho estão a adoção de uma grade adaptativa balanceada, a implementação de um dicionário para acelerar os cálculos dos pesos dos operadores diferenciais e o uso do *reseeding* apenas antes do primeiro passo de tempo da simulação. O *pipeline* utilizado foi o mesmo do PIC implementado por Kim [28], com as mudanças das estruturas de dados, método para obtenção dos operadores diferenciais e transferência da velocidade entre grade e partículas. Os passos



(a) (b) (c)

Figura 3.2: Exemplos de estênceis para o cálculo do divergente em uma *quadtree* balanceada. Os pontos azuis escuro e o ponto rosa central formam o estêncil para a velocidade na componente x e os pontos azuis claro e o ponto rosa formam o estêncil para a componente y.



(a) (b) (c)

Figura 3.3: Exemplos de estênceis para o cálculo do gradiente em uma *quadtree* balanceada.

do solucionador estão detalhados na Seção 2.2.4.

Tanto no **P1** e **P3**, a transferência da velocidade segue os mesmos passos do PIC, com exceção da interpolação, que é calculada com o RBF-FD, como no Furoo. A computação da gravidade e atualização da posição das partículas continuam como no PIC, com mudanças pontuais para acomodar a mudança na grade. Por fim, o cálculo da pressão e incompressibilidade sofrem a maior modificação, uma vez que os operadores diferenciais são utilizados neste passo. Na Tabela 3.1 os métodos do solucionador que têm implementação igual aos solucionadores de Nakanishi et al. [37] e Kim [28], com exceção das estruturas de dados, estão marcados.

Tabela 3.1: Comparação entre os métodos do *pipeline* entre os solucionadores.

Etapa do *pipeline*	Kim	Furoo	Novo
Condição de borda	•		
Transferir de velocidade de partícula para grade			•
Computar gravidade	•		
Computar viscosidade		•	
Computar pressão		•	•
Transferir velocidade da grade para partícula		•	
Computar advecção	•		
Reseeding das partículas		•	

3.3 *Tree Based Graded Grid*

A estrutura da árvore utilizada neste solucionador é uma extensão de uma implementação disponível na API de funções auxiliares para aplicações gráficas desenvolvida pelo grupo de simulação e processamento geométrico da FACOM-UFMS, que possui implementações de *quadtree* e *octree*, disponível na classe *PointTree*. Essa implementação usa ponteiros para relacionar os nós pais e filhos, diferente do *Cellgraph* apresentada na Seção 2.3. A classe *PointTree* implementa uma grade adaptativa balanceada com partículas de propósito geral, e possui métodos básicos a essa estrutura, como métodos para criar uma árvore, refiná-la, balanceá-la e meios para obter iteradores para suas folhas. Essa estrutura implementa um refinamento *top-down*, ou seja, a estrutura parte de uma única folha e vai subdividindo-a até que o critério de refinamento escolhido seja obedecido. Para a implementação do solucionador proposto, um dos critérios para o refinamento é a quantidade de partículas em cada folha, sendo que uma folha não pode ter mais do que um certo número (configurável) de partículas. Também como critério há a obrigatoriedade das células que formam a interface fluido e não fluido estarem refinadas até o nível máximo da árvore.

Como há passos da simulação que serão resolvidos na grade, é preciso acessar os dados necessários para os cálculos na grade, ou seja, a grade precisa ser capaz de acessar pressões nos centros das células e velocidades nos centros das faces, como discutido na Seção 2.2.4. Para tal, a classe *PointTree* precisou ser estendida para a classe *Tree Based Graded Grid* (TBGG), ilustrada na Figura 3.4, que acomoda células e faces. Uma TBGG possui uma lista de faces *Face*, além de já ser capaz de acessar os iteradores para as folhas da árvore, ou seja, para as células da grade; e, dado um iterador para uma folha e uma direção d, a TBGG é capaz de retornar, para aquela célula, o centro da face na direção d.

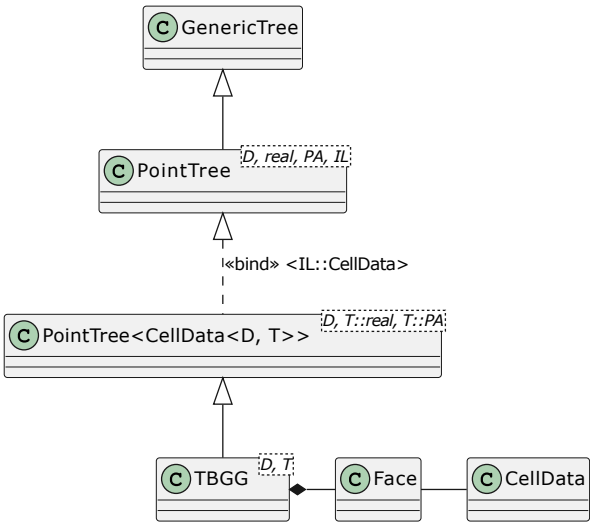


Figura 3.4: Diagrama de classes UML da hierarquia das classes que modela a *Tree Based Graded Grid*, que adiciona faces à árvore e as associa às células da estrutura.

Uma vez que a grade herdada de *PointTree* foi implementada de forma genérica e não possui faces explícitas, apesar de ser possível recuperar um iterador para uma célula, essa

célula não possui conhecimento de suas faces. Logo, a classe *CellData*, que guarda uma lista das suas faces e ainda é capaz de retornar suas faces em uma direção d foi implementada. Por fim, a classe *Face* guarda os iteradores para as suas células vizinhas, além de conhecer o eixo ao qual é perpendicular.

3.4 Diferenças finitas com funções de base radial

Para resolver o RBF-FD, foram criadas classes para representar as funções de base radial, os operadores diferenciais e os tipos de configurações de estênceis gerados por esses. Foram implementadas duas funções de base radial nas classes *Cubic* e *Quintic* — parametrizadas por uma dimensão D e um ponto flutuante *real* —, sendo elas as spline poli-harmônicas $\phi(\|\mathbf{r}\|) = \|\mathbf{r}\|^3$ e $\phi(\|\mathbf{r}\|) = \|\mathbf{r}\|^5$ respectivamente. Para cada uma dessas funções, foram calculados os valores dos operadores diferenciais dado um vetor $\mathbf{r}$. Tomando $\|\mathbf{r}\| = t$ e da regra da cadeia para uma spline poli-harmônica $\phi(t) = t^s$ genérica, tem-se:

$$\frac{\partial \phi}{\partial r_i}(t) = s t^{s-2} r_i, \tag{3.1}$$

$$\frac{\partial^2 \phi}{\partial r_i^2}(t) = s(s-2) t^{s-4} r_i^2 + s t^{s-2}, \tag{3.2}$$

em que r_i é o i-ésimo termo do vetor $\mathbf{r}$. Da definição do Laplaciano na Equação (2.9) e da Equação (3.2), assumindo $s = 3$ e $i = 2$, ou seja, calculando o Laplaciano para uma simulação bidimensional utilizando uma spline poli-harmônica cúbica, obtém-se:

$$\begin{aligned} & 3\left(\frac{r_x^2}{t} + t\right) + 3\left(\frac{r_y^2}{t} + t\right) \\ &= 3\left(\frac{r_x^2}{t} + \frac{r_y^2}{t} + 2t\right) \\ &= 6t + \frac{3}{t}(r_x^2 + r_y^2) \\ &= 3Dt + 3t \\ &= 3(D+1)t. \end{aligned} \tag{3.3}$$

De forma análoga, deduz-se os outros operadores, tanto para a spline cúbica quanto para a quíntica. Em especial, a implementação do gradiente toma como argumento, fora o vetor $\mathbf{r}$, uma direção d, que corresponde o eixo do gradiente da velocidade que está sendo calculado.

Implementou-se também as classes dos operadores diferenciais e operadores para interpolação propriamente ditos, sendo elas, *LaplacianOp*, *DivergenceOp*, *GradientOp*, *UnityOp*, *ParticleToGridOp* e *GridToParticleOp*, todas parametrizadas em função do D e do *real*, como as classes das funções de base radial. As classes dos operadores são responsáveis por, dado uma função de base radial, devolver o resultado do operador sobre um vetor $\mathbf{r}$, o escalar

1 ou um polinômio p_k, necessário para montar o sistema RBF-FD, como visto no sistema linear 2.21.

Para a montagem e solução do sistema RBF-FD, duas funções parametrizadas em função do *D*, *real*, uma função de base radial *RBF*, um operador *DOp* e um estêncil *Stencil* — discutido na Seção 3.4.1 — foram implementadas. As funções *computeWeights* e *computeInterpolationWeights* calculam os pesos dos operadores diferenciais e das interpolações entre grade e partículas, respectivamente. Ambas funções montam os sistemas lineares discutidos na Seção 2.3, sendo que a primeira função implementa o sistema linear 2.21 e a segunda o sistema linear 2.18, e os resolve utilizando a biblioteca Eigen [22].

3.4.1 Estênceis

As configurações de estênceis precisam ser determinadas dada uma face ou célula alvo e o operador diferencial sendo calculado, para tal, classes responsáveis por essa determinação e, quando necessário, pela geração da chave de identificação de cada configuração de estêncil — detalhada na Seção 3.5 — foram implementadas. A hierarquia dessas classes está ilustrada na Figura 3.5, sendo a *StencilBase* a classe que implementa os métodos básicos comuns a todas as configurações de estênceis, como o retorno da árvore a que pertence, o tamanho do estêncil e o seu alvo.

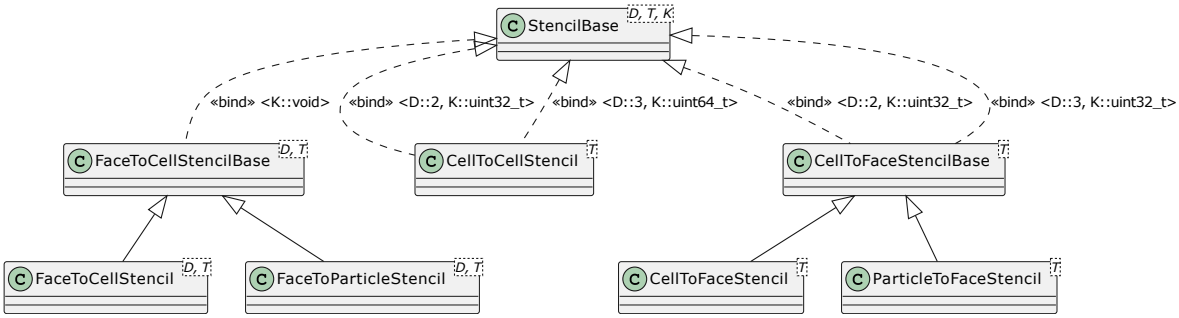


Figura 3.5: Diagrama de classes UML da hierarquia das classes que modelam os estênceis para o cálculo das interpolações e dos operadores diferenciais.

Como pode ser observado na Figura 3.5, herdam de *StencilBase*, as classes *CellStencilBase*, *CellToFaceStencil*, *ParticleToGridStencil* e *GridToParticleStencil*. A primeira encapsula os métodos comuns às configurações de estênceis que tomam como alvo o centro de uma célula, como é o caso das configurações para os operadores Laplaciano (*CellToCellStencil*) e divergente (*FaceToCellStencil*); já a segunda é a classe que monta as configurações para o cálculo do gradiente, que toma o centro de um face como alvo e os centros de células como vizinhos. As últimas duas classes são responsáveis por montar os estênceis para o cálculo da interpolação da velocidade entre grade e partículas. Exemplos de todas essas configurações são ilustradas na Figura 2.7 e Figura 2.6.

Todas as classes podem calcular o número máximo de vizinhos que um estêncil pode ter dada a dimensão da sua simulação, além de ter acesso ao seu i-ésimo vizinho. Para

determinar a vizinhança e, quando preciso for, gerar a chave de identificação da configuração, as classes invocam o método *setTarget*, que cria uma vizinhança de iteradores e depois, o método *points* gera o conjunto de posições dos pontos da vizinhança, ambos específicos para cada configuração. Para a configuração do Laplaciano, para cada uma das $D \times 2$ direções em que pode-se ter células vizinhas face a face, ou seja, aquelas que dividem uma mesma face, determina-se quantas vizinhas existem e seus níveis em relação à célula alvo, sendo as opções: nenhuma vizinha, uma vizinha no mesmo nível do alvo, uma vizinha em um nível acima do alvo e 2 (no caso bidimensional) ou 4 (no caso tridimensional) vizinhas, que sempre serão um nível abaixo do alvo. Em seguida, para cada nó da árvore — que não precisa ser folha — vizinho à célula alvo, com o auxilio de uma tabela estática pré-computada, busca seu vizinho que tem potencial de compartilhar um vértice com a célula alvo. Verificações são realizadas para classificar a célula obtida em vizinha ou não.

A construção da vizinhança para o cálculo do divergente e para a transferência da velocidade da grade para a partícula é idêntica, sendo que a única diferença entre os estênceis formados por estes é o ponto alvo da operação, que para o divergente é o centro da célula alvo e para a transferência são as partículas contidas na célula alvo, formando assim um estêncil para cada partícula. Os pontos dos estênceis destes operadores são obtidos recuperando-se os centros das faces de todas as células vizinhas face a face à célula alvo. A vizinhança do gradiente e da transferência da velocidade de partículas para a grade também são semelhantes, uma vez que as células que participam da vizinhança são obtidas através das vizinhas face a face das duas células que dividem a face alvo. Nestes casos, o ponto alvo não muda, em contrapartida, os pontos da vizinhança mudam, sendo os centros das faces das células vizinhas para o gradiente e todas as partículas contidas nas células vizinhas para a transferência.

3.5 Dicionário

Como introduzido na Seção 3.2, o uso da grade adaptativa balanceada torna as configurações de estênceis fixos, ou seja, pode-se obter os sistemas RBF-FD para cada caso e gerar uma função para resolvê-los. Para acessar essas funções, é empregado um dicionário, que é uma estrutura de dados cujo elementos são pares de chave e valor. O dicionário armazena então os endereços das funções para a resolução dos sistemas RBF-FD e a chave para acesso é gerada a partir da configuração de estêncil para a qual quer se resolver o sistema.

A chave é uma sequência de *bits* obtida a partir do número de vizinhos do alvo e níveis em relação ao alvo analisado, como ilustrado na Figura 3.6 e Figura 3.7. As chaves para o estêncil do Laplaciano 2D podem ser divididas em 3 partes: os *bits* que indicam a profundidade do alvo, os que indicam a quantidade de vizinhos face a face em relação ao alvo em cada direção e os que indicam a quantidade de vizinhos vértice a vértice do alvo em cada direção. Da sequência de *bits*, 4 são reservados para indicar a profundidade do alvo, 12 para identificar os vizinhos face a face e 12 para identificar os vizinhos vértice a vértice.

Os 12 *bits* reservados para identificar os vizinhos face a face são divididos em duplas que representam os vizinhos pra cada direção, esquerda, direito, baixo, cima, frente e trás, sendo as últimas duas direções ignoradas quando no caso bidimensional. Já os *bits* das diagonais são divididos em trios para cada diagonal, começando pela diagonal entre a direção esquerda e baixo, seguindo em sentido anti-horário. As duplas e trios de *bits* representam os níveis e quantidade de vizinhos de acordo com a Tabela 3.2.

Tabela 3.2: Significado dos códigos (duplas ou trios de *bits*) que compõem a chave do Laplaciano. Para os vizinhos face a face um nível abaixo da célula alvo, há duas opções de quantidade de vizinhos, 2 vizinhos para o caso bidimensional e 4 vizinhos para o caso tridimensional.

Código	Tipo de vizinho	Quantidade de vizinhos	Nível em relação ao alvo
00	Face a face	0	Não aplicável
01	Face a face	1	Mesmo nível
11	Face a face	1	1 nível acima
10	Face a face	2 ou 4	1 nível abaixo
000	Vértice a vértice	0	Não aplicável
001	Vértice a vértice	1	Mesmo nível
010	Vértice a vértice	1	1 nível abaixo
011	Vértice a vértice	1	1 nível acima
100	Vértice a vértice	1	2 níveis abaixo
101	Vértice a vértice	1	2 níveis acima

A sequência gerada pelas configurações do operador gradiente pode ser dividida em duas sequências, cada uma tomando como alvo uma das células que divide a face alvo. Como no Laplaciano, 4 *bits* são reservados para indicar a profundidade do alvo, e 18 *bits* são separados em trios que indicam a quantidade e posição dos vizinhos face a face das duas células vizinhas à face alvo. A Tabela 3.3 mostra os códigos que o trio *bits* podem assumir.

Tabela 3.3: Significado dos trios de *bits* que compõem a chave do gradiente.

Código	Tipo de vizinho	Quantidade de vizinhos	Nível em relação ao alvo	Vizinho é compartilhado com célula alvo?
000	Face a face	0	Não aplicável	Não aplicável
001	Face a face	1	Mesmo nível	Não aplicável
011	Face a face	1	1 nível acima	Não
111	Face a face	1	1 nível acima	Sim
010	Face a face	2 ou 4	1 nível abaixo	Não aplicável

A primeira estratégia testada foi o uso de um dicionário cujas entradas armazenavam ponteiros para funções que calculariam os pesos RBF-FD, funções essas que foram obtidas através de um programa de matemática simbólica. Esse dicionário foi apelidado de dicionário simbólico. Em seguida, testou-se o uso de um dicionário que armazena os pesos RBF-FD que já foram calculados para posterior consulta, apelidado de dicionário iterativo.

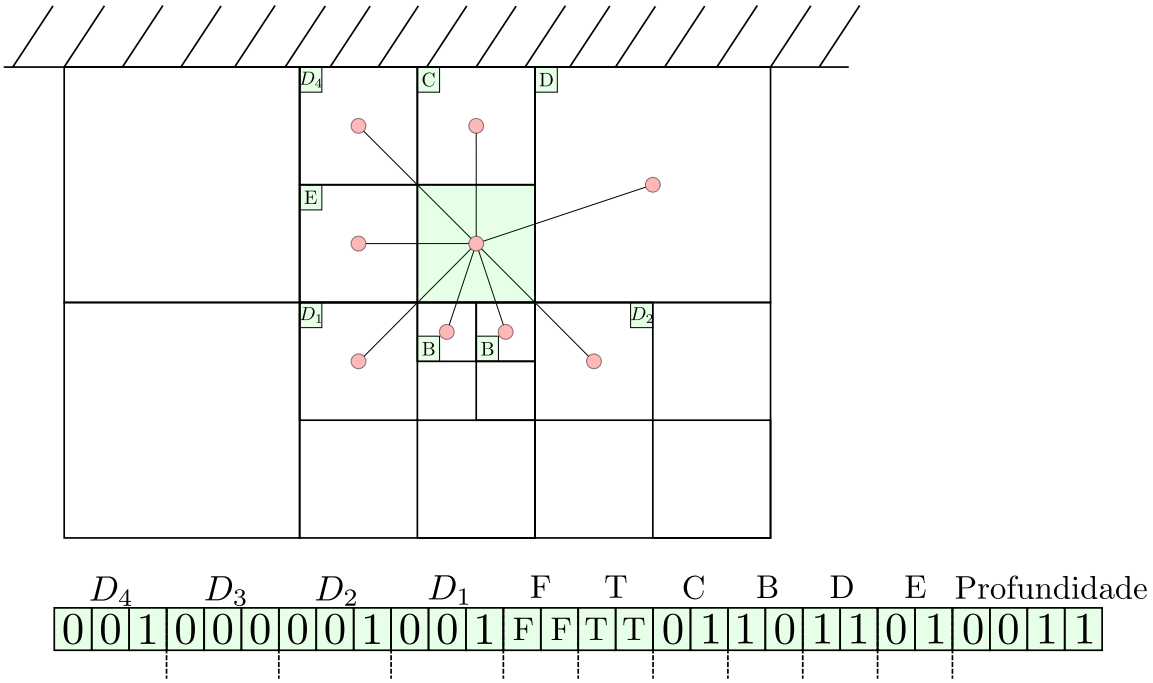


Figura 3.6: Exemplo de chave para um estêncil do operador Laplaciano em uma grade bidimensional.

3.5.1 Dicionário simbólico

A primeira proposta para acelerar o cálculo dos sistemas RBF-FD foi resolver todos os sistemas lineares para cálculo de pesos de forma simbólica uma vez que ferramentas de computação simbólica podem ser utilizadas para obter o código de funções para a resolução dos sistemas lineares, evitando assim calcular numericamente sistemas lineares diferentes para cada configuração de estêncil gerada. Estas funções podem ser parametrizadas em função da vizinhança da célula para a qual se deseja calcular os operadores diferenciais. Para exemplificar, o *script* MATLAB a seguir gera uma função para cálculo dos pesos de uma configuração de estênceis que possui 5 pontos, utilizando como função ϕ uma spline poli-harmônica de expoente 5, como sugerido por Nakanishi et al. [37].

```
syms x1 x2 x3 x4 x5 y1 y2 y3 y4 y5 real

p = @(r) r.^5;

r12 = sqrt((x1 - x2) .^ 2);
r13 = sqrt((x1 - x3) .^ 2);
r14 = sqrt((x1 - x4) .^ 2);
r15 = sqrt((x1 - x5) .^ 2);
r23 = sqrt((x2 - x3) .^ 2);
r24 = sqrt((x2 - x4) .^ 2);
r25 = sqrt((x2 - x5) .^ 2);
r34 = sqrt((x3 - x4) .^ 2);
r35 = sqrt((x3 - x5) .^ 2);
r45 = sqrt((x4 - x5) .^ 2);

p0 = p(0);
p12 = p(r12);
```

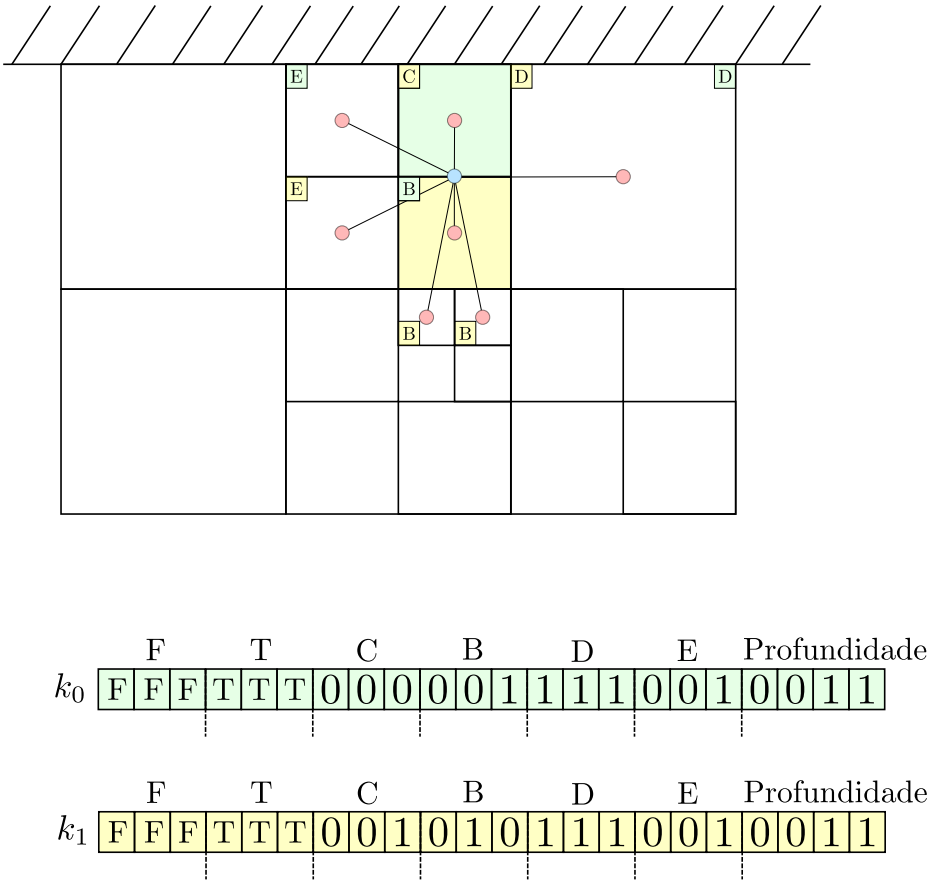


Figura 3.7: Exemplo de chave para um estêncil do operador gradiente em uma grade bidimensional.

```
p13 = p(r13);
p14 = p(r14);
p15 = p(r15);
p23 = p(r23);
p24 = p(r24);
p25 = p(r25);
p34 = p(r34);
p35 = p(r35);
p45 = p(r45);

A = [p0 p12 p13 p14 p15; p12 p0 p23 p24 p25; p13 p23 p0 p34 p35; p14 p24 p34 p0 p45; p15 p25 p35 p45
↪  p0];
A = [A [1 x1 y1; 1 x2 y2; 1 x3 y3; 1 x4 y4; 1 x5 y5]];
A = [A; [1 1 1 1 1 0 0 0; x1 x2 x3 x4 x5 0 0 0; y1 y2 y3 y4 y5 0 0 0]];
Ainv = inv(A);
ccode(Ainv,'File','quali-invA.c');
```

O *script* gera uma arquivo em C que contém o código de uma função que determina a inversa da matriz do sistema linear para os casos que tem 4 vizinhos e o ponto central, totalizando 5 pontos. Com a inversa, pode-se calcular facilmente o sistema. Esta função toma como parâmetros os valores da função de base radial e os valores dos polinômios nos pontos do estêncil. Um teste foi executado em um computador com Intel Core i7-8750H 2.20GHz e 16GB de RAM para avaliar a viabilidade de resolver a inversão da matriz de forma simbólica. A geração do código simbólico durou mais de 24 horas, além disso, para

a solução de 1000 matrizes aleatórias, a versão simbólica gastou quase 10 vezes mais tempo que a inversão das matrizes feita pela biblioteca Eigen [22].

3.5.2 Dicionário iterativo

Alternativamente à proposta simbólica, decidiu-se por adotar a biblioteca Eigen para resolver os sistemas RBF-FD e manter um dicionário iterativo dos pesos já calculados e consultá-los posteriormente. Esse dicionário é vazio no início da simulação e a medida que os passos de tempo são processados, vai sendo populado. Ao obter a chave do operador diferencial sendo calculado, o dicionário é consultado e a existência de uma entrada com o valor da chave é verificada. Caso exista uma entrada no dicionário correspondente à chave, os pesos RBF-FD são recuperados, evitando o cálculo do sistema linear RBF-FD. Caso não exista uma entrada no dicionário, o sistema linear será montado e resolvido para obter os pesos RBF-FD e a dupla chave e pesos é adicionada ao dicionário para consultas futuras.

As classes do dicionário seguem como ilustradas na Figura 3.8. A classe *DictionaryBase* possui os métodos comuns às especializações de *Dictionary*, como os métodos que fazem a impressão dos pesos ou de todo o dicionário e o cálculo da reusabilidade do dicionário, que é o quanto os pesos já armazenados pelo dicionário são reutilizados. Além desses métodos, há um que retorna os iteradores para as células ou faces de um estêncil e outro que calcula os pesos dado um alvo *Target*. O *Target* é uma face ou uma célula, alvo do operador diferencial sendo calculado, e este método verifica se para aquele alvo já existe uma dupla chave-peso no dicionário, se sim, ele devolve os pesos que estão no dicionário, caso contrário ele invoca o método *generateWeights*, que é especializado para cada operador, que calcula os pesos para essa nova configuração de estêncil.

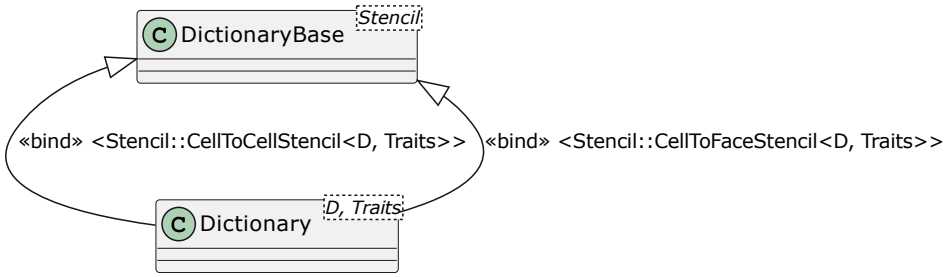


Figura 3.8: Diagrama de classes UML da hierarquia das classes que modelam os dicionários utilizados para acelerar o cálculo dos operadores diferenciais.

Há 2 classes que herdam de *DictionaryBase* e especializam *Dictionary*, uma classe abstrata, cada uma delas especializando para um estêncil de operadores diferenciais utilizados na simulação de fluidos, Laplaciano ou gradiente. Cada uma dessas especializações implementa o *generateWeights* e também o método de impressão da chave dos estênceis do operador diferencial, uma vez que as chaves para cada um deles são diferentes. Como os estênceis interpolação da velocidade da grade para as partículas e vice-versa não são fixos como os estênceis dos operadores diferenciais, optou-se por não criar um dicionário para eles, além

disso, o divergente também foi excluído do dicionário, uma vez que não foi definida uma chave para as configurações desse operador.

3.6 Considerações finais

Esse capítulo detalha aspectos de implementação do solucionador proposto neste trabalho, passando pelos seus aspectos gerais, como a mudança da estrutura da grade da *CellGraph* do Furoo para uma *quadtree* visando diminuir o número de cálculos envolvendo RBF-FD e o *pipeline* do solucionador e as principais diferenças entre a implementação e os que serviram de base apara ela, sendo elas as apresentadas por Nakanishi et al. [37] e Kim [28]. Um vez que o código implementado neste trabalho utilizou a API de funções auxiliares para aplicações gráficas desenvolvida pelo grupo de simulação e processamento geométrico da FACOM-UFMS, as extensões necessárias para obter-se uma grade adaptativa que conhece as informações essenciais para a resolução das equações de Navier-Stokes também foram explicitadas.

Ademais, as classes responsáveis por montar e resolver os sistemas lineares para o cálculo dos pesos RBF-FD foram especificadas, com destaque aos cálculos que reescrevem os operadores diferenciais em termos da dimensão da simulação e do vetor sobre o qual a operação está sendo realizada para funções de base radial cúbica e quíntica. As principais funções das classes que representam cada tipo de estêncil também são detalhadas, como as funções responsáveis por determinar a vizinhança de um estêncil dado o ponto alvo da operação. Por fim, o dicionário, estrutura empregada para acelerar os cálculos RBF-FD também foi discutido. Uma versão preliminar falha e a versão final do dicionário foram detalhadas, com explicações da lógica por trás dessas e especificidades das funções da versão final.

Capítulo 4

Resultados

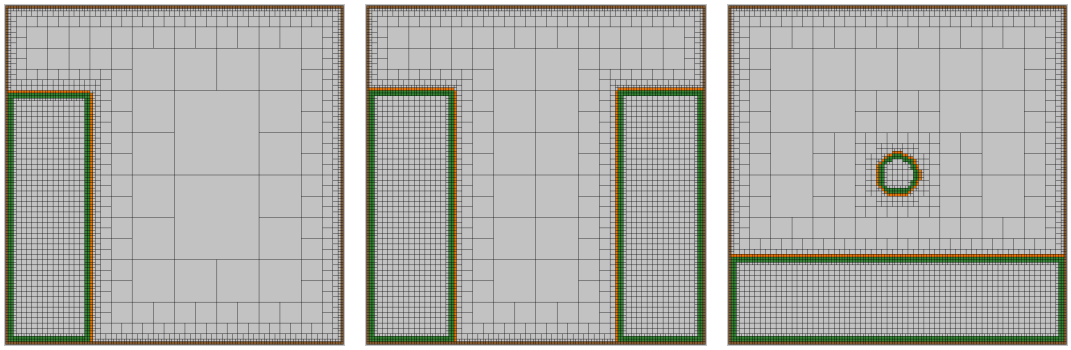
4.1 Considerações iniciais

Este capítulo discuti os resultados obtidos pelos experimentos realizados que quantificam a eficácia do uso ou não de dicionários para o cálculo dos pesos RBF-FD gerados por grades adaptativas balanceadas. Os cenários utilizados nos testes foram descritos e os parâmetros utilizados para a execução das simulações são expostos. O desempenho dos mapas são observados ao comparar o tempo de processamento gasto para o cálculo dos pesos dos operadores diferenciais e ao analisar o uso dos dicionários. Além disso, as dificuldades encontradas na implementação de um simulador de fluidos baseado no Furoo também foram comentadas.

4.2 Cenários iniciais

Os cenários escolhidos para testar o solucionador implementado foram os clássicos *dam break*, *double dam break* e *water drop*, que consistem, respectivamente em simulações de: uma coluna de água em um extremo do domínio que cai sob a ação da gravidade; duas colunas de água, cada uma em um extremo do domínio, que caem sob a ação da gravidade; e uma bola de água que cai sobre água em repouso. Para todos estes exemplos, o nível máximo e mínimo da árvore e o espaçamento entre partículas — $1/2^{m+1}$, em que m é o nível máximo — no momento de inicialização da cena foram retirados dos testes realizados no Furoo. Nas simulações bidimensionais o nível mínimo assumido é 4 e o máximo é 7, com o espaçamento entre as partículas sendo $0,00390625$ e o domínio é uma AABB quadrada de área 1, com o seu vértice inferior esquerdo na origem do plano, em que uma faixa de células no nível máximo de refinação na borda do domínio é composta de células sólidas. Essas células sólidas formam os limites de uma área que será utilizada como um domínio menor, que chamaremos de subdomínio, em que a simulação irá ocorrer.

Considerando o subdomínio descrito acima, a coluna de água da simulação *dam break* tem a origem como o ponto inferior esquerdo e $(0,25; 0,75)$ como o ponto superior direito. Esta mesma coluna está presente também no *double dam break*, além de uma segundo coluna que tem $(0,75; 0)$ e $(1; 0,75)$ como vértice inferior esquerdo e vértice superior direito, respectivamente. Na simulação *water drop*, o fluido em repouso tem o vértice inferior esquerdo na origem e o superior direto em $(1; 0,25)$ e a bola de água tem centro em $(0,5; 0,5)$ e o raio $0,625$. Essas cenas iniciais são ilustradas na Figura 4.1.



(a) (b) (c)

Figura 4.1: Quadro 0 dos cenários simulados para obter os resultados. As células em marrom representam os limites do domínio (sólido), as células laranjadas representam as células de ar que são vizinhas pelo menos uma célula de fluido e as células verdes representam as células de fluido que fazem parte da interface entre fluido e não fluido. (a) Quadro 0 para o *dam break*. (b) Quadro 0 para o *double dam break*. (c) Quadro 0 para o *water drop*.

4.3 Desempenho com o uso do dicionário

Os testes descritos acimas foram executados em experimentos inciais para medir quantitativamente o impacto do cálculo dos pesos com e sem o uso do dicionário (ou mapa) descrito na Seção 3.5. Estes cenários foram executados por um simulador de corpos granulares — simulador mais simples — implementado com base no método dos elementos discretos, em que considera-se a massa do corpo granular e as forças, além da gravidade, de repulsão, amortecimento e cisalhamento. Mais informações sobre o método de elementos discretos podem ser obtidas em no trabalho de Mishra [35]. A simulação de corpos granulares, aqui também chamados de partículas, não realiza o cálculo de operadores diferenciais, mas foi utilizada para calcular o desempenho do dicionário de forma independente. Para tal, simulou-se os cenários com o cálculo dos pesos dos operadores com e sem o mapa.

Os cenários foram simulados em um computador equipado com um processador AMD Ryzen 9 5950X, 16-Core, 32-Thread, 3.4GHz e 64 GB de DRAM e para o *dam break* e *double dam break* foram simulados mil passos — *frames* — e para o *water drop* foram simulados 600 passos, o que totalizou $16,7$ segundos de simulação para os dois primeiros cenários e 10 segundos de simulação para o último. A massa considerada para os grânulos foi de $0,0015$ e os coeficientes de repulsão, amortecimento e cisalhamento foram, respectivamente, $0,8$, $0,02$

e $0,02$; ademais, a gravidade assumiu uma velocidade de $-0,1$. Com esses parâmetros, foram gerados os quadros expostos na Figura 4.2, Figura 4.3 e Figura 4.4. Ainda, na Tabela 4.1 pode-se observar o percentual médio que o tempo de processamento de cada etapa do *pipeline* representa no tempo total da execução do passo.

Tabela 4.1: Diferença entre os tempos de processamento médio para os pesos do Laplaciano e gradiente com e sem o uso de mapas.

Etapa	Cenário	Tempo em relação ao total (%)
Atualizar grade	DB	2,43
	DDB	1,69
	WD	2,26
Transferência da velocidade das partículas para grade	DB	73,61
	DDB	50,70
	WD	60,22
Forças externas	DB	0,01
	DDB	0,01
	WD	0,01
Pressão	DB	12,86
	DDB	39,22
	WD	28,18
Transferência da velocidade da grade para as partículas	DB	11,07
	DDB	8,36
	WD	9,32
Mover as partículas	DB	0,02
	DDB	0,02
	WD	0,01

Comparando-se o tempo de processamento por passo de tempo das simulações com e sem o dicionário, como ilustrado na Figura 4.5, é possível notar que há melhoras de desempenho em partes da simulação, com uma média de melhora em torno de $0,1$ milissegundo para o cálculo dos pesos do Laplaciano (C2C) — cerca de 30% de melhora — e de $0,05$ milissegundo para o cálculo do gradiente (C2F) — cerca de 15% de melhora — para os cenários testados. O cenário *double dam break* apresentou as maiores diferenças nas médias no tempo de processamento entre simuladores com e sem dicionário, como tabulado na Tabela 4.2. Apesar da melhora percentual ser maior do que 15% em média, a pouca melhora de desempenho em segundos pode ser explicada pela porcentagem que o tempo de processamento da etapa do cálculo da pressão assume em relação ao tempo de processamento total para cada passo, como indicado na Tabela 4.1.

Ainda pela Figura 4.5, observa-se que no início da simulação há pouca diferença entre o uso ou não do dicionário, uma vez que o dicionário ainda está sendo populado. Ao avançar os passos, a diferença de desempenho entre as duas opções aumenta, apesar de em partes o uso do dicionário prejudicar o desempenho, uma vez que a leitura e a escrita no mapa é uma região crítica em que os processos não possuem acesso paralelo. Sendo assim, há instâncias em que o tempo do cálculo dos pesos em paralelo apresenta desempenho melhor do que o

Tabela 4.2: Diferença entre os tempos de processamento médio para os pesos do Laplaciano e gradiente com e sem o uso de mapas.

Cenário	Operador	Média (%)	Máximo (ms)	Média (ms)
DB	C2C	33,27	0,44	0,12
	C2F	21,26	0,40	0,08
DDB	C2C	51,07	0,56	0,20
	C2F	31,03	0,49	0,10
WD	C2C	30,73	0,41	0,11
	C2F	10,74	0,20	0,04

tempo de espera dos processos para acessar o dicionário.

Da Figura 4.6, que mostra a quantidade de consultas ao dicionário em cada passo, é possível obter os passos em que há o menor uso do dicionário, passos estes ilustrados na Figura 4.7. Desses quadros observa-se que o pouco uso do dicionário é devido à maior quantidade de estênceis regulares, gerados pelo fluxo de partículas e condições de refinamento que regem o refinamento da árvore, como descrito na Seção 3.3. Além disso, nota-se que esses são os passos em que há um maior número de partículas com no mínimo uma célula de distância de outra célula que contém um corpo granular, o que seria traduzido nas interfaces entre fluido e não fluido, caso células com partículas fossem consideradas células que contém fluido.

Corrobora ainda à conclusão de que o dicionário é pouco consultado apenas em passos em que há poucas configurações de estênceis que precisam de RBF-FD para encontrar os pesos a Figura 4.8, que mostra o total de estênceis regulares e estênceis RBF em todos os passos das simulações, além de ilustrar a quantidade configurações que tiveram seus pesos resolvidos com a consulta ao dicionário e a quantidade de configurações que ocorreram apenas uma vez na simulação. De todas as configurações de estênceis para os cenários testados, em média há apenas 436 configurações únicas para o Laplaciano e 150 para o gradiente. Ou seja, apesar do grande número de estênceis regulares, a implementação de um dicionário para pesos RBF-FD não torna-se inútil.

4.4 Simulação de fluidos

Um dos objetivos deste trabalho foi desenvolver um simulador de fluidos para utilizar o dicionário, e assim comparar o impacto do dicionário em simulações envolvendo fluidos e tal implementação seria baseada no Furoo. Na Seção 4.3, foi descrito os experimentos que apontam a melhora de desempenho ao utilizar um mapa de pesos e evitar recalculá-los. Contudo, encontrou-se diversas dificuldades na revisão da abordagem apresentada pelo Furoo, entre elas destacam-se as dificuldades com o refinamento, *reseeding*, condições de contorno e transferência de velocidade entre partículas e faces da grade.

Como apresentado na Seção 3.3, o refinamento inicialmente proposto para a árvore do

simulador sendo implementado seguia uma abordagem *top-down*, ou seja, partia-se de uma única célula que continha todo o domínio e subdividia-se as células de acordo com o critério de refinamento, baseado apenas no número de partículas. Porém, ao adotar a abordagem *top-down*, a classificação de células de interface, imprescindível para o total refinamento das células na interface fluido e não fluido que é condição para o funcionamento do método Furoo, deixou de ser trivial. Para contornar esse problema, uma abordagem parecida com a do Furoo foi utilizada, ilustrada na Figura 4.9. Uma grade totalmente refinada no maior nível é construída, porém essa estrutura destina-se apenas a classificar o centro de uma célula em fluido ou sólido. Em seguida, a árvore se refina e utiliza como condição, além do número de partículas, a grade construída anteriormente; as suas células subdividem-se se possui mais partículas do que o limite para refinação ou se conter o centro de uma célula de fluido e uma de suas vizinhas não.

O *reseeding*, dito essencial no Furoo, tornou-se fonte de um dilema na implementação. Na sequência de passos apresentada no Furoo, o *reseeding* seria executado após o cálculo do gradiente da pressão e antes da advecção das partículas. Porém, ao seguir essa ordem, observou-se que o *reseeding* não desempenhava seu papel, uma vez que as novas partículas geradas para manter as células com o número adequado de partículas logo seriam movidas para suas novas posições após a atualização de suas velocidades. E, se por ventura o *reseeding* executasse após a advecção das partículas, o refinamento da árvore tonaria-se obsoleto, já que o número de partículas nas células poderiam não obedecer mais à condição de refinamento.

Ademais, para cumprir com as condições de contorno (de Neumann e Dirichlet) com o ferramental disponível no simulador até então implementado, é necessário estênceis regulares nas bordas, o que também motivou a modificação no método de refinamento da árvore já descrito. Além disso, notou-se que a interface entre fluido e ar é sensível a qualquer mudança no refinamento, o que tornou o simulador implementado instável. Já na transferência de velocidades entre grade e partículas, notou-se que as vizinhanças RBF responsável pela transferência de velocidade entre partículas e faces da grade não são finitas como as configurações de estênceis para o cálculo dos operadores diferenciais, uma vez que não há critérios que limitem as posições que as partículas podem assumir. Essa falta de limitação pode gerar sistemas RBF-FD de muitas variáveis, o que acaba exigindo maior tempo de processamento para calcular os pesos, e ainda, gera um excesso de pressão na simulação, já que permite que partículas assumam posições muito próximas uma da outra e aproxima a distância entre elas de zero. No momento, a transferência de velocidades dificulta a estabilidade do simulador.

4.5 Considerações finais

Esse capítulo discutiu os cenários utilizados para testar a efetividade de um dicionário de pesos RBF-FD, além dos parâmetros utilizados para os testes, como os níveis máximo e mínimo da árvore e espaçamento inicial entre as partículas no momento de inicializar a

cena de simulação. Ademais, os experimentos realizados e seus resultados foram analisados, concluindo-se que o uso do dicionário apresenta melhora de desempenho no cálculo dos pesos RBF-FD. Discutiu-se também as dificuldades na implementação do simulador de fluido, assim como a solução adotada para alguns desses obstáculos.

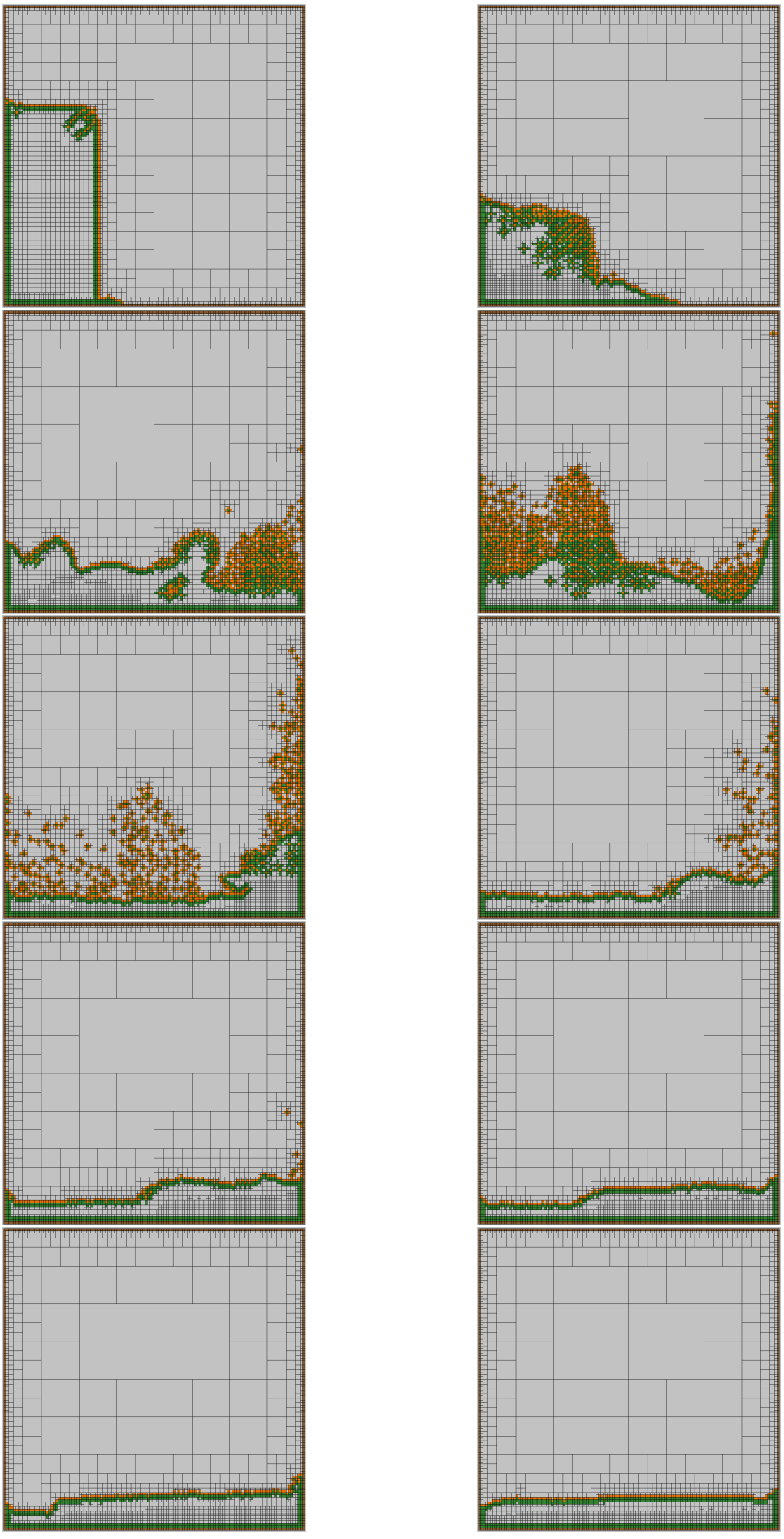


Figura 4.2: Simulação *dam break* para corpos granulares, ilustrada por 10 passos, escolhidos a cada 100 passos, iniciando no passo 100.

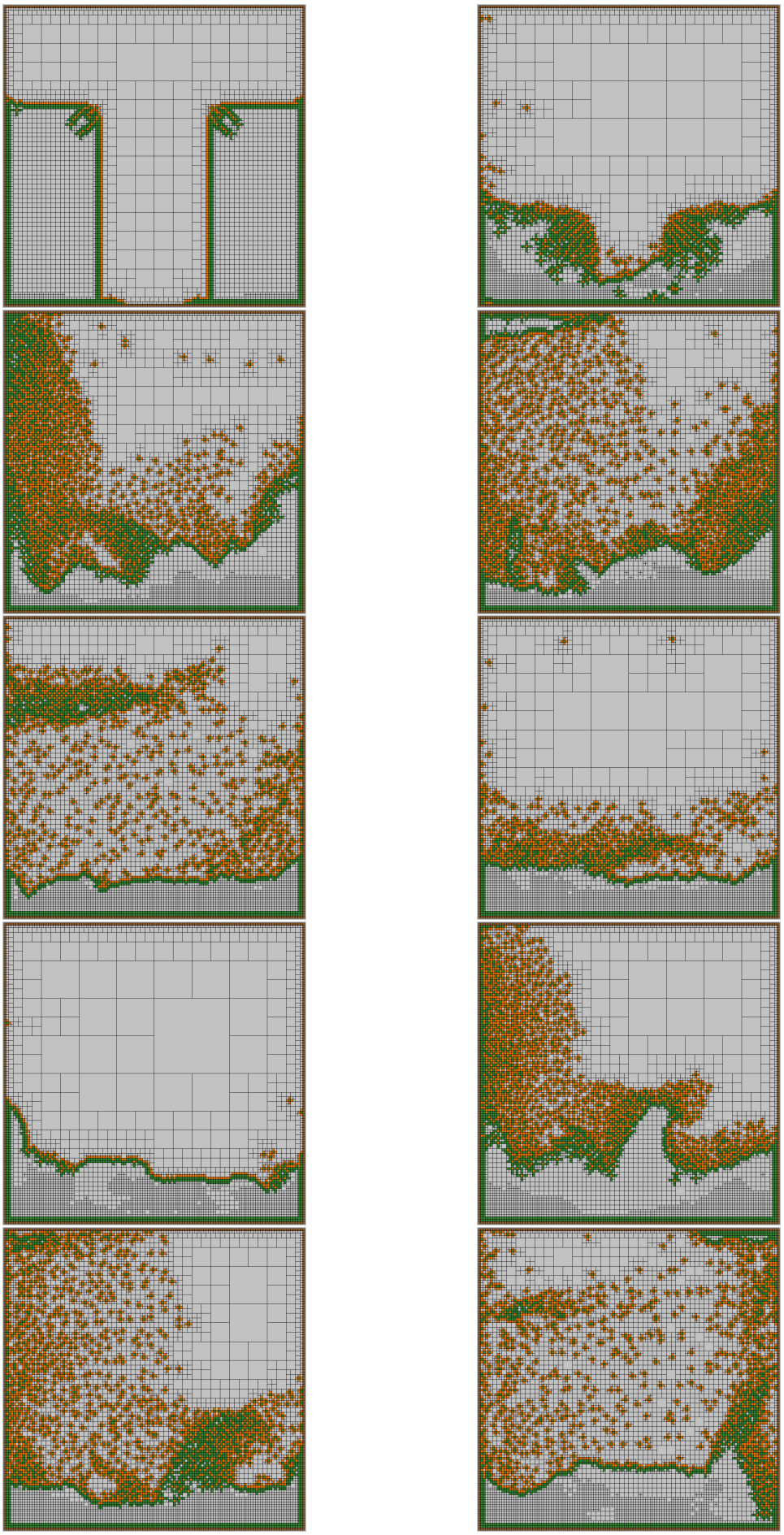


Figura 4.3: Simulação *double dam break* para corpos granulares, ilustrada por 10 passos, escolhidos a cada 100 passos, iniciando no passo 100.

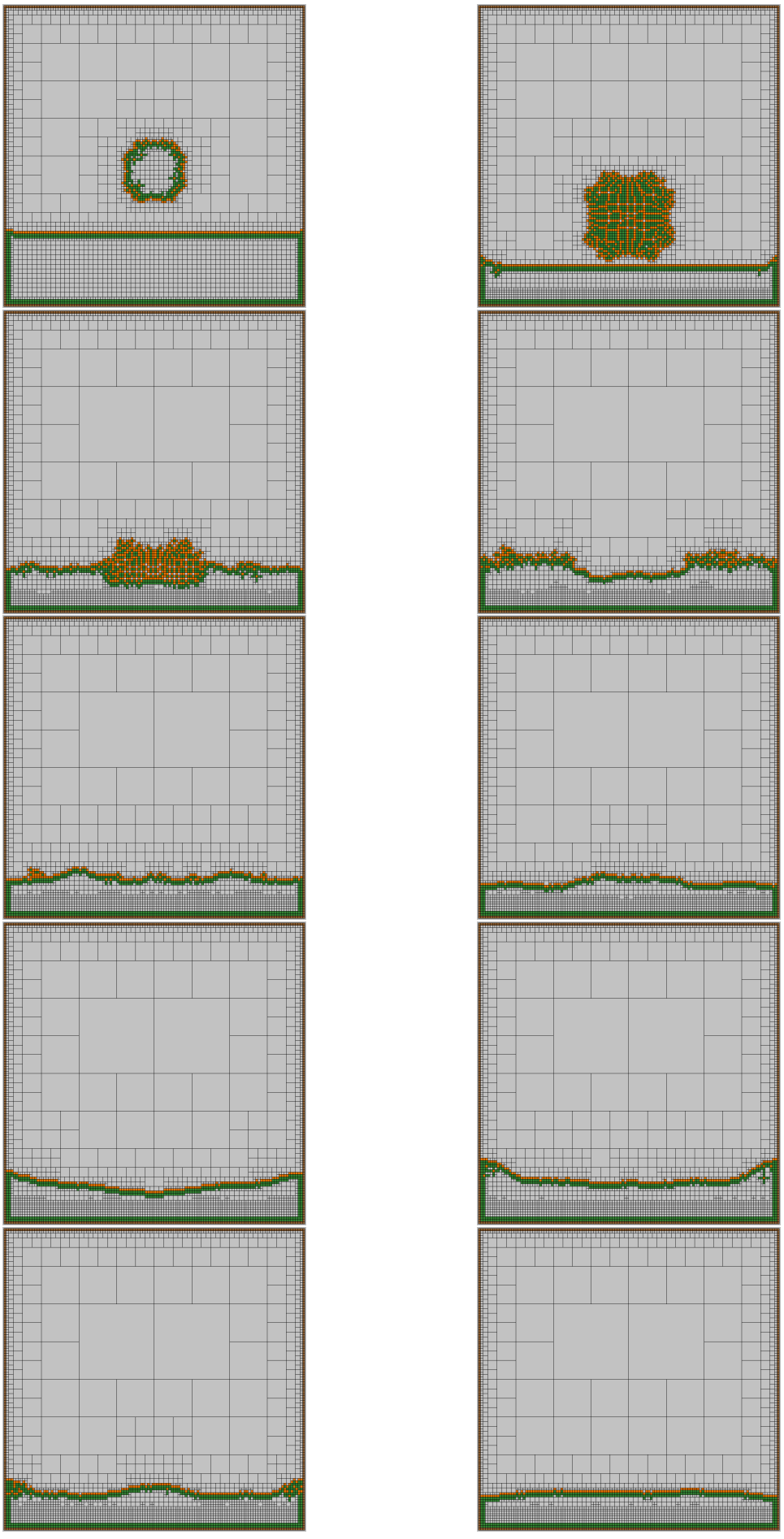


Figura 4.4: Simulação *water drop* para corpos granulares, ilustrada por 10 passos, escolhidos a cada 100 passos, iniciando no passo 100.

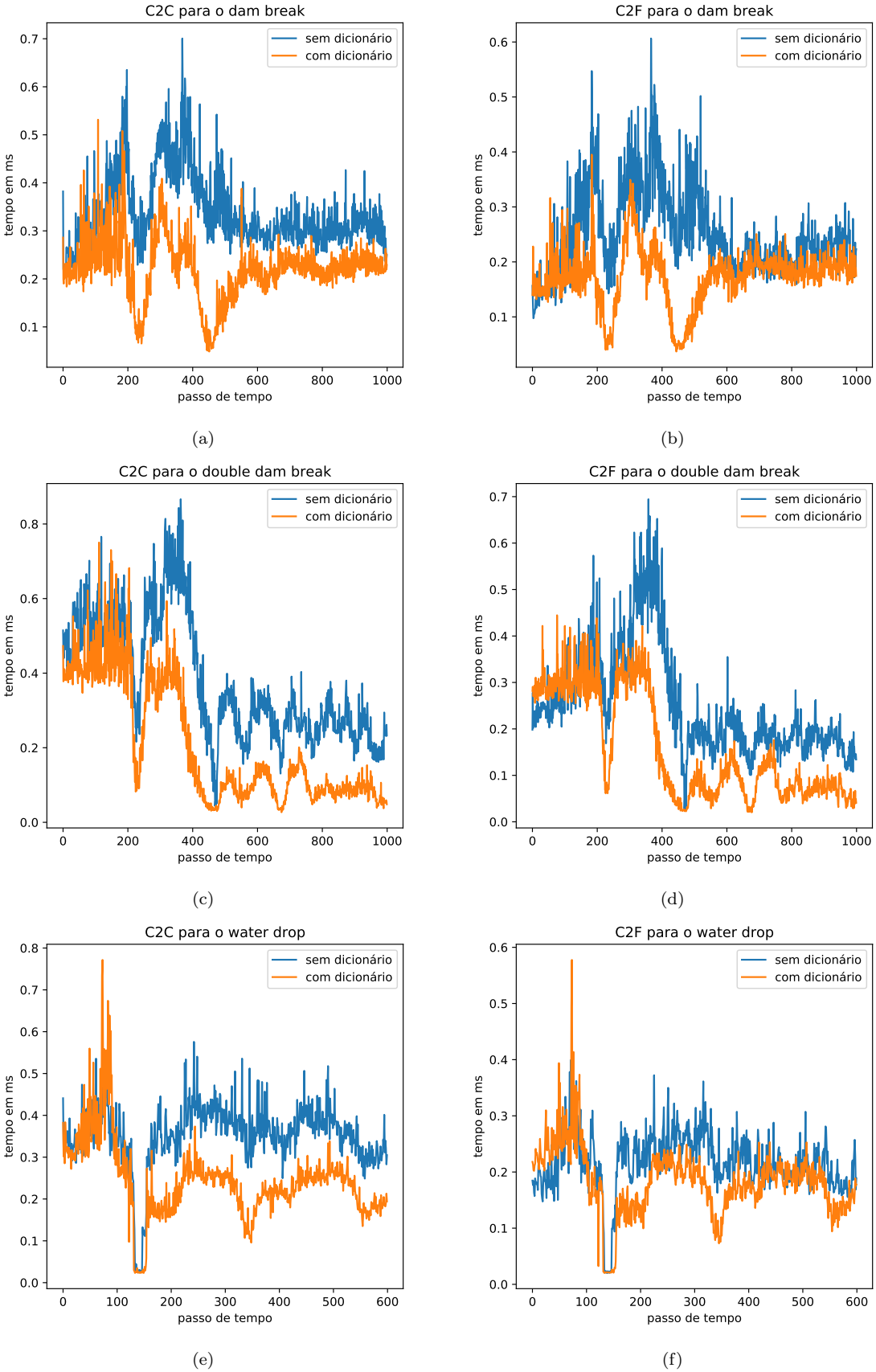
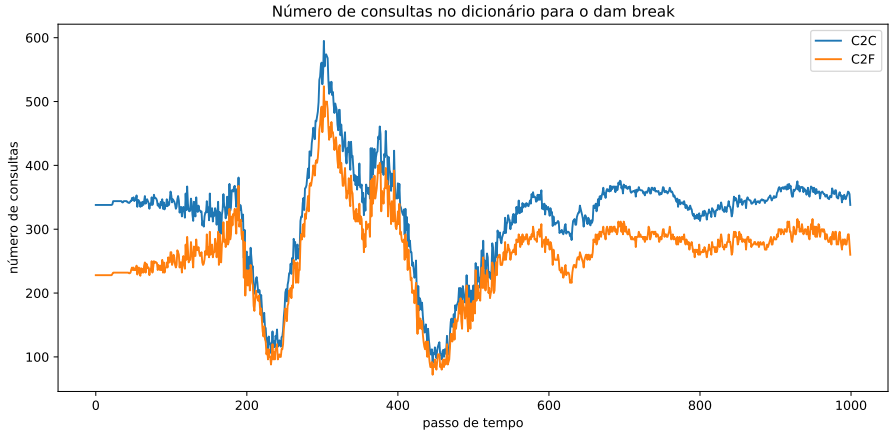
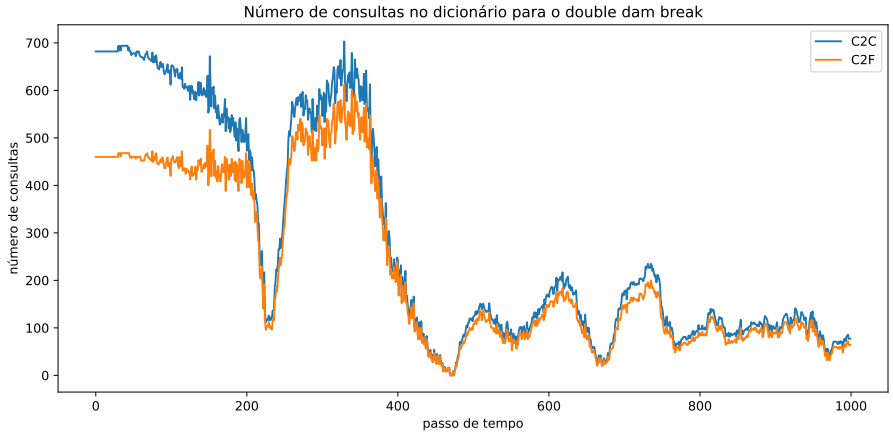


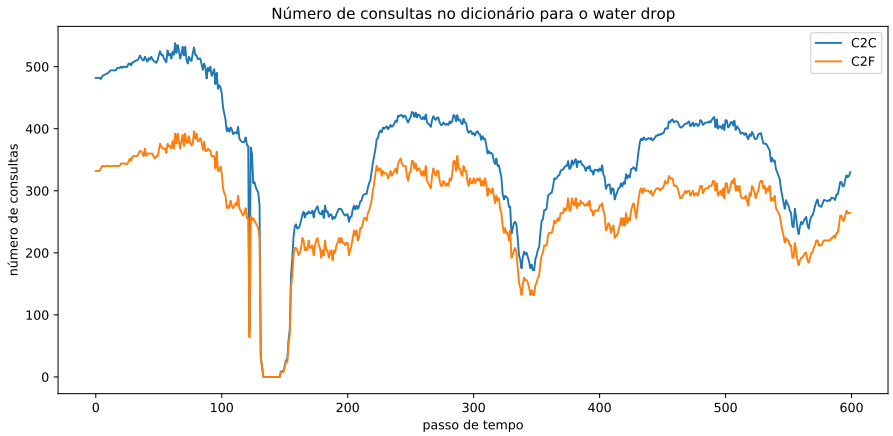
Figura 4.5: Comparação do tempo de processamento para cada passo de tempo entre simulação com e sem o dicionário.



(a)

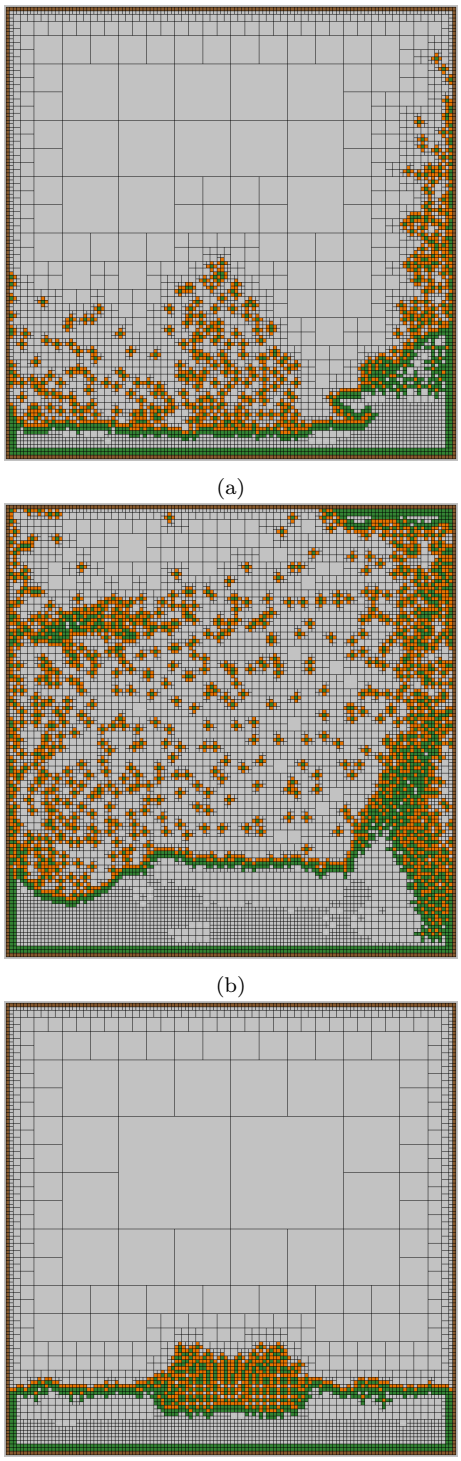


(b)



(c)

Figura 4.6: Número de consultas aos dicionários dos pesos do Laplaciano (C2C) e gradiente (C2F) em que há os pesos RBF-FD registrados para a chave consultada.

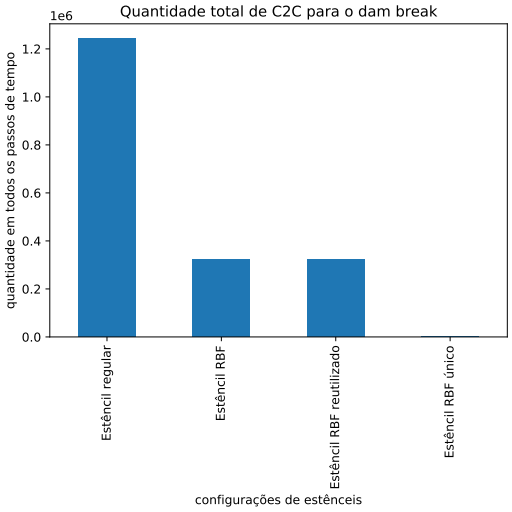


(a)

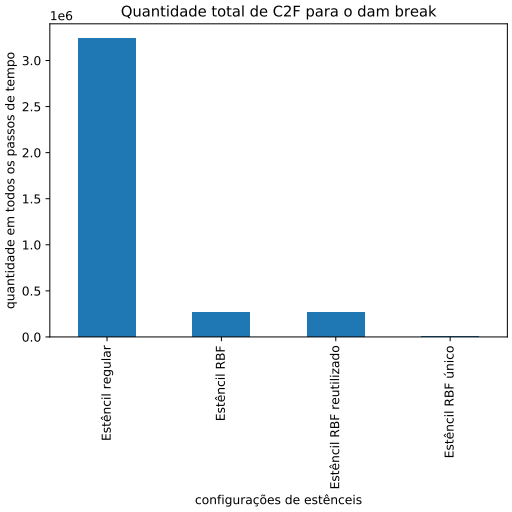
(b)

(c)

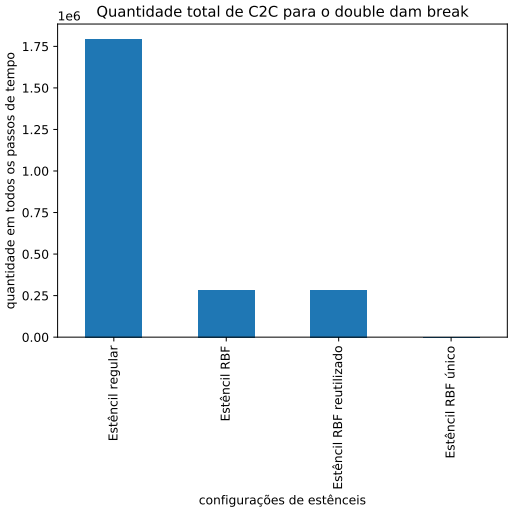
Figura 4.7: Exemplo de passos que possuem pouca consulta aos dicionários. (a) Passo 500 do *dam break* é um exemplo de passo com poucas consultas aos dicionários quando comparado com outros passos. (b) Passo 1000 do *double dam break* é um exemplo de passo com poucas consultas aos dicionários quando comparado com outros passos. (c) Passo 180 do *water drop* é um exemplo de passo com poucas consultas aos dicionários quando comparado com outros passos.



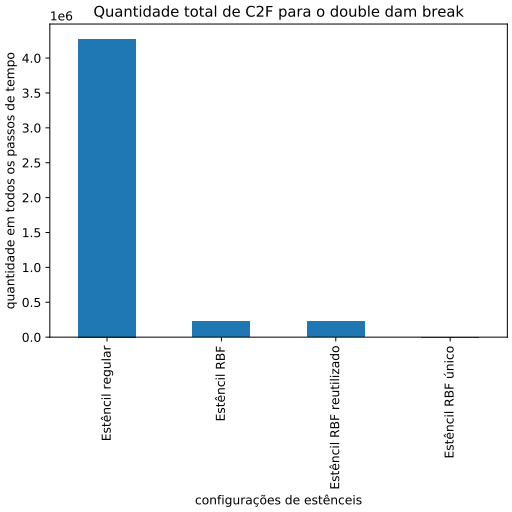
(a)



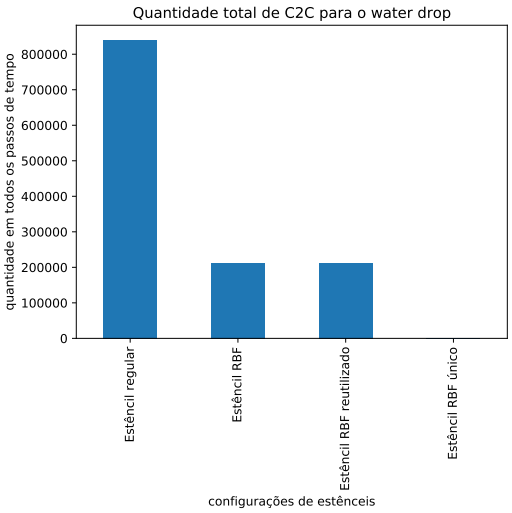
(b)



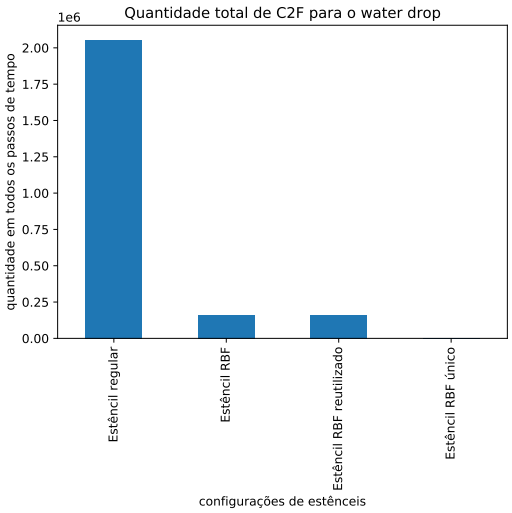
(c)



(d)

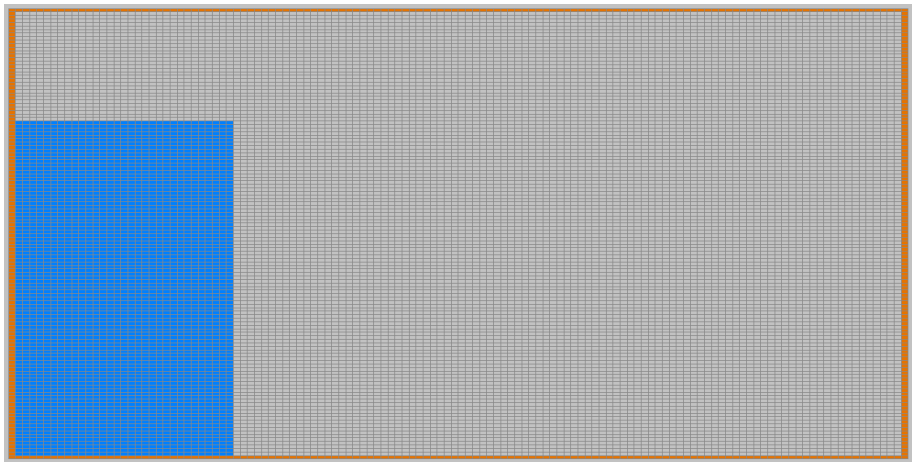


(e)

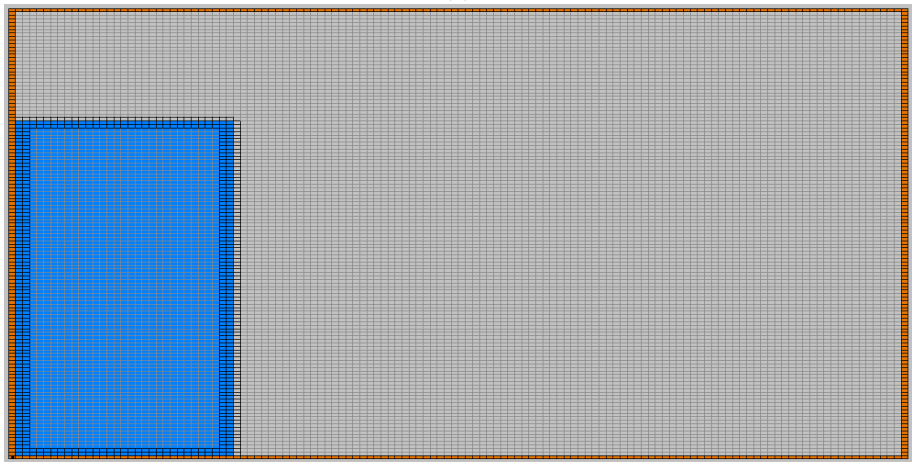


(f)

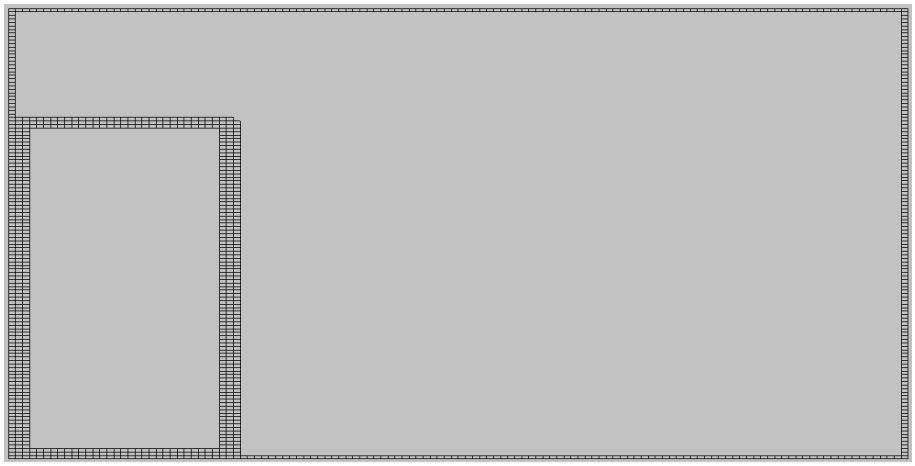
Figura 4.8: Quantidade de estênceis que ocorrem em todos os passos de tempo das simulações dividido por tipo de estêncil.



(a)



(b)



(c)

Figura 4.9: Processo de refinamento da TBGG. (a) Grade totalmente refinada pra classificar as células em fluido (células azuis) ou sólido (células laranjas). (b) Árvore construída e refinada com base na grade. (c) Árvore obtida após o refinamento.

Capítulo 5

Conclusão

5.1 Considerações sobre o trabalho desenvolvido

Neste trabalho foi proposto um simulador de fluidos baseado no PIC que utilize grades adaptativas balanceadas e RBF-FD, com base no trabalho de Nakanishi et al. [37]. Entre os objetivos do trabalho, como descritos no Capítulo 1, estavam apresentar e implementar um solucionador que utilizasse grades adaptativas balanceadas e RBF-FD, além de estudar sua performance quando comparado com outros solucionadores. Para tanto, um estudo sobre simulações de fluidos foi feito, em que as equações que regem fluidos foram analisadas, além dos operadores diferenciais que são essenciais para resolvê-las. Ademais, o estudo também englobou três diferentes métodos que existem para representar e simular fluidos, sendo eles, simulações baseadas em partículas, baseadas em grades regulares e métodos híbridos. Os métodos híbridos foram explorados com mais detalhes, uma vez que o PIC é um método híbrido, passando pelos passos que o solucionador toma para resolver as equações de Navier-Stokes. Esses estudos foram caracterizados no Capítulo 2.

Ainda nesse capítulo, introduziu-se os conceitos de grades adaptativas e RBF-FD, utilizados no simulador proposto e também no trabalho de Nakanishi et al. [37]. Uma explicação dos sistemas de equações utilizados para obter-se os pesos da interpolação e operadores diferenciais em configurações de estênceis não regulares, como são gerados em uma grade adaptativa, foi feita, além de especificar como a vizinhança para criar os estênceis foram determinadas para a transferência de velocidade da grade para partículas e vice-versa. As regras para obter a vizinhança para os operadores diferenciais que são usados nas equações de Navier-Stokes também foram descritas.

Já no Capítulo 3, o solucionador proposto neste trabalho foi detalhado, assim como os passos tomados para chegar nele. Em específico, foram discutidos os passo do novo *pipeline* e esses foram comparados com os passos tanto do PIC quanto do Furoo. A estrutura da árvore que foi implementada para suportar uma grade adaptativa balanceada também foi descrita,

dando ênfase na implementação das células e faces das células da grade. Em seguida, a implementação das classes responsáveis pelas diferenças finitas que calculam os operadores diferenciais é comentada, juntamente com as deduções matemáticas que foram desenvolvidas. Além disso, as classes que representam as configurações de estênceis que são geradas na árvore também são citadas, descrevendo os seus principais métodos, como o método responsável por calcular os pesos RBF-FD. Por fim, ainda no Capítulo 3, discute-se o dicionário simbólico (primeira proposta de dicionário neste trabalho) e o dicionário iterativo (dicionário efetivamente utilizado pelo solucionador).

Por fim, no Capítulo 4 é discutido os resultados obtidos pelo solucionador, com e sem o uso do dicionário. Os cenários utilizados para teste — *dam break*, *double dam break* e *water drop* — e seus parâmetros — domínio da simulação, espaçamento inicial entre partículas, níveis máximo e mínimo da árvore e critério de refinamento da árvore — foram descritos. A diferença de número de partículas e folhas da árvore após o *reseeding* inicial causado pelo critério de refinamento também é exposto. As comparações de desempenho do solucionador com e sem o dicionário foram apresentadas, das quais conclui-se que o dicionário acelera o cálculo dos pesos RBF-FD, além de ser consultado em mais de 90% dos casos. Porém, a impossibilidade de ler e escrever no mapa de forma paralela pode igualar o desempenho do simulador com dicionário ao sem dicionário em partes da simulação. Pode-se concluir também que, como o dicionário é pouco consultado quando há a ocorrência de mais estênceis regulares, o cenário em que o dicionário obteria maior desempenho em relação à simulação sem o uso dele é em cenários de simulações mais longas do que as testadas neste trabalho e em que há baixa ocorrência de estênceis regulares. Além disso, expôs-se as adversidades encaradas na implementação do simulador de fluidos, como foi o caso do refinamento, *reseeding*, condições de contorno e transferência de velocidade entre partículas e faces das células.

5.2 Limitações e dificuldades

As etapas da simulação que mais demandam tempo são as transferências da velocidade entre partículas e faces, como discutido na Seção 4.4, uma vez que não foi possível criar condições para que as configurações de estênceis se repetissem e pudessem ser reutilizadas na forma de um dicionário para evitar recalcular pesos. Além disso, apenas os operadores Laplaciano da pressão e gradiente da pressão utilizam o dicionário, uma vez que uma solução para classificar os estênceis do divergente da velocidade não foi encontrada. Quanto à estrutura da árvore, esta poderia ser melhorada ao tornar possível refinar ou unir células de uma árvore já existente para torná-la balanceada, uma vez que neste trabalho a árvore é reconstruída a cada passo de tempo.

Além dos cálculos com RBF-FD, este trabalho tomou como base o *pipeline* implementado no Furoo, e o *reseeding*, presente nesse, foi fonte de dilemas no solucionador proposto neste trabalho. No Furoo, o *reseeding* é feito em todo passo de tempo antes da transferência da

velocidade das partículas para a grade, porém, se o mesmo for feito no solucionador proposto, as partículas irão sofrer o *reseeding* depois de já refinada a árvore, ou seja, o refinamento irá se tornar inválido. Além das faces da grade ainda não possuírem as velocidades necessárias para determinar a velocidade da nova partícula gerada.

O refinamento da árvore seguindo os critérios propostos no Furoo mostrou-se desafiador, uma vez que em uma abordagem *top-down* não é trivial classificar uma célula em célula de interface para refiná-la até o nível máximo da árvore. As condições de contorno, que criam requisitos diferentes para a determinação de configuração de estênceis no contorno do fluido e a sensibilidade dos resultados na interface entre ar e fluido gerada por qualquer diferença no refinamento, também mostraram-se impeditivas. Por fim, a transferência da velocidade entre grade e partículas adicionou dois problemas ao simulador, sendo eles o tamanho da vizinhança gerada para montar os sistemas RBF-FD e o excesso de pressão adicionado à simulação pela proximidade das partículas.

5.3 Trabalhos futuros

Há métodos para acelerar os cálculos da simulação de fluidos que não foram exploradas neste trabalho, dado o tempo limitado para a execução do projeto. Uma das possíveis propostas é a introdução do solucionador proposto em ambientes paralelos heterogêneos, ou seja, adaptar o solucionador para que ele distribua a carga de trabalho entre as unidades de processamento (CPU e GPU) disponíveis. Neste caso, uma nova representação da *quad-tree*/*octree* para GPU deve ser proposta, de forma a mantê-la coerente entre as unidades de processamento. Além disso, deve-se estudar como particionar o *pipeline* da simulação para a distribuição da carga de trabalho.

Ademais, testar um simulador sem a presença de partículas, que por consequência elimina as transferências de velocidade entre grade e partículas (passo que mais ocupa tempo de processamento no simulador atual) também é uma opção. Estudar o uso de outros tipos de grade — como a grade colocada — e o uso de outras funções de base radial para o cálculo dos pesos RBF-FD também são possibilidades. Ainda, métodos para gerar versões mais compactas das configurações de estênceis (principalmente das configurações da transferência de velocidade) podem acelerar o método ao diminuir o tamanho do sistema linear a ser montado e resolvido.

Referências Bibliográficas

[1] Ando, R., and Batty, C. A practical octree liquid simulator with adaptive surface resolution. *ACM Trans. Graph. 39*, 4 (July 2020).

[2] Ando, R., Thürey, N., and Wojtan, C. Highly adaptive liquid simulations on tetrahedral meshes. *ACM Trans. Graph. 32*, 4 (2013), 103:1–103:10.

[3] Barnett, G. A., Flyer, N., and Wicker, L. J. An rbf-fd polynomial method based on polyharmonic splines for the navier-stokes equations: Comparisons on different node layouts, 2015.

[4] Batty, C., Xenos, S., and Houston, B. Tetrahedral Embedded Boundary Methods for Accurate and Flexible Adaptive Fluids. *Computer Graphics Forum 29*, 2 (2010), 695–704.

[5] Bayona, V., Flyer, N., Fornberg, B., and Barnett, G. A. On the role of polynomials in RBF-FD approximations: Numerical solution of elliptic PDEs. *J. Comput. Phys. 332* (2017), 257–273.

[6] Boyd, L., and Bridson, R. Multiflip for energetic two-phase fluid simulation. *ACM Trans. Graph. 31*, 2 (Apr. 2012).

[7] Brackbill, J. U., and Ruppel, H. M. FLIP: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions. *J. Comput. Phys. 65*, 2 (1986), 314–343.

[8] Bridson, R. *Fluid Simulation*, 2nd ed. A. K. Peters, Natick, MA, 2015.

[9] Chentanez, N., and Müller, M. Real-time eulerian water simulation using a restricted tall cell grid. *ACM Trans. Graph. 30*, 4 (July 2011).

[10] Chinchapatnam, P. P., Djidjeli, K., Nair, P. B., and Tan, M. A compact rbf-fd based meshless method for the incompressible navier—stokes equations. *Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment 223*, 3 (2009), 275–290.

[11] Edwards, E., and Bridson, R. Detailed water with coarse grids: Combining surface meshes and adaptive discontinuous galerkin. *ACM Trans. Graph. 33*, 4 (July 2014).

[12] Elliott, S., Kumar, R. R. P., Flyer, N., Ta, T., and Loft, R. Implementation of a scalable, performance portable shallow water equation solver using radial basis function-generated finite difference methods. *Int. J. High Perform. Comput. Appl.* (2018).

[13] Enright, D., Fedkiw, R., Ferziger, J., and Mitchell, I. A hybrid particle level set method for improved interface capturing. *Journal of Computational Physics 183*, 1 (2002), 83–116.

[14] Ferstl, F., Ando, R., Wojtan, C., Westermann, R., and Thuerey, N. Narrow Band FLIP for Liquid Simulations. *Computer Graphics Forum 35*, 2 (2016), 225–232.

[15] Ferstl, F., Ando, R., Wojtan, C., Westermann, R., and Thuerey, N. Narrow band FLIP for liquid simulations. *Computer Graphics Forum (Proc. Eurographics) 35*, 2 (2016), 225–232.

[16] Flyer, N., Barnett, G. A., and Wicker, L. J. Enhancing finite differences with radial basis functions: Experiments on the Navier-Stokes equations. *J. Comput. Phys. 316* (2016), 39–62.

[17] Flyer, N., Fornberg, B., Bayona, V., and Barnett, G. A. On the role of polynomials in RBF-FD approximations: Interpolation and accuracy. *J. Comput. Phys. 321* (2016), 21–38.

[18] Flyer, N., Lehto, E., Blaise, S., Wright, G. B., and St-Cyr, A. A guide to rbf-generated finite differences for nonlinear transport: Shallow water simulations on a sphere. *Journal of Computational Physics 231*, 11 (2012), 4078–4095.

[19] Fornberg, B., and Flyer, N. Solving {PDEs} with radial basis functions. *Acta Numerica 24* (2015), 215–258.

[20] Fu, C., Guo, Q., Gast, T., Jiang, C., and Teran, J. A polynomial particle-in-cell method. *ACM Trans. Graph. 36*, 6 (Nov. 2017).

[21] Goldade, R., Wang, Y., Aanjaneya, M., and Batty, C. An adaptive variational finite difference framework for efficient symmetric octree viscosity. *ACM Trans. Graph. 38*, 4 (July 2019).

[22] Guennebaud, G., Jacob, B., and Others. Eigen v3. http://eigen.tuxfamily.org, 2010.

[23] Harlow, F. The particle-in-cell method for numerical solution of problems in fluid dynamics. Tech. rep., Los Alamos National Laboratory (LANL), Los Alamos, NM, 1962.

[24] Harlow, F., and Welch, J. E. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Physics of Fluids 8* (1965), 2182–2189.

[25] Ihmsen, M., Orthmann, J., Solenthaler, B., Kolb, A., and Teschner, M. SPH Fluids in Computer Graphics. In *Eurographics 2014 - State of the Art Reports* (2014).

[26] Jiang, C., Schroeder, C., Selle, A., Teran, J., and Stomakhin, A. The affine particle-in-cell method. *ACM Trans. Graph. 34*, 4 (July 2015).

[27] Kim, B. Multi-phase fluid simulations using regional level sets. In *ACM SIGGRAPH Asia 2010 Papers* (New York, NY, USA, 2010), SIGGRAPH ASIA '10, Association for Computing Machinery.

[28] Kim, D. *Fluid Engine Development.* CRC Press, Boca Raton, FL, 2016.

[29] Lentine, M., Zheng, W., and Fedkiw, R. A novel algorithm for incompressible flow using only a coarse grid projection. *ACM Trans. Graph. 29*, 4 (July 2010).

[30] LeVeque, R. *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems.* Society for Industrial and Applied Mathematics, 2007.

[31] Liu, M. B., and Liu, G. R. Smoothed Particle Hydrodynamics (SPH): an Overview and Recent Developments. *Arch. Comput. Methods Eng. 17*, 1 (2010), 25–76.

[32] Liu, S., Ban, X., Wang, B., and Wang, X. A symmetric particle-based simulation scheme towards large scale diffuse fluids. *Symmetry 10* (03 2018), 86.

[33] Losasso, F., Gibou, F., and Fedkiw, R. Simulating water and smoke with an octree data structure. *ACM Trans. Graph. 23*, 3 (2004), 457–462.

[34] Malvern, L. *Introduction to the Mechanics of a Continuous Medium.* Prentice-Hall, 1969.

[35] Mishra, B. A review of computer simulation of tumbling mills by the discrete element method: Part i—contact mechanics. *International Journal of Mineral Processing 71*, 1 (2003), 73–93.

[36] Monaghan, J. J. Smoothed particle hydrodynamics. *Annual Review of Astronomy and Astrophysics 30*, 1 (1992), 543–574.

[37] Nakanishi, R., Nascimento, F., Campos, R., Pagliosa, P., and Paiva, A. Rbf liquids: An adaptive pic solver using rbf-fd. *ACM Trans. Graph. 39*, 6 (Nov. 2020).

[38] Sato, T., Wojtan, C., Thuerey, N., Igarashi, T., and Ando, R. Extended Narrow Band FLIP for Liquid Simulations. *Computer Graphics Forum 37*, 2 (2018), 169–177.

[39] Shahane, S., Radhakrishnan, A., and Vanka, S. P. A high-order accurate meshless method for solution of incompressible fluid flow problems, 2020.

[40] Solenthaler, B., and Pajarola, R. Predictive-corrective incompressible sph. In *ACM SIGGRAPH 2009 Papers* (New York, NY, USA, 2009), SIGGRAPH '09, Association for Computing Machinery.

[41] Stam, J. Stable fluids. In *Proc. of SIGGRAPH '99* (1999), ACM, pp. 121–128.

[42] Tan, J., and Yang, X. Physically-based fluid animation: A survey. *Science in China Series F: Information Sciences 52* (05 2009), 723–740.

[43] teschner, m., cornelis, j., peer, a., and ihmsen, m. Iisph-flip for incompressible fluids. *Computer Graphics Forum* (2014).

[44] Xi, R., Luo, Z., Feng, D. D., Zhang, Y., Zhang, X., and Han, T. Survey on smoothed particle hydrodynamics and the particle systems. *IEEE Access 8* (2020), 3087–3105.

[45] Zhu, Y., and Bridson, R. Animating sand as a fluid. *ACM Trans. Graph. 24*, 3 (July 2005), 965–972.

[46] Zienkiewicz, O., Talor, R., and Zhu, J. *The Finite Element Method: Its Basis and Fundamentals*, seventh ed. Elsevier, 2013.