Giovanni Belloni Fernandes Braga

Desenvolvimento de Software de Mecânica dos Fluidos utilizando a técnica Smoothed Particle Hydrodynamics

Macaé

2018

Giovanni Belloni Fernandes Braga

Desenvolvimento de Software de Mecânica dos Fluidos utilizando a técnica Smoothed Particle Hydrodynamics

Trabalho de Conclusão de Curso apresentado à Universidade Federal do Rio de Janeiro - Campus Macaé - como parte dos requisitos para a obtenção do título de Graduado em Engenharia Mecânica.

Universidade Federal do Rio de Janeiro – UFRJ - Campus Macaé

Orientador: Bernardo Mattos Tavares

Macaé

2018

CIP - Catalogação na Publicação

B813d

Braga, Giovanni
Desenvolvimento de software de mecânica dos fluidos utilizando a técnica smoothed particle hydrodynamics / Giovanni Braga. -- Macaé, 2018.
112 f.

Orientador: Bernardo Tavares.
Trabalho de conclusão de curso (graduação) - Universidade Federal do Rio de Janeiro, Colegiado de Ensino de Graduação - Macaé, Bacharel em Engenharia Mecânica, 2018.

1. SPH. 2. C++. 3. Hidrodinâmica. I. Tavares, Bernardo, orient. II. Título.

CDD 532.5

Ficha catalográfica elaborada pela Biblioteca com os
dados fornecidos pelo(a) autor(a)
Campus UFRJ-Macaé Professor Aloisio Teixeira
Bibliotecária Rosana Silva de Oliveira CRB7 – 5849

Giovanni Belloni Fernandes Braga

Desenvolvimento de Software de Mecânica dos Fluidos utilizando a técnica Smoothed Particle Hydrodynamics

Trabalho de Conclusão de Curso apresentado à Universidade Federal do Rio de Janeiro - Campus Macaé - como parte dos requisitos para a obtenção do título de Graduado em Engenharia Mecânica.

Trabalho aprovado. Macaé, 07 de junho de 2018:

Bernardo Mattos Tavares
Orientador

Carlos Eduardo Magalhães de Aguiar
Avaliador 1

Lucas Lisbôa Vignoli
Avaliador 2

Macaé
2018

Resumo

Desenvolvido em 1977, por duas linhas de pesquisa paralelas para a resolução de problemas astrofísicos tridimensionais de elevada assimetria, o SPH é um método numérico de abordagem lagrangeana, de partículas e sem malha. A versatilidade do método despertou um intenso interesse da comunidade científica e possibilitou uma rápida extensão de sua aplicação para diversas áreas da ciência e da indústria. A natureza do SPH é representar o fluido por um conjunto de partículas, as quais carregam informações sobre as propriedades físicas e termodinâmicas. As aproximações dos campos de interesse se dão por meio da representação integral de uma função através da convolução da mesma com uma função de suavização que em seguida é discretizada para a implementação computacional. Isto permite que as equações diferenciais parciais sejam reduzidos a simples sistemas equações ordinárias que podem ser resolvidos através de métodos de integração de segunda ordem já consolidados na literatura. Neste trabalho foi desenvolvido um código em C++ para a aplicação do método SPH em 2D a problemas simples de hidrodinâmica de fluidos viscosos sob o efeito do campo gravitacional. E a partir dos dados obtidos foram feitas animações para visualização dos resultados e então disponibilizado o acesso remoto através de códigos QR. O ambiente utilizado para programação foi o Geany, para o tratamento de dados o Gnuplot e para as as animações o GIMP, sendo todos softwares livres. As decisões tomadas acerca da formulação empregada e as rotinas pertinentes do algoritmo foram detalhadas. Foi feito um apanhado do formalismo básico do método e exposto em detalhes, incluindo demonstrações e digressões de tópicos pertinentes para a compreensão da essência do SPH.

Palavras-chaves: SPH. C++. Hidrodinâmica.

Abstract

Developed simultaneously by two research lines in 1977 to solve astrophysical problems of high asymmetry on a three dimensional space, the SPH is a meshfree particle method that makes use of the Lagrangian approach. Its adaptability aroused an intense interest from the scientific community and made possible a quick extension of its applications to an wide set of industrial and science fields. The SPH nature is to represent the fluid as a set of particles, and each of them carry information about physical and thermodynamic properties. The fields of interest are approximated by the concept of integral representation, wich takes the convolutional integral of that field with a smoothing function. Then it is discretized for implementation purposes. This allows partial differential equations to be reduced into a simple set of ordinary differential equations, which can be solved through the application of any well known integration method. In this work a C++ code was developed to apply the SPH method on a two dimensional space to simple hydrodynamics problems with viscous fluids under the effect of gravitational field. Some animations was created from the obtained data to better visualize the results and then released for remote access through QR codes. The programming environment used was the Geany, for data processing the Gnuplot and for animation's creation the GIMP, being all of then free softwares. The taken decisions beyond the employed formulation and pertinent algorithm routines were detailed. The basic method formalism was fully approached and exposed in details, including some demonstrations and digressions about some important topics for the comprehension of the SPH's essence.

Key-words: SPH. C++. Hydrodynamics.

Sumário

1 Introdução

A simulação numérica assumiu ao longo das últimas décadas um papel de extrema importância nas pesquisas sobre os fenômenos naturais mais variados, o que se deve inicialmente ao crescente avanço da computação neste mesmo período. O estudo dos métodos numéricos deu forma a uma nova ferramenta de investigação científica que supre as carências deixadas tanto pela abordagem experimental quanto pela teórica. Os métodos experimentais em geral exigem um investimento alto de dinheiro e tempo, pois requerem o controle adequado de diversos parâmetros para a representação do fenômeno e medições precisas das grandezas envolvidas, além de por vezes resultar em situações que oferecem riscos aos operadores. Já o estudo teórico é limitado pela falta de ferramentas matemáticas para a solução das equações que descrevem os fenômenos em situações mais gerais, sendo possível encontrar soluções apenas para casos bastante simplificados. Portanto, os métodos numéricos se destacam pela capacidade de atacar o problema diretamente, sem fazer muitas simplificações, de uma forma segura, cada vez mais precisa, e consideravelmente mais barata.

1.1 Métodos Tradicionais

Os primeiros métodos, e muitos outros que vieram em seguida, foram fundamentados a partir da ideia de se discretizar o domínio do problema por meio da construção de uma malha computacional. Esta malha possuiria diferentes propriedades dependendo da escolha de diversos fatores como o referencial a ser adotado pelo método, o qual poderia ser Euleriano ou Lagrangiano.

O referencial Euleriano é fixo no espaço, onde a evolução temporal das grandezas físicas é observada para cada ponto do domínio do problema. O que leva a representação destas grandezas na forma de campos, vetoriais ou escalares, que são funções das coordenadas espaciais e do tempo. Consequentemente, a malha computacional é composta por células estáticas e deverá se estender por todo espaço de interesse. Para se realizar o cálculo da variação temporal das propriedades do objeto por meio deste referencial é necessário que se contabilize a ocorrência de dois fenômenos. Um é a variação no tempo da propriedade em um ponto fixo do espaço, chamada de derivada local. Já o outro, é a variação que ocorre pelo deslocamento de um ponto no espaço para um outro local aonde a propriedade em questão possui valor diferente, chamado de derivada convectiva. Se torna então evidente que um dos pontos chave desta abordagem é a representação dos fluxos de massa, momento e energia que se dá pelas fronteiras das células, e cada uma delas pode ser modelada como um volume de controle (sistema aberto) infinitesimal. De forma

a exemplificar, considere as equações abaixo.

$$\frac{\partial \rho}{\partial t} + \vec{V}.\nabla\rho = -\rho.\nabla.\vec{V} \tag{1.1}$$

e,

$$\frac{\partial V_\xi}{\partial t} + \vec{V}.\nabla V_\xi = -\frac{1}{\rho}\frac{\partial p}{\partial \xi}. \tag{1.2}$$

Onde $\vec{V} = u\hat{i} + v\hat{j} + w\hat{k}$, $V_\xi = \vec{V}.\hat{\xi}$ e $\xi = \{x, y, z\}$.

A primeira é a expressão para a conservação da massa e a segunda é a expressão para a conservação do momento linear para um eixo qualquer, ambas na forma diferencial. Dessa forma a contribuição do termo local, $\frac{\partial}{\partial t}$, e do termo convectivo, $\vec{V}.\nabla$, se tornam explícitas. As demonstrações podem ser encontradas com algum detalhe em (8), onde é considerado o caso do escoamento de uma massa fluida qualquer. É pertinente observar que na equação 1.2 foram omitidos os termos referentes a viscosidade e forças de corpo apenas por não serem necessários neste momento. Além disso, a malha por ser fixa no espaço não tem sua estrutura afetada pela deformação dos objetos simulados. Logo, sua forma e volume não se alteram ao longo do tempo. Esta característica torna os métodos Eulerianos bastante apropriados para a simulação de fenômenos hidrodinâmicos, o mais conhecido é o Método das Diferenças Finitas (MDF). A imagem abaixo (Figura 1) exemplifica uma malha Euleriana.

Por outro lado, o referencial Lagrangiano é aquele que acompanha o movimento do material estudado. Com isso as propriedades físicas não são mais definidas em pontos fixos do espaço e sim nos pontos do objeto. Dessa forma, a malha Lagrangiana é criada sobre o material apenas, o que reduz o custo computacional. Além disso, o cálculo da derivada temporal das propriedades é mais simples, pois não há um termo convectivo. A equivalência das duas formas de cálculo pode ser mostrada considerando um elemento que se move de um ponto a outro dentro de um campo da propriedade genérica $q = q(r(t), t)$, onde $r(t) = (x(t), y(t), z(t))$. A variação no tempo desta propriedade para o elemento considerado será dada por.

$$\begin{aligned} \frac{dq}{dt} &= \frac{\partial q}{\partial t} + \sum_{\xi=\{x,y,z\}} \frac{dq}{d\xi}\frac{d\xi}{dt} \\ &= \frac{\partial q}{\partial t} + \vec{V}.\nabla q \end{aligned} \tag{1.3}$$

Note que o lado esquerdo da equação (1.3) é a derivada que acompanha o elemento considerado, derivada total ou material, que também é comumente denotado por $\frac{D}{Dt}$. E o lado direito é a soma da derivada local com a convectiva, que como já visto, é o referencial Euleriano. Além disso, no referencial Lagrangiano o tratamento de geometrias complexas é

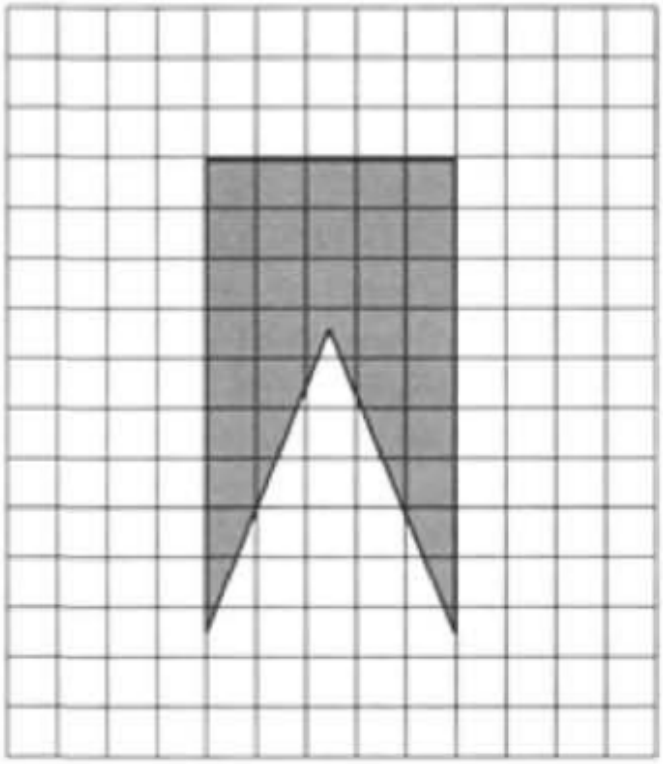

Figura 1 – Exemplo de malha Euleriana. Fonte: (11).

feito através da construção de malhas convenientemente irregulares. No entanto, como cada nó segue o caminho do ponto do material no qual está fixado, poderá haver deformação da malha devido a esse movimento relativo entre os nós. O principal exemplo desta categoria é o Método dos Elementos Finitos (MEF) e é melhor aplicado a problemas como os da mecânica dos solos, onde as deformações envolvidas em geral são pequenas. A figura 2 exemplifica uma malha Lagrangiana.

Esta abordagem apresenta inúmeras vantagens sobre os métodos Eulerianos. O código se torna mais simples devido a ausência do termo convectivo nas derivadas, requer menor poder de processamento e geometrias complexas podem ser representadas convenientemente através de malhas irregulares.

1.1.1 Limitações dos Métodos Tradicionais

Os métodos baseados em malha proporcionaram um avanço gigantesco no entendimento de diversas áreas da ciência e seu sucesso é indiscutível, porém ainda sofrem com limitações que são inerentes a sua concepção. Uma questão comum às duas abordagens, embora se diferenciem em detalhes, é a do tratamento e criação da malha. Com isso fica evidente que a própria essência desses métodos já os limita.

Com relação aos de abordagem Euleriana, a geração de uma malha regular sobre geometrias complexas requer extensos cálculos adicionais aos da resolução do problema

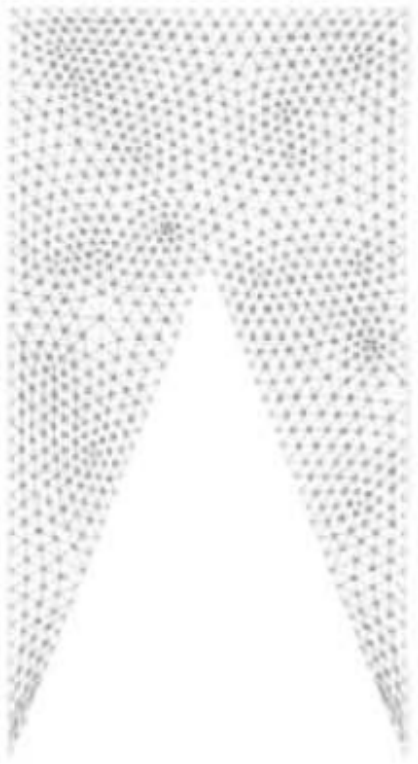

Figura 2 – Exemplo de malha Lagrangiana. Fonte: (11).

físico em si, o que resulta em um custo computacional elevado. Além disso, a necessidade de discretizar todo o domínio onde possivelmente o material se deslocará também recai sobre o mesmo problema. Como consequência, são utilizadas malhas com refinamento baixo que depreciam a qualidade do resultado final. Ainda é possível apontar outras dificuldades provenientes do fato da malha Euleriana ser fixa no espaço como a de analisar variáveis em ponto do material ao longo do tempo e localizar precisamente superfícies livres, interfaces e fronteiras deformáveis. Já a malha Lagrangiana, por ser fixada ao objeto simulado, é proporcionalmente afetada pelas eventuais distorções do mesmo e com isso a precisão é reduzida. Uma maneira de se remediar este problema é criar uma nova malha, sem distorções, por cima da antiga e interpolar as propriedades para as novas células. Entretanto, este processo requer mais tempo computacional e com o aumento do número de vezes em que este algoritmo é repetido dentro de uma simulação esta cada vez mais adquire um caráter Euleriano em sua essência.

Por fim, é válido ressaltar que há uma interseção entre as qualidades e defeitos dos métodos baseados em malha das duas abordagens. Dito isso, é intuitiva a ideia de se desenvolver métodos combinados de forma a aproveitar as qualidades de cada um. Além disso, de forma geral os métodos que envolvem malhas não se adequam a problemas onde pelo menos uma parte do sistema é discreta, tais como o transporte de sedimentos e interação de estrelas.

1.2 Métodos sem Malha

Um grande esforço foi despendido pela comunidade cientıfica para a resolução de problemas nos quais os tradicionais métodos baseados em malha eram mais frágeis. Deste movimento surgiram os métodos livres de malha, os quais deveriam ser capazes de produzir soluções numéricas estáveis e precisas para os sistemas de equações, integrais ou diferenciais, que descrevem os fenômenos físicos. Estes métodos lançam mão de um conjunto de pontos, nós ou partículas, arbitrariamente distribuidos sem a necessidade de malhas que os dêem conectividade para a obtenção das soluções desejadas. Os diferentes formalismos desenvolvidos para esta classe de métodos numéricos se diferenciam basicamente na maneira com que as funções são aproximadas e nos processos de implementação do algoritmo.

1.2.1 Smoothed Particle Hydrodynamics (SPH)

Em 1977, Lucy (13) em paralelo a Monaghan e Gingold (9) propuseram métodos, bastante semelhantes entre si, de partículas Lagrangeanas e sem malha aplicando-o diretamente na resolução de problemas da astrofísica. O primeiro abordou o estudo da hipótese de fissão como o mecanismo pelo qual binários próximos são formados, partindo de uma massa de gás simétrica em rotação que em determinado ponto sofre uma perturbação antissimétrica. Já o segundo aplica o método a uma gama de sistemas estelares politrópicos de elevada assimetria esférica.

Como a maioria dos problemas de interesse desta área da física não apresenta qualquer tipo de simetria esférica, ou por já não possuí-la em seu estado inicial ou pelo surgimento de forças assimétricas a partir de rotações ou de campos magnéticos, o refinamento de malha necessário para garantir uma precisão satisfatória torna o custo computacional proibitivo. Outra questão inerente aos sistemas físicos discutidos que motivaram a criação do método é a ausência de fronteiras, o que é quase impossível de se tratar com os métodos tradicionais.

O SPH é, portanto, livre de malhas de abordagem Lagrangiana que consiste na discretização do objeto de estudo através de um conjunto de partículas e faz uso de ferramentas estatísticas para recuperar as expressões analíticas para as variáveis físicas envolvidas a partir de uma distribuição arbitrária de partículas. Estas possuem massa, se movem de acordo com as leis de movimento do problema em questão e com isso carregam todas as propriedades físicas e termodinâmicas pertinentes. O SPH possui um processo de aproximação de funções em uma partícula, incluindo suas derivadas, que usa a informação das partículas vizinhas dentro de uma região de influência. Isto se dá inicialmente através da integral de convolução da função a ser aproximada com uma função de suavização, ou núcleo, que suaviza a influência de cada partícula vizinha sobre a considerada de acordo com sua distância. Esta região, vizinhança, de influência é formalmente definida

pela propriedade de suporte compacto da função núcleo. Em seguida, esta integral é transformada em um somatório para que os cálculos sejam realizados computacionalmente. Após a execução das discretizações dos objetos e das equações governantes, as forças resultantes são contabilizadas e consequentemente a aceleração de cada partícula é obtida. Por fim, a simulação avança no tempo através de um dos diversos algoritmos disponíveis na literatura para resolução de equações diferenciais ordinárias.

1.3 O SPH na Indústria

Devido a alta capacidade de lidar com grandes deformações, superfícies livres, interfaces e relativa facilidade de se implementar relações física complicadas o método SPH foi rapidamente empregado em diversos outros ramos da astrofísica, e em sequência extendido e aplicado em uma vasta quantidade de áreas da ciência, tais como a hidrodinâmica Newtoniana, magneto-hirodinâmica, mecânica dos sólidos e física nuclear. Este grande interesse gerado na comunidade científica levou a uma crescente investigação e aprimoramento do método (11).

O SPH já se mostrou capaz de suprir muitas das deficiências dos métodos tradicionais e vêm sendo amplamente utilizado na investigação de fenômenos da natureza. No entando, a sua aplicação no âmbito industrial ainda não está consolidada. Para tal, Shadloo (22) afirma que as dificuldades numéricas inerentes da aplicação do método devem ser devidamente tratadas. Isto inclui que deve ser possível modelar diferentes tipos de materiais e as interações entre si, modelar corretamente as discontinuidades e continuidades das interfaces destes materiais, deve conservar todas as grandezas físicas conserváveis e por fim deve ser capaz de lidar de forma simples com fenômenos multifísicos. Apesar da ainda incerteza e desconhecimento de todo o potencial do método SPH este já desempenha um grande papel numa ampla gama de setores da pesquisa industrial.

Em 2004 a ONERA (Office national d'études et de recherches aérospatiales) realizou uma parceria com a DMSE (Departament of Material Science and Engineering) para a simulação do carregamento estrutural de uma fuselagem completa de um avião sob condições reais de colisão com o intuito de gerar um banco de dados sobre o assunto e auxiliar no projeto de melhorias que aumentem a capacidade de sobrevivência de passageiros e tripulação. O modelo da fuselagem foi desenvolvido pela ONERA através do Método dos Elementos Finitos e então foi feito o acomplamento com o SPH para a modelagem da água. Na modelagem SPH foram empregadas aproximadamente 400.000 (quatrocentas mil) partículas e o tempo necessário para se realizar 500ms de simulação foi de aproximadamente 300 horas (21).

Além do estudo citado a cima que se situa na área de engenharia aeroespacial, existem aplicações do método SPH em problemas industriais da área de engenharia

automotiva como a análise do chacoalhar de combustíveis dentro de caminhões tanque em diversas condições de trânsito e também em veículos de corrida. Na engenharia naval e costal o SPH ganha seu espaço devido a capacidade de lidar com superfícies livres complexas e também da modelagem da interação entre materiais. A aplicação nesta área se extende à simulações de marés e correntes, dinâmica das ondas marítimas e transporte de sedimentos. No setor de produção de energia, extração de petróleo bruto, é necessário lidar com vários aspectos dos fluidos, escoamentos multifásicos por fraturas e meios porosos. Além disso também há a modelagem do processo de separação de fases por meio da aplicação de um campo elétrico externo. O SPH ainda é aplicado à processos industriais que envolvem a formação de materiais, misturas de líquidos e jatos líquidos de alta pressão e também na modelagem de fenômenos geológicos (22).

2 Revisão Bibliográfica

2.1 Construção

A forma com que as funções são aproximadas na formulação do SPH se dá em duas etapas distintas. A primeira se dá a partir da noção da aproximação por função núcleo ou, como é frequentemente chamada no meio do SPH, função de suavização. A segunda etapa é a aproximação por partículas, onde o sistema é discretizado para que os cálculos possam ser realizados.

2.1.1 Representação Integral de uma Função

Seja $f(x)$ uma função definida num domínio Ω, a sua aproximação integral é dada da seguinte forma:

$$\langle f(x)\rangle = \int_{\Omega} f(x')W(x-x';h)dx', \tag{2.1}$$

onde h é a medida do tamanho do domínio de influência da função W.

Para entender o conceito da representação integral de uma função considere $d_\tau(x - x_0)$ uma função definida por

$$d_\tau(x-x_0) = \begin{cases} \frac{1}{2\tau}, & x_0-\tau < x < x_0+\tau, \\ 0, & x \leq x_0-\tau \text{ ou } x \geq x_0+\tau. \end{cases} \tag{2.2}$$

onde $\tau > 0$ é uma constante. Agora defina,

$$I(\tau) = \int_{-\infty}^{\infty} d_\tau(x-x_0)dx. \tag{2.3}$$

É direto observar que independente do valor de τ, $I(\tau) = 1$. E, por fim, aplica-se o conceito de limite para a função $d_\tau(x - x_0)$ quando $\tau \longrightarrow 0$. Com isso, tem-se

$$\lim_{\tau\to 0} d_\tau(x-x_0) = 0, \quad \text{quando } x \neq x_0.$$

Além disso, como $I(\tau) = 1$ para cada $\tau \neq 0$,

$$\lim_{\tau\to 0} I(\tau) = 1.$$

Todo este cenário criado pode ser interpretado fisicamente como se a função $d_\tau(x - x_0)$ descrevesse o comportamento de uma força qualquer ao longo do tempo e, por definição, $I(\tau)$ é o Impulso desta força. Ao se considerar o limite dessa função quando $\tau \to 0$ se obtém a idealização de uma "função"cujo valor é zero em todo $t \neq 0$ e em $x = x_0$ ela

assume um valor suficientemente grande de forma que o impulso gerado seja exatamente um. Pode-se então definir, com um rigor minimamente aceitável, a distribuição Delta de Dirac, $\delta(x - x_0)$ (3).

$$\begin{cases} \delta(x - x_0) = 0, \quad \text{quando } x \neq x_0 \\ \int_{-\infty}^{\infty} \delta(x - x_0)dx = 1 \end{cases} \tag{2.4}$$

Considere a integral de convolução da Delta de Dirac com uma função qualquer $f(x)$ que pode ser escrita a partir da definição anterior da integral de convolução de $f(x)$ com $d_\tau(x - x_0)$.

$$\begin{aligned} \int_{-\infty}^{\infty} f(x)\delta(x - x_0)dx &= \int_{-\infty}^{\infty} \lim_{\tau\to 0} f(x)d_\tau(x - x_0)dx \\ &= \lim_{\tau\to 0} \int_{-\infty}^{\infty} f(x)d_\tau(x - x_0)dx \end{aligned} \tag{2.5}$$

Resolvendo primeiro a integral, tem-se,

$$\begin{aligned} \int_{-\infty}^{\infty} f(x)d_\tau(x - x_0)dx &= \int_{x_0-\tau}^{x_0+\tau} f(x)\frac{1}{2\tau}dx \\ &= \frac{1}{2\tau}\int_{x_0-\tau}^{x_0+\tau} f(x)dx. \end{aligned} \tag{2.6}$$

Pelo Teorema do Valor Médio para Integrais,

$$\begin{aligned} \frac{1}{2\tau}\int_{x_0-\tau}^{x_0+\tau} f(x)dx &= \frac{1}{2\tau}(x_0 + \tau - x_0 + \tau)f(\overline{x}), \quad \text{com } \overline{x} \in [(x_0 - \tau; x_0 + \tau] \\ &= f(\overline{x}). \end{aligned} \tag{2.7}$$

E, aplicando o limite quando $\tau \to 0$, tem-se que $\overline{x} \to x_0$ e,

$$\lim_{\tau\to 0} \int_{-\infty}^{\infty} f(x)d_\tau(x - x_0)dx = f(x_0). \tag{2.8}$$

Portanto, trocando x por x' e x_0 por x para se adequar a notação comumente usada no SPH,

$$f(x) = \int_{-\infty}^{\infty} f(x')\delta(x' - x)dx' \tag{2.9}$$

2.1.2 Aproximação por Partículas

A equação 2.9 agora deve ser discretizada para que a integração numérica seja realizada, completando assim o processo de aproximação de funções do método SPH. Então, tem-se a seguinte expressão:

$$\langle f(x_i)\rangle = \sum_j f(x_j)W(x_i - x_j; h)\Delta V_j \tag{2.10}$$

Segundo (2), há alguma dificuldade em se manter a coerência na representação do termo ΔV_j, sendo ele uma espécie de medida do domínio entorno do nó, ou partícula, j. Surgindo assim uma ambiguidade com a definição da soma Riemanniana. Nas aplicações hidrodinâmicas, a questão da ambiguidade da definição de ΔV_j pode ser amenizada com o uso da Equação da Continuidade, o que resulta em:

$$\langle f(x_i)\rangle = \sum_j f(x_j) W(x_i - x_j; h) \frac{m_j}{\rho_j} \tag{2.11}$$

onde m_j e ρ_j são a massa e a densidade da partícula j.

2.2 Função Núcleo

A Função Núcleo é uma peça chave da base do método como um todo. É ela quem determina, para cada partícula, quais outras partículas irão influenciar no cálculo de suas propriedades e qual a magnitude desta influência. Dessa forma, para que uma função núcleo possa ser considerada como tal esta deve satisfazer as condições abaixo.

1. $W(x - x'; h) > 0$ em Ω_I, com $\Omega_I \subset \Omega$,
2. $W(x - x'; h) = 0$ em $\Omega \backslash \Omega_I$,
3. $\int_\Omega W(x - x'; h) dx' = 1$,
4. $W(s, h)$ é uma função monótona decrescente, onde $s = \|x - x'\|$.
5. No limite quando $h \longrightarrow 0$, $W(s, h) \longrightarrow \delta(s)$, onde δ é a Delta de Dirac

As duas primeiras condições são, de certa forma, complementares, pois juntas elas descrevem o comportamento de uma função que possua suporte compacto. Que, de forma mais geral, poderia assumir valores negativos dentro de seu suporte. Esta "restrição"a mais da função núcleo (de ser sempre não-negativa) se mostra necessária por argumentos físicos, pois como os campos a serem aproximados por meio dela representam propriedades físicas das partículas fluidas que não fazem sentido ao assumir valores negativos, o que eventualmente acontecerá caso a função núcleo não satisfaça esta condição. Já o fato de ser nula em todo o domínio fora da região de suporte é crucial para o desempenho computacional do método, pois limita o número de partículas consideradas nos cálculos. Em outras palavras, leva a operação de um caráter global a um caráter local.

A terceira é a condição de normalização, a qual surge diretamente dos argumentos que garantem a consistência desta aproximação integral como um todo, embora não seja suficiente.

A quarta propriedade decorre do argumento físico de que dada uma partícula fluida, as partículas mais próximas exercerão uma maior influência sobre ela do que as mais afastadas.

A quinta propriedade é uma forma de garantir que no limite onde o comprimento do suporte compacto tende a zero a aproximação de uma função tenderá ao valor exato da função, pois a aproximação integral tenderá a representação integral de Dirac (11) (2) (23).

2.3 Aproximação Integral das Diferenciais

Grande parte do interesse se concentra na aproximação das duas primeiras derivadas, pois as equações diferenciais que governam os fenômenos em sua maioria são no máximo de segunda ordem. Em mecânica dos fluidos tem-se a equação de Navier-Stokes e a equação da conservação da massa. O conceito que será exposto a seguir é também facilmente estendido para as derivadas superiores.

2.3.1 Primeira Derivada

Considere uma função $g(x) = f'(x)$, onde $x \in \mathbb{R}$ e $\frac{df(x)}{dx} = f'(x)$. A aproximação integral de $f'(x)$ é dada por:

$$\langle f'(x) \rangle = \int_\Omega f'(x')W(x - x', h)dx'. \tag{2.12}$$

Para desenvolver esta igualdade deve-se aplicar a integração por partes.

$$\langle f'(x) \rangle = \int_\Omega \frac{d}{dx'}(f(x')W(x - x', h))dx' - \int_\Omega f(x')W'(x - x', h)dx'. \tag{2.13}$$

Aplicando o teorema da divergência no primeiro termo do lado direito,

$$\langle f'(x) \rangle = \int_{\partial\Omega} f(x')W(x - x', h)ndS - \int_\Omega f(x')W'(x - x', h)dx'. \tag{2.14}$$

E, pela propriedade de suporte compacto da função núcleo e por sua continuidade,

$$W(x - x', h)|_S = 0. \tag{2.15}$$

Portanto,

$$\langle f'(x) \rangle = -\int_\Omega f(x')W'(x - x', h)dx'. \tag{2.16}$$

2.3.2 Segunda Derivada

O procedimento é bastante análogo ao da primeira derivada. Partindo da expressão para a aproximação integral,

$$\langle f''(x)\rangle = \int_\Omega f''(x')W(x-x',h)dx', \tag{2.17}$$

e aplicando o método de integração por partes duas vezes se obtém a equação a abaixo:

$$\begin{aligned}\langle f''(x)\rangle = f'(x')W(x-x',h) - f(x')W'(x-x',h) \\ + \int_\Omega f(x')W''(x-x',h)dx'.\end{aligned} \tag{2.18}$$

Note que, para a primeira integreação por partes,

$$\begin{aligned} u &= W(x-x',h) & dv &= f''(x')dx' \\ du &= W'(x-x',h)dx' & v &= f'(x'). \end{aligned}$$

E para a segunda,

$$\begin{aligned} u &= W'(x-x',h) & dv &= f'(x')dx' \\ du &= W''(x-x',h)dx' & v &= f(x'). \end{aligned}$$

Aplicando o teorema da divergência nos dois primeiros termos do lado direito da equação 2.18 e, observando que do mesmo modo que o núcleo se anula no bordo de seu suporte compacto a sua derivada também se anulará, tem-se que

$$\langle f''(x)\rangle = \int_\Omega f(x')W''(x-x',h)dx'. \tag{2.19}$$

2.4 Consistência

Todos os diferentes métodos de se aproximar uma função inevitavelmente estão sujeitos a classificação de acordo com o seu grau de consistência. Este, por sua vez, é em geral caracterizado pela máxima ordem do polinômio que pode ser exatamente reproduzido pelo método de aproximação (12). No método dos Elementos Finitos, por exemplo, o grau de consistência da aproximação obtida através de uma determinada função de forma é mensurada desta forma e este conceito é estendido ao SPH para determinar sua consistência. Embora o ponto final da aproximação do SPH seja discreta, uma breve análise das condições de consistência da forma contínua pode mostrar alguns dos efeitos que elas têm sobre a função núcleo.

2.4.1 Função Núcleo

Para uma aproximação ser dita de Ordem Zero de Consistência ou de Consistência C^0 esta deve reproduzir exatamente uma função constante, $f(x) = c$. Então,

$$\begin{aligned} f(x) &= \int_\Omega cW(x - x', h)dx' = c \\ &= c\int_\Omega W(x - x', h)dx', \end{aligned} \tag{2.20}$$

o que leva a

$$\int_\Omega W(x - x', h)dx' = 1. \tag{2.21}$$

Fica evidente então o motivo, a importância, da terceira condição imposta para as funções núcleo, a de normalização. Sem ela não seria possível garantir nem a consistência de ordem zero para a aproximção.

Seguindo o mesmo raciocínio, para uma função linear do tipo $f(x) = c_0 + c_1x$ ser exatamente reproduzida,

$$\begin{aligned} f(x) &= \int_\Omega (c_0 + c_1x')W(x - x', h)dx' \\ &= c_0\int_\Omega W(x - x', h)dx' + c_1\int_\Omega x'W(x - x', h)dx' \\ &= c_0 + c_1\int_\Omega x'W(x - x', h)dx', \end{aligned} \tag{2.22}$$

o que leva a

$$\int_\Omega x'W(x - x', h)dx' = x. \tag{2.23}$$

Esta é a condição para que a função núcleo possua consistência de ordem um, ou consistência C^1. E, unindo as equações 2.21 e 2.23, através de uma pequena manipulação algébrica, é possível chegar a um resultado importante sobre o comportamento das funções núcleo. Como

$$x\int_\Omega W(x - x', h)dx' = x. \tag{2.24}$$

Então,

$$x\int_\Omega W(x - x', h)dx' - \int_\Omega x'W(x - x', h)dx' = x - x,$$

$$\int_\Omega (x - x')W(x - x', h)dx' = 0. \tag{2.25}$$

Note que esta é a integral do produto de uma função ímpar com o núcleo e, esta se anula. Portanto, conclui-se que $(x - x')W(x - x', h)dx'$ é ímpar. Como $(x - x')$ é ímpar,$W(x - x', h)$ é par. Logo, é possível observar que a função núcleo será simétrica. De fato, a maioria dos núcleos utilizados no SPH são simétricos.

Para as derivadas de ordens superior as condições são análogas. Dessa forma, para um polinômio de ordem k, $P_k(x) = c_0 + c_1x + \ldots + c_kx^k$, ser exatamente representado pela função núcleo, esta deverá satisfazer as condições abaixo:

$$\begin{cases} \int_\Omega W(x - x', h)dx' = 1 \\ \int_\Omega x'W(x - x', h)dx' = x \\ \vdots \\ \int_\Omega (x')^k W(x - x', h)dx' = x^k \end{cases} \tag{2.26}$$

Ao se realizar algebrismos análogos aos que resultaram na equação 2.25 em cada uma das equações de 2.26 se obtém o seguinte conjunto de equações:

$$\begin{cases} \int_\Omega W(x - x', h)dx' = 1 \\ \int_\Omega [x - x']W(x - x', h)dx' = 0 \\ \vdots \\ \int_\Omega [(x)^k - (x')^k]W(x - x', h)dx' = 0 \end{cases} \tag{2.27}$$

2.4.2 Aproximação de uma Função

Segundo Belytschko (2), equações diferenciais parciais de segunda ordem requerem que a aproximação possua consistência C^1 para a aplicação de métodos como o SPH. Isto é, polinômios de ordem um, ou campos lineares, devem ser representados exatamente. Então, se a função núcleo tiver consistência de ordem um e os campos a serem representados forem pelo menos de classe C^2, é possível estimar o erro cometido na aproximação utilizando expansão em série de Taylor de $f(x')$ em torno de x'. Dessa forma, tem-se que,

$$\begin{aligned} f(x) &= \int_\Omega [f(x) + f'(x)(x' - x) + O((x' - x)^2))]W(x - x'; h)dx' \\ &= f(x)\int_\Omega W(x - x'; h)dx' \\ &\quad + f'(x)\int_\Omega (x' - x)W(x - x'; h)dx' \\ &\quad + \int_\Omega O((x' - x)^2)W(x - x'; h)dx' \end{aligned} \tag{2.28}$$

onde $O((x - x')^2) = \frac{f^{(2)}}{2!}(c)(x - x')^2 \quad \text{com } c \in \Omega_I$.

Note que, a primeira integral do lado direito da equação é a condição de normalização, portanto é igual a 1. Para lidar com a integral que aparece no segundo termo é

necessário supor ainda que o núcleo seja uma função par, o que resulta em um valor nulo para esta integral. E, por fim, a última integral não pode ser resolvida, pois não se conhece a forma exata do polinômio $O((x'-x)^2)]$). No entanto, é razoável assumir que o polinômio resultante será pelo menos da mesma ordem. Pondo todos estes resultados juntos,

$$f(x) = f(x) + O((x'-x)^2) \tag{2.29}$$

Portanto, pode ser visto que o erro cometido na aproximação integral de uma função realizada através de um núcleo é de segunda ordem. Em outras palavras, é possível reproduzir exatamente um polinômio de ordem um por este método.

Entretanto, este método de aproximação de funções não se limita ao que acaba de ser exposto. Na verdade, é possível obter uma aproximação aonde o erro cometido seja de ordem n, onde $n \in \mathbb{N}$ e pode ser mostrado da mesma forma que foi feita para o caso anterior, apenas difere no número de termos utilizados da série de Taylor. Então,

$$\begin{aligned} f(x) &= \int_\Omega [\sum_{k=0}^{n-1} \frac{f^{(k)}(x)}{k!}(x'-x)^k + O((x'-x)^n)]W(x-x';h)dx' \\ &= \sum_{k=0}^{n-1} \frac{f^{(k)}(x)}{k!} \int_\Omega (x'-x)^k W(x-x';h)dx' + O((x'-x)^n) \end{aligned} \tag{2.30}$$

Comparadando o lado esquerdo da igualdade com o direito se obtêm as condições para função de suavização.

$$\begin{cases} \int_\Omega W(x-x',h)dx' = 1 \\ \int_\Omega (x-x')W(x-x',h)dx' = 0 \\ \vdots \\ \int_\Omega (x-x')^{n-1}W(x-x',h)dx' = 0 \end{cases} \tag{2.31}$$

2.4.3 Aproximação das Derivadas de uma Função

Assim como já foi dito anteriormente no texto, na hidrodinâmica as duas primeiras derivadas de uma função desempenham todo o papel nos fenômenos. Dessa forma, será mostrado como obter as condições da função núcleo apenas para as duas primeiras derivadas, no entanto o procedimento é análogo para as derivadas superiores.

Partindo das equações 2.12 e 2.17, aplica-se a a expansão em série de Taylor de

$f(x')$ em torno de x'.

$$\begin{aligned} f'(x) &= -\int_\Omega f(x')W'(x-x',h)dx' \\ &= -\int_\Omega [\sum_{k=0}^{n-1} \frac{f^{(k)}(x)}{k!}(x'-x)^k + O((x'-x)^n)]W'(x-x';h)dx' \\ &= -\sum_{k=0}^{n-1} \frac{f^{(k)}(x)}{k!} \int_\Omega (x'-x)^k W'(x-x';h)dx' + O((x'-x)^n) \end{aligned} \tag{2.32}$$

E,

$$\begin{aligned} f''(x) &= \int_\Omega f(x')W''(x-x',h)dx'. \\ &= \int_\Omega [\sum_{k=0}^{n-1} \frac{f^{(k)}(x)}{k!}(x'-x)^k + O((x'-x)^n)]W''(x-x';h)dx' \\ &= \sum_{k=0}^{n-1} \frac{f^{(k)}(x)}{k!} \int_\Omega (x'-x)^k W''(x-x';h)dx' + O((x'-x)^n) \end{aligned} \tag{2.33}$$

Ao se comparar os dois lados de cada equação são obtidas novas condições que a função de suavização deverá satisfazer para que se garanta um erro de ordem n, onde $n \in \mathbb{N}$, para a aproximação integral. As condições estão expressas abaixo.

Para a primeira derivada,

$$\begin{cases} \int_\Omega W'(x-x',h)dx' = 0 \\ \int_\Omega (x-x')W'(x-x',h)dx' = 1 \\ \int_\Omega (x-x')^2 W'(x-x',h)dx' = 0 \\ \vdots \\ \int_\Omega (x-x')^{n-1} W'(x-x',h)dx' = 0. \end{cases} \tag{2.34}$$

E, para a segunda derivada,

$$\begin{cases} \int_\Omega W''(x-x',h)dx' = 0 \\ \int_\Omega (x-x')W''(x-x',h)dx' = 0 \\ \int_\Omega (x-x')^2 W''(x-x',h)dx' = 2 \\ \vdots \\ \int_\Omega (x-x')^{n-1} W''(x-x',h)dx' = 0. \end{cases} \tag{2.35}$$

Então, como a propriedade de suporte compacto do núcleo deverá ser mantida, o que implica em se anular em toda a superfície, para a k-ésima derivada o conjunto de

condições abaixo deverá ser satisfeito.

$$\begin{cases} W(x-x',h)|_S = 0 \\ W'(x-x',h)|_S = 0 \\ W''(x-x',h)|_S = 0 \\ \vdots \\ W^{(k)}(x-x',h)|_S = 0 \end{cases} \tag{2.36}$$

2.4.4 Aproximação por Partícula

Até o presente momento se discutiu a consistência do núcleo e da aproximação de uma função e suas derivadas, sempre na forma contínua, a partir de integrais. Isto, no entanto, nada diz sobre o próximo passo no processo de aproximação do método SPH, que é a discretização das equações através de uma abordagem particulada. Dessa forma, o que irá garantir a consistência por partículas, se tratando de uma consistência de ordem um, será a forma discreta da equação 2.31. E, evidentemente, mantendo a propriedade de suporte compacto.

$$\begin{cases} \sum_{j=0}^{N} W(x-x_j,h)\Delta x_j = 1 \\ \sum_{j=0}^{N} (x-x_j)W(x-x_j,h)\Delta x_j = 0 \end{cases} \tag{2.37}$$

onde N é o número total de partículas contidas no domínio de suporte da partícula localizada em x.

No geral, estas duas condições não são satisfeitas. Pois, para partículas próximas às fronteiras, mesmo se distribuidas uniformemente, o domínio de influência será truncado, culminando numa contribuição resultante desbalanceada das partículas. E, no caso de uma distribuição irregular, independentemente da posição da partícula de referência, as condições de consistência também não serão satisfeitas. Dessa forma é possível notar que existe uma inconsistência na aproximação por partícula e então o método SPH não possui nem uma consistência C^0. O que influencia diretamente a imprecisão da solução obtida.

2.4.5 Restaurando a Consistência por Partícula

Existem diversas maneiras de se restaurar as condições de consistência para o caso discreto, cada um com uma abordagem distinta, porém com o mesmo fim. Uma possível forma, baseando-se na criação de uma função de suavização sob medida para cada situação foi proposta por (12). E, é apresentada a seguir.

Para que seja garantida uma aproximação por partícula de consistência C^k o seguinte núcleo deverá ser tomado:

$$\begin{aligned} W(x-x_j,h) &= b_0(x,h) + b_1(x,h)(\frac{x-x_j}{h}) + b_2(x,h)(\frac{x-x_j}{h})^2 + \dots \\ &= \sum_{I=0}^{k} b_I(x,h)(\frac{x-x_j}{h})^I \end{aligned} \tag{2.38}$$

Com isso, as condições para a consistência de ordem k, na forma discreta, são escritas como

$$\begin{cases} \sum_{j=0}^{N}[\sum_{I=0}^{k} b_I(x,h)(\frac{x-x_j}{h})^I]\Delta x_j = 1 \\ \vdots \\ \sum_{j=0}^{N}(\frac{x-x_j}{h})^k[\sum_{I=0}^{k} b_I(x,h)(\frac{x-x_j}{h})^I]\Delta x_j = 0 \end{cases} \tag{2.39}$$

Ou, observando que $\sum_{j=0}^{N}(\frac{x-x_j}{h})^k$ é constante com relação ao índice I, é possível inverter a posição dos somatórios e reescrever o conjunto de igualdades de uma forma mais conveniente.

$$\begin{cases} \sum_{I=0}^{k} b_I(x,h)\sum_{j=0}^{N}(\frac{x-x_j}{h})^I\Delta x_j = 1 \\ \vdots \\ \sum_{I=0}^{k} b_I(x,h)\sum_{j=0}^{N}(\frac{x-x_j}{h})^{I+k}\Delta x_j = 0 \end{cases} \tag{2.40}$$

Agora, assumindo que,

$$m_k(x,h) = \sum_{j=0}^{N}(\frac{x-x_j}{h})^k\Delta x_j, \tag{2.41}$$

os $k+1$ coeficientes $b_I(x,h)$ podem ser determinados através da solução do seguinte sistema de equações:

$$\begin{bmatrix} m_0(x,h) & m_1(x,h) & \dots & m_k(x,h) \\ m_1(x,h) & m_2(x,h) & \dots & m_{1+k}(x,h) \\ \vdots & \vdots & \ddots & \vdots \\ m_k(x,h) & m_{k+1}(x,h) & \dots & m_{k+k}(x,h) \end{bmatrix} \begin{Bmatrix} b_0(x,h) \\ b_1(x,h) \\ \vdots \\ b_k(x,h) \end{Bmatrix} = \begin{Bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{Bmatrix} \tag{2.42}$$

Após a determinação dos coeficientes $b_I(x,h)$, a função de suavização, na forma com que foi proposta nesta argumentação, está completa e então pode ser calculada

numericamente. E, por construção, esta satisfaz exatamente todas as condições para a garantia de uma consistência de ordem k. Ao contrário das funções núcleo tradicionais que dependem apenas da distância entre partículas, esta nova abordagem produz núcleos que são dependentes da posição absoluta das pasrtículas. O que implica diretamente em um aumento do custo computacional envolvido, pois será necessária a resolução do sistema de equações para cada partícula, a cada intervalo de tempo, afinal as partículas estão se movendo. Além disto, é evidente a necessidade de uma determinante não nulo para a matriz dos coeficientes para a obtenção de uma solução única. E, para isto, a movimentação, posicionamento, das partículas deverá respeitar algum tipo de condição que garanta este determinante. O que por sua vez não condiz com a natureza randômica da distribuição das partículas no método SPH original.

Em suma, a restauração da consistência por partícula, na forma apresentada, é sim um grande avanço no que diz respeito a exatidão da solução obtida através do método SPH. No entanto, esta restauração gera problemas em outras áreas, por assim dizer, que devem ser considerados. Além do já exposto sobre o aumento do custo computacional devido a característica local do novo núcleo e sobre a eventual restrição da distribuição das partículas, deve ser obervado que a nova função de suavização obtida apresentará valores negativos em algumas regiões. Além disso, ela provavelmente não será monotônicamente decrescente e nem simétrica. Tais questões podem gerar grandes problemas nos resultados da simulação, tais como valores sem sentido físico para densidade e ouras grandezas físicas.

Chen e Beraun (4), propuseram uma forma generalizada da aproximação executada no método SPH. A proposição visa resolver diretamente a questão da inconsistência nas fronteiras, garantindo então uma consistência C^0 para a aproximação. Em seu trabalho foi apresentado o algoritmo de aproximação num sistema de coordenadas cartesianas em uma e duas diemnsões.

Resumidamente, o método consiste na manipulação direta da expansão em série de Taylor. Dessa forma, considere-a para uma função f em torno do ponto $x = (x_i, y_i)$. E, além disto, multiplicada dos dois lados pela função de suavização W e integrada em todo o domínio Ω.

$$\begin{aligned}\int_\Omega f(\bar{x})W d\bar{x} = f(\bar{x_i})\int_\Omega W d\bar{x} + f_x(\bar{x_i})\int_\Omega (x-x_i)W d\bar{x} \\ + f_y(\bar{x_i})\int_\Omega (y-y_i)W d\bar{x} + \frac{f_{xx}(\bar{x_i})}{2}\int_\Omega (x-x_i)^2 W d\bar{x} \\ + f_{xy}(\bar{x_i})\int_\Omega (x-x_i)(y-y_i)W d\bar{x} \\ + \frac{f_{yy}(\bar{x_i})}{2}\int_\Omega (y-y_i)^2 W d\bar{x} + \dots\end{aligned} \tag{2.43}$$

Se todos os termos diferenciais forem desconsiderados, se obtém uma forma corrigida

da aproximação de uma função.

$$\langle f(\bar{x}_i)\rangle = \frac{\sum_{j=1}^{N} f_j W_{ij} m_j/\rho_j}{\sum_{j=1}^{N} W_{ij} m_j/\rho_j} \tag{2.44}$$

Onde $W_{ij} = W(\bar{x}_i - \bar{x}_j, h)$ e a etapa de aproximação por partícula já foi realizada.

Quanto a consistência desta nova forma de aproximação, é possível observar que as integrais de $(x - x_i)W$ e $(y - y_i)W$ irão se anular para partículas internas ao domínio, pois W é uma função par. O que irá garantir um erro da aproximação de $O(h^2)$. E, para partículas próximas da fronteiras isto já não é mais verdade, garantindo então um erro de $O(h)$.

No que diz respeito a aproximação por partículas, como já discutido anteriormente para o caso convencional, a distribuição irregular das partículas impedirá a garantia do erro de ordem $O(h^2)$ para as partículas interiores ao domínio. No entanto, é fácil mostrar que esta nova expressão garante a reprodução exata de uma função constante para partículas distantes das fronteiras, ou seja, possui consistência C^0. E, por fim, embora não possa se garantir o mesmo para as partículas próximas ou nas fronteiras, há um aprimoramento na aproximação feita se comparada com a forma tradicional.

Seguindo na explanação no algoritmo, a aproximação das derivadas de uma função é feita exatamente o mesmo procedimento. Para facilitar, é feita a primeira demonstração para o caso unidimensional. Com isso, troca-se $\bar{x}$ por x. A primeira derivada é dada por,

$$\langle f_{xi}\rangle = \frac{\int_\Omega [f(x) - f_i]\hat{W} dx}{\int_\Omega (x - x_i)\hat{W} dx}. \tag{2.45}$$

E, a segunda derivada por,

$$\langle f_{xxi}\rangle = \frac{\int_\Omega [f(x) - f_i]\tilde{W} dx - f_{xi}\int_\Omega (x - x_i)\tilde{W} dx}{(1/2)\int_\Omega (x - x_i)^2\tilde{W} dx}. \tag{2.46}$$

Esta nova forma de se estimar as derivadas de uma função difere em alguns pontos pertinentes da forma tradicional. Não há mais a necessidade de se trabalhar com as respectivas derivadas da função núcleo. A seleção da função de suavização nesta estimativa generalizada irá depender da ordem da derivada, pois uma das condições a se satisfazer será a capacidade de gerar um denominador não-nulo para todo o domínio. Com isso, note que para primeira derivada $\hat{W}$ deverá ser simétrica e, para a segunda derivada $\tilde{W}$ deverá ser uma função antissimétrica. Portanto, se torna claro que as funções empregadas nesta nova estimativa não compartilham das mesmas características dos núcleos tradicionais.

Já para o caso bidimensional, sendo análogo para mais dimensões, o algoritmo se torna menos direto como no apresentado anteriormente. Ao se desprezar as derivadas de

ordem superior a 1 para a obtenção da aproximação para a primeira derivada o resultado é o surgimento de duas equações acopladas, envolvendo tanto a primeira derivada em x quanto em y. Com isso a matriz dos coeficientes deverá ser invertida, o que gera um aumento no custo computacional envolvido. O processo para as demais derivadas e também mais dimensões é análogo, aumentando apenas o tamanho da matriz a se inverter.

Outras formas de se remediar a questão da inconsistência por partículas nas aproximações das derivadas são modificações da forma originalmente apresentada por Monaghan em 1977, (9). Dessa forma, ao se fazer

$$\nabla(f_i) = \sum_{j=1}^{N}(f_j - f_i)\nabla W_{ij} m_j/\rho_j \tag{2.47}$$

a consistência C^0 é restaurada, pois a derivada de uma constante é exatamente reproduzida. Por outro lado, ao se fazer

$$\nabla(f_i) = \sum_{j=1}^{N}(f_j + f_i)\nabla W_{ij} m_j/\rho_j \tag{2.48}$$

a ordem zero de consistência não é mais garantida, porém o momento linear e angular é conservado. O que é de grande contribuição para a estabilidade da simulação.

2.5 Construção de Funções de Suavização

Durante a seção anterior foram apresentadas algumas formas de se remediar, ou de fato recuperar, a inconsistência por partícula. Fenômeno que é inerente a aplicação direta da segunda etapa do processo de aproximação que fundamenta o método SPH. E como pôde ser visto, um dos métodos consiste na reconstrução sistemática da função núcleo, o que gera um aumento significativo no custo computacional da simulação. Além disso, o núcleo gerado pode ser negativo em regiões do seu suporte, o que é catastrófico para simulações de hidrodinâmica.

Nesta seção será resumida a digreção feita por (12), onde é considerado que a função núcleo possui a forma de um polinômio dependente apenas da distância relativa entre os pontos considerados. Dessa forma, dentro do domínio de suporte de raio h, o núcleo terá a seguinte forma geral.

$$W(x - x', h) = W(R, h) = a_0 + a_1 R + a_2 R^2 + \cdots + a_n R^n \tag{2.49}$$

Primeiro, é importante observar que o núcleo deverá ser simétrico radialmente,

propriedade 4, logo R representa a distância do ponto x ao ponto x'.

$$R = |x - x'| \geqslant 0. \tag{2.50}$$

Em seguida, para verficar as condições a serem satisfeias para garantir a consistência de ordem k ao núcleo é necessário conhecer suas derivadas. Derivada esta que é tomada com relação a variável x',

$$\frac{dW(R,h)}{dx'} = \frac{dW}{dR}\frac{dR}{dx'}. \tag{2.51}$$

Onde,

$$\frac{dR}{dx'} = \begin{cases} 1, & \text{se } x - x' < 0 \\ -1, & \text{se } x - x' > 0. \end{cases} \tag{2.52}$$

Após as considerações feitas a cima, considere a seguinte expressão como sendo forma geral para a i-ésima derivada da função núcleo.

$$W^{(i)}(R,h) = \begin{cases} \Lambda\ , & \text{se } x - x' < 0 \\ (-1)^i\Lambda\ , & \text{se } x - x' > 0. \end{cases} \tag{2.53}$$

Onde, $\Lambda = [i!a_i + (i+1)!a_{i+1}R + \cdots + \prod_{n}^{n-i+1} a_n R^{n-i}]$.

A validade desta expressão pode ser verificada por indução finita. Será mostrado apenas para o caso onde $x - x' > 0$, o outro caso é análogo. Com efeito, note que é válido para $i = 1$,

$$W^{(1)}(R,h) = (-1)[a_1 + (2)!a_2R + \cdots + (n)a_nR^{n-1}] = \frac{dW}{dx'}. \tag{2.54}$$

Agora suponha que valha para $i = k$,

$$W^{(k)}(R,h) = (-1)^k[i!a_k + (k+1)!a_{k+1}R + \cdots + \prod_{n}^{n-k+1} a_n R^{n-k+1}]. \tag{2.55}$$

Então, para $k+1$,

$$\begin{aligned}\frac{dW^{(k)}(R,h)}{dx'} &= (-1)^{k+1}[(k+1)!a_{k+1} \\ &\qquad + (k+2)!a_{k+2}R + \cdots + \prod_{n}^{n-k} a_n R^{n-(k+1)}] \\ &= (-1)^{\xi}[(\xi)!a_{\xi} + (\xi+1)!a_{\xi+1}R + \cdots + \prod_{n}^{n-\xi+1} a_n R^{n-\xi}],\end{aligned} \tag{2.56}$$

Onde $k+1=\xi$. Verifica-se, então, que a expressão proposta é válida para todo i, onde $i \in \mathbb{N}$.

Por fim é necessario ainda analisar as condições para a existência da i-ésima derivada do núcleo. O ponto crítico desta análise é onde $x - x' = 0$, pois se terá

$$W^{(i)}(0,h) = \begin{cases} i!a_i, & \text{se } x - x' \to 0_- \\ (-1)^i i!a_i, & \text{se } x - x' \to 0_+. \end{cases} \tag{2.57}$$

Dessa forma, para o caso onde $i=1$, por exemplo,

$$\begin{aligned} a_1 &= -a_1 \\ &= 0. \end{aligned} \tag{2.58}$$

E, para a segunda derivada,

$$W^{(2)}(0,h) = 2a_2. \tag{2.59}$$

No entanto, para que ela exista, a primeira derivada também deverá existir, o que leva novamente à condição expressa em 2.58. Com isso é possível observar que, para garantir a existência da i-ésima derivada, a condição abaixo deverá ser satisfeita.

$$\begin{cases} a_1, a_3, \ldots, a_{i-1} = 0 & \text{se } i \text{ é par.} \\ a_1, a_3, \ldots, a_i = 0 & \text{se } i \text{ é ímpar.} \end{cases} \tag{2.60}$$

No método SPH é necessário apenas que a primeira e a segunda derivada das funções núcleo existam, o que remete à 2.58. E portanto, a expressão geral dada em 2.49 se reduz à

$$W(x-x',h) = W(R,h) = a_0 + a_2R^2 + \cdots + a_nR^n \tag{2.61}$$

Todas as considerações a cima foram feitas para o caso unidimensional, onde $R = |x - x'|$. Para estender o raciocínio para m dimensões basta observar que

$$R = \sqrt{(x_1 - x_1')^2 + \cdots + (x_m - x_m')^2}\ . \tag{2.62}$$

E,

$$\nabla R = -\frac{|x_1 - x_1'|\hat{x}_1 + \cdots + |x_m - x_m'|\hat{x}_m}{R} \tag{2.63}$$

Note que ∇R é um vetor unitário e será positivo ou negativo dependendo de uma condição para m dimensões análoga a expressa em 2.52. Alguns exemplos das funções núcleos mais utilizadas podem ser encontrados em (11).

2.6 Formulação SPH

Em seções anteriores foram desenvolvidas as aproximações integrais para as derivadas de uma função, no entanto para a resolução de problemas através do método SPH ainda é necessário discretizar as integrais utilizando partículas.

Nesta seção serão desenvolvidas as formulações mais básicas dos operadores do SPH que são amplamente empregadas em trabalhos disponíveis na literatura.

2.6.1 Primeira Derivada

Antes de tudo, vale ressaltar que esta formulação deverá ser válida tanto para campos escalares quanto vetoriais, ou seja, este operador apresentará a mesma forma tanto para aplicações como divergente quanto para como gradiente. Dito isto, considere o resultado já obtido para a aproximação integral da derivada de uma função,

$$\begin{aligned} \langle \nabla f(x) \rangle &= \int_\Omega \nabla f(x') W(x - x', h) dx' \\ &= -\int_\Omega f(x') \nabla W(x - x', h) dx'. \end{aligned} \tag{2.64}$$

Na forma discreta esta esquação será dada por

$$\begin{aligned} \langle \nabla f(x_i) \rangle &= -\sum_{j \in \Omega_i} \frac{m_j}{\rho_j} f(x_j) \nabla_j W_{ij} \\ &= \sum_{j \in \Omega_i} \frac{m_j}{\rho_j} f(x_j) \nabla_i W_{ij} \end{aligned} \tag{2.65}$$

Note que $W(x_i - x_j, h)$ foi abreviado por W_{ij} e foi utilizada a propriedade de antissimetria da primeira derivada do núcleo, ∇W_{ij}. Como as operações de derivação são realizadas coordenada a coordenada, será mostrada a antissimetria para uma coordenada qualquer.

$$\begin{aligned} \frac{dW}{dx(||x - x'||)} &= \frac{dW(||x - x'||)}{d(x - x')} \frac{d(x - x')}{dx} \\ &= \frac{dW(||x - x'||)}{d(||x - x'||)} \frac{d(||x - x'||)}{d(x - x')}.1 \end{aligned} \tag{2.66}$$

E,

$$\begin{aligned}\frac{dW}{dx'(||x-x'||)} &= \frac{dW(||x-x'||)}{d(x-x')}\frac{d(x-x')}{dx'} \\ &= -\frac{dW(||x-x'||)}{d(||x-x'||)}\frac{d(||x-x'||)}{d(x-x')}\end{aligned} \tag{2.67}$$

Então,

$$\frac{dW}{dx(||x-x'||)} = -\frac{dW}{dx'(||x-x'||)}. \tag{2.68}$$

Esta formulação para os operadores divergente e gradiente embora simples e obtida pela aplicação direta dos conceitos discutidos até aqui, não garante nem mesmo a consistência de ordem zero. Outra questão é que o momento linear não se conserva. Para mostrar tal resultado basta verificar que a interação sobre a partícula i devido a j não será igual e oposta a interação de i sobre j. Ou,

$$\frac{m_i}{\rho i}\frac{m_j}{\rho_j}f(x_j)\nabla_i W_{ij} \neq -\frac{m_j}{\rho j}\frac{m_i}{\rho_i}f(x_i)\nabla_j W_{ij} \tag{2.69}$$

Uma forma de se contornar este problema, proposta por (14), é obtida através da Lagrangiana do sistema. No entanto, é possível se chegar ao mesmo resultado fazendo uso de identidades do operador gradiente. Considere ϕ uma função escalar qualquer e seja $f(x)$ um campo escalar ou vetorial. Tem-se que

$$\nabla(\phi f(x)) = \phi\nabla f(x) + f(x)\nabla\phi. \tag{2.70}$$

Resolvendo para $\nabla f(x)$,

$$\nabla f(x) = \frac{1}{\phi}[\nabla(\phi f(x)) - f(x)\nabla\phi]. \tag{2.71}$$

A partir desta expressão geral é possível obter inúmeras formas alternativas de se escrever o gradiente da função $f(x)$. No entanto, a escolha da função escalar ϕ deve ser criteriosa de forma a resultar em identidades que conservem o momento linear. Ao tomar $\phi = 1/\rho$, se chega a expressão proposta por Monaghan.

$$\nabla f(x) = \rho\left[\nabla(\frac{f(x)}{\rho}) + \frac{f(x)}{\rho^2}\nabla\rho\right]. \tag{2.72}$$

Neste caso, se fossem realizadas as aproximações para cada termo da expressão haveria um aumento expressivo do custo computacional e da sensibilidade à desordem

das partículas. Logo, as aproximações são realizadas apenas no termos envolvendo os operadores diferenciais. Dessa forma,

$$\begin{aligned}\langle \nabla f(x_i) \rangle &= \rho_i \sum_{j \in \Omega_i} \frac{m_j}{\rho_j} \left[\frac{f(x_j)}{\rho_j} + \frac{f(x_i)}{\rho_i^2} \rho_j \right] \nabla_i W_{ij} \\ &= \rho_i \sum_{j \in \Omega_i} m_j \left[\frac{f(x_j)}{\rho_j^2} + \frac{f(x_i)}{\rho_i^2} \right] \nabla_i W_{ij}\end{aligned} \tag{2.73}$$

A conservação do momento linear é observada imediatamente a partir desta equação, pois esta é simétrica. A resultante da interação entre duas partículas é nula.

$$m_i m_j \left(\frac{f(x_i)}{\rho_i^2} + \frac{f(x_j)}{\rho_j^2} \right) \nabla_i W_{ij} + m_j m_i \left(\frac{f(x_j)}{\rho_j^2} + \frac{f(x_i)}{\rho_i^2} \right) \nabla_j W_{ji} = 0 \tag{2.74}$$

2.6.2 Segunda Derivada

A segunda derivada de um campo esfalar f, ou Laplaciano, nada mais é do que o divergente do gradiente ($\nabla^2 f = \nabla(\nabla f)$). Que é obtido diretamente da segunda derivação do interpolador do SPH, como já foi mostrado, a forma integral é dada por

$$\begin{aligned}\langle \nabla^2 f(x) \rangle &= \int_\Omega \nabla^2 f(x') W(x - x', h) dx' \\ &= \int_\Omega f(x') \nabla^2 W(x - x', h) dx'.\end{aligned} \tag{2.75}$$

Na forma discreta,

$$\langle \nabla^2 f(x_i) \rangle = \sum_{j \in \Omega_i} \frac{m_j}{\rho_j} f(x_j) \nabla^2 W_{ij} \tag{2.76}$$

Onde $W(x_i - x_j, h)$ foi abreviado por W_{ij} e $\nabla_i^2 = \nabla_j^2 = \nabla^2$.

Esta, portanto, é uma forma simples e direta de se aproximar a segunda derivada de uma função através do método SPH. No entanto, esta formulação é consideravelmente sensitiva a desordem das partículas, (17).

Uma possível forma de se contornar esta sensibilidade é usar a equivalência do operador laplaciano já apresentada para se realizar a aproximação. Dessa forma a segunda derivada da função núcleo não estaria mais envolvida no cálculo. Com efeito,

$$\langle \nabla^2 f(x_i) \rangle = \frac{1}{\rho_i} \sum_{j \in \Omega_i} m_j [(\nabla f(x_j)) - (\nabla f(x_i))] \nabla W_{ij} \tag{2.77}$$

Sendo,

$$\langle \nabla f(x_k) \rangle = \frac{1}{\rho_k} \sum_{l \in \Omega_k} m_l [f(x_l) - f(x_k)] \nabla W_{kl}. \tag{2.78}$$

Embora a questão relacionada a desordem das partículas seja contornada nessa abordagem, será necessário realizar dois somatórios por partícula. O que torna o custo computacional proibitivo. Cleary e Monaghan (5) propuseram novas formulações para o laplaciano com a finalidade de se modelar a fenômeno de transferência de calor por condução. O caso onde a condutividade térmica do fluido é constante ou depende da temperatura é diretamente extendido ao caso onde a viscosidade do fluido é constante ou depende da velocidade. E, sua aproximação integral é dada por

$$\frac{1}{\rho(x)} \int_\Omega [\mu(x') + \mu(x)][v(x) - v(x')] G(x - x') dx' \tag{2.79}$$

Com

$$G(q) = \frac{q \nabla W(q)}{q^2 + \eta^2}, \tag{2.80}$$

onde η é um valor imposto manualmente para prevenir singularidades. Além disso, se os núcleos atendem a condição de serem radialmente simétricos, é possível escrever

$$\nabla W(q) = qF(q), \tag{2.81}$$

e, portanto, é fácil ver que $G = F$ e que não haverá singularidades.

A forma discreta do interpolador será dada então por

$$\frac{1}{\rho} \nabla(\mu \nabla v) = \sum_{j \in \Omega_i} \frac{m_j}{\rho_i \rho_j} (\mu_i + \mu_j)(v_i - v_j) F_{ij}. \tag{2.82}$$

2.7 Integração Temporal

Como o procedimento de aproximação do método SPH reduz as equações diferenciais parciais contínuas a um conjunto de equações diferenciais ordinárias, qualquer algoritimo de integração temporal estável para este tipo de equação pode ser usado (17). Em (14), Monaghan relata que os estudos recentes havia sido empregado o método Leapfrog nos casos mais simples e que W. Benz em 1987 haveria conseguido resultados satisfatórios utilizando métodos Runge-Kutta de baixa ordem. Para situações envolvendo uma física mais complicada, o método Preditor-corretor é empregado. O qual dá uma conservação da energia satisfatória e conserva exatamente o momento linear e angular. Um bom

algoritimo de integração temporal deve ser capaz de conservar a energia e o momento, ser computacionalmente eficiente e suportar o maior passo de tempo possível (11)

Os métodos de integração são classificados como explícitos ou implícitos. A diferença principal entre eles é que os explícitos utilizam apenas o estado atual, eventualmente o anterior, para o cálculo do estado seguinte. Já os implícitos fazem uso, além dos estados já mencionados, do estado seguinte. O que resulta num sistema de equações (23). Todos os métodos de integração citados até o momento são explícitos.

Método de Euler

O método de Euler, ou Runge-Kutta de primeira ordem, é o método mais simples. A sua forma vem direto da definição de derivada.

$$\frac{dv}{dt} = \lim_{\delta t \to 0} \frac{v(t+\delta t) - v(t)}{\delta t} \approx \frac{v(t+\delta t) - v(t)}{\delta t} \tag{2.83}$$

Então,

$$v_i(t+\delta t) = v_i(t) + a_i(t)\delta t \tag{2.84}$$

E,

$$x_i(t+\delta t) = x_i(t) + v_i(t+\delta t)\delta t \tag{2.85}$$

Note que a formulação do método envolve apenas os instantes de tempo t e $t+\delta t$, o que possibilita o cálculo direto do tempo seguinte a partir do estado atual. O que torna o método consideravelmente veloz e simples.

Quanto a precisão do método, esta pode ser estimada através da aplicação direta da expansão em série de Taylor. O que dá

$$v(t+\delta t) = v(t) + \delta t \frac{dv}{dt}(t) + O(\delta t^2) \tag{2.86}$$

Então, reorganizando,

$$\frac{v(t+\delta t) - v(t)}{\delta t} = \frac{dv}{dt}(t) + O(\delta t). \tag{2.87}$$

O que mostra que o erro cometido ao se utilizar o método de Euler é de primeira ordem (23).

Leapfrog

A vantagem deste método é o baixo uso de memória. No entanto, quando o comprimento de suavização começa a se tornar muito pequeno, o passo mínimo de avanço no tempo necessário pode ser proibitivo (11).

De forma semelhande ao método de Euler, o Leapfrog (23) também se baseia na aproximação da derivada. A diferença é que a estimativa é feita através de diferenças centradas.

$$\frac{dv}{dt} = \lim_{\delta t \to 0} \frac{v(t + \frac{1}{2}\delta t) - v(t - \frac{1}{2}\delta t)}{\delta t} \approx \frac{v(t + \frac{1}{2}\delta t) - v(t - \frac{1}{2}\delta t)}{\delta t} \tag{2.88}$$

Neste método as velocidades são atualizadas a cada meio passo de tempo e as posições a cada passo inteiro. Este tipo de andamento proporciona uma sensação de que as velocidades e posições estão saltando umas sobre as outras, daí que surge o nome "leapfrog"que significa algo como "pulo do sapo"em inglês.

$$v_i(t + \frac{1}{2}\delta t) = v_i(t - \frac{1}{2}\delta t) + a_i(t)\delta t \tag{2.89}$$

e,

$$x_i(t + \delta t) = x_i(t) + v_i(t + \frac{1}{2}\delta t)\delta t \tag{2.90}$$

Também se faz necessário calcular as velocidades em um instante de tempo inteiro, para isso se pode tomar a média dos valores já calculados para os instantes intermediários.

$$v_i(t) = \frac{1}{2}[v_i(t + \frac{1}{2}\delta t) + v_i(t - \frac{1}{2}\delta t)], \tag{2.91}$$

onde se $v_i(t + \frac{1}{2}\delta t)$ for substituido pela igualdade 2.89, pode-se mostrar que

$$v_i(t) = v_i(t - \frac{1}{2}\delta t) + a_i(t)\frac{1}{2}\delta t. \tag{2.92}$$

O que nada mais é do que avançar meio passo de tempo utilizando a aceleração do instante inteiro atual.

No instante inicial do método há a necessidadde de uma adaptação, pois não se possui informação sobre $v_i(-\frac{1}{2}\delta t)$. Para isto o método de Euler é empregado,

$$v_i(-\frac{1}{2}\delta t) = v_i(0) - \frac{1}{2}a_i(0)\delta t. \tag{2.93}$$

A partir deste passo inicial todo o método pode se desenvolver em sequência naturalmente. Em (7) uma abordagem ligeiramente diferente é usada para se calcular a velocidade nos instantes inteiros e também para contornar a questão do primeiro passo de tempo. Utiliza-se a aceleração do tempo t e a velocidade do tempo $t + \frac{1}{2}\delta t$ para se calcular a velocidade no tempo inteiro seguinte, $t + \delta t$. Com efeito,

$$v_i(t + \delta t) = v_i(t + \frac{1}{2}\delta t) + a_i(t)\frac{1}{2}\delta t. \tag{2.94}$$

Para efeito de comparação, basta retroceder em δt a expressão a cima. Dessa forma, tem-se que

$$v_i(t) = v_i(t - \frac{1}{2}\delta t) + a_i(t - \delta t)\frac{1}{2}\delta t. \tag{2.95}$$

O que é semelhante a forma já apresentada, porém ao invés da aceleração no tempo t usa-se a do tempo $t - \delta t$.

E, por fim, para o instante inicial é feito

$$v_i(\frac{1}{2}\delta t) = v_i(0) + \frac{1}{2}a_i(0)\delta t. \tag{2.96}$$

Assim como feito para o método de Euler, a precisão do algoritmo do Leapfrog pode ser estimada fazendo uso da expansão em série de Taylor. Com efeito,

$$v(t - \frac{1}{2}\delta t) = v(t) - \frac{\delta t}{2}\frac{dv}{dt}(t) + \frac{\delta t^2}{4}\frac{d^2v}{dt^2}(t) + O(\delta t^3) \tag{2.97}$$

E,

$$v(t + \frac{1}{2}\delta t) = v(t) + \frac{\delta t}{2}\frac{dv}{dt}(t) + \frac{\delta t^2}{4}\frac{d^2v}{dt^2}(t) + O(\delta t^3) \tag{2.98}$$

Subtraindo uma equação da outra e rearranjando,

$$\frac{v(t + \frac{1}{2}\delta t) - v(t - \frac{1}{2}\delta t)}{\delta t} = \frac{dv}{dt}(t) + O(\delta t^2). \tag{2.99}$$

Portanto, o erro cometido através ao avançar no tempo através do método Leapfrog é dito ser de segunda ordem.

Verlet

Monaghan (17) descreve de forma bastante simples e direta como se dá o algoritimo do método Verlet de segunda ordem. Para um sistema unidimensional

$$\frac{dx}{dt} = v, \tag{2.100}$$

$$\frac{dv}{dt} = f(x), \tag{2.101}$$

Assumindo um passo de tempo constante, δt,

$$x(t + \frac{1}{2}\delta t) = x(t) + \frac{1}{2}v(t)\delta t, \tag{2.102}$$

$$v(t + \delta t) = v(t) + f(x(t + \frac{1}{2}\delta t))\delta t, \tag{2.103}$$

$$x(t + \delta t) = x(t + \frac{1}{2}\delta t)\frac{1}{2}v(t + \delta t)\delta t. \tag{2.104}$$

Uma forma alternativa do algoritimo, com a mesma precisão, é dada por

$$v(t + \frac{1}{2}\delta t) = v(t) + \frac{1}{2}f(t)\delta t, \tag{2.105}$$

$$x(t + \delta t) = x(t) + v(t + \frac{1}{2}\delta t)\delta t, \tag{2.106}$$

$$v(t + \delta t) = v(t + \frac{1}{2}\delta t) + \frac{1}{2}f(t + \delta t)\delta t. \tag{2.107}$$

A primeira forma do algoritimo é chamada de "drift-kick-drift", enquanto esta última é chamada de "kick-drift-kick". O "kick"é a variação na velocidade devido a força, e o "drift"é o deslocamento da partícula devido a velocidade "inicial"daquela iteração do algoritmo. Em alguns casos pode ser interessante ter as forças sendo atualizadas no final do passo de tempo, assim como ocorre na variação "kick-drift-kick".

3 Metodologia

Neste capítulo serão apresentadas todas as questões gerais pertinentes do desenvolvimento de um código para resolução de problemas hidrodinâmicos em duas dimensões através do método SPH, que em seguida será aplicado para a realização de quatro experimentos numéricos simples. Sendo então primeiro apresentadas e discutidas as equações que modelam o sistema e as suas formas finais após a aproximação por partícula. Em Seguida será dada uma visão geral da concepção e do funcionamento do programa e também discutidas questões acerca da implementação do método, tais como criação e posicionamento de partículas e busca de vizinhos. Por fim será apresentado o roteiro escolhido para o tratamento de dados e sua visualização. O código completo está disponível no apêndice A.

O primeiro experimento consiste numa versão simplificada de um rompimento de barragem, onde um quadrado de fluido, inicialmente estacionário, escoa após a sua "barragem"ser instantaneamente retirada e na sequência colide contra uma parede. O segundo simula a queda de uma gota de fluido, que é aproximada por um círculo, dentro de um recipiente de paredes rígidas. No terceiro experimento, uma gota de fluido é derramada dentro do mesmo recipiente, onde agora há uma porção de fluido em repouso. O quarto e último experimento consiste na simulação da colisão de dois escoamentos opostos.

3.1 Equações Governantes

As características do sistema físico para os quatro experimentos são as mesmas, logo os detalhamentos feitos a partir deste ponto valerão para todos os casos. Dito isto, o sistema estudado possui apenas duas dimensões e está imerso em um campo gravitacional constante, $\vec{g} = g(-\hat{j})$. O fluido simulado é a água, logo a equação da conservação do momento linear, Navier-Stokes (8), será dada por

$$\rho\frac{D\vec{V}}{Dt} = -\nabla p + \mu\nabla^2\vec{V} + \rho\vec{g}. \tag{3.1}$$

Onde μ é a viscosidade do fluido. E a conservação da massa é dada por

$$\frac{D\rho}{Dt} = -\rho.\nabla\vec{V}\ . \tag{3.2}$$

Que é igual zero, pois o fluido é incompressível.

3.1.1 Formulação SPH

Para calcular as derivadas espaciais contidas no lado direito da equação 3.1, as grandezas devem ser interpoladas através das funções núcleo para obter o campos suavizados. Dessa forma, aplicando diretamente o conceito da aproximação por partículas, o gradiente da pressão será dado por

$$\langle \nabla P(x_i) \rangle = \sum_j m_j \frac{p_j}{\rho_j} \nabla W(x_i - x_j; h). \tag{3.3}$$

No entanto, esta forma de cálculo resulta em contribuições não simétricas, ou seja, o momento linear não é conservado. Além disso, nem mesmo a consistência de ordem zero pode ser garantida, como já foi discutido anteriormente. Para o caso, a forma simetrizada será

$$\langle \nabla P(x_i) \rangle = \sum_j m_j \frac{p_i + p_j}{2\rho_j} \nabla W(x_i - x_j; h). \tag{3.4}$$

É inerente da aproximação do SPH a necessidade de se conhecer a massa e a densidade das partículas. A massa é constante para cada partícula, já a densidade é calculada através da soma ponderada das densidades das partículas dentro do domínio de suporte da partícula em questão. E expressão empregada neste trabalho se dá diretamente do conceito de aproximação por partícula.

$$\langle \rho_i \rangle = \sum_j m_j W(x_i - x_j; h). \tag{3.5}$$

Embora esta abordagem seja simples e conserve exatamente a massa total do sistema (11), as partículas próximas das fronteiras sofrerão com a deficiência de vizinhos, o que gera a inconsistência por partícula. A qual pode ser recuperada através da renormalização do interpolador da densidade, assim como discutido na seção 2.4.5. A densidade das partículas também pode ser atualizada através da equação da continuidade, onde se determina a sua variação no tempo. Dessa forma, a densidade inicial é definida e então evoluida no tempo. A forma discreta da equação da continuidade é, aplicando a aproximação por partícula apenas no termo do gradiente,

$$\frac{D\rho_i}{Dt} = -\rho_i \sum_j \frac{m_j}{\rho_j} v_j \nabla_i W(x_i - x_j; h). \tag{3.6}$$

De forma a reduzir os erros gerados pela inconsistência por partículas, Monaghan (14) propõe uma antissimetrização da equação a cima que parte da igualdade $\nabla_i(v_i - v_j) =$

$\nabla_i(v_i)$ e leva a

$$\frac{D\rho_i}{Dt} = -\rho_i \sum_j \frac{m_j}{\rho_j}(v_i - v_j)\nabla_i W(x_i - x_j; h). \tag{3.7}$$

Com isso a densidade das partículas irá variar apenas quando elas possuem movimento relativo entre si. Outras formas de antissimetrização são encontradas em (11).

Em suma, as duas abordagens possuem vantagens e desvantagens. A equação 3.5 conserva a massa total do sistema, porém sofre com as questões de fronteira. Já a equação 3.7, como depende do movimento relativo das partículas, não é influenciada pela deficiência de vizinhos. Além disso, há uma vantagem computacional, pois não há a necessidade de se realizar um laço exclusivo para o cálculo da densidade. A desvantagem desta abordagem é que ela não conserva exatamente a massa total do sistema, o que compromete a estabilidade (15).

O termo da pressão no método SPH é calculado explicitamente através de uma equação de estado apropriada para o caso de estudo. De acordo com Monaghan (15) pode ser o quão complicada quanto se queira. O SPH foi inicialmente desenvolvido para tratar escoamentos compressíveis onde o movimento das partículas é dado pelos gradientes de densidade. Consequentemente, o método carrega em si uma dificuldade para lidar com escoamentos incompressíveis. A aplicação direta da equação de estado verdadeira para um fluido incompressível irá requerer um passo de tempo extremamente pequeno para que seja estável (11).

A restrição da densidade constante das partículas pode ser introduzida no formalismo do SPH, no entanto as equações resultantes ainda não são possíveis de resolver sem mais aproximações (16). No entanto, a imcompressibilidade de um fluido é uma idealização feita para efeito de cálculos teóricos no desenvolvimento da mecânica dos fluidos para aqueles nos quais a velocidade caracteríStca do escoamento é muito menor que a velocidade de propagação do som no meio considerado. Isto é analiticamente representado por

$$\frac{\delta\rho}{\rho} = \frac{vL}{c^2\tau} \to 0\ . \tag{3.8}$$

Onde v, L e τ são as ordens de grandeza caracteristicas do escoamento para a velocidade, o comprimento e o tempo, respectivamente. E, c é a velocidade do som para o meio.

Dessa forma, para se modelar fluidos incompressíveis pelo método SPH é necessário considerar que na realidade estes são compressíveis. O ponto chave se resume na escolha da equação de estado adequada para tal questão. Uma expressão amplamente usada na literatura é (19), (11) e (20).

$$p = c^2\rho\ . \tag{3.9}$$

A escolha do valor da velocidade do som é crucial para o andamento da simulação. Usar o valor real irá novamente levar à necessidade de passos de tempo proibitivamente pequenos. No entanto, o valor escolhido ainda deverá ser grande o suficiente para que o comportamento do fluido seja suficientemente próximo ao do real.

Monaghan (16), utiliza uma equação de estado que descreve as ondas sonoras precisamente modificada para incorporar um valor menor para a velocidade do som. A equação é dada por

$$p = B((\frac{\rho}{\rho_0})^\gamma - 1). \tag{3.10}$$

Onde ρ_0 é um valor de referência para a densidade, γ é uma constante usualmente tomada como sendo $\gamma = 7$. E B é o parâmetro que determina a velocidade do som.

Morris (19) faz considerações sobre essa equação, mais especificamente sobre o uso de $\gamma = 7$. Isto tende a amplificar o efeito das variações de densidade sobre o valor da pressão. O que implica no aumento da sensibilidade da equação com relação às flutuações na densidade. Para escoamentos com baixos números de Reynolds é aconselhado o uso de $\gamma = 1$.

No programa desenvolvido foi utilizada a equação estado proposta por Desbrun (6), que nada mais é do que a equação dos gases ideais. Em seu trabalho esta equação é aplicada de fato aos gases, porém devido a simplicidade de implementação e à vantagens numéricas ressaltadas por Desbrun esta mesma equação será aplicada a dinâmica dos líquidos. A lei dos gases ideais é dada por

$$pV = nRT. \tag{3.11}$$

Onde n é o número de mols e R é a constante dos gases ideais.

Observando que $\frac{m}{n} = M$, onde M é a massa molar da substância, é possível reescrever a equação de estado como

$$\frac{p}{\rho} = R_M T = k. \tag{3.12}$$

Onde $R_M = R/M$ é a constante dos gases ideais modificada e k é uma constante, pois só serão tratados casos isotérmicos.

Como na equação dos gases ideais é empregada a pressão absoluta, é possível subtrair dos dois lados da igualdade a pressão de referência. Que nada mais é do que a aplicação da densidade de referência na equação de estado, $p_0 = k\rho_0$. Com isso,

$$p = k(\rho - \rho_0). \tag{3.13}$$

É importante notar que neste momento p representa $p - p_0$. Não será usado nenhum subíndice para não carregar a notação e, como o interesse se dá sobre o gradiente, não haverá problemas. Pois $\nabla P = \nabla(p - p_0)$. Então, novamente, k é uma constante e possui unidade energia por unidade de massa, $[J/Kg]$.

Esta equação de estado é projetada para que a densidade seja mantida oscilando em torno de um valor constante. No caso, como as partículas possuem todas a mesma massa, o resultado será a propensão a se manter uma distribuição uniforme dentro do objeto simulado.

Comparando as equações 3.9 e 3.13 é possível notar que

$$k = c^2. \tag{3.14}$$

Note que, com uma pequna manipulação das unidades é possível mostrar que $[J/Kg] = [m^2/s^2]$. Com isso a velocidade do som artificial empregada pode ser obtida.

Por fim, é interessante ressaltar que as três equações de estado mencionadas a cima são relacionadas e portanto equivalentes. Através da definição da velocidade do som isotérmica,

$$c_T = \sqrt{\left(\frac{\partial p}{\partial \rho}\right)_T} = \sqrt{\frac{P}{\rho}}, \tag{3.15}$$

e da 3.14 se obtêm as equações 3.9 e 3.13 diretamente. Já a equação 3.10, tomando $\gamma = 1$, é reescrita como

$$p = \frac{B}{\rho}(\rho - \rho_0). \tag{3.16}$$

Com isso, nota-se que

$$B = \rho_0 c^2 = p_0. \tag{3.17}$$

É fato que o real valor para a velocidade do som é encontrado modelando a propagação da onda sonora como adiabática, porém como o objetivo das considerações feitas até o momento é utilizar um valor reduzido para velocidade do som, não há prejuizos nesta operação.

A aproximação do termo viscoso, aplicando diretamente o conceito de partículas leva a

$$\langle \nabla^2 V \rangle = \sum_j \frac{m_j}{\rho_j} V_j \nabla^2 W_{ij}. \tag{3.18}$$

Esta formulação, como já comentado na seção 2.6.2, não é costumeiramente empregada pois em geral o laplaciano da função núcleo apresenta alta sensibilidade a desordem das partículas. Para simular a viscosidade foram propostas várias expressões que nada tem a ver com a viscosidade real do fluido ou com o formalismo da aproximação do SPH, porém promovem a dissipação da energia do sistema. (15), ressalta que a forma mais utilizada até então na literatura é dada por

$$\prod_{ij} = \begin{cases} \frac{1}{\bar{\rho_{ij}}}(-\alpha \bar{c}_{ij}\mu_{ij} + \beta\mu_{ij}^2) & V_{ij}.x_{ij} < 0 \\ 0 & V_{ij}.x_{ij} > 0 \end{cases} \tag{3.19}$$

Onde a notação $\bar{A}_{ij} = \frac{1}{2}(A_i + A_j)$, $A_{ij} = A_i - A_j$, c é a velocidade do som, α e β são parâmetros ajustáveis e,

$$\mu_{ij} = \frac{hV_{ij}.x_{ij}}{(x_{ij}^2 + \eta^2)} \tag{3.20}$$

Onde η é um parâmetro utilizado para prevenir singularidades.

Esta forma de se modelar a viscosidade é atrativa pois conserva tanto o momento linear quanto o angular, além de ser zero para rotações de corpo rígido e invariante por transformações de Galileu (14). O uso da viscosidade artificial pode se dar simplesmente para a estabilização do algoritmo (17).

Para este trabalho foi feita uma simetrização do Laplaciano da velociadade análoga a já realizada para o gradiente da pressão. Com isso, o termo viscoso se dá por

$$\mu\langle\nabla^2 V\rangle = \mu\sum_j \frac{m_j}{\rho_j}(V_j - V_i)\nabla^2 W_{ij}. \tag{3.21}$$

Apesar das desvantagens já mencionadas do uso do laplaciano do núcleo, é possível construir uma função de suavização especialmente para este uso. O que será discutido na próxima seção. O uso da aproximação do termo viscoso real é de implementação mais simples e adequadas para pequenos testes, principalmente quando não se considera a conservação da energia. Esta forma simétrica do termo viscoso promove a conservação do momento linear e pode ser interpretada como se cada partícula fosse acelerada na direção da velocidade relativa da sua vizinhança.

3.1.2 Núcleos de Suavização

A velocidade, estabilidade e precisão das simulações de SPH são extremamente dependentes da função de suavização utilizada. Dada essa importância, é possível desenvolver e empregar diferentes núcleos numa mesma simulação. Sendo cada um deles mais adequado a uma interpolação específica.

Neste trabalho são empregadas funções de suavização na forma proposta por Muller (20). No entanto as constantes de normalização devem ser ajustadas para o caso 2D. Os núcleos propostos possuem forma geral definida por

$$W(x,h) = \frac{1}{Ch^d} \begin{cases} f(q), & 0 \leq \ q \leq 1 \\ 0, & q \geq 1 \end{cases} \tag{3.22}$$

Onde C é a constante de normalização, d é a dimensão do domínio, h é comprimento de suavização e $q = ||x||/h$.

Para a completa aplicação deste núcleo geral no método SPH é necessário conhecer a forma de suas duas primeiras derivadas. As contas referentes à obtenção da forma geral tanto para o gradiente quanto para o laplaciano estão detalhadas no apêndice B.

O gradiente, que é obtido através da derivação coordenada a coordenada, possui a forma

$$\nabla W = \frac{1}{Ch^{d+2}} \begin{cases} \dfrac{f^{(1)}(q)}{q} x, & 0 \leq \ q \leq 1 \\ 0, & q \geq 1 \end{cases} \tag{3.23}$$

Já o Laplaciano demanda mais atenção a detalhes no cálculo e sua forma geral é dada por

$$\nabla^2 W = \frac{1}{Ch^{d+2}} \begin{cases} f^{(2)}(q) + (d-1)\dfrac{f^{(1)}(q)}{q}, & 0 \leq \ q \leq 1 \\ 0, & q \geq 1 \end{cases} \tag{3.24}$$

Nas simulações realizadas são utilizados três núcleos diferentes, cada um atendendo a um propósito específico. Para interpolações quaisquer, (7) propõe o uso de um polinômio de grau 6 que pode ser escrito como

$$f_{p6}(q) = (1 - q^2)^3. \tag{3.25}$$

Para o cálculo do gradiente da pressão na equação do momento linear, Desbrun (6) projeta uma função de suavização na forma

$$f_s(q) = (1 - q)^3. \tag{3.26}$$

E, por fim, para interpolar as forças viscosas é usada uma versão modificada por Fall do núcleo proposto por Desbrun. A alteração se dá pela necessidade da adaptação das

características particulares deste núcleo de 3D para 2D.

$$f_v(q) = \frac{q^2}{4} - \frac{q^3}{9} - \frac{1}{6}\log(q) - \frac{5}{36} \tag{3.27}$$

Em seguida é necessário impor a condição de normalização para esses núcleos propostos de forma a fazer com que eles possuam consistência de ordem zero. Como as funços já são projetadas para serem simétricas, pois dependem somente da distância $q = \frac{||x||}{h}$, a integral de normalização se dá de forma mais simples com o uso de coordenadas polares no caso 2D. Dessa forma,

$$\frac{2\pi}{C}\int_0^1 f(q)qdq = 1. \tag{3.28}$$

Estas integrais são bastante simples, pois os integrandos são apenas polinômios. Os valores obtidos para as constantes de normalização são

$$\begin{aligned} C_{p6} &= \pi/4 \\ C_s &= \pi/10 \\ C_v &= \pi/40. \end{aligned} \tag{3.29}$$

A escolha de W_{p6} se dá pelo seu formato do tipo de sino, similar a Gaussiana. Se respeitadas todas as condições já discutidas sobre os núcleos em geral, é garantida a consistência de ordem 1 a menos da inconsistência por partícula. Fall (7) ainda ressalta que há uma pequena vantagem computacional deste interpolador sobre os outros similares devido à variável q só se apresentar na forma quadrada. Isto implica em não se calcular raiz quadrada no algoritmo, o que reduz o acúmulo de erros de truncamento. Na 3.1.2 pode-se observar a forma deste interpolador com $h = 1$ e, dentro do suporte compacto, este é dado por

$$W_{p6} = \frac{4}{\pi}(1 - q^2)^3. \tag{3.30}$$

É importante ressaltar que o domínio de interesse se resume à região interna de um círculo de raio unitário. No gráfico de W_{p6} é possível notar regiões negativas, porém estas se localizam fora do domínio citado. Isto acontece pois o gráfico foi gerado sobre um domínio na forma cartesiana, $[-1:1]$x$[-1:1]$.

Por outro lado, ∇W_{p6} não apresenta comportamento adequado para o cálculo do gradiente da pressão nos casos hidrodinâmicos. Desbrun (6) faz considerações acerca do gráfico do gradiente da spline cúbica, que é um interpolador de características muito

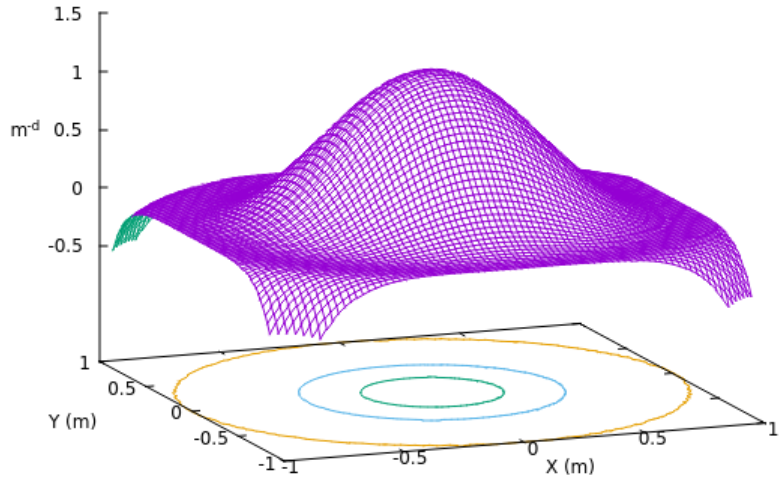

Figura 3 – Núcleo empregado na interpolação da densidade.

próximas às do polinômio aqui empregado. A saber, o gradiente de W_{p6} é escrito como, dentro do suporte compacto,

$$\nabla W_{p6} = \frac{-24}{\pi}(1-q^2)^2. \tag{3.31}$$

E, a 3.1.2 ilustra o gráfico do gradiente dessa função de suavização.

Do gráfico de ∇W_{p6} é possível notar que a contribuição das forças de interação das partículas próximas será cada vez mais atenuada até se anular na origem, isto é, em $q = 0$. Tal fato favorece a criação de conglomerados de partículas, o que é exatamente o comportamento oposto ao desejado para que se mantenha distribuição aproximadamente regular de partículas e densidade constante. É importante pontuar que nas aplicações da astrofísica, por outro lado, este comportamento da spline, e de W_{p6}, é desejado.

De forma a resolver a questão W_s foi proposto. Este interpolador possui formato de um "espinho", assim como seu gradiente. Além de atender a todas as condições gerais, promove uma força de repulsão adequada à medida que as partículas se aproximam demais umas das outras. O que consequentemente evita a criação de aglomerados de partículas. Este núcleo também se anula nas fronteiras, como esperado. A 3.1.2 ilustra este comportamento e a expressão completa para ∇W_s é obtida diretamente da aplicação

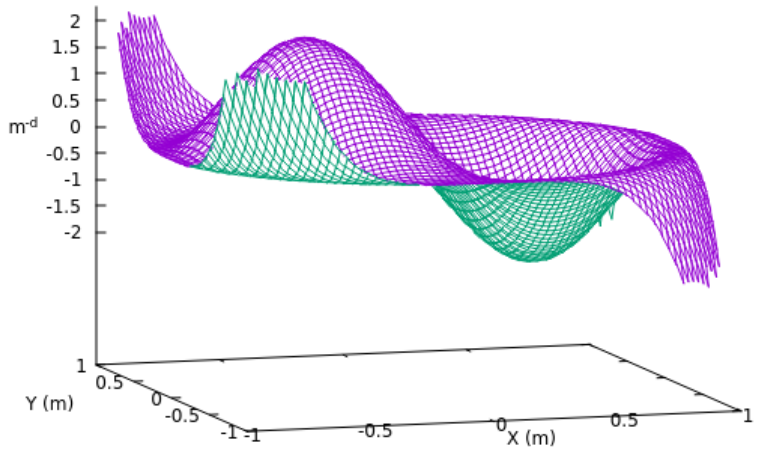

Figura 4 – Gráfico de ∇W_{p6}. Não empregado nas simulações devido a propensão de criar aglomerados de partículas (6).

de $f_s(q)$ na formulação geral dos núcleos já apresentada.

$$\nabla W_s = \frac{-30}{\pi}\frac{(1-q)^2}{q}x. \tag{3.32}$$

O projeto do núcleo utilizado nos cálculos da viscosidade se dá a partir do desejo de que o seu laplaciano seja uma função proporcional a $(1-q)$ além de se anular nas fronteiras, assim como o gradiente e a primitiva. Isto resulta em um núcleo com Laplaciano dado por

$$\nabla^2 W_v = \frac{40}{\pi}(1-q). \tag{3.33}$$

Outros núcleos, como os dois anteriores já discutidos, possuem laplacianos com regições negativas. O que não é desejável, pois acarretaria em forças viscosas que não dissipam energia do sistema e sim adiciona a ele. Muller (20), afirma que o uso deste interpolador, 3.1.2, na forma como foi construído para o cálculo da viscosidade proporciona estabilidade o suficiente para que não seja necessário o uso de nenhum outro termo dissipativo na equação do momento linear.

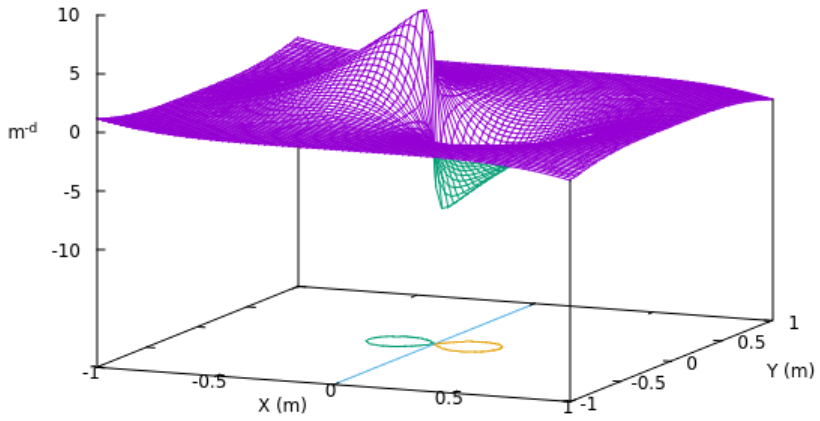

Figura 5 – Gradiente do núcleo empregado na interpolação do termpo das forças de pressão, ∇W_s.

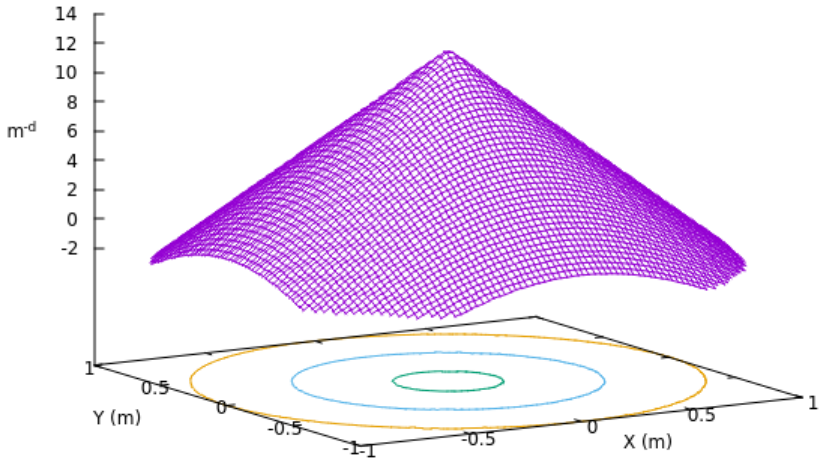

Figura 6 – Laplaciano do núcleo empregado na interpolação do termo viscoso, $\nabla^2 W_v$.

3.2 Implementação

Na seção anterior foram apresentadas as equações básicas da mecânica dos fluidos que descrevem os sistemas já considerando a atuação de forças de corpo externas, no caso o campo gravitacional constante no qual está imerso. E, em seguida, discretizadas de acordo com o formalismmo do método SPH. De posse destas equações e dos núcleos adequados é possível iniciar a implementação do método. O código foi desenvolvido por meio da linguagem de programação C++ (10) dentro de um ambiente de programação chamado Geany, que é um software livre para distribuições Linux. O sistema operacional utilizado foi o Ubuntu 16.04 Lts.

De forma a manter a clareza e lógica do fluxo de informações, minimizar erros e aumentar a robustez, o desenvolvimento do código se inicia pelo encapsulamento de cada pequena rotina necessária para a solução na forma de um método. Ao analisar cada um destes métodos é possível identificar que estes se dividem em quatro grandes grupos de acordo com o escopo da sua atuação. Com isto surgem as quatro classes principais que compõem a estrutura do programa: "Simulation"; "Geometry"; "Particle"e "Solver".

A primeira classe, "Simulation", reúne todas as informações pertinentes da simulação como um todo. Isto é, os parâmetros físicos e numéricos. A partir destes é possível definir qual o fluido que será simulado, pois são especificados os valores da densidade de referência, viscosidade e módulo de elasticidade artificial. A força de corpo externa, qual o refinamento espacial e temporal, duração total e também o intervalo de escrita dos dados de saída.

Em seguida é necessário definir a geometria do objeto, porção de fluido, daí a classe "Geometry". Esta classe é puramente virtual e fornece métodos para as classes que dela herdam, os quais devem ser sobreescritos. As classes herdeiras são as que de fato diferenciam os três expermentos realizados e a discussão será feita posteriormente. De volta a classe "Geometry", esta se encarrega de determinar quantas partículas serão necessárias de acordo com os parâmentros da simulação previamente definidos. E, por fim, realizar o posicionamento das partículas.

A terceira classe aqui discutida é talvez a mais intuitiva. "Particle"nada mais é do que a classe cujos objetos são exatamente as partículas SPH que dão corpo ao método. Dessa forma, assim como descrito na teoria do método, elas carregam as informações de densidade, posição, velocidade, aceleração e a massa, que no caso é definida como estática para todos os objetos desta classe. Além disso, também há um método que é responsável por normalizar a massa das partículas de acordo com densidade de referência e e disposição no espaço.

Por fim, a classe "Solver"que é responsável por fazer a simulação acontecer, ou seja, avançar no tempo. Em seus métodos estão as rotinas de cálculo para a atualização da densidade, das forças atuantes e da posição, velocidade e aceleração de cada partícula. Nela

também se encontra a rotina de escrita dos dados em arquivos de texto, que é executada a cada frame.

O programa tem como saída de dados arquivos nomeados com o tempo absoluto da simulação e é composto por uma matriz com as informações de posição, velocidade e densidade instantâneas daquele passo de tempo. A ferramenta utilizada para a geração de gráficos é o software livre Gnuplot e o processo é automatizado através de um script escrito por meio da linguagem Shell, que se encontra no apêndice C.

3.2.1 Posicionamento de Partículas

Uma forma conveniente de se definir a posição inicial das partículas SPH para simulações bidimensionais é através de uma malha retangular regular, posicionando em cada vértice uma partícula (15). Sendo esta então a forma empregada. Uma opção mais vantajosa, segundo Monaghan (16), seria posicioná-las por meio de uma malha hexagonal. Por fim, (11) sugere o uso dos já estabelecidos algoritmos de geração de malha dos métodos tradicionais e então posicionar as partículas no baricentro de cada célula.

3.2.2 Normalização da Massa

Embora a partícula seja representada como um ponto material ela ocupa um espaço finito. Como se está a tratar de uma distribuição uniforme de matéria, é possível obter um valor aproximado do espaço ocupado por cada partícula dividindo o valor da área do domínio pelo número de partículas. Como a massa das partículas é inicializada com um valor qualquer, no caso como sendo unitário, é possível calcular diretamente a densidade média das partículas. E este valor estará em desacordo com o de referência. Além disso, ao se executar pela primeira vez a rotina de suavização da densidade será observado o mesmo fato. Em (7) é proposto um algoritmo de suavização que irá fazer com que a massa das partículas seja alterada de forma com que sua densidade se aproxime, na média, da de referência. Esta operação não considera os efeitos de borda e o valor da massa é calculado a partir de

$$m = m_0 \rho_0 \frac{\sum_j \rho_j}{\sum_j \rho_j^2}. \tag{3.34}$$

3.2.3 Busca de Vizinhos

O algoritmo de busca de vizinhos, ou seja, partículas dentro do comprimento de suavização, foi o conhecido como "busca por força bruta". Este consiste por checar a condição da distância relativa de todas as partículas, o que resulta num total de N^2 operações, onde N é o número de partículas. Como a suavização da densidade e das forças são realizadas em momentos separadas, este valor se duplica. Há uma pequena

vantagem que pode ser tirada a partir da simetria do sistema, o que reduz para um total de $\sum_i^N (N - i)$ operações em cada cálculo.

3.2.4 Tratamento de Fronteiras

A maneira escolhida para se representar as fronteiras do domínio da simulação foi a de barreiras sólidas. Com ela é calculada a colisão inelástica das partículas, coeficiente de restituição menor do que um (7). Para ilustrar o algoritmo, considere uma barreira vertical localizada em x_b. O tempo passado desde o momento em que haveria a colisão é calculado por

$$t^* = (x - x_b)/v_x, \tag{3.35}$$

Onde x e v_x são a posição e a velocidade da partícula no eixo horizontal, x. Em seguida, como há um fator de amortecimento ligado a inelasticidade da colisão com a barreira sólida, é feito um primeiro ajuste da posição da partícula de acordo com a real magnitude da velocidade pós colisão. Dessa forma,

$$\alpha^* = \alpha - v_\alpha(1 - damp)t^*, \tag{3.36}$$

Onde $\alpha = \{x, y\}$ e $damp$ é o fator de amortecimento. O índice $*$ indica o momento deste primeiro ajuste.

Por fim as componetes perpendiculares a barreira, no caso a horizontal, da posição e velocidade são refletidas. E, as velocidades amortecidas.

$$x = [x_b + (x_b - x^*)]damp \tag{3.37}$$

E,

$$\begin{cases} v_x = -(v_x)damp \\ v_x = (v_x)damp \end{cases} \tag{3.38}$$

3.2.5 Passo de Tempo, Comprimento de suavização e Método de Integração

Na literatura é possível encontrar condições que determinam o valor máximo admissível do passo de tempo de forma a torná-lo variável no decorrer da simulação. O mesmo vale para o comprimento de suavização, aumentando ou diminuindo a resolução de acordo com a necessidade (14), (18), (11). Para este trabalho foram tomados como referência os valores declarados por (7) em sua publicação. No entanto, por meio de testes

numéricos foram atingidos valores mais adequados à situação. Isto vale tanto para o passo de tempo quanto para o comprimento de suavização.

No que se refere ao método de avanço no tempo, tendo como base a breve discussão e explanção apresentada no capítulo 1, foi implementado o método de integração Leapfrog neste código SPH.

3.3 Tratamento de Dados

Os dados de instante de tempo atual da simulação, posição, velocidade e densidade de cada partícula são impressos em um arquivo de texto separado para cada intervalo de tempo de escrita. A geração de gráficos da posição das partículas e exportação dos mesmos em formato "jpeg"é feita através do software livre Gnuplot, cujo processo foi automatizado através de um script escrito em Shell. Para uma melhor visualização dos resultados da simulação usou-se um outro software livre, o GIMP, para unir as imagens no formato de uma animação.

4 Resultados

Neste capítulo serão apresentados os resultados numéricos decorrentes das quatro experiências numéricas incorporadas ao código SPH desenvolvido. No entanto, antes disto será feita uma pequena descrição mais detalhada do procedimento de discretização do domínio considerado para cada caso. Isto é, uma continuação da explicação geral já dada no capítulo anterior para a classe "Geometry". E por fim será disponibilizado um link e um código QR para a visualização e download das animações criadas.

4.1 Distribuição inicial

Como já exposto anteriormente, a classe "Geometry"é puramente virtual, ou seja, fornece métodos para classes que dela irão se derivar, herdar publicamente. No caso, serão as classes que definirão o domínio a ser discretizado e o posicionamento das partículas no mesmo. As classes desenvolvidas dão nome aos casos: "Box"; "Circle"; "DropInThePool"e "Colision".

Na estrutura do código desenvolvido o que ocorre primeiro é a declaração de um ponteiro do tipo "Geometry"e então após a escolha do usuário este ponteiro é "reduzido"para um dos quatro tipos que desta classe herdam. Isto implica em necessariamente todos os métodos das classes derivadas serem métodos sobrescritos da primitiva (10). Dessa forma, estas quatro novas classes compartilharão os mesmos métodos. Sendo um responsável por verificar se num determinado ponto deverá existir uma partícula ou não, o outro encarregado de contabilizar o número total de partículas e o terceiro e último método responsável por de fato posicionar as partículas.

4.1.1 Box

Nesta classe é criado um retângulo de fluido, no caso um quadrado com $0,5m$ de lado. O qual será submetido ao campo gravitacional durante a simulação e às interações partícula-partícula. Como já explicado na seção 3.2.1, as partículas foram posicionadas de acordo com uma malha regular quadrada. A Figura 7 ilustra a distribuição inicial das partículas para este caso, que foi gerada com uma resolução mais grosseira com 169 partículas para facilitar a visualização do posicionamento regular das mesmas.

4.1.2 Circle

Para construir uma gota de fluido esta classe faz a varredura de um quadrado que circunscreve o circulo que dará forma a gota e para isso é preciso definir o raio deste círculo.

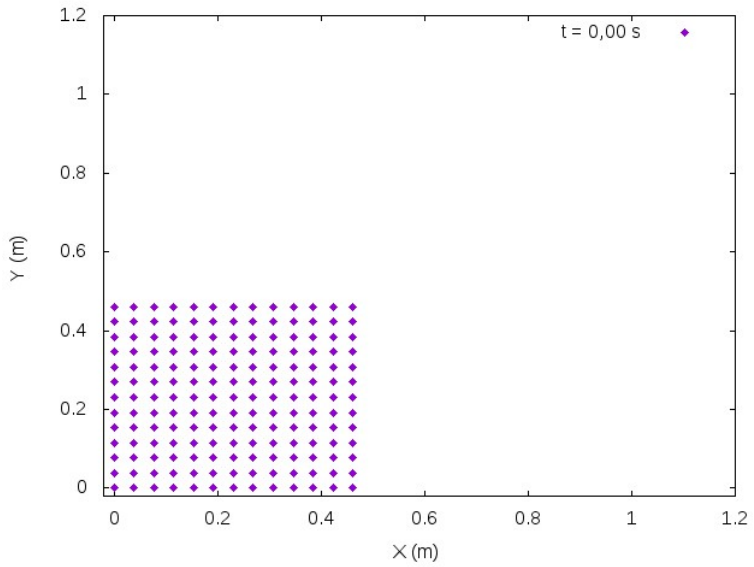

Figura 7 – Distribuição inicial das partículas para o caso do rompimento de barragem. Resolução grosseira, $h = 5.10^{-2}m$, o que resulta em 169 partículas.

No caso, é construída um porção circular de fluido com $0,25m$ de raio com seu centro a uma altura de $0,3m$ do chão. A Figura 8 ilustra a distribuição inicial das partículas para este caso, que foi gerada com uma resolução mais grosseira com 131 partículas

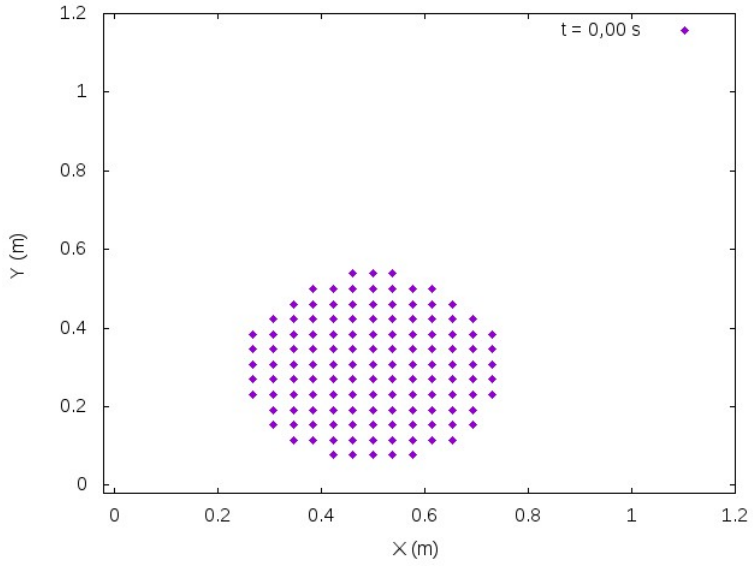

Figura 8 – Distribuição inicial das partículas para o caso do cair de uma gota no chão. Resolução grosseira, $h = 5.10^{-2}m$, o que resulta em 131 partículas.

4.1.3 DropInThePool

Este terceiro caso se mostra como uma junção das técnicas utilizadas nos dois anteriores. É composto por um círculo de fluido com $0,20m$ de raio a uma altura de $0,80m$

do chão e um retângulo de fluido em repouso com $2m$ de comprimento e $0,20m$ de altura. A Figura 9 ilustra a distribuição inicial das partículas para este caso, que foi gerada com uma resolução mais grosseira com 503 partículas

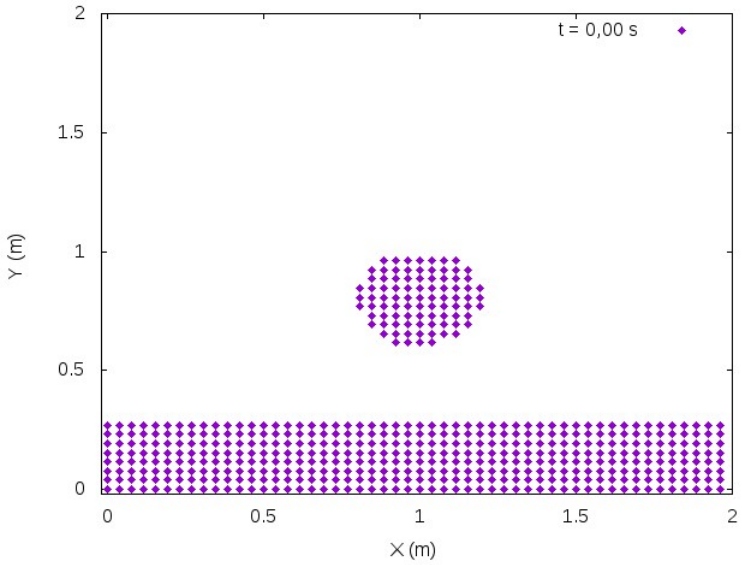

Figura 9 – Distribuição inicial das partículas para o caso do cair de uma gota numa poça. Resolução grosseira, $h = 5.10^{-2}m$, o que resulta em 503 partículas.

4.1.4 Colision

O quarto caso de estudo se aproxima de uma modificação do caso BOX, pois o usa como base para a construção dos blocos de fluido. Entretanto foi criada uma estrutura, classe, separada para este caso de forma a incorporar o caso independentemente ao código.

É simulada a quebra de duas barragens, opostas, que então se chocam em dado momento. Cada bloco possui $0,4m$ de comprimento e $0,9m$ de altura. Neste caso as paredes rígidas foram deslocadas para formar um quadrado com $2m$ de lado. A 10 ilustra a distribuição inicial com um refinamento mais grosseiro, empregando 780 partículas.

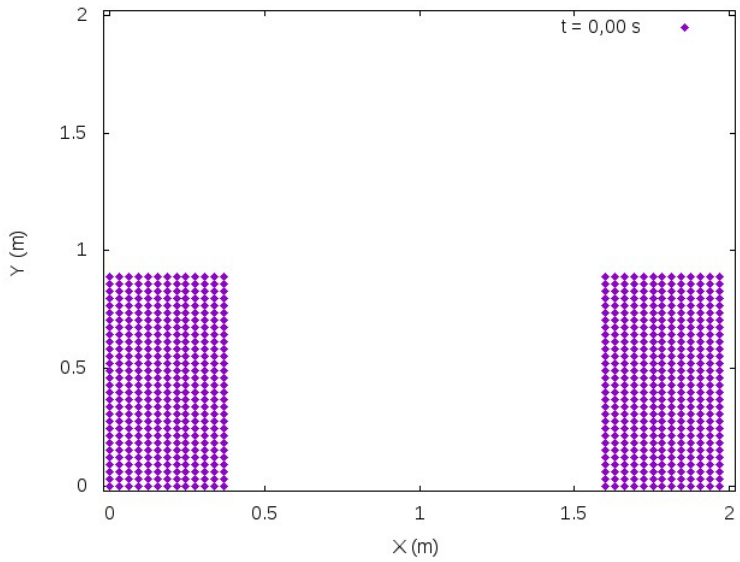

Figura 10 – Distribuição inicial das partículas para o caso de colisão. Resolução grosseira, $h = 4.10^{-2}m$, o que resulta em 780 partículas.

4.2 Rompimento de Barragem

O primeiro caso de estudo realizado é bastante comum na literatura para testes iniciais de diversos métodos e também alvo de estudo com variações mais sofisticadas (16), (11) e (18). Neste trabalho, como já mencionado, foi criada uma distribuição inicial de partículas na forma de um quadrado com $0,5m$ de lado e deixou-se que o campo gravitacional e as interações partícula-partícula atuassem sobre o sistema. Além disso ainda há interações do tipo partícula-parede, pois o fluido está confinado entre paredes rígidas na forma de um quadrado com $1,0m$ de lado.

A Figura 11 mostra 12 imagens representativas do escoamento do fluido simulado. Foram empregadas 900 partículas, o que decorre do uso de um comprimento de suavização de $h = 2,2.10^{-2}m$ e espaçamento definido por $hh = h/1,3$. O passo de tempo utilizado foi $\Delta t = 10^{-4}s$.

O sistema dissipa sua energia e se aproxima bastante do equilíbrio a partir de 3 segundos de simulação, aos 5 segundos e meio não há mais nenhum movimento expressivo das partículas, apenas flutuações numéricas. A figura 12 motra o estado do sistema nestes dois instantes mencionados.

4.2.1 Modificação

Após a convergência do primeiro caso com uma distribuição inicial quadrada de fluido, foi feita uma sutil modificação na geometria para que esta se tornasse retangular. Então, as dimensões iniciais do bloco de fluido passaram a ser de $0,7m$ de altura e $0,4m$

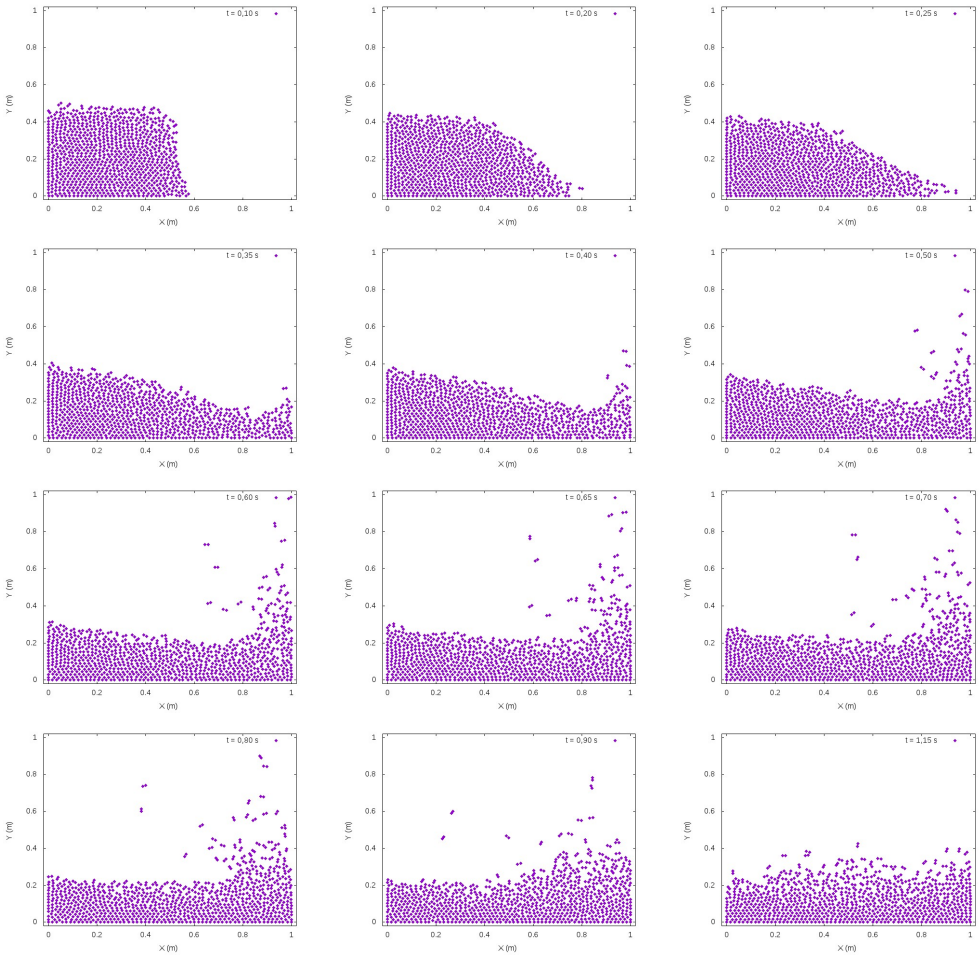

Figura 11 – Doze instantes representativos da simulação do rompimento de barragem. O passo de tempo empregado foi de $\Delta t = 10^{-4}s$. Possui 900 partículas e $h = 2,2.10^{-2}m$.

de comprimento.

Esta alteração resultou em uma maior velocidade atingida pelas partículas, pois a altura de queda passou a ser 40% maior que a anterior. Com isso, mantendo-se o valor de h e diminuindo o passo de tempo para $\Delta t = 10^{-5}s$, a solução divergiu após aproximadamente 2 segundos. A 13 mostra o estado da simulação em 3 momentos representativos.

Para que a solução convergisse, o comprimento de suavização foi reduzido para $h = 10^{-2}m$. O que aumenta muito a demanda computacional. Além disso o passo de tempo foi aumentado para o valor de referência $\Delta t = 10^{-4}s$, reduzindo um pouco o custo de processamento. Ainda assim a diferença de tempo para a execução dos cálculos foi bastante expressiva. A 14 mostra três instantes desta simulação, que ao fim se encontra no equilíbrio assim como o caso anterior.

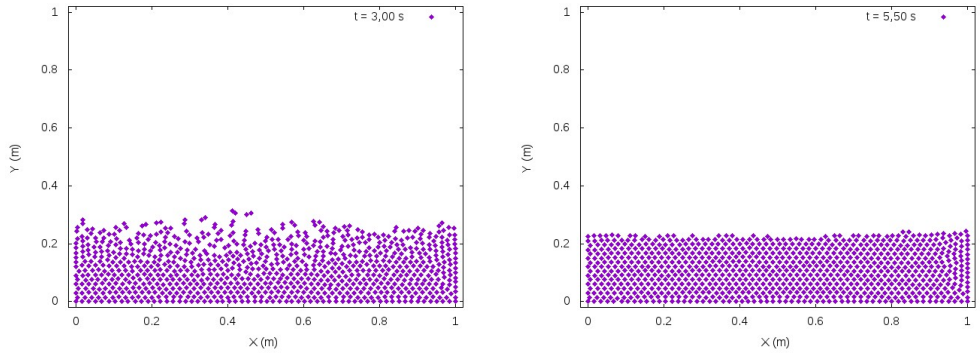

Figura 12 – Estado do sistema após passados 3 segundos, à esquerda, e 6 segundos, á direita.

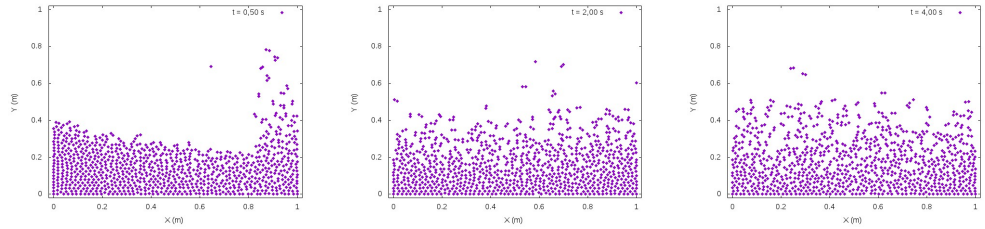

Figura 13 – Três instantes representativos da simulação do rompimento de barragem modificado. O passo de tempo empregado foi de $\Delta t = 10^{-5}s$. Possui 920 partículas e $h = 2,2.10^{-2}m$.

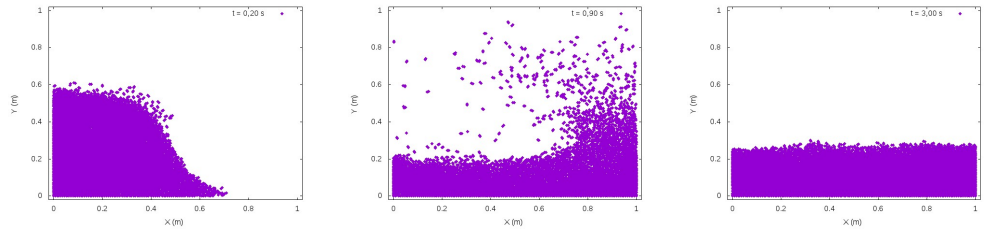

Figura 14 – Três instantes representativos da simulação do rompimento de barragem modificado. O passo de tempo empregado foi de $\Delta t = 10^{-4}s$. Possui 4732 partículas e $h = 10^{-2}m$.

4.3 Gota no Chão

Neste caso a geometria passa a ser circular, $0,25m$ de raio e suspensa a $0,30m$ de altura antes do momento inicial. As partículas estão sujeitas às mesmas interações do caso anterior. A 15 ilustra 6 momentos da simulação, que foi realizada com $h = 2,2.10^{-2}m$ e $\Delta t = 10^{-5}s$.

Assim como no caso anterior, aos 2 segundos a simulação já se aproxima do equilíbrio e aos 4 segundos não se observa mais movimentos das partículas além de flutuações numéricas. Isto pode ser observado na figura 16

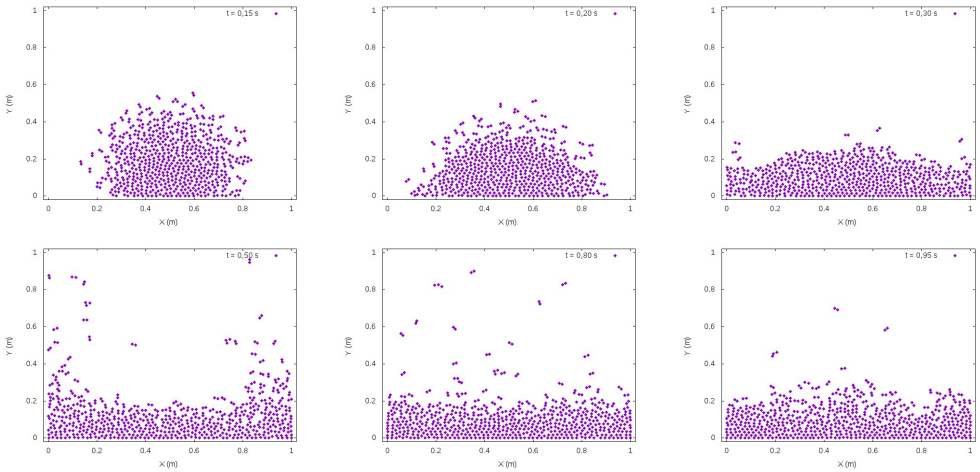

Figura 15 – Seis instantes representativos da simulação da colisão da gota de fluido com o solo. O passo de tempo empregado foi de $\Delta t = 10^{-5}s$. Possui 628 partículas e $h = 2,2.10^{-2}m$.

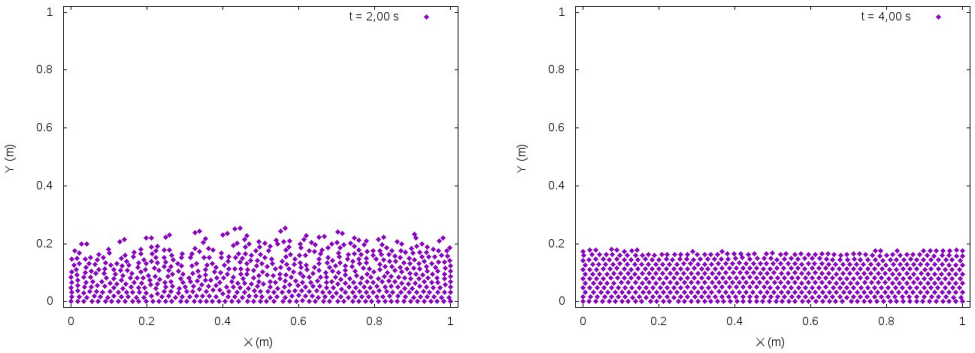

Figura 16 – Simulação da colisão de uma gota de fluido com o solo se aproxima do equilíbrio aos $2s$ e se estabiliza em $4s$.

4.4 Gota na Piscina

Com o intuito de pôr um pouco mais à prova a versatilidade do código SPH desenvolvido, o terceiro caso de estudo foi criado. Consiste basicamente na união dos dois casos anteriores. É preenchido por partículas de fluido um retângulo de $0,2m$ de altura e $2,0m$ de comprimento e também um círculo de fluido com $0,2m$ de raio suspenso à $0,8m$ de altura. Neste caso a posição das paredes rígidas são alteradas, formando agora um retângulo com $1,5m$ de altura e $2,0m$ de comprimento.

No primeiro teste realizado foi utilizado um comprimento de suavização foi de $h = 2,2.10^{-2}m$ e passo de tempo de $\Delta t = 10^{-5}s$. Com esta configuração a simulação divergiu a partir de aproximadamente 2 segundos, figura 17. Neste teste foram utilizadas 2339 partículas.

Para que a solução convirja é necessário aumentar o refinamento da discretização

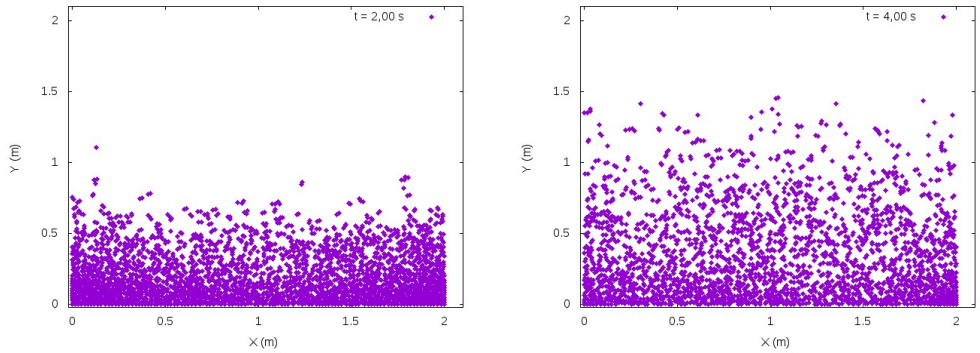

Figura 17 – Momento em que o primeiro teste da simulação da colisão de uma gota com uma poça de fluido diverge. Foram empregadas 2339 partículas, $h = 2,2.10^{-2}m$ e $\Delta t = 10^{-5}s$. À esquerda o estado para $t = 2,0s$ e à direita para $t = 4,0s$.

por partículas. Então o comprimento de suavização foi reduzido para $h = 10^{-2}m$ e o passo de tempo aumentado para $10^{-4}s$. Este aumento no passo de tempo não causa perda de qualidade na solução. Com esta nova configuração 12254 partículas compõem o sistema e foi possível levar a solução até o estado de equilíbrio, figura 18. É importante pontuar que o custo de processamento deste experimento é demasiadamente maior que os anteriores. Devido ao maior consumo de poder computacional a simulação foi encerrada após se identificar que a mesma já se encontrava bastante próxima do equilíbrio. O que ocorreu já a partir de $t = 1,90s$ e o encerramento se deu aos $2,84$ segundos.

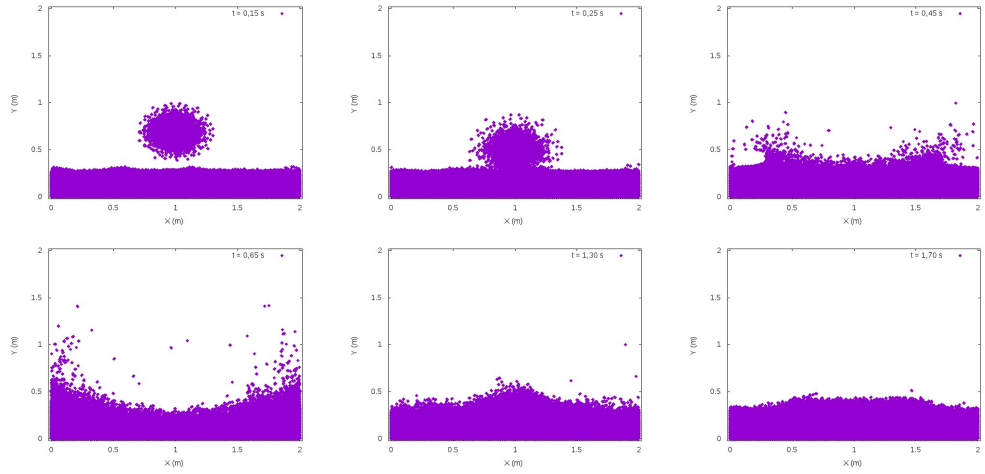

Figura 18 – Seis momentos da simulação da colisão de uma gota com uma poça de fluido. Foram empregadas 12254 partículas, $h = 10^{-2}m$ e $\Delta t = 10^{-4}s$.

É bastante provável que este experimento seja capaz de convergir fazendo uso de um número menor de partículas, ou seja, maior comprimento de suavização. Porém, para se observar este fato seria necessário a realização de mais testes. O que foge do escopo deste trabalho.

4.5 Colisão

O último experimento visa reproduzir o comportamento da colisão de dois escoamentos independentes que se encontram em determinado momento. A simulação consiste de dois retângulos de fluido nas duas extremidades do domínio computacional, o qual foi extendido para $2,0m$ de comprimento e $1,5m$ de altura. Cada bloco de fluido possui dimensão de $0,4m$ de comprimento e $0,9m$ de altura.

Foram realizados testes com diferentes refinamentos, valores de h, e todos com o mesmo passo de tempo, $10^{-4}s$. Os valores foram: $1,9.10^{-2}m$; $1,7.10^{-2}m$, $1,3.10^{-2}m$ e $1,1.10^{-2}m$. Consequentemente, o número de partículas empregadas foi de 3410, 4209, 7371 e 10165 respectivamente. Os três primeiros foram capazes de representar satisfatoriamente os instantes iniciais, no entanto a partir de aproximadamente $t = 1,50s$, $t = 1,80s$ e $t = 4,3s$ respectivamente, estes demonstraram não possuir resolução suficiente para representar os efeitos da agitação do fluido. Já o último mostrou-se bem mais adequado. Com isso, foi possível ver que o tempo total de simulação padrão de seis segundos não foram suficientes para visualizar o sistema se encaminhando para o equilíbrio. A turbulência resultante do impacto é bastante expressiva e a dissipação de energia devido à viscosidade e às colisões contra as paredes não são suficientee para amortecer todo o movimento dentro deste tempo. Os instantes iniciais da simulação são mostrados na figura 19.

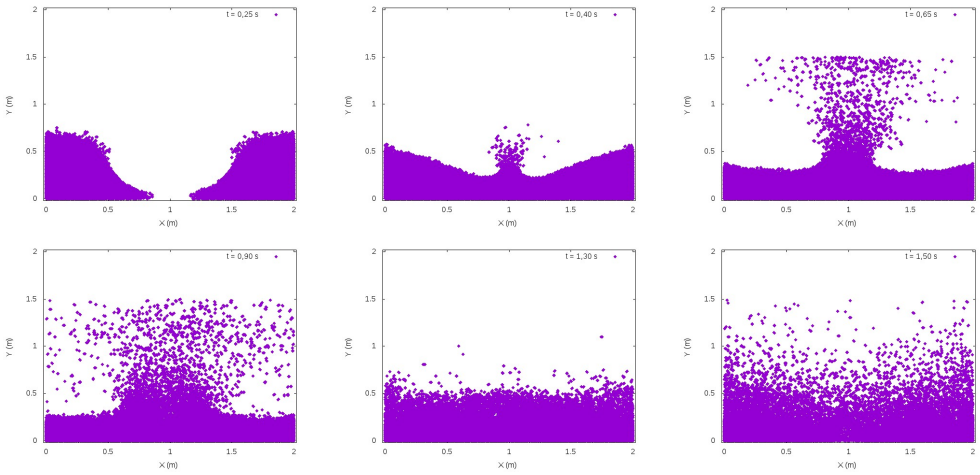

Figura 19 – Seis momentos da simulação da colisão entre duas correntes de fluido. Foram empregadas 10165 partículas, $h = 1,1.10^{-2}m$ e $\Delta t = 10^{-4}s$.

Este último teste, mais refinado, foi capaz de representar melhor a condição do fluido nos instantes finais da simulação. Esta qual foi encerrada com $\Delta t = 5,92s$, pois tornou-se claro que não se chegaria ao equilíbrio até os seis segundos predeterminados da simulação. O comportamento do fluido nestes segundos finais adquiriu um caráter oscilatório, aparentemente devido a alta agitação do fluido. No entanto não é possível afirmar que esta oscilação também não seja relacionada à questões numéricas. Os instantes

finais deste experimento estão representados na figura 20.

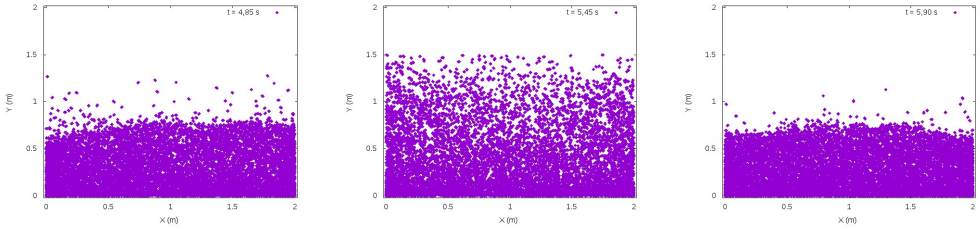

Figura 20 – Instantes finais da simulação da colisão entre duas correntes de fluido. Foram empregadas 10165 partículas, $h = 1,1.10^{-2}m$ e $\Delta t = 10^{-4}s$.

Devido à questões técnicas de implementação da versão atual do código só é possível iniciar simulações a partir do estado inicial, $\Delta t = 0,00s$. Somado a este fato está o enorme tempo de computação requerido para este experimento. Então, tornou-se inviável um novo teste.

4.6 Comprimento de Suavização e Espaçamento das Partículas

Os valores empregados para o comprimento de suavização (h) e espaçamento das partículas (hh) foram determinados por meio de experimentação, porém primeiro foram tomados como referência os valores fornecidos por (7). O parêmtro utilizado para avaliar o resultado gerado pelos valores escolhidos foi a densidade das partículas, pois quando esta "converge"para o valor de referência (em geral após o primeiro passo de tempo) os valores de h e hh são considerados adequados. Deve-se observar que a convergência da densidade não é literal, o que de fato ocorre é que esta se aproxima do valor de referência e então oscila em torno dele. Isto se dá pois deve ser permitida uma pequena compressibilidade do fluido, como já discutido na seção 3.1.1. Por fim, os valores utilizados para os testes gerais foram

$$\begin{cases} h &= 2,2.10^{-2}m, \\ hh &= h/1,3. \end{cases} \tag{4.1}$$

A definição do espaçamento das partículas se manteve constante para todos os experimentos, porém naqueles que divergiram nos primeiros testes o valor de h foi reduzido. Para a geometria retangular do experimento do rompimento de barragem e para o da gota de fluido numa poça foi utilizado o comprimento de suavização $h = 10^{-2}m$.

4.7 Passo de Tempo

Foram realizados testes com grandes variações do passo de tempo e encontrou-se que o valor proposto por (7) de fato é bastante próximo do ideal. A saber,

$$\Delta t = 10^{-4} s. \tag{4.2}$$

Tomando como referência o caso de geometria mais simples, o do rompimento da barragem, foi testado o uso de $\Delta t = 10^{-3} s$. Logo após o primeiro frame impresso a simulação claramente já havia divergido. A 4.7 abaixo representa o estado do sistema após $0,01 s$ de uma simulação realizada com $h = 2,2.10^{-2} m$. Nestes testes foram envolvidas 900 partículas.

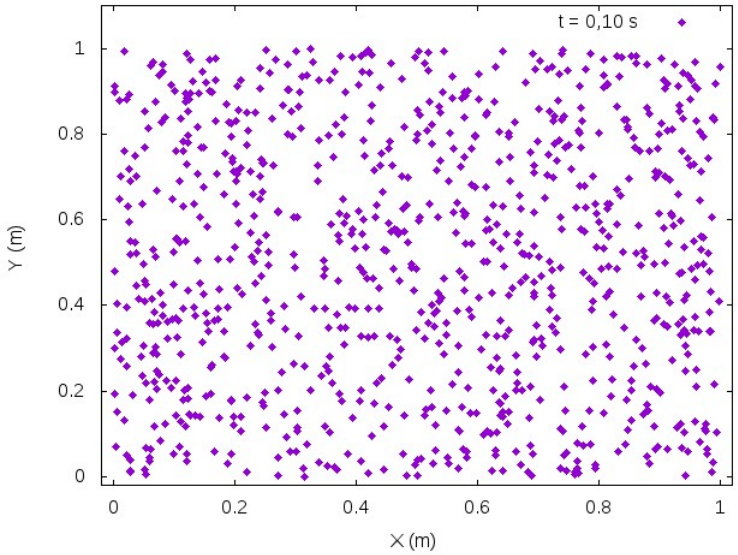

Figura 21 – Posição das partículas em $t = 0,01 s$ de simulação. Foram 900 partículas empregadas, $h = 2,2.10^{-2} m$ e $\Delta t = 10^{-3} s$.

Outro teste também realizado foi com um passo de tempo menor, $\Delta t = 10^{-5} s$. Não foi observada nenhuma alteração significativa no comportamento da solução com este passo de tempo 10 vezes menor que o de referência. Apesar de ser possível notar uma pequena diferença na "cauda"do escoamento, no decorrer da simulação não há alterações significativas. Comparação ilustrada pela figura 22.

Desbrun (6) faz comentários acera do uso de uma versão modificada do critério de estabilidade de Courant-Friedrichs-Lewy amplamente empregado em métodos como o de Diferenças Finitas. O critério diz que

$$Co = v\frac{\Delta t}{\Delta x} \leqslant 1. \tag{4.3}$$

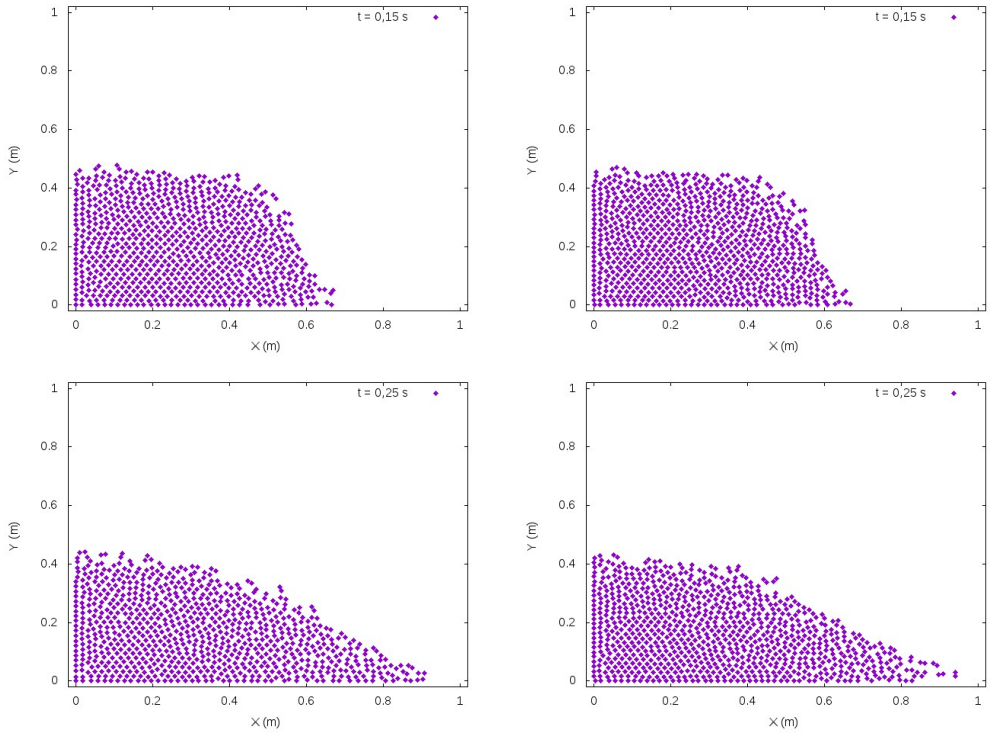

Figura 22 – Comparação entre o uso dos passos de tempo $\Delta t = 10^{-4}s$, à esquerda, e $\Delta t = 10^{-5}s$, à direita. Foram 900 partículas empregadas e $h = 2,2.10^{-2}m$.

Onde Δx é o menor comprimento de uma célula da malha e v é a maior velocidade atingidade.

Se for feita a substituição de v por c e Δx por h se obtém uma forma de se estimar, mesmo que de forma grosseira, o maior valor admissível para o passo de tempo no SPH. Esta é dada por

$$\Delta t < \frac{h}{c}. \tag{4.4}$$

Ao se aplicar os valores de h e c padrões usados nas simulações se obtém $\Delta t < 6,9.10^{-4}s$. O que corrobora os resultados encontrados. Isto é, com $\Delta t = 10^{-3}s$ de fato não deveria ser possível obter a convergência da simulação e também, pelo menos em uma primeira análise, é desnecessária a utilização de $\Delta t < 10^{-4}s$ para as configurações atuais das simulações.

Por fim, devido ao significativo aumento da demanda de tempo de computação para os casos em que se mostrou necessário usar um maior refinamento de partículas, foram realizados testes com o intuito de avaliar a possibilidade de utilização de passos de tempo maiores, $\Delta t > 10^{-4}s$, para que então o custo computacional fosse reduzido.

Os testes foram realizados com experimento modificado do rompimento de barragem, seção 4.2.1, e com o da gota no chão, seção 4.3. E foi encontrado que o passo de tempo

pode ser aumentado sem prejuízos até o valor referente a $Co \approx 0,58$.

4.8 Módulo de Elasticidade Artificial

O parâmetro k utilizado nas simulações, que é diretamente relacionado a velocidade do som artificial da equação de estado, é interpretado por (6) como um parâmetro que determina a rigidez do sistema de partículas. O seu valor influencia de forma linear a magnitude do termo da equação do momento linear referente às forças de pressão. Monaghan (16), comenta sobre o uso de $c \sim 10m/s$ como o valor da velocidade do som artificial. Então, $k \sim 100m^2/s^2$. Ainda segundo Monaghan, isto implicaria em uma flutuação de aproximadamente 1% da densidade do fluido.

Em todas as simulações referentes aos resultados apresentados até aqui foi empregado o valor de $k = 10^3 J/Kg$. Não foi empregado nenhum marcador para monitorar os valores máximos e mínimos das densidades das partículas a cada espaço de tempo. No entanto é possível fazer uma análise visual do arquivo de saída de dados e foi possível observar que houve uma variação de 4% em torno do valor de referência da densidade. O que é consideravelmente maior que o valor previsto por Monaghan. Apesar deste resultado, a simulação se deu normalmente.

4.9 Instabilidade inicial

Em todos os casos de estudo, uns mais que os outros, é possível identificar uma instabilidade nas partículas logo nos primeiros centésimos de segundo das simulações, figura 23. Note que as partículas das bordas são atraídas para dentro nestes instantes.

As instabilidades ficam mais evidentes nos casos envolvendo formas circulares, figura 24. Onde é possível notar que uma fina camada mais externa de partículas é impulsionada para fora. O que gera uma perda da geometria circular inicial. No entanto é visível que este efeito influencia significativamente uma pequena fração do total de partículas, que após o impacto passam a escoar em consonância com o restante.

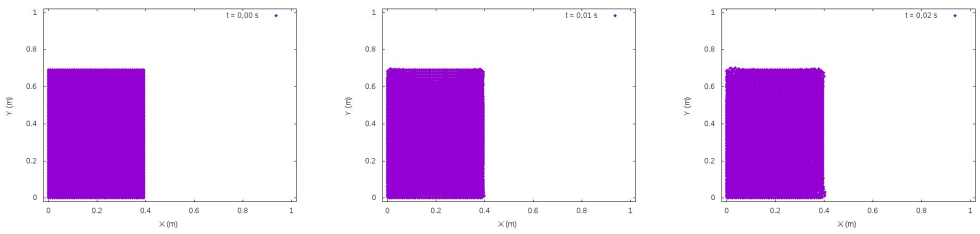

Figura 23 – Leve instabilidade inicial das partículas no caso do rompimento de barragem.

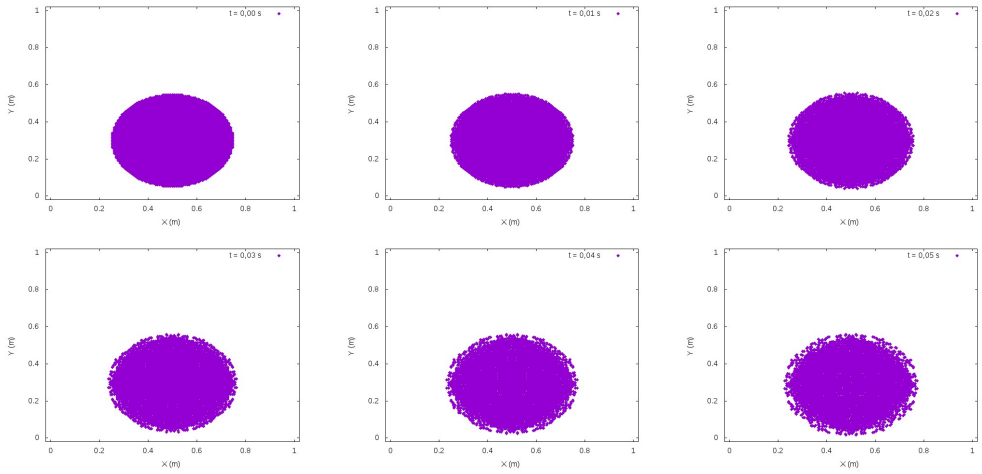

Figura 24 – Instabilidade inicial das partículas no caso da queda de uma gota no chão.

O uso da equação 3.5 para o cálculo da densidade e o posicionamento regular das partículas é o que causa esta instabilidade. Esta forma de atualização da densidade sofre com a deficiência de vizinhos nas fronteiras, resultando em uma densidade menor para as partículas ali localizadas. O que gera um gradiente de pressão. Além disso, nada garante que o posicionamento das partículas através de uma malha regular construa um sistema em equilíbrio. É inerente à esta disposição o surgimento de gradientes de pressão que, ao iniciar a simulação, produzam forças entre partículas que as movam de forma inesperada nos primeiros instantes.

Um experimento bem simples pode ser realizado para ilustrar este fato. As paredes rígidas são aproximadas para formar um quadrado com $0.5m$ de lado e pequenas alterações são feitas na classe "BOX"para que ela construa um retângulo de fluido com $0.5m$ de comprimento e $0.3m$ de altura. E então é deixado agir sobre este sistema as forças de pressão, viscosidade e gravidade, figura 25. Este procedimento é basicamente o que (16) descreve para amortecer o sistema até o estado de equilíbrio antes de se iniciar a simulação.

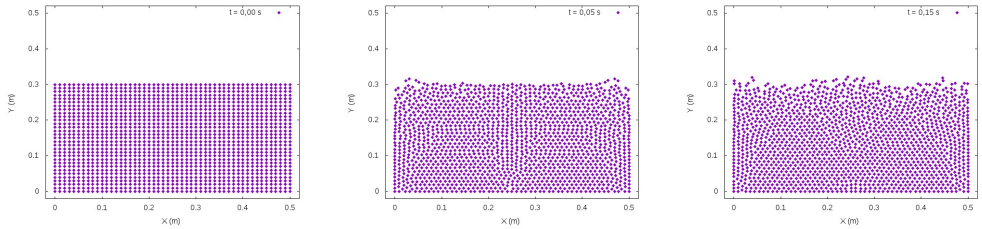

Figura 25 – Oscilação da interface do fluido devido às instabilidades iniciais. Passo de tempo padrão de $\Delta t = 10^{-4}s$ e comprimento de suavização $h = 1,3.10^{-2}m$.

Neste experimento foi usado o passo de tempo padrão de $10^{-4}s$ e comprimento de suavização $h = 1,3.10^{-2}m$, o que resultou em 1581 partículas. É possível observar claramente como o sistema é perturbado e oscila devido às tensões internas geradas

pela deficiência de vizinhos nas fronteiras e pela distribuição inicial regular de partículas. Após aproximadamente 0,7 segundos o sistema entra em equilíbrio nos termos já ditos anteriormente, como pode ser visto na figura 26.

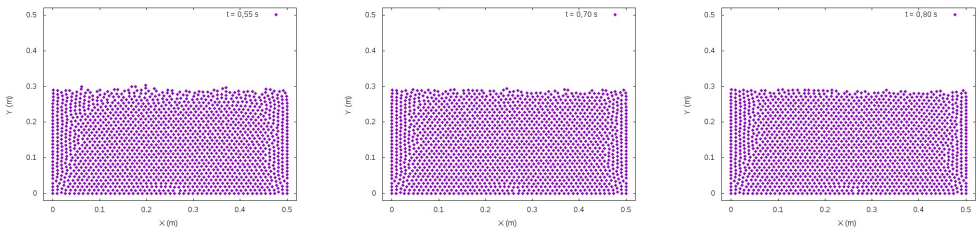

Figura 26 – Estado do sistema para $t = 0,55s$, $t = 0,70s$ e $0,80s$ da esquerda para a direita. Passo de tempo padrão de $\Delta t = 10^{-4}s$ e comprimento de suavização $h = 1,3.10^{-2}m$.

Por questões de implementação no código e necessidade de mais testes esta rotina de inicialização das simulações não foi incorporada ao código desenvolvido. Para o caso de simulações simples, como as que foram propostas, estas flutuações numéricas nos instantes iniciais não influenciam no resultado final.

4.10 Animações

Uma melhor visualização das simulações como um todo pôde ser obtida através da conversão da sequência de imagens em uma animação. Isto foi realizado através do processador de imagem GIMP, que é um software livre de código aberto. Portanto, foram geradas animações para os quatro experimentos incluindo a modificação do Rompimento de Barragem e para o experimento demonstrativo das instabilidades iniciais.

O conteúdo completo das animações está disponível na internet e pode ser acessado através dos dois códigos do tipo "QR"abaixo, figura 27. O código à esquerda é mais indicado para o caso de uma conexão mais lenta com a internet ou pacote de dados reduzidos, já o à direita o oposto.

Figura 27 – Códigos do tipo "QR"para compartilhamento das animações. À esquerda o ideal para conexões mais lentas e, http://l.ead.me/bauILl, e à direita o oposto, http://l.ead.me/bauH2i.

5 Considerações Finais e Conclusão

Neste trabalho foram apresentados as principais diferenças conceituais entre os métodos que dependem de uma malha, que são os chamados "tradicionais", e os que não são dependentes e as vantagens e desvantagens de cada um. Tendo como foco um dos principais representantes da classe dos métodos sem malha, o Smoothed Particle Hydrodynamics, este teve seu formalismo básico exposto em detalhes. Dessa forma é possível garantir que um leitor de graduação de qualquer curso da área de ciências exatas, com um conhecimento básico prévio de mecânica dos fluidos, será capaz de entender como as aproximações são realizadas e como se dá o avanço do tempo neste método numérico.

O código desenvolvido em C++ é capaz de resolver problemas simples de hidrodinâmica sendo escolhidos quatro casos que se diferem pela geometria inicial das partículas. A escolha do caso simulado é permitida ao usuário em um menu inicial. Os parâmetros básicos da simulação, dimensões das fronteiras e das geometrias podem ser facilmente alteradas na parte inicial do código para o caso de se realizar novos testes. Não foi implementado nenhum algoritmo sofisticado para a busca de vizinhos, tendo sido aproveitada apenas a simetria dos operadores para reduzir a complexidade do algoritmo de $O(n^2)$ para $O(n(n-1)/2)$. Então, fica para trabalhos futuros a implementação de listas encadeadas para a busca de vizinhos, que é um algoritmo de otimização comumente empregado na literatura. Além disso, outras melhorias que podem ser feitas futuramente ao código se referem a implementação de um algoritmo de inicialização de partículas para reduzir oscilações devido a tensões internas que decorrem do posicionamento inicial. Ao uso de partículas fantasmas para o tratamento de fronteiras e também a implementação de métodos de integração que suportem passo de tempo variável. Devido aos núcleos de suavização empregados não foi necessária a utilização de algoritmos para a prevenção de formação de aglomerados de partículas e nem para a estabilização da simulação.

Os dados obtidos das simulações dos quatro casos implementados, incluindo modificações e demais testes, foram transformados em gráficos da posição das partículas no plano xy para cada intervalo de tempo predeterminado. Com isso foi possível observar a evolução temporal das partículas individuais e como um todo, representando o movimento de um fluido sob as influências do campo gravitacional, das interações entre partícualas e das fronteiras rígidas. Os gráficos foram gerados através do software livre Gnuplot e o processo foi automatizado por meio de um Shell script. Em seguida as imagens foram unidas na forma de uma animação dentro de outro software livre, o GIMP. Com isso foi possível visualizar os resultados de uma forma bastante natural, observando o comportamento do fluido ao longo do tempo para cada caso.

O método SPH se difundiu no meio acadêmico e se mostrou capaz de suprir diversas lacunas deixadas pelos métodos numéricos tradicionais. Muito de sua teoria formal ainda não é devidamente compreendida, porém o esforço para tal é crescente. O que tem levado ao surgimento de uma grande quantidade de adaptações, correções e otimizações para o modelo proposto em 1977. A plena aplicação industrial deste método ainda não é realidade, porém caminha consistentemente para tal. A disseminação da técnica SPH tão acelerada dentro de tão variadas áreas da ciência e da indústria é devido a sua capacidade de lidar com geometrias complexas, grandes deformações, escoamentos de alta velocidade, tratamento de superfícies livres e interfaces fluidas, modelagem e interação de diferentes materiais, fenômenos multifísicos e acoplamento com outros métodos. Entretanto, apesar do cenário promissor, o avanço esbarra em questões numéricas cruciais que permanecem em aberto tais como convergência, estabilidade, custo computacional e tratamento de fronteiras.

Referências

1 Nenhuma citação no texto.

2 T. Belytschko, Y. Krongauz, D. Organ, M. Fleming, and P. Krysl. Meshless methods: An overview and recent developments. *Computer methods in applied mechanics and engineering*, 1996. Citado 3 vezes nas páginas 21, 22 e 25.

3 W. E. Boyce and R. C. DiPrima. *Elementary Differential Equations and Boundary Value Problems*. John Wiley & Sons, Inc. Citado na página 20.

4 J. K. Chen and J. E. Beraun. A generalized smoothed particle hydrodynamics method for nonlinear dynamic problems. *Computer methods in applied mechanics and engineering*, 2000. Citado na página 30.

5 P. W. Cleary and J. J. Monaghan. Conduction modelling using smoothed particle hydrodynamics. *Journal of Computational Physics*, 1998. Citado na página 38.

6 M. Desbrun and M. Cani. Smoothed particle: A new paradigm for animating highly deformable bodies. *Eurographics Workshop on Computer Animation and Simulation*, 1996. Citado 6 vezes nas páginas 46, 49, 50, 52, 69 e 70.

7 B. Fall. Applications of parallel computers. 2011. Citado 6 vezes nas páginas 41, 49, 50, 55, 56 e 68.

8 Fox, McDonald, and Pritchard. *Introdução à Mecânica dos Fluidos*. gen LTC. Citado 2 vezes nas páginas 12 e 43.

9 R. A. Gingold and J. J. Monaghan. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Royal Astronomical Society*, 1977. Citado 2 vezes nas páginas 15 e 32.

10 J. Liberty and B. Jones. *Teache Yourself C++ in 21 Days*. 2005. Citado 2 vezes nas páginas 53 e 59.

11 G. R. Liu and M. B. Liu. *Smoothed Particle Hydrodynamics: a meshfree particle method*. World Scientific Publishing Co. Pte. Ltd. Citado 12 vezes nas páginas 13, 14, 16, 22, 35, 39, 40, 44, 45, 55, 56 e 61.

12 G. R. Liu, M. B. Liu, and K. Y. Lam. Constructing smoothing funnctions in smoothed particle hydrodynamics with applications. *Journal of Computational and Applied Mathematics*, 2003. Citado 3 vezes nas páginas 23, 28 e 32.

13 L. B. Lucy. A numerical approach to the testing of the fission hypothesis. *The Atronomical Journal*, 1977. Citado na página 15.

14 J. J. Monaghan. An introduction to sph. *Computer Physics Communications*, 1988. Citado 5 vezes nas páginas 36, 38, 44, 48 e 56.

15 J. J. Monaghan. Smoothed particle hydrodynamics. *Annual Review Astronomy and Astrophysics*, 1992. Citado 3 vezes nas páginas 45, 48 e 55.

16 J. J. Monaghan. Simulating free surface flow with sph. *Journal of Computational Physics*, 1994. Citado 6 vezes nas páginas 45, 46, 55, 61, 70 e 72.

17 J. J. Monaghan. Smoothed particle hydrodynamics. *Reports on Progress in Physics*, 2005. Citado 4 vezes nas páginas 37, 38, 42 e 48.

18 J. J. Monaghan. Smoothed particle hydrodynamics and its diverse applications. *Annual Review Fluid Mechanics*, 2012. Citado 2 vezes nas páginas 56 e 61.

19 J. P. Morris, P. J. Fox, and Y. Zhu. Modeling low reynolds number incompressible flow using sph. *Journal of Computational Physics*, 1997. Citado 2 vezes nas páginas 45 e 46.

20 M. Muller, D. Charypar, and M. Gross. Particle-based fluid simulation for interactive applications. *The Eurographics Association*, 2003. Citado 3 vezes nas páginas 45, 49 e 52.

21 R. Ortiz, J. L. Charles, and J. F. Sobry. Structural loading of a complete aircraft under realistic crash conditions: generation of a load database for passenger safety and innovative design. *International Congress of the Aeronautical Sciences*, 2004. Citado na página 16.

22 M. S. Shadloo, G. Oger, and D. Le Touzé. Smoothed particle hydrodynamics methor for fluids flows, towards industrial applications: Motivation, current state and challenges. *Computers and Fluids*, 2016. Citado 2 vezes nas páginas 16 e 17.

23 Adriano Sueke Takata. Aspectos teórico-numéricos dos métodos sph e mps, 2015. Citado 3 vezes nas páginas 22, 39 e 40.

Apêndices

APÊNDICE A – Código Fonte

```
#include<iostream>
#include<fstream>
#include<cstring>
#include<string>
#include<math.h>

using namespace std;

const float pi = 3.141592653;

const float DAMP = 0.75;

// ***** Definir dimensões do domínio ***** //
const float XMIN = 0.0;
const float XMAX = 1.0; // 2.0
const float YMIN = 0.0;
const float YMAX = 1.0; // 1.5
// ************************************* //


// ***** Definir tamanho da repreza ***** //
const float Alt_BOX = 0.5; // 0.7 0.9 0.2
const float Comp_BOX = 0.5; // 0.4 0.4 XMAX
// ************************************* //

// ***** Definir raio da gota
//    e altura do centro **** //
const float rDrop = 0.25; // 0.2
const float hDrop = 0.3; // 0.8
// ************************************* //

enum DIRECTION {VERTICAL, HORIZONTAL};

class Simulation
{
```

```
public:
Simulation();
~Simulation()
{cout << "Simulation - Destructor called" << endl;}
//Accessors
int GetFrame() {return nframes;}
int GetStFrame() {return npframes;}
float Geth() {return h;}
float Gethh() {return hh;}
float Getrho0() {return rho0;}
float Getg() {return g;}
float Getk() {return k;}
float Getmu() {return mu;}
double Getdt() {return dt;}

void SetStFrame(int sframe) {npframes = sframe;}
void SetFrame(int frame) {nframes = frame;}

fstream SimLog;

private:
int nframes;
int npframes;
float h0;
float hh;
float h;
double dt;
float rho0;
float k;
float mu;
float g;
};

Simulation::Simulation ():
nframes(600),
npframes(100),  // Resulta num writeInterval de 0,1s
h0(2.1e-2),
hh((h0)/1.3),
h(2*h0),
```

```
dt(1e-4),
rho0(1000),
k(1.5e3),
mu(0.1),
g(9.8)
{cout << "Simulation - Constructor called" << endl;}

class Particle
{
public:
Particle ();
~Particle () {}

float rho;
float x[2];
float vh[2];
float v[2];
float a[2];

static int GetNumber() {return NumberOfParticles;}
static float Getmass() {return mass;}

//
static void NormalizeMass
      (Particle * Fluid, Simulation & rS);

private:

static int NumberOfParticles;
static float mass;
};

Particle::Particle ():
rho(0)
{
int i;
for (i = 0; i < 2; i++)
{
x[i] = 0; vh[i] = 0;
```

```
v[i] = 0; a[i] = 0;
}
NumberOfParticles++;
}

void Particle::NormalizeMass
(Particle * Fluid, Simulation & rS)
{
Particle * pfluid = Fluid;

float rho0 = rS.Getrho0();
float rho2s = 0;
float rhos = 0;

int i;
for (i = 0; i < Particle::GetNumber(); i++)

{
rho2s += (pfluid[i].rho) *(pfluid[i].rho);
rhos += (pfluid[i].rho);
}
Particle::mass *= (rho0*rhos / rho2s);

cout << "\n\n rhos = " << rhos << "\t";
cout << "rho2s = " << rho2s << endl;
cout << "(rhos/rho2s) = " << rhos/rho2s << endl;
}

int Particle::NumberOfParticles = 0;
float Particle::mass = 1;

class Geometry // Pure Virtual
{
public:
Geometry()
{cout << "Geometry Constructor Called" << endl;}
virtual ~Geometry()
{cout << "Geometry Destructor Called" << endl;}
```

```
virtual int ComputeDomain (Simulation &) = 0;
virtual int Indicator(float, float) = 0;
virtual void ParticlePlacement
(Particle *, Simulation &) = 0;

private:
// No Pivate
};

void Geometry::ParticlePlacement
(Particle * Fluid, Simulation & rS)
{
int i = 0;
float x, y;
float hh = rS.Gethh();
for (x = XMIN; x < XMAX; x+=hh)
{
for (y = YMIN; y < YMAX; y+=hh)
{
if (Indicator(x, y))
{
Fluid[i].x[0] = x;
Fluid[i].x[1] = y;
i++;
}
}
}
}

class Box : public Geometry
{
public:
Box (){}
~Box (){}
int Indicator (float x, float y)
{
return (x < Comp_BOX) && (y < Alt_BOX);
}
int ComputeDomain (Simulation & );
```

```
void ParticlePlacement (Particle *, Simulation &);

private:
// No Private
};

int Box::ComputeDomain (Simulation & S)
{
int count = 0;
float x, y;
float hh = S.Gethh();
for (x = XMIN; x < XMAX; x+=hh)
for (y = YMIN; y < YMAX; y+=hh)
count += Indicator(x, y);
return count;
}

void Box::ParticlePlacement
(Particle * Fluid, Simulation & rS)
{
Geometry::ParticlePlacement(Fluid, rS);
}


class Circle : public Geometry
{
public:
Circle (){}
~Circle (){}
int Indicator (float x, float y);
int ComputeDomain (Simulation & );
void ParticlePlacement (Particle *, Simulation &);

private:
// No Private
};
int Circle::Indicator (float x, float y)
{
float dx = (x-(XMAX/2));
```

```
float dy = (y-hDrop);
float r2 = dx*dx + dy*dy;
return (r2 < rDrop*rDrop);
}

int Circle::ComputeDomain (Simulation & S)
{
int count = 0;
float x, y;
float hh = S.Gethh();
for (x = XMIN; x < XMAX; x+=hh)
for (y = YMIN; y < YMAX; y+=hh)
count += Indicator(x, y);
return count;
}

void Circle::ParticlePlacement
(Particle * Fluid, Simulation & rS)
{
Geometry::ParticlePlacement(Fluid, rS);
}

class DropInThePool : public Geometry
{
public:
DropInThePool (){}
~DropInThePool (){}
int Indicator (float x, float y);
int ComputeDomain (Simulation & );
void ParticlePlacement (Particle *, Simulation &);

private:
// No Private
};

int DropInThePool::Indicator (float x, float y)
{
float dx = (x-(XMAX/2));
float dy = (y-hDrop);
```

```
float r2 = dx*dx + dy*dy;
if (r2 < rDrop*rDrop)
return 1;
if ((x < Comp_BOX) && (y < Alt_BOX))
return 1;
else
return 0;
}

int DropInThePool::ComputeDomain (Simulation & S)
{
int count = 0;
float x, y;
float hh = S.Gethh();
for (x = XMIN; x < XMAX; x+=hh)
for (y = YMIN; y < YMAX; y+=hh)
count += Indicator(x, y);
return count;
}

void DropInThePool::ParticlePlacement
(Particle * Fluid, Simulation & rS)
{
int i = 0;
float x, y;
float hh = rS.Gethh();
for (x = XMIN; x < XMAX; x+=hh)
{
for (y = YMIN; y < YMAX; y+=hh)
{
if (Indicator(x, y))
{
Fluid[i].x[0] = x;
Fluid[i].x[1] = y;
i++;
}
}
}
}
```

```
class Colision : public Geometry
{
public:
Colision (){}
~Colision (){}
int Indicator (float x, float y);
int ComputeDomain (Simulation & );
void ParticlePlacement (Particle *, Simulation &);

private:
// No Private
};

int Colision::Indicator (float x, float y)
{
if (x < Comp_BOX || x > (XMAX - Comp_BOX))
{
if (y < Alt_BOX)
return 1;
}
return 0;
}

int Colision::ComputeDomain (Simulation & S)
{
int count = 0;
float x, y;
float hh = S.Gethh();
for (x = XMIN; x < XMAX; x+=hh)
for (y = YMIN; y < YMAX; y+=hh)
count += Indicator(x, y);
return count;
}

void Colision::ParticlePlacement
(Particle * Fluid, Simulation & rS)
{
Geometry::ParticlePlacement(Fluid, rS);
```

```
}


class Solver
{
public:

Solver(){}
~Solver(){}

void DensityStart (Particle * Fluid, Simulation & rS)
{ComputeDensity(Fluid, rS);}
void IterationStart(Particle *, Simulation &, char *, int);
void Iteration(Particle *, Simulation &, char *, int);

private:

void ComputeDensity (Particle *, Simulation  &);
void ComputeForces (Particle *, Simulation  &);

void DampReflect
(DIRECTION, float, float *, float *, float *);
void ReflectBC (Particle *, int);

void LeapfrogStart (Particle *, double, int);
void LeapfrogStep (Particle *, double, int);

void WriteFrameData (fstream &,  Particle *, float);
};

void Solver::ComputeDensity
(Particle * Fluid, Simulation & rS)
{
int n = Particle::GetNumber();

Particle * pfluid = Fluid;

float h = rS.Geth();
float h2 = h*h;
```

```
float h8 = (h2*h2)*(h2*h2);
float C = 4*(Particle::Getmass())/(pi*h8);

int i = 0, j = 0;
float dx, dy, r2, z, rho_ij;

for (i = 0; i < n; i++)
pfluid[i].rho = 0;

for (i = 0; i < n; i++)
{
pfluid[i].rho += 4*(Particle::Getmass())/(pi*h2);

for (j = i+1; j < n; j++)
{
dx = pfluid[i].x[0] - pfluid[j].x[0];
dy = pfluid[i].x[1] - pfluid[j].x[1];
r2 = dx*dx + dy*dy;
z = h2 - r2;
if (z > 0)
{
rho_ij = C*z*z*z;
pfluid[i].rho += rho_ij;
pfluid[j].rho += rho_ij;
}
}
}
}

void Solver::ComputeForces
(Particle * Fluid, Simulation & rS)
{
ComputeDensity(Fluid, rS);

int i, j, n = Particle::GetNumber();
float dx, dy, r2, q, u, w0, wp, wv, dvx, dvy;

for (i = 0; i < n; i++)
{
```

```
(Fluid[i].a[0]) = 0;
(Fluid[i].a[1]) = -rS.Getg();
}

float h = rS.Geth();
float h2 = (h) * (h);
float h4 = (h2) * (h2);
float C0 = Particle::Getmass() / (pi * h4);
float Cp = 15 * (rS.Getk());
float Cv = -40 * (rS.Getmu());

for (i = 0; i < n; i++)
{
for (j = i+1; j < n; j++)
{
dx = (Fluid[i].x[0]) - (Fluid[j].x[0]);
dy = (Fluid[i].x[1]) - (Fluid[j].x[1]);
r2 = dx*dx + dy*dy;
if (r2 < h2)
{
q = sqrt(r2)/h;
u = 1-q;
w0 = C0 * u/((Fluid[i].rho)*((Fluid[j].rho)));
wp = w0 * Cp;
wp *= ((Fluid[i].rho)+(Fluid[j].rho)-2*rS.Getrho0());
wp *= u/q;
wv = w0 * Cv;
dvx = (Fluid[i].v[0]) - (Fluid[j].v[0]);
dvy = (Fluid[i].v[1]) - (Fluid[j].v[1]);
(Fluid[i].a[0]) += (wp*dx + wv*dvx);
(Fluid[i].a[1]) += (wp*dy + wv*dvy);
(Fluid[j].a[0]) -= (wp*dx + wv*dvx);
(Fluid[j].a[1]) -= (wp*dy + wv*dvy);
//
}
}
}
}
```

```
void Solver::DampReflect
(DIRECTION direction, float barrier,
float * x, float * v, float * vh)
{
if (v[direction] == 0)
return;

float tbounce = (x[direction]-barrier)/v[direction];
x[0] -= v[0]*(1-DAMP)*tbounce;
x[1] -= v[1]*(1-DAMP)*tbounce;

x[direction] = 2*barrier - x[direction];
v[direction] = -v[direction];
vh[direction] = -vh[direction];

v[0] *= DAMP; vh[0] *= DAMP;
v[1] *= DAMP; vh[1] *= DAMP;
}

void Solver::ReflectBC (Particle * Fluid, int choice)
{
int n = Particle::GetNumber();
int i;


for (i = 0; i < n; i++)
{
if (Fluid[i].x[0] < XMIN)
DampReflect(VERTICAL, XMIN,Fluid[i].x,
Fluid[i].v, Fluid[i].vh);
//
if (Fluid[i].x[0] > XMAX)
DampReflect(VERTICAL, XMAX, Fluid[i].x,
Fluid[i].v, Fluid[i].vh);
//
if (Fluid[i].x[1] < YMIN)
DampReflect(HORIZONTAL, YMIN, Fluid[i].x,
Fluid[i].v, Fluid[i].vh);
//
```

```
if (Fluid[i].x[1] > YMAX)
DampReflect(HORIZONTAL, YMAX, Fluid[i].x,
Fluid[i].v, Fluid[i].vh);
}
}


void Solver::LeapfrogStart
(Particle * Fluid, double dt, int choice)
{
int i, n = Particle::GetNumber();

for (i = 0; i < n; i++)
{
Fluid[i].vh[0] = Fluid[i].v[0] + (Fluid[i].a[0]) * dt/2;
Fluid[i].vh[1] = Fluid[i].v[1] + (Fluid[i].a[1]) * dt/2;



}
for (i = 0; i < n; i++)
{
Fluid[i].v[0] += (Fluid[i].a[0]) * dt;
Fluid[i].v[1] += (Fluid[i].a[1]) * dt;
}
for (i = 0; i < n; i++)
{
Fluid[i].x[0] += (Fluid[i].vh[0]) * dt;
Fluid[i].x[1] += (Fluid[i].vh[1]) * dt;
}
ReflectBC(Fluid, choice);
}

void Solver::LeapfrogStep
(Particle * Fluid, double dt, int choice)
{
int i;
for (i = 0; i < Particle::GetNumber(); i++)
{
Fluid[i].vh[0] += (Fluid[i].a[0]) * dt;
```

```
Fluid[i].vh[1] += (Fluid[i].a[1]) * dt;
}
for (i = 0; i < Particle::GetNumber(); i++)
{
Fluid[i].v[0] = Fluid[i].vh[0] + (Fluid[i].a[0]) * dt/2;
Fluid[i].v[1] = Fluid[i].vh[1] + (Fluid[i].a[1]) * dt/2;
}
for (i = 0; i < Particle::GetNumber(); i++)
{
Fluid[i].x[0] += (Fluid[i].vh[0]) * dt;
Fluid[i].x[1] += (Fluid[i].vh[1]) * dt;
}
ReflectBC(Fluid, choice);
}

void Solver::WriteFrameData
(fstream & stream, Particle * Fluid, float time)
{
stream << "# Particles Mass m = ";
stream << Particle::Getmass() << endl;
int i;
stream << "#\n#\t" << "dt(s)\t" << "X(x)\t" << "X(y)\t";
stream << "V(x)\t" << "V(y)\t"<< "Rho" << endl;

for (i = 0; i < Particle::GetNumber(); i++)
{
stream << "\t" << time;
stream << "\t" << Fluid[i].x[0] << "\t" << Fluid[i].x[1];
stream << "\t" << Fluid[i].v[0] << "\t" << Fluid[i].v[1];
stream << "\t" << Fluid[i].rho << endl;
}
}

void Solver::IterationStart
(Particle * Fluid, Simulation & rS,
char * FileName, int choice)
{
cout << "DEBUG FileName is: " << FileName << endl;
```

```
(rS.SimLog).open(FileName, ios::out);

if ((rS.SimLog).is_open())
{
WriteFrameData(rS.SimLog, Fluid, 0);

ComputeForces(Fluid, rS);
LeapfrogStart(Fluid, rS.Getdt(), choice);
}
else
cout << "File is not opened" << endl;

(rS.SimLog).close();


FileName[3] += 1;

cout << "Atualização de FileName\n";
cout << "DEBUG FileName is: " << FileName << endl;
}

void Solver::Iteration
(Particle * Fluid, Simulation & rS,
char * FileName, int choice)
{
int frame, step;

for (frame = 1; frame < rS.GetFrame(); ++frame)
{
for (step = 0; step < rS.GetStFrame(); ++step)
{
ComputeForces(Fluid, rS);
LeapfrogStep(Fluid, rS.Getdt(), choice);
}

cout << "DEBUG FileName is: " << FileName << endl;
(rS.SimLog).open(FileName, ios::out);

if ((rS.SimLog).is_open())
```

```
{
WriteFrameData(rS.SimLog, Fluid,
(rS.GetStFrame()*rS.Getdt()*frame));
}
else
cout << "File is not opened" << endl;
(rS.SimLog).close();


if (FileName[3] == '9')
{
FileName[3] = '0';
if (FileName[2] == '9')
{
FileName[2] = '0';
FileName[0] += 1;
}
else
FileName[2] += 1;
}
else
FileName[3] += 1;

cout << "Atualização de FileName\n";
cout << "DEBUG FileName is: " << FileName << endl;
}
}

int main ()
{
char FileName[] = "0.00";

Simulation S;
Solver SPH;
Geometry * pGeometry;

short int key = 2; // Basta ser diferente de 0 ou 1.
short int choice;
int N;
```

```
while (key)
{
cout << "\n(1) for a BOX; (2) for a CIRCLE; ";
cout << "(3) for a Drop In The Pool; ";
cout << "(4) for a Colision" << endl;
cout << "(0) To QUIT" << endl;

cin >> choice;
switch (choice)
{
case 0:
cout << "Quitting...." << endl;
key = 0;
return 0;
// break;
// The Quitting Feature is not fully developed yet
case 1:
pGeometry = new Box;
key = 0;
break;
case 2:
pGeometry = new Circle;
key = 0;
break;
case 3:
pGeometry = new DropInThePool;
key = 0;
break;
case 4:
pGeometry = new Colision;
key = 0;
break;
default:
cout << "Invalid Option" << endl;
break;
}
}
```

```
N = pGeometry->ComputeDomain(S);
cout << "\n" << N << endl;

Particle * Fluid = new Particle[N];
pGeometry->ParticlePlacement(Fluid, S);

SPH.DensityStart(Fluid, S);

Particle::NormalizeMass (Fluid, S);
//
cout << "Calculating...." << endl;
//
SPH.IterationStart(Fluid, S, FileName, choice);
SPH.Iteration(Fluid, S, FileName, choice);

return 0;
}
```

APÊNDICE B – Forma Geral dos Núcleos e suas Derivadas

Os núcleos propostos possuem forma geral definida por

$$W(x,h) = \frac{1}{Ch^d} \begin{cases} f(q), & 0 \leq q \leq 1 \\ 0, & q \geq 1 \end{cases} \tag{B.1}$$

Onde C é a constante de normalização, d é a dimensão do domínio, h é comprimento de suavização e $q = ||x||/h$.

o caso de estudo o parâmentro da dimensão é ajustado para $d = 2$, porém os cálculos abaixo são realizados primeiro para o caso unidimensional e em seguida estendidos para o caso geral de dimensão finita qualquer.

Para o caso unidimensional o vetor posição relativa de uma partícula até outra dentro de seu suporte compacto é dado por $x = |x|\hat{x}$, onde $||x|| = |x|$. O gradiente do núcleo terá a forma

$$\begin{aligned} \nabla W &= \frac{dW}{dx} = \frac{dW}{dq}\frac{dq}{dx} \\ &= \frac{1}{Ch} f^{(1)}(q) \frac{dq}{dx}\hat{x} \\ &= \frac{1}{Ch} f^{(1)}(q) \frac{x}{h|x|} \\ &= \frac{1}{Ch^2} \frac{f^{(1)}(q)}{q} |x|\hat{x} \end{aligned} \tag{B.2}$$

Onde foi usada a relação $|x| = hq$. Sendo válida para o intervalo $0 \leq q \leq 1$, pois fora dele o valor é nulo.

Para se obter o laplaciano em uma dimensão basta derivar novamente. Observando que $\frac{|x|}{q} = h$, o que é a mesma relação já utilizada,

$$\nabla^2 W = \frac{1}{Ch^2} f^{(2)}(q). \tag{B.3}$$

Em se tratando do caso de n dimensões, o vetor posição será dado por $x = |x_1|\hat{x_1} + \cdots + |x_n|\hat{x_n}$ e a norma $||x|| = \sqrt{x_1^2 + \cdots + x_n^2}$. O gradiente é calculado coordenada

a coordenada, logo o processo é idêntico ao do caso unidimensional.

$$\nabla W = \frac{1}{Ch^{d+2}} \begin{cases} \dfrac{f^{(1)}(q)}{q}x, & 0 \leq \ q \leq 1 \\ 0, & q \geq 1 \end{cases} \tag{B.4}$$

O cálculo do laplaciano é um tanto quanto mais delicado, sendo necessário se atentar a alguns detalhes. Dessa forma, derivando a expressão anterior com respeito a i-ésima coordenada, no intervalo onde $0 \leq \ q \leq 1$, tem-se que

$$\begin{aligned} \frac{\partial^2 W}{\partial x_i^2} &= \frac{\partial}{\partial q}(\frac{\partial W}{\partial x_i})\frac{\partial q}{\partial x_i} \\ &= \frac{1}{Ch^{d+2}}[(\frac{f^{(2)}(q)}{q} - \frac{f^{(1)}(q)}{q^2})x_i.\frac{x_i}{h||x||} + \frac{f^{(1)}(q)}{q}.1] \\ &= \frac{1}{Ch^{d+2}}[f^{(2)}(q) - \frac{f^{(1)}(q)}{q}\frac{|x_i|^2}{||x||^2} + \frac{f^{(1)}(q)}{q}] \end{aligned} \tag{B.5}$$

Somando todas as n coordenadas,

$$\begin{aligned} \nabla^2 W &= \frac{1}{Ch^{d+2}}[f^{(2)}(q) - \frac{f^{(1)}(q)}{q}\frac{(|x_1|^2 + \cdots + |x_n|^2)}{||x||^2} + \frac{f^{(1)}(q)}{q}.(n)] \\ &= \frac{1}{Ch^{d+2}}[f^{(2)}(q) + (n-1)\frac{f^{(1)}(q)}{q}]. \end{aligned} \tag{B.6}$$

Então, o Laplaciano da função de suavização é dado por

$$\nabla^2 W = \frac{1}{Ch^{d+2}} \begin{cases} f^{(2)}(q) + (d-1)\dfrac{f^{(1)}(q)}{q}, & 0 \leq \ q \leq 1 \\ 0, & q \geq 1 \end{cases} \tag{B.7}$$

APÊNDICE C – Script em Shell

```
#! /bin/bash
#
###############################################
# Escreve um arquivo tipo gnu com os comandos
# para a configuração do gráfico
# Gráfico que será criado é apenas o de X x Y
###############################################
i=0
j=0
k=0
File="$i,$j$k.gnu"

while [ -a "$i.$j$k" ]
do

echo $i
echo $j
echo $k
echo $File

echo "set yrange [-0.02:1.02]" > $File

echo "set xrange [-0.02:1.02]" >> $File

echo "set xlabel 'X (m)'" >> $File

echo "set ylabel 'Y (m)'" >> $File

echo "set term jpeg" >> $File

echo "set out '$i.$j$k.jpeg'" >> $File

echo "plot '$i.$j$k' using 2:3 title 't = $i,$j$k s'
with points pt 13 " >> $File
```

```
##############################################
# Abre o gnuplot e carrega o arquivo tipo gnu
##############################################
gnuplot > load "$i,$j$k.gnu"

if (($k == 9))
then
k=0
if (($j == 9))
then
j=0
((i=$i+1))
else
((j=$j+1))
fi
else
((k=$k+1))
fi

File="$i,$j$k.gnu"

done

exit
```