Artigo Original

DOI:10.5902/2179460X24887

Ciência e Natura, Santa Maria v.39 n.3, 2017, Set - Dez, p. 569 – 594
Revista do Centro de Ciências Naturais e Exatas - UFSM
ISSN impressa: 0100-8307 ISSN on-line: 2179-460X

CIÊNCIA e NATURA

Simulação numérica de escoamentos com superfícies livres empregando o método *Weakly Compressible Smoothed Particle Hydrodynamics* (WCSPH) e o processamento em paralelo com Unidades de Processamento Gráfico e a API CUDA

Numerical simulation of free-surface flows using the method Weakly Compressible Smoothed Particle Hydrodynamics (WCSPH) and parallel processing with Graphics Processing Unit and the API CUDA

Helio Pedro Amaral Souto[1], Josecley Fialho Góes[2] e Marciana Lima Goés[1]
helio.souto@pq.cnpq.br; cleymat@gmail.com; marcianaamorimlima@gmail.com

[1]Universidade do Estado do Rio de Janeiro, RJ, Brasil
[2]Universidade Federal do Oeste do Pará, PA, Brasil

Resumo

Neste trabalho, é feita uma introdução ao método lagrangiano de partículas, livre de malhas, Smoothed Particle Hydrodynamics (SPH) voltado para a simulação numérica de escoamentos de fluidos newtonianos compressíveis e quase-incompressíveis. Neste método, o domínio do problema é discretizado empregando partículas que possuem propriedades físicas, que são calculadas através de uma interpolação, empregando-se a função de suavização, a partir das propriedades das suas partículas vizinhas. A busca pelas partículas vizinhas, dependendo do algoritmo utilizado, pode chegar a um custo computacional da ordem de N^2, onde N é a quantidade de partículas usada na simulação numérica. Portanto, buscou-se desenvolver um simulador numérico, nas versões bidimensional e tridimensional, paralelizado empregando-se a Application Programming Interface (API) Compute Unified Device Architecture (CUDA). A CUDA possibilita o processamento em paralelo empregando os núcleos das Graphics Processing Units (GPUs) das placas de vídeo. Para aumentar a eficiência computacional, foi minimizado o acesso à memória global e a transferência de dados entre a Central Processing Unit (CPU) e a GPU, e as variáveis do tipo float4 foram usadas para alocar os vetores, garantindo-se o tamanho e as exigências de alinhamento da CUDA C. Além disso, os dados foram referenciados para a memória textura, sempre que possível. Os resultados numéricos foram validados considerando-se a resolução do problema do escoamento, bidimensional e tridimensional, de um fluido newtoniano, em um vertedouro com degraus de uma barragem.

Palavras-chave: *CUDA, Dinâmica dos Fluidos Computacional, Métodos de partículas, Smoothed Particle Hydrodynamics.*

Abstract

In this work we introduce the Lagrangian meshfree Smoothed Particle Hydrodynamics (SPH) method oriented to the numerical simulation of compressible and weakly-compressible flows of Newtonian fluids. In this method, the problem domain is discretize with particles which carry physical properties that are calculated via interpolation, using the kernel function, from the properties of its neighboring particles. Depending on the algorithm used, the search for neighboring particles can reach a computational cost of the order of N^2, where N is the number of particles used in the numerical simulation. Therefore, we sought to develop a two- and three-dimensional numerical simulator paralelized employing the Compute Unified Device Architecture (CUDA) API. The CUDA enables parallel processing using multi-core processors of Graphics Processing Units (GPUs). To increase the computational efficiency global memory access was minimized as well CPU to GPU data transfer, and float4 type variables were used to allocate vectors such that CUDA C size and alignement requirements were assured. In addition, the data were referenced to the texture memory whenever possible. The numerical results were validated considering the resolution of two- and three-dimensional flow problems (for a Newtonian fluid) in a stepped spillway of a dam.

Keywords: *Computational fluid dynamics, CUDA, Particle Methods, Smoothed Particle Hydrodynamics.*

Recebido: 25/11/2016 Aceito: 27/06/2017

1 Introducão

O estudo do escoamento de fluidos newtonianos, do transporte de massa e de energia possui uma grande importância hoje em dia devido à sua aplicação em diferentes áreas do conhecimento científico e tecnológico. Na análise de fenômenos físicos ligados à engenharia, pode-se citar os problemas envolvendo o escoamento de fluidos no interior de tubulações, ao redor de obstáculos, na atmosfera, nos rios, nos mares, em meios porosos, em escoamentos contendo interfaces e superfícies livres etc.

O escoamento de fluidos newtonianos é governado por um conjunto de equações diferenciais parciais acopladas, que são as equações da continuidade (conservação da massa), de Navier-Stokes (conservação do *momentum*) e da energia (conservação da energia). Com exceção de alguns casos simples, não é possível obter-se uma solução analítica dessas equações e, portanto, recorre-se ao desenvolvimento de simuladores numéricos robustos, computacionalmente eficientes, que sejam capazes de fornecer resultados acurados quando da resolução dessas equações.

Tradicionalmente, na Dinâmica dos Fluidos Computacional (DFC), os problemas são resolvidos numericamente mediante o emprego de métodos numéricos convencionais que empregam malhas computacionais, tais como os métodos das diferenças finitas, dos volumes finitos e dos elementos finitos, dentre outros. Uma característica desses métodos é a de que eles necessitam de uma malha computacional euleriana (para os métodos das diferenças finitas e dos volumes finitos) ou lagrangiana (método dos elementos finitos) a fim de que as equações de balanço possam ser discretizadas e a solução numérica seja obtida nos nós destas malhas, que representam o domínio físico particionado. Entretanto, quando se quer simular alguns tipos de escoamentos particulares, que envolvam grandes deformações, fronteiras deformáveis ou superfícies livres, esses métodos podem apresentar algumas dificuldades práticas quando das suas aplicações (Liu, 2003).

Por outro lado, a modelagem numérica do escoamento de superfícies livres ou interfaces ainda representa um problema desafiador. Como a posição da interface que separa os fluidos, ou da superfície livre, é a priori desconhecida, algoritmos específicos devem ser utilizados na determinação da sua posição. Várias estratégias descritas na literatura podem ser empregadas nas simulações numéricas envolvendo o escoamento de múltiplos fluidos. Neste estudo, está-se particularmente interessado nos métodos que possam identificar a região espacial ocupada por cada um destes fluidos e nos escoamentos envolvendo superfícies livres. Dentre as abordagens eulerianas que podem ser empregadas na resolução destes problemas, cita-se os métodos do tipo *volume of fluid, level set e pseudo-concentration*, dentre outros (Malidi et al., 2005). Entretanto, esses métodos podem levar a resultados imprecisos caso a estratégia numérica não seja adequadamente escolhida. Tal fato se deve às incertezas associadas ao uso de uma região de transição a fim de se localizar as interfaces livres, o que torna impossível a imposição explícita de condições de contorno nesta interface. Além disto, esses métodos também podem introduzir oscilações espúrias e difusão numérica nos seus resultados.

Portanto, uma alternativa seria o emprego de um método lagrangiano do tipo *meshfree*, melhor adaptado à resolução de problemas apresentando interfaces e superfícies livres. Estes métodos têm sido desenvolvidos para a simulação computacional tanto de problemas da mecânica dos sólidos quanto de problemas da dinâmica dos fluidos. Como principal atrativo, eles apresentam uma capacidade específica para tratarem de problemas envolvendo grandes deformações, superfícies livres, a captura de interfaces móveis, geometrias complexas, descontinuidades, singularidades etc. Os métodos de partículas livres de malhas, *Meshfree particle methods* (MPM), resolvem o problema físico mediante o uso de um conjunto de partículas, as quais representam um objeto físico ou uma parcela do domínio. Nos problemas de DFC, as variáveis como a massa específica, a energia, a posição, a velocidade, a pressão etc., são calculadas e atribuídas à cada partícula (Crespo, 2008).

No que diz respeito aos métodos livres de malhas, o método lagrangiano conhecido como *Smoothed Particle Hydrodynamics* (SPH) vem sendo largamente aplicado a problemas físicos de diversas áreas, principalmente por ter se mostrado promissor na resolução de problemas apresentando grandes deformações, explosões, materiais com interfaces móveis, fronteiras deformáveis e superfícies livres (Antuono et al., 2010, 2012; Federico et al., 2012; Marrone et al., 2013). No método SPH, a discretização do domínio do problema dá-se através do emprego de partículas e as equações são discretizadas a partir da *Representação Integral* das funções e da *Aproximação por Partículas*, que representa a forma discreta da primeira representação (Liu, 2003). Conforme já dito, o domínio do problema é discretizado através do uso de partículas, sem a necessidade da existência de uma malha computacional. As partículas carregam individualmente as informações sobre a massa, a posição, a velocidade etc. das partículas do fluido, mas também podem possuir massa específica, pressão e temperatura, dentre outras propriedade físicas, dependendo do problema considerado. As propriedades físicas no método SPH são obtidas através de uma média ponderada, pela função de suavização, das propriedades das partículas vizinhas à partícula da qual deseja-se calcular os valores físicos (Góes, 2011).

Devido às suas características, o método SPH pode ser facilmente expandido da versão unidimensional para a versão tridimensional e pode ser adaptado para o processamento em paralelo devido às suas características intrínsecas. Além disso, como as condições de contorno e o fluido são representados por um conjunto discreto de N partículas e a quantidade de partículas influencia na acurácia dos resultados (Liu, 2003), torna-se necessário o uso de muitas partículas (milhares ou até milhões) em uma simulação, exigindo-se assim um alto custo computacional, o que justifica, por exemplo, o uso do processamento paralelo com o recurso das GPUs (*Graphics Processing Units*). Elas são projetadas principalmente para a aceleração dos processos que permitem a visualização gráfica, e a paralelização é viabilizada através da API CUDA. A CUDA (*Compute Unified Device Architecture*) é um modelo de plataforma de computação e programação paralela desenvolvida pela NVIDIA *Corporation* que permite um

aumento significativo da eficiência computacional, aproveitando o poder de processamento intrínseco das GPUs (NVIDIA, 2016b).

Este trabalho tem como objetivo principal o desenvolvimento de um simulador numérico, com versões bi e tridimensional, baseado no método de partículas livre de malhas SPH (Lucy, 1977; Gingold e Monaghan, 1982), para a resolução das equações de balanço que governam o escoamento isotérmico de fluidos newtonianos, envolvendo superfícies livres, com o uso da API CUDA C. A formulação do método SPH é utilizada na discretização espacial das equações de balanço (de massa e de *momentum*), resultando num sistema de equações diferenciais ordinárias com relação ao tempo. Este sistema de equações será resolvido mediante a sua integração no tempo empregando-se um método explícito do tipo previsor-corretor (Góes, 2011).

Como exemplo de aplicação do simulador desenvolvido, considerou-se exclusivamente o problema do escoamento isotérmico, bi e tridimensional, de um fluido newtoniano em um vertedouro com degraus de uma barragem fluvial.

2 O método SPH

O *Smoothed Particle Hydrodynamics* é um método que foi desenvolvido, inicialmente, para resolver problemas astrofísicos (Lucy, 1977; Gingold e Monaghan, 1982). No entanto, como o movimento das partículas também é governado por equações diferenciais parciais, o seu emprego foi estendido ao caso dos escoamentos que podem ser modelados pelas equações da Mecânica do Contínuo. Atualmente, esse método vem sendo usado cada vez mais em computação gráfica para simular-se fenômenos do mundo real (explosões, inundações, choques, tsunamis etc.) e, principalmente, em jogos, onde a acurácia não é tão importante como a interatividade e a sensação de realidade.

2.1 Representação Integral

O método SPH baseia-se na representação integral de uma função, ou seja, uma dada função $F(\mathbf{x})$ é descrita mediante o uso de uma função suave $W(|\mathbf{x}-\mathbf{x}'|,h)$, denominada de função de suavização (ou núcleo) (Liu, 2003):

$$F(\mathbf{x}) \cong \int_{\Omega} F(\mathbf{x}')\, W(|\mathbf{x}-\mathbf{x}'|,h)\, d\mathbf{x}' \tag{1}$$

onde $\mathbf{x}$ é o vetor posição e a variável h é conhecida como sendo o *comprimento de suavização* da função de suavização. Essa variável determina a área ou o volume do domínio de suporte compacto Ω ou domínio de influência da função W. A Fig. 1 mostra a representação volumétrica do domínio de suporte compacto da função de suavização W, com relação à partícula i (no centro da esfera), e as partículas que encontram-se dentro do seu domínio de influência (no interior da esfera) e fora dele (no exterior da esfera).

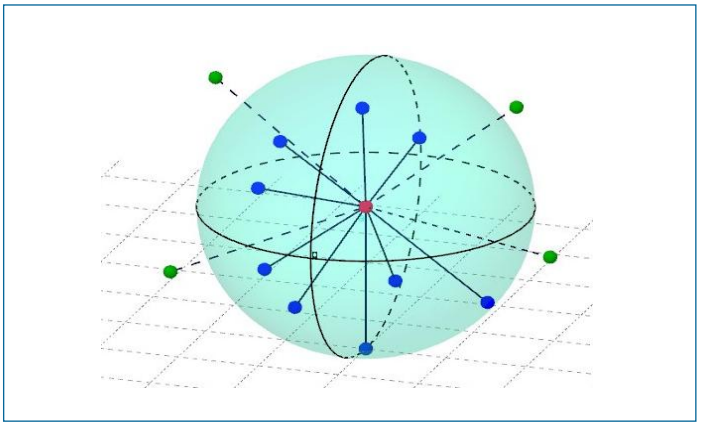


Figura 1: Domínio de suporte compacto tridimensional.

Nos métodos baseados em malhas as derivadas são aproximadas, por exemplo, por diferenças finitas, empregando-se uma expansão em série de Taylor e, dependendo da ordem de aproximação, faz-se um truncamento dos termos da série e determina-se a quantidade de pontos nodais que serão empregados na aproximação numérica da derivada da função. Por outro lado, no método SPH o gradiente de uma função é calculado a partir de uma integração por partes da integral da Representação Integral da função gradiente

$$\nabla F(\mathbf{x}) \cong \int_{\Omega} \nabla F(\mathbf{x}')\, W(|\mathbf{x}-\mathbf{x}'|,h)\, d\mathbf{x}'$$

e levando-se em consideração que a função de suavização possui um suporte compacto (Góes, 2011). Assim sendo, elimina-se a necessidade de conhecer-se o gradiente da função, uma vez que basta determinar-se (analiticamente) o gradiente da função de suavização

$$\nabla F(\mathbf{x}) \cong -\int_{\Omega} F(\mathbf{x}')\nabla W(|\mathbf{x}-\mathbf{x}'|,h)\, d\mathbf{x}' \tag{2}$$

O mesmo procedimento pode ser aplicado ao operador divergente e a outros operadores (Liu, 2003; Crespo, 2008).

2.2 Aproximação por Partículas

A representação dada pela Eq. (1) é substituída por uma aproximação discreta da função $F(\mathbf{x})$, avaliada na posição ocupada por uma dada partícula i, e é denominada de Aproximação por Partículas (Liu, 2003),

$$F(\mathbf{x}_i) = \sum_{j=1}^{N} \frac{m_j}{\rho_j} F(\mathbf{x}_j) W_{ij}$$

onde a função $F(\mathbf{x})$, na posição $\mathbf{x}_i$, é aproximada através da soma ponderada (pela função de suavização W) dos seus valores determinados nas posições correspondentes às N partículas vizinhas, que se encontram dentro do domínio de influência da partícula i. A massa e a massa específica são representadas por m_j e ρ_j, respectivamente, e $W_{ij} = W(|\mathbf{x}_i - \mathbf{x}_j|, h)$ representa a função de suavização.

De modo análogo, pode-se obter a versão discretizada da Eq. (2)

$$\nabla F(\mathbf{x}_i) = \sum_{j=1}^{N} \frac{m_j}{\rho_j} F(\mathbf{x}_j) \nabla_i W_{ij}$$

onde

$$\nabla_i W_{ij} = \frac{\mathbf{x}_i - \mathbf{x}_j}{|\mathbf{x}_i - \mathbf{x}_j|} \frac{\partial W_{ij}}{\partial |\mathbf{x}_i - \mathbf{x}_j|} = \frac{\mathbf{x}_{ij}}{r_{ij}} \frac{\partial W_{ij}}{\partial r_{ij}}$$

A função de suavização W desempenha um papel importantíssimo no método SPH e a sua escolha deve ser feita de modo a que ela satisfaça a algumas condições específicas (Liu, 2003; Góes, 2011). As aproximações por partículas empregadas no método SPH, para serem acuradas, dependem da existência de um número suficiente de partículas dentro do domínio de suporte compacto, κh. O comprimento de suavização h influencia diretamente na eficiência computacional e na acurácia dos resultados numéricos. Caso ele seja muito pequeno, não existirá um número suficiente de partículas na vizinhança de uma determinada partícula i, de maneira que pode-se obter resultados pouco acurados. Por outro lado, um comprimento de suavização muito grande pode resultar na perda de informações locais, o que também pode levar a uma inacurácia dos resultados numéricos (Liu, 2003; Góes, 2011).

Qualquer função apresentando as propriedades requeridas (Liu, 2003) pode ser uma função de suavização do método SPH. Várias funções de suavização foram propostas e pesquisadas na literatura. Maiores detalhes sobre como construir uma função de suavização, bem como outros exemplos de funções de suavização utilizadas no método SPH, podem ser encontrados nas referências Liu (2002, 2003) e Li (2004). No que se segue, define-se a distância adimensional entre as partículas por $q = \frac{r}{h}$, onde r é a distância entre as partículas i e j. Neste trabalho são utilizadas as seguintes funções de suavização, *Spline* Cúbica:

$$W(r,h) = \alpha_d \begin{cases} \frac{2}{3} - q^2 + \frac{1}{2}q^3 & 0 \leq q < 1 \\ \frac{1}{6}(2-q)^3 & 1 \leq q < 2 \\ 0 & q \geq 2 \end{cases}$$

onde α_d é $\frac{1}{h}$ para um problema unidimensional (1D), $\frac{15}{7\pi h^2}$ para o bidimensional (2D) e $\frac{3}{2\pi h^3}$ para o tridimensional (3D), e *Spline* Quíntica:

$$W(r,h) = \alpha_d \left(1 - \frac{q}{2}\right)^4 (2q+1) \qquad 0 \leq q < 2$$

onde α_d é $\frac{3}{4h}$ em 1D, $\frac{7}{4\pi h^2}$ em 2D e $\frac{21}{16\pi h^3}$ em 3D.

3 Equações governantes e aspectos numéricos

Nesta seção, são apresentadas as equações que governam o escoamento de um fluido newtoniano. Estas equações são oriundas dos balanços de massa, da quantidade de movimento e de energia, e expressam a conservação da massa, do *momentum* e da energia. Devido à descrição material adotada para o método SPH, as equações de balanço são fornecidas na sua forma lagrangiana.

3.1 Equações governantes

Como considera-se somente o escoamento isotérmico de um fluido newtoniano, as equações governantes são:

1. Equação da Continuidade;
2. Equação de Navier-Stokes.

Utilizando-se os índices gregos α e β para representar as coordenadas direcionais, essas equações podem ser escritas em notação indicial como:

1. A equação do balanço de massa:

$$\frac{D\rho}{Dt} = -\rho \frac{\partial v^\beta}{\partial x^\beta} \tag{3}$$

2. A equação de Navier-Stokes (sem as forças de corpo):

$$\frac{Dv^\alpha}{Dt} = \frac{1}{\rho}\frac{\partial \sigma^{\alpha\beta}}{\partial x^\beta} \tag{4}$$

Na Eq. (4) $\sigma^{\alpha\beta}$ representa as componentes do tensor de tensão

$$\sigma^{\alpha\beta} = -p\delta^{\alpha\beta} + \tau^{\alpha\beta}$$

onde p é a pressão termodinâmica, δ é o delta de Kronecker e $\boldsymbol{\tau}$ é o tensor de tensão viscosa.

Para fluidos newtonianos a tensão de cisalhamento viscosa deve ser proporcional à taxa de deformação, representada pelo tensor $\boldsymbol{\varepsilon}$,

$$\tau^{\alpha\beta} = \mu\varepsilon^{\alpha\beta}$$

onde μ é a viscosidade dinâmica do fluido e

$$\varepsilon^{\alpha\beta} = \frac{\partial v^\beta}{\partial x^\alpha} + \frac{\partial v^\alpha}{\partial x^\beta} - \frac{2}{3}(\nabla \cdot \mathbf{v})\delta^{\alpha\beta}$$

3.2 Equações governantes discretizadas

Na formulação SPH, as equações que governam o escoamento de um fluido newtoniano, Eqs. (3)-(4), são discretizadas empregando-se a Aproximação por Partículas para os operadores diferenciais (Góes, 2011):

1. A equação do balanço de massa:

$$\frac{D\rho_i}{Dt} = \sum_{j=1}^{N} m_j v_{ij}^\beta \frac{\partial W_{ij}}{\partial x_i^\beta}$$

2. A equação de Navier-Stokes:

$$\frac{Dv_i^\alpha}{Dt} = \sum_{j=1}^{N} m_j \left(\frac{\sigma_i^{\alpha\beta}}{\rho_i^2} + \frac{\sigma_j^{\alpha\beta}}{\rho_j^2}\right)\frac{\partial W_{ij}}{\partial x_i^\beta}$$

onde $v_{ij}^\beta = v_i^\beta - v_j^\beta$. Para maiores detalhes sobre a discretização dessas equações, recomenda-se a leitura dos trabalhos de Liu (2003) e Góes (2011).

3.3 Aspectos numéricos

Alguns aspectos fundamentais ligados à implementação numérica do método WCSPH são tratados na sequência. Devido à abordagem desse método o passo de tempo é limitado pela velocidade do som (conforme será visto adiante), mas ele pode ser paralelizado sem maiores dificuldades. Além disso, o modelo WCSPH não exige um tratamento explícito para detectar-se as superfícies livres, desde que a condição de contorno ao longo da superfície livre seja satisfeita implicitamente (Antuono et al., 2012).

As simulações com o método WCSPH são geralmente realistas. No entanto, o campo de pressão associado às partículas pode apresentar variações (não físicas) nos seus valores quando da resolução de problemas de escoamento de fluidos incompressíveis. Estas oscilações espúrias no campo de pressão e na massa específica podem resultar na aglomeração ou espalhamento das partículas durante a simulação numérica. Nos últimos anos, diferentes técnicas têm sido propostas a fim de evitar-se o seu surgimento. As principais correções aqui implementadas são descritas a seguir.

3.3.1 Dissipação viscosa

No tratamento do termo viscoso da equação de Navier-Stokes ele é decomposto em duas contribuições: uma devida ao modelo laminar (Lo e Shao, 2002) e outra ao modelo turbulento *Sub-Particle Scale* (SPS) (Gotoh et al., 2004; Dalrymple e Rogers, 2006; Ricardo et al., 2016). Usando-se uma formulação simétrica na discretização do termo de tensão (Lo e Shao, 2002) obtém-se:

$$\frac{D\mathbf{v}_i}{Dt} = -\sum_{j=1}^{N} m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right) \nabla_i W_{ij} + \sum_{j=1}^{N} m_j \left[\frac{4\ \nu_l\ \mathbf{x}_{ij} \cdot \nabla_i W_{ij}}{(\rho_i + \rho_j)(\mathbf{x}_{ij}^2 + \eta^2)}\right] \mathbf{v}_{ij} + \sum_{j=1}^{N} m_j \left(\frac{\boldsymbol{\tau}_i}{\rho_i^2} + \frac{\boldsymbol{\tau}_j}{\rho_j^2}\right) \cdot \nabla_i W_{ij},$$

onde ν_l é a viscosidade laminar e $\boldsymbol{\tau}_k$ representa o tensor de tensão sub-partículas (que contém a contribuição turbulenta) dado por

$$\tau_i^{\alpha\beta} = 2\rho_i \nu_t \left(S^{\alpha\beta} - \frac{1}{3} S^{\xi\xi} \delta^{\alpha\beta}\right) - \frac{2}{3} \rho_i C_I |r_{ij}|^2 |\mathbf{S}|^2 \delta^{\alpha\beta} \tag{5}$$

onde a parte simétrica do tensor de tensão, $\mathbf{S}$, é dada por

$$S^{\alpha\beta} = \frac{1}{2}\left(\frac{\partial v^\alpha}{\partial x^\beta} + \frac{\partial v^\beta}{\partial x^\alpha}\right)$$

sendo $\nu_t = (C_s |r_{ij}|)^2 |\mathbf{S}|$ a viscosidade turbulenta, C_s é a constante de Smagorinsky (igual a 0,12), $C_I = 0{,}0066$, $|r_{ij}|$ a distância inicial entre as partículas e a taxa de tensão local é dada por $|\mathbf{S}| = \sqrt{2 S^{\alpha\beta} S^{\alpha\beta}}$.

3.3.2 Compressibilidade artificial

No método WCSPH, o escoamento do fluido é tratado como sendo fracamente compressível, permitindo a utilização de uma equação de estado, do tipo quase-incompressível, na modelagem de escoamentos incompressíveis para determinar-se a pressão do fluido. Monaghan (1994) empregou a equação conhecida como equação de estado de Tait para a modelagem de escoamentos com superfícies livres

$$p = B\left[\left(\frac{\rho}{\rho_0}\right)^\gamma - 1\right] \tag{6}$$

onde γ é uma constante, ρ_0 é a massa específica de referência, $B = \frac{c_0^2 \rho_0}{\gamma}$, que estabelece a variação máxima da massa específica e $c_0 = c(\rho_0)$ é a velocidade do som de referência.

De acordo com Solenthaler e Pajarola (2009), o uso da equação de estado de Tait, Eq. (6), pode levar ao aparecimento de oscilações numéricas no campo de pressões, o que introduz uma instabilidade no método. Além disso, a necessidade da escolha de uma alta velocidade do som, para efeito de resolução numérica, pode levar a severas restrições no passo de tempo em razão da condição de Courant-Friedrichs-Lewy (CFL). Para resolver esse problema foram propostas as correções δ_{SPH} e APD.

3.3.3 Correção δ_{SPH}

A correção δ_{SPH} corresponde à introdução de um termo de difusão artificial para a equação do balanço de massa, adicionado à equação da continuidade, a fim de remover as oscilações espúrias de alta frequência no campo de pressão (Marrone et al., 2011). Esse termo difusivo foi proposto por Molteni e Colagrossi (2009) e a equação da continuidade é reescrita como

$$\frac{D\rho_i}{Dt} = \sum_{j=1}^{N} m_j v_{ij} \frac{\partial W_{ij}}{\partial x_i^\beta} + 2\delta_{SPH} h c_0 \sum_{j=1}^{N} m_j \left(\frac{\rho_i}{\rho_j} - 1\right) \frac{\mathbf{x}_{ij}}{|\mathbf{x}_{ij}|^2} \cdot \nabla_i W_{ij}$$

onde δ_{SPH} é um parâmetro a ser determinado, geralmente o valor recomendado é $0{,}1$ para a maioria das aplicações.

3.3.4 Correção APD

A correção *Artificial Particle Displacement* (APD) foi proposta para prevenir a aglomeração de partículas. Segundo Ozbulut et al. (2014), se as partículas do domínio físico do problema adquirirem uma distribuição não-uniforme e aglomerada, a estabilidade e a fiabilidade dos resultados numéricos podem não ser mantidas e, em casos extremos, o algoritmo numérico pode falhar. Para corrigir estas anomalias na trajetória das partículas, adiciona-se ao vetor posição das partículas um deslocamento artificial definido por (Shadloo et al., 2012):

$$\delta\mathbf{x}_i = \beta \sum_{j=1}^{N} \frac{\mathbf{x}_{ij}}{|\mathbf{x}_{ij}|^3}\ x_0^2\ v_{max}\ \Delta t$$

onde $\delta\mathbf{x}_i$ é o vetor do deslocamento artificial de partículas, β é um parâmetro que depende do problema, $x_0 = \sum_{j=1}^{N} \frac{|\mathbf{x}_{ij}|}{N}$ é a média das distâncias das partículas vizinhas e v_{max} é a magnitude da velocidade máxima no domínio do problema e para o passo de tempo considerado. N é o número de partículas vizinhas dentro do domínio do suporte compacto da partícula i.

3.3.5 Busca pelas partículas vizinhas

No método SPH, somente as partículas dentro do domínio de suporte compacto da função de suavização, para uma dada partícula i, contribuem para o cálculo das propriedades físicas que cada partícula carrega, ou seja, somente as partículas, na vizinhança da partícula i, que se encontram na sua zona de influência devem ser consideradas quando do uso da Aproximação por Partículas. A este grupo de partículas denomina-se de *partículas vizinhas próximas*.

Os procedimentos de determinação das partículas vizinhas próximas, mais utilizados nos métodos SPH, são a *Busca Direta*, a *Busca em Lista* e a *Busca em Árvore* (Liu, 2003). Na Busca Direta, todos os pares de partículas são testados e as partículas vizinhas atualizadas a cada passo de tempo, sendo necessárias N^2 operações para testar todos os pares de partículas, onde N é a quantidade de partículas. Por outro lado, na Busca em Lista o domínio computacional é dividido em células quadradas de tamanho kh, onde k é um fator que depende da função de suavização (por exemplo, $k = 2$ para a função *Spline* Cúbica). Então, para uma partícula localizada em uma determinada célula, são consideradas apenas as iterações com as partículas das células vizinhas, reduzindo-se de N^2 para $N \log N$ o número de operações de busca pelas partículas vizinhas a cada passo de tempo. Foram implementadas, neste trabalho, as buscas Direta e em Lista.

3.3.6 Condições de contorno

Quando uma partícula de fluido se aproxima da fronteira física, apenas as partículas que encontram-se no domínio de suporte da função de suavização interagem e contribuem para o cálculo das propriedades física. Entretanto, próximo das fronteiras o número de partículas vizinhas cai significativamente. Como consequência, obtém-se soluções que não são corretas nas proximidades das paredes sólidas das fronteiras onde embora a velocidade das partículas seja nula, o mesmo não acontece com as outras variáveis tais como a massa específica e a pressão. Para minimizar esse problema, diferentes soluções consideram a introdução de partículas virtuais nas fronteiras do problema. Neste trabalho, foram implementados dois tipos de tratamento das fronteira (força repulsiva e partículas dinâmicas) e uma condição de periodicidade:

1. Força repulsiva: foram propostas inicialmente por Monaghan (1994) com a finalidade de produzir uma força repulsiva sobre as partículas que se encontram próximas da fronteira, evitando que estas partículas atravessem os limites do domínio físico do problema. Quando uma partícula de fluido se aproxima de uma partícula, que se encontra sobre a fronteira, de uma distância r_0, uma força é aplicada par-a-par ao longo da linha de centro ligando estas duas partículas (Liu, 2003):

$$Fr_{ij} = \begin{cases} D\left[\left(\frac{r_0}{r_{ij}}\right)^{n_1} - \left(\frac{r_0}{r_{ij}}\right)^{n_2}\right]\frac{\mathbf{r}_{ij}}{r_{ij}^2} & \left(\frac{r_0}{r_{ij}}\right) \leq 0 \\ 0 & \left(\frac{r_0}{r_{ij}}\right) > 1 \end{cases}$$

   onde n_1 e n_2 são parâmetros que devem ser fornecidos, geralmente 12 e 4 respectivamente. D é um parâmetro que depende do problema considerado e deve ser escolhido de modo que o seu valor tenha a mesma ordem de grandeza do quadrado da maior velocidade das partículas. A variável r_0 representa a distância máxima para que a força de repulsão seja aplicada.

2. Partículas dinâmicas: foram introduzidas por Dalrymple e Knio (2001). As partículas são dispostas de modo a formar duas colunas de partículas nas fronteiras. As mesmas propriedades calculadas para as partículas de fluido também são determinadas para as partículas dinâmicas, mas as suas posições permanecem sempre inalteradas.

3. Condição de periodicidade: podem ser implementadas nas fronteiras abertas. As partículas que encontram-se a uma distância menor que o comprimento de suavização, das fronteiras abertas do domínio do problema, interagem com as partículas que encontram-se nas suas posições correspondentes (após ou antes da fronteira) em função da periodicidade, evitando-se assim o truncamento do comprimento de suavização. Ao atingirem esses limites, deixando o domínio físico, elas são reposicionadas na fronteira de entrada correspondente (Crespo, 2008).

3.3.7 Correção XSPH

Monaghan (1989) utilizou a técnica *eXtend Smoothed Particle Hydrodynamics* (XSPH) na simulação de escoamentos de fluidos envolvendo superfícies livres e com altas velocidades, de modo a mantê-las ordenadas e também evitar uma possível interpenetração das partículas. A técnica consiste em calcular uma média, ponderada pela função de suavização, das velocidades das partículas vizinhas e corrigir a velocidade de cada partícula através desse valor médio, multiplicado por um parâmetro constante $\epsilon \in [0,1]$,

$$\mathbf{v}_i^{XSPH} = \mathbf{v}_i + \epsilon \sum_{j=1}^{N} \frac{2m_j}{\rho_i + \rho_j}(\mathbf{v}_i - \mathbf{v}_j)W_{ij} \ ,$$

onde $\mathbf{v}_i^{XSPH}$ é a velocidade corrigida e $\mathbf{v}_i$ a velocidade calculada via integração numérica. Assim, o deslocamento das partículas é calculado usando essa velocidade corrigida.

3.3.8 Correção VXSPH

Uma outra possibilidade de correção é fornecida pela técnica VXSPH. Segundo Ozbulut et al. (2014), o algoritmo VXSPH introduz uma tensão superficial artificial nas partículas que compõem a superfície livre, impedindo a fuga individual de partículas e mantendo-as ligadas à superfície livre. Essa correção é aplicada tanto nos cálculos do deslocamento das partículas quanto no movimento das partículas.

3.3.9 Integração no tempo

A forma final discretizada das equações da continuidade e do *momentum*, juntamente com a equação para a determinação da posição das partículas, formam um conjunto de equações diferenciais ordinárias com relação ao tempo

$$\frac{D\rho_i}{Dt} = \sum_{j=1}^{N} m_j v_{ij} \frac{\partial W_{ij}}{\partial x_i^{\beta}} + 2\delta_{SPH} h c_o \sum_{j=1}^{N} m_j \left(\frac{\rho_i}{\rho_j} - 1\right) \frac{\mathbf{x}_{ij}}{|\mathbf{x}_{ij}|^2} \cdot \nabla_i W_{ij}, \tag{7}$$

$$\frac{D\mathbf{v}_i}{Dt} = -\sum_{j=1}^{N} m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right) \nabla_i W_{ij} + \sum_{j=1}^{N} m_j \left[\frac{4\,\nu_l\,\mathbf{x}_{ij} \cdot \nabla_i W_{ij}}{(\rho_i + \rho_j)(|\mathbf{x}_{ij}|^2 + \eta^2)}\right] \mathbf{v}_{ij} + \sum_{j=1}^{N} m_j \left(\frac{\boldsymbol{\tau}_i}{\rho_i^2} + \frac{\boldsymbol{\tau}_j}{\rho_j^2}\right) \cdot \nabla_i W_{ij} + \mathbf{g}, \tag{8}$$

$$\frac{D\mathbf{x}_i}{Dt} = \mathbf{v}_i^{XSPH}, \tag{9}$$

onde o tensor tensão $\boldsymbol{\tau}$, para o modelo turbulento, é dado pela Eq. (5).

Para resolver as Eqs. (7)–(9) foi empregado o método de integração numérica *Leapfrog* (LF) (Góes, 2011). As principais características desse método são:

1. Método explícito;
2. Fácil de implementar;
3. Necessita de uma pequena quantidade de memória RAM para alocação das variáveis computacionais;
4. Tem ordem de convergência quadrática;
5. Condicionalmente estável, onde a sua estabilidade é garantida caso a condição de CFL seja verificada.

No método *Leapfrog* (LF) a massa específica, Eq. (7), e a velocidade, Eq. (8), são atualizadas em tempos intermediários àqueles em que são atualizadas as posições das partículas, Eq. (9), exceto no primeiro passo tempo, onde são atualizadas, primeiramente, a meio passo de tempo. Portanto, numa primeira etapa, os valores conhecidos da massa específica e velocidade, no instante t_0, são utilizados na determinação dos seus respectivos valores em $t_0 + \Delta t/2$, enquanto que as posições das partículas são avaliadas para um instante de tempo igual a $t_0 + \Delta t$ (Góes, 2011).

A estabilidade do método é garantida com o passo de tempo sendo determinado por (Lee et al., 2008)

$$\Delta t = min\left(0{,}25\frac{h}{c_0 + |\mathbf{v}_{max}|}; 0{,}125\frac{h^2}{\nu}; 0{,}25\sqrt{\frac{h}{|\mathbf{f}_i|_{max}}}\right)$$

onde c_0 é a velocidade do som, $\mathbf{v}_{max}$ é a velocidade máxima para o passo de tempo considerado, ν é a viscosidade cinemática dada por $\frac{\mu}{\rho}$ e $|\mathbf{f}_i|_{max}$ é a magnitude da força por unidade de massa (aceleração) máxima no tempo atual.

4 O código SPH paralelizado

A simulação de problemas que envolvem o escoamento de fluidos em tempo real, usando apenas versões seriais do método SPH, é uma tarefa que apresenta um custo computacional elevado. No entanto, com o surgimento dos aceleradores gráficos, simulações realísticas de fluidos, em tempo real, tornou-se uma realidade. As GPUs representam um dos recursos comumente utilizados, atualmente, na aceleração dos processos que permitem a visualização gráfica e o processamento em paralelo.

4.1 Computação paralela com a API CUDA

Em novembro de 2006, o processamento em GPUs ganhou grande destaque com o lançamento da API para computação paralela de alto desempenho, desenvolvida e mantida pela NVIDIA *Corporation*, denominada *Compute Unified Device Architecture* (CUDA). Com a CUDA é possível utilizar-se o poder de processamento dos muitos núcleos, que estão presentes nas GPUs das placas de vídeo, para o processamento em paralelo. O seu uso é relativamente fácil de se aprender, para programadores familiarizados com a linguagem de programação C/C++. A CUDA está em constante atualização, acompanhando a evolução das GPUs. Hoje ela está na versão 7.5 e capacidade de computação 5.3.

Além das vantagens mencionadas, baseada na mais popular plataforma de programação, Eclipse, a NVIDIA disponibilizou um editor de código fonte unificado para CPU e GPU, o *NsightTM Eclipse Edition*, que foi projetado para auxiliar os programadores em todas as fases de desenvolvimento das aplicações CUDA, tanto para a compilação quanto para a depuração (NVIDIA, 2016b).

Um código computacional usando a API CUDA permite combinar a programação sequencial com a programação paralela de uma forma heterogênea, onde o *host*, conhecido como CPU, e o *device*, ou GPU, são encarados como unidades de processamento distintas e com memórias diferentes, de modo que um conjunto de marcadores e diretivas ou prefixos, que são extensões mínimas ao ambiente de programação C, permitem que o compilador reconheça se o código em questão deve ser executado na GPU ou na CPU. Ao ser identificada a parte do código que será executada no *device*, as instruções contidas na arquitetura CUDA atuam sobre o dispositivo, sendo feita a cópia dos dados para a memória do mesmo. Em seguida, as partes do programa são executadas em paralelo e os dados são retornados para o *host*.

Na versão paralelizada, as partes do código que deverão ser executadas no *device* são reescritas, pelo programador, na forma de *kernels*. Um *kernel* é responsável pela execução, no *device*, de um conjunto de *threads* simultaneamente que, para o seu gerenciamento, são agrupados em grades de blocos de *threads*. Uma *thread* é um fluxo único de controle sequencial dentro de um programa. Em um bloco de *threads* existe uma comunicação por meio de sincronização do tipo barreira, que faz com que todas as *threads* dentro de um bloco aguardem até serem autorizadas a prosseguir, e o acesso compartilhado a um espaço de memória exclusivo de cada bloco de *threads*. Por outro lado, em uma grade, os blocos são executados independentemente.

As *threads* podem acessar informações disponíveis em vários espaços de memória durante a sua execução, como ilustrado na Fig. 2. Cada *thread* tem uma memória privada local. Cada bloco de *threads* possui uma memória compartilhada acessível a todas as *threads* do bloco e com tempo de vida igual ao do bloco. Todas as *threads* têm acesso à memória global. Vale ressaltar que o custo do acesso à memória compartilhada é muito menor do que o custo do acesso à memória global. Assim, uma estratégia para diminuir a latência da memória do dispositivo seria fazer a cópia de porções de dados reutilizáveis para as memórias de acesso rápido. No entanto, a utilização de tais memórias deve ser cuidadosa, uma vez que o seu mau uso pode prejudicar o desempenho da simulação.

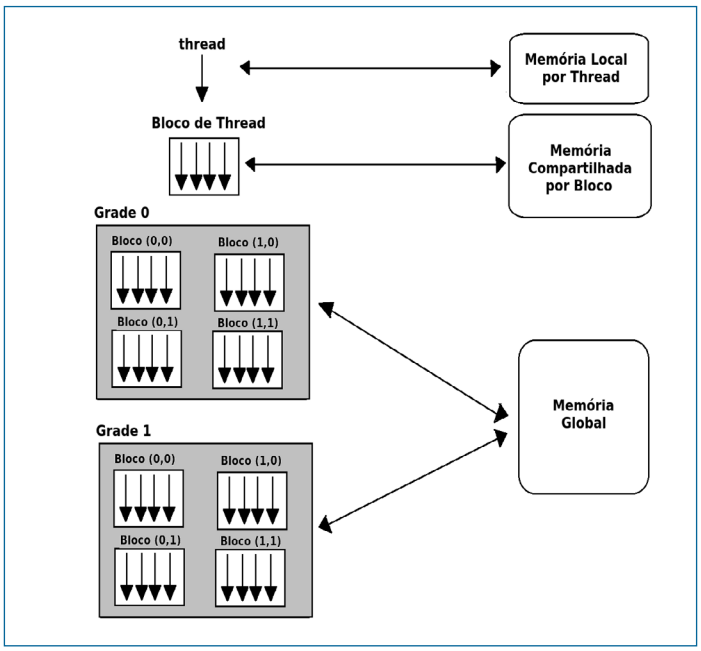


Figura 2: Modelo da distribuição de *threads* nas grades de blocos.

Ainda na Fig. 2, pode ser visualizada, por exemplo, a distribuição de 32 *threads* em duas grades bidimensionais de quatro blocos, também bidimensionais. Portanto, cada bloco conterá quatro *threads* que poderão se comunicar através do acesso à memória compartilhada presente no seu respectivo bloco.

A plataforma CUDA introduz algumas extensões à linguagem de programação C, dentre elas encontram-se os qualificadores

de tipo de função e os qualificadores de tipo de variável. Os qualificadores de tipo de função definem a unidade lógica de execução de uma função:

1. *__device__* : determina que uma função será executada no *device* sendo chamada a partir do próprio *device*;

2. *__host__* : determina que uma função será executada no *host* sendo chamada a partir do próprio *host*;

3. *__global__* : determina o que a plataforma CUDA chama de *kernel*, ou seja, uma função que é executada no *device* sendo chamada a partir do *host* e deve sempre retornar *void*. A chamada de um *kernel* é identificada através da sintaxe de configuração de execução $func <<< Dg,Db,Ns >>> (parâmetros)$. *func* é o nome da função que será executada em paralelo na GPU; Dg é do tipo *Dim3* e determina a dimensão da grade; Db é do tipo *size_t* e especifica o número de *bytes* da memória compartilhada que será alocado dinamicamente por cada bloco, caso não seja definida, por padrão será igual a zero; e $parâmetros$ representa as variáveis de entrada e saída da função.

Vale ressaltar que os qualificadores *__host__* e *__global__* não podem ser usados juntos. Já os qualificadores *__host__* e *__device__* podem ser usados conjuntamente, desde que a função seja executada tanto no *host* quanto no *device*. Os qualificadores de variáveis especificam o armazenamento das variáveis no *device*:

1. *__device__*: define que uma variável reside na memória global do *device*, a qual é acessível a todas as *threads* de uma grade e também pelo *host* através da biblioteca *runtime* da CUDA, o seu tempo de vida é o da aplicação;

2. *__shared__* : define que uma variável reside na memória compartilhada do *device*, sendo acessível apenas pelas *threads* de um mesmo bloco e o seu tempo de vida é o do bloco.

3. *__constant__*: define que uma variável reside na memória constante do *device*, sendo acessível pela grade. Ela é usada pelo compilador para carregar qualquer variável que aponte para a memória global. Identifica uma variável que é apenas de leitura no *kernel* e não depende do índice da *thread*, nem do bloco.

4. *texture* : define uma variável que reside na memória textura do *device*. Ela pode ser acessada pela API referência de textura na forma de um vetor ou uma matriz. É declarada como global estática, tendo a sintaxe: $texture < Type,Dim,ReadMode > Texref$, onde $Type$ é o tipo de dado *texel* (elemento de textura) retornado de um acesso à textura. Os elementos podem ser do tipo inteiro ou flutuante, sendo que os vetores podem ter 1, 2 ou 4 elementos embutidos, por exemplo, *uint2* e *float4*; Dim é a dimensão da memória textura, podendo ser 1, 2 ou 3. Caso não seja identificada a dimensão, o padrão é 1; e $ReadMode$ controla a conversão de *texel* retornada por um acesso, com duas opções de escolha. Ao ser substituído por *cudaReadModeElementType* nenhuma conversão será realizada. A segunda opção é a *cudaReadModeNormalizedFloat* e, se o *type* for do tipo inteiro (int), o valor devolvido é mapeado para $[-1,0,1,0]$ e $[0,0,1,0]$ para o tipo *unsigned int* (NVIDIA, 2016b).

As funções para alocar e desalocar os dados na memória no *device* são a $cudamalloc(\,)$ e a $cudafree(\,)$, respectivamente. Para a transferência de dados da memória do *host* para a memória do *device*, ou vice–versa, é usada a função $cudaMemcpy(\,)$. A diferença está no último argumento desta função: *cudaMemcpyHostToDevice* copia do *host* para o *device* enquanto que *cudaMemcpyDeviceToHost* copia do *device* para o *host*.

Algumas recomendações disponíveis nos manuais da API CUDA (NVIDIA, 2016b,a) devem ser levadas em consideração a fim de aumentar-se a eficiência computacional de um código numérico:

1. Minimizar a transferência de dados entre o *host* e o *device*;

2. Minimizar o uso da memória *global*. Se possível, usar a memória compartilhada (*shared*) ou criar *referências* para a memória textura (*texture*);

3. Evitar que os núcleos fiquem ociosos, ou seja, identificar quantos blocos de *threads* a GPU executa ao mesmo tempo;

4. Definir os vetores que tenham o tamanho e as exigências de alinhamento requeridas pela CUDA C: tipo *float4*.

Elas foram levadas em consideração quando da paralelização do simulador numérico desenvolvido neste trabalho.

4.2 Implementação de um *kernel* usando a API CUDA C

Para exemplificar a paralelização de uma função, é dada ênfase aos procedimentos requeridos para a paralelização da função que realiza a busca das partículas vizinhas, chamada de *DirectFind*. São apresentadas três formas de paralelizar essa função, uma usando apenas a memória *global* (Algoritmo 1), outra usando a memória *shared* (Algoritmo 2) e, por fim, uma otimizada usando a memória *texture* e o tipo *float4* (Algoritmo 3).

Nos *kernels*, o número máximo de *threads* por bloco depende da configuração da placa de vídeo, de modo que nenhum dos núcleos fique ocioso. Para obter-se o número de blocos, faz-se a divisão *NumParticles%numero_maximo_de_threads*, caso a divisão seja exata, do contrário adiciona-se uma unidade ao resultado da divisão, sendo que *NumParticles* representa o número total de partículas reais e virtuais, conforme a relação:

$$numBlocks = (a \,\%\, b\,!=0)?(1+a/b) : (a/b),$$

onde $a = NumParticles$ e $b = numero\_maximo\_de\_threads$.

4.2.1 Busca Direta paralelizada usando a memória global

A estrutura de dados foi implementada usando *vetores* ao invés de matrizes, sendo que as grandezas vetoriais como a velocidade, a posição e a aceleração foram armazenadas em variáveis do tipo *float4*, capazes de armazenar até quatro componentes de um vetor. Embora as grandezas vetoriais utilizadas necessitem apenas das três primeiras componentes, esta estrutura de dados tem o tamanho e as exigências de alinhamento requeridas pela CUDA C (NVIDIA, 2016b), pois cada componente é armazenada como um número real com precisão simples de 4 *bytes*, totalizando 16 *bytes* de memória, sendo necessária apenas uma instrução para acessar uma posição qualquer destes vetores. As grandezas escalares como o massa específica e a massa de cada partícula foram alocadas nas quartas componentes dos vetores velocidade e posição, respectivamente. Esses vetores são copiados para a memória textura, onde a redução no número de acessos, viabilizada pelo uso do tipo de dados *float4*, torna a leitura da memória textura mais rápida do que a leitura da memória global. A pressão é calculada através de uma função do tipo *device* sempre que necessário, dispensando-se a alocação vetorial.

O Algoritmo 1 mostra o esquema de um *kernel* usado na paralelização do método da Busca Direta de partículas, função $DirectFind(\cdots)$, lançado no *device*, usando a memória global. Nesse algoritmo, os índices identificando as partículas são alocados em um vetor do tipo *float* e *float3*. Neste caso, o tipo *float3* tem apenas três componentes, com 4 $bytes$ cada, totalizando 12 $bytes$, sendo necessárias três instruções para acessar os dados alocados nas componentes, uma vez que o tamanho e as exigências de alinhamento não são satisfeitas na API CUDA, tornando o acesso mais lento. Os índices das *threads* e dos blocos são identificados por *threadIdx.x* e *blockIdx.x*, respectivamente, e a dimensão dos blocos por *blockDim.x*.

Algoritmo 1: Busca Direta usando a memória global.

```
__global__ void KernelDirectFind(uint np,float *hsml,float3 *pos,...)
{
  uint i = __mul24(blockIdx.x,blockDim.x) + threadIdx.x;
  if(i >= np) return;
  float3 dxiac;
  float driac, mhsml;
  uint scale_k = paramsgpu.scalek;
  uint count = 0;
  for( uint j=0; j < np; j++ ){
      if(j==i)  continue;
      dxiac = pos[i] - pos[j];
      driac = dot(dxiac,dxiac);
      mhsml = (hsml[i] + hsml[j])/2.0f;
      if(driac < (scale_k*scale_k*mhsml*mhsml)){
      count++;
      neighbors[(i * paramsgpu.MaxNumberNeighbors + count)] = j;
      }
  }
  neighbors[(i*paramsgpu.MaxNumberNeighbors)] = count;
}
```

As partículas são mapeadas por $i = blockDim.x * blockId.x + threadIdx.x$ (linha 3). O mapeamento inicia dos índices locais das *threads* (*threadIdx.x*) para o índice global (*blockDim.x* blockId.x*). Na linha 12 é feito o cálculo da distância entre duas partículas. O valor da posição das partículas no vetor pos da partícula i é acessado na memória global, a cada incremento no *loop* (linhas 9-18), o que é desnecessário, uma vez que o valor de $pos[i]$ é constante (linha 11) no *loop*. Estes acessos repetitivos de dados reusáveis podem prejudicar o desempenho computacional. O mesmo acontece para a posição i do vetor escalar $hsml$ (linha 13).

4.2.2 Busca Direta paralelizada usando a memória compartilhada

A fim de minimizar-se o acesso à memória global, que tem uma maior latência, os dados reusáveis são copiados para as memórias de acesso rápido ou de menor latência. O Algoritmo 2 mostra a aplicação prática dessa ideia, através do uso da memória compartilhada. Primeiro criou-se os vetores na memória compartilhada (linhas 4-5), depois os valores do vetores posição pos e do comprimento de suavização $hsml$ para as partículas i foram copiados para os vetores pos_shared e $hsml\_shared$ (linhas 7-8), na memória compartilhada. Observe que na linha 9 há uma função chamada $\_\_syncthreads()$ no *kernel*. Essa função impede que uma ou mais *threads* avancem antes que todas elas tenham atingido este mesmo ponto. Ao utilizar a função $\_\_syncthreads()$, pode-se garantir que todas as *threads* estão na mesma iteração. Assim, assegura-se o acesso aos valores corretos dos vetores que estão na memória compartilhada. Vale ressaltar que o vetor pos_shared possui a dimensão do bloco, isto é, cada *thread* carrega apenas o valor da posição de uma partícula i. Desta forma, evita-se o acesso à memória de maior latência na obtenção do valor da posição da partícula i no *loop* (linhas 16-25), que são acessados no vetor pos_shared, na memória compartilhada (linha 13) e alocada em um escalar $posI$, obtendo-se um melhor desempenho.

Algoritmo 2: Busca Direta usando a memória compartilhada.

```
__global__ void KernelDirectFind(uint np,float *hsml,float3 *pos,...){
  uint ti = threadIdx.x;
  uint i = __mul24(blockIdx.x,blockDim.x) + threadIdx.x;
  __shared__ float3 pos_shared[DimBlocks];
  __shared__ float hsml_shared[DimBlocks];
  if(i >= np) return;
  pos_shared[ti]  = pos[i];
  hsml_shared[ti] = hsml[i];
  __syncthreads();
  float3 dxiac,posI ;
  uint scale_k = paramsgpu.scalek;
  float driac, mhsml,hsmlI;
  posI = pos_shared[ti];
  hsmlI = hsml_shared[ti];
  uint count = 0;
  for( uint j=0; j < np; j++ ){
      if(j==i) continue;
        dxiac = posI - pos[j];
        driac = dot(dxiac,dxiac);
        mhsml = (hsmlI + hsml[j])/2.0f;
        if(driac < (scale_k*scale_k*mhsml*mhsml)){
        count++;
        neighbors[(i * paramsgpu.MaxNumberNeighbors + count)] = j;
        }
  }
  neighbors[(i*paramsgpu.MaxNumberNeighbors)] = count;
}
```

4.2.3 Busca Direta paralelizada e otimizada

Uma forma de implementação otimizada é descrita no Algoritmo 3, onde o tipo *float3* foi substituído pelo tipo *float4*, garantindo-se o tamanho e as exigências de alinhamento. Além disso, os dados foram referenciados para a memória textura, que é muito mais rápida de ser acessada do que a memória global, sendo desnecessário fazer uma cópia dos dados para a memória compartilhada. Uma referência de textura é vinculada à memória textura usando-se a função $cudaBindTexture()$ para a memória linear, e a função $cudaUnbindTexture()$ é usada para desvincular-se a referência de textura.

4.2.4 O código numérico

Além da busca pelas partículas vizinhas (Buscas Direta e em Lista), todas as demais funções do código numérico foram paralelizadas usando-se a memória textura e o tipo $float4$ para os vetores. A Busca em Lista, com grade uniforme, foi paralelizada

Algoritmo 3: Busca Direta usando a memória textura.

```
texture<float4, 1, cudaReadModeElementType> PosTex;
__global__ void KernelDirectFind(uint np, uint *neighbors) {
  uint i = __mul24(blockIdx.x, blockDim.x) + threadIdx.x;
  if (i >= np) return;
  float4 p = tex1Dfetch(PosTex, i);
  float infRadius = paramsgpu.influenceRadius;
  uint countnp = 0;
  for (uint j = 0; j < np; j++){
      if (j == i) continue;
      float4 neib_pos = tex1Dfetch(PosTex, j);
      float3 relPos;
      float r;
      getNeibData(j,p, neib_pos, relPos, r);
      if (r <= infRadius) {
      neighbors[(i * paramsgpu.MaxNumberNeighbos + countnp)] =j;
      countnp++;
      }
  }
}
```

baseando-se no exemplo *particles* do *CUDA samples* (NVIDIA, 2016a) e foi efetivamente utilizada na obtenção dos resultados numéricos.

Conforme visto, o código numérico foi desenvolvido considerando-se a formulação do método WCSPH e paralelizado empregando-se a API CUDA C para a simulação de escoamento de fluidos newtonianos. Este código foi escrito apenas para seu uso em computadores com a plataforma Linux 64 *bits*, com suporte às unidades de processamento gráfico da NVIDIA, com capacidade a CUDA C e com o compilador gcc, da GNU. Ele está configurado para ser compilado e executado com a versão CUDA 7.5.

4.2.5 Eficiência computacional

Como um dos principais objetivos deste trabalho é a paralelização do código numérico, empregando a API CUDA, simulações foram realizadas de modo a avaliar-se a eficiência computacional da versão paralelizada. Portanto, a simulação da quebra de uma barragem bidimensional foi escolhida para a realização dos testes de eficiência computacional variando-se o número de partículas (partículas reais) empregadas na discretização do fluido que escoa (Góes et al., 2014).

Os resultados foram obtidos para simulações empregando-se 1.250, 2.500, 5.000 e 10.000 partículas de fluido e um tempo final de simulação igual a 2,3 s. Os valores dos tempos total de execução das versões paralelizada (GPU) e serial (CPU), assim como o *speedup* (relação entre os tempos de execução das versões em série e em paralelo), podem ser vistos na Tab. 1.

Conforme pode ser verificado, a versão paralelizada foi capaz de aumentar consideravelmente a eficiência do simulador numérico, atingindo para os casos considerados um *speedup* máximo de 29,82 para 10.000 partículas.

Tabela 1: Eficiência computacional.

Partículas	GPU (s)	CPU (s)	*Speedup*
1.250	138	807	5,85
2.500	332	3.570	10,75
5.000	689	12.487	18,12
10.000	2.096	62.505	29,82

5 Simulação de Vertedouros

A construção de obras hidráulicas é, geralmente, condicionada ao projeto de estruturas que possam controlar o escoamento de grandes volumes de água e altas pressões. Uma forma de extravio do excesso de fluido armazenado é requerida de forma a garantir uma operação segura de tais instalações. Com esse propósito, vertedouros surgem como uma forma de maximizar a perda de

energia quando do transporte do fluido de um nível a montante até um nível a jusante. Atuando como dissipadores de energia, os vertedouros possuem geometrias diversas, recebendo classificações variadas, mas sempre com a finalidade de conduzir o fluido até a base da estrutura sem que haja riscos de erosão e solapamento. Os desenvolvimentos atuais focam no projeto de estruturas que possam dissipar a energia de forma mais eficiente que as convencionais, aliado a uma redução de custos de construção e manutenção (Conterato, 2014).

Vertedouros do tipo em degraus surgem como uma opção neste contexto, nos quais uma grande parcela da energia é dissipada durante o escoamento através de seus componentes. Tal opção de construção se mostrou proeminente nas últimas décadas, devido a novas técnicas de construção que conciliam um menor custo de obra com uma maior dissipação de energia, em função do acabamento empregado nos degraus (Simões, 2008). Vertedouros com geometrias convencionais, onde empregam-se canais com calha em concreto liso, trazem uma pequena contribuição à perda de energia, conforme observado inicialmente por Peterka (1957) e posteriormente por Simões (2008). No entanto, trabalhos experimentais mostraram a possibilidade de ser alcançada uma dissipação de energia do escoamento da ordem de 60% com a presença de degraus nos vertedouros (Tozzi, 1992). Para chegar a estas conclusões, tais trabalhos empregam, em geral, um modelo físico em escala que possibilite a caracterização do escoamento ao longo do vertedouro (Simões, 2008).

No Brasil, conforme Arantes (2007), desde a década de 1980 são realizados estudos com modelos reduzidos de vertedouros em degraus, que objetivam verificar a ocorrência de cavitação e a configuração dos degraus que leva a uma maior perda de energia relativa. No entanto, o número de combinações geométricas e de configurações possíveis elevam consideravelmente os esforços empreendidos na execução de ensaios com tais modelos (Arantes, 2007).

Sanagiotto (2003) realizou um estudo experimental visando à caracterização do escoamento sobre vertedouros em degraus, buscando compreender o início do fenômeno da aeração no escoamento, a determinação das pressões médias e a avaliação da energia dissipada. Ele enfatizou, ainda, a dificuldade em se medir a posição da linha d'água após o início da aeração e sugeriu a realização de novos testes empregando modelos com maiores dimensões (Sanagiotto, 2003). Simões (2008) buscou definir critérios para a identificação dos diferentes regimes de escoamento que ocorrem, além de analisar a aeração do escoamento, os perfis de distribuição de pressões nos degraus e a cavitação através de uma compilação de resultados experimentais. O mesmo autor destacou a grande dificuldade de determinação de tais itens, uma vez que eles dependem do uso de métodos numéricos para a resolução das equações de Navier-Stokes associadas a modelos de turbulência (Simões, 2008).

Prá et al. (2012) buscaram, de forma experimental, determinar a posição de início da aeração e as pressões médias nos degraus bem como verificar, também, o comportamento da superfície livre d'água, considerando o caso de degraus com declividade de $45°$, restringindo seus resultados a vertedouros com alturas máximas de 25 m e com degraus com um máximo de $0{,}9$ m de altura. Conterato (2014) usou um modelo experimental para comparar os esforços de bacia de dissipação com e sem soleira, com vertedouros em degraus e vertedouros de calha lisa. No trabalho do referido autor, foram analisados dados de pressões médias e de flutuações de pressões, e foi proposta uma metodologia para o dimensionamento do tamanho e da posição de uma soleira.

O estabelecimento de parâmetros representativos para a energia dissipada, ao longo da calha do vertedouro, foi o foco do trabalho de Prá et al. (2012). Para tanto, eles fizeram uma análise experimental de quatro modelos físicos (três destes com degraus e um liso) na busca de relações entre a dissipação nas calhas, com e sem degraus, bem como para a dissipação de energia relativa a montante. Foi constatado que uma dissipação mais elevada é encontrada em escoamentos aerados e, também, nos degraus com maior altura.

Exemplos de trabalhos de simulação numérica do escoamento de vertedouros, no Brasil, podem ser encontrados em Fill (2011). Especificamente em Arantes (2007) para vertedouros com degraus, que empregou o *software* CFX na geração da malha computacional e na modelagem do escoamento. Lobosco e Schulz (2010), por sua vez, utilizaram o *software* de código-aberto OpenFoam na simulação do escoamento, tendo a malha computacional sido gerada pelo *software* Salome.

Os vertedouros com degraus podem ser divididos em dois regimes no que diz respeito ao escoamento Prá et al. (2012):

1. Regime de escoamento em quedas sucessivas (*nappe flow*): caracteriza-se por uma sucessão de quedas livres e com ressalto hidráulico pleno ou parcialmente desenvolvido.

2. Regime de escoamento deslizante sobre turbilhões (*skimming flow*): caracteriza-se por um escoamento principal deslizando sobre os degraus.

As simulações numéricas foram realizadas utilizando-se um nó do Laboratório de Mídia da Universidade Federal Fluminense (MediaLab), acessado via protocolo de rede SSH (*Secure Shell*), exclusivo para os usuários que se cadastraram no site da NVIDIA para o *GPU Test Drive*. Este equipamento possui uma placa NVIDIA Tesla K40, com 2.880 processadores CUDA e 12 GB de Memória RAM.

O simulador numérico foi testado mediante a resolução do mesmo problema de escoamento (bi e tridimensional, respectivamente) em um vertedouro com degraus de uma barragem fluvial. Os resultados calculados numericamente foram comparados com os obtidos experimentalmente por Chanson e Toombes (2002, 2003) e numericamente por Cheng et al. (2014).

5.1 Vertedouro bidimensional

Para a verificação do simulador numérico desenvolvido, na versão bidimensional, a geometria do problema foi a mesma que a descrita em Chanson e Toombes (2002) e Cheng et al. (2014) (Fig. 3). Neste modelo são construídos nove degraus, com altura de 0,1 m e 0,25 m de comprimento, com um ângulo de inclinação de $21{,}8°$. Os parâmetros usados na simulação numérica do escoamento no vertedouro bidimensional estão descritos na Tab. 2.

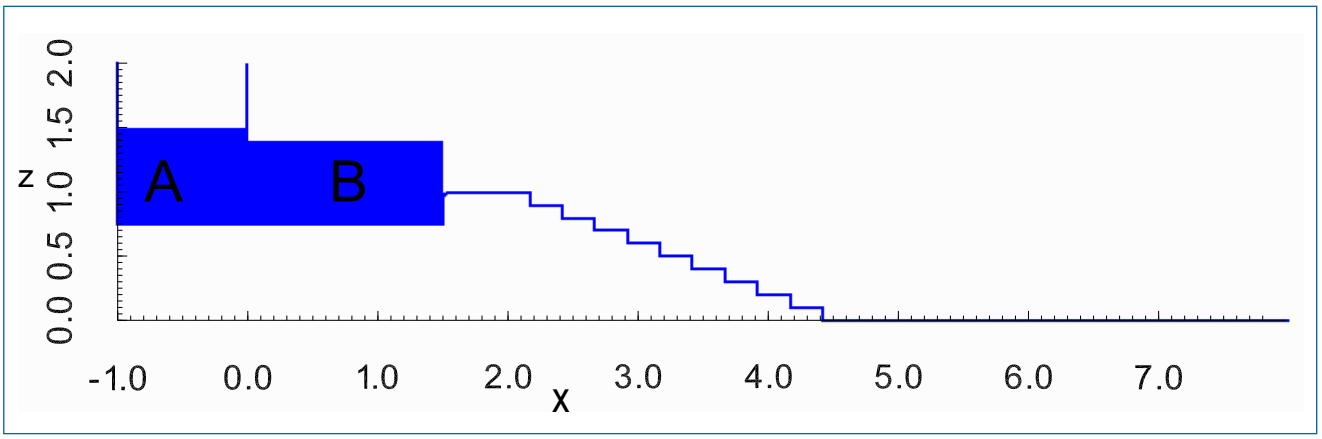
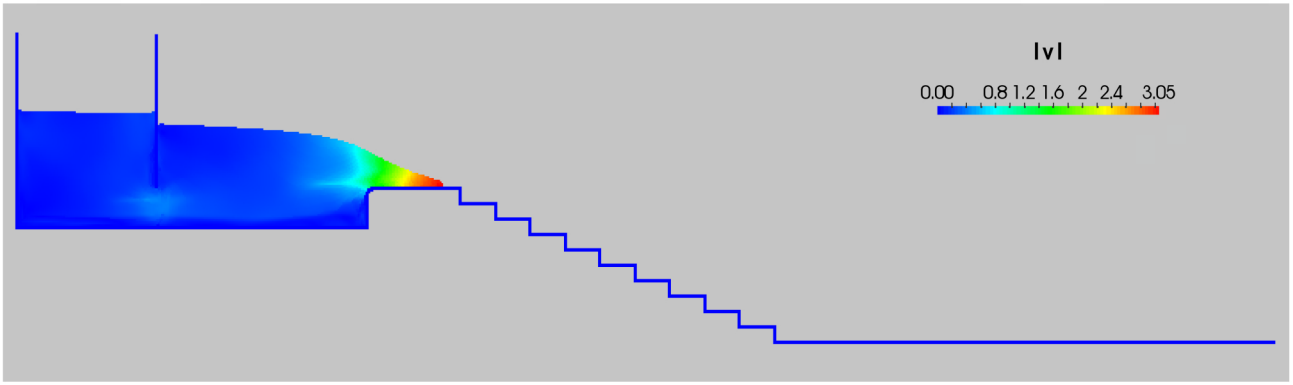


Figura 3: Geometria do vertedouro bidimensional.

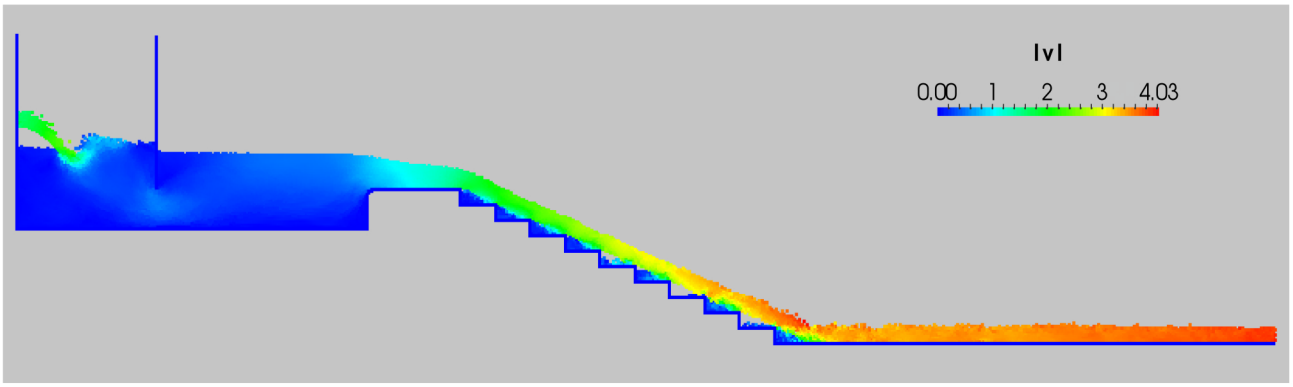
Tabela 2: Parâmetros utilizados para o problema bidimensional.

Parâmetro	Nomenclatura	Valor
Comprimento da coluna de água	l_x	1,0 m
Altura da coluna de água	l_z	0,75 m
Comprimento do domínio	L_x	4,5 m
Altura do domínio	L_z	1,5 m
Viscosidade cinemática	ν	10^{-7} m^2/s
Massa específica	ρ	10^3 kg/m^3
Velocidade inicial do fluido	$\mathbf{v}$	(0,0) m/s
Espaçamento entre as partículas	$\Delta x = \Delta y$	0,01 m
Comprimento de suavização	h	0,009994 m
Coeficiente da correção XSPH	ε	0,1
Coeficiente da correção APD	β	0,001
Coeficiente da correção δ-SPH	δ_{SPH}	0,1
Partículas de fluido	np_f	16.887
Partículas da borda	np_b	1.247
Passo de tempo	Δt	0,000114 s
Tempo de simulação	t_s	30,5 s

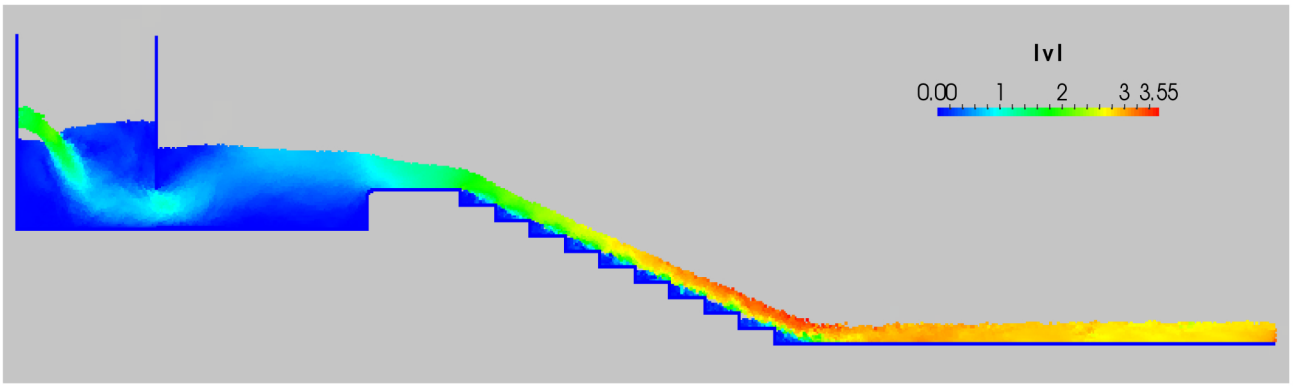
A Fig. 4 mostra os campos de velocidade no vertedouro com degraus com as partículas das fronteiras visíveis. Como pode ser visto, o domínio do problema é preenchido com as partículas que representam o fluido e com as partículas que definem as fronteiras, chamadas de partículas virtuais do tipo I. Em função do que já foi visto neste trabalho, sabe-se que elas emitem uma força repulsiva sobre as partículas de fluido que encontram-se próximas das fronteiras, impedindo que elas saiam do domínio físico, exceto na parte superior, pois não existem partículas virtuais nesta região. A visualização dos resultados, para diferentes instantes de tempos, foi possível mediante o uso do *software* Paraview (Rober, 2014) para a representação do domínio físico e geração dos campos de velocidade. As partículas de fluido que deixam o domínio computacional são reinjetadas no reservatório com velocidade constante de 0,25 m/s e com posição vertical determinada tomando-se como referência o nível de fluido no reservatório. O tempo total de execução com a placa Tesla K40 foi de 47,53 minutos para um tempo físico de simulação igual a 30,5 segundos, com a versão paralelizada.



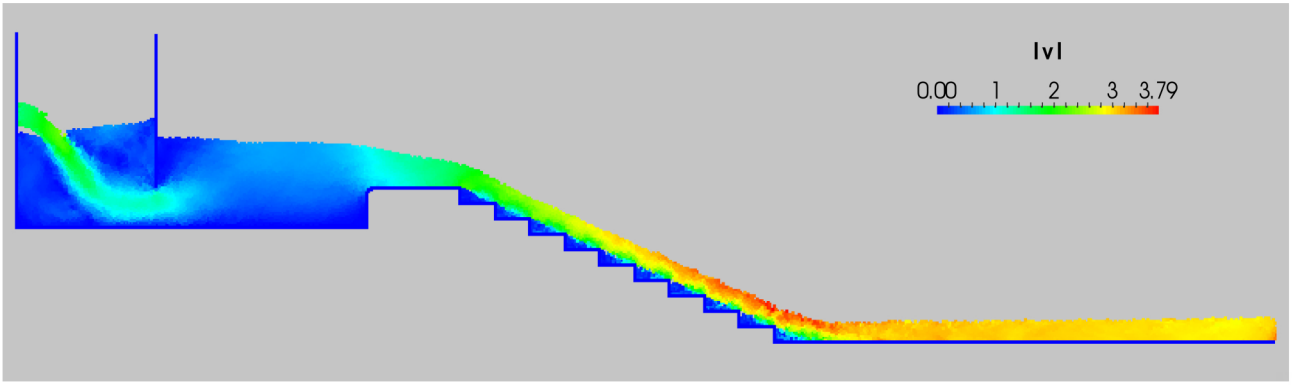
(a) inicial (t=0,3 s).



(b) intermediário (t=2,5 s).



(c) intermediário (t=5,4 s).



(d) final (t=30,5 s).

Figura 4: Campo de velocidades.

5.1.1 Campo de velocidade

Conforme descrito por Chanson e Toombes (2004), o problema simulado apresenta um escoamento com baixa turbulência e a mesma aparece a partir do quinto degrau, quando a camada limite atinge a superfície livre, em conformidade com o que foi observado nos experimentos realizados por Chanson e Toombes (2003). Cheng et al. (2014) mostraram os campos de velocidade para o mesmo escoamento, sendo que a magnitude da velocidade máxima, obtida numericamente, é igual a 3,8 m/s. A Fig. 5 mostra a variação da magnitude da velocidade obtida neste trabalho, com magnitude máxima sendo igual a 3,79 m/s para $t = 30{,}5$ s, final da simulação, mostrando uma boa concordância entre os dois valores. A variação da velocidade aumenta entre o quarto e o quinto degraus na direção do escoamento, caracterizando o início do escoamento turbulento, sendo que a maior variação da velocidade ocorre próxima à superfície livre.

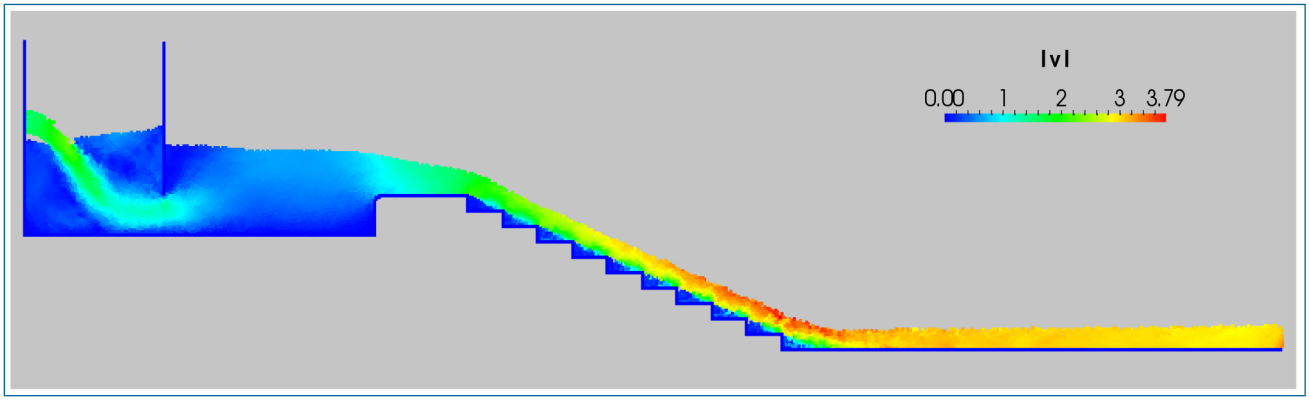


Figura 5: Perfil de velocidade para o vertedouro bidimensional (t=30,5 s).

Na Fig. 6, é possível observar as recirculações capturadas pela simulação nos degraus 1-4, devido às tensões de cisalhamento causadas pelo escoamento acima da extremidade dos degraus, sendo que o degrau número 3 foi ampliado para uma melhor visualização da recirculação. As recirculações são importantes neste tipo de escoamento uma vez que elas dissipam a energia do fluido nos vertedouros em degraus. Segundo Heidari e Ghasemi (2014), os principais elementos que levam à dissipação de energia no vertedouro em degraus são a geometria do vertedouro (a altura do vertedouro, o ângulo de inclinação do vertedouro e a quantidade e altura dos degraus) e os parâmetros hidráulicos (descarga, velocidade do escoamento e profundidade crítica).

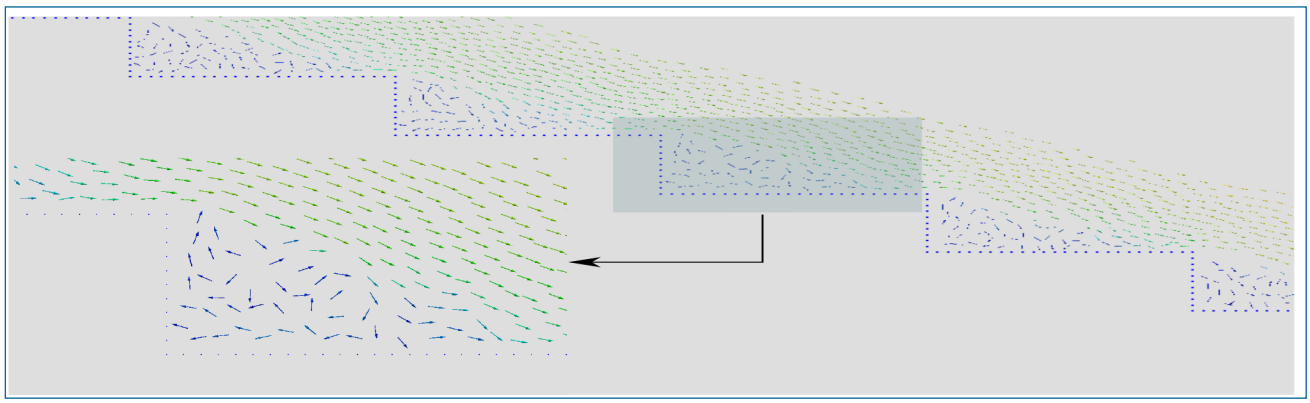


Figura 6: Captura das recirculações nos degraus para o escoamento bidimensional (t=30,5 s).

Outra característica importante do escoamento de um fluido em um vertedouro com degraus é a velocidade na superfície livre, uma vez que a superfície livre é difícil de ser capturada pelos métodos numéricos convencionais, dependentes do uso de malhas computacionais. Neste tipo de escoamento, o perfil de velocidades tende a aumentar da base dos degraus em direção à superfície livre na região com o escoamento turbulento, ou seja, o escoamento ao longo de um vertedouro em degraus é turbulento, variando rapidamente com o tempo, e existem recirculações na concavidade dos degraus (Chanson e Toombes, 2003).

González (2005) obteve vários resultados de forma experimental para a determinação do comportamento do perfil de velocidade na região aerada do escoamento, ou seja, a partir do quinto degrau. A magnitude da velocidade adimensionalizada (V/V_{max}) foi obtida em função da ordenada adimensionalizada Y/Y_{90}, onde Y_{90} é o valor da ordenada onde a concentração de ar fica em torno de 90% da mistura água-ar. Como neste trabalho o simulador numérico é monofásico, a velocidade adimensionalizada foi obtida considerando-se que Y_{90} corresponde ao valor máximo da ordenada y perpendicular à direção do escoamento nos degraus,

a partir da rotação do plano cartesiano xy de $21.8°$ no sentido horário. Os resultados correspondentes à variação da magnitude da velocidade nos degraus 5 - 9 são apresentados na Fig. 7 para t=30,5 s. Vale ressaltar que a magnitude da velocidade foi considerada desde o interior dos degraus até a superfície livre, caracterizando dois perfis de velocidade adimensionalizada:

1. Uma variação entre $0{,}0$ e $0{,}3$ m/s foi obtida no interior dos degraus, onde ocorre as recirculações devido as tensões de cisalhamento causadas pelo escoamento acima da extremidade dos degraus;

2. Uma variação entre $0{,}7$ e $1{,}0$ m/s em todos os degraus foi obtida para a região externa aos degraus até a superfície livre, cujo perfil de velocidade é semelhante em todos os degraus.

Na Fig. 7(f), todos os resultados encontram-se representados num único gráfico, configurando o comportamento do perfil de velocidade do escoamento nesta região. Nela, também encontram-se os valores adimensionalizados da velocidade dada segundo a lei de potência

$$\frac{V}{V_{max}} = \left(\frac{y}{y_{max}}\right)^{1/n} \tag{10}$$

derivada da equação correspondente fornecida por González (2005) para o escoamento com aeração. Segundo esse autor, os valores experimentais foram corretamente reproduzidos para 7,8<n<11,8. Na referida figura, foram representadas as curvas (Eq. (10)) para n=7,8 e 11,8.

Em função dos resultados obtidos, pode-se concluir que o simulador numérico reproduziu corretamente os perfis de velocidade, quando os mesmos são comparados com aqueles obtidos por Chanson e Toombes (2002, 2003, 2004), González (2005) e Cheng et al. (2014) para o escoamento aerado.

5.1.2 Campo de pressão

A análise do campo de pressão em escoamentos em vertedouros com degraus é importante no sentido de prevenir o fenômeno de cavitação nos degraus, em decorrência do surgimento de pressões negativas, de modo a viabilizar o projeto de estruturas cada vez mais altas, submetidas a maiores vazões e com degraus com maiores alturas (Prá et al., 2012). Ainda segundo Prá et al. (2012), se a estrutura for idealizada de forma inadequada, os degraus do vertedouro serão submetidos a maiores velocidades e apresentarão escoamentos sem aeração (penetração do ar na água) propiciando o surgimento da cavitação. Tozzi (2004) menciona o fato de que o uso do concreto compactado com rolo (RCC), adotado na construção de vertedouros com degraus, quando submetido a determinadas vazões específicas e para variações de velocidades mais elevadas, aumenta o risco do aparecimento da cavitação, principalmente na região não aerada. Como consequência, tem-se o comprometimento da dissipação de energia, pondo em risco a base da estrutura e acarretando na necessidade de manutenção periódica dos degraus.

A Fig. 8 mostra a variação da magnitude do campo de pressão em quatro *frames* (instantes de tempo) durante o período de simulação numérica, sendo que no último *frame*, Fig. 8(d), a pressão varia entre -3.484,53 e 21.988,95 Pa, que convertidos para metros de coluna de água (mca) ficam entre -0,36 e 2,24 mca. Tozzi (2004) mostrou que pode ocorrer cavitação para pressões abaixo de -9,0 mca, ou -88.200 Pa, para um vertedouro igual ao que foi aqui estudado.

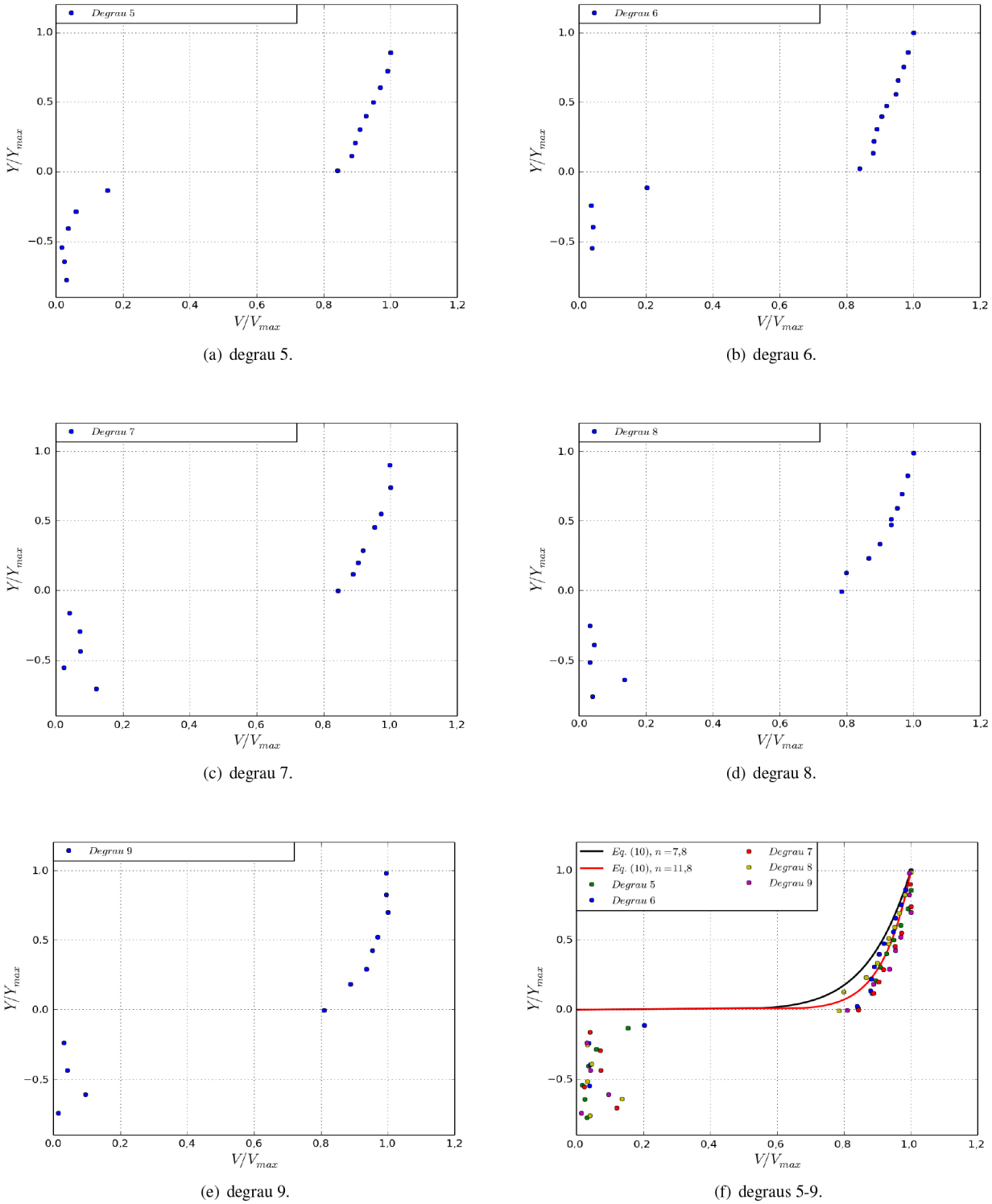
5.2 Vertedouro tridimensional

Como exemplo de aplicação da versão tridimensional do código computacional, foi empregado o mesmo problema físico considerado para a simulação bidimensional. Agora, os resultados numéricos são apenas analisados quanto à capacidade do simulador em reproduzir realisticamente o escoamento em vertedouros com degraus, através da visualização dos campos de velocidade e pressão com o uso do programa Paraview.

A Fig. 9 mostra a geometria do domínio físico tridimensional, com vistas lateral e superior. Na Fig. 9(a) (vista lateral) pode-se observar as partículas da borda que formam as paredes laterais do vertedouro e na Fig. 9(b) tem-se uma vista superior do vertedouro, embora a base do vertedouro e as partículas de fluido apareçam como um meio contínuo, devido à aplicação de filtros do *software* Paraview: *Delaunay2D* para as partículas da borda inferior e *Delaunay3D* para as partículas de fluido, o domínio computacional é discretizado, na sua totalidade, por partículas.

Como no método SPH o domínio do problema é representado por partículas, é necessária uma grande quantidade de partículas para gerar o domínio computacional (incluindo o vertedouro e o fluido). Tal fato influencia diretamente no custo computacional e na capacidade de obter-se resultados realísticos. A quantidade de partículas usada para representar o domínio do problema foi igual a 120.516 partículas. Destas, 73.245 são partículas de fluido (partículas reais) e 45.391 são partículas da borda (partículas virtuais do tipo I). Os demais parâmetros empregados nesta simulação encontram-se na Tab. 3.

Para a simulação do escoamento tridimensional foram necessários 241,83 minutos de tempo de execução para atingir o tempo final (físico) de simulação correspondente a 30,5 segundos, ressaltando que foi utilizado um total de 120.516 partículas. Apesar do código numérico ter sido paralelizado, vê-se que a simulação de um problema físico real pode exigir um esforço computacional considerável, mesmo para uma placa Tesla K40 da NVIDIA.



(a) degrau 5.

(b) degrau 6.

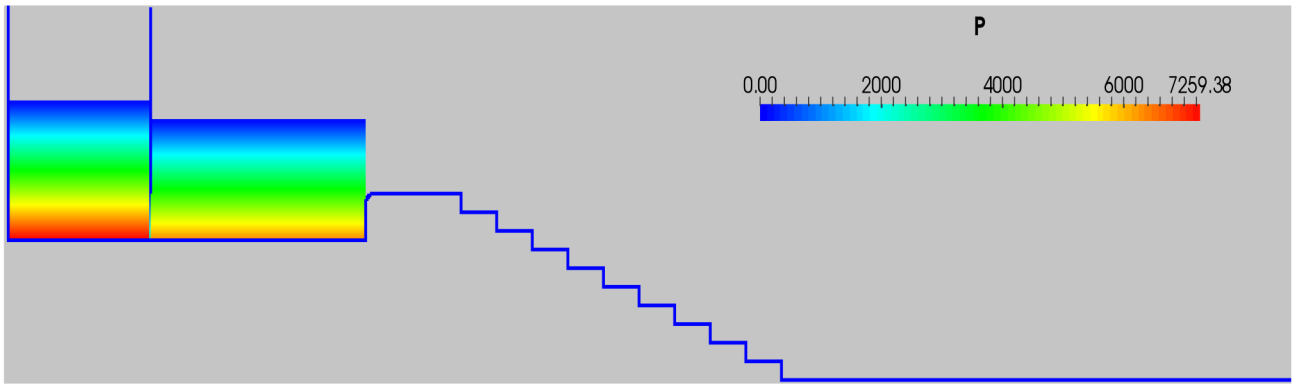
(c) degrau 7.

(d) degrau 8.

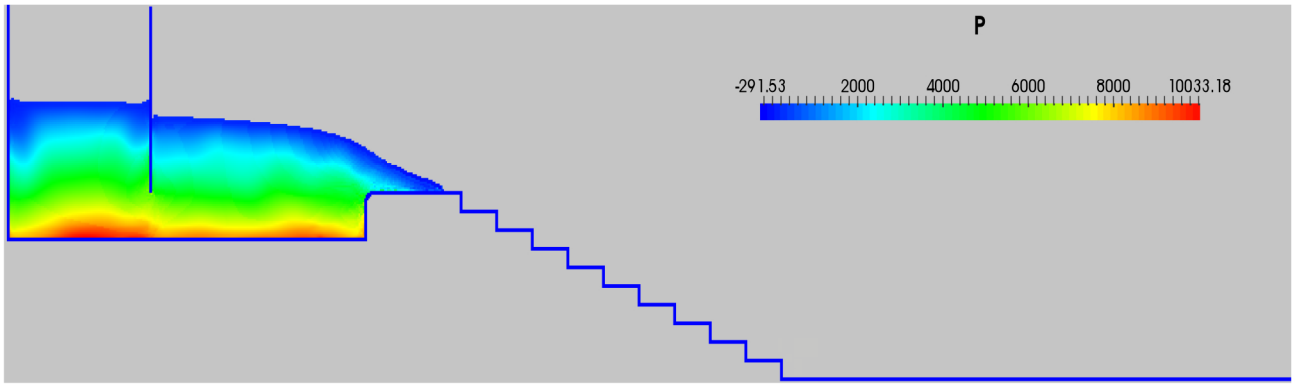
(e) degrau 9.

(f) degraus 5-9.

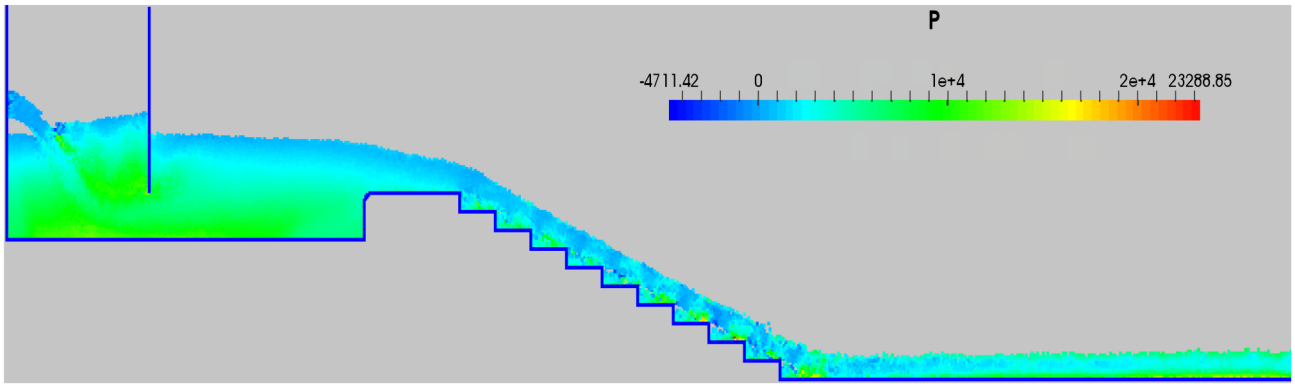
Figura 7: Perfis de velocidade nos degraus de 5 a 9 para t=30,5 s.



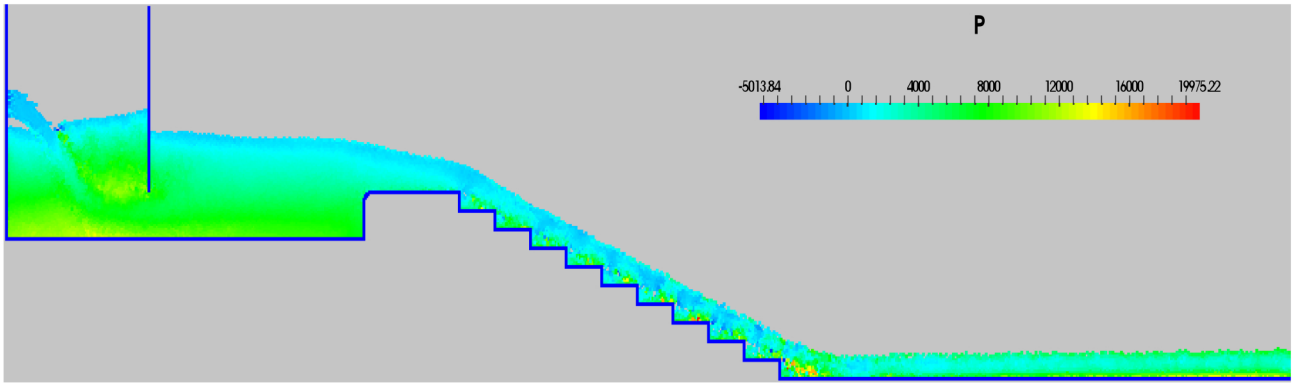
(a) inicial (t=0 s).



(b) intermediário (t=0,3 s).

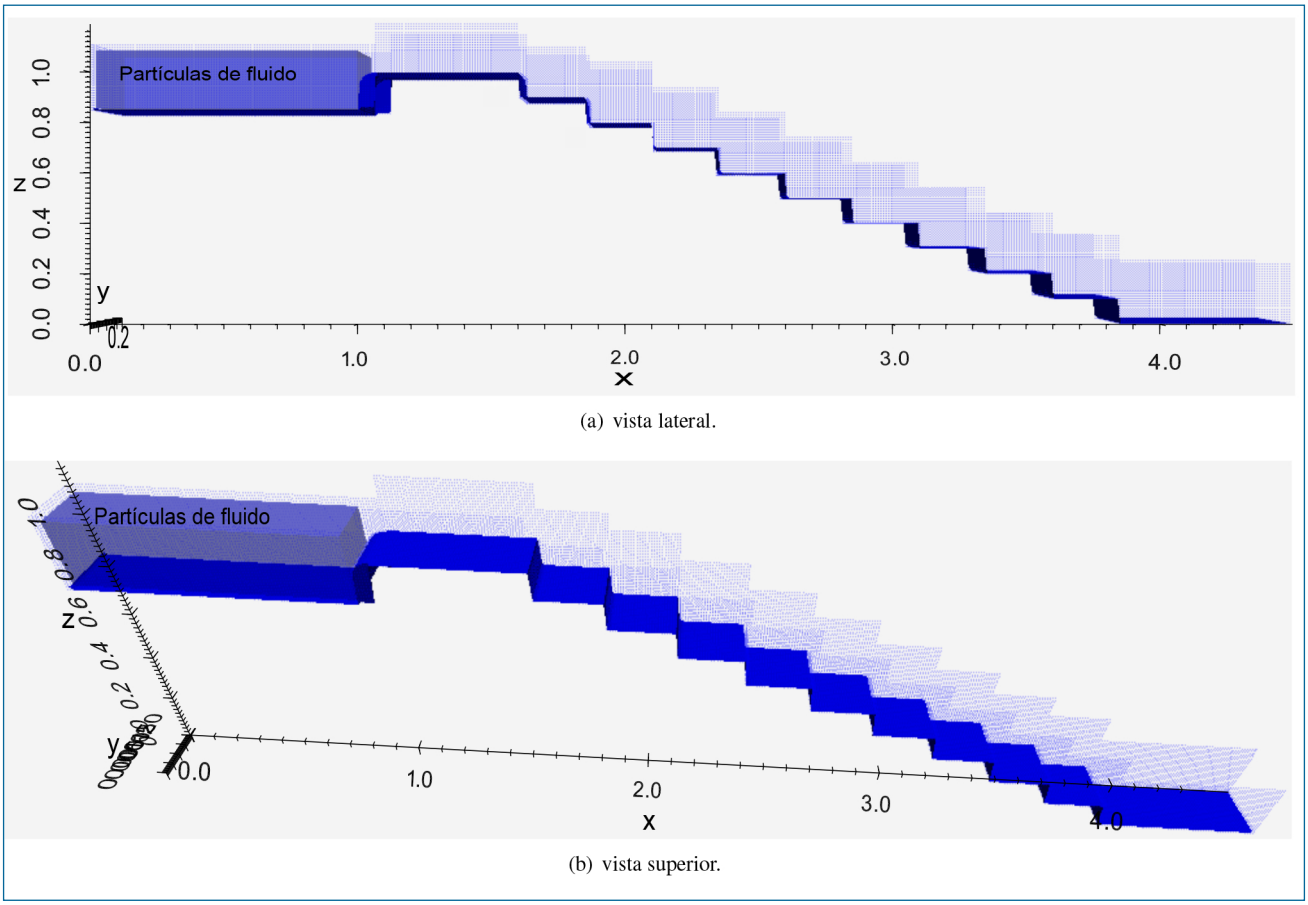


(c) intermediário (t=5,4 s).



(d) final (t=30,5 s).

Figura 8: Campo de pressão.



(a) vista lateral.

(b) vista superior.

Figura 9: Geometria do vertedouro tridimensional.

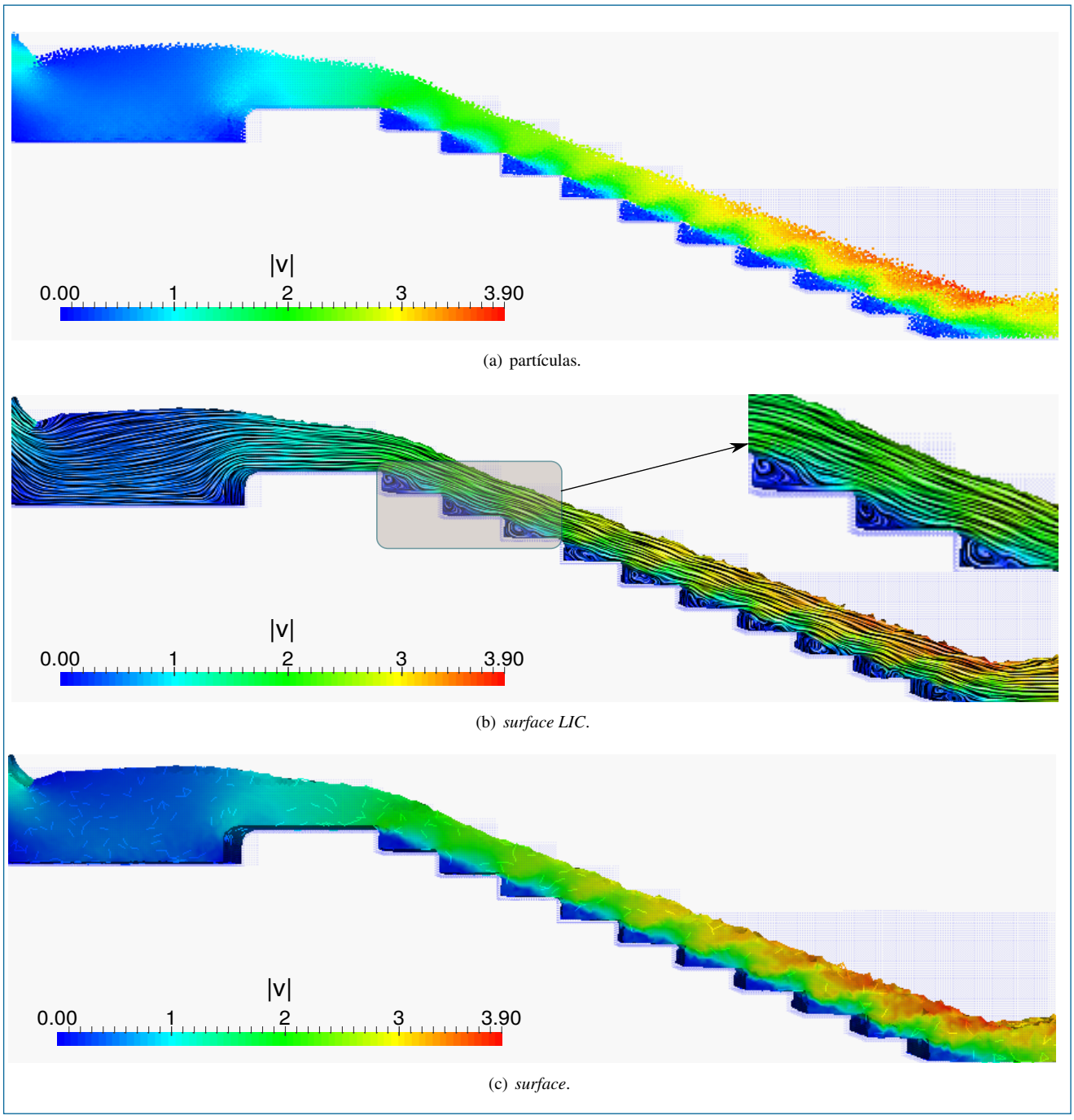
Tabela 3: Parâmetros utilizados para o problema tridimensional.

Parâmetro	Nomenclatura	Valor
Comprimento da coluna de água	l_x	1,0 m
Altura da coluna de água	l_z	0,3 m
Largura da coluna de água	l_z	0,2 m
Comprimento do domínio	L_x	4,5 m
Altura do domínio	L_z	1,5 m
Largura do domínio	L_z	0,3 m
Viscosidade cinemática	ν	10^{-7} m^2/s
Massa específica	ρ	10^3 kg/m^3
Velocidade inicial do fluido	$\mathbf{v}$	(0,0,0) m/s
Espaçamento entre as partículas	$\Delta x = \Delta y = \Delta z$	0,01 m
Comprimento de suavização	$hsml$	0,012168 m
Coeficiente da correção XSPH	ε	0,1
Coeficiente da correção APD	β	0,001
Coeficiente da correção δ-SPH	δ_{SPH}	0,1
Partículas de fluido	np_f	73.245
Partículas da borda	np_b	45.391
Passo de tempo	Δt	5×10^{-5} s
Tempo de simulação	t_s	30,5 s

5.2.1 Campo de velocidade

Os resultados obtidos com a versão tridimensional seguem os mesmos padrões dos obtidos com a versão bidimensional, embora o vertedouro agora não possua uma comporta. Na Fig. 10, apresenta-se o campo de velocidade de três formas diferentes e para o tempo final de simulação. Na Fig. 10(a), destaca-se o campo de velocidade e a magnitude da velocidade máxima é igual a 3,90

m/s para $t = 30.5$s. Já na Fig. 10(b), a representação do campo de velocidade foi obtida aplicando-se o filtro *Delaunay3D* e o *plugin Surface LIC*, ressaltando-se as linhas de correntes e as recirculações no interior dos degraus. A convolução da integral de linha, *Line Integral Convolution* (LIC), é um método de visualização para campos vetoriais capaz de tornar possível a sua geração no todo ou em partes (Rober, 2014), produzindo linhas de correntes que são tangentes ao campo vetorial de velocidade. Na última figura, Fig. 10(c), realça-se a visualização da superfície livre, capturada pelo filtro *Delaunay3D* e com o uso do *plugin Surface*, mostrando que a região do escoamento turbulento aumenta a partir do quinto degrau, onde a variação da velocidade é maior, conforme o mapa de cores, o que está de acordo com os resultados de Chanson e Toombes (2003) e Cheng et al. (2014).



(a) partículas.

(b) *surface LIC*.

(c) *surface*.

Figura 10: Campo de velocidade.

Um outro resultado importante consiste em caracterizar-se as recirculações no interior dos degraus, causadas pelas tensões de cisalhamento existentes no escoamento externo aos degraus e necessárias para a dissipação da energia nos vertedouros em degraus. As recirculações são mostradas na Fig. 11, onde os primeiros degraus foram ampliados para que a recirculação fosse visualizada com mais clareza. Essa figura foi criada com a ferramenta *glyphVector* do Paraview.

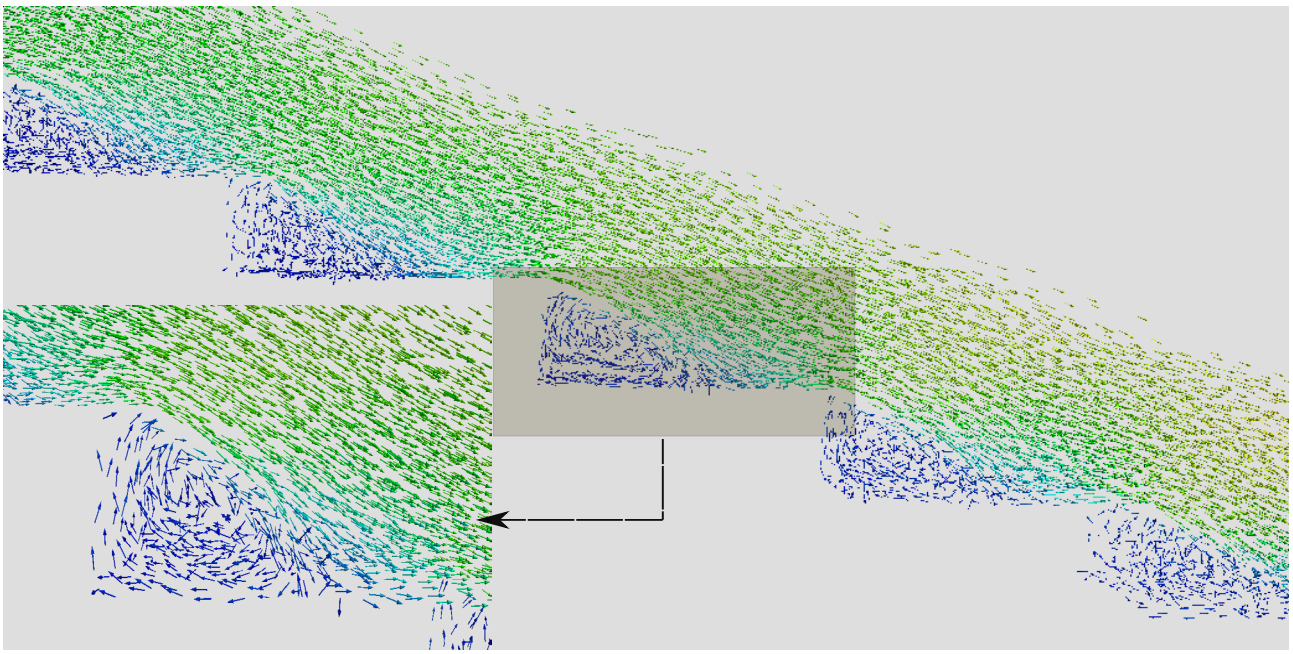


Figura 11: Visualização das recirculações nos degraus.

5.2.2 Campo de pressão

Os resultados numéricos obtidos para o campo de pressão estão ilustrados na Fig. 12. O campo de pressão sofre rápidas variações nos degraus, sendo que a maior pressão ocorre nas superfícies horizontais dos degraus (pisos), onde o impacto do fluxo sobre os degraus é maior e vai diminuindo em direção à superfície livre, a partir do limite vertical dos degraus. Conforme mencionado anteriormente, o estudo do campo de pressão é importante para o desenvolvimento de vertedouros com degraus com capacidade de receber maiores vazões sem a existência do risco da ocorrência de cavitação nos mesmos.

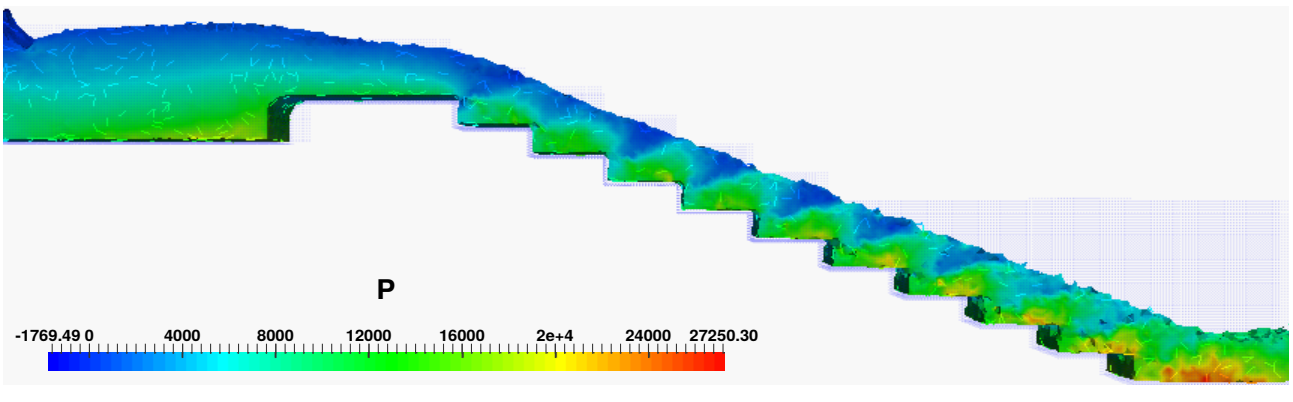


Figura 12: Visualização do campo de pressão.

6 Conclusões

Neste trabalho, um simulador numérico é desenvolvido usando a API CUDA C da NVIDIA. O código numérico é implementado voltado para a simulação de problemas que envolvam o escoamento bi e tridimensional de fluidos newtonianos com superfícies livres, empregando-se o método de partículas lagrangiano, livre de malhas, *Smoothed Particle Hydrodynamics*. Algumas técnicas de programação paralela são apresentadas, bem como outros aspectos numéricos inerentes ao método SPH. Além disso, a fim de poder-se considerar o caso de escoamentos turbulentos, uma equação específica para o tensor de tensões é introduzida.

Como exemplo de aplicação considera-se o escoamento bi e tridimensional em um vertedouro com degraus de uma barragem fluvial. Na versão bidimensional do simulador, as velocidades nos degraus onde sabidamente o escoamento é turbulento são consideradas. Isto propicia a comparação do perfil da velocidade, na referida região, com os resultados obtidos por Chanson e

Toombes (2002, 2003, 2004); González (2005) e Cheng et al. (2014). Pode-se concluir que os valores calculados são condizentes com os previstos nesses trabalhos.

Devido às características inerentes ao método SPH, a simulação de problemas reais tridimensionais, possivelmente em tempo real, torna-se competitiva do ponto de vista computacional, quando comparada com os métodos tradicionais de resolução de equações diferenciais parciais, mediante a paralelização do código numérico. No caso do escoamento bidimensional, vê-se que um *speedup* de até 29,82 pode ser obtido com o uso de 10.000 partículas de fluido para o problema da quebra de uma barragem (Góes et al., 2014).

Com o uso do método SPH é possível capturar-se sem maiores dificuldades a superfície livre do escoamento bidimensional, bem como as recirculações nas concavidades dos degraus e o campo de velocidade nessas regiões. Ademais, a partir da determinação do campo de pressão, não verifica-se a presença do fenômeno prejudicial da cavitação nos degraus, que para o problema estudado surge para pressões negativas abaixo de -88.200 Pa.

Por outro lado, o problema tridimensional do escoamento em um vertedouro com degraus permite avaliar-se a capacidade do código numérico desenvolvido na resolução de problemas reais. Os mesmos padrões de escoamento obtidos na versão bidimensional também foram reproduzidos e são condizentes com a física do problema. O uso do Paraview possibilita uma visualização realística do comportamento do escoamento. A sua utilização permite que as principais características presentes no escoamento no vertedouro com degraus, tais como os campos de velocidade e pressão, as linhas de corrente, as recirculações e a superfície livre, sejam ressaltadas e nitidamente representadas.

Finalmente, entende-se que o simulador encontra-se apto a ser utilizado como uma ferramenta auxiliar na execução de projetos de novos vertedouros com degraus, por exemplo, a fim de minimizar-se o custo, tanto computacional quanto financeiro, em comparação com os necessários para a obtenção de resultados equivalentes com o uso de modelos experimentais, embora estes últimos sejam indispensáveis.

Agradecimentos

Os autores gostariam de agradecer os recursos financeiros fornecidos pela CAPES e pelo CNPq.

Referências

Antuono, M., Colagrossi, A., Marrone, S., Molteni, D. (2010). Free-surface flows solved by means of SPH schemes with numerical diffusive terms. *Computer Physics Communications*, *181*(3), 532–549.

Antuono, M., Colagrossi, A., Marrone, S. (2012). Numerical diffusive terms in weakly-compressible SPH schemes. *Computer Physics Communications*, *183*(12), 2570 – 2580.

Arantes, E. J. (2007). Caracterização do escoamento sobre vertedouros com degraus via CFD. Tese de Doutorado, Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos.

Chanson, H., Toombes, L. (2002). Air–water flows down stepped chutes: turbulence and flow structure observations. *International Journal of Multiphase Flow*, *28*(11), 1737–1761.

Chanson, H., Toombes, L. (2003). Strong interactions between free-surface aeration and turbulence in an open channel flow. *Experimental Thermal and Fluid Science*, *27*(5), 525 – 535.

Chanson, H., Toombes, L. (2004). Hydraulics of stepped chutes: The transition flow. *Jornal of Hydraulic Research*, *42*(1), 43–54.

Cheng, X., Gulliver, J. S., Zhu, D. (2014). Application of displacement height and surface roughness length to determination boundary layer development length over stepped spillway. *Water*, *6*(12), 3888.

Conterato, E. (2014). Determinação de critérios de dimensionamento de soleira terminal em bacia de dissipação a jusante de vertedouro em degraus. Dissertação de Mestrado, Instituto de Pesquisas Hidráulicas, Universidade Federal do Rio Grande do Sul, Porto Alegre.

Crespo, A. J. C. (2008). Application of the smoothed particle hydrodynamics model SPHysics to free surface hydrodynamics. Tese de Doutorado, Universidade de Vigo.

Dalrymple, R., Knio, O. (2001). Sph modelling of water waves. *Proc Coastal Dynamics*, p 779–787.

Dalrymple, R., Rogers, B. (2006). Numerical modeling of water waves with the SPH method. *Coastal Engineering*, *53*(2–3), 141 – 147.

Federico, I., Marrone, S., Colagrossi, a., Aristodemo, F., Antuono, M. (2012). Simulating 2D open-channel flows through an SPH model. *European Journal of Mechanics - B/Fluids*, *34*, 35–46.

Fill, G. C. (2011). Modelagem hidrodinâmica 3D de escoamentos em vertedouros. Dissertação de Mestrado, Setor de Tecnologia, Universidade Federal do Paraná, Curitiba.

Gingold, R. A., Monaghan, J. J. (1982). Kernel estimates as a basis for general particle method in hydrodynamics. *Journal of Computational Physics*, (46), 429–453.

Góes, J. F. (2011). Resolução numérica de escoamentos compressíveis empregando um método de partículas livre de malhas e o processamento paralelo (CUDA). Mestrado em Modelagem Computacional, Instituto Politécnico, Universidade do Estado do Rio de Janeiro, Rio de Janeiro.

Góes, J. F., Góes, M. L., Amaral Souto, H. P. (2014). Estudo comparativo das versões serial e paralela (CUDA) do método SPH empregando-se uma técnica iterativa para a determinação do campo de pressões. Em: *Anais do XXXV Ibero Latin American Congress on Computational Methods in Engineering*, Fortaleza, CE.

González, C. A. (2005). An experimental study of free-surface aeration on embankment stepped chutes. Tese de Doutorado, Department of Civil Engineering, Faculty of Engineering, Physical Sciences and Architecture, The University of Queensland, Brisbane, Australia.

Gotoh, H., Shao, S., Memita, T. (2004). SPH-LES model for numerical investigation of wave interaction with partially immersed breakwater. *Coastal Engineering Journal*, *46*(01), 39–63.

Heidari, A., Ghasemi, P. (2014). Evaluation of steps slope on energy dissipation in stepped spillway. *International Journal of Engineering & Technology*, *3*(4), 501.

Lee, E. S., Moulinec, C., Xu, R., Violeau, D., Laurence, D., Stansby, P. (2008). Comparisons of weakly compressible and truly incompressible algorithms for the SPH mesh free particle method. *Journal of Computational Physics*, *227*(18), 8417–8436.

Li, S. (2004). *Meshfree particle methods*. Springer, Berlin; New York.

Liu, G. R. (2002). *Mesh free methods: moving beyond the finite element method*. CRC Press, Boca Raton Fla..

Liu, G. R. (2003). *Smoothed particle hydrodynamics: A Meshfree Particle Method*. World Scientific, Singapore.

Lo, E. Y., Shao, S. (2002). Simulation of near-shore solitary wave mechanics by an incompressible SPH method. *Applied Ocean Research*, *24*(5), 275 – 286.

Lobosco, R. J., Schulz, H. E. (2010). Análise computacional do escoamento em estruturas de vertedouros em degraus. *Mecánica Computacional, Asociación Argentina de Mecánica Computacional*, *XXIX*.

Lucy, L. B. (1977). Numerical approach to testing the fission hypothesis. *Astronomical Journal*, (82), 1013–1024.

Malidi, A., Dufour, S., N'dri, D. (2005). A study of time integration schemes for the numerical modelling of free surface flows. *Int J Numer Meth Fluids*, (48), 1123–1147.

Marrone, S., Antuono, M., Colagrossi, A., Colicchio, G., Touzé, D. L., Graziani, G. (2011). δ-sph model for simulating violent impact flows. *Computer Methods in Applied Mechanics and Engineering*, *200*(13–16), 1526 – 1542.

Marrone, S., Colagrossi, a., Antuono, M., Colicchio, G., Graziani, G. (2013). An accurate SPH modeling of viscous flows around bodies at low and moderate Reynolds numbers. *Journal of Computational Physics*, *245*(July), 456–475.

Molteni, D., Colagrossi, A. (2009). A simple procedure to improve the pressure evaluation in hydrodynamic context using the SPH. *Computer Physics Communications*, *180*(6), 861 – 872.

Monaghan, J. J. (1989). On the problem of penetration in particle methods. *Journal of Computational Physics*, *82*, 1–15.

Monaghan, J. J. (1994). Simulating free surface flow with SPH. *Journal of Computational Physics*, *110*, 399–406.

NVIDIA (2016a). *CUDA SAMPLES: Reference Manual*. NVIDIA Corporation, version 7.5 edn.

NVIDIA (2016b). *NVIDIA CUDA C: Programming Guide*. NVIDIA Corporation, version 7.5 edn.

Ozbulut, M., Yildiz, M., Goren, O. (2014). A numerical investigation into the correction algorithms for SPH method in modeling violent free surface flows. *International Journal of Mechanical Sciences*, *79*, 56 – 65.

Peterka, A. J. (1957). Hydraulic design of stilling basins and energy dissipation. *Washington US Govt*.

Prá, M. D., Suzuki, L. E. A., Alves, A. A. M., Marques, M. G. (2012). Um estudo sobre vertedouros em degraus de declividade 1v:1h. *Associação Portuguesa de Recursos Hídricos*, *33*(01), 17–28.

Ricardo, A. M., Canelas, R. B., Ferreira, R. M. L. (2016). *Characterization of vortex interaction with Lagrangian Coherent Structures*. River Flow 2016, Taylor & Francis Group.

Rober, N. (2014). *Paraview tutorial: for the visualization of Earth - and climate science data*.

Sanagiotto, D. G. (2003). Características do escoamento sobre vertedouros em degraus de declividade 1v:0,75 h. Dissertação de Mestrado, Instituto de Pesquisas Hidráulicas, Universidade Federal do Rio Grande do Sul, Porto Alegre.

Shadloo, M. S., Zainali, A., Yildiz, M., Suleman, A. (2012). A robust weakly compressible sph method and its comparison with an incompressible sph. *International Journal for Numerical Methods in Engineering*, *89*(8), 939–956.

Simões, A. L. A. (2008). Considerações sobre a hidráulica de vertedouros em degraus. Dissertação de Mestrado, Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos.

Solenthaler, B., Pajarola, R. (2009). Predictive-corrective incompressible sph. *ACM Transactions on Graphics*, *28*(3), 1.

Tozzi, M. (1992). Caracterização/comportamento de escoamentos em vertedouros com paramento em degraus. Tese de Doutorado, USP, São Paulo.

Tozzi, M. (2004). Vertedouros em degraus. *Da Vinci*, *1*(1), 9–27.