Improved MPS method and its variations for simulating incompressible fluids on GPU

André Luiz Vieira-e-Silva, Caio Brito, Mozart William Almeida and Veronica Teichrieb
Voxar Labs, Informatics Center
Federal University of Pernambuco
Recife, Brazil
Emails: {albvs, cjsb, mwsa, vt}@cin.ufpe.br

***Abstract*—Meshless methods to simulate fluid flows have been increasingly evolving through the years since they are a great alternative to deal with large deformations, which is where mesh-based methods fail to perform efficiently. A well known meshless method is the Moving Particle Semi-implicit (MPS) method, which was designed to simulate free-surface truly incompressible fluid flows. Many variations and refinements of the method's accuracy and precision have been proposed through the years and, in this paper, a reasonably wide literature review was performed together with their theoretical and mathematical explanations. Due to these works, it has proved to be very useful in a wide range of naval and mechanical engineering problems. However, one of its drawbacks is a high computational load and some quite time-consuming functions, which prevents it to be more used in Computer Graphics and Virtual Reality applications. Graphics Processing Units (GPU) provide unprecedented capabilities for scientific computations. To promote the GPU-acceleration, the solution of the Poisson Pressure equation was brought into focus. This work benefits from some of the techniques presented in the related work and also from the CUDA language in order to get a stable, accurate and GPU-accelerated MPS-based method, which is this work's main contribution. It is shown that the GPU version of the method developed can perform from, approximately, 6 to 10 times faster with the same reliability as the CPU version, both extended to three dimensions. Lastly, a simulation containing a total of 62,600 particles is fully rendered in 3D.**

I. Introduction

Some of the most common problems in naval hydrodynamics involve the study of fluid flow. For this, it is necessary to deal with large deformations such as those presented in a good portion of computational mechanics problems [1].

Conventional methods, as the Finite Element Methods (FEM), Finite Difference Methods (FDM) and other mesh-based methods, are considered well consolidated and accurate. However, they are relatively inefficient when dealing with certain problems where it is required the simulation of large deformations. The best approach considered to deal with large deformations and the moving discontinuities caused by them is to constantly regenerate the mesh in order to keep its discontinuities coincident through the simulation [2].

Clearly, this constant remeshing makes the process quite expensive in terms of computation, probably even causing accuracy degradation [3]. As an attempt to reduce those issues, methods that use meshes and discrete elements, called particles, were proposed. One example is the Particle Finite Elements Method (PFEM) [4]; another alternative, which has presented great potential over the years, are the entirely meshfree methods. They enable, mainly, that free-surface flow can be discretized and solved the Navier-Stokes equations without the need of a grid of any kind, such as in the work of Frey and Alauzet [5], achieving flexibility in situations where the classic methods are too complex. Each particle carries a set of physics quantities and constitutive properties, such as mass, velocity and position, and they are responsible for characterizing the system state and its evolution through time. An interesting advantage of the meshless methods with Lagrangian characteristics, is that it allows an easy tracking of each particle's quantities in any step of the simulation.

Some of the techniques fully free of meshes are the Moving Particle Semi-implicit method (MPS) and the well-known Smoothed Particle Hydrodynamics (SPH). The SPH was designed in the 1970s by Lucy [6] and Gingold and Monaghan [7] and intended to astrophysics applications. The first method mentioned, the MPS, was introduced in 1996 with the work of Koshizuka and Oka [8] and it was idealized to simulate the flows of incompressible fluids, which refers to a fluid that its material density is constant within a fluid parcel. In many scenarios the changes in temperature and pressure are so small that the density fluctuation is negligible; in such cases the flow may be modeled as incompressible. Its main difference from the original SPH method, which is considered a notable advantage for the MPS method, is that the calculations adopt a semi-implicit predictor-corrector model (which later has been similarly used in some incompressible SPH methods [9]). However, the SPH has been preferred in Computer Graphics (CG) and Virtual Reality (VR) applications [10] [11] due the high computational load occasioned by the precise MPS calculations, including solving the Poisson Pressure Equation (PPE), in spite of starting efforts to change this [12]–[14].

The MPS method was chosen in this work to be studied and implemented due to its appealing intrinsic incompressibility since this type of fluid flow presents environmental importance [15] and appears in many industrial applications [16]. Gotoh and Khayyer [17] presents the current applications and future perspectives of the MPS method and its variations in addition to the incompressible SPH in ocean engineering common problems. Regarding the MPS method, various scenarios are considered, such as wave breaking [18] [19], wave overtopping [20], wave impact [21] [22], green water on ships [23],

sediment transport [24], landslide-generated waves [25] and fluid-structure interactions [26] [27]. This shows how the MPS can be explored in order to help mitigate environmental and natural disasters involving water and/or other liquids.

The main issues of meshfree methods, in general, are in the modeling of solid boundary interaction, fluid flow and, in the specific case of the MPS method, spurious pressure oscillation of the particles [21]. Therefore, several solutions have been proposed in the literature, such as local particle refinement and corrected formulations [28]–[31].

A lot of improvements and adaptations of the original method of both SPH and MPS techniques have been proposed in order to adequate them to the simulation of various kinds of physical phenomena or, more commonly, to get better stability and accurate calculations. In the methodology section, the improvements used in this work are addressed. A set of modifications to the MPS and the SPH method can be seen in the works of Vieira-e-Silva et al. [32] and Almeida et al. [33].

A. Goals

A significant disadvantage of fluid simulation models that value numerical precision is time spent in the application execution, more specifically in the simulation generation [34]. The challenge of dealing with this problem has been diminished through the use of computational platforms that provide Application Programming Interfaces (APIs) making it possible to benefit from the various processing cores of a Graphics Processing Unit (GPU). Some works provide various kinds of performance speedups but always focusing on the standard MPS method [13], [34], [35].

In our work, a significant literature review is made, MPS applications are visited, and the method proposed by Gotoh [36] (CMPS-HS-HL-ECS), is implemented. Initially the implementation is done sequentially to run in CPU and later, using CUDA [37], the same code is implemented to run in GPU, exploiting its parallelism and aiming at near iterative rates for at least relatively low particle number cases (order of 10^3 particles). Lastly, the simulation is rendered using a screen space approach for illustration purposes.

This points in a direction where not only truly incompressible but also quite accurate fluid simulation methods can be used in VR and CG applications, allowing even more realism, now focusing on the physics instead of the rendering.

B. Contributions

A substantial survey in the literature is fulfilled related to the MPS method where a set of papers were read and analysed with the purpose of understand and expose the strong and weak points of the MPS technique, what the community has been proposing in relation to this technique and for which purposes this method can be applied. Mathematical formulations and theoretical explanations of MPS variations, which were not necessarily implemented in this work, are also presented.

Based on the work of Gotoh [36], a stable free-surface incompressible fluid simulation method, was implemented to run in CPU and, subsequently, an unprecedented parallelized version of this method was developed to run in GPU, through the use of the CUDA with speedups ranging from 6.41 to 10.69 times. Then, the method was extended to three-dimensions in both versions, so the whole scene can be fully simulated in 3D.

At last, a simulation of a 3D dam break scenario was generated from the developed method with $62,600$ particles and then coupled with a rendered solution with the purpose of displaying the method's potential in CG and VR.

C. Outline

In the second section of this paper there is a background context on the area presenting the state of the art and related works. Next, there is a section explaining the technique and presenting a set of variations, improvements and applications of it, in which a subset of them has been implemented. A brief explanation on the CPU and GPU versions implementations of the code is presented. Afterwards, a case study is showcased, showing the scenario utilized and its variations, the most lasting parts (absolute and relative durations) of both versions, memory usage, as well as the speedups provided by the GPU version and the rate of generated frames. Lastly, there are the conclusions, final remarks and future works suggestions.

II. State of the Art

Through the years, disasters involving natural phenomena have triggered several researches in many different areas on how to avoid them. Fluid simulation focusing on liquids is one of these areas. To simulate liquids correctly, the fluid flow should be incompressible or weakly compressible, which guarantees that the fluid density fluctuations are kept to a minimum. One of the reasons of simulating liquids in general has been on how to solve these kind of problems. The MPS method has also been providing great assistance in that field, since it was intentionally created for simulating incompressible flows. The work of Chen et al. is a great reference of this area [38].

A. MPS and Its Variations

The MPS method and variations of it has already been used for various purposes and in various fields, such as nuclear engineering phenomena applied to molten core solidification behavior in nuclear power plant accidents and others [39]–[41]. Another example is chemical engineering phenomena applied to eutectic reactions, as well as multiphase fluid simulation [42], [43].

As already has been stated, the MPS method was introduced focusing on the modeling of the behavior of incompressible fluids [8]. Other subsequent works apply the method to certain areas of research, such as coastal and mechanical engineering, among others. In 1998, Koshizuka et al. [44] applied the method to wave breaking in a beach. The authors, in this same work presented an optimization in the neighborhood calculation (from $O(n^2)$ to $O(n^{1.5})$). The previous way (naive) provoked a higher computational load, since the algorithm required that each particle position had to be checked with

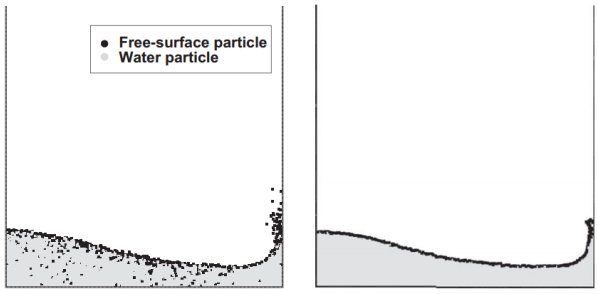
all the others in the system in order to know which ones were its neighbors. Unfortunately, similarly to the other meshfree methods, the MPS technique suffers from instability problems. Some of these issues are related to numerical errors at the boundaries, i.e., at free-surfaces or when interacting with solid boundaries. There are works that describe why those instability problems arise from the MPS method [22], [45].

In attempts to overcome these issues, some authors changed the method in order to improve it; one of the biggest issues with the MPS method is the spurious pressure oscillation. Various works already tried successfully to diminish this. In the work of Kondo et al. [45] an artificial pressure is adopted to stop gradual density change (one of the conditions to express incompressibility). The stabilization process consists in eliminate negative pressure after solving the PPE, setting the negative pressures to zero. This problem is due to the particle number densities near the surface being small, which causes the particles' pressure to be negative, thus causing instability in the system. With the scheme proposed, the authors claimed to obtain smoother pressure variations using a dam break test case for the analysis.

Ataie-Ashtiani and Farhadi [46] used a meshless numerical approach to solve Eulers equation, which is the governing equation of the irrotational flow of ideal fluids. Since the time integration of the equations of inviscid flow (mass and momentum conservation) presents difficulties when dealing with incompressible, or nearly incompressible fluids, a fractional step method, which consists of splitting each time step in two, was proposed in order to facilitate solving the inviscid flow equations. Regarding the MPS method stability, various kernel functions were considered and applied to the method, and, as a result of this study, the most suitable kernel function was employed so that the method could increase its stability. The authors concluded that the developed method is quite useful for solving problems with irregular free-surface in hydraulic and coastal engineering when an accurate prediction of free water surface is required.

Lee et al. [22] stated that the MPS method, when it was initially proposed, had several defects including non-optimal source term of the Poisson Pressure equation (PPE), gradient and collision models, and search of free-surface particles, which led to less-accurate fluid motions. In that sense, the authors proposed step-by-step improvements in the processes referred above, originating what they called the PNU-MPS method. After analyzing the improvements using the dam break problem (shown in Figure 1) and the problem of liquid sloshing inside a rectangular tank, the authors concluded that the numerical results for violent free-surface motions and impact pressures are in good agreement with their respective experimental data.

Duan and Chen [47] discussed the effects of setting up time step and space step on the stability and accuracy of the viscosity term in the MPS method, which is noted to be a very important property of fluids but not quite easy to simulate. In that work, using the MPS method, two conditions for the setup of time step and initial particle distance in a viscous shear



(a) Original MP (b) PNU MPS

Fig. 1: Identification of free-surface particles [22]

flow simulation method are prescribed to be used specially for simulation flows where viscous forces are dominant. The authors concluded that the stability condition of the viscous term can provide a stable simulation. As for the accuracy condition of the viscous term, it is capable of producing the most accurate simulation for steady laminar flow, and can also provide a realistic and accurate simulation of the molecular viscosity term for unsteady turbulent flow at the expense of a high computational cost though.

A set of papers by Khayyer and Gotoh presents valuable insights and improvements to this problem. Most of them proposed corrected differential operator models (laplacian and gradient). In one of their first attempts they proposed a Corrected MPS (CMPS) method [28] for the accurate tracking of water surface in breaking waves. Modifications and corrections in gradient operator model used in the standard MPS method are made with the goal to achieve momentum conservation in the calculations of viscous incompressible free-surface flow.

Then, in 2009, Khayyer and Gotoh [21] proposed new modifications to the MPS method in order to diminish spurious pressure fluctuation. The authors introduced a new formulation of the source term of the PPE, which was referred as a Higher order Source term (HS), thus creating the CMPS-HS method after combining this modification with their previous work. Another modification was allowing slight compressibility to the method, that being, adding part of an equation of state (EOS) to the right hand side of the PPE. The compressible term in the equation would have a stabilizing effect on the particle's pressure calculation. It was shown that the proposed methods are applicable for an approximate estimation of wave impact pressure on a coastal structure.

In 2010, Khayyer and Gotoh [29] focused on the Laplacian model used in the MPS method. They noticed that to further refine and stabilize the pressure calculation, a Higher order Laplacian model (HL) for discretization of the Laplacian operator should be derived. This model was applied in both Laplacian of pressure and the one corresponding to the viscous forces. By merging this new model with previous modifications proposed by the same authors, the CMPS-HS-HL was originated. The authors remarked that, although the improvements enhanced pressure calculations, the numerical results still presented some unphysical numerical oscillation

during tests.

After that, in 2011, following the conclusion in their previous work, Khayyer and Gotoh [30] presented two new modifications in order to resolve the shortcomings that were present in the method proposed in their previous work. The first improvement deals with unphysical numerical oscillation caused by the source term in the PPE, so, extra terms were added to it, which are referred by the authors as Error Compensating parts in the Source term of the PPE (ECS) and, by combining with previous works, the CMPS-HS-HL-ECS was conceived. The second change is meant to deal with situations with tensile instability [48]. It consists of a corrective matrix inserted in the pressure gradient calculations to achieve a more accurate approximation of the differential operator in question.

It is noteworthy that a big portion of these variations are going to be detailed, exposing its calculations and nomenclatures, more specifically the ones by Khayyer and Gotoh since they were used in this work, further in the section about the MPS technique (subsection III-B).

B. Related Works

Since the MPS is fully meshless, the particles are not connected explicitly by any edge, therefore, it is possible to optimize some computational aspects of the simulation, such as by parallelization, by cluster technology or General Purpose GPU (GPGPU) techniques.

Tsukamoto [49] used shared memory parallelization as a way to accelerate the MPS method. His goal was to simulate floating bodies in highly nonlinear waves and he achieved significant performance gains compared with the sequential version of the simulation.

Ikari and Gotoh [50] compared two problem decomposition methods, one based on particles decomposition and the other on a domain decomposition. They verified that domain decomposition, in most cases, presents a smaller runtime to finish the calculations.

Gotoh [51] developed a MPS version to be executed in parallel, combining domain decomposition techniques with dynamic boundaries, periodically recalculating based on the center of mass of each subdomain to enhance load balancing in the processors and also a process of preconditioner matrix restructuring for accomplishing the forward/backward process of the Conjugate Gradient in parallel. The authors concluded that the proposed method successfully simulated the studied models, but the parallelized model still needed further refinement, that being in precision and computational efficiency. This could be achieved through the development of more accurate and consistent numerical models of differential operators, such as time integration.

Iribe et al. [52] presented simulation results of the parallelized MPS for a PC cluster. The authors identified that the bottleneck of the iterative solver parallelization in shared memory is the computational cost of the communication between subdomains. To minimize this communication, a sophisticated particle renumbering process based in packages and in a communication list was used. With these techniques, they were able to accelerate the communication process. A 237-hour simulation of a tsunami, pictured in Figure 2, with six million particles was generated. The authors concluded that the reordering process proposed can be used to elaborate an efficient scheme of unidimensional decomposition process.

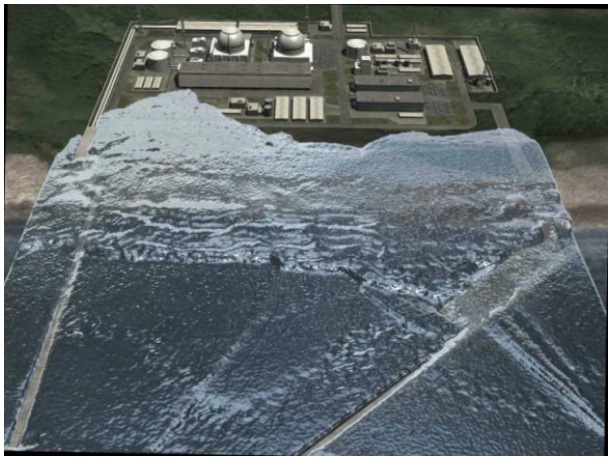


Fig. 2: Tsunami simulation [52]

Hori et al. [13] developed a GPU-accelerated version of a MPS code using NVIDIA's CUDA. The authors focused on the search of neighboring particles and the iterative solution of the linear system generated by the PPE, which generates a large computational. The optimization of the search for neighboring particles is achieved through a cell grid, in which each particle is stored in a specific cell according to the particle's position. In order to compare accuracy and performance between the CPU and GPU-based codes, 2-dimensional calculations of an elliptical drop evolution and a dam break flow have been carried out. Finally, the reported speedup achieved in that work is about 3 to 7 times.

Zhu et al. [34] developed a GPU-based MPS model using CUDA. To find the neighbors for a specific particle i a similar approach to Hori et al. [13] was used, where background grids are employed in order to reduce significantly memory access, taking only $O(kNP)$ times. The authors built four different test cases to evaluate the GPU program optimization, all based in the dam break scenario. To solve the PPE, the Bi-Conjugate Gradient method (BiCG) is used and it is shown that the percentage of time used for solving the pressure equation decreases from 66% to 40% as the total number of particles raises. The authors concluded through a numerical analysis that the models based on CPU and GPU have the same precision and, through a performance comparison, a 26 times speedup can be obtained with the MPS-GPU in contrast to the MPS-CPU.

In Fernandes's PhD thesis [53], he developed a computational framework of hybrid parallelization of the MPS method. An altered version of the pressure formulation was used, where the stability is higher at the cost of introducing a limited compressibility to the method, something unacceptable in systems

with rigorous incompressibility requirements. He concluded that his work contributed to the MPS method consolidation as a practical tool to investigate complex engineering problems, since the method has its applicability extended to scenarios with millions of particles, and could be used, for instance, in the influence of ship movements in waves, phenomena involving fragmentation and dealing with large deformations.

Differently from the works here presented, our work's contribution lies in the parallelization to achieve relevant speedups with a GPU execution of an enhanced MPS method with significant changes from the basic technique ensuring a stabler and more accurate method. Equally important, the characteristics of the PPE are not changed, therefore, maintaining its full incompressible property.

III. The Moving Particle Semi-implicit Method

Here, an explanation of the MPS method is done, showing its governing equations, discretized differential operations, as the set of the used improvements to the basic MPS and a few illustrations of the MPS algorithm and calculations.

A. Standard Method & Governing Equations

This method models the fluid as an assembly of interacting particles, in which their motion is determined through the interaction with neighboring particles and according to the governing equations of fluid motion. To describe the motion of a viscous fluid flow, there is the continuity equation and Navier-Stokes equation as follows in Equation 1 and Equation 2, respectively.

$$\frac{1}{\rho}\frac{D\rho}{Dt} + \nabla.\mathbf{u} = 0 \tag{1}$$

$$\frac{D\mathbf{u}}{Dt} = -\frac{1}{\rho}\nabla p + \mathbf{g} + \nu\nabla^2\mathbf{u} \tag{2}$$

where $\mathbf{u}$ is the fluid velocity vector, t is the time, ρ is the fluid density, p is the pressure, $\mathbf{g}$ is the gravitational acceleration vector and ν is the laminar kinematic viscosity. To adapt these equations so that a fluid can be represented by discrete elements, some of these physical quantities become particles attributes; so $\mathbf{u}$ becomes the velocity vector of a particle, ρ now stands for the density of the particle and p, the pressure of a particle. The left hand side of the continuity equation (Equation 1) is represented, in the case of incompressible flow, by a simple volume continuity equation, as presented in Equation 3 [36]:

$$\nabla.\mathbf{u} = 0 \tag{3}$$

A particle interacts with its neighbors through a kernel function $w(r)$, r being the distance between two particles. The most common form of kernel function employed in MPS, and used for the implementation in this work, is in Equation 4:

$$w\left(|\mathbf{r}_j - \mathbf{r}_i|\right) = \begin{cases} \frac{r_e}{|\mathbf{r}_j - \mathbf{r}_i|} - 1 & , \quad 0 \leq r < r_e \\ 0, & r_e \leq r \end{cases} \tag{4}$$

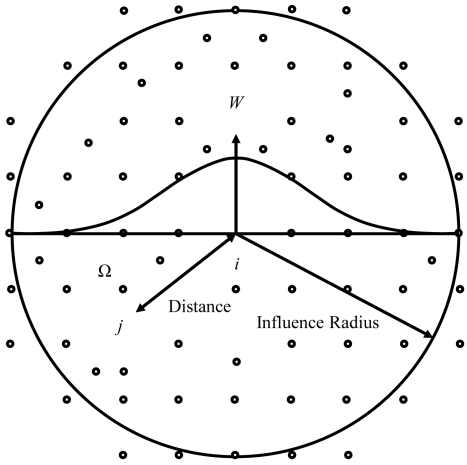


Fig. 3: Influence radius of a particle in a two-dimensional problem

where r_e is the radius of the interaction area and $\mathbf{r}_i$ and $\mathbf{r}_j$ are the positions of particles i and j, respectively. Clearly, a larger kernel size implies in an interaction with more particles, as seen in Figure 3.

There are also other types of kernel functions and a detailed analysis on the subject can be seen in the work of Ataie-Ashtiani [46].

To find all the neighboring particles j of each particle i the all-pair search algorithm is used. In this algorithm, the distance between each particle of the simulation is checked to see whether they are in the target's radius of influence and thus determine its neighbors.

The particle number density n at the particle's position $\mathbf{r}_i$, which is proportional to the neighbors number of i, is defined in Equation 5.

$$n_i = \sum_{j \neq i} \left(|\mathbf{r}_j - \mathbf{r}_i|\right) \tag{5}$$

The continuity equation is satisfied if the particle number density remains constant, and this constant value is denoted by n_0. As stated before, in the original MPS method the derivative of a kernel is not calculated, instead, the gradient or Laplacian are obtained by local weighted averaging of these operators calculated between a pair of particle i and a neighbor, particle j.

The gradient model formulation used in MPS of a physical quantity φ is shown in Equation 6.

$$\nabla\varphi_i = \frac{D_S}{n_0}\sum_{j \neq i}\frac{\left(\varphi_j - \varphi_i\right)}{\left|\mathbf{r}_j - \mathbf{r}_i\right|^2}\left(\mathbf{r}_j - \mathbf{r}_i\right) w\left(|\mathbf{r}_j - \mathbf{r}_i|\right) \tag{6}$$

In Equation 6 and Equation 7, D_S is the number of space dimensions present in the simulation. This model is ultimately applied to the pressure gradient term. The Laplacian of φ, applied to the pressure and in the viscous stress calculation in this method, is discretized as shown in Equation 7.

$$\nabla^2 \varphi_i = \frac{2D_S}{n_0 \lambda} \sum_{j \neq i} \left(\varphi_j - \varphi_i \right) w \left(|\mathbf{r}_j - \mathbf{r}_i| \right) \tag{7}$$

where λ is the weighted average of the squared distance between particles i and j (or r_{ij}^2), as can be seen in [8].

The satisfaction of the continuity equation is indispensable to model incompressibility, so, the fluid density must remain constant. When the particle number density n^* calculated in an intermediate step is not equal to n_0, it is implicitly adjusted to n_0.

Differently from many SPH-based calculations where the equations are solved explicitly, the pressure in MPS is implicitly calculated by solving a PPE. The other terms are approximated explicitly, thus giving the name of the method. To solve the PPE it is necessary a two step prediction-correction process. In the first step there is the explicit integration in time, while, in the second step, the implicit computation of a divergence-free velocity field occurs. The calculation of the intermediate velocity field $\mathbf{u}^*$ is derived from the implicit pressure gradient term as:

$$\mathbf{u}_i^* = \mathbf{u}_i^k + \frac{\Delta t}{\rho_i^*} \nabla p^{k+1} \tag{8}$$

where k indicates the current time step in the simulation, ρ_i^* is the density calculated at time step k of the particle i and p indicates the particle pressure. The velocity and particle densities in Equation 8 satisfy the mass conservation law as in Equation 9.

$$\frac{1}{\rho} \frac{D\rho}{Dt} + \nabla . \left(\mathbf{u}_i^{k+1} - \mathbf{u}_i^* \right) = 0 \tag{9}$$

By representing the derivative of the ρ as $\frac{\rho_0 - \rho_k^*}{\Delta t}$ and substituting ρ for n, it is possible to deduce the PPE [8]:

$$\nabla^2 p_i^{k+1} = -\frac{\rho}{\Delta t^2} \frac{n_i^* - n_0}{n_0} \tag{10}$$

The Incomplete Cholesky Conjugate Gradient (ICCG) method is usually employed to solve the linear system [8], [21]. By solving the PPE, the velocity in time step $k+1$ $(\mathbf{u}^{k+1})$ can be calculated, and, at last, the particle positions, denoted by r in Equation 11, are updated through a simple first-order Euler integration.

$$\mathbf{r}_i^{k+1} = \mathbf{r}_i^k + \mathbf{u}_i^{k+1} \Delta t \tag{11}$$

The solid boundaries in standard MPS, as walls and fixed obstacles, are represented by fixed particles with no velocity. Some of these particles, however, are considered to solve the PPE. To tell which will be used for the pressure calculations, it is important to explain that there are two layers of wall particles. One of these layers will be referred as inner wall particles (those that, initially, come into direct contact with the fluid particles) and the other as dummy particles (which complement the solid boundary). Usually, just some lines (often two) of dummy particles are used [44]. A model can be seen in Figure 4. The PPE is solved by taking into account the inner wall particles only to repel the fluid from the solid boundaries, while the dummy particles were introduced so that the particle number density at the inner wall particles is not small and that they are not recognized as free-surface.

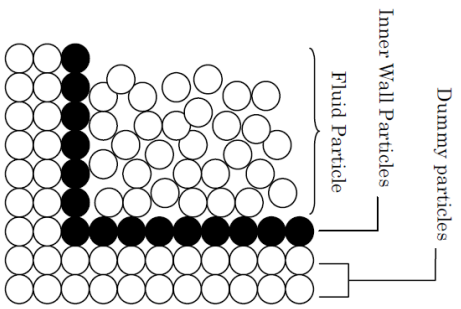


Fig. 4: Dummy boundary scheme

To identify a free-surface particle, the particle number density of the ith particle just needs to satisfy the condition presented in Equation 12 since on the free-surface the particle number density drops abruptly.

$$n_i < \beta n_0 \tag{12}$$

The bigger β is, the bigger will be the number of particles recognized as free-surface. Koshizuka and Oka [8] recommends that it should be set to 0.97. An overview of the MPS algorithm can be seen in Figure 5.

B. MPS Enhancements

In this section, improvements of the standard MPS that were implemented in this work are described. It is noteworthy that the universe of variations is much larger and the ones that were selected stand between the improvement impact size and implementation cost until, finally, a version that was considered sufficiently stable and physically accurate was achieved. It is also shown other modifications to the standard method which expand the range of applications of the MPS method.

1) Kernel Functions: As discussed before, the motion of each particle depends on the interaction with its neighbors, and this relation is ruled by the kernel function. So, along the years, various kernel functions were suggested for the purpose of achieving better performance and numerical accuracy in the simulation. In [46] six kernel functions previously proposed are considered and applied to study the simulation behavior and computational performance in order to reveal which one enhances numerical stability best. The kernels considered in this work are presented in Table I.

This study shows that the kernel function proposed by [15] was found to improve the most the stability of the MPS method, in way that the collapse of a water column simulation was successful, including the loss of the water momentum to the point where it stands still inside the container, a significant feature amongst other particle methods.

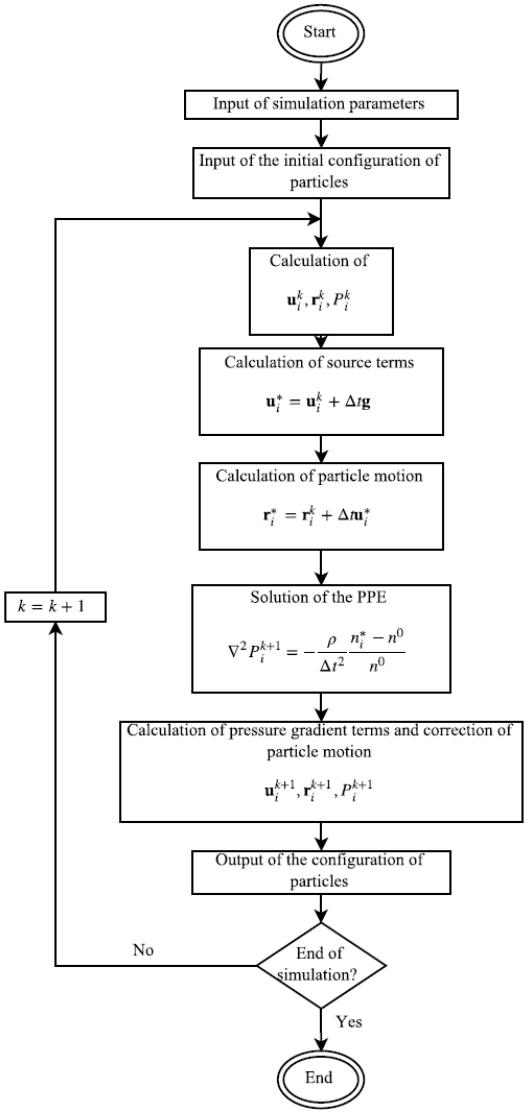


Fig. 5: Algorithm of MPS method

2) Momentum Conservation: In the work of Suzuki et al. [56] have developed the Hamiltonian MPS (HMPS) in which the momentum and mechanical energy of the system are preserved. However, HMPS carries heavy theory to its calculations making it extremely complicated to implement in comparison to the standard MPS method. A simple way to achieve a consistent conservation of linear momentum is to ensure a better discretization of the gradient model, which is directly connected to the linear momentum conservation.

Equation 13 shows the suggested alteration on the pressure gradient formulation by Khayyer and Gotoh [28].

TABLE I: Different kernel functions

Kernel function formulation	Work
$w(r) = \begin{cases} e^{-\left(\frac{r}{r_e}\right)^2}, & 0 \le r \le r_e \\ 0, & r_e < r \end{cases}$	[54]
$w(r) = \begin{cases} \frac{2}{3} - 4\left(\frac{r}{r_e}\right)^2 + 4\left(\frac{r}{r_e}\right)^3, & 0 \le r \le \frac{r_e}{2} \\ \frac{4}{3} - 4\left(\frac{r}{r_e}\right) + 4\left(\frac{r}{r_e}\right)^2 + \frac{4}{3}\left(\frac{r}{r_e}\right)^3, & \frac{r_e}{2} < r \le r_e \\ 0, & r_e < r \end{cases}$	[54]
$w(r) = \begin{cases} 1 - 6\left(\frac{r}{r_e}\right)^2 + 8\left(\frac{r}{r_e}\right)^3 - \frac{4}{3}\left(\frac{r}{r_e}\right)^4, & 0 \le r \le r_e \\ 0, & r_e < r \end{cases}$	[54]
$w(r) = \begin{cases} -2\left(\frac{r}{r_e}\right)^2 + 2, & 0 \le \frac{r}{r_e} < \frac{1}{2} \\ \left(2\frac{r}{r_e} - 2\right)^2, & \frac{1}{2} \le \frac{r}{r_e} < 1 \\ 0, & r_e \le r \end{cases}$	[8]
$w(r) = \begin{cases} \frac{r_e}{r} - 1, & 0 \le r < r_e \\ 0, & r_e \le r \end{cases}$	[44]
$w(r) = \begin{cases} \frac{40}{7\pi r_e^2}\left(1 - 6\left(\frac{r}{r_e}\right)^2 + 6\left(\frac{r}{r_e}\right)^3\right), & 0 \le r < \frac{r_e}{2} \\ \frac{10}{7\pi r_e^2}\left(2 - 2\frac{r}{r}\right)^3, & \frac{r_e}{2} < r < r_e \\ 0, & r > r_e \end{cases}$	[55]

$$\nabla p_i = \frac{D_S}{n_0}\left(\sum_{j \neq i} \frac{(p_i + p_j) - (\hat{p}_i + \hat{p}_j)}{|\mathbf{r}_j - \mathbf{r}_i|^2} (\mathbf{r}_j - \mathbf{r}_i)\, w\left(|\mathbf{r}_j - \mathbf{r}_i|\right)\right) \quad (13)$$

$$\hat{p}_i = min_{j \in J}(p_i, p_j), J = \{j : w(|\mathbf{r}_j - \mathbf{r}_i|) \neq 0\} \quad (14)$$

When the anti-symmetric Equation 13 is applied, linear momentum is exactly conserved. This method is referred by the authors as Corrected MPS (CMPS).

3) Pressure Calculation: One of the major issues of the MPS method, and consequently widely explored, is the spurious pressure oscillation. Recent works that presented substantial improvements in this area, making few and simple modifications to the method, have been proposed [21], [29]. The first one is called by the authors as the MPS method with a Higher order Source term (MPS-HS), since it basically presents a new formulation for the calculation of the derivative of the particle number density ($\frac{Dn}{Dt}$). Using this method, the Equation 10 is replaced by the Equation 15 [31].

$$\nabla^2 p_i^{k+1} = -\frac{\rho}{n_0 \Delta t}\left(\sum_{i \neq j} \frac{r_e}{r_{ij}^3}(x_{ij}u_{ij} + y_{ij}v_{ij} + z_{ij}w_{ij})\right)^* \quad (15)$$

where r_{ij} is the distance between particles i and j. x_{ij}, y_{ij} and z_{ij} represent the distance between particles i and j in each dimension and u_{ij}, v_{ij} and w_{ij} the velocity difference

of particles i and j in each dimension. It is important to note that all the enhancements shown so far can be, and most of them normally are (specially when they are suggested by the same authors), combined in one single method to produce a more robust outcome.

The other improvement to the method's pressure calculation implemented was the proposition of a higher order Laplacian model for both two and three (Equation 16) dimensional simulations [29], [31].

$$\nabla^2 \varphi_i = \frac{1}{n_0} \sum_{i \neq j} \left(\frac{2\varphi_{ij} r_e}{r_{ij}} \right) \tag{16}$$

where φ is a generic physical quantity. This new derivation was named by the authors as MPS with a Higher order Laplacian of pressure (MPS-HL).

4) Numerical Stability: Khayyer and Gotoh [30] came up with a PPE's source term with error-compensating parts to enhance even further pressure and velocity field calculations. The error-compensating terms should be measures for instantaneous and accumulative violations of fluid incompressibility. Equation 17 shows the suggested terms to be added to the source term of the PPE and Equation 18 shows the complete modified PPE.

$$\begin{aligned} ECS = & \left| \left(\frac{n^k - n_0}{n_0} \right) \right| \left[\frac{1}{n_0} \left(\frac{Dn}{Dt} \right)_i^k \right] \\ & + \left| \left(\frac{\Delta t}{n_0} \left(\frac{Dn}{Dt} \right)_i^k \right) \right| \left[\frac{1}{\Delta t} \frac{n^k - n_0}{n_0} \right] \end{aligned} \tag{17}$$

$$\nabla^2 p_i^{k+1} = \frac{\rho}{n_0 \Delta t} \left(\frac{Dn}{Dt} \right)_i^* + ECS \tag{18}$$

The combination of the refinements shown so far gives as outcome the Corrected MPS with a Higher order Source term - Higher order Laplacian of pressure - Error Compensating parts in the Source term (CMPS-HS-HL-ECS) method. According to Gotoh [36], the said method ensures satisfactory accuracy and stable computation, more specifically, under the absence of tensile stress. A comparison between CMPS-HS-HL-ECS and the standard MPS can be seen in [36] which presents a standard test case found in the literature, the breaking waves. This comparison is abridged in Figure 6, where different shades of gray represent different pressure levels.

5) Weak Compressibility: Although a weakly compressible model was not implemented in this work, it is noteworthy, since it can be coded relatively quick and integrated to the MPS method in order to severely diminish computational load in exchange of some numerical precision. In the work of [57], the incompressible model is replaced by a weakly compressible one on the grounds that assembling and solving the PPE in each step takes a considerable amount of computation time: About two thirds of the computational time in each step for a case where the number of particles in the simulation is in the order of 10^3. In the mentioned work, the PPE is replaced

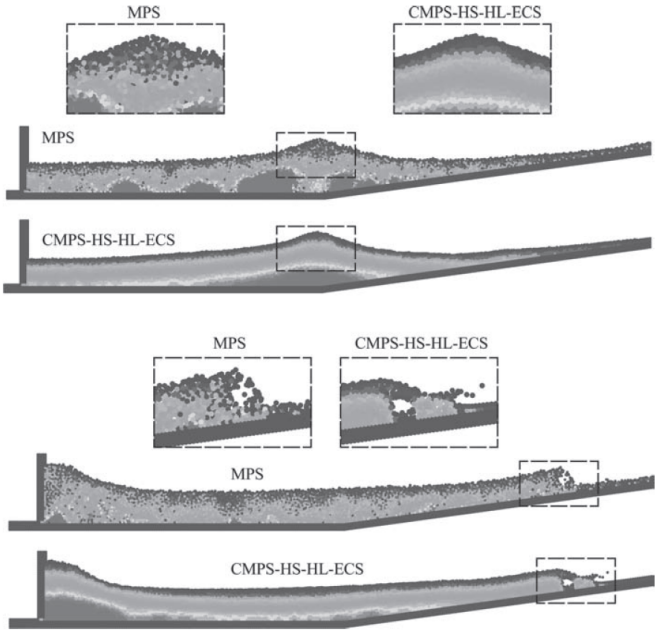


Fig. 6: Comparison between standard MPS and CMPS-HS-HL-ECS through breaking waves test case [36]

by an explicit relation, more specifically an equation of state described by [58] and modified by [59] that is shown below.

$$p_i^{k+1} = \frac{\rho c_0^2}{\gamma} \left(\left(\frac{n_i^*}{n_0} \right)^\gamma - 1 \right) \tag{19}$$

The typical value used for $\gamma = 7$. c_0 is the speed of sound in the reference density. By keeping a small compressibility value, the fluid is treated as a weakly incompressible fluid. This study in fact shows a decrease in process time per time step while the simulation remains similar to the fully incompressible method. Authors refer to this modified MPS as WC-MPS. The work of [21] proposes another compressible form for the PPE, which is presented in Equation 20:

$$\nabla^2 p_i^{k+1} = -\frac{1}{\Delta t^2 c_0^2} \left(p_i^{k+1} - p_i^k \right) + \frac{\rho}{\Delta t} \left(\nabla . \mathbf{u}_i^* \right) \tag{20}$$

where c_0 is another representation of the speed of sound. In this work, the compressible term, which is the first term on the right hand side of Equation 20, works as a stabilizer for the pressure calculation, softening part of the noise caused by the second term on the right hand side in Equation 20, thus resulting in a somewhat lower fluctuation in the pressure field. This modified MPS is usually referred as WC-MPS.

6) Multiphase Flow: A model that significantly increases the number of possible applications for the MPS method is the one that supports multiphase flow. Here, an enhanced stabilized MPS method for simulation of multiphase flows characterized by high density ratios is discussed. This method benefits from previous enhancements also suggested by Khayyer & Gotoh and a new one for accurate, consistent modeling of density at the phase interface. One of the challenging

issues in simulation of multiphase flows characterized by high density ratios, corresponds to the mathematical discontinuity of density at the phase interface. The simplest way, according to [60], to deal with discontinuity is to evaluate the calculated density at a target particle i based on a simple spatial averaging. So, two schemes referred as the Zeroth-order and First-order accurate Density Smoothening schemes, abbreviated as ZDS and FDS, are shown in Equation 21 and Equation 22, respectively.

$$\rho_i = \frac{1}{\sum_{j \in I} W_{ij}} \sum_{j \in I} \rho_j W_{ij} \tag{21}$$

$$\rho_i = \frac{1}{\sum_{j \in I} W_{ij}} \sum_{j \in I} \left(\rho_j - \frac{\partial \rho_i}{\partial x_{ij}} x_{ij} - \frac{\partial \rho_i}{\partial y_{ij}} y_{ij} \right) W_{ij} \tag{22}$$

where I corresponds to target particle i and all its neighboring particles j and W_{ij} represents a different kernel function from the standard MPS kernel called Wendland kernel adopted for all test cases in that study, as can be seen in [61].

IV. Implementation Methodology

In this section the hardware and programming language used are stated, and also there is an overview of the CPU and GPU versions development of the enhanced MPS method, showcasing code listings to exemplify some of their implementation differences.

A. Software and Hardware Infrastructure

The development of the whole system was divided in two parts. In the first part, the MPS technique was implemented in C++ and, subsequently, each one of the improvements shown in the previous section were implemented to run in the CPU. In the second part another version of the code was developed using CUDA to exploit the parallelism provided by the GPU.

The CPU used was a Intel® Core™ i7-4790 CPU @ 3.60 GHz [62] with 7.86 GB of RAM and a 64-bit operating system (x64). The GPU was a NVIDIA GeForce GTX 760 [63] with 2048 MB of RAM.

B. Code Development

It is important to state the reason of using CUDA over OpenCL [64], for example, since it is an open standard and is able to run in most GPUs. CUDA often offers, in average, a higher performance since updates for CUDA happen frequently supporting new features offered by the new NVIDIA GPUs. Also, CUDA allows a higher level of abstraction making the kernel calls and coding in general quite straightforward, providing a smoother learning curve. Besides, there is a great deal of educational material available regarding NVIDIA GPU programming. Finally, the goal of this study is to show how fast the developed method can perform by using a parallel approach without initially concerning with other platforms.

The code is written in a way that the GPU could be much explored, using information like number of CUDA kernels, maximum number of threads within a block and number of blocks within a grid, which varies according to the compute capability of the GPU used.

Regarding the implementations, while the process to parallelize a portion of the functions are quite straightforward, some of them had to be adapted in order to be possible to develop a parallelized version of each of them.

An example of a straightforward function parallelization is the external forces calculation, which in the test case used in this work is only the gravitational acceleration. Listing 1 and Listing 2 illustrate the CPU and GPU implementation differences of this calculation in bold font.

Listing 1: CPU code of the external forces calculation

```
void ParticlesActions::external_force(
    Particle2D* particles, double dt, double g
    , int nump)
{
        Point2D velocity;
        for (int a = 0; a < nump; a++)
        {
                if((particles)[a].is_fluid())
                {
                        velocity = (particles)
                            [a].get_v();
                        (particles)[a].set_v(
                            velocity.x,
                            velocity.y - g*dt)
                            ;
                }
        }
}
```

Listing 2: GPU code of the external forces calculation

```
__global__ void external_force_kernel(int
    offset, Particle2D* particles, double dt,
    double g)
{
        unsigned int a = offset + (blockDim.x
            * blockIdx.x + threadIdx.x);

        Point2D velocity;
        if (particles[a].is_fluid()){
                velocity = (particles)[a].
                    get_v();
                (particles)[a].set_v(velocity.
                    x, velocity.y - g*dt);
        }
}
```

Regarding the PPE, the Cusp library [65], an open-source project by NVIDIA Research based on Thrust [37], was used to solve the linear system directly in the GPU.

After converting the matrix from the regular dense format to the Compressed Sparse Row (CSR) type in order to fully benefit from this solver's capabilities, the information is assigned to the data structure of matrices and arrays provided by the Cusp library so the linear system equations can be solved. The Biconjugate gradient stabilized method (BiCGStab) also

provided by Cusp is used to solve the system as can be seen in Listing 3.

Listing 3: Solving $A * x = B$ with GPU-optimized code

```
(...)

// Assigning coefficient matrix info to csrA
cusp::csr_matrix<int, double, cusp::
    device_memory> csrA;

// Allocating resulting array
cusp::array1d<double, cusp::device_memory> x(
    csrA.num_rows, 0);

// Assigning source term information to array
cusp::array1d<double, cusp::device_memory>
    array1dB(dev_ptr_srcB_d, dev_ptr_srcB_d +
    size_B);

// Setting stop criteria: iteration limit =
    100, relative tolerance = 1e-6, absolute
    tolerance = 0, verbose = false
cusp::monitor<double> monitor(array1dB, 100, 1
    e-16, 0, false);

// Configuring preconditioner (identity)
cusp::identity_operator<double, cusp::
    device_memory> M(csrA.num_rows, csrA.
    num_rows);

// Solving linear system A*x = b
cusp::krylov::bicgstab(csrA, x, array1dB,
    monitor, M);

(...)
```

All the functions developed in the CPU version were successfully ported to CUDA C/C++ to run in the GPU.

V. Case Study

The collapse of a water column has been widely used in the literature to validate and study various fluid simulation methods. Originally, the dam break problem for the MPS method was modeled by Koshizuka and Oka [8].

Although the dam break model used in this work is not equal to Koshizuka and Oka's, it follows a similar approach. Figure 7 shows the dam break model used.

Height H and length L are equal in the tests performed. The size of the recipient's floor in the model employed is four times the length L of the water column. The size of the water column varies depending on how many particles the simulation has. The average particle distance is 1×10^{-2} meters and the time step of the simulation is 1×10^{-3} seconds. The parameter β, shown in Equation 14, is 0.97. The influence radius r_e of a particle should be $< 3.0 l_0$, where l_0 is the average particle distance, otherwise the particles will gather near the free-surface [8]. On the other hand, the discretization of the Laplacian model is more accurate when the influence radius has a higher value. To satisfy this, two different kernel sizes are commonly employed, $r_e = 2.1 l_0$ and $r_{e\ Lap} = 3.1 l_0$.

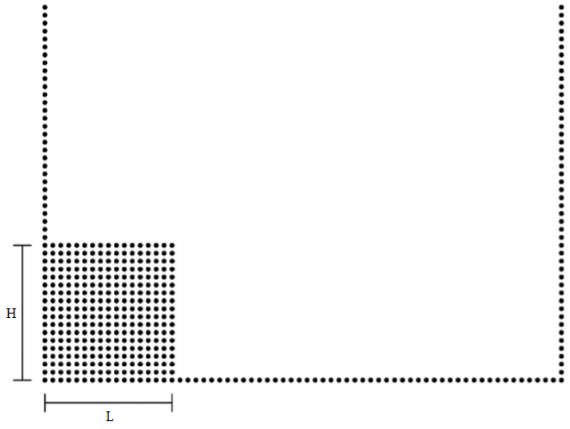


Fig. 7: Dam break model employed

A. Results

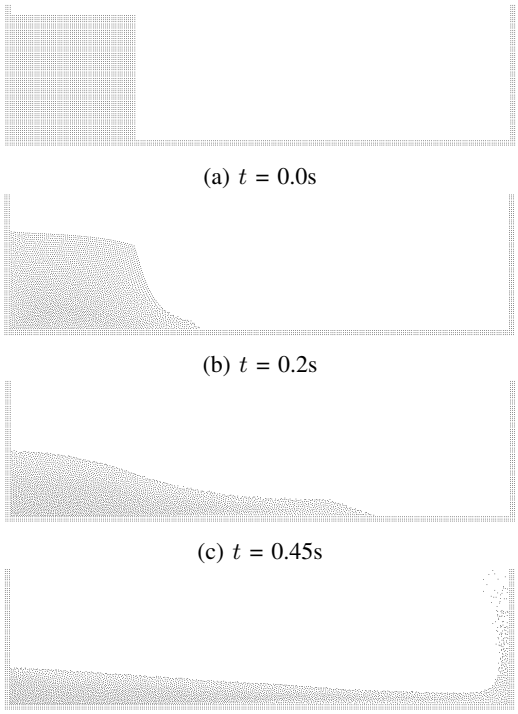
Firstly, the visual results of one of the simulations used as parameter for all the analyses will be shown. After that, a numerical analysis of both codes developed (in the CPU and the GPU) will be made in order to show the absence of accuracy and stability loss in these implementations. Then, a performance analysis will be done. It includes the time spent on each function, memory usage and charts comparing the execution time of the programs as the total particle number increases. Lastly, the speedup of the GPU version over the CPU version will be calculated, analysed and presented graphically, as well as the frame rate of the GPU simulation.

1) Simulation: Here, one of the simulations based on the dam break problem presented previously and used as parameter for the analyses will be exposed in order to show the visual coherence of the simulation. In Figure 8 the length L and height H of the water column is $0.6\ m$. The time of every screenshot taken is presented too.

2) Numerical Analysis: During the development of the GPU version of the method, strict attention was paid to data types used in order to not compromise the numerical precision. This was done so that the precision of the GPU version was maintained (compared to the CPU version). Figure 9 shows the comparison between the methods. For this comparison, the wave front position in relation to the x-axis (its absolute value is represented by x) is being monitored since the beginning of the dam burst and it is represented by a dimensionless format, (x/L), which for this numerical analysis is equal to $0.6\ m$. As said before, the size of the floor is four times the size of L, implying that the maximum value x can reach is four times L. The time in the chart is represented by a dimensionless format as well: $t\sqrt{g/L}$, where t is the time in seconds and g the gravitational acceleration ($9.8\ m/s^2$).

As evidenced in Figure 9, the GPU version of the method behaved in an extremely similar way as the CPU version did, making it clear that the numerical precision from the CPU version was virtually maintained.

3) Performance Analysis: To analyse the performance of the implementations, the percentage of time that was spent



(a) t = 0.0s

(b) t = 0.2s

(c) t = 0.45s

(d) t = 0.8s

Fig. 8: CMPS-HS-HL-ECS method visual results for $L = H = 0.6\ m$

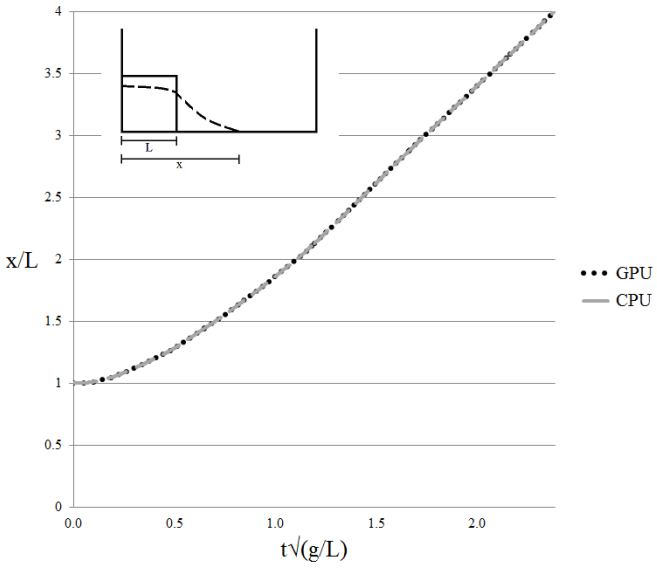


Fig. 9: Evolution of the water wave front through dimensionless time

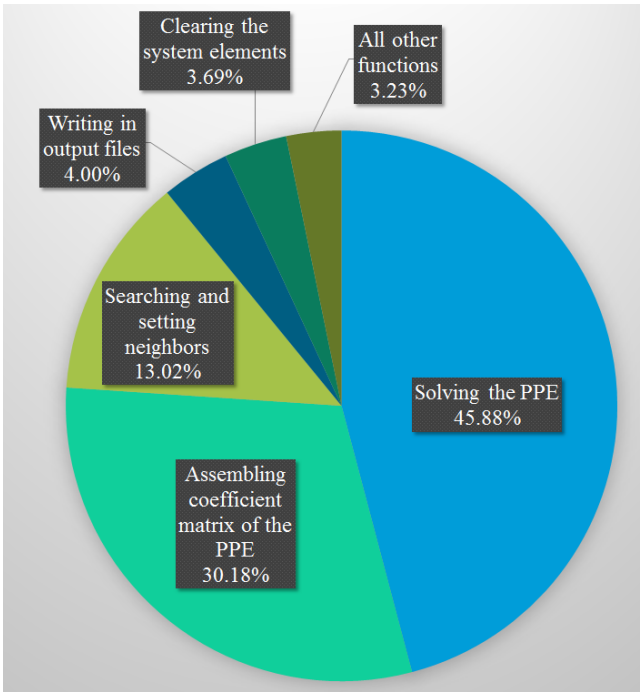


Fig. 10: Functions relative execution time in the CPU

by each function and memory usage in both versions will be exposed. Finally, the absolute time spent by each version is shown in order to calculate the speedup reached by the GPU implementation.

a) Functions Duration & Memory Usage:

The Performance and Diagnostics tool of Microsoft Visual Studio 2013 was used to get the duration of each function of the CPU version as presented in Figure 10. This result was generated by the same simulation scenario presented in subsubsection V-A2.

It is possible to see that the assembly and resolution of the PPE takes about three-quarters of the program's execution time. Also, a good portion of the execution time (13.02%) is due to the search and setting of the particle neighbors. The remaining 10.92% is due to all the other calculations and functions of the code, showing the significance of these three functions, taking almost 90% of time program's execution time in the CPU.

In this sense, special attention was paid to the PPE during the implementation of the GPU version. Through the use of the NVIDIA Visual Profiler, absolute and relative times of each CUDA kernel function were extracted. Figure 11 shows the relative amount of time each kernel spent executing in the GPU.

As it can be seen, the shrinkage of the relative amount of time taken by the PPE's solution is significant, as the gain for this operation was about 38.88%. This result shows the relevance of parallelizing the solution of the PPE's linear system. Assembling the coefficient matrix of the PPE and the neighborhood search and setting are now the main bottleneck

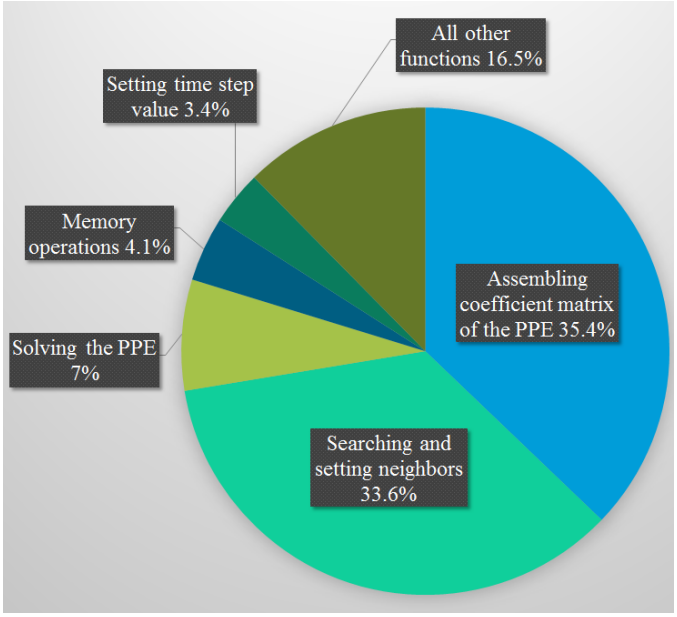


Fig. 11: Functions relative execution time in the GPU

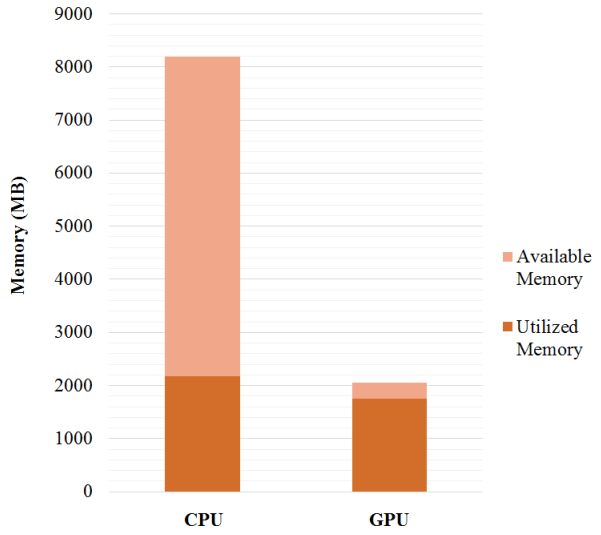


Fig. 12: Memory consumption in both implementations

of the program, however, it is notable the execution time of the functions is better distributed than in the CPU version, showing that the parallelization of the functions that dominated the execution in the CPU are taking less time in their execution, diminishing the bottleneck caused by them. All other functions spent less than 2% of the total execution time in the GPU.

The Performance and Diagnostics tool of Visual Studio was also utilized in order to evaluate memory usage in the CPU version. As for the GPU version, the CUDA function `cudaMemGetInfo` was called before the deallocation of the variables and arrays in the GPU memory. Figure 12 shows the amount of memory in megabytes used in both code versions for a test case with $6,622$ particles.

The memory usage is similar in both versions ($2,170$ MB from the CPU against $1,758$ MB from the GPU) even though for the CPU there is more unused memory than for the GPU execution, showing that the GPU version uses a smaller amount of memory than the CPU.

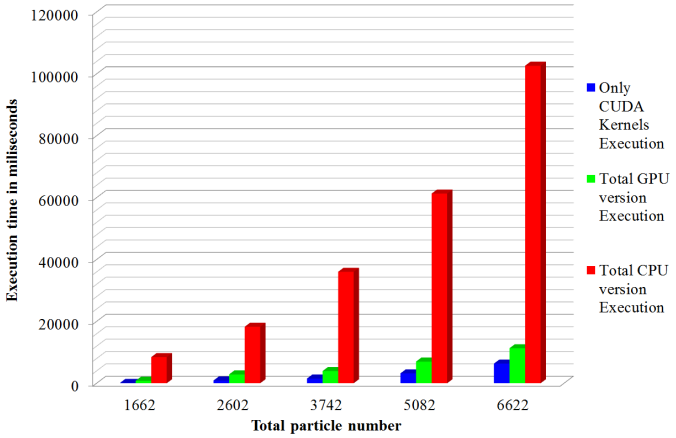


Fig. 13: Execution time versus total particle number in the system

b) Speedups:

For this analysis, various test cases were built based on the dam break problem, only increasing the total particle number, but always keeping the proportions shown in the beginning of section V. By dividing the absolute execution time in the CPU by the GPU's in each test, the speedups provided by the GPU version were obtained.

In each test, the size of the water column was increased in $0.1\ m$ from $0.3\ m$ to $0.7\ m$, such in length as in height (keeping the column and floor proportions). The execution time of each scenario and its total particle number, can be seen side by side in Figure 13.

The CPU execution time clearly exceeds by far the GPU execution time, and exceeds even more when only considering the CUDA kernel execution times, where is not taken into consideration transfer operations between host and device memory, in any direction.

It is now possible to calculate how many times the GPU version of the implementation is faster than the CPU version, in other words, the relative speedup for each test case built. The chart presented in Figure 14 shows the speedup values, depending on the total particle number of the test case. The speedups range from 6.41 to 10.69, and the average speedup in this set of scenarios is a considerable 8.82 times.

Unexpectedly, as the system size increased the speedup did not increase, as expected in a GPU-optimized code. In order to investigate this issue, a comparison of execution times was made, as exhibited in Figure 15. It shows the absolute time, in milliseconds, spent in the most time consuming functions of the execution for each test case. The function represented by '1' is the neighborhood calculation, '2' the coefficient matrix assembly, '3' the PPE solving, '4' the calculation of the time

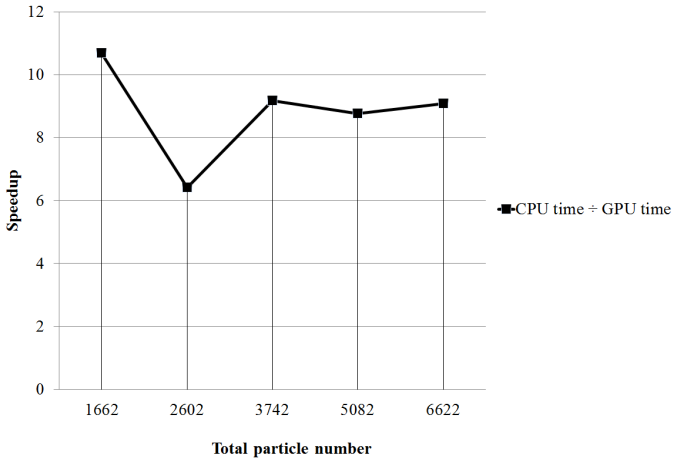


Fig. 14: Speedups of the GPU version over the CPU version

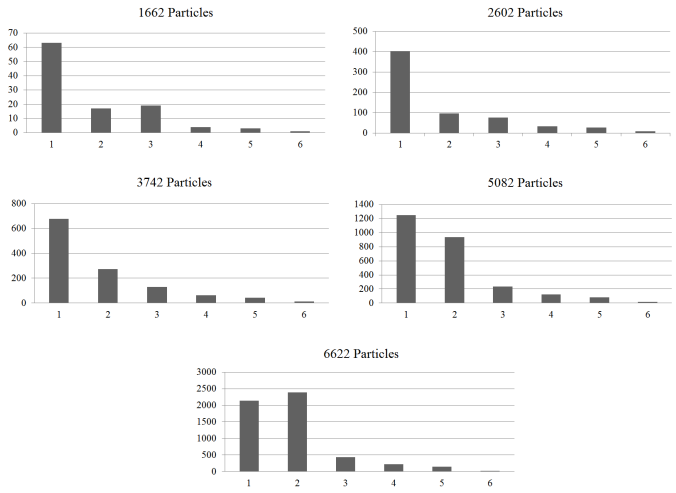


Fig. 15: Execution time in milliseconds of each function for each test case on GPU

step duration, '5' the PPE's source term calculation and '6' represents the velocity correction calculation.

It is noted that, as the total particle number increases, the time for assembling the coefficient matrix (function 2) increases more rapidly than the other functions. Since the size of the PPE's coefficient matrix is the total number of particles (TNP) in the system squared (TNP^2), which is the majority of particles in the system, any minor change or issue in the implementation can cause an expressive change in the execution time or even problems related to memory unavailability due to the high quantity of data allocated to it. Certainly, this is preventing the GPU-accelerated system from achieving the higher speedup values it is capable of, and, consequently achieve real-time for more complex and bigger simulation examples. A possible solution to this issue is to assemble the matrix directly into its sparse format since allocating memory to its dense form, performing operations to remove certain useless matrix elements and then slicing/resizing it to convert it to its sparse format, is consuming a lot of memory and processing time.

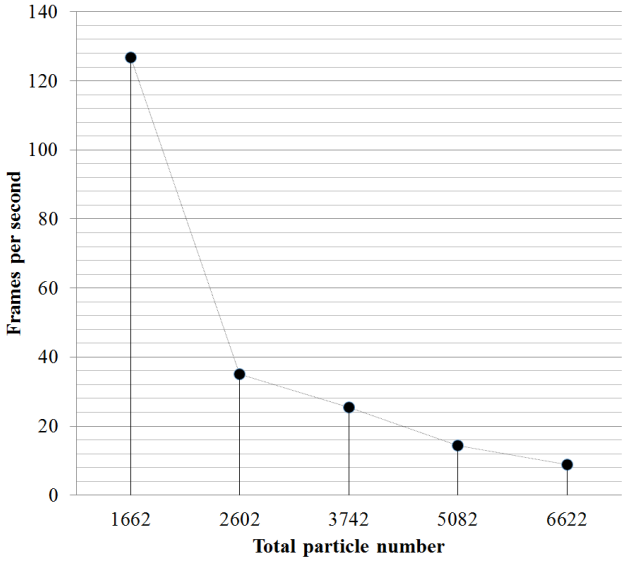


Fig. 16: Frame rate as the total particle number increases on GPU

Even though this issue occurred, the achieved speedups enable entering the field of real-time simulation or at least interactive applications (depending on the number of particles) since more than one simulation frame is being generated within a second [66]. Taking the last scenario with $6,622$ particles, where the total execution time of 100 iterations in the GPU is $11,306$ ms, approximately one frame is generated every 113.06 ms, which gives a rate of about 8.85 frames per second (fps) being generated. It is noteworthy that the speedup is shortened by the memory copy operations, i.e. when copying the particles' information from the device memory to the host memory to save the particles state of that time step. These operations, specifically, are not necessary when exhibiting the simulation in real-time using some graphics library, such as OpenGL [67] or DirectX [68]. Figure 16 shows the frame rate for each test case considering all operations in the code of the GPU version.

The memory copy operation from device to host memory is not needed when the simulation is being displayed in real-time, so, depending on how big the rendering overhead is, the frame rates presented here can reach higher values for the same amount of particles presented here when using a graphics library for the exhibition.

4) Rendering: With the purpose of showing our improved MPS method applicability in VR a simulation result was rendered using a technique based on the work of van der Laan [69]. The approach can be summarized into three steps: using the fluid particle's position, the surface depth and thickness are computed into different buffers. Then, the surface depth is smoothed using a bilateral filter and a final pass is done to combine depth, thickness and the scene behind the fluid into

the final image.

Fluid color can be calculated as Equation 23, where F is the Fresnel function, a is the refracted fluid color, b, the reflected scene color, k_s and α are constants for specular highlight, $\mathbf{n}$ is the surface normal and $\mathbf{h}$ is the half-angle between camera and light and $\mathbf{v}$ is the camera vector.

$$C_{out} = a(1 - F(\mathbf{n}\cdot\mathbf{v})) + bF(\mathbf{n}\cdot\mathbf{v}) + k_s(\mathbf{n}\cdot\mathbf{h})^{\alpha} \qquad (23)$$

Despite of the possibility of using a technique based on implicit function, which would impact positively on the realism of the resulting images, there is a negative impact on the rendering process performance, since these techniques are generally used on offline applications [70].

A simulation containing a total of $62,600$ particles, $21,296$ of these being fluid particles (which are the ones making the liquid column), was generated. Here, it is worth noting the high number of static elements (wall particles), which is not a very common issue. This happens due to some steps of the algorithm not being fully optimized (neighborhood search and matrix assembly), so a bigger test case could not be set up, which would mean more fluid particles than static elements. The time step of the displayed simulation is $0.001\ s$ and its complete duration is $3\ s$. The initial particle spacing is $0.0125\ m$. In this scenario, the height of the liquid column is a little over than half a meter ($0.55\ m$) and it has a square base, so its width and length are both equal to $0.275\ m$, half the size of its height. Figure 17, Figure 18 and Figure 19 show the state of the simulation at $0.0\ s$, $0.43\ s$ and $1.6\ s$, respectively. At $0.0\ s$ the simulation is at its initial state, at $0.43\ s$ the liquid column, which was close to the left wall, is colliding with the right wall and at $1.6\ s$ the liquid is colliding with the left wall, back from the other side. The different shades of blue of the fluid are due to amount of liquid concentrated in a single location; more liquid in one place, the darker the color at that place is and also the lesser transparent the liquid is at that place. Particles torn from the main portion of the fluid form the sprinkling aspect of the liquid in Figure 19.

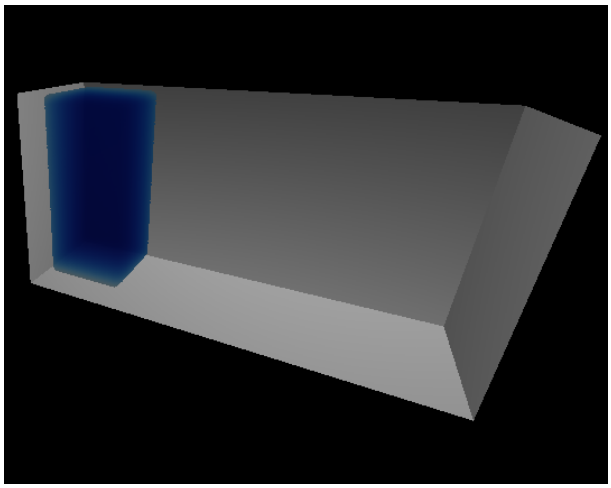


Fig. 17: Simulation state at $0.0\ s$

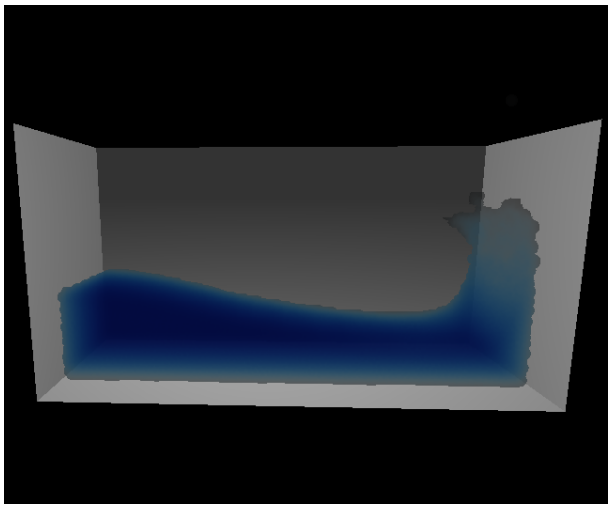


Fig. 18: Simulation state at $0.43\ s$

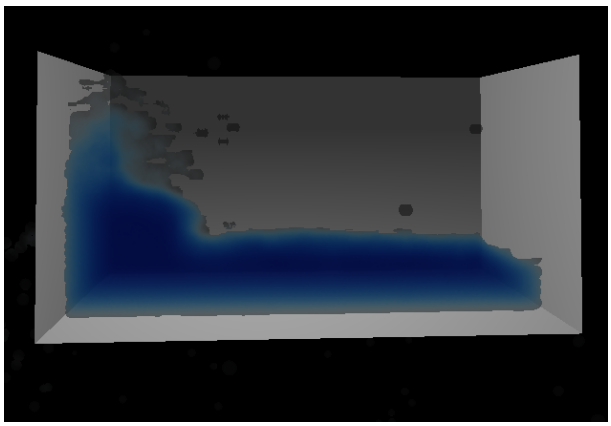


Fig. 19: Simulation state at $1.6\ s$

VI. Conclusion

The various works making efforts to improve further and further the stability and accuracy of the MPS method, show the complexity of this task, the importance of the method to the community and the great potential it has to simulate, increasingly more realistically, incompressible fluid flows. Regarding the MPS optimization, it has been gradually evolving through the combination of increasingly more sophisticated algorithms to minimize the communication during the solution of the system of linear equations (PPE) and to reorganize the systems coefficient matrix. Another aspect that helps the evolution of the method's parallelization is the hardware development.

This work provides a stable, GPU-accelerated, free-surface incompressible 3D fluid simulation method with speedups ranging from 6.41 to 10.69 times, which are considerable speedup values when compared to those found in related works. Since one of this work's goals is to enable the use

of the MPS method in VR and CG applications, the frame rate generated by each scenario was calculated. To exemplify, for a system with $2,602$ particles the frame rate reached is approximately $35.13\ fps$.

A simulation of a 3D dam break scenario was generated from the developed method with $62,600$ particles and then coupled with a rendering solution with the purpose of displaying the method's capability and potential in a VR application.

A. Future Works

The neighboring particles search algorithm can still experience improvement, both in the CPU and GPU versions of the code, with the implementation of cell grids in order to narrow the search for neighbors to the closest particles to a target particle i [13], [34]. Also, refinements in every part of the code will lead to more a optimized version; this is considered the path to a notable real-time simulation.

Further investigation in the assembly of the PPE's coefficient matrix implementation and in the program's memory usage is necessary since it most likely forbade that the system could be fully explored with respect to its size, restraining the results to smaller systems. It also prevented that the GPU version speedup could achieve higher values as the particle number increased.

ACKNOWLEDGMENT

The present work was supported by the CNPq, Conselho Nacional de Desenvolvimento Científico e Tecnológico - Brasil (National Council of Scientific and Technological Development - Brazil).

REFERENCES

[1] P. Cleary, M. Prakash, and J. Ha, "Novel applications of smoothed particle hydrodynamics (sph) in metal forming," *Journal of materials processing technology*, vol. 177, no. 1, pp. 41–48, 2006.

[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*, vol. 139, no. 1, pp. 3–47, 1996.

[3] A. A. Johnson and T. E. Tezduyar, "Advanced mesh generation and update methods for 3d flow simulations," *Computational Mechanics*, vol. 23, no. 2, pp. 130–143, 1999.

[4] E. Oñate, S. R. Idelsohn, F. Del Pin, and R. Aubry, "The particle finite element method?an overview," *International Journal of Computational Methods*, vol. 1, no. 02, pp. 267–307, 2004.

[5] P.-J. Frey and F. Alauzet, "Anisotropic mesh adaptation for cfd computations," *Computer methods in applied mechanics and engineering*, vol. 194, no. 48, pp. 5068–5082, 2005.

[6] L. B. Lucy, "A numerical approach to the testing of the fission hypothesis," *The astronomical journal*, vol. 82, pp. 1013–1024, 1977.

[7] R. A. Gingold and J. J. Monaghan, "Smoothed particle hydrodynamics: theory and application to non-spherical stars," *Monthly notices of the royal astronomical society*, vol. 181, no. 3, pp. 375–389, 1977.

[8] S. Koshizuka and Y. Oka, "Moving-particle semi-implicit method for fragmentation of incompressible fluid," *Nuclear science and engineering*, vol. 123, no. 3, pp. 421–434, 1996.

[9] R. Xu, P. Stansby, and D. Laurence, "Accuracy and stability in incompressible sph (isph) based on the projection method and a new approach," *Journal of Computational Physics*, vol. 228, no. 18, pp. 6703–6725, 2009.

[10] C. J. S. Brito, M. W. S. Almeida, A. L. B. V. e Silva, J. M. X. N. Teixeira, and V. Teichrieb, "Screen space rendering solution for multiphase SPH simulation," in *2017 19th Symposium on Virtual and Augmented Reality (SVR)*, Nov 2017, pp. 309–318.

[11] ——, "Large viscoelastic fluid simulation on GPU," in *XVI Simpósio Brasileiro de Jogos e Entretenimento Digital (SBGames)*, Nov 2017, pp. 462–469.

[12] A. L. B. Vieira e Silva, M. W. S. Almeida, C. Brito, and V. Teichrieb, "Improved meshless method for simulating incompressible fluids on GPU," in *2017 19th Symposium on Virtual and Augmented Reality (SVR)*, Nov 2017, pp. 297–308.

[13] C. Hori, H. Gotoh, H. Ikari, and A. Khayyer, "Gpu-acceleration for moving particle semi-implicit method," *Computers & Fluids*, vol. 51, no. 1, pp. 174–183, 2011.

[14] D. T. Fernandes, L.-Y. Cheng, E. H. Favero, and K. Nishimoto, "A domain decomposition strategy for hybrid parallelization of moving particle semi-implicit (mps) method for computer cluster," *Cluster Computing*, vol. 18, no. 4, pp. 1363–1377, Dec 2015. [Online]. Available: https://doi.org/10.1007/s10586-015-0483-3

[15] S. Shao and E. Y. Lo, "Incompressible sph method for simulating newtonian and non-newtonian flows with a free surface," *Advances in Water Resources*, vol. 26, no. 7, pp. 787–800, 2003.

[16] S. Koshizuka, H. Tamako, and Y. Oka, "A particle method for incompressible viscous flow with fluid fragmentation," *Comput. Fluid Dynamics J.*, 1995.

[17] H. Gotoh and A. Khayyer, "Current achievements and future perspectives for projection-based particle methods with applications in ocean engineering," *Journal of Ocean Engineering and Marine Energy*, vol. 2, no. 3, pp. 251–278, Aug 2016. [Online]. Available: https://doi.org/10.1007/s40722-016-0049-3

[18] H. GOTOH and T. SAKAI, "Lagrangian simulation of breaking waves using particle method," *Coastal Engineering Journal*, vol. 41, no. 3 & 4, pp. 303–326, 1999. [Online]. Available: https://www.worldscientific.com/doi/abs/10.1142/S0578563499000188

[19] A. Khayyer, "Improved particle methods by refined models for free-surface fluid flows," Ph.D. dissertation, Kyoto University, 2008.

[20] H. GOTOH, H. IKARI, T. MEMITA, and T. SAKAI, "Lagrangian particle method for simulation of wave overtopping on a vertical seawall," *Coastal Engineering Journal*, vol. 47, no. 2 & 3, pp. 157–181, 2005. [Online]. Available: https://www.worldscientific.com/doi/abs/10.1142/S0578563405001239

[21] A. Khayyer and H. Gotoh, "Modified moving particle semi-implicit methods for the prediction of 2d wave impact pressure," *Coastal Engineering*, vol. 56, no. 4, pp. 419–440, 2009.

[22] B.-H. Lee, J.-C. Park, M.-H. Kim, and S.-C. Hwang, "Step-by-step improvement of mps method in simulating violent free-surface motions and impact-loads," *Computer methods in applied mechanics and engineering*, vol. 200, no. 9, pp. 1113–1125, 2011.

[23] K. Shibata and S. Koshizuka, "Numerical analysis of shipping water impact on a deck using a particle method," *Ocean Engineering*, vol. 34, no. 3, pp. 585–593, 2007.

[24] H. Gotoh and T. Sakai, "Key issues in the particle method for computation of wave breaking," *Coastal Engineering*, vol. 53, no. 2, pp. 171 – 179, 2006, coastal Hydrodynamics and Morphodynamics. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S037838390500133X

[25] L. Fu and Y.-C. Jin, "Investigation of non-deformable and deformable landslides using meshfree method," *Ocean Engineering*, vol. 109, pp. 192 – 206, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0029801815004497

[26] K. Shibata, S. Koshizuka, M. Sakai, and K. Tanizawa, "Lagrangian simulations of ship-wave interactions in rough seas," *Ocean Engineering*, vol. 42, pp. 13 – 25, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0029801812000315

[27] S.-C. Hwang, A. Khayyer, H. Gotoh, and J.-C. Park, "Development of a fully lagrangian mps-based coupled method for simulation of fluid-structure interaction problems," *Journal of Fluids and Structures*, vol. 50, pp. 497 – 511, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0889974614001595

[28] A. Khayyer and H. Gotoh, "Development of cmps method for accurate water-surface tracking in breaking waves," *Coastal Engineering Journal*, vol. 50, no. 02, pp. 179–207, 2008.

[29] ——, "A higher order laplacian model for enhancement and stabilization of pressure calculation by the mps method," *Applied Ocean Research*, vol. 32, no. 1, pp. 124–131, 2010.

[30] ——, "Enhancement of stability and accuracy of the moving particle semi-implicit method," *Journal of Computational Physics*, vol. 230, no. 8, pp. 3093–3118, 2011.

[31] ——, "A 3d higher order laplacian model for enhancement and stabilization of pressure calculation in 3d mps-based simulations," *Applied Ocean Research*, vol. 37, pp. 120–126, 2012.

[32] A. L. B. Vieira e Silva, M. W. Almeida, C. J. Brito, V. Teichrieb, J. M. Barbosa, and C. Salhua, "A qualitative analysis of fluid simulation using a sph variation," in *Proceedings of the Congress on Numerical Methods in Engineering*, 2015. [Online]. Available: http://www.dem.ist.utl.pt/cmn2015/html/CD-Proceedings/PDF/Papers/CMN_2015_submission_289.pdf

[33] M. Almeida, C. Brito, A. L. B. Vieira e Silva, V. Teichrieb, and J. M. Barbosa, "Meshless methods," in *Applied Topics in Marine Hydrodynamics*, G. Assi, H. Brinati, M. de Conti, and M. Szajnbok, Eds. São Paulo: Escola Politécnica da Universidade de São Paulo (ISBN 978-85-86686-89-4), 2016, ch. 8, pp. 8.1–8.38.

[34] X. Zhu, L. Cheng, L. Lu, and B. Teng, "Implementation of the moving particle semi-implicit method on gpu," *SCIENCE CHINA Physics, Mechanics & Astronomy*, vol. 54, no. 3, pp. 523–532, 2011.

[35] D. Taniguchi, L. Sato, and L. Cheng, "Explicit moving particle simulation method on gpu clusters," *Blucher Mech. Eng. Proc. 1*, vol. 1, p. 1155, 2014.

[36] H. Gotoh, "Advanced particle methods for accurate and stable computation of fluid flows," *Frontiers of Discontinuous Numerical Methods and Practical Simulations in Engineering and Disaster Prevention*, p. 113, 2013.

[37] NVIDIA, "Cuda zone — nvidia developer," https://developer.nvidia.com/cuda-zone, accessed: 2016-01-09.

[38] G. Chen, Y. Onishi, L. Zheng, and T. Sasaki, *Frontiers of Discontinuous Numerical Methods and Practical Simulations in Engineering and Disaster Prevention*. Taylor & Francis Group, London, 8 2013.

[39] T. Kawahara and Y. Oka, "Ex-vessel molten core solidification behavior by moving particle semi-implicit method," *Journal of Nuclear Science and Technology*, vol. 49, no. 12, pp. 1156–1164, 2012.

[40] X. Sun, M. Sakai, K. Shibata, Y. Tochigi, and H. Fujiwara, "Numerical modeling on the discharged fluid flow from a glass melter by a lagrangian approach," *Nuclear Engineering and Design*, vol. 248, pp. 14–21, 2012.

[41] K. Shibata, S. Koshizuka, and Y. Oka, "Numerical analysis of jet breakup behavior using particle method," *Journal of nuclear science and technology*, vol. 41, no. 7, pp. 715–722, 2004.

[42] R. Chen, W. Tian, G. Su, S. Qiu, Y. Ishiwatari, and Y. Oka, "Numerical investigation on coalescence of bubble pairs rising in a stagnant liquid," *Chemical Engineering Science*, vol. 66, no. 21, pp. 5055–5063, 2011.

[43] A. P. A. Mustari, Y. Oka, M. Furuya, W. Takeo, and R. Chen, "3d simulation of eutectic interaction of pb–sn system using moving particle semi-implicit (mps) method," *Annals of Nuclear Energy*, vol. 81, pp. 26–33, 2015.

[44] S. Koshizuka, A. Nobe, and Y. Oka, "Numerical analysis of breaking waves using the moving particle semi-implicit method," *International Journal for Numerical Methods in Fluids*, vol. 26, no. 7, pp. 751–769, 1998.

[45] M. Kondo, K. Suto, M. Sakai, and S. Koshizuka, "Incompressible free surface flow analysis using moving particle semi-implicit method."

[46] B. Ataie-Ashtiani and L. Farhadi, "A stable moving-particle semi-implicit method for free surface flows," *Fluid Dynamics Research*, vol. 38, no. 4, pp. 241–256, 2006.

[47] G. Duan and B. Chen, "Stability and accuracy analysis for viscous flow simulation by the moving particle semi-implicit method," *Fluid Dynamics Research*, vol. 45, no. 3, p. 035501, 2013.

[48] J. Monaghan, "Sph without a tensile instability," *Journal of Computational Physics*, vol. 159, no. 2, pp. 290 – 311, 2000. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0021999100964398

[49] M. M. Tsukamoto, K. Nishimoto, and T. Asanuma, "Development of particle method representing floating bodies with highly non-linear waves," in *18th International Congress of Mechanical Engineering, COBEM*, 2005.

[50] H. Ikari and H. Gotoh, "Parallelization of mps method for 3d wave analysis," in *Advances in Hydro-science and Engineering, 8th International Conference on Hydro-science and Engineering (ICHE)*, 2008.

[51] H. Gotoh, A. Khayyer, H. Ikari, and C. Hori, "3d-cmps method for improvement of water surface tracking in breaking waves," in *Proceedings of 4th SPHERIC Workshop. Nantes, France,:[sn]*. World Scientific, 2009, pp. 265–272.

[52] T. Iribe, T. Fujisawa, and S. Koshizuka, "Reduction of communication in parallel computing of particle method for flow simulation of seaside areas," *Coastal Engineering Journal*, vol. 52, no. 04, pp. 287–304, 2010.

[53] D. T. Fernandes, "Implementação de framework computacional de paralelização híbrida do moving particle semi-implicit method para modelagem de fluidos incompressíveis," Ph.D. dissertation, Universidade de São Paulo, 2013.

[54] 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*, vol. 139, no. 1, pp. 3 – 47, 1996. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S004578259601078X

[55] S. Shao and E. Y. Lo, "Incompressible sph method for simulating newtonian and non-newtonian flows with a free surface," *Advances in Water Resources*, vol. 26, no. 7, pp. 787 – 800, 2003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0309170803000307

[56] Y. Suzuki, S. Koshizuka, and Y. Oka, "Hamiltonian moving-particle semi-implicit (hmps) method for incompressible fluid flows," *Computer Methods in Applied Mechanics and Engineering*, vol. 196, no. 29, pp. 2876–2894, 2007.

[57] A. Shakibaeinia and Y.-C. Jin, "A weakly compressible mps method for modeling of open-boundary free-surface flow," *International journal for numerical methods in fluids*, vol. 63, no. 10, pp. 1208–1232, 2010.

[58] G. Batchelor, "K. 1967 an introduction to fluid dynamics," 1970.

[59] J. J. Monaghan, "Simulating free surface flows with sph," *Journal of computational physics*, vol. 110, no. 2, pp. 399–406, 1994.

[60] A. Khayyer and H. Gotoh, "Enhancement of performance and stability of mps mesh-free particle method for multiphase flows characterized by high density ratios," *Journal of Computational Physics*, vol. 242, pp. 211–233, 2013.

[61] H. Wendland, "Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree," *Advances in computational Mathematics*, vol. 4, no. 1, pp. 389–396, 1995.

[62] "Intel Processor i7 4790 Specifications," http://ark.intel.com/products/80806/Intel-Core-i7-4790-Processor-8M-Cache-up-to-4_00-GHz, accessed: 2016-01-15.

[63] "NVIDIA GPU GeForce GTX 760 Specifications," http://www.geforce.com/hardware/desktop-gpus/geforce-gtx-760/specifications, accessed: 2016-01-15.

[64] J. E. Stone, D. Gohara, and G. Shi, "Opencl: A parallel programming standard for heterogeneous computing systems," *Computing in Science Engineering*, vol. 12, no. 3, pp. 66–73, May 2010.

[65] S. Dalton, N. Bell, L. Olson, and M. Garland, "Cusp: Generic parallel algorithms for sparse matrix and graph computations," 2015, version 0.5.1. [Online]. Available: http://cusplibrary.github.io/

[66] T. Harada, S. Koshizuka, and Y. Kawaguchi, "Smoothed particle hydrodynamics on gpus," in *Computer Graphics International*. SBC Petropolis, 2007, pp. 63–70.

[67] D. Shreiner, B. T. K. O. A. W. Group *et al.*, *OpenGL programming guide: the official guide to learning OpenGL, versions 3.0 and 3.1*. Pearson Education, 2009.

[68] B. Bargen and P. Donnelly, *Inside DirectX: in-depth techniques for developing high-performance multimedia applications*. Microsoft Press, 1998.

[69] W. J. van der Laan, S. Green, and M. Sainz, "Screen space fluid rendering with curvature flow," in *Proceedings of the 2009 symposium on Interactive 3D graphics and games*. ACM, 2009, pp. 91–98.

[70] J. Yu and G. Turk, "Reconstructing surfaces of particle-based fluids using anisotropic kernels," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 1, p. 5, 2013.