Large Viscoelastic Fluid Simulation on GPU

Caio Brito[1]* André Luiz B. Vieira-e-Silva[1] † Mozart William S. Almeida[1] ‡ João Marcelo Teixeira[2] §
Veronica Teichrieb[1] ¶

[1]UFPE, Voxar Labs - CIn, Brazil
[2]UFPE, Departamento de Eletrônica e Sistemas, Brazil

ABSTRACT

Viscoelastic materials, such as gels, gelatin, and mucus, are increasingly present in movies and games. One common method to simulate this kind of fluid is the Smoothed Particle Hydrodynamics (SPH) using a velocity correction which limits the fluid deformation providing visually consistent results. However, it is very time-consuming. This paper presents the acceleration of viscoelastic SPH using graphics processing unit using CUDA being able to simulate a large number of particles, up to 1 million. The method was implemented as an extension of the DualSPHysics open source project and the performance was compared to an OpenMP implementation, being able to achieve an average of 7.76 speedup.

Keywords: SPH, viscoelastic fluid, GPU computing , CUDA.

1 INTRODUCTION

Conventional methods, as the Finite Element Methods (FEM) [10], Finite Difference Methods (FDM) [14] and other mesh-based methods [39], are considered well consolidated and highly accurate to solve computational mechanics problems. Some of these problems, such as solid modeling, casting and the simulation of manufacturing processes, require dealing with large deformations [7]. When simulating fluid flows these deformations become even larger and the best approach for these mesh-based methods to deal with this kind of problem is to reconstruct the mesh every iteration step of the simulation [3], which is quite costly, computationally. It becomes clear that these methods favor entirely numerical precision over performance.

Physics simulations in general are commonly utilized in games, virtual reality (VR) and computer graphics (CG) applications [25] [21] [20] [5] [4], which often need to reach interactive rates or even real-time, thus, performance is an obvious key factor in these [22].

The meshless methods were proposed as an alternative to deal with those large deformations (that cause performance degradation in mesh-based methods). They use discrete elements, called particles, characterizing the system state and its evolution through time. Each particle carries a set of physical quantities and constitutive properties, such as mass, velocity, position and any other related to the problem being simulated. One of the most known meshless methods to simulate hydrodynamics problems nowadays is the Smoothed Particle Hydrodynamics (SPH) [27] [16].

Silva et al. [13] developed a weakly compressible SPH (WCSPH) method that relies only on the XSPH formulation, proposed by Schechter and Bridson [38], to simulate viscosity and to prevent the particle penetration problem in boundary condition calculation. This approach led to an SPH with less and smaller formulations with a relatively high numerical precision, which can be used for interactive applications due to its small number of calculations compared to other methods such as Solenthaler and Pajarola [43] and Ihmsen et al. [19].

With that in mind, viscoelastic materials are generally used to represent common materials, for example, egg white, gels and slime. They produce appealing visual effects, so the video games and film industries quite often require an accurate simulation of these viscoelastic properties. Exaggerated representations that real world materials do not exhibit, like very large deformations, are frequently required by such industries in order to make certain characters or effects cause a greater impression in the audience.

The work of Takahashi et al. [47] proposes a particle-based hybrid method for simulating volume preserving viscoelastic fluids with large deformations. It combines SPH and Position-based Dynamics (PBD), the later proposed by Mller et al. [34], to approximate the dynamics of viscoelastic fluids, where the idea of adaptative connections between particles is used to correct particle velocities, which are carefully calculated to not negatively affect volume preservation of materials. The authors claim that examples show the proposed hybrid method can sufficiently preserve fluid volumes and robustly generate a variety of viscoelastic behaviors, such as splitting and merging, large deformations, and Barus effect. Despite the visual quality of Takahashi's work, the method takes an average of $10s/step$ in a simulation with 110.8k particles.

only This workś contribution lies in the extension of the weakly compressible SPH method with less and smaller formulations proposed by Silva et al. [13] where the viscoelastic properties formulations proposed in the work of Takahashi et al. [47] are added to the fluid. To achieve even higher rates, aiming real-time simulation, NVidia's CUDA was used to accelerate the simulation generation, as well as OpenMP to explore the parallelism provided by multiple CPU cores. A parallelized CPU version using OpenMP and a parallelized GPU version using CUDA were developed so the simulation could achieve its maximum performance with a large number of particles, reaching interactive rates. Both versions developed were compared regarding performance.

In the next section, the state of art of SPH simulation is presented along with the related work of viscoelastic methods. Then, our method is explained to deal with viscoelastic simulation. In section 4, the DualSPHysics code is explained and what was modified in order to allow a GPU based viscoelastic SPH simulation. In section 5 the test cases to validate the approaches proposed are presented, and then the visual and performance results are presented in section 6. Finally, in section 7, the conclusions are discussed together with future possibilities and enhancements.

2 STATE OF THE ART

As previously said, since the SPH's creation, it has been vastly extended to simulate fluids and even solids due to its particle-based characteristics over mesh-based methods. One of the most straightforward adaptations of the original SPH method is the application for weakly compressible fluids, since the pressure can be calculated

*e-mail: cjsb@cin.ufpe.br
†e-mail: albvs@cin.ufpe.br
‡e-mail: mwsa@cin.ufpe.br
§e-mail: jmxnt@cin.ufpe.br
¶e-mail: vt@cin.ufpe.br

by a simple equation of state, differently from the truly incompressible formulation, in which a Poisson Pressure Equation (PPE) is solved every iteration [1]. In some works, weakly compressible and truly incompressible SPH implementations are explicitly compared in relation to its precision and performance [46] [40] [26]. The main difference between the original method and the newer ones is the inclusion of boundary conditions [45] [58].

Besides the compressibility factor present in the fluids, some other elements can be introduced to the fluid behaviour, depending on the kind of problem being studied, like the viscosity, presence or absence of turbulence, type of smoothing function, among others. Some works present results for adjustments in the viscosity according to the problem being focused [36] [52] [37] [42] [56]. For turbulent flows, some papers try to adapt known methods into the particles systems [15] [41].

Given the discrete characteristics of the simulation, parallel solutions of the method become straightforward, using specific languages to extract the hardware's parallelism (i.e. OpenMP), cluster technology and general purpose programming for graphics processors (GPGPU) techniques, in order to divide the task between core processors, thus, decreasing the time consumption of the simulation computation [8] [18] [24].

As for the gaming industry, several works that use the SPH method focus on simulating fluids for interactive applications and for real-time ones, starting with the work of Mller et al. [33]. Other works focus on simulating fluids with different properties and features, so it is possible to represent most of the fluid types and behaviors. Performance needs to be the focus if the application will be utilized for gaming purposes, so the aim should always be a simulation running at least near to real-time, however, the fluid also needs to preserve its physical properties and present them coherently, as much as possible [21] [22].

2.1 Related Work

Works involving the SPH method augmented with viscoelastic formulations are quite recent, still, lots of works are already benefiting from each other. The visual appeal and the high range of applicability of this type of simulation in fields such as medicine, biology and the entertainment industry [55, 57] may be the reason of its instant popularity. In this subsection, some similar works to this one are presented, showing its importance to the community.

Clavet et al. [6] took advantage of the method proposed by Miller and Pearce [29] and Terzopoulos et al. [49], which is a spring-based method, and combined it with SPH to simulate materials with elasticity, plasticity, and viscosity, adopting a prediction-relaxation scheme. This spring-based model computes attraction and repulsion forces between particles to successfully simulate the viscoelastic properties of certain materials. Another similar spring-based method was also proposed by Takahashi et al. [48], who used PBD to simulate fluids with viscosity and elasticity in a unified framework.

Muller et al. [35] proposed the addition of an elasticity term to formulations which uses Moving Least Square (MLS) to simulate elastoplastic objects. Solenthaler et al. [44] adopted the formulation of this elasticity term and computed it using SPH instead of MLS to allow for robustly simulating fluid with various properties under some conditions. The method Solenthaler et al. [44] proposed was extended to handle rotational motions of elastic materials [2]. Mao and Yang [28] introduced a viscoelastic force term into the Navier-Stokes equations to simulate viscoelastic fluids.

Xu et al. [55] propose a viscoelastic SPH to be used in biological applications, more specifically a multiscale SPH method to simulate transient viscoelastic flows by using a bead-spring chain description of polymer molecule. To achieve the obtained results the authors came up with a methodology that couples macroscopic conservation equations for mass and momentum with a differential equation for bead-spring chain dynamics, which, when solved, the polymeric stress is obtained.

Xu et al. [54] proposed an improved weakly compressible SPH method to simulate transient free surface flows of viscous and viscoelastic fluids. The improvement to the WCSPH formulations include a greater accuracy and stability due to a correction in the kernel gradient calculation, and an enhanced computation of pressure distribution in the dynamics of the fluid due to corrections in the continuity equation. The effectiveness of the method is successfully proved through a series of test scenarios common in the literature, like the dam breaking flow, stretching of a water drop and a viscoelastic fluid drop against a wall.

Heck et al. [17] also propose a viscoelastic SPH to biology purposes, although this time to model extracellular matrix viscoelasticity for an extracellular matrix in contact with a migrating cell. Also, it improves contact mechanics by modeling it based on an existing boundary method in SPH, which is extended to allow the modeling of moving boundaries in contact with a viscoelastic solid. This result should enable the field researchers to model and understand realistic cellmatrix interactions in the future.

3 A Viscoelastic Weakly Compressible SPH Method

In this section, the weakly compressible SPH method, based on the work of Silva et al. [13], and its modifications to support viscoelastic fluids, are explained.

3.1 SPH Formulation

The SPH is a Lagrangian method created originally to simulate astrophysics problems and lately has been used mainly to simulate hydrodynamics problems solving the Navier-Stokes equation, defined by Eq. (1).

$$\frac{d\mathbf{u}}{dt} = -\frac{1}{\rho}\nabla P + \frac{1}{\rho}\nabla\cdot\boldsymbol{\tau} + Fext \tag{1}$$

The Navier-Stokes equation describes the fluid movement regarding three main components: pressure, viscosity and external forces. The SPH solves the fluid movement by considering the fluid as a weakly compressible system, which is based on the fact that every incompressible fluid is a little compressible, and because of that, the method simulates a quasi-incompressible equation to model the simulation.

Silva et al. [13] propose a SPH formulation into a series of steps:

The first step is to calculate the particle density, which is calculated using the continuity equation as in Eq. (2).

$$\frac{d\rho_i}{dt} = \sum_j m_j(\mathbf{u}_i - \mathbf{u}_j)\nabla W_{ij} \tag{2}$$

where m is the particle mass, ρ is the particle density, $\mathbf{u}$ is the particle velocity and W is the kernel function.

After calculating the density of the particles in the system, the next step is to solve their pressures, which are calculated by the Tait's equation (3), as shown in the work of Silva et al. [13]:

$$P_i = B((\frac{\rho_i}{\rho_0})^\gamma - 1) \tag{3}$$

where B is the pressure constant, ρ_0 is the rest density of the fluid and γ is a constant that usually has a value of 7.

The pressure force is commonly calculated using Eq. (4). This approach ensures a modular equality between two particles and conserves linear and angular momentum, leading to a more stable simulation, as shown in [32]:

$$\frac{1}{\rho_i}\nabla P_i = \sum_j m_j(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2})\nabla W_{ij} \tag{4}$$

To simulate the viscosity and to prevent a particle penetration problem, the XSPH method is used. This method forces particles near each other to move with almost the same velocity. The method is computationally cheaper than other methods and has only one tunable parameter making it easy to change [38]. This parameter controls the viscosity influence on the fluid; the higher the parameter value, the greater the influence of the viscosity of the fluid.

To conserve linear and angular momentum, Eq. (5) was used to calculate an intermediate velocity u^* and Eq. (6) was used to calculate the new velocity.

$$u_i^* = u_i + a_i \triangle t \tag{5}$$

$$u_i = u_i^* + \varepsilon \sum_j m_b \frac{u_i^* - u_j^*}{\rho_j} w_{ij} \tag{6}$$

where a_i is the particles' acceleration and ε is the tunable parameter of the XSPH method.

Finally, the last term in the Navier-Stokes governing equation is related to the external forces in the system which in most systems is the gravity. The particles' new positions are calculated using a simple first order Euler time integration Eq. (7), as suggested in [38]:

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{u}_i^{t+1} \triangle t \tag{7}$$

where $\mathbf{a}_i$ is the particle's i acceleration and $\triangle t$ is the time step of the simulation.

The fluid flow is simulated until reaching the stop criteria, which in this work is the total time of simulation. The SPH method flow can be found in Fig. 1.

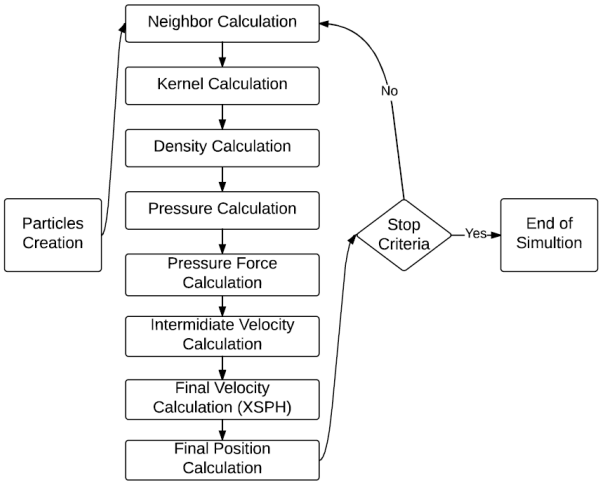

Figure 1: SPH method algorithm flow.

3.2 Viscoelastic Scheme

To handle viscoelastic simulations, [47] proposes a velocity correction $\triangle \mathbf{v}$ that is based on a set of pairwise connections that is created in the beginning of the simulation with distance $r_i j$, which is the initial particle distance. The velocity correction is calculated as Eq. (8):

$$\triangle \mathbf{u} = -\frac{1}{\triangle t} \sum_j^{C_f} \frac{c_i + c_j}{2} \frac{m_i}{m_i + m_j} D_{ij} \frac{\mathbf{x}_{ij}}{\|\mathbf{x}_{ij}\|} \tag{8}$$

where C_f is the number of connected fluid particles to particle i, c is the correction coefficient and D is a function defined as $D_{ij} = max(\|\mathbf{x}_{ij}\| - rij, 0)$.

The function D_{ij} expresses that the velocity correction is only performed when the particle distance is larger than the initial particle distance $r_i j$, which means that the fluid is in expansion and the correction coefficient controls the stiffness of viscoelastic materials.

3.2.1 One Way Solid-Fluid Coupling

In order to compute a one way solid-fluid coupling and create a sticking behavior on the boundary, Eq. (8) must also be computed using the boundary particles as described in Eq. (9), which assumes $m_k = \infty$ [47]:

$$\triangle \mathbf{u} = -\frac{1}{\triangle t} \sum_j^{C_f} \frac{c_i + c_j}{2} \frac{m_i}{m_i + m_j} D_{ij} \frac{\mathbf{x}_{ij}}{\|\mathbf{x}_{ij}\|} - \frac{1}{\triangle t} \sum_k^{C_b} \frac{c_i + c_k}{2} D_{ik} \frac{\mathbf{x}_{ik}}{\|\mathbf{x}_{ik}\|} \tag{9}$$

where C_b is the number of connected boundary particles to particle i.

Using this coupling approach, when the fluid hits a boundary, it should stick on the boundary instead of bouncing back into the air.

The viscoelastic SPH method flow can be found in Fig. 2.

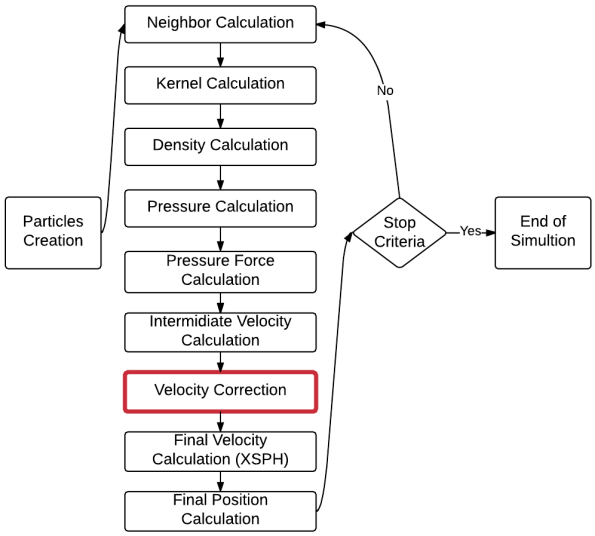

Figure 2: Viscoelastic SPH method algorithm flow.

4 GPU Implementation

In this section, the parallel implementation is explained along with its modifications to support viscoelastic fluid.

4.1 DualSPHysics

DualSPHysics is an open-source project created with the purpose of encouraging other researchers to study SPH and it has GNU General Public License as published by the Free Software Foundation [9]. The code is available for CPU and GPU SPH simulation being able to compute the fluid behavior with numerical stability, accuracy and it has been used for many applications [30] [50] [31]. The code is written in C++ using OpenMP for the parallel CPU implementation and uses CUDA for the GPU implementation.

The SPH code implemented on the DualSPHysics can be divided into 3 main phases [9]: (1) Neighborhood organization, (2) particles interaction and (3) time integration.

In phase 1, the particles neighborhood is computed using a cell-linked list (CLL) approach [11]. In the CLL, the domain is divided into square cells (2D) or cube cells (3D) with twice the influence radius (h) and the particles are stored into a list depending on the cell they belong to.

So, in order to create the neighborhood of a particle, only particles of adjacent cells are considered potential neighbors. It is worth noticing that a neighbor list is not created but a list of particles is reordered depending on the cell they belong to. This approach is faster and consumes less memory than creating a real list of neighbor particles [11]. To conclude the first step, every array is reordered using the list of particles.

The second phase calculates the interaction between neighbor particles by solving the momentum and continuity equation. The interaction between two particles occurs if the distance is less than $2h$.

Using the results from the second phase, the time integration is calculated, such as verlet and symplectic. In this phase, the new particles density, velocity and position are calculated, a new timestep can be computed, particle information is stored in the hard drive and the arrays are ordered so that particles inside the same cell can be close to each other.

4.2 DualSPHysics Modifications

Three parts of the DualSPHysics code were modified: (1) Euler integration, (2) XSPH calculation and (3) velocity correction for viscoelastic behavior.

4.2.1 Euler Integration

The Euler integration is used to calculate the new position of a fluid particle as shown in Eq. (7). The particles' position and velocity are stored in `Posxyg`, `Poszg` and `Velrhopg` arrays for the GPU solution, which are already used on the original DualSPHysics implementation.

4.2.2 XSPH Calculation

To calculate the XSPH, the summation from Eq. (6) is computed using a CUDA kernel which interacts through the particles neighborhood and is stored into a `float3`. In sequence, another CUDA kernel computes the final velocity as Eq. (6) and stores the result into the velocity array as shown in Listing 1. Line 4 calculates the particle index using the CUDA kernel information and, if it is a fluid particle, the velocity `velrhopnew` is updated using the XSPH result.

Listing 1: XSPH GPU Code.

```
1  template<bool floating, bool shift> __global__ void
       KerComputeStepVelocity
2  (unsigned n, unsigned npb, const float4 *velrhop1, const
       float3 *xsph, double dt, word *code, float4 *
       velrhopnew, double eps)
3  {
4  unsigned p = blockIdx.y*gridDim.x*blockDim.x + blockIdx.x
       *blockDim.x + threadIdx.x;
5    if (p<n){
6          if (p<npb){// -Boundary particle
7              velrhopnew[p] = make_float4(0, 0, 0, velrhop1
                   [p].w);
8          }
9          else{ // -Fluid particle
10
11             float4 rvelrhop = velrhop1[p];
12             float3 xsphp = xsph[p];
13
14             //velocity update using the xsph
15             rvelrhop.x = float(double(rvelrhop.x) -
                   double(xsphp.x)*eps);
16             rvelrhop.y = float(double(rvelrhop.y) -
                   double(xsphp.y)*eps);
17             rvelrhop.z = float(double(rvelrhop.z) -
                   double(xsphp.z)*eps);
18             velrhopnew[p] = rvelrhop;
19
20     }
21   }
22 }
```

4.2.3 Velocity Correction for Viscoelastic Behavior

In order to simulate the fluid viscoelastic behavior, first it is necessary to create the connections from the fluid particles, so the initial fluid position is kept into a $float4$ array which is used for creating the connections using a CLL. To calculate the one way solid-fluid coupling, the connections are updated to the point where the fluid has the highest number of boundary neighbor particles. In sequence, two CUDA kernel are created. The first one interacts through the particles connections and calculates the velocity correction as Eq. (8). Then, the second kernel uses the velocity correction and updates the fluid velocity as shown in Listing 2, which updates the the fluid velocity `velrhopnew` using the velocity correction result.

Listing 2: Velocity Correction GPU Code.

```
1  template<bool floating, bool shift> __global__ void
       KerComputeStepVelocityCorr
2  (unsigned n, unsigned npb, const float4 *velrhop1, const
       float3 *velcorr, double dt, word *code, float4 *
       velrhopnew)
3  {
4          unsigned p = blockIdx.y*gridDim.x*blockDim.x +
               blockIdx.x*blockDim.x + threadIdx.x;
5    if (p<n){
6          if (p<npb){ //-Boundary particle
7                  velrhopnew[p] = make_float4(0, 0, 0,
                       velrhop1[p].w);
8          }
9          else{ // -Fluid particle
10
11                 float4 rvelrhop = velrhop1[p];
12                 float3 velcorrp = velcorr[p];
13                 /* Velocity update using the
14                    viscoelastic correction */
15                 rvelrhop.x = float(double(rvelrhop.x) -
                       double(velcorrp.x)*(1 / dt));
16                 rvelrhop.y = float(double(rvelrhop.y) -
                       double(velcorrp.y)*(1 / dt));
17                 rvelrhop.z = float(double(rvelrhop.z) -
                       double(velcorrp.z)*(1 / dt));
18                 velrhopnew[p] = rvelrhop;
19
20     }
21   }
22 }
```

5 Test Cases

To validate the visual results from the SPH method, three different objects were released with no initial velocity towards the ground: a sphere, a cube and a triangular base pyramid.

The sphere is centered in the point $(0.3, 0.3, 0.3)$ with a $0.2m$ radius. The cube is centered in the point $(0.45, 0.45, 0.45)$ with lateral size equal to $0.3m$. Both sphere and cube have initial particle spacing of $0.01m$ and composed by 40k particles. The pyramid is composed by the points $(1,1,2)$, $(4,1,2)$, $(1,5,2)$ and $(2,2,4)$ and has initial particle spacing of 0.05 resulting in 60k particles.

The sphere test case is also repeated using three correction coefficients: 0.01, 0.001 and 0.0005 to understand the influence of the parameters on the fluid behavior. This case is also used to compare the performance of the GPU solution with the OpenMP implementation, by using $100k$, $300k$, $500k$, $750k$ and $1M$ fluid particles with 0.01 correction coefficient value.

To validate the one way solid-fluid coupling, the sphere test and the pyramid test are used with the same aforementioned parameter but the pyramid has a initial velocity $v_0 = (10,0,0)$ which makes the object to be thrown against the wall on $x = 5$.

6 RESULTS

Firstly, a visual analysis of the methods will be made using different shapes and correction coefficient values. Then, a performance analysis will be done comparing the computational time of the GPU and CPU implementations.

A technique based on the work of van der Laan [51] was used to render the simulation results. 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. The fluid color can be calculated as by Eq. (10), where F is the Fresnel function, a is the refracted fluid color, b is the reflected scene color, k_s and α are constants for the specular highlight, $\mathbf{n}$ is the surface normal and $\mathbf{h}$ is the half-angle between the camera and the light, and $\mathbf{v}$ is the camera vector.

$$C_{out} = a(1 - F(\mathbf{n}\cdot\mathbf{v})) + bF(\mathbf{n}\cdot\mathbf{v}) + k_s(\mathbf{n}\cdot\mathbf{h})^{\alpha} \quad (10)$$

The CPU used to run the test cases was an Intel Core i7-4790L CPU @ 4.00 GHz with 32 GB of installed RAM and a Windows 10 64-bit operating system (x64). The GPU used was a NVIDIA GeForce 960 with 4 GB of RAM.

6.1 Visual Analysis

As previously described, the different shapes were released towards the ground: a sphere, a cube and a pyramid. By the fact that viscoelastic fluids are being simulated, it is expected the fluid to have a tendency of returning to its initial shape and the fluid should not spread on the floor like a regular fluid, but instead it should show a elastic deformation.

The expected result is achieved with the three objects as shown in Fig. 3, Fig. 4 and Fig. 5, as the fluid tends to keep its original shape and has a gelatinous appearance. Also, the fluid bounces back to the air because the solid-fluid coupling is not being considered.

When the one way solid-fluid coupling is computed the fluid does not bounce back and sticks on the boundary. This behavior is shown in Fig. 6 and in Fig. 7 in which the fluid particles stick on the boundary but still surfer the influence of the gravitational force. This behavior can be seen in the point of the pyramid which tends to bend in the gravity direction.

Also, the influence of the correction coefficient $(c)$was analyzed and as the coefficient value decreases, less elastic the fluid becomes and spreads on the floor losing its original shape. This result can be seen in Fig. 8, which shows the three different fluids, with the same original shape but different correction coefficients, after hitting the floor.

The achieved results are visually coherent and resemble a gelatin, being visually similar to the one found in the work of Takahashi et al. [47]. In addition, it is possible to handle large deformations, different from the work of Mao and Yang [28].

6.2 Performance Analysis

Table 1 shows the simulation times of the parallel CPU implementation using OpenMP and the parallel GPU implementation using CUDA for 5 different numbers of particles. The time on the table represents the computation time of calculating the particles interactions, the new particle positions and ordering the arrays according to their cell.

Table 1: CPU and GPU computation times and their respective speedups for each test case.

# Particles	CPU (ms)	GPU (ms)	Speedup
100k	461	64	7.2
300k	1544	196	7.8
500k	2579	325	7.9
750k	3956	495	7.9
1M	5368	670	8.0

From the aforementioned results, it is possible to notice that the GPU version provides an average of 7.76 speedup in comparison to the parallel CPU implementation.

Two main calculations occur in a timestep: solving the new particle position and reordering the arrays using the particle cell position. In both the GPU and CPU implementations, the reordering calculation takes less than 1% of the computation time. But, in the GPU version the computation time increases from 1ms to 3ms, while in the CPU version increases from 3ms to 46ms when compared using 100k and 1M particles, respectively.

The work of Takahashi et al. [47] is able to solve the viscoelastic SPH in $10s/step$ in a simulation with 110.8k particles. The proposed solution in this work presents a faster simulation being able to solve a simulation with 100k particles in 64ms (15fps) and also to simulate 1M particles in 670ms which is better performance than the work of Takahashi. But, unlike the work of Takahashi et al., our work is not able to deal with connection control and a full solid-fluid coupling.

7 CONCLUSION

This work had a twofold contribution: to extend the work of Silva et al. [13] to be able to simulate a viscoelastic fluid using a SPH method and a parallel implementation using the DualSPHysics open-source code. A parallelized CPU version using OpenMP and a parallelized GPU version using CUDA were developed and can simulate up to millions of particles in interactive rates. The GPU implementation was able to achieve a maximum of 15 fps with 100k particles and an average speedup of 7.76 in comparison to the parallel CPU implementation.

7.1 Future Works

This work can be improved in many ways. First, the SPH model can be improved to deal with connection control being able to simulate interaction between two different fluids and splitting behavior [47]. Another possibility is to compared the numerical solution with other works, such as [47] and [28].

Also, different particle based methods can be implemented using the viscoelastic approach and the GPU code, such as, ISPH [53] and MPS [23] [12].

REFERENCES

[1] M. Asai, A. M. Aly, Y. Sonoda, and Y. Sakai. A stabilized incompressible sph method by relaxing the density invariance condition. *Journal of Applied Mathematics*, 2012, 2012.

[2] M. Becker, M. Ihmsen, and M. Teschner. Corotated sph for deformable solids. In *NPH*, pages 27–34, 2009.

[3] 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*, 139(1):3–47, 1996.

[4] R. Bridson. *Fluid simulation for computer graphics*. CRC Press, 2015.

[5] C. Chae and K. Ko. Introduction of physics simulation in augmented reality. In *Ubiquitous Virtual Reality, 2008. ISUVR 2008. International Symposium on*, pages 37–40. IEEE, 2008.

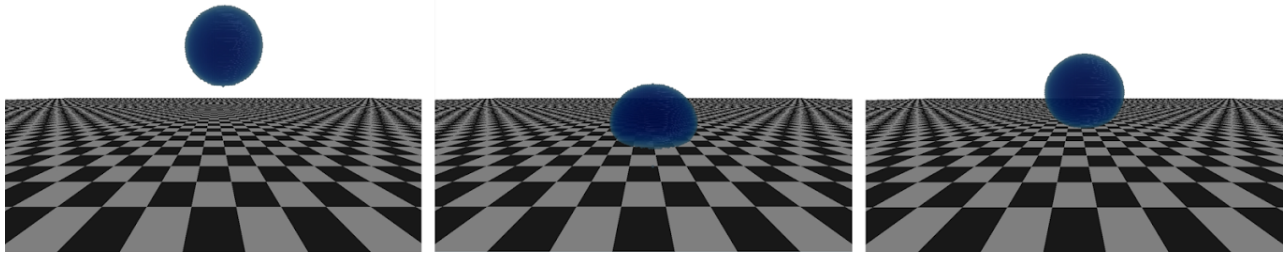

Figure 3: Sphere Test Case Results.

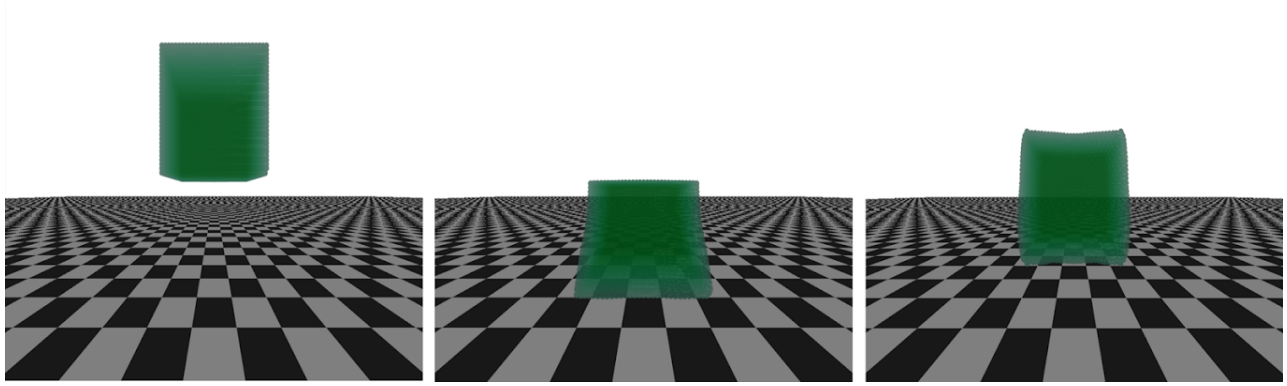

Figure 4: Square test case results.

[6] S. Clavet, P. Beaudoin, and P. Poulin. Particle-based viscoelastic fluid simulation. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 219–228. ACM, 2005.

[7] P. Cleary, M. Prakash, and J. Ha. Novel applications of smoothed particle hydrodynamics (sph) in metal forming. *Journal of materials processing technology*, 177(1):41–48, 2006.

[8] A. Crespo, J. M. Dominguez, A. Barreiro, M. Gómez-Gesteira, and B. D. Rogers. Gpus, a new tool of acceleration in cfd: efficiency and reliability on smoothed particle hydrodynamics methods. *PLoS One*, 6(6):e20685, 2011.

[9] A. Crespo, J. Domnguez, B. Rogers, M. Gmez-Gesteira, S. Longshaw, R. Canelas, R. Vacondio, A. Barreiro, and O. Garca-Feal. Dualsphysics: Open-source parallel {CFD} solver based on smoothed particle hydrodynamics (sph). *Computer Physics Communications*, 187:204 – 216, 2015.

[10] G. Dhatt, G. Touzot, et al. *Finite element method*. John Wiley & Sons, 2012.

[11] J. M. Domínguez, A. J. Crespo, M. Gómez-Gesteira, and J. C. Marongiu. Neighbour lists in smoothed particle hydrodynamics. *International Journal for Numerical Methods in Fluids*, 67(12):2026–2042, 2011.

[12] G. Duan, B. Chen, X. Zhang, and Y. Wang. A multiphase mps solver for modeling multi-fluid interaction with free surface and its application in oil spill. *Computer Methods in Applied Mechanics and Engineering*, 320:133 – 161, 2017.

[13] A. L. V. 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 *Congresso de Mtodos Numricos em Engenharia*, 2015.

[14] E. Fadlun, R. Verzicco, P. Orlandi, and J. Mohd-Yusof. Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations. *Journal of computational physics*, 161(1):35–60, 2000.

[15] M. Ferrand, D. Laurence, B. D. Rogers, D. Violeau, and C. Kassiotis. Unified semi-analytical wall boundary conditions for inviscid, laminar or turbulent flows in the meshless sph method. *International Journal for Numerical Methods in Fluids*, 71(4):446–472, 2013.

[16] R. A. Gingold and J. J. Monaghan. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly notices of the royal astronomical society*, 181(3):375–389, 1977.

[17] T. Heck, B. Smeets, S. Vanmaercke, P. Bhattacharya, T. Odenthal, H. Ramon, H. V. Oosterwyck, and P. V. Liedekerke. Modeling extracellular matrix viscoelasticity using smoothed particle hydrodynamics with improved boundary treatment. *Computer Methods in Applied Mechanics and Engineering*, 322:515 – 540, 2017.

[18] C. Hori, H. Gotoh, H. Ikari, and A. Khayyer. Gpu-acceleration for moving particle semi-implicit method. *Computers & Fluids*, 51(1):174–183, 2011.

[19] M. Ihmsen, J. Cornelis, B. Solenthaler, C. Horvath, and M. Teschner. Implicit incompressible sph. *IEEE Transactions on Visualization and Computer Graphics*, 20(3):426–435, 2014.

[20] M. Joselli, M. Zamith, E. W. G. Clua, A. Montenegro, R. C. P. Leal-Toledo, L. Valente, and B. Feijó. An architecture with automatic load balancing and distribution for digital games. In *Games and Digital Entertainment (SBGAMES), 2010 Brazilian Symposium on*, pages 59–70. IEEE, 2010.

[21] J. R. d. S. Junior, E. W. Clua, A. Montenegro, and P. A. Pagliosa. Fluid simulation with two-way interaction rigid body using a heterogeneous gpu and cpu environment. In *Games and Digital Entertainment (SBGAMES), 2010 Brazilian Symposium on*, pages 156–164. IEEE, 2010.

[22] J. R. d. S. Junior, M. Joselli, M. Zamith, M. Lage, E. Clua, E. Soluri, and N. Tecnologia. An architecture for real time fluid simulation using multiple gpus. *XI SBGames, Brasília*, page 8, 2012.

[23] S. Koshizuka and Y. Oka. Moving-particle semi-implicit method for fragmentation of incompressible fluid. *Nuclear science and engineering*, 123(3):421–434, 1996.

[24] Ø. E. Krog and A. C. Elster. Fast gpu-based fluid simulations using

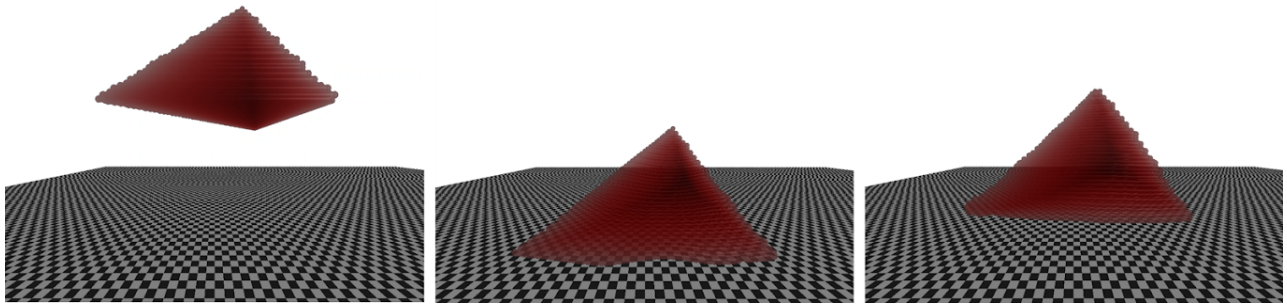

Figure 5: Pyramid test case results.

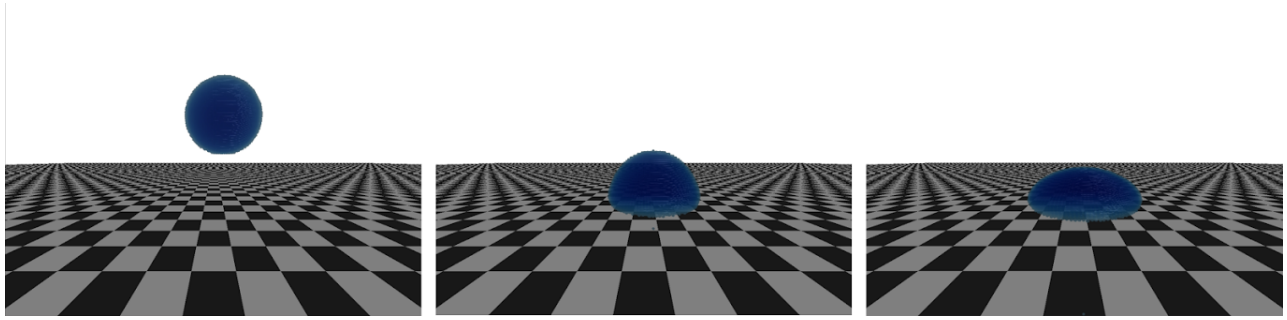

Figure 6: Sphere test case with one way solid-fluid coupling.

sph. In *International Workshop on Applied Parallel Computing*, pages 98–109. Springer, 2010.

[25] O. Lamotte, S. Galland, J.-M. Contet, and F. Gechter. Submicroscopic and physics simulation of autonomous and intelligent vehicles in virtual reality. In *Advances in System Simulation (SIMUL), 2010 Second International Conference on*, pages 28–33. IEEE, 2010.

[26] E.-S. Lee, C. Moulinec, R. Xu, D. Violeau, D. Laurence, and P. Stansby. Comparisons of weakly compressible and truly incompressible algorithms for the sph mesh free particle method. *Journal of computational physics*, 227(18):8417–8436, 2008.

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

[28] H. Mao and Y.-H. Yang. Particle-based non-newtonian fluid animation with heating effects. *University of Alberta, Tech. Rep*, 2006.

[29] G. Miller and A. Pearce. Globular dynamics: A connected particle system for animating viscous fluids. *Computers & Graphics*, 13(3):305–309, 1989.

[30] A. Mokos, B. D. Rogers, and P. K. Stansby. A multi-phase particle shifting algorithm for sph simulations of violent hydrodynamics with a large number of particles. *Journal of Hydraulic Research*, 55(2):143–162, 2017.

[31] A. Mokos, B. D. Rogers, P. K. Stansby, and J. M. Domínguez. Multi-phase sph modelling of violent hydrodynamics on gpus. *Computer Physics Communications*, 196:304–316, 2015.

[32] J. J. Monaghan. Smoothed particle hydrodynamics. *Reports on progress in physics*, 68(8):1703, 2005.

[33] M. Müller, D. Charypar, and M. Gross. Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 154–159. Eurographics Association, 2003.

[34] M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff. Position based dynamics. *Journal of Visual Communication and Image Representation*, 18(2):109–118, 2007.

[35] M. Müller, R. Keiser, A. Nealen, M. Pauly, M. Gross, and M. Alexa. Point based animation of elastic, plastic and melting objects. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 141–151. Eurographics Association, 2004.

[36] J. Pozorski and A. Wawreńczuk. Sph computation of incompressible viscous flows. *Journal of Theoretical and Applied Mechanics*, 40(4):917–937, 2002.

[37] A. Rafiee, M. Manzari, and M. Hosseini. An incompressible sph method for simulation of unsteady viscoelastic free-surface flows. *International Journal of Non-Linear Mechanics*, 42(10):1210–1223, 2007.

[38] H. Schechter and R. Bridson. Ghost sph for animating water. *ACM Transactions on Graphics (TOG)*, 31(4):61, 2012.

[39] J. A. Sethian. *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*, volume 3. Cambridge university press, 1999.

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

[41] J. Shao, H. Li, G. Liu, and M. Liu. An improved sph method for modeling liquid sloshing dynamics. *Computers & Structures*, 100:18–26, 2012.

[42] L. D. G. Sigalotti, J. Klapp, E. Sira, Y. Meleán, and A. Hasmy. Sph simulations of time-dependent poiseuille flow at low reynolds numbers. *Journal of computational physics*, 191(2):622–638, 2003.

[43] B. Solenthaler and R. Pajarola. Predictive-corrective incompressible sph. In *ACM transactions on graphics (TOG)*, volume 28, page 40. ACM, 2009.

[44] B. Solenthaler, J. Schläfli, and R. Pajarola. A unified particle model for fluid–solid interactions. *Computer Animation and Virtual Worlds*, 18(1):69–82, 2007.

[45] B. Song and L. Dong. A new boundary treatment method for sph and application in fluid simulation. In *Information and Computing (ICIC), 2010 Third International Conference on*, volume 4, pages 82–85. IEEE, 2010.

[46] K. Szewc, J. Pozorski, and J.-P. Minier. Analysis of the incompressibility constraint in the smoothed particle hydrodynamics method. *International Journal for Numerical Methods in Engineer-*

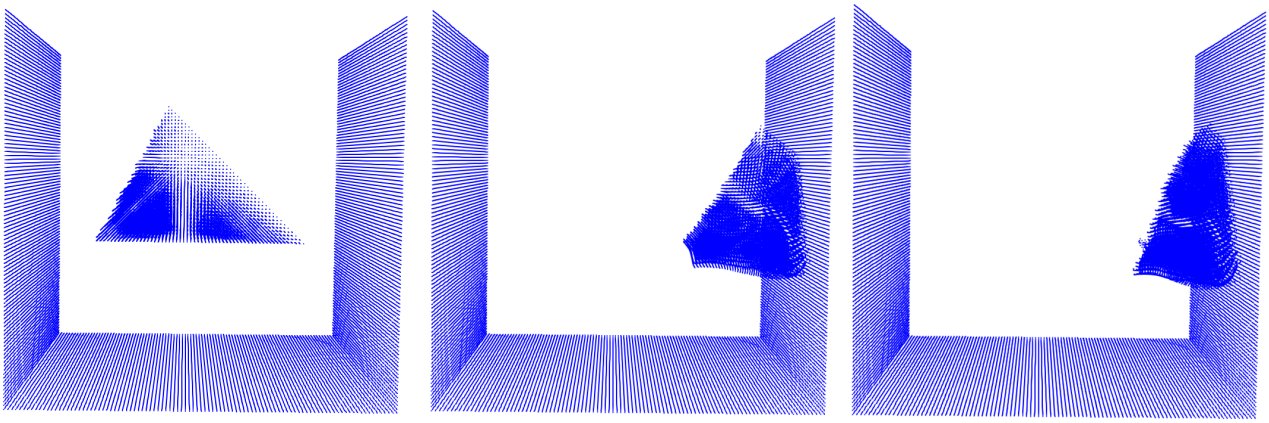

Figure 7: Pyramid test case with one way solid-fluid coupling and $v_0 = (10, 0, 0)$.

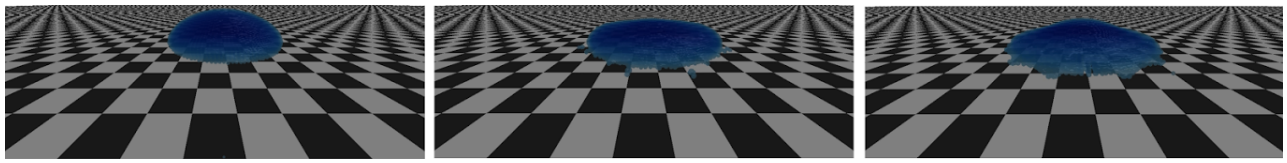

Figure 8: Sphere Test Case Results. Left: c = 0.01. Middle: c = 0.001. Right: c = 0.0005.

ing, 92(4):343–369, 2012.

[47] T. Takahashi, Y. Dobashi, I. Fujishiro, and T. Nishita. Volume preserving viscoelastic fluids with large deformations using position-based velocity corrections. *The Visual Computer*, 32(1):57–66, 2016.

[48] K. Takamatsu and T. Kanai. A fast and practical method for animating particle-based viscoelastic fluids. *International Journal of Virtual Reality*, 10(1):29 – 35, 2011.

[49] D. Terzopoulos, J. Platt, and K. Fleischer. Heating and melting deformable models. *Computer Animation and Virtual Worlds*, 2(2):68–73, 1991.

[50] R. Vacondio, B. Rogers, P. Stansby, and P. Mignosa. Variable resolution for sph in three dimensions: Towards optimal splitting and coalescing for dynamic adaptivity. *Computer Methods in Applied Mechanics and Engineering*, 300:442–460, 2016.

[51] 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*, pages 91–98. ACM, 2009.

[52] S. Watkins, A. Bhattal, N. Francis, J. Turner, and A. P. Whitworth. A new prescription for viscosity in smoothed particle hydrodynamics. *Astronomy and Astrophysics Supplement Series*, 119(1):177–187, 1996.

[53] 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*, 228(18):6703–6725, 2009.

[54] X. Xu and X.-L. Deng. An improved weakly compressible sph method for simulating free surface flows of viscous and viscoelastic fluids. *Computer Physics Communications*, 201:43 – 62, 2016.

[55] X. Xu and P. Yu. A multiscale sph method for simulating transient viscoelastic flows using bead-spring chain model. *Journal of Non-Newtonian Fluid Mechanics*, 229:27 – 42, 2016.

[56] X. Yang, M. Liu, and S. Peng. Smoothed particle hydrodynamics modeling of viscous liquid drop without tensile instability. *Computers & Fluids*, 92:199–208, 2014.

[57] T. Y. Yeh, P. Faloutsos, and G. Reinman. Enabling real-time physics simulation in future interactive entertainment. In *Proceedings of the 2006 ACM SIGGRAPH Symposium on Videogames*, Sandbox '06, pages 71–81, New York, NY, USA, 2006. ACM.

[58] M. Yildiz, R. Rook, and A. Suleman. Sph with the multiple boundary tangent method. *International journal for numerical methods in engineering*, 77(10):1416–1438, 2009.