Implicit Incompressible SPH

JONAS BERGER, Institut Polytechnique de Paris, France

1 INTRODUCTION

In this report, we will analyze the IISPH method, which is a variant of the famous SPH fluid simulation method. We will detail the math behind its modelization, see how to implement it, and observe the quantitative and qualitative improvements and flaws of this method over SPH. The article used for implementing IISPH is [1]. This report is mostly an analysis of what is written in it.

2 RELATED WORKS

The SPH [2] is the basis of a lot of different fluid simulation methods. It is quite simple to implement and offers a lot of variants for specific cases such as weakly compressible fluids (WCSPH) [3] or simply for improvements, such as Predictive-Corrective Incompressible SPH (PCISPH) [4]. It relies on an equation of states giving expressions of density, pressure, and different forces according to the position and velocity of particles. One of its main issue is that is doesn't take into account the incompressibility approximation of liquid. Some articles addresses this issue by creating incompressible SPH, like [4], [5] or [6], but faces important performance issues due to the number of equations that such restriction implies and the efficiency of the solver. The IISPH article [1] proposes an alternative with a model that does not explicitely constraint the compressibility, but will rather induces particles to converge their density to a standard one. The method proves to be a lot more efficient than the previous ones, and has a satisfying rendering improvment compared to SPH.

3 METHOD MODELIZATION

The fluid mecanic is based on a few general equations, some of which serve as the basis for the expressions of the method. The first one is the continuity equation, derived from the conservation of mass: $\frac{D\rho}{Dt} = -\rho\nabla\cdot\mathbf{v}$. Using the SPH model, we have a detailed expression of the divergence of the velocity: $\nabla\cdot\mathbf{v}_i = -\frac{1}{\rho_i}\sum_j m_j\mathbf{v}_{ij}\nabla W_{ij}$, with W the kernel function and $\mathbf{v}_{ij} = \mathbf{v}_i - \mathbf{v}_j$. We deduce from discretization the following relation between density and velocity:

$$\frac{\rho_i(t+\Delta t) - \rho_i(t)}{\Delta t} = \sum_j m_j\mathbf{v}_{ij}(t+\Delta t)\nabla W_{ij}(t) \tag{1}$$

The second equation is the incompressibility of the fluid, which implies $\frac{d\rho}{dt} = 0$. The main idea of the IISPH method is to put this condition indirectly into the equations by searching a state where $\rho_i(t+\Delta t) = \rho_0$ with ρ_0 the rest density.

We will consider the three forces in action during the process:

Author's address: Jonas BERGER, Institut Polytechnique de Paris, France.

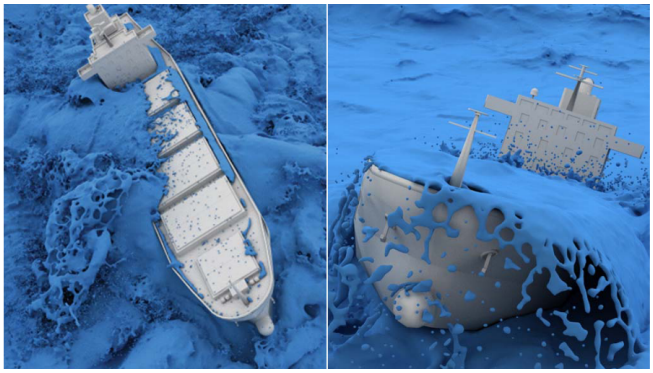

Fig. 1. Simulation of IISPH with up to 30 million particles. The picture is taken from [1].

$$\text{Gravity: } \mathbf{F}_i^g = m_i\mathbf{g} \tag{2}$$

$$\text{Viscosity}^1\ \mathbf{F}_i^v = 2\nu m_i\sum_j \frac{m_j}{\rho_j}\mathbf{v}_{ij}\frac{\mathbf{x}_{ij}\cdot\nabla W_{ij}}{\mathbf{x}_{ij}\cdot\mathbf{x}_{ij} + 0.01h^2} \tag{3}$$

$$\text{Pressure force: } \mathbf{F}_i^p = -m_i\sum_j m_j\left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right)\nabla W_{ij} \tag{4}$$

Given the discretization of Newton's second law

$$\mathbf{v}_i(t+\Delta t) = \mathbf{v}_i(t) + \frac{\Delta t}{m_i}\sum \mathbf{F}_i(t) \tag{5}$$

we now have a complete set of equations of state. The aim here is to mix those equations in order to have a system of equations with the pressures $p_i(t)$ as variables. In order to do this, we will consider gravity and viscosity forces, which do not take pressure into account, and use them into (5) and (1) to express two coefficients who can already be computed: the intermediate velocity and the intermediate density respectively.

$$\mathbf{v}_i^{int} = \mathbf{v}_i(t) + \Delta t\frac{\mathbf{F}_i^g(t) + \mathbf{F}_i^v(t)}{m_i} \tag{6}$$

$$\rho_i^{int} = \rho_i(t) + \Delta t\sum_j m_j\mathbf{v}_{ij}^{int}\nabla W_{ij}(t) \tag{7}$$

We can then deduct from (1) the following equations:

$$\rho_0 - \rho_i^{int} = \Delta t^2\sum_j m_j\left(\frac{\mathbf{F}_i^p(t)}{m_i} - \frac{\mathbf{F}_j^p(t)}{m_j}\right)\nabla W_{ij}(t) \tag{8}$$

In order to make the next equations easier to reap and to compute, we will introduce a few other coefficients:

[1] Here, h is the kernel's smoothing length.

$$\mathbf{d}_{ij} = -\Delta t^2 \frac{m_j}{\rho_j^2} \nabla W_{ij} \text{ for } i \neq j \tag{9}$$

$$\mathbf{d}_{ii} = -\Delta t^2 \sum_j \frac{m_i}{\rho_i^2} \nabla W_{ij} \tag{10}$$

$$a_{ij} = \sum_{k \neq i,j} m_k \left(\mathbf{d}_{ij} - \mathbf{d}_{kj} \right) \cdot \nabla W_{ik} \tag{11}$$

$$\mathbf{c}_i = \sum_{j \neq i} \mathbf{d}_{ij} p_j \tag{12}$$

Now let's see where to use those new tools. First of all, notice that by putting (9) and (10) into (4), we have $\Delta t^2 \frac{\mathbf{F}_i^p}{m_i} = \mathbf{d}_{ii} p_i + \sum_j \mathbf{d}_{ij} p_j$. Then by putting this expression into (8), we have $\rho_0 - \rho_i^{int} = \sum_j m_j \left(\mathbf{d}_{ii} p_i + \sum_{k \neq i} \mathbf{d}_{ik} p_k - \mathbf{d}_{jj} p_j - \sum_{k \neq j} \mathbf{d}_{jk} p_k \right) \cdot \nabla W_{ij}(t)$, which can then be rearranged and simplified into:

$$\rho_0 - \rho_i^{int} = \sum_j a_{ij} p_j \tag{13}$$

Though we finally have a rather simple system of equations in theory, we must address the issue of implementing a solver for such a huge system, and even of storing all those coefficients. Keep in mind the $\mathbf{c}_i$, it will soon be used.

4 RESOLUTION AND IMPLEMENTATION

Luckicy, we won't have to store, or even to compute, all those coefficients; the two matrix of quadratic dimension would represent too much useless computations. The article propose to use the weighted Jacobi method in order to solve (13).

For an equation $A\mathbf{x} = \mathbf{b}$ with A a diagonally dominant matrix, we can decompose $A = D + N$ with D containing only the diagonal composants of A. Then, we define an arbitrary vector $\mathbf{x}^{(0)}$ and the sequence : $\mathbf{x}^{(l+1)} = (1-\omega)\mathbf{x}^{(l)} + \omega D^{-1} \left(b - N\mathbf{x}^{(l)} \right)$, for ω between 0 and 1. Not only does the sequence converge to a vector $\mathbf{x}$, but it is also solution of the initial equation[2].

In our case, even if we cannot prove that A is diagonally dominant, the implementation show that the weighted Jacobi method converge rather quickly (c.f. the result analysis). The equation here is:

$$p_i^{l+1} = (1-\omega) p_i^n + \omega \frac{\rho_0 - \rho_i^{int} - \sum_{j \neq i} a_{ij} p_j^l}{a_{ii}} \tag{14}$$

Note that in the case where a_{ii} is null, then the particle doesn't have any neighbor, and so its pressure must be null too.

For implementation, we can replace that equation with a new one, using the $\mathbf{c}_i$ coefficients.

$$\begin{aligned} p_i^{l+1} =& (1-\omega) p_i^n + \omega \frac{1}{a_{ii}} \Bigg(\rho_0 - \rho_i^{int} \\ & - \sum_j m_j \left(\mathbf{c}_i - \mathbf{c}_j + \mathbf{d}_{ji} p_i^l - \mathbf{d}_{jj} p_j^l \right) \cdot \nabla W_{ij} \Bigg) \end{aligned} \tag{15}$$

[2]The diagonally dominant matrix is not a necessary condition for convergence, but it is a sufficient one, and is generally the reason why we use Jacobi method. Especially, the weighted Jacobi method ensures convergence for a small enough ω.

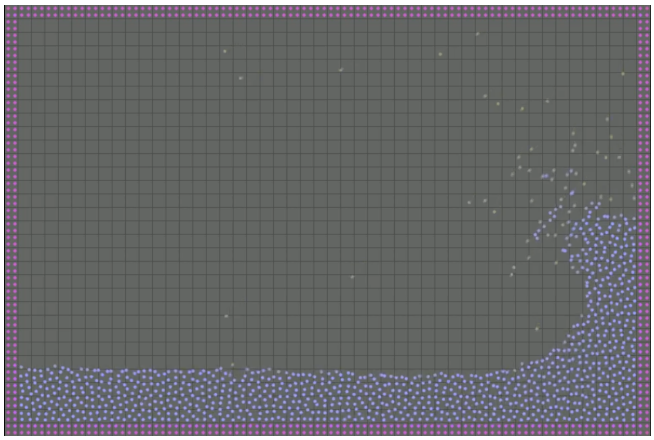

Fig. 2. Simulation of IISPH [1].

With this, the implementation becomes pretty straightforward, though with a lot of coefficients to compute: at each time step, we first compute $\rho_i(t)$, $\mathbf{v}_i^{int}$, ρ_i^{int}, $\mathbf{d}_{ii}$ and a_{ii}. Then, we iterate the sequence of the weighted Jacobi method : we initiate p_i^0, and at each iteration, we compute $\mathbf{c}_i^l$ (notice the exponant, as it now depend on the iteration's pressure) and p_i^{l+1}. After convergence, we can use equations (2) through (5) to compute velocity, and then position of the particles.

Note that in order to avoid errors in the pressure forces, we need to clamp any negative pressure at each iteration to 0. The article mentions that an attractive pressure force might be interpreted as surface tension, but is too important in practice, and therefore must be avoided.

5 OPTIMIZATIONS OF THE METHOD

The article [1] is a bit unclear about the way to decide the number of iterations of the weighted Jacobi method. Their stopping criteria is written $\rho_{avg}^l - \rho_0 \leq \eta$, with eta an arbitrary constant, and with a minimum of two iterations. Though we do not compute any density during the iterations, it can be interpreted as $-\frac{1}{N} \sum_{i,j} a_{ij} p_j \leq \eta$ with N the number of particle.

There are three issues with this criteria. The first one is the amount of operations to compute just for this number. The algorithm specifically avoid computing every coefficient a_{ij}, so a simpler expression is still to be found. The second one is that this criteria ignore the situation where the density is inferior to the rest density ρ_0, which may directly happen at first iteration and not specifically desirable. It can just be addressed by computing the absolute expression. The final issue is that the criteria is unrelated with the convergence of the system of equations. Having a mean density approximately equal to the rest density doesn't mean that the pressures computed are correct; in fact, they might be completely random.

In order to avoid this, it might be more important to focus on the variation of pressure between two iteration. By choosing a norm, we can use a criteria of the form $\left\| \mathbf{p}^{l+1} - \mathbf{p}^l \right\| < \eta$. There are a lot of ways to compute this, for instance with an l_2 norm. We chose one that is quick to compute, that is:

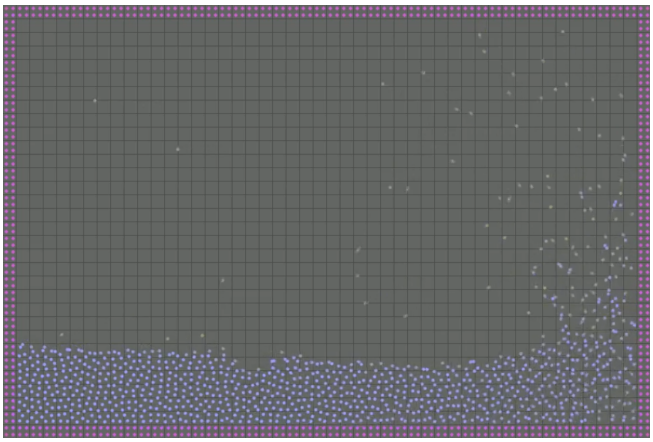

Fig. 3. Implementation of the SPH model [2].

$$\text{For all i, } \left|p_i^{l+1} - p_i^l\right| < \eta p_i^l \tag{16}$$

This criteria observe the relative variation of pressure, and handle well transitions between null and non-null pressure. Especially, it ensures that the final pressures are of the right order of magnitude. The η defines the precision we want on the computation. As a good compromise between computation time and precision, we used $\eta = 1$ in our simulation.

The coefficient ω is a compromise between assured convergence (small value) and fast convergence (value near 1). However, an high ω does not assure a fast computation, because 1) it may not converge, or on wrong values, and 2) it will increase the pressures variations between two iterations. So for a same stopping criteria, an higher ω will require more iterations to reduce this difference. The article choose a value of 0.5 by default, and after a few test, it seems that it is the best choice for reducing the number of iterations. With those numbers, the loop iterates between around 5 and 20 times per time step.

The initialization of $\mathbf{p}^0$ does not have a real impact on the performance of the algorithm, as the pressure computer is roughly the same after a few step. We reused the idea of the article, that is, computing $\mathbf{p}^0 = \alpha\mathbf{p}(t - \Delta t)$. After trying a few different values between 0 and 1, there is no apparent difference, except for $\alpha = 0$ which seems to increase considerably the number of iterations. For the simulation, we simply chose $\alpha = 0.5$.

Finally, as mentioned in the article, there are no specific boundary handling needed for the IISPH method. I used a basic one, using stationary particles as walls and using them in the previous equations but imposing their velocity to always be null.

The article mentioned an implementation of a method from [7], which is rather similar. Basically, for wall particles, the mass is replaced by a density-dependant one : $\Psi_b(\rho_0) = \frac{\rho_0}{\sum_{b'} W_{bb'}}$, and the pressure is null, making the pressure force exerced on a fluid particle:

$$\mathbf{F}^p_{i \leftarrow b} = -m_i \Psi_b(\rho_0) \frac{p_i}{\rho_i^2} \nabla W_{ij} \tag{17}$$

The mass term is basically a variant of the initial one. A second, compromise implementation between the basic model and this one has been tried (cf. fig. 4) where $\mathbf{c}_b$ and p_b are imposed to be null for the wall particles but the mass is the same as fluid particles (a fixed value). The iteration for the weighted Jacobi method thus becomes:

$$\begin{aligned} p_i^{l+1} =& (1-\omega)p_i^n + \omega \frac{1}{a_{ii}} \Bigg(\rho_0 - \rho_i^{int} \\ & - \sum_j m_j \left(\mathbf{c}_i - \mathbf{c}_j + \mathbf{d}_{ji} p_i^l - \mathbf{d}_{jj} p_j^l \right) \cdot \nabla W_{ij} \\ & - \sum_b m_b \mathbf{c}_i \cdot \nabla W_{ib} \Bigg) \end{aligned} \tag{18}$$

6 OBSERVATIONS

The simulations were done using a 12 threads 6 core 2.70GHz Intel I5-11400H with 16Gb or RAM, and adapt a base code from an SPH implementation in a 2D grid. For each of them, a frame is drawn every 10 time steps, and we fixed $\Delta t = 0.5ms$. Note that the algorithm has a linear complexity in the number of particles and is adapted for parallelization.

The first test (cf. fig. 2) was done with a block of 1024 particles and walls composed of 624 stationary particles, for a total of 1648 particles, and as much equations in (13). It took 16 minutes and 32 seconds to compute 3000 frames, which means that the algorithm computes one time step every 33ms. At 60fps, one second of video is computed in 19.86s.

For comparison, a similar test was done using the classic SPH method (cf. fig. 3). It took 2 minutes and 28 seconds for the same number of time steps, which represents 4.9ms to compute one time step, 2.96s to compute one second of video. The IISPH algorithm thus is about 6.7 time longer. However, those numbers depend on the context of the simulation, as the weighted Jacobi method have a variable number of iterations, depending on the complexity of the particles arrangement (as mentioned before, rounghly between 5 and 20 times).

On a qualitative point of view, the IISPH method greatly improves the behaviour of the liquid. It draw smoother shapes, due to a kind of adhesion effect between the particles, created by the implicit incompressibility. Especially, we can notice a wave effect, with ebb and flow and small swirls. On resting positions too, a kind of surface tension appears, and the fluid seems to keep a regular structure between the particles. We invite you to see those improvement yourself by watching the simulation videos.

As mentionned before, another implementation was done, with a more sophisticated boundary handling. The result can be seen on fig. 4.

The qualitative result is not really obvious, though we may notice that the particles stay closer to the walls on resting positions, and avoid some small bouncing when they slide on the ground.

In theory, this method should be used to speed up the weighted Jacobi method, as the coefficients $\mathbf{v}_i^{int}$, ρ_i^{int}, $\mathbf{d}_{ii}$, a_{ii}, $\mathbf{c}_i^l$ and p_i^l do not need to be computed for wall particles. In our case, only 1024 particles out of 1648 need to be computed in the method. But after experimentation, it seems that the optimization is not that significant: the simulation is about 4% faster. We suppose that it is due to

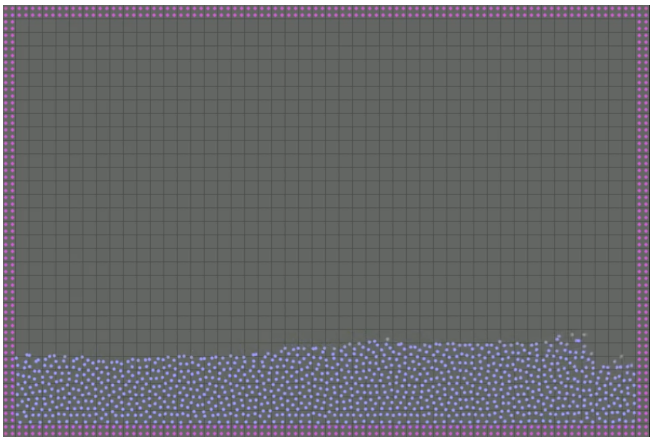

Fig. 4. Implementation of the new boundary model.

the Jacobi method taking more step to converge. It is still a welcome improvement nevertheless.

7 CONCLUSION

While being notably slower than the original SPH method, the IISPH greatly improves the visual of the fluid simulation, especially for a liquid. It does not fix the density to an arbitrary value, but incites the particles to go in a configuration where the density is equal to this value. The Jacobi method is not the quickest to resolve a system of equations, but is especially adapted for a parallel computation, with a multi-threaded process, making this model faster than a majority of other incompressible SPH like PCISPH. It has a good balance between efficiency and quality of the simulation.

REFERENCES

[1] M. Ihmsen, J. Cornelis, B. Solenthaler, C. Horvath and M. Teschner, 2014. Implicit incompressible SPH. IEEE Transactions on Visualization and Computer Graphics 20, 426– 435.

[2] J. Monaghan, "Smoothed Particle Hydrodynamics," Ann. Rev. Astronomy and Astrophysics, vol. 30, pp. 543-574, 1992.

[3] M. Becker and M. Teschner, "Weakly Compressible SPH for Free Surface Flows," Proc. ACM SIGGRAPH/Eurographics Symp. Com- puter Animation, pp. 209-217, 2007.

[4] B. Solenthaler and R. Pajarola, "Predictive-Corrective Incompres- sible SPH," ACM Trans. Graphics, vol. 28, pp. 40:1-40:6, 2009.

[5] S. Shao and 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

[6] M. Ellero, M. Serrano, and P. Espanol, "Incompressible Smoothed Particle Hydrodynamics," J. Computational Physics, vol. 226, no. 1, pp. 1731-1752, 2007.

[7] N. Akinci, M. Ihmsen, B. Solenthaler, G. Akinci, and M. Teschner, "Versatile Rigid-Fluid Coupling for Incompressible SPH," ACM Trans. Graphics, vol. 30, no. 4, pp. 72:1-72:8, 2012.