Université de Liège

MATH0471-A-A : Multiphysics integrated computational project

Smoothed-particle hydrodynamics: microfluidics applications

KOLLASCH Killian s202793
SANTORO Luca s201807

Professor: C. Geuzaine
Professor: R. BOMAN

Academic year : 2023-2024

Contents

List of Figures

List of Tables

I Introduction

Smoothed Particle Hydrodynamics (SPH) is a powerful computational method used to simulate fluid dynamics and other continuum mechanics problems. In SPH, the fluid domain is discretized into particles, each carrying properties such as mass, density, and velocity. These particles interact with each other through a smoothing kernel function, which computes weighted contributions from neighbouring particles.

It offers several advantages, including its meshfree nature, ability to handle large deformations, and suitability for complex geometries. It finds applications in various fields, from astrophysics and oceanography to engineering simulations like fluid flow and impact dynamics. Understanding the term "Smoothed Particle Hydrodynamics" entails dissecting its components:

- **Particle**: At the core of SPH lies the concept of particles, serving as the primary entities within the system. Interactions among these particles define the behaviour of the fluid or matter being simulated.
- **Smoothed**: This refers to the smoothing or weighting function employed to account for the influence of neighbouring particles on a given one, ensuring the continuity and coherence of the simulation.
- **Hydrodynamics**: SPH finds its most natural application in the realm of hydrodynamics, where its particle-based approach proves particularly effective in simulating fluid flow and related phenomena.

I.1 Motivation

In the frame of our Multiphysics integrated computational project, it has been proposed to improve the SPH method implemented in the thesis of Louis Goffin [3] by specialize it for microfluidic applications.

I.2 Basics of the method

The SPH method is said to be Lagrangian, meshfree and particle-based. All these three concepts are presented below:

Lagrangian Grid

A Lagrangian grid is a grid fixed on the material. This means that as the object under study deforms, the grid also deforms accordingly. Refer to Fig.1 for a visual representation of a Lagrangian grid. This technique finds extensive use in the finite element method. However, when the material experiences significant deformation, the grid itself becomes highly distorted. This distortion compromises the accuracy of calculations. To mitigate this, it becomes necessary to remesh when the grid distortion exceeds a certain threshold. However, this remeshing process increases the computational time required for the analysis.

Meshfree

SPH is considered mesh free because it does not require a mesh for field variable interpolation. However, SPH is not strictly mesh free; initialization typically involves arranging particles using a mesh, and mesh data may be required when searching for neighbouring particles.

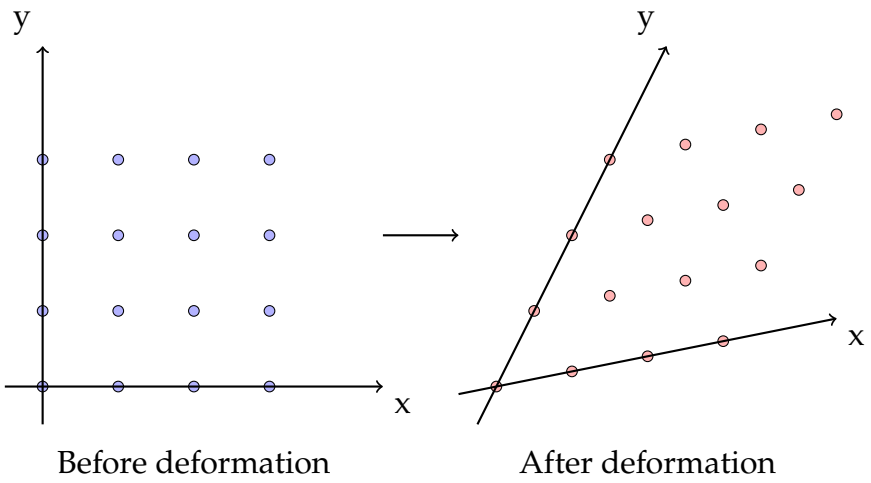

Figure 1: Lagrangian grid.

Particle-based

The matter or fluid is represented by a collection of particles. These particles interact with each other based on their distances, forming the foundation of the method's computational framework.

- **Particle**: At the core of SPH lies the concept of particles, serving as the primary entities within the system. Interactions among these particles define the behaviour of the fluid or matter being simulated.
- **Smoothed**: This refers to the smoothing or weighting function employed to account for the influence of neighbouring particles on a given one, ensuring the continuity and coherence of the simulation.
- **Hydrodynamics**: SPH finds its most natural application in the realm of hydrodynamics, where its particle-based approach proves particularly effective in simulating fluid flow and related phenomena.

Understanding these fundamental concepts lays a solid foundation for developing deeper understandings of SPH simulations and their applications in various fields of science and engineering.

II SPH algorithm

II.1 Integral representation of a function and its derivatives

The main idea behind SPH mesh-free method is to use the integral representation of a function as follows:

$$f(\boldsymbol{x}) = \int_{\Omega} f(\boldsymbol{x}')\delta(\boldsymbol{x} - \boldsymbol{x}')d\boldsymbol{x}', \tag{II.1}$$

where Ω is the domain of integration and δ is the Dirac function defined as follows:

$$\delta(\boldsymbol{x} - \boldsymbol{x}') = \begin{cases} \infty & \text{if } \boldsymbol{x} = \boldsymbol{x}', \\ 0 & \text{if } \boldsymbol{x} \neq \boldsymbol{x}', \end{cases} \tag{II.2}$$

This representation holds true for any values of $\boldsymbol{x}'$, yet it cannot be directly applied. The reason being, the entirety of the physical domain contributes to the value of any function at any point within the space. To address this limitation, the Dirac function is approximated by

a normalized function, referred to as the kernel function $W(\boldsymbol{x}-\boldsymbol{x'},h)$ (discussed in Sec.II.2). This function depends on the distance between the point where the value is to be computed and a smoothing length denoted as h.

The smoothing length signifies the range at which particles influence each other and remains constant throughout our Multiphysics integrated computational project.

Consequently, a function f and its derivative assessed at a specific point $\boldsymbol{x}$ are approximated using a kernel function in the following manner:

$$f(\boldsymbol{x}) \approx \int_{\Omega} f(\boldsymbol{x'})W(\boldsymbol{x}-\boldsymbol{x'},h)d\boldsymbol{x'}, \tag{II.3}$$

$$\nabla.f(\boldsymbol{x}) \approx -\int_{\Omega} f(\boldsymbol{x'}).\nabla W(\boldsymbol{x}-\boldsymbol{x'},h)d\boldsymbol{x'}. \tag{II.4}$$

II.2 Smoothing function

In order to respect the integral representation, the associated trial function (here the smoothing function) has to be defined on a compact support such that this function vanishes at the boundaries. The smoothing function is also called *kernel function*. Since the kernel function has to be defined on a compact support, it implies that the support domain $2\kappa h$ is contained in the problem domain, ensuring the validity of both Eq.(II.3) and Eq.(II.4)

The kernel used in all our simulations is the *cubic spline* which is defined as follows:

$$W(r) = \begin{cases} \alpha\left(1-\left(\frac{r}{h}\right)^2+0.5\left(\frac{r}{h}\right)^3\right) & \text{if } 0<\frac{r}{h}<1, \\ \frac{1}{6}\left(2-\frac{r}{h}\right)^3 & \text{if } 1<\frac{r}{h}<2, \\ 0 & \text{otherwise}, \end{cases} \tag{II.5}$$

where r is the distance between the particle i and its neighbour j and α is a constant of normalisation.

The scaling α value depends on both of the spatial dimension and the kernel type. For instance, using the cubic spline, the constant can take different values:

$$\alpha = \begin{cases} \frac{1}{h} & \text{(1D)}\ , \\ \frac{15}{7\pi h^2} & \text{(2D)}\ , \\ \frac{3}{2\pi h^3} & \text{(3D)}\ . \end{cases} \tag{II.6}$$

Indeed, this kernel was selected for its accuracy within the domain and its relatively minor error near the boundaries. The kernel and its derivative vanish at the upper boundary and also at $r=0$. Both are illustrated in Fig.2.

It can be observed that the smoothing length h, within α, is raised to a power corresponding to the increasing dimensional domain. This adjustment is essential to maintain the independence of the kernel function from the selected smoothing length h (and thus spacing s). This observation applies to any kernel used in further sections (Sec.IV.1 and Sec.IV.2). To facilitate the use of the linked list algorithm for determining the neighbouring support of all particles (discussed in Sec.II.5), the smoothing length h must remain constant.

II.3 Spatial discretization

The physical domain undergoes discretization through the use of material points or particles. These particles possess constant mass and are propelled by the local flow velocity at each time-step while carrying the fluid's physical properties. Consequently, Eq.II.3 and Eq.II.4 are discretized in the following manner:

$$f(\boldsymbol{x_i}) = \sum_{j=1}^{N} \frac{m_j}{\rho_j} f(\boldsymbol{x_j}) W_{ij}, \tag{II.7}$$

$$\nabla . f(\boldsymbol{x}) = -\sum_{j=1}^{N} \frac{m_j}{\rho_j} f(\boldsymbol{x_j}) . \nabla W_{ij}, \tag{II.8}$$

where i denotes the particle undergoing discretization of the function, j denotes all particles within the support domain of i (including particle i itself). m_j and ρ_j represent respectively the mass and density of the particle j, and $W_{ij} = W(\boldsymbol{x_i} - \boldsymbol{x_j}, h)$.

The impact of both mass and density of each particle is crucial within the SPH algorithm. Particularly, the mass of each particle is determined prior to simulation initiation via the total volume occupied by all particles. This necessitates the prior knowledge of this total volume, achieved by establishing a set of regularly spaced particles in a parallelepiped rectangle configuration, such that the total volume aligns with the dimensions of this rectangle as depicted in Fig.3.

In a fixed Cartesian system $\boldsymbol{e_x}, \boldsymbol{e_y}, \boldsymbol{e_z}$, where particles are evenly spaced, the following relationships hold:

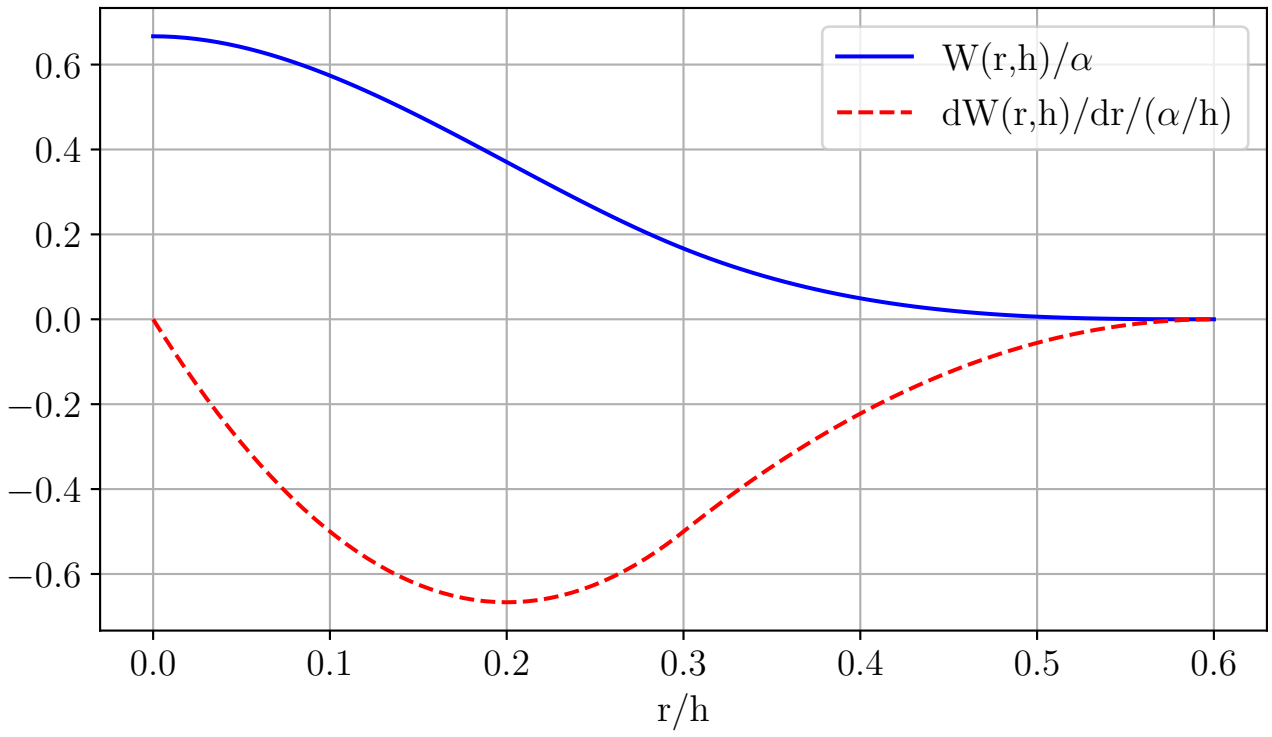

Figure 2: Cubic spline function and its derivative on the support domain $2\kappa h$ where (κ, h, s) = (2, 1.2s, 0.25).

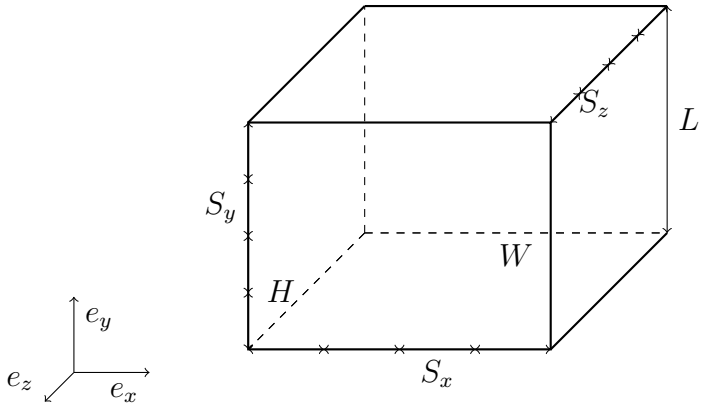

Figure 3: Mass particles initialisation.

$$V_{\text{tot}} = W \times L \times H = N_x S_x \times N_y S_y \times N_z S_z = \frac{M_{\text{tot}}}{\rho} = \frac{\sum_{i=0}^{N} m_i}{\rho}. \tag{II.9}$$

Here, W, L, and H represent the length in $\boldsymbol{e_x}$, $\boldsymbol{e_y}$ and $\boldsymbol{e_z}$ direction of the rectangle. $N_{x,y,z}$ and $S_{x,y,z}$ denote the number and spacing of particles in the specified direction.

Assuming uniformity among all particles within a single rectangle, the mass of an individual particle is determined by:

$$m_i = \rho_i \times S_x S_y S_z, \tag{II.10}$$

in this study $S_x = S_y = S_z = S$.

Consequently, each particle's volume equals the total volume of the rectangle divided by the number of particles contained within it.

Furthermore, density is initially determined based on the selected state equation (as discussed in Sec.II.8).

II.4 Time discretization

Time discretization stands out as a crucial aspect of the SPH algorithm. While other elements of the method mainly impact the accuracy of particle variables, inaccuracies in time discretization could potentially disrupt the entire simulation. The thesis of Louis [3] employed two finite difference schemes : the Euler explicit and the second-order Runge-Kutta methods.

Euler explicit

The Euler explicit scheme emerges as an initial choice due to its simplicity and relatively low computational cost for function integration. It is defined as:

$$y_{i+1}(x_i) = y_i(x_i) + \Delta x \frac{dy}{dx}(x_i). \tag{II.11}$$

Runge-Kutta

The second-order Runge-Kutta method (RK22) necessitates two evaluations of the given function to obtain the desired value:

$$\begin{cases} y_{i+1}(x_i) = y_i(x_i) + \Delta x \left[(1-\theta)k_1 + \theta k_2\right], \\ k_1 = \dfrac{dy}{dx}(x_i, y_i), \\ k_2 = \dfrac{dy}{dx}(x_i + \dfrac{1}{2}\Delta x, y_i + \dfrac{1}{2}\Delta x k_1). \end{cases} \tag{II.12}$$

The RK22 algorithm essentially comprises a weighted sum of two Euler explicit schemes computed at two distinct locations. The parameter θ determines the weighting of $k_{1,2}$, representing the slope of the function at the beginning and end of the step. A value of $\theta = 1/2$ corresponds to the midpoint method.

Comparison of the two methods

The difference between the two ones can be highlighted by an example that mimic the behaviour of the Navier-Stokes equations. Let us use an arbitrary set of two non-linear ordinary differential equations (ODE) of order one:

$$\begin{cases} \dfrac{dx}{dt} &= xyt, \\ \dfrac{dy}{dt} &= -2x + \sin(y). \end{cases} \tag{II.13}$$

As depicted in Fig.4, the choice of the timestep is crucial to compare the two methods. Indeed, as the timestep decreases, the difference between the two methods decreases accordingly. This behaviour is simply the application of Taylor series.

One chooses to employ the Euler explicit scheme, considering that both methods yield similar results as the timestep decreases. A more precise evaluation of the accuracy of the Euler explicit method involves computing the relative error compared to the RK22 method (with RK22 being considered more accurate and used as a reference):

$$\text{error} = \frac{|(x,y)_{RK22} - (x,y)_{Euler}|}{(x,y)_{RK22}}. \tag{II.14}$$

As can be seen in Fig. 5, the relative error for numerous timestep exponentially decreases such that the error is acceptable even for a relative small number of steps.

II.5 Neighbours search

To assess the neighbours of each particle may be tedious while trying to get efficient CPU performances.

Naive algorithm

The first method, so-called *naive algorithm*, is the simplest one and relatively straightforward. Indeed, given a particle i, the algorithm consists in iterating over all other particles and keep only particles which are on the support domain (thus at a distance smaller than κh):

Obviously, this method is not optimal at all since:

$$CPU_{\text{naive algo}} \sim \mathcal{O}(n^2), \tag{II.15}$$

where n is a multiple computational operation. This algorithm will thus not be used later.

Algorithm 1 Naive algorithm

```
for i = 1, 2, ..., N do
    for j = 0, ..., N do
        Compute distance r_ij between particle i and j
        if r_ij^2 ≤ (κh)^2 then
            Add particle j to neighbours of i
        end if
    end for
end for
```

Linked-list algorithm

Another more efficient method is the so-called *linked-list algorithm*. The primary concept of this approach, as outlined in L. Goffin's thesis, involves enclosing all particles within cells (subdomains) and exclusively examining the neighbours of a given particle in adjacent cells. These cells have dimensions equal to or greater than the neighbouring radius κh. Currently, κ is set to 1 and h is set to 1.2s. The choice of κ will be either 1, 2, or 3 depending on the kernel selected for computation. Also, the *linked-list* algorithm requires (for efficient performances) that the smoothing length h is kept as constant between each time step.], its efficiency is estimated to:

$$CPU_{\text{linked list}} \sim \mathcal{O}(n), \tag{II.16}$$

if the number of particles per cell is low enough.

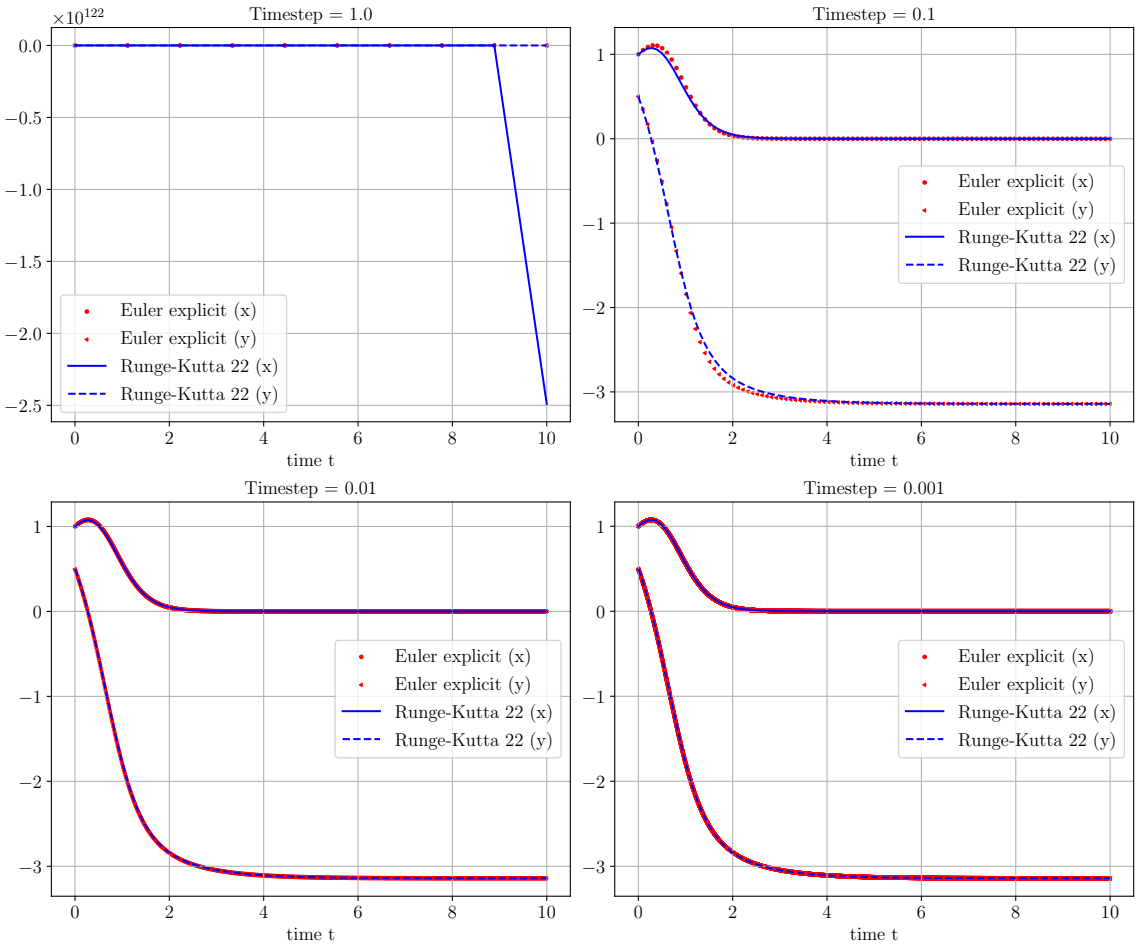

Figure 4: Comparison of the Euler explicit and RK22 method with different timestep for a dummy set of non-linear ODE's of order one.

One first needs to determine the total number of cells in each direction x, y, z of the domain (assuming a 3D rectangular domain). Given $L[0], L[1], L[2]$ (denoted L_0, L_1, L_2 lately) being respectively the lengths of the domain in the x, y, z direction, the number of cells in a prescribed direction is given by:

$$N_{x,y,z} = \left\lfloor \frac{L_{0,1,2}}{\kappa h} \right\rfloor . \tag{II.17}$$

For instance, if $L_1 = 1.0$ and $s = 0.25$:

$$N_x = \left\lfloor \frac{1}{1 \times 1.2 \times 0.25} \right\rfloor = 3, \tag{II.18}$$

hence, each cell will be referenced by three indexes (i, j, k).

Once the total number of cells in each direction is set, one iterates over all particles in order to assign, for each particles, their associated cell (e.g. where they lye). The relationship between a position (x, y, z) of a particle and its associated cell is:

$$\text{index}_{i,j,k} = \frac{\text{position}_{x,y,z}}{L_{x,y,z}/N_{x,y,z}}, \tag{II.19}$$

where i, j, k are respectively associated to x, y, z.

But one needs to pay attention if an edging cell is encountered. This is done by the following coding lines:

It can be noticed that particles outside the domain are skipped such that they have no influence any more on the particle inside the domain.

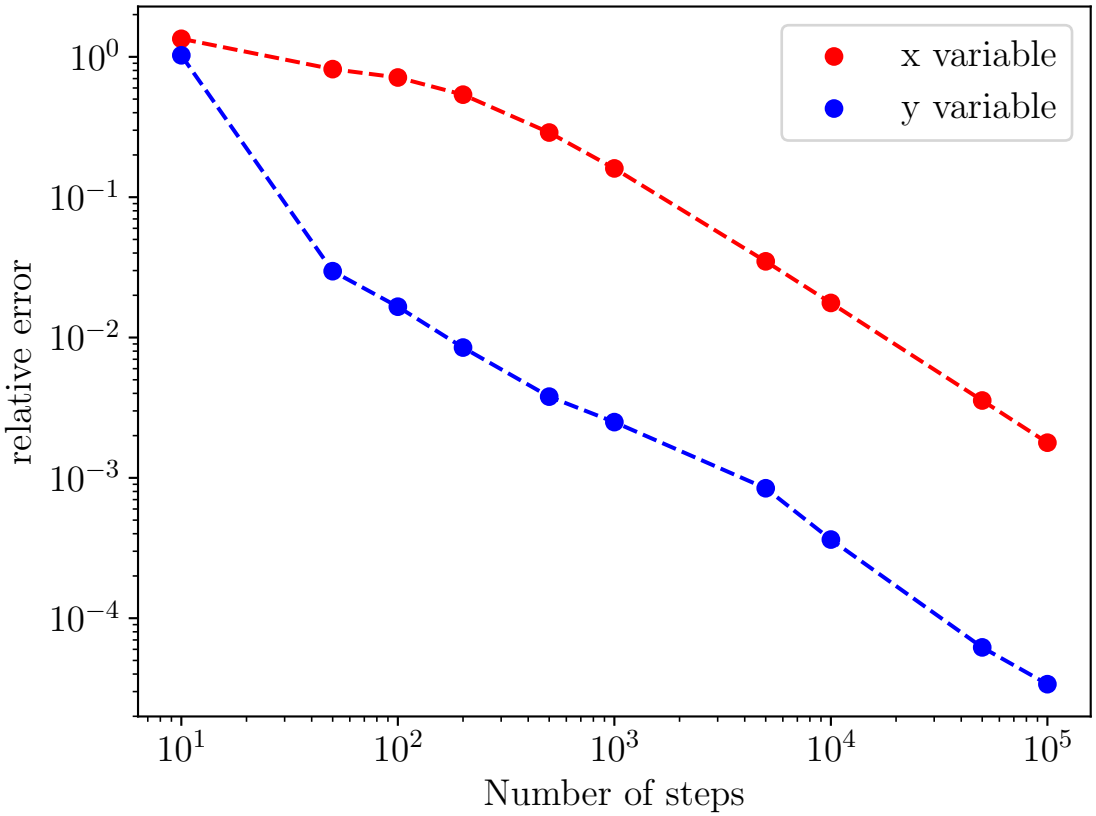

Figure 5: Relative error of the Euler explicit method with respect to the RK22 method.

Algorithm 2 Linked list (1/2) : cell sorting

for $n = 0, 1, \ldots, N$ **do**
  $i \leftarrow \frac{x}{Lx/Nx}$
  $j \leftarrow \frac{y}{Ly/Ny}$
  $k \leftarrow \frac{z}{Lz/Nz}$
  if $(i, j, k) < 0$ or $(i, j, k) > N_{x,y,z}$ **then**
    Skip particle n (outside the domain)
  end if
  if $(i, j, k) = N_{x,y,z}$ **then**
    reduce index (i,j or k) by 1 (particle at boundaries)
  end if
  Add particle n to corresponding cell
end for

Next, one can accurately iterate through neighbouring cells and then iterate through all their associated particles to identify the corresponding neighbours. Subsequently, when a particle is encountered in a neighbouring cell (or in the same cell as the particle i), their relative distance r_{ij} is calculated and compared to κh. If $r_{ij}^2 \leq (\kappa h)^2$, the particle j is added to the neighbour list of particle i:

Algorithm 3 Linked list (1/2) : find neighbours

for $k_{adjacent} = k - 1, k, k + 1$ **do**
  for $j_{adjacent} = j - 1, j, j + 1$ **do**
    for $i_{adjacent} = i - 1, i + 1$ **do**
      cell $\leftarrow$ cell_matrix$[i + j \cdot Nx + k \cdot Nx \cdot Ny]$
      for $idx = 0$ to $size\_cell - 1$ **do**
        $idx\_cell \leftarrow$ cell$[idx]$
        if $idx\_cell \neq n$ **then**
          Compute r_{ij}^2
          if $r_{ij}^2 \leq (\kappa h)^2$ **then**
            $neighbours[n] \leftarrow idx\_cell$
          end if
        end if
      end for
    end for
  end for
end for

Also, the maximum number of neighbours for each particle is set to 100 to avoid dynamic memory allocation. Also, the *type* variable in the code simply indicates if a moving or fixed particle in encountered.

Comparison of the two methods

It is clear that the *linked-list* method is faster than the *naive* method. To determine the point at which the naive method becomes impractical, one can measure the average time taken by both algorithms to sort all the particles over a hundred timesteps for varying numbers of particles. For the comparison, one uses the *linked_list_comparison.json* file has been used.

As shown in Fig. 6, the *linked-list* algorithm becomes significantly more advantageous when the number of particles exceeds a thousand. Also, the data relative to this figure have been obtained using the OpenMP version of the code.

II.6 Navier-Stokes equations

The SPH method, being Lagrangian and mesh-free, uses governing equations in the Lagrangian frame. These governing equations are the conserving mass and momentum, expressed as:

$$\frac{D\rho}{Dt} = \boldsymbol{u}.\nabla\rho - \nabla.(\rho\boldsymbol{u}), \tag{II.20}$$

$$\frac{D\boldsymbol{u}}{Dt} = -\frac{\nabla p}{\rho} + \frac{\boldsymbol{F}}{\rho} = -\left(\nabla\left(\frac{p}{\rho}\right) + \frac{p}{\rho^2}\nabla\rho\right) + \frac{\boldsymbol{F}}{\rho}, \tag{II.21}$$

where ρ is the density, D/Dt the material derivative, $\boldsymbol{u}$ the velocity field, p the local static pressure and $\boldsymbol{F}$ the volume body forces. In this study, only body and friction forces are taken into account.

Using the spatial discretization introduced in Sec.II.3 allows us to rewrite these two equations in the following manner:

$$\frac{D\rho_i}{Dt} = \sum_{j=1}^{N} m_j \boldsymbol{u_{ij}} \cdot \boldsymbol{\nabla} W_{ij}, \tag{II.22}$$

$$\frac{D\boldsymbol{u_i}}{Dt} = -\sum_{j=1}^{N} m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} + \Pi_{ij}\right) \boldsymbol{\nabla} W_{ij} + \boldsymbol{F}, \tag{II.23}$$

where $\boldsymbol{u}_{ij}$ is the relative velocity between particle i and j, Π_{ij} is the artificial viscosity introduced in the following.

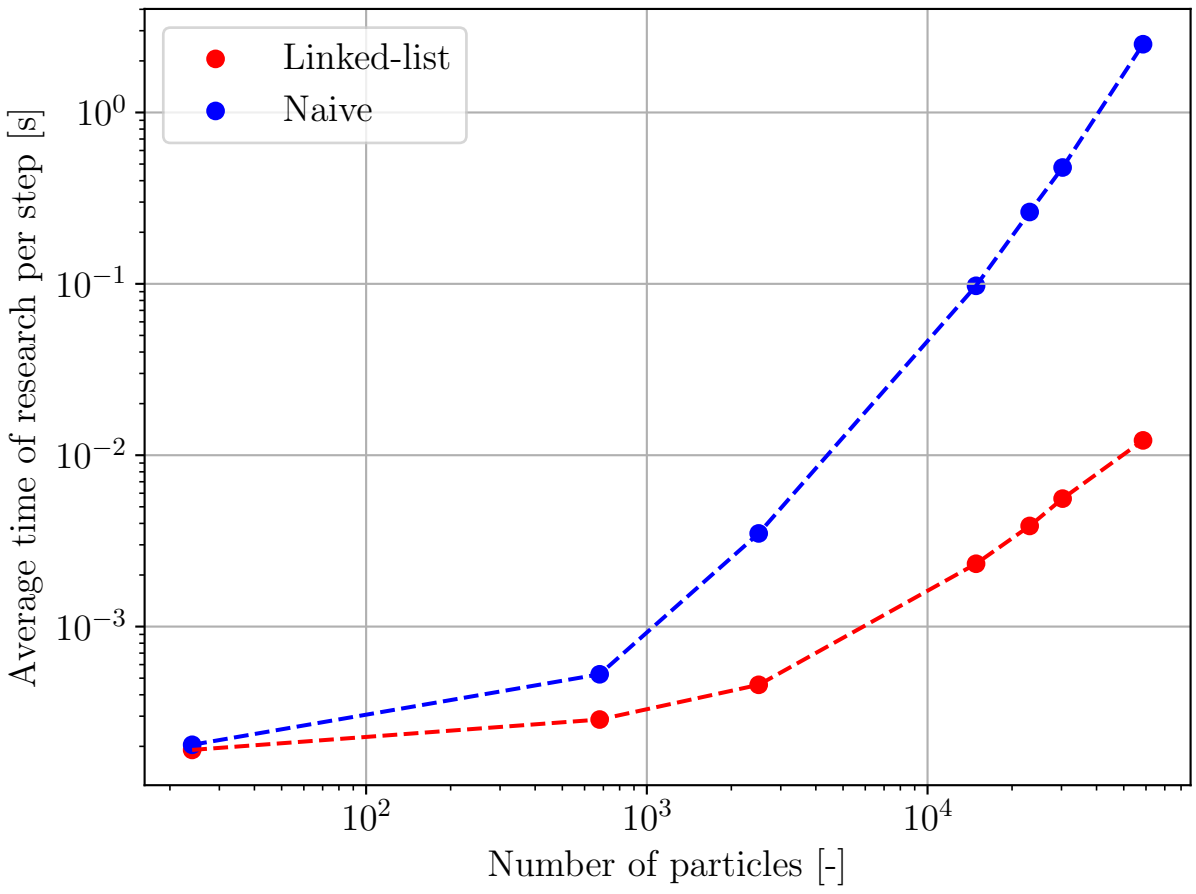

Figure 6: Comparison of the Linked-list and Naive sorting algorithm.

Artificial viscosity

The friction forces will be considered in this work with an artificial viscosity. As explained in L.Goffin master's thesis.This term has two main purposes:

1. to create a viscosity term in order to reproduce the friction that occurs in a real fluid

2. to avoid numerical instabilities when the particles are moving away from each other. This artificial viscosity has been introduced by [Monaghan, 1992]. Its expression is:

$$\Pi_{ab} = \begin{cases} \dfrac{\alpha \bar{c_{ij}} \mu_{ij} + \beta\, \mu_{ij}^2}{\bar{\rho_{ij}}} & \text{for } \boldsymbol{u_{ij}} \cdot (\boldsymbol{x_i} - \boldsymbol{x_j}) < 0, \\ 0 & \text{for } \boldsymbol{u_{ij}} \cdot (\boldsymbol{x_i} - \boldsymbol{x_j}) \geq 0, \end{cases} \tag{II.24}$$

with $\bar{\rho_{ij}}$ and $\bar{c_{ij}}$ being the mean density and speed of sound between particle i and j. α and β are parameters given by the user and μ_{ij} has the following formula:

$$\mu_{ij} = \frac{h \boldsymbol{u_{ij}} \cdot (\boldsymbol{x_i} - \boldsymbol{x_j})}{(\boldsymbol{x_i} - \boldsymbol{x_j})^2 + \eta^2}, \tag{II.25}$$

with $\eta^2 = 0.01\ h^2$.

Volume body forces

The two body forces taken into account in this study are the gravity force and the surface tensions.

The gravity is straightforward to implement, such that:

$$\boldsymbol{F_{gravity}} = -g \boldsymbol{e_z}, \tag{II.26}$$

where $g = 9.81$ [m/s^2].

Nevertheless, the surface tensions are much harder to handle due to the fact that the outside boundaries of the fluid are no trivially known (discussed in Sec.IV).

II.7 Ghost particles

In order to extract physical variables from the simulations, *ghost particles* are introduced. These ghost particles have neighbours, but they are not detected by other fluid or fixed particles.

By doing so, one can apply onto them the *summation density* method which is nothing else than Eq.(II.7). This method ensures exact mass conservation [4] but offers less stability and accuracy if scenarios involve high-velocity free surface flows. Nevertheless, since *ghost particles* do not have any influence on the other particles (by definition), the *summation density* method can be applied to them without any problem.

II.8 State equation

The relation between the pressure is everything but trivial and has thus to be discussed. While lots of research have been performed to assess accurate relation between ρ and p, one has simply used two well known state equations, being the *ideal gas law* and *quasi incompressible fluid*. Also, as explained in Sec.II.3, the initialization of the density depends on these state equations.

Ideal gas law

The *ideal gas law* reads:

$$p = \frac{RT}{M}\left(\frac{\rho}{\rho_0} - 1\right), \tag{II.27}$$

where p [N/m^2] is the absolute pressure, ρ [kg/m^3] the density, R = 8.314 [J/(K mol)] the ideal gas constant, T [K] the absolute temperature (= 293.15 [K] and kept constant) and M [kg/mol] is the molar mass.

The key aspect of this law is that it implies a linear relationship between pressure and density, allowing density to increase significantly under high pressure conditions. This equation proves particularly valuable for examining highly compressible fluids like air. However, this research focuses on water, renowned for its quasi-incompressible nature. Therefore, alternative models need to be explored.

Another vital parameter in the SPH method is the speed of sound, denoted as c, for each particle. This value dictates the speed at which information propagates among particles, thus influencing the time integration process (as discussed in Sec.II.9). The speed of sound is determined as follows:

$$c_i = c_0, \tag{II.28}$$

where $i \in \{0, 1, ..., N\}$ is the particle index and c_0 a constant speed of sound.

Here is the evolution of the pressure and speed of sound:

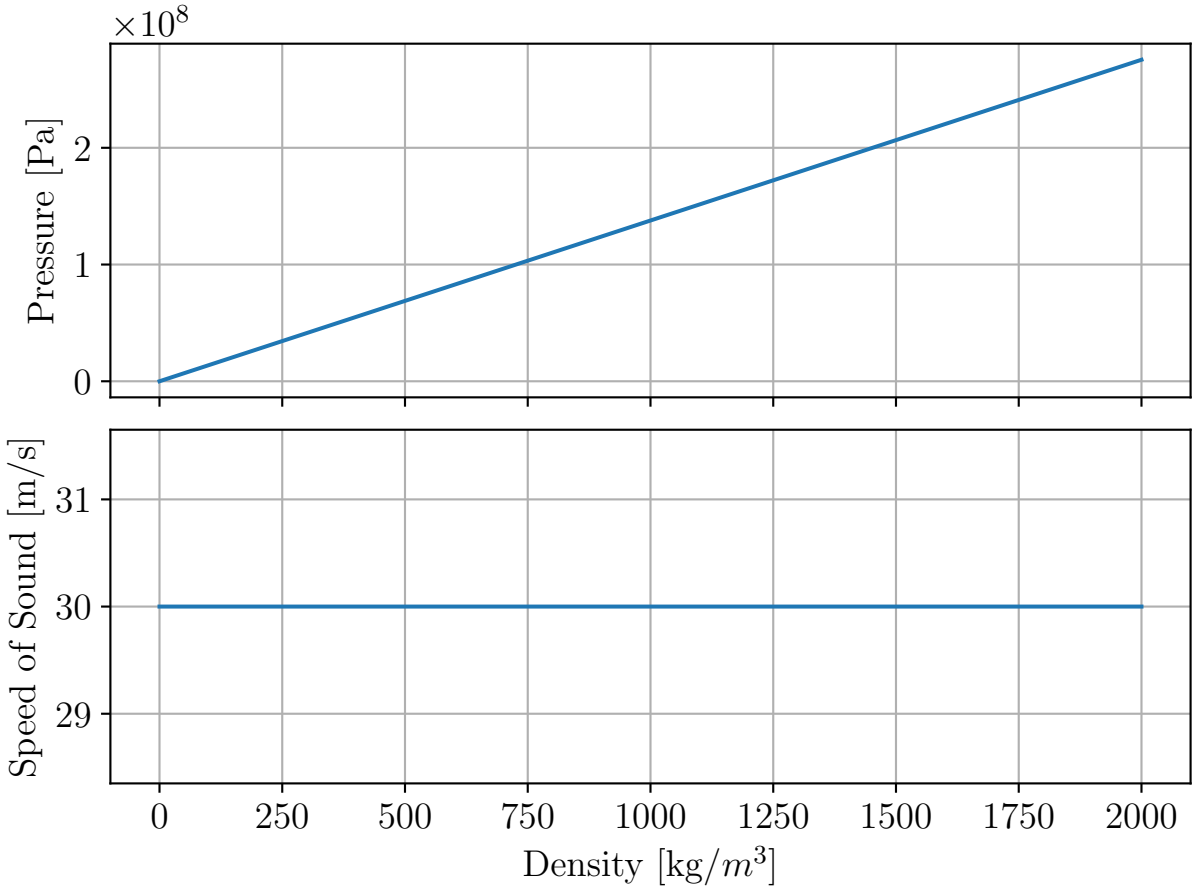

Figure 7: Ideal gas law with c_0 = 30 [m/s], ρ_0 = 1000 [kg/m^3], R = 8.314 [J/kg.mol], T = 298.15 [K], M = 18e-3 [kg/mol].

Finally, the initialization of the density given a hydrostatic pressure for the *ideal gas law* reads:

$$\rho = \rho_0 \left(1 + \frac{M}{RT}\rho_0 g h\right), \tag{II.29}$$

Quasi incompressible fluid

The *quasi compressible* equation of state used in this study is the one used by Monaghan [5] as follows:

$$p_i = B\left(\frac{\rho_i}{\rho_0}\right)^{\frac{1}{\gamma}}, \tag{II.30}$$

where B is a constant parameter which decides the maximum density fluctuation or compressibility, ρ_0 is the reference density, and γ is the heat capacity ration equal to 7 in this study.

B is obtained from:

$$B = \frac{c_0^2 \rho_0}{\gamma}, \tag{II.31}$$

in which c_0 is the speed of sound at reference density and is usually set to 10 times the maximum flow velocity in order to limit the compressibility of the flow.

$$c = c_0 \left(\frac{\rho}{\rho_0}\right)^{\gamma-1}, \tag{II.32}$$

for a *quasi-incompressible* fluid.

Finally, the initialization of the density given a hydrostatic pressure reads:

$$\rho = \rho_0 \left(1 + \frac{1}{B}\rho_0 g h\right)^{1/\gamma}, \tag{II.33}$$

Here is the evolution of the pressure and speed of sound:

Determination of density

To derive the pressure field from the equation of state, it is necessary to initially compute the density of each particle. Density can be determined either through the discretized Eq.(II.20) or by utilizing the SPH representation of density from Eq.(II.7).

As discussed in Sec.II.7, the *summation density* method ensures exact mass conservation, whereas the *continuity density* method generally struggles to conserve mass [5]. Conversely, the *continuity density* method tends to offer greater stability and accuracy in scenarios involving high-velocity free surface flows and impacts, such as the Dam break problem. This advantage comes from the underestimation of the density on the free surfaces in the *summation density* method, coming from the truncated support domain of the kernel function.

In this study, the *continuity density* method is used to update the density and velocity of each particle.

II.9 Time integration

Given the discussions raised in Sec. II.3 and Sec.II.6, the time integration (using Euler explicit scheme for simplicity) is performed using the following formula:

$$\rho_{i+1} = \rho_i + \Delta t \frac{D\rho_i}{Dt}, \tag{II.34}$$

$$\boldsymbol{u}_{i+1} = \boldsymbol{u}_i + \Delta t \frac{D\boldsymbol{u}_i}{Dt}, \tag{II.35}$$

where Δt is the time increment between each step.

To ensure stability of the numerical scheme, the time step has to be carefully chosen smaller than a critical value. But one needs to determine this threshold value.

The increment Δx in both Eq.(II.11) and Eq.(II.12) is equal to Δt and its threshold value is assessed thanks to two condition.

Body force condition

The first expression given is relative to the body forces:

$$\Delta t_f = \min_i \left\{ h_i |F_i| \right\}, \tag{II.36}$$

with $a = 1, \ldots, N$, N being the total number of particles.

If other body forces will be added, $|F_i|$ will be modified such that:

$$|F_i| = \sqrt{\sum_a \left(F_i^{(a)} \right)^2}, \tag{II.37}$$

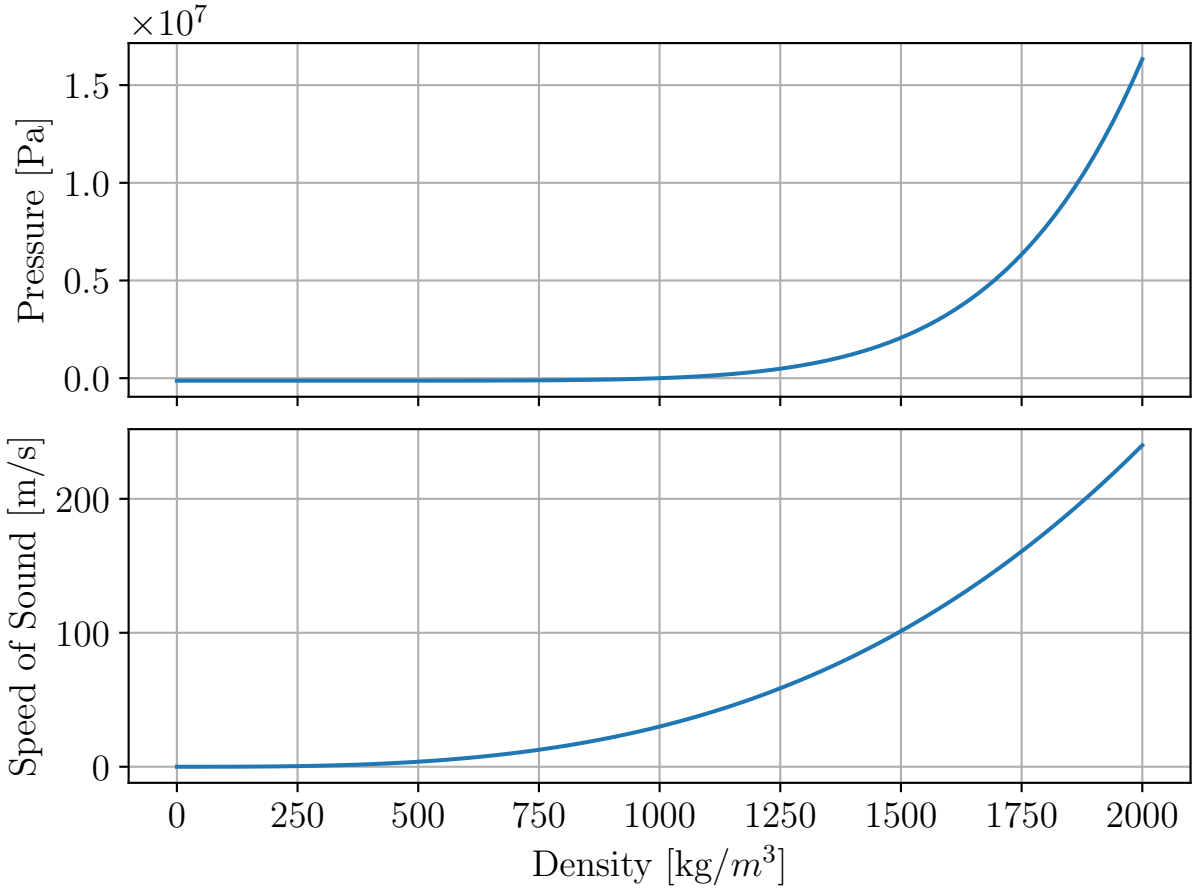

Figure 8: Quasi incompressible fluid with $c_0 = 30$, $\rho_0 = 1000$, $B = 128571.43$, $\gamma = 7$.

where the sum over a represents all the body forces applied to particle i.

Courant condition

The second expression is relative to the Courant condition as well as the viscous diffusion:

$$\Delta t_{cv} = \min_i \left\{ h_i c_i + 0.6(\alpha c_i + \beta \max_j \mu_{ij}) \right\}, \tag{II.38}$$

where the coefficients α and β are the same coefficients from Eq.(II.24) and c_i is the speed of sound at the particle i.

The final time step is obtained by:

$$\Delta t_{qi} = \min\left(0.4\Delta t_f,\, 0.25\Delta t_{cv}\right). \tag{II.39}$$

The coefficients 0.25 and 0.4 come from computer experiments [5].

II.10 Speed-up due to non OOP implementation

The primary drawback of Louis's SPH algorithm lies in the adoption of objects in his Fortran code, inevitably impacting the simulation's CPU time due to the frequent retrieval of information stored in these object variables, especially when employing OpenMP. Thus, the initial improvement involves developing a non-OOP implementation of the SPH algorithm.

To evaluate the speed-up resulting from this new implementation, one compares the CPU performance of the code with and without utilizing the OpenMP speed-up. Assuming that both the OOP and non-OOP implementations yield identical results in serial code execution. However, the parallelism introduced by OpenMP should highlight the influence of the non-OOP implementation.

For the comparison, one uses the *Other/linked_list_comparison.json* file each time, but the spacing s (and thus the number of particles) will vary to assess the speed-up induced by the parallelisation:

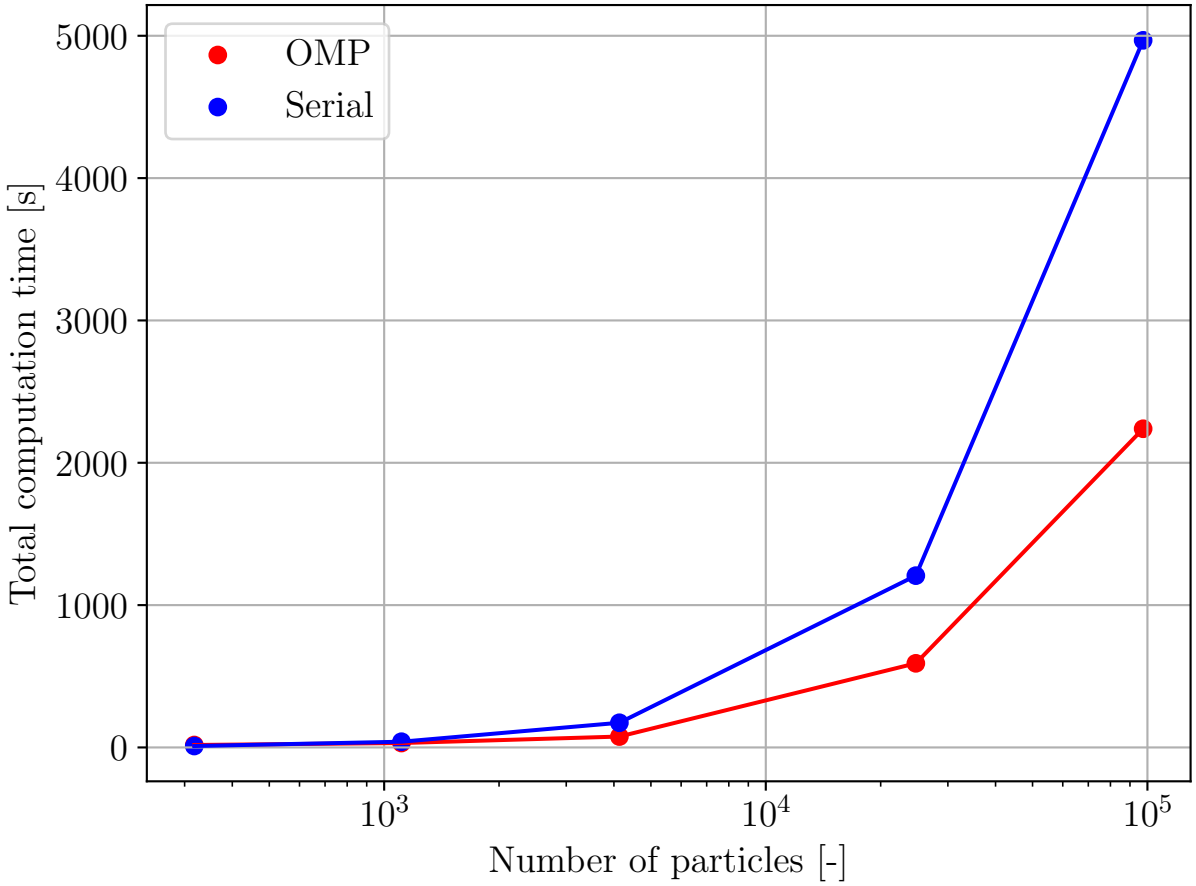

Figure 9: OMP with 16 threads and serial code comparison.

As can be seen on Fig.9, for a lower number of particles than 10^3, the speed-up induced by OpenMP is negligible due to:

1. **Thread Initialization**: Creating and managing threads incurs overhead. For a small number of particles, this parallelization overhead can be comparable to or even exceed the actual computation time.
2. **Synchronization**: Threads need to synchronize to access shared resources or perform reduction operations. This synchronization adds additional overhead, which is only worth it when there is enough parallel work to be done.
3. **Workload per Thread**: Each thread has a small amount of work, leading to suboptimal utilization of the threads. With more particles, each thread has more work, increasing parallelization efficiency.

Nevertheless, once the number of particles exceeds 10^3, the speed-up induced becomes notable and will approximately reach a value of two, as shown in Fig.10.

III Classical application : Cube water at rest in a tank

The first thing one needs to verify is the simplest simulation, being a cube of water at rest, in order to verify the hydrostatic pressure induced by this configuration once it has stabilized.

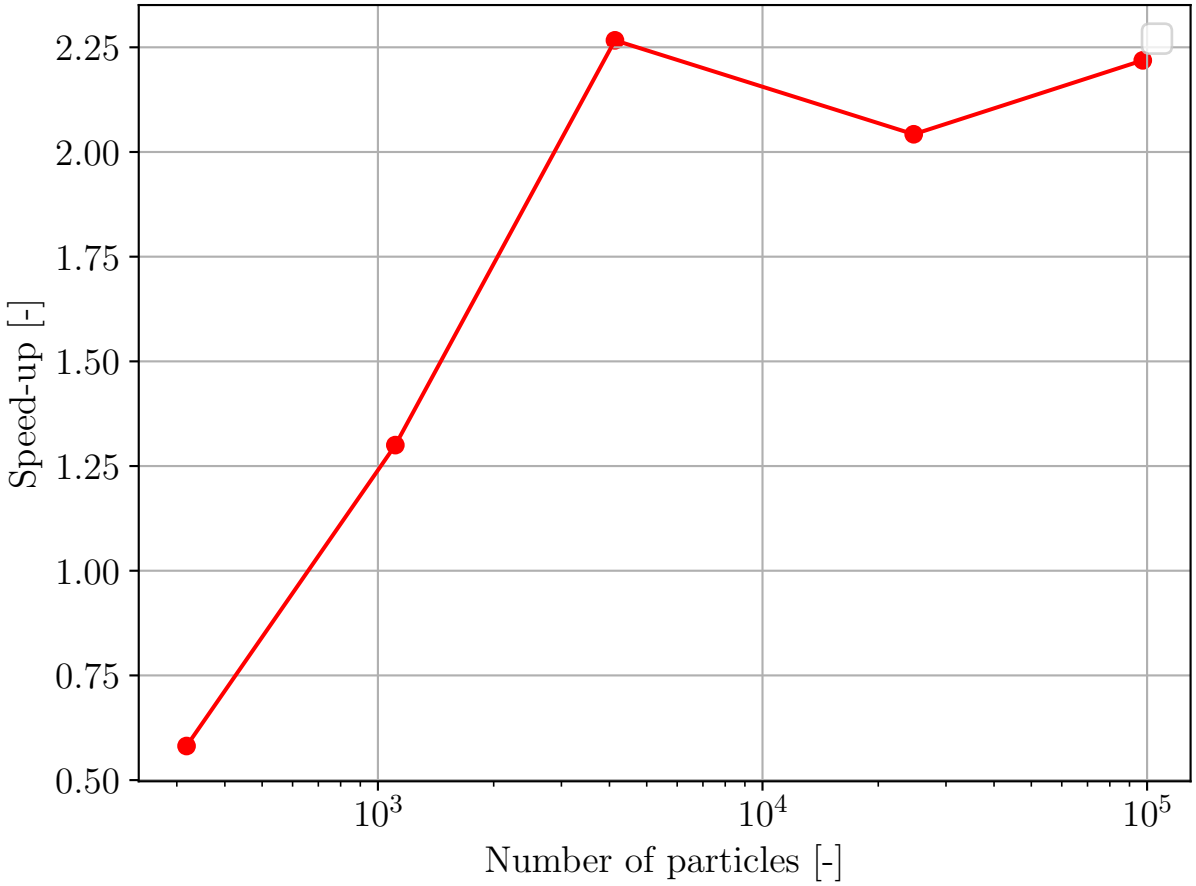

Figure 10: Speed-up induced by OpenMP with 16 threads.

Theoretical hydrostatic pressure

Reminding the momentum equation:

$$\frac{D\boldsymbol{u}}{Dt} = -\frac{\nabla p}{\rho} + \frac{\boldsymbol{F}}{\rho} = -\left(\nabla\left(\frac{p}{\rho}\right) + \frac{p}{\rho^2}\nabla\rho\right) + \frac{\boldsymbol{F}}{\rho}. \tag{III.1}$$

If the fluid is at rest, the left-hand-sided equation is null. Also, the single volume body force is the gravity. Hence, this latter equation reads:

$$\frac{\nabla p}{\rho} = g \Rightarrow p(z) = \rho q z \approx 9810 \times z, \tag{III.2}$$

where z is the height of a given particle and where the density ρ is approximated constant at 1000 [kg/m^3] and by considering an initial condition where the pressure is zero at a zero z coordinate.

One thus expects a linear trend of the pressure from the SPH simulation. In Fig.11 is depicted the comparison between the theoretical and SPH evaluated pressure (based on the *2D_hydrostatic.json* file) :

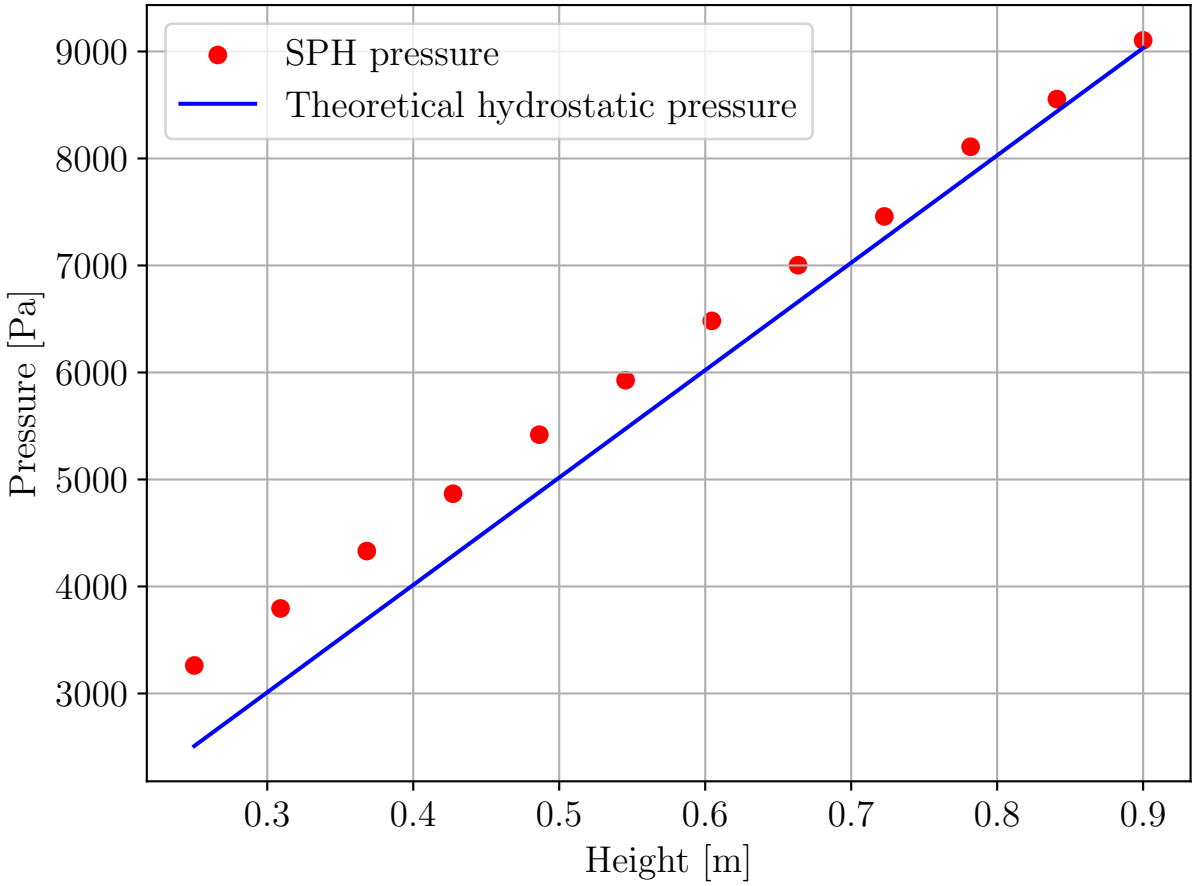

Figure 11: Pressure computed by ghost particles placed in a horizontal line which starts at (0.6,0,0.25) and ends to (0.6,0,9) at frame 249.

Parameters	hydrostatic
s	0.05
L	1.2
$Nb\ MP$	506
$c_0\ [m/s]$	30
dt_{init}	10^{-4}
α	0.5
β	0

Table 1: Parameters of the cube to sphere simulation.

Indeed, the expected behaviour has been retrieved. This 2D simulation has been made. The ghost particles used to evaluate the pressure have been placed in the middle of the tank along the entire height.

IV Microfluidic phenomenon

The main advantage of the SPH method is the mesh-free model. But this feature may become a drawback while surface tensions are involved, since the "boundary surfaces" have to be determined to apply surface tensions onto it.

Many papers use different methods to achieve this, our focus has been on a specific paper from Akinci et al. [1].

IV.1 Surface tension implementation

In their work, Akinci et al. introduce a cohesion method from a macroscopic perspective. Rather than solely focusing on cohesive forces, they also identify forces aimed at minimizing surface area. Their approach calculates the cohesive force of a particle as:

$$F_{\text{cohesion}_{i\leftarrow j}} = -\alpha_{\text{cohesion}} m_i m_j \frac{x_i - x_j}{\|x_i - x_j\|} W_{\text{cohesion}}(\|x_i - x_j\|), \tag{IV.1}$$

where i and j are neighbouring particles, α_{cohesion} a scaling factor and W_{cohesion} a special cohesion kernel which is defined by:

$$W_{\text{cohesion}}(r) = \frac{32}{\pi h'^9} \begin{cases} (h'-r)^3 r^3 & \text{if } \frac{h'}{2} < r \leq h', \\ 2(h'-r)^3 r^3 - \frac{h'^6}{64} & \text{if } 0 < r \leq \frac{h'}{2}, \\ 0 & \text{otherwise,} \end{cases} \tag{IV.2}$$

Where h' represents the support radius and in this project it corresponds to $\kappa \times h = 2 \times 1.2 \times s$ with the cubic-spline Kernel. The cohesion kernel is represented in Fig 12.

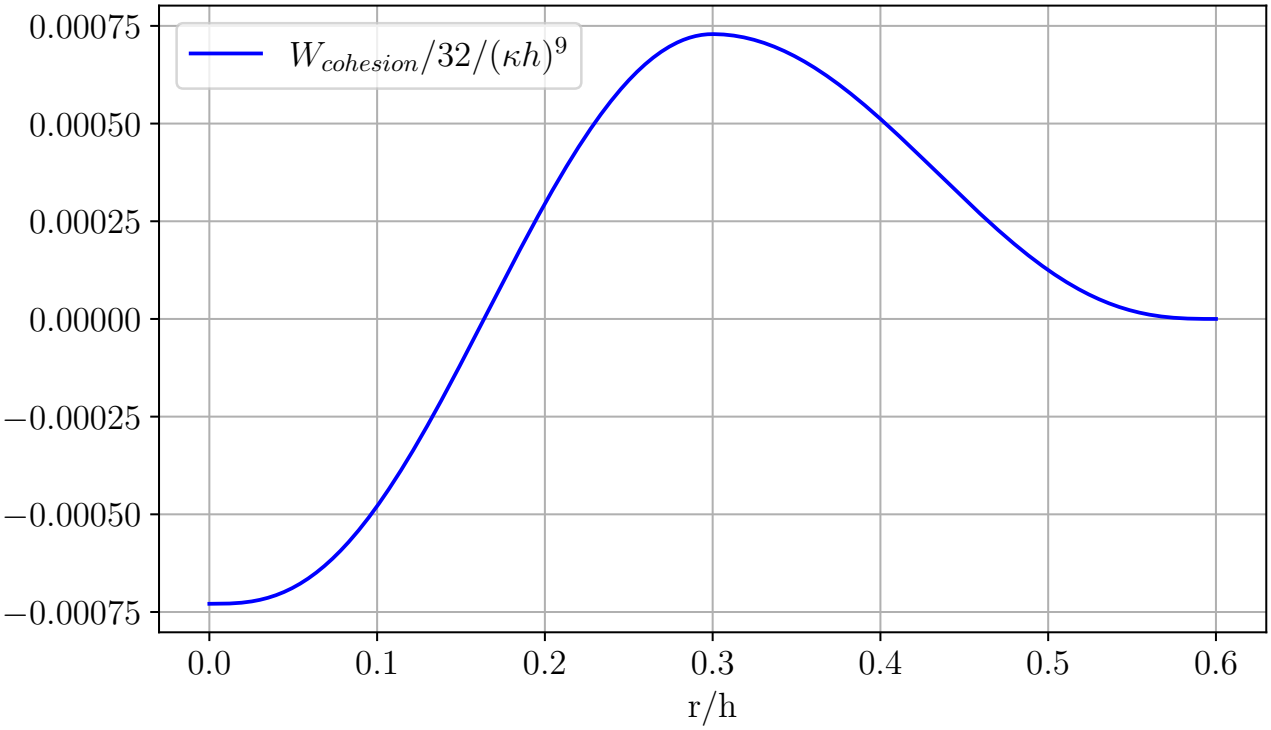

Figure 12: Cohesion kernel function with $\kappa = 2$ and $s = 0.025$.

Unlike the model proposed by Becker and Teschner [2], where cohesive forces are strictly positive, Akinci et al.'s approach allows for both positive and negative cohesive forces. This feature generates repulsion forces between closely positioned particles, preventing undesirable clustering at the free surface.

Furthermore, Akinci et al. incorporate a force aimed at minimizing surface area. This supplementary force counteracts surface curvature, necessitating the computation of surface normals. These normals can be obtained using a technique called the colour field method. Essentially, this involves assigning a colour value of 1 to the particle and 0 elsewhere, then computing the gradient of the smoothed colour field:

$$\mathbf{n}_i = h \sum_j \frac{m_j}{\rho_j} \nabla W_{ij}, \tag{IV.3}$$

This calculation provides a surface normal directed inward into the fluid, with the factor h utilized to ensure normal scale independence. Within the fluid interior, the resulting vector's magnitude approaches zero, while at the free surface, it becomes proportional to curvature. Consequently, a symmetric force, opposing curvature, can be formulated as:

$$F_{\text{curvature}_{i\leftarrow j}} = -\alpha_{curv} m_i(\mathbf{n}_i - \mathbf{n}_j). \tag{IV.4}$$

where α_{curv} is a simple scaling factor (different from the previous α used). This force is used to minimize the surface area. It is important to notice that $\alpha_{\text{cohesion}} = \alpha_{\text{curv}} = \gamma$ in Akinci et al. model, since these two forces act together to generate the surface tensions. In the simulations of this study, one calls this parameter $\alpha_{\text{s.t.}}$.

Finally, both forces are combined as:

$$F_{\text{surface tension}_{i\leftarrow j}} = K_{ij}(F_{\text{cohesion}_i} + F_{\text{curvature}_i}), \tag{IV.5}$$

where, $K_{ij} = \frac{2\rho_0}{\rho_i+\rho_j}$ is as a symmetric factor enhancing surface tension forces at the free surface. Near the surface, ρ_i and ρ_j are underestimated due to particle deficiency, resulting in $K_{ij} > 1$. Conversely, for a particle with a complete neighbourhood, K_{ij} approximates 1.

IV.1.1 Analysis

In order to check if the surface tension equation given by Eq.(IV.5) can be used for microfluidic application. One simulates a cube of fluid particles (without any other fixed particles) that should turn into a sphere. Physically, surface tensions tend to minimize the surface area of the fluid, such that the sphere is the shape that allow this condition. Because of the curvature of the fluid, a pressure difference is induced between the outside and the inside, this pressure is called the *Laplace pressure* Δp, and is given by:

$$\Delta p = \sigma \left(\frac{1}{R_1} + \frac{1}{R_2} \right), \tag{IV.6}$$

where σ is the surface tension [Pa/m^2] and R_1, R_2 are the principal radii of curvature.

For a sphere, $R_1 = R_2 = R\,[m]$ and so Eq.(IV.6) becomes:

$$\Delta p = \sigma \frac{2}{R}. \tag{IV.7}$$

Parameters of the simulation are given in Table 2, It represents a cube with a side length of 1 [m] where only surface tension (excluding gravity) is considered. At the macro scale, the surface-to-volume ratio is so small that surface tensions are negligible compared to volume body forces like gravity and hydrostatic pressure. However, since one can manually choose to apply only the surface tension to the water cube, surface tension effects can be included regardless of the simulation scale. Additionally, the cube of water is assumed to flow in a vacuum, so no hydrostatic pressure is generated. Some frame of the simulation are shown on Fig

From that, the pressure distribution can be illustrated. On the one hand, a pressure difference (considering the outside pressure is zero) occurs and is shown on Fig 18. Even if some fluctuations occur, a mean value in the inside can be calculated and is equal to 4415 [Pa]. On the other hand, the order of magnitude does not correspond to the reality. Indeed, the radius of the sphere is 0.575 [m], and with the mean pressure found, it gives a surface tension of σ = 126.93 [Pa/m^2]. In comparison, the surface tension of water is 0.072 [Pa/m^2], which is four order of magnitude lower than the value found numerically. The JSON used for the different

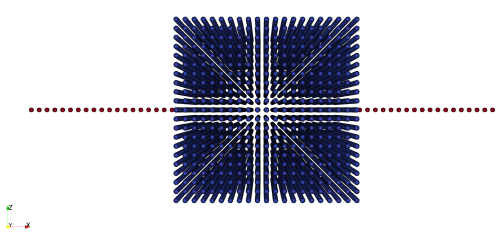

Figure 13: Frame 1

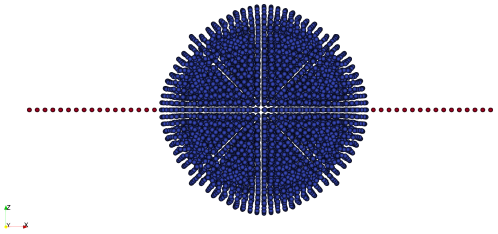

Figure 14: Frame 250

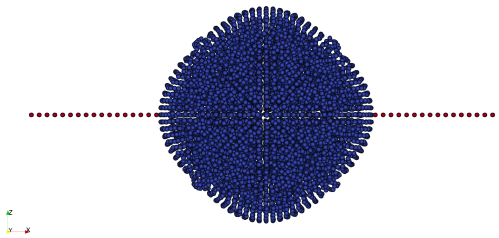

Figure 15: Frame 500

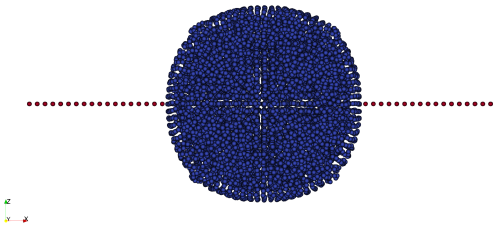

Figure 16: Frame 800

Figure 17: Frame of a simulation of a cube (blue points) of 1[m] with a spacing s = 0.05 where the surface tension is applied with $\alpha_{\text{s.t.}}$ = 10. Red points represent ghost particles for processing data.

simulation is *3D_cube_to_sphere.json* where parameters s and α_{st} was modified in order to compare different simulations.

Parameters	cube to sphere
s	0.05
L	1
$Nb\ MP$	9261
α_{curv}	10
$c_0\ [m/s]$	30
dt_{init}	10^{-6}
α	0.5

Table 2: Parameters of the cube to sphere simulation.

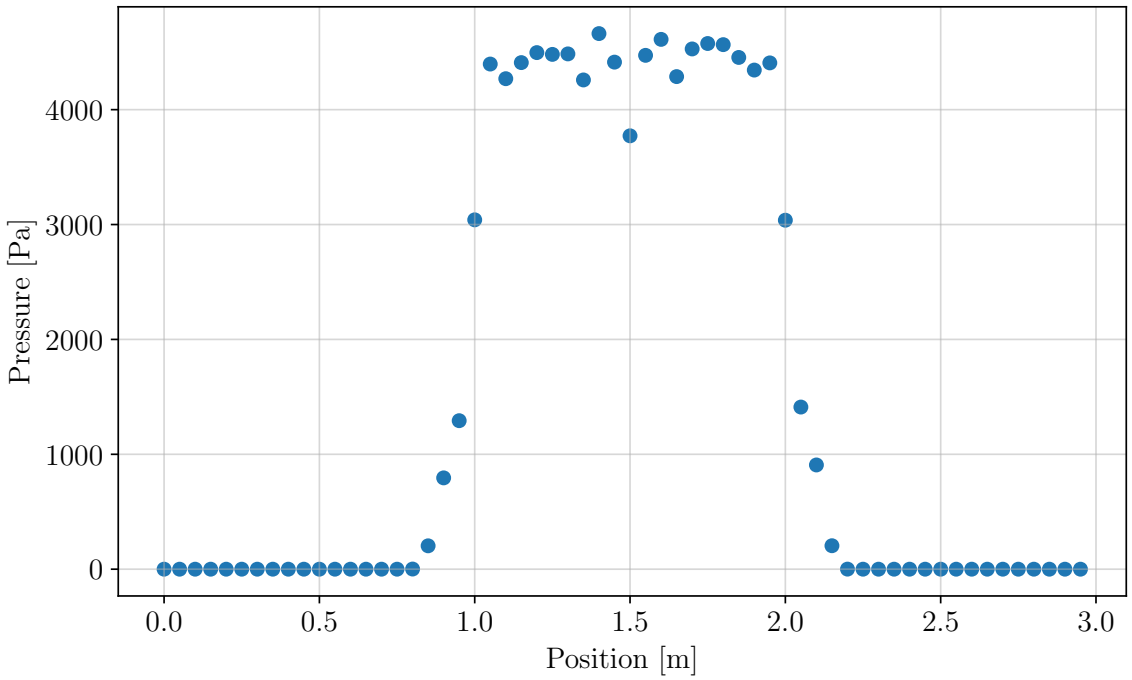

Figure 18: Pressure computed by ghost particles placed in a horizontal line which starts at (0,1.5,1.5) and ends to (3,1.5,1.5) with $s = 0.05$ for frame 800.

Fig. 19 illustrates a slice of the sphere and its inside pressure. One can see that the pressure is less important at the border, but is more or less constant inside. The low pressure values can be explained by the fact that the Kernel Correction is not taking into account for this project, and so for moving particle near the surface there is a lack of neighbours that impacts the pressure. It is also visible that the shape is not a prefect sphere. The non-uniform pressure (without taking into account the low pressure value at the border) and the non-perfect sphere may be due to the system which is not completely stabilized and by the simplicity of this method. However, a longer simulation would have required too much calculation time.

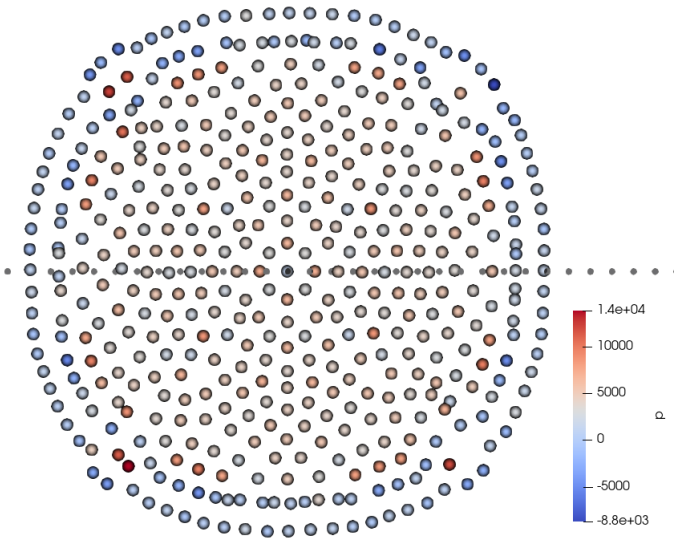

Figure 19: Pressure distribution on a slice of the droplet, where grey points represent ghost particles for frame 800.

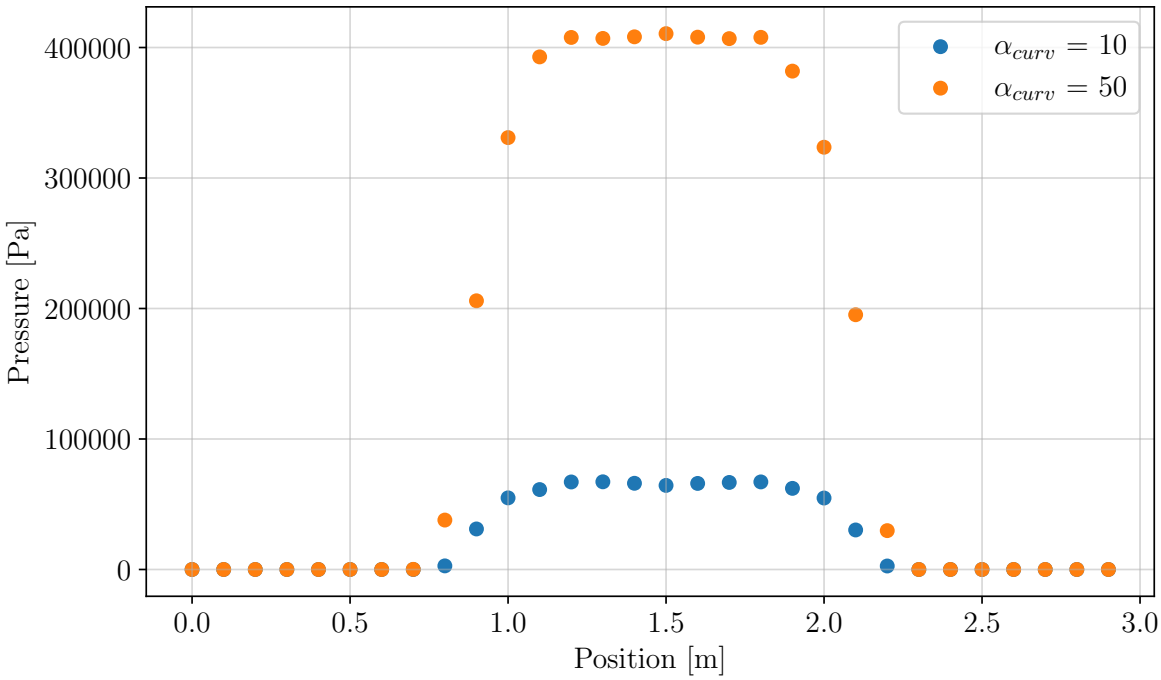

Figure 20: Pressure computed by ghost particles placed in a horizontal line which starts at (0,1.5,1.5) and ends to (3,1.5,1.5) with $s = 0.1$ for frame 500 for different $\alpha_{\text{s.t.}}$. Simulation of a cube with $L = 1$ [m].

Modifying the parameter alpha does influence the result. Indeed, as illustrated on Fig.20, the pressure inside the sphere changes too, even if the radius is kept constant. The influence of $\alpha_{\text{s.t.}}$ can be seen as a representation of the physical surface tension coefficient σ, though there are some limitations. Specifically, a higher σ results in higher pressure for a given radius. However, the simulated Laplace pressure also tends to depend on the particle spacing. Fig.21 shows that the pressure is several orders of magnitude higher for larger spacing and thus a smaller number of fluid particles. It is well known that the precision of the SPH method depends on the number of particles, but the results should not vary significantly with such small changes in spacing.

In summary, this method is quite simple, and thus are the results. With this model, it is possible to have visual results but seems not really suited for quantitative results due to $\alpha_{\text{s.t.}}$ which is not a physical parameter but a numerical one. With the comparison of the Laplace pressure, it looks like $\alpha_{\text{s.t.}}$ plays the same role of σ of formula from Eq.(IV.5), but it is not a direct linked due to the dependence on s too. Additionally, it is important to note that Akinci et al. [1] do not present quantitative results and focus solely on visual results that mimic the "real" behaviour of fluid, specifically water.

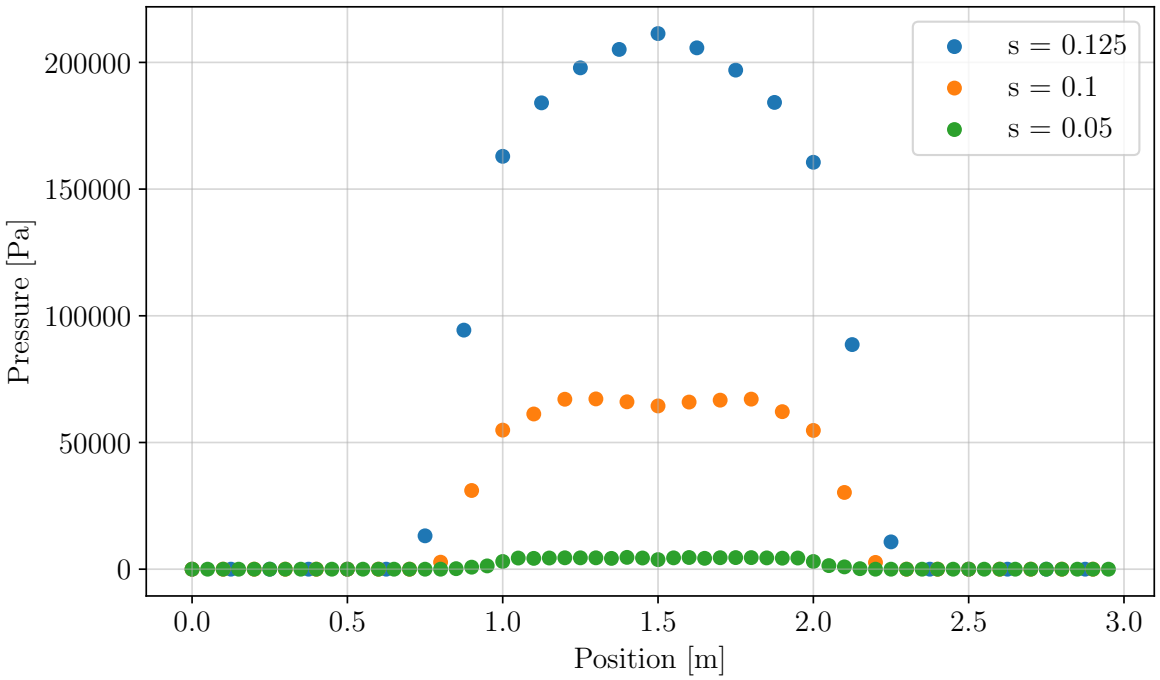

Figure 21: Pressure computed by ghost particles placed in a horizontal line which starts at (0,1.5,1.5) and ends to (3,1.5,1.5) with $\alpha_{\text{s.t.}} = 10$ for frame 500 for different spacing s. Simulation of a cube with $L = 1$ [m].

IV.2 Adhesion effect

To simulate the attractive forces between fluid particles and fixed particles, an adhesion force is introduced as:

$$\boldsymbol{F}_i^{\text{adhesion}} = -\beta_{adh} \sum_k m_i m_k \frac{\boldsymbol{x}_i - \boldsymbol{x}_k}{||\boldsymbol{x}_i - \boldsymbol{x}_k||} W_{\text{adhesion}}(||\boldsymbol{x}_i - \boldsymbol{x}_k||). \tag{IV.8}$$

where β is the adhesion coefficient and k denotes a neighbouring boundary particle.

Also, a specialized kernel is used to mimic the adhesion influence:

$$W_{\text{adhesion}}(r) = \frac{0.007}{h'^{3.25}} \begin{cases} \sqrt[4]{-\frac{4r^2}{h'} + 6r - 2h'} & \text{if } \frac{h'}{2} < r < h', \\ 0 & \text{otherwise.} \end{cases} \tag{IV.9}$$

Where h' represents the support radius and in this project it corresponds to $\kappa \times h = 2 \times 1.2 \times s$ with the cubic-spline Kernel. This adhesion Kernel is illustrated on Fig 22.

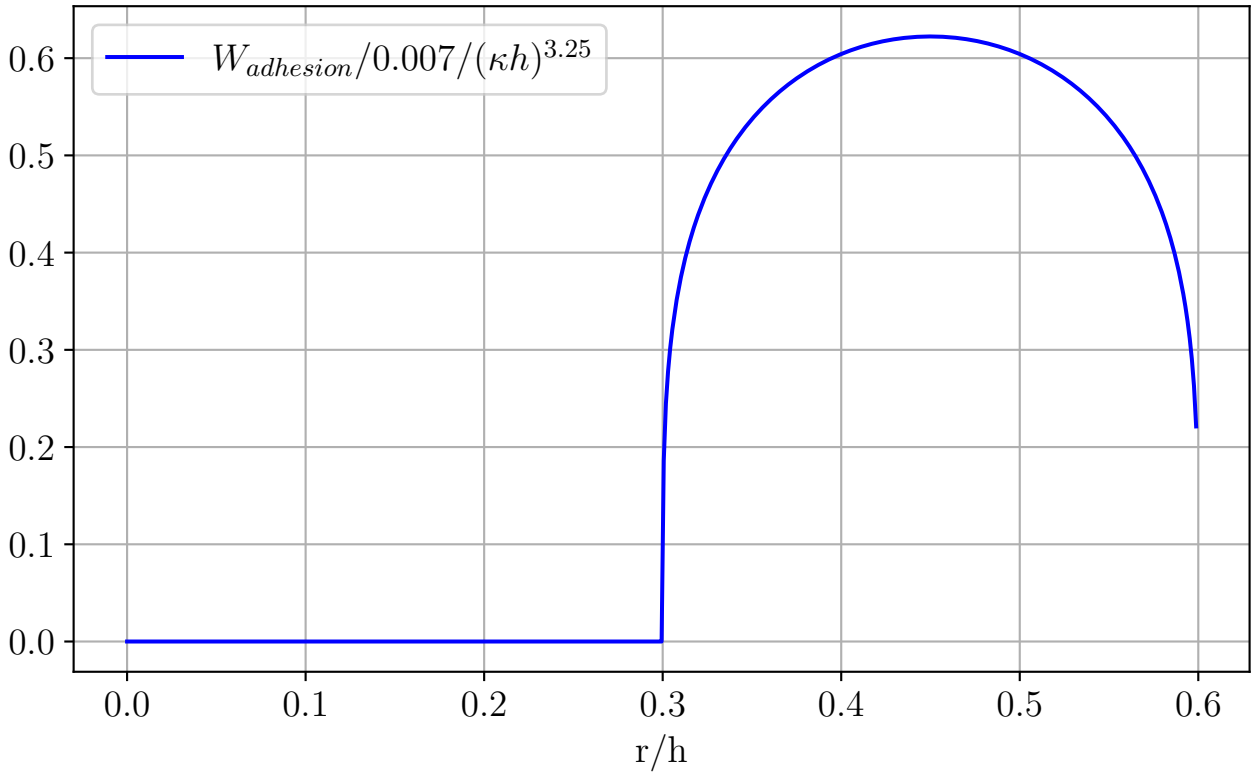

Figure 22: Adhesion kernel function for κ =2 and s = 0.25.

The specific shape of this new kernel is designed to attract particles only at distances between $h/2$ and h, while strongly attracting almost all the particles in the neighbourhood of a given particle. The primary goal of this design is to ensure that all fixed neighbouring particles of a given particle play a significant role.

IV.2.1 Analysis

In order to understand the implication of the adhesion force, a simple simulation where a cube of water starts close to an upper boundary and in which gravity is applied. On the one hand, without adhesion, all particles of the cube should fall. On the other hand, with adhesion, particles of the cube close to the boundary should be attracted to and their fall should be slow down or even completely stop if the adhesion force surpasses gravity. Two simulation are compared on Fig 23[1], it can be seen that the particles near the upper boundary (ceiling) are attracted, and this force cancel out gravity. However, particles which are not close to the ceiling fall in the same manner, either adhesion is applied or not. As for $\alpha_{\text{s.t.}}$, β_{adh} is a numerical instead of a physical parameter and the same issues as before occurs. The JSON file *3D_adhesion.json* has been used.

[1]In order to have the same simulation time, a small initial timestep is chosen ($dt_{initial} = 10^{-5}[s]$), to avoid changing it during the simulation.

(a) Adhesion: $t = 0[s]$ (b) Adhesion: $t = 0.25[s]$ (c) Adhesion: $t = 0.5[s]$ (d) Adhesion: $t = 1[s]$

(e) No adhesion: $t = 0[s]$ (f) No adhesion: $t = 0.25[s]$ (g) No adhesion: $t = 0.5[s]$ (h) No adhesion: $t = 1[s]$

Figure 23: Simulation of a fluidic cube (red particles) near an upper boundary (blue particles), falling due to gravity and where adhesion is applied (a, b, c, d) and where is not (e, f, g, h). With a spacing s = 0.1, a length L = 1 and β_{adh} = 20.

V Conclusion

In this study, we first introduced the three fundamental concepts of the Smoothed-particle hydrodynamics (SPH) method (Sec. I). Firstly, the SPH method is Lagrangian, meaning that the computational grid deforms with the object under study. Secondly, it is mesh-free, as it does not require a mesh for field variable interpolation, although mesh data may be needed for locating neighboring particles. Lastly, the method represents matter or fluid as a collection of particles that interact based on their distances (smoothing function), forming the core of the method's computational framework.

Next, the main mathematical concepts used in the study were introduced. The central concept is the *integral representation* (Eq. (II.1)), which allows for evaluating a function f(x) at any point in a domain Ω using the Dirac delta function. The second key concept is the smoothing of the Dirac delta function using a *smoothing function*, which represents the influence of a given neighborhood on a specific function evaluated at a point. This neighborhood support is described by a radius κh around a given location. The smoothing length h is kept constant throughout the simulation and is chosen as $h = 1.2s$ where s is the minimum spacing between particles. The smoothing function, also known as the *kernel function*, used in this study is the cubic spline kernel (Eq. (II.5)) with $\kappa = 2$, chosen for its renowned accuracy in classical numerical results. Spatial discretization is achieved using the *summation density* method (Eq. (II.7)) to represent a function at a point. For time discretization, the *Euler explicit method* (Eq. (II.11)) was chosen for its relatively low computational cost and the associated timestep threshold (Eq. (II.39)). Following this, the particle sorting algorithm, specifically the *Linked-list* algorithm, was discussed in Sec. II.5. This algorithm is more efficient than the *naive* algorithm for particle numbers exceeding one thousand. Finally, the Navier-Stokes equations adapted for the SPH method (Eq.(II.22) and Eq.(II.23)) were introduced, along with the *artificial viscosity* variables used to mimic the friction between particles (Eq.(II.24)).

Subsequently, two relationships between pressure and density were investigated (Sec.II.8): the *ideal gas law* (Eq.(II.27)) and the *quasi incompressible* equation (Eq.(II.30)). The focus was primarily on the second equation, as the study mainly dealt with water particles. These two

state equations also influence how the density is initialized at the beginning of the simulation, depending on the desired physical behaviour. For instance, a hydrostatic pressure law is used to assign the initial density in dam break cases. Next, the hydrostatic pressure was correctly retrieved from the *cube of water at rest in a tank* case (Sec.III) with a reasonably low number of particles.

Last but not least, the speedup induced by the parallelization of the code using OpenMP was assessed (Sec.II.10) based on the number of particles processed. A quasi-constant speedup ratio of two was found for at least 5×10^3 particles. All particle loops within the code were parallelized, as each particle operation is independent of the others.

Finally, microfluidic applications in this project were approached using the method proposed by Akinci et al. [1]. This method introduces both surface tension and adhesion forces. The cohesion forces were modelled by introducing two sub-forces: the *cohesion forces* (Eq.(IV.1)) and a force that minimizes the surface area (Eq.(IV.4)). These two forces effectively mimic the attractive and repulsive behaviour of fluid particles. A special associated kernel was also used (Eq.(IV.2)). Some visual experiments were performed, and attempts were made to retrieve the *Laplace pressure* from these simulations. Unfortunately, the parameters involved were more numerical than physical, indicating that they are too simplistic to simulate real applications. However, more advanced algorithms for surface tension, which yield more physical results due to the inclusion of the real surface tension parameter σ, can be used as explained in [1]. Additionally, adhesion forces were introduced to mimic the behaviour of boundaries on fluid particles with an adhesion force Eq.(IV.8) and an associated kernel Eq.(IV.9). Although visual results were obtained, no physical variables were measured.

After many researches, it is clear that microfluidic applications can be derived from the SPH, but more elaborated algorithms need to be employed to hopefully retrieve some physical microfluidic variables.

After extensive research and numerous investigations, it has become evident that while microfluidic applications can indeed be derived from the Smoothed Particle Hydrodynamics (SPH) method, achieving accurate and reliable physical microfluidic variables requires the implementation of more sophisticated and elaborated algorithms. The basic SPH method, while versatile and powerful in many respects, often relies on numerical parameters that lack direct physical correlation, especially in the realm of microfluidic where precision and accurate representation of physical phenomena are paramount.

References

[1] Akinci, Nadir ; Akinci, Gizem ; Teschner, Matthias: Versatile surface tension and adhesion for SPH fluids. In: ACM Transactions on Graphics (TOG) 32 (2013), Nr. 6, S. 1–8

[2] Becker, Markus ; Teschner, Matthias: Weakly compressible SPH for free surface flows. In: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation, 2007, S. 209–217

[3] Goffin, Louis: Development of a didactic SPH model, ULiège - Université de Liège, Diplomarbeit, June 2013

[4] Liu, Gui-Rong ; Liu, Moubin B.: Smoothed particle hydrodynamics: a meshfree particle method. World scientific, 2003

[5] Monaghan, Joe J.: Smoothed particle hydrodynamics. In: In: Annual review of astronomy and astrophysics. Vol. 30 (A93-25826 09-90), p. 543-574. 30 (1992), S. 543–574