Chapter 1

Complex system modelization

1.1 Introduction

In this section we give details on our choices for the generic application confronted to both computation and communication walls in irregular context. The problem we choose, the Smoothed Particle Hydrodynamics simulation method, is described on a physical aspect. We point out the difficulties involved in the resolution on supercomputers and especially hybrid architectures. A lot of work have been done on the comprehension of the physical aspect to produce a code that meet the behavior of real physics problems.

The first section is a presentation of the SPH method itself and the overall limitation we had to face. Then we describe different kind of problems we use as a benchmark of the application itself.

1.2 Physical modelization and downsides

We identified two main walls in our metrics: The computational wall and the communication/memory one. We conducted tests on both keeping as a background the irregularity behavior. We show how this is perfectly meet in the same problem with SPH. In order to add another complexity layer in term of irregularity for both computation and communication we targeted astrophysical simulations. This requires the computation of gravitation on a very high number of particles. This part describes the SPH method itself and the gravitation computation we used based on fast multipole methods.

1.2.1 Smoothed Particles Hydrodynamics

Smoothed Particle Hydrodynamics (SPH) is an explicit numerical mesh-free Lagrangian method. It is used to solve hydrodynamical partial differential equations (PDEs) by discretized them into a set of fluid elements called particles. This computational method was invented for the purpose of astrophysics simulations by Monaghan, Gingold and Lucy in 1977 [Luc77, GM77]. This first SPH work conserved mass and they later proposed a method which also conserves linear and angular moment [GM82]. The method was extended for general fluid simulation and many more fields from ballistics to oceanography. The development of new reliable, parallel and distributed tools for this method is a challenge for future HPC architectures with the upcoming exascale systems.

The method, as illustrated in figure 1.1, computes the evolution of a group of particles representing physical quantities. Those physical quantities are either invariant or computed for every particle regarding its neighbors in the radius of its smoothing length h. The particles in this radius are then valued according to their distance using a smoothing function W, also called a kernel. The fundamental SPH formulation for any physical quantity A is then to compute

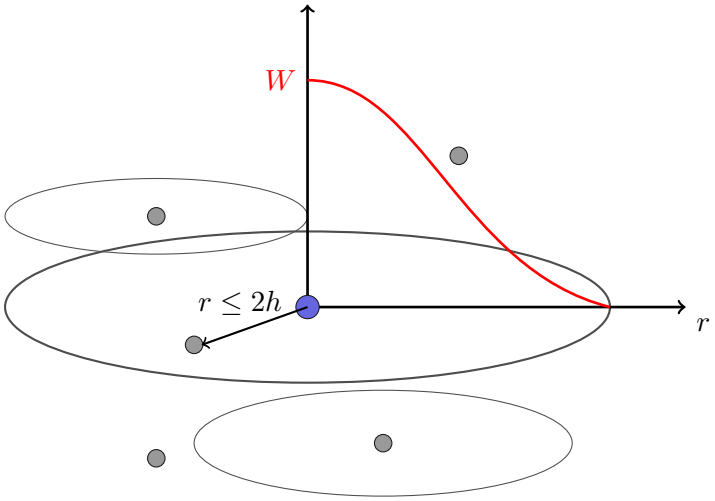


Figure 1.1: SPH kernel W and smoothing length h representation

with all the neighbors of b of a a particle by:

$$A(\vec{r})_a \simeq \sum_b \frac{m_b}{\rho_b} A(\vec{r}_b) W(|\vec{r} - \vec{r}_b|, h) \tag{1.1}$$

On a physics aspect, this method has several advantages: it can handle deformations, low densities, vacuum, and makes particle tracking easier. It also conserves mass, linear and angular momenta, and energy by its construction that implies independence of the numerical resolution. Another strong benefit of using SPH is its exact advection of fluid properties. Furthermore, the particle structure of SPH easily combines with tree methods for solving Newtonian gravity through N-body simulations. As a mesh-free method, it avoids the need of grid to calculate the spatial derivatives.

However, there are cons to consider using SPH: it cannot be extend to all PDE formulations; it requires careful setup of initial distribution of particles; further, it can be struggle to resolve turbulence-dominated flows and special care must be taken when handling high gradients such as shocks and surface structure of neutron stars. Many works are leading to handle more cases and to push the limitations of this method [DRZR17, LSR16, RDZR16].

In this work, we are solving Lagrangian conservation equations (Euler equations) for density, energy and momentum of an ideal fluid [LL59] such that:

$$\frac{d\rho}{dt} = -\rho(\nabla \cdot \vec{v}), \quad \frac{du}{dt} = -\frac{P}{\rho}(\nabla \cdot \vec{v}), \quad \frac{d\vec{v}}{dt} = -\frac{1}{\rho}(\nabla P) \tag{1.2}$$

with ρ the density, P the pressure, u the internal energy and v the velocity, ∇ the nabla operator and where $d/dt = \partial/\partial_t + \vec{v} \cdot \nabla$ which is convective derivative.

By using the volume element $V_b = m_b/\rho_b$, we can formulate the Newtonian SPH scheme [Ros09] such that

$$\rho_a = \sum_b m_b W_{ab}(h_a) \tag{1.3}$$

$$\frac{du_a}{dt} = \frac{P_a}{\rho_a^2} \sum_b m_b \vec{v}_{ab} \cdot \nabla_a W_{ab} \tag{1.4}$$

$$\frac{d\vec{v}_a}{dt} = -\sum_b m_b \left(\frac{P_a}{\rho_a^2} + \frac{P_b}{\rho_b^2} \right) \nabla_a W_{ab} \tag{1.5}$$

where $W_{ab} = W(|\vec{r}_a - \vec{r}_b|, h)$ is the smoothing kernel. The equations we would like to solve allow for emergence of discontinuities from smooth initial data. At discontinuities, the entropy

increases in shocks. That dissipation occurs inside the shock-front. The SPH formulation here is inviscid so we need to handle this dissipation near shocks. There are a number of way to handle this problem, but the most widespread approach is to add artificial viscosity (or artificial dissipation) terms in SPH formulation such that:

$$\left(\frac{du_a}{dt}\right)_{art} = \frac{1}{2}\sum_b m_b \Pi_{ab} \vec{v}_{ab} \cdot \nabla_a W_{ab} \tag{1.6}$$

$$\left(\frac{d\vec{v}_a}{dt}\right)_{art} = -\sum_b m_b \Pi_{ab} \nabla_a W_{ab} \tag{1.7}$$

In general, we can express the equations for internal energy and acceleration with artificial viscosity

$$\frac{du_a}{dt} = \sum_b m_b \left(\frac{P_a}{\rho_a^2} + \frac{\Pi_{ab}}{2}\right) \vec{v}_{ab} \cdot \nabla_a W_{ab} \tag{1.8}$$

$$\frac{d\vec{v}_a}{dt} = -\sum_b m_b \left(\frac{P_a}{\rho_a^2} + \frac{P_b}{\rho_b^2} + \Pi_{ab}\right) \nabla_a W_{ab} \tag{1.9}$$

Π_{ab} is the artificial viscosity tensor. As long as Π_{ab} is symmetric, the conservation of energy, linear and angular momentum is assured by the form of the equation and antisymmetry of the gradient of kernel with respect to the exchange of indices a and b. Π_{ab} may define in different ways and here we use [MG83] such as:

$$\Pi_{ab} = \begin{cases} \frac{-\alpha \bar{c}_{ab} \mu_{ab} + \beta \mu_{ab}^2}{\bar{\rho}_{ab}} & \text{for } \vec{r}_{ab} \cdot \vec{v}_{ab} < 0 \\ 0 & \text{otherwise} \end{cases} \tag{1.10}$$

$$\mu_{ab} = \frac{\bar{h}_{ab} \vec{r}_{ab} \cdot \vec{v}_{ab}}{r_{ab}^2 + \epsilon \bar{h}_{ab}^2} \tag{1.11}$$

Using the usual form c_s as $c_s = \sqrt{\frac{\partial p}{\partial \rho}}$. The values of ϵ, α, and β have to be set regarding the problem targeted. As an example we used for the Sod shock tube problem: $\epsilon = 0.01h^2$, $\alpha = 1.0$, and $\beta = 2.0$.

There are many possibilities for the smoothing function, called the kernel. As an example the Monaghan's cubic spline kernel is given by:

$$W(\vec{r_{ij}}, h) = \frac{\sigma}{h^D} \begin{cases} 1 - \frac{3}{2}q^2 + \frac{3}{4}q^3 & \text{if} \quad 0 \leq q \leq 1 \\ \frac{1}{4}(2-q)^3 & \text{if} \quad 1 \leq q \leq 2 \\ 0 & \text{otherwise} \end{cases} \tag{1.12}$$

where $q = r/h$, r the distance between the two particles, D is the number of dimensions and σ is a normalization constant with the values:

$$\sigma = \begin{cases} \frac{2}{3} & \text{for 1D} \\ \frac{10}{7\pi} & \text{for 2D} \\ \frac{1}{\pi} & \text{for 3D} \end{cases} \tag{1.13}$$

In the computation of forces we also need to apply the gradient of the smoothing kernel. This is, in our example, for the cubic spline kernel:

$$\vec{\nabla}_i W(\vec{r_{ij}}, h) = \frac{\sigma}{h^{D+1}} \times \begin{cases} (-\frac{3}{h} + \frac{9}{4h}q)\vec{r_{ij}}, & \text{si } 0 \leq \frac{r}{h} < 1 \\ (\frac{-3}{r} + \frac{3}{h} - \frac{3}{4h}q)\vec{r_{ij}}, & \text{si } 1 \leq \frac{r}{h} < 2 \\ 0, & \text{si } \frac{r}{h} \geq 2 \end{cases} \tag{1.14}$$

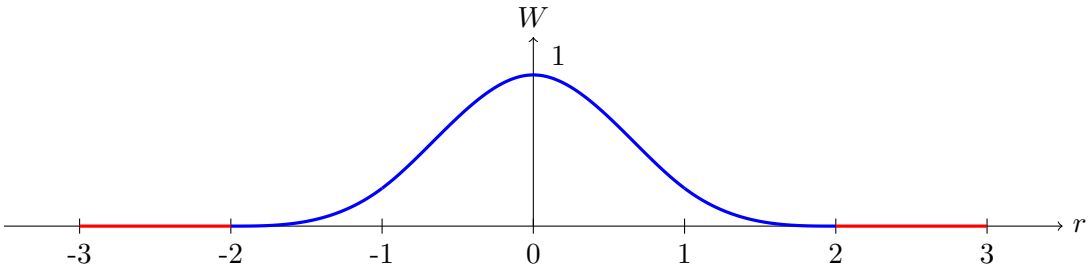


Figure 1.2: Cubic spline kernel example with $\sigma = 1$ and $h = 1$

Figure 1.2 is a representation of the cubic spline kernel with $\sigma = 1$ and $h = 1$. The abscissa axis represent r, distance between the particles and the ordinate the value of the smoothing kernel. When the support of the function, 2 is reached the particles are ignored $W = 0$, represented in red on the figure.

To sum up, the SPH resolution scheme and its routines are presented on algorithm 1. The Equation of State (EOS) and the integration are problem dependent and will be define for each test case in section 1.3.

Algorithm 1 SPH loop algorithm

```
1: while not last step do
2:     Compute density for each particle (1.3)
3:     Compute pressure using EOS
4:     Compute acceleration from pressure forces (1.9)
5:     Compute change of internal energy for acceleration (1.8)
6:     Advance particles after integration
7: end while
```

The main downside for the implementation of this method is the requirement for local computation on every particle. The particles have to be grouped locally to perform the computation of (1.3), (1.8) and (1.9). A communication step is needed before and after (1.3) to get the local physical data to be able to compute (1.8) and (1.9). The tree data structure allows us to perform $O(Nlog(N))$ neighbor search but also add a domain decomposition and distribution layer.

1.2.2 Gravitation

In order to target hard irregular simulation facing both the communication and computation wall in irregular behavior we decided to simulate astrophysical events. This choice is also accurate since our code, FleCSPH, will be use by the LANL astrophysicists in the near future. In order to perform those simulation the computation of gravitation/self-gravitation is required. This part present our implementation choice and expose the main problems for HPC implementation.

For classical problems like fluid flow the gravitation can directly be applied on the particles with the force:

$$\vec{a_g} = m\vec{g} \tag{1.15}$$

In order to consider astrophysics problems we need to introduce self-gravitation and gravitation. Each particle imply an action on the others based on its distance and mass. The equation of gravitation for a particle i with j other particles is:

$$\vec{f}_{ai} = \sum_j -G\frac{m_i m_j}{|\vec{r_i} - \vec{r_j}|^3}\vec{r_{ij}} \tag{1.16}$$

This computation involve an $O(N^2)$ complexity and thus is not applicable directly. We applied the method called Fast Multipole Method, FMM and discussed in [BG97]. This method is perfectly adapted to a tree representation of the domain and particles.

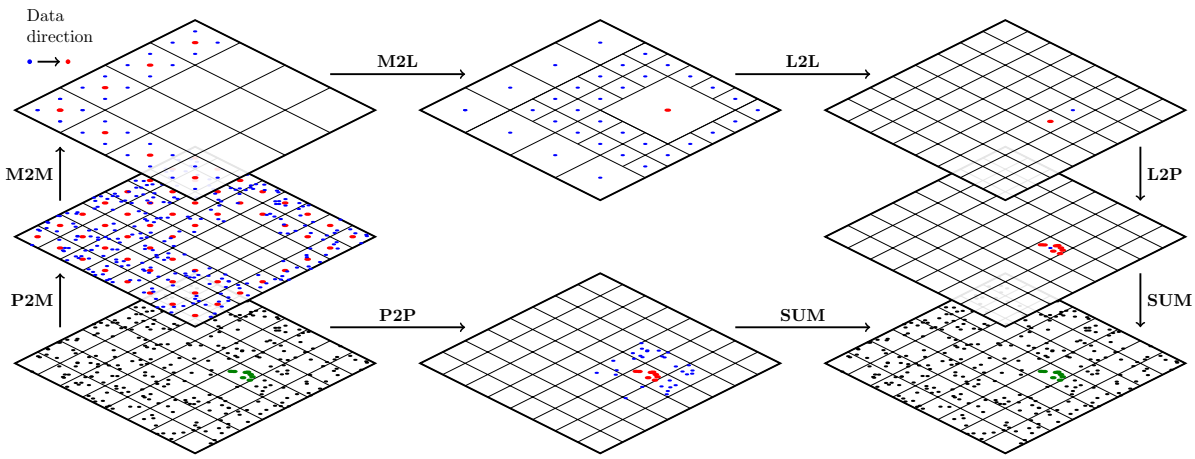


Figure 1.3: Fast Multipole Method schematics. Particles to Multipole (P2M), Multipole to Multipole (M2M), Multipole to Particles (M2P), Multipole to Local (M2L), Local to Local (L2L) and Particles to Particles (P2P). Schematic inspired from [YB11]

This method aim to compute the gravitation up to an approximation determined by the user. Details are given in figure 1.3, from left bottom to right bottom for a group of particles. We identify three main actors in this method:

Particles: the bodies on which we need to compute the gravitation regarding the other particles. In figure 1.3 we separate the green particles, on which we are computing the gravitation, from the other, blue, particles.

Multipoles: the center of mass for a group of particles. In our example they regroup the mass and barycenter of the sub-particles. There are several level of multipoles: particles' multipole and multipoles' multipole.

Locals: the center of mass for the reduction on the particles concerned in this walk. They have the same behavior as the multipoles but the information goes down to particles instead of up.

In order to compute the gravitation for a group of particles in the domain the steps are:

Particles to Particles (P2P): For the particles that are close, use the direct $O(N^2)$ algorithm. This is the part that grow if the user desires more accurate results.

Particles to Multipoles (P2M): Gather the data of all the sub-particles to the centers of mass, the multipoles. This is the first layer of the tree, the leaves.

Multipoles to Multipole (M2M): Gather the data of multipoles on higher level of the tree from the leaves to the root.

Multipoles to Local (M2L): Compute the gravitation part of all the distant multipole to the local.

Local to Local (L2L): Go down in the tree and spread the component to sub-locals.

Local to Particles (L2P): When a leaf of the tree is reached, compute the application of the local for all sub-particles.

Summation: At the end of the computation for both P2P and L2P the two interactions can be summed up to compute the gravitation applied to the particles.

This scheme have to be repeat for every group of particles. The P2M-M2M steps are done just once before the FMM method for all the groups of particles. For the choice between either P2P or M2L we use a criterion call MAC, Multipole Acceptance Criterion. In this study we used an angle between the local center of mass and the edge of distant multipole. If the angle fits

the criterion we use the current multipole, otherwise we go lower in the tree to consider smaller multipole. If the criterion never match, we are too close and consider P2P.

For the P2P step, the classical gravitation computation is used, like presented in equation 1.16. But for the interaction with distant multipoles, we use a Taylor series.
The gravitation function of equation 1.16 can be approximate on a particle at position $\vec{r}$ by the gravitation computed at the centroid at position $\vec{r_c}$:

$$\vec{f}(\vec{r}) = \vec{f}(\vec{r_c}) + ||\frac{\partial \vec{f}}{\partial \vec{r}}|| \cdot (\vec{r} - \vec{r_c}) + \frac{1}{2}(\vec{r} - \vec{r_c})^\intercal \cdot ||\frac{\partial \vec{f}}{\partial \vec{r}\partial \vec{r}}|| \cdot (\vec{r} - \vec{r_c}) \tag{1.17}$$

From equation 1.16 we compute the term $||\frac{\partial \vec{f}}{\partial \vec{r}}||$:

$$\frac{\partial \vec{f}}{\partial \vec{r}} = -\sum_p \frac{m_p}{|\vec{r_c} - \vec{r_p}|^3} \begin{bmatrix} 1 - \frac{3(x_c - x_p)(x_c - x_p)}{|\vec{r_c} - \vec{r_p}|^2} & -\frac{3(y_c - y_p)(x_c - x_p)}{|\vec{r_c} - \vec{r_p}|^2} & -\frac{3(z_c - z_p)(x_c - x_p)}{|\vec{r_c} - \vec{r_p}|^2} \\ -\frac{3(x_c - x_p)(y_c - y_p)}{|\vec{r_c} - \vec{r_p}|^2} & 1 - \frac{3(y_c - y_p)(y_c - y_p)}{|\vec{r_c} - \vec{r_p}|^2} & -\frac{3(z_c - z_p)(y_c - y_p)}{|\vec{r_c} - \vec{r_p}|^2} \\ -\frac{3(x_c - x_p)(z_c - z_p)}{|\vec{r_c} - \vec{r_p}|^2} & -\frac{3(y_c - y_p)(z_c - z_p)}{|\vec{r_c} - \vec{r_p}|^2} & 1 - \frac{3(z_c - z_p)(z_c - z_p)}{|\vec{r_c} - \vec{r_p}|^2} \end{bmatrix} \tag{1.18}$$

And we propose a compact version of the matrix with:

$$||\frac{\partial f^a}{\partial r^b}|| = -\sum_c \frac{m_c}{|\vec{r} - \vec{r_c}|^3} \Big[\delta_{ab} - \frac{3.(r^a - r_c^a)(r^b - r_c^b)}{|\vec{r} - \vec{r_c}|^2}\Big] \tag{1.19}$$

With δ_{ab} the Kronecker delta:

$$\delta_{ab} = \begin{cases} 1, & \text{if } a = b. \\ 0, & \text{if } a \neq b. \end{cases} \tag{1.20}$$

We note that a and b variate from 0 to 2 and $r^0 = x$, $r^1 = y$, and $r^2 = z$ as usual sense.
For the term $||\frac{\partial \vec{f}}{\partial \vec{r}\partial \vec{r}}||$ we give the compact version by:

$$\begin{aligned} ||\frac{\partial^2 f^a}{\partial r^b \partial r^c}|| = -\sum_c \frac{3m_c}{|\vec{r} - \vec{r_c}|^5} \Big[& \frac{5(r^a - r_c^a)(r^b - r_c^b)(r^c - r_c^c)}{|\vec{r} - \vec{r_c}|^2} - \\ & \Big(\delta_{ab}(r^c - r_c^c) + \delta_{bc}(r^a - r_c^a) + \delta_{ac}(r^b - r_c^b)\Big) \Big] \end{aligned} \tag{1.21}$$

The equations 1.19 and 1.21 are use during the M2L step. Then to go down in the tree and apply the gravitation to locals and then particles in L2L and L2P we use equation 1.17.

This methods impose a lot of communications and exchanges between the processes. In our distributed version the particles are separate for each processes. Indeed, as each of them will hold part of the particles, the multipole in M2L computation imposes to share data. The P2P computation will face issues on the edge of each sub-domain. The irregular behavior is also present for the choice of the multipole to consider during the M2L step based on the MAC criterion.

1.3 Test cases with SPH

The generic SPH method and the gravitation computation, either simple or with FMM, allow us to run standard example to check the physical accuracy of our code. We choose four mains examples in this purpose: the Sod shock tube, the Sedov blast wave, fluid flow simulation and astrophysical examples like binary neutron stars coalescence. We present those problems in this section and their characteristics regarding the computer science difficulties involved.

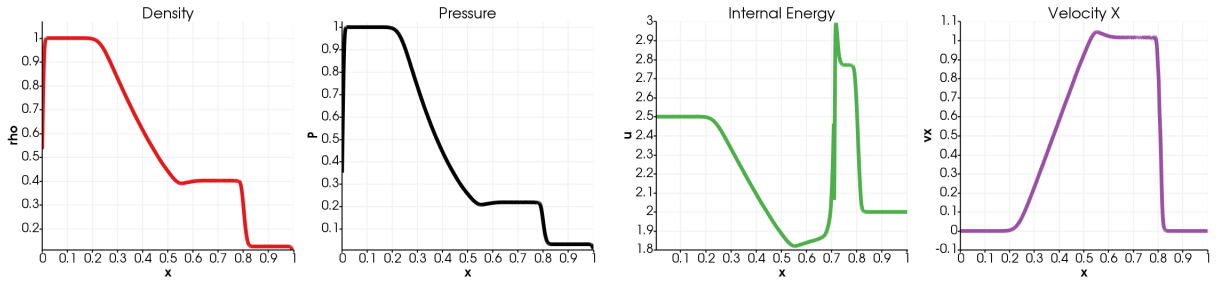


Figure 1.4: Sod shock tube with FleCSPH

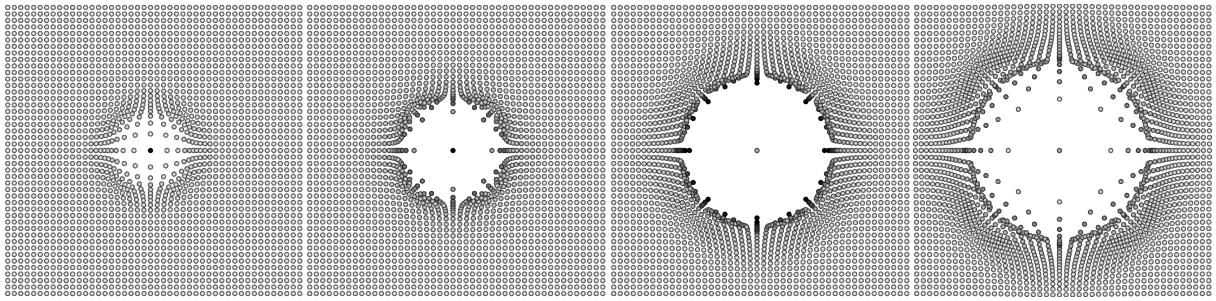


Figure 1.5: Sedov Blast Wave with FleCSPH at respectively $t = 0.01$, $t = 0.03$, $t = 0.06$ and $t = 0.1$

1.3.1 Simple tests: tree and communications

Sod shock tube

The Sod shock tube is the test consisting of a one-dimensional Riemann problem with the following initial parameters [Sod78].

$$(\rho, v, p)_{t=0} = \begin{cases} (1.0, 0.0, 1.0) & \text{if} \quad 0 < x \leq 0.5 \\ (0.125, 0.0, 0.1) & \text{if} \quad 0.5 < x < 1.0 \end{cases} \tag{1.22}$$

In our code, we use the same initial data as in section 1.2.1 with ideal gas EOS such as:

$$P(\rho, u) = (\Gamma - 1)\rho u \tag{1.23}$$

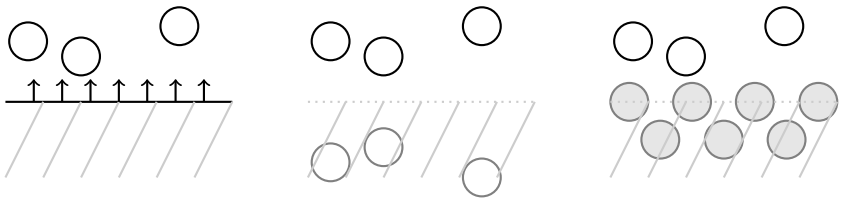
where Γ is the adiabatic index of the gas, we set $\Gamma = 5/3$.

This test is used to check the physical accuracy of the code and thus the tree search itself. A simulation of our Sod shock experimentation is presented on figure 1.4 and shows physically correct results. The first difficulty of this problem is the tree repartition, the physics behind is not complicated and does not involve specific optimizations.

Sedov blast wave

A blast wave is the pressure and flow resulting from the deposition of a large amount of energy in a small very localized volume. There are different versions of blast wave test and we consider comparing it with the analytic solution for a point explosion as given by Sedov [Sed46], making the assumption that the atmospheric pressure relative to the pressure insider the explosion negligible. Here, we test 2D blast wave. In this simulation, we use ideal gas EOS with $\Gamma = 5/3$ and we are assuming that the undistributed area is at rest with a pressure $P_0 = 1.0^{-5}$. The density is constant ρ_0, also in the pressurized region.

An example of our Sedov Blast wave experimentation is presented on figure 1.5 and shows physically correct results. This problem have the same conclusion as the Sod shock tube, his purpose is the test the tree behavior with 2 dimensions.



Repulsive force Mirror particles Dummy particles

Figure 1.6: Different boundaries condition methods

1.3.2 High number of particles and specific formulation

After performing the tests regarding the physics reliability, we worked on fluid flow problem in 2D and 3D to reach high number of particles. The details can be found in [GGRC+12]. This test is based on an ideal EOS given by:

$$P = B\Big[\big(\frac{\rho}{\rho_0}\big)^{\gamma} - 1\Big] \tag{1.24}$$

with $\gamma = 7$ and $B = c_0\rho_0/\gamma$ being $\rho_0 = 1000\, kg.m^{-3}$ the reference density.

Boundary conditions

For this experiment, realistic boundaries conditions were needed. Several methods are possible with SPH and we focused on the main ones: the repulsive wall, the mirror particles [LP91] and the dummies particles implementation [AHA12]. Those boundaries conditions implementation are presented in figure 1.6.

On our first implementation we used repulsive forces, but they imposed a lot of computation during the integration step and does not handle all the shape of boundaries easily.

For the current implementation we used the dummies particles method. This method is accurate and allows us to consider all the shape of boundaries. The wall particles are just considered as normal particles, with specific equations, and their quantities are evolved during the run. The main difference is that their position does not evolve at the end of the step. They are identified in the code with a specific type, provided during the data generation.

In this fluid flow example we were able to go up to a high number of particles with 2 or 3 dimensions. The handling of boundaries conditions added a pressure on the irregularity behavior.

1.3.3 SPH and gravitation

The final aim of our tests is to simulate astrophysical events. We are interested in one of the most important event recently discovered. Last year the Laser Interferometer Gravitational Wave, LIGO, detected the first gravitational wave generated by binary neutron stars merging [AAA+17b] and also more complexes event with Binary Black Holes coalescence in [AAA+17a]. We decided to conduct tests on Binary Neutron Star, BNS, coalescence. This problem represent all the aspect needed in our benchmark with the SPH method but also the self-gravitation and gravitation computation. In order to perform realistic simulations we consider two stars that are stable. We give details on the way to generate those data and the physics of the fusion itself. A lot of work have been involve in this understanding since it is quite out of the field of the primary researches.

Solving Lane-Emden Equation

In order to consider BNS we first consider two individual stars. The star is characterized by its radius and mass but also the density repartition inside. Indeed, we have to determine the density

	NS_1	NS_2	NS_3	NS_4
Radius (cm)	$R = G = M = 1$	1500000	1400000	960000
K	0.636619	95598.00	83576.48	39156.94

Table 1.1: Examples of the proportionality constant K regarding the star stellar radius R

function based on the radius. For this purpose we used the so called Land-Emden equation.

REFERENCE As we consider the star as a polytropic fluid, we use the equation of Lane-Emden which is a form of the Poisson equation:

$$\frac{d^2\theta}{d\xi^2} + \frac{2}{\xi}\frac{d\theta}{d\xi} + \theta^n = 0 \tag{1.25}$$

With ξ and θ two dimensionless variables. There is only exact solutions for a polytropic index $n = 0.5$, 1 and 2. In our work we use a polytropic index of 1 which can correspond to a NS simulation.

For $n = 1$ the solution of equation 1.25 is:

$$\theta(\xi) = \frac{sin(\xi)}{\xi} \tag{1.26}$$

We note $\xi_1 = \pi$, the first value of ξ with $\theta(\xi) = 0$. $\theta(\xi)$ is also defined as:

$$\theta(\xi) = \left(\frac{\rho(\xi)}{\rho_c}\right)^{\frac{1}{n}} = \frac{\rho(\xi)}{\rho_c} \tag{1.27}$$

With ρ_c the internal density of the star and ρ the density at a determined radius. ξ is defined as:

$$\xi = \sqrt{\frac{4\pi G}{K(n+1)}\rho_c^{(n-1)/n}} \times r = \sqrt{\frac{2\pi G}{K}} \times r = Ar \text{ (for } n = 1 \text{ and } A = \sqrt{\frac{2\pi G}{K}})$$

With K a proportionality constant, r the radius and G the gravitational constant.

From the previous equations we can write the stellar radius R, the total radius of the star, as:

$$R = \sqrt{\frac{K(n+1)}{4\pi G}}\rho_c^{(1-n)/2}\xi_1 = \sqrt{\frac{K}{2\pi G}} \times \xi_1 \text{ (for } n = 1) \tag{1.28}$$

We note that for $n = 1$ the radius does not depend of the central density.

Here as an example we use dimensionless units as $G = R = M = 1$ (for the other results we use CGS with $G = 6.674 \times 10^{-8} cm^3 g^{-1} s^{-2}$) We can compute K as:

$$K = \frac{R^2 2\pi G}{\xi_1^2} \tag{1.29}$$

Figure 1.1 give example of radius and the proportionality constant K.

Then we deduce the density function of r as :

$$\rho(\xi) = \frac{sin(A \times r)}{A \times r} \times \rho_c \text{ with } A = \sqrt{\frac{2\pi G}{K}}$$

As we know the total Mass M, the radius R and the gravitational constant G we can compute the central density as:

$$\rho_c = \frac{MA^3}{4\pi(sin(AR) - ARcos(AR))}$$

Then we normalize the results to fit $R = M = G = 1$: $K' = K/(R^2 G)$, $m_i' = m_i/M$, $h_i' = h_i/R$, $\vec{x_i}' = \vec{x_i}/R$

Generating Binary Neutron Stars initial data

The initial data are based on a cubic lattice within a sphere of radius R. The density function, based on radius, $\rho(\vec{r})$ is known using the result of the Lane-Emden equation (we use polytropic index $n = 1$ here). The mass associate to each particle i of the total N particles:

$$m_i = \frac{\rho(\vec{r_i})}{n_r} \text{ with } n_r = \frac{3N}{4\pi R^3}$$

The smoothing length is define constant and the same for all particles for all the simulation:

$$h = \frac{1}{2}\sqrt{\frac{3N_N}{4\pi n}}$$

Here we choose N_N, the average number of neighbors, to be 100.

Roche lobe problem: To create Hydrostatic Equilibrium Models we use a different equation of motion. This version use Roche Lobe:

$$\frac{d\vec{v_i}}{dt} = \frac{\vec{F}_i^{Grav}}{m_i} + \frac{\vec{F}_i^{Hydro}}{m_i} + \vec{F}_i^{Roche} - \frac{\vec{v_i}}{t_{relax}} \tag{1.30}$$

With $t_{relax} \leq t_{osc} \sim (G\rho)^{-1/2}$ and where $\vec{F}_i^{Roche}$ is:

$$\vec{F}_i^{Roche} = \mu(3+q)x_i\hat{\vec{x}} + \mu q y_i\hat{\vec{y}} - \mu z_i\hat{\vec{z}}$$

With μ to be determined (for us $\mu = 0.069$) and $q = \frac{M'}{M} = 1$ as the two polytropes have the same total mass. This is apply to each star to get the equilibrium and the simulate the tidal effect.

Darwin problem: This is the way we use to generate the final simulation. The equation of motion for the relaxation is now:

$$\frac{d\vec{v_i}}{dt} = \frac{\vec{F}_i^{Grav}}{m_i} + \frac{\vec{F}_i^{Hydro}}{m_i} + \vec{F}_i^{Rot} - \frac{\vec{v_i}}{t_{relax}} \tag{1.31}$$

With t_{relax} same as before and $\vec{F}_{Rot}$ defined by:

$$\vec{F}_{Rot} = \Omega^2(x_i\hat{\vec{x}} + y_i\hat{\vec{y}}) \tag{1.32}$$

With $\Omega = \sqrt{\frac{G(M+M')}{a^3}}$, $L_z = Q_{zz}\Omega$ and $Q_{zz} = \sum_i(x_i^2 + y_i^2)$. At $t = 0$ we compute the total angular moment L_z which stay constant. Using it during the relaxation we can compute Ω as: $\Omega = \frac{L_z}{Q_{zz}}$ just by recomputing Q_{zz}. Here the scheme is in N^2 but just for the relaxation step.

For this relaxation we use two stars generated as before, applying equation of motion 1.31. Using a as the distance between the two polytropes (Here $a = 2.9$ for $R = 1$) and $\hat{\vec{x}}$ going for the center of the first to the second star, and $\hat{\vec{z}}$ is like the rotation vector.

After the generation we are able to perform BNS coalescence. This provides us a code using SPH, self-gravity and gravitation.

1.4 Conclusion

The problem we presented in this part fulfill all the objectives for our metric. The computation wall is targeted via the physics and gravitation computation. The communication wall on the other hand is central regarding the exchanges needed for SPH and the fast multipole method. This problem is also very irregular for both cases. Indeed, the particles moves at every iteration

without any prediction on their new position and data. In the same time the locality impose a very high level of irregularity to reach efficiently all the neighbors of a specific particle. This implies the usage of a tree data structure, the work of our two metrics can be apply in this context.

As the SPH method is used in a large panel of fields from astrophysics to fluid mechanic, there are numerous related works. We can cite a code developed in the LANL, 2HOT [War13] that introduced the Hashed Oct Tree structure used in our implementation. There is also GADGET-2 [Spr05], GIZMO [Hop14] and the most recent publication is GASOLINE [WKQ17] based on PKDGRAV, a specific tree+gravity implementation. Several implementations already implement GPU code and tree construction and traversal, one can cite GOTHIC [MU17], presenting gravitational tree code accelerated using the latest Fermi, Kepler and Maxwell architectures. But a lot of GPU accelerated work still focused on fluid problems and not on astrophysical problems [HKK07, CDB+11]. We also note that these implementations focus on SPH problems and does not provide a general purpose and multi-physics framework like we intent to provide through FleCSPH and FleCSI.

Our implementation have not to be considered as a revolution for the SPH method itself, neither the use of accelerators. This study provide a code easy to use by domain scientists moving the complexity of distribution, load balancing and accelerator use to the back-end framework. This tool will allows us to push forward and provide different type of accelerator in the future keeping a full transparency for the user.

Bibliography

[AAA+17a] Benjamin P Abbott, R Abbott, TD Abbott, F Acernese, K Ackley, C Adams, T Adams, P Addesso, RX Adhikari, VB Adya, et al. Gw170814: A three-detector observation of gravitational waves from a binary black hole coalescence. *Physical review letters*, 119(14):141101, 2017.

[AAA+17b] Benjamin P Abbott, Rich Abbott, TD Abbott, Fausto Acernese, Kendall Ackley, Carl Adams, Thomas Adams, Paolo Addesso, RX Adhikari, VB Adya, et al. Gw170817: observation of gravitational waves from a binary neutron star inspiral. *Physical Review Letters*, 119(16):161101, 2017.

[AHA12] S Adami, XY Hu, and NA Adams. A generalized wall boundary condition for smoothed particle hydrodynamics. *Journal of Computational Physics*, 231(21):7057–7075, 2012.

[BG97] Rick Beatson and Leslie Greengard. A short course on fast multipole methods. *Wavelets, multilevel methods and elliptic PDEs*, 1:1–37, 1997.

[CDB+11] Alejandro C Crespo, Jose M Dominguez, Anxo Barreiro, Moncho Gómez-Gesteira, and Benedict D Rogers. Gpus, a new tool of acceleration in cfd: efficiency and reliability on smoothed particle hydrodynamics methods. *PloS one*, 6(6):e20685, 2011.

[DRZR17] Zili Dai, Huilong Ren, Xiaoying Zhuang, and Timon Rabczuk. Dual-support smoothed particle hydrodynamics for elastic mechanics. *International Journal of Computational Methods*, 14(04):1750039, 2017.

[GGRC+12] Moncho Gomez-Gesteira, Benedict D Rogers, Alejandro JC Crespo, Robert A Dalrymple, Muthukumar Narayanaswamy, and José M Dominguez. Sphysics–development of a free-surface fluid solver–part 1: Theory and formulations. *Computers & Geosciences*, 48:289–299, 2012.

[GM77] Robert A Gingold and Joseph J Monaghan. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly notices of the royal astronomical society*, 181(3):375–389, 1977.

[GM82] RA Gingold and JJ Monaghan. Kernel estimates as a basis for general particle methods in hydrodynamics. *Journal of Computational Physics*, 46(3):429–453, 1982.

[HKK07] Takahiro Harada, Seiichi Koshizuka, and Yoichiro Kawaguchi. Smoothed particle hydrodynamics on gpus. In *Computer Graphics International*, pages 63–70. SBC Petropolis, 2007.

[Hop14] Philip F Hopkins. Gizmo: Multi-method magneto-hydrodynamics+ gravity code. *Astrophysics Source Code Library*, 2014.

[LL59] L. D. Landau and E. M. Lifshitz. *Fluid mechanics*. 1959.

[LP91] Larry D Libersky and AG Petschek. Smooth particle hydrodynamics with strength of materials. In *Advances in the free-Lagrange method including contributions on adaptive gridding and the smooth particle hydrodynamics method*, pages 248–257. Springer, 1991.

[LSR16] SJ Lind, PK Stansby, and Benedict D Rogers. Incompressible–compressible flows with a transient discontinuous interface using smoothed particle hydrodynamics (sph). *Journal of Computational Physics*, 309:129–147, 2016.

[Luc77] Leon B Lucy. A numerical approach to the testing of the fission hypothesis. *The astronomical journal*, 82:1013–1024, 1977.

[MG83] J.J Monaghan and R.A Gingold. Shock simulation by the particle method sph. *Journal of Computational Physics*, 52(2):374 – 389, 1983.

[MU17] Yohei Miki and Masayuki Umemura. Gothic: Gravitational oct-tree code accelerated by hierarchical time step controlling. *New Astronomy*, 52:65–81, 2017.

[RDZR16] Huilong Ren, Zili Dai, Xiaoying Zhuang, and Timon Rabczuk. Dual-support smoothed particle hydrodynamics. *arXiv preprint arXiv:1607.08350*, 2016.

[Ros09] Stephan Rosswog. Astrophysical smooth particle hydrodynamics. *New Astronomy Reviews*, 53(4):78 – 104, 2009.

[Sed46] Leonid I Sedov. Propagation of strong shock waves. *Journal of Applied Mathematics and Mechanics*, 10:241–250, 1946.

[Sod78] Gary A Sod. A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws. *Journal of Computational Physics*, 27(1):1 – 31, 1978.

[Spr05] Volker Springel. The cosmological simulation code gadget-2. *Monthly Notices of the Royal Astronomical Society*, 364(4):1105–1134, 2005.

[War13] Michael S Warren. 2hot: an improved parallel hashed oct-tree n-body algorithm for cosmological simulation. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, page 72. ACM, 2013.

[WKQ17] James W Wadsley, Benjamin W Keller, and Thomas R Quinn. Gasoline2: a modern smoothed particle hydrodynamics code. *Monthly Notices of the Royal Astronomical Society*, 471(2):2357–2369, 2017.

[YB11] Rio Yokota and Lorena A Barba. Treecode and fast multipole method for n-body simulation with cuda. In *GPU Computing Gems Emerald Edition*, pages 113–132. Elsevier, 2011.