공학석사 학위논문

Development of GPU-based Adaptive Particle Refinement Algorithm for Smoothed Particle Hydrodynamics

완화입자유체동역학을 위한 GPU 기반
입자 분할/병합 알고리즘 개발

2021 년 8 월

서울대학교 대학원
에너지시스템공학부
김 도 현

Development of GPU-based Adaptive Particle Refinement Algorithm for Smoothed Particle Hydrodynamics

완화입자유체동역학을 위한 GPU 기반
입자 분할/병합 알고리즘 개발

지도교수 김 응 수

이 논문을 공학석사 학위논문으로 제출함
2021 년 8 월

서울대학교 대학원
에너지시스템공학부
김 도 현

김도현의 석사 학위논문을 인준함
2021 년 8 월

위 원 장 조 형 규

부위원장 김 응 수

위 원 정 재 호

Abstract

Development of GPU-based Adaptive Particle Refinement Algorithm for Smoothed Particle Hydrodynamics

Do Hyun Kim

Department of Energy System Engineering

The Graduate School

Seoul National University

Recent nuclear safety issues are not only limited to thermal-hydraulics, but consist of complex phenomena including fuel melt, materials, chemical reactions, and multi-phase flow. The traditional reactor safety analysis is mainly based on Computational Fluid Dynamics(CFD) with Eulerian grid-based methods. But recently, Lagrangian particle-based methods are also being actively studied, due to their well-known advantages in handling free surface, interfacial flow, and large deformation. Smoothed Particle Hydrodynamics (SPH) is one of the representative Lagrangian-based methods in which the fluid system is represented as the finite number of particles.

In particle-based CFD, high resolution generally guarantees high-accuracy results, but it causes a high computational load as the number of particles in the domain increases. Most existing particle-based analysis codes adopt a single-resolution method using particles of the same size in the entire computational domain. However, in the case that requires different levels of particle resolution depending on the flow region, such as turbulence, boiling/condensation, and shock wave analysis, this method may create unnecessary computational load or reduce the computational accuracy. Therefore, in order to improve this, it is necessary to introduce a multi-resolution analysis that can control the size of particles locally within the computational domain.

Accordingly, in this study, an adaptive particle refinement (APR) method was developed and implemented in SPH, in a form suitable for GPU parallel computation to accelerate the model. The basic concept of the APR methodology is to use different resolutions in localized regions within the computational domain by splitting or merging particles under specific conditions during simulation. Multiple particle resolutions can be implemented by splitting or merging the SPH particles under certain conditions (position, volume, velocity gradient, etc.). However, in the methods used in previous studies, the velocity of the merged particle is determined only by the momentum conservation equation in the process of merging. Since this method conserves mass and momentum well but does not conserve the kinetic energy of particles, a new merging model for kinetic energy conservation is proposed in this study. In addition, when the smoothing length of particles is changed during the APR process, the accuracy of calculation may be reduced at the interface of the resolution where particles of

different sizes interact. A continuous smoothing length change model to improve this was also proposed.

Due to the nature of GPU parallel computation, when multiple threads simultaneously access and perform operations on the same memory, the order of operations between threads is twisted, resulting in a race condition that yields different results than expected. When the APR methodology is parallelized, such condition occurs in the process of storing newly generated particles, resulting in a collision of memory addresses of generated particles. To solve this problem, a locking algorithm that can prevent inter-thread computational interference was implemented using the atomic operation provided by the CUDA C language, and the algorithm was optimized to prevent computational speed degradation due to excessive serialization.

Model validation and performance evaluation were performed with the applied APR model. From the analysis results of hydrostatic pressure formation, pipe flow, dam collapse, and Karman vortex, it was confirmed that the APR model developed can stably implement multi-resolution, and showed high accuracy and computational efficiency. In addition, application to multi-fluid and multi-phase flow was performed through jet break up and air bubble rising simulation, and quantitatively compared with experimental data. As a result, it was confirmed that the model well simulates the real phenomena, and the computational efficiency was greatly improved by controlling the number of particles.

In this study, a multi-resolution analysis system was constructed within the SPH methodology by developing a particle refinement model and optimizing it using a GPU. This is significant in that it can provide a solution to the problem of excessive computational load due to the increase in resolution that the existing single-resolution particle-based analysis system inevitably had. It is expected to contribute to the analysis of complex flows that require multiple particle resolutions within the phenomenon, such as severe accidents in a nuclear reactor.

Keywords

Smoothed Particle Hydrodynamics(SPH), Adaptive Particle Refinement(APR), Multi-resolution, GPU parallelization, Race condition, Atomic operation

Student Number: 2019-27978

List of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background and Motivation

Recent nuclear safety issues are not limited to thermal-hydraulics, but consist of complex phenomena including fuel melt, materials, chemical reactions, and multi-phase flow. The traditional reactor safety analysis is mainly based on Computational Fluid Dynamics(CFD) with Eulerian grid-based methods. But recently, Lagrangian particle-based methods are also being actively studied, due to their well-known advantages in handling free surface, interfacial flow, and large deformation. Smoothed Particle Hydrodynamics (SPH) is one of the representative Lagrangian-based methods in which the fluid system is represented as the finite number of particles.

The conventional SPH method use a uniform size of particles within the computational domain. It is an intuitive and convenient, but it also has drawbacks. In a complex fluid motion, different particle resolutions may be required for flow regions. For example, in Fuel Coolant Interaction(FCI), the region where molten corium meets and reacts with coolant contains very complex phenomena including heat transfer, boiling, fragmentation, etc., and requires precise analysis.

On the other hand, at the lower cavity region, which does not interact directly with the melt, the phenomenon is relatively much less complex and of little interest. For high-accuracy results, the resolution of particles must be adjusted to the former region, so using the constant particle size, in this case, is a huge waste of computational power. A reasonable approach would be to use multiple resolutions, using higher resolution at the surface where two fluids meet, and lower resolution at the lower cavity region.

From this background, Adaptive Particle Refinement (APR), which is a method of controlling the size of particles locally within the computational domain, has been developed in the SPH field to enable multi-resolution analysis. By adjusting the particle size, APR can improve the local accuracy of the simulation, using smaller sizes of particles in complex flow regions, while increasing overall computational efficiency by reducing the total number of particles. The strength of the model is expected to be highlighted in large-scale simulations with a large number of required particles or simulations requiring very small particle size and precise analysis. Therefore, it is essential to introduce multi-resolution analysis into the SPH method, and the development of the APR model could be a powerful solution.

1.2 Previous Studies

APR method has been actively studied in recent years, and the methodology has been well established through previous studies. Various refinement patterns and criteria were tested to find the optimal methodology. Some of the main contents of the studies are described below.

Vacondio et al. (2013) performed error analysis on various particle refinement patterns including triangular, square, and hexagonal shape in 2D. In the same group, a study on 3D refinement patterns was also conducted, and various patterns such as cubic, dodecahedron, and icosahedron were tested. (Vacondio et al., 2016) From the result of the error analysis, hexagonal pattern and icosahedron patterns were selected in 2D and 3D, respectively, and model performance was evaluated through benchmark simulations.

Barcarolo et al. (2014) used square refinement pattern and proposed a new merging procedure that allows erasing the particles that were created within the region of interest once they are no longer needed. In model validation, refinement was introduced up to three stages, so that one coarse particle was split into 64 fine particles at the final stage. The model stably estimated the pressure field of the jet impacting simulation, implementing 4 stages of particle resolution.

Liu et al. (2017) conducted a parametric study to determine the particle spacing ratio and smoothing length ratio in the splitting process, to achieve optimal accuracy. Square refinement pattern and particle position-based criterion were used, which are the most widely used and easily applicable. From the dam breaking and Karman vortex simulation, the model was evaluated in terms of total computing time.

Sun et al. (2019) used the APR method to handle the compressible flow and analyzed underwater explosions by splitting and merging particles according to the volume-based criterion. In the simulation, when particle volume reaches the maximum limit, which needs to be pre-defined, the particle splits into four fine particles arranged in a square shape. On the other hand, when the volume of a particle is reduced to a minimum limit, two particles merge into a single coarse

particle. In merging process, the inter-particle distance between two particles has been considered, to guarantee that only sufficiently close particles are merged.

Recently, a new concept of particle refinement was proposed by Hu (2020), which is named as Hybrid Particle Interacting technique. The main difference of the idea is to set the transition zone at the resolution interface, where the coarse particles and fine particles coexist. Particles only interact with neighbor particles of the same size, and this leads to higher accuracy by inherently satisfying the unity condition of SPH smoothing kernel functions.

Despite the efforts from previous studies, there remain some challenges to be solved, which are directly related to the objectives of this study. (Vacondio et al., 2020) First, error minimization. Even though it is impossible to avoid the introduction of error, the model should guarantee that the error has been minimized. In particular, the APR method is a method of changing the spatial resolution, so that it is important to minimize the error at the interface where resolution is changed. Second, implementation with high-performance computing devices. Even the APR is a method of improving the efficiency of the code, the model itself needs some form of hardware acceleration. Finally, robust schemes for all applications. To date, only simple cases of applications have been simulated using APR model. The development of a robust scheme is needed for the general availability of the model.

1.3 Objectives

The purpose of this study is to develop an improved Adaptive Particle Refinement model in in-house SPH code, SOPHIA. The research objectives in

detail are as follows.

1) Development of new sub-models to improve the global and local accuracy of the APR model. A new particle merging scheme was introduced, and models to improve the accuracy at the resolution interface were presented.
2) GPU-parallelization of APR model in order to maximize the performance. The race condition that occurred in this process was resolved.
3) Validation and application of the developed model, and performance evaluation.

This thesis is composed as follows; Chapter 2 describes the basic concept of SPH method and its governing equations. Chapter 3 explains the APR model developed in this study, and the process of GPU-parallelization would be described in detail. In chapter 4, results of benchmark simulations for model validation are introduced, and the application of the model is presented. Chapter 5 summarizes and concludes this study.

Chapter 2

Smoothed Particle Hydrodynamics

2.1 Smoothed Particle Hydrodynamics (SPH)

Smoothed Particle Hydrodynamics (SPH) is a meshless Lagrangian method developed in the astrophysical area by Gingold and Monaghan(1977), and it has been increasingly used for simulating fluid flows. The basic idea of SPH is to represent a fluid system as a set of a finite numbers of particles, which carry individual physical properties, such as mass, density, temperature, and velocity. (Figure 2.1) Each particle interacts with its neighboring particles to solve the governing equations that are discretized as SPH formulation. Thanks to its meshless nature, the SPH method has great advantages in handling free surfaces, flows with large deformations, and multiphase flows, compared to the mesh-based Eulerian methods. In this section, the basic concept of SPH and SPH particle approximation are described, including kernel function properties.

2.1.1 SPH Particle Approximation

In SPH, the physical quantities of the particles are defined as the weighted

average of the surrounding particles. Mathematically, an arbitrary function can be expressed in the integral form using the delta function as follows.

$$f(r) = \int_{\Omega} f(r')\delta(r - r')dr' \tag{2.1}$$

where r denotes the point in volume Ω, and δ denotes the Dirac delta function. Delta function is a discrete function that has infinite value at a specific point and zero at the rest, whose integration over the entire region is equal to one.

$$\delta_{\varepsilon}(x) = \lim_{\varepsilon \to 0} \begin{cases} 0 & x < -\frac{\varepsilon}{2} \\ \frac{1}{\varepsilon} & -\frac{\varepsilon}{2} < x < \frac{\varepsilon}{2} \\ 0 & x > \frac{\varepsilon}{2} \end{cases} \tag{2.2}$$

$$\int_{-\infty}^{+\infty} \delta(x)dx = \int_{-\varepsilon/2}^{\varepsilon/2} \frac{1}{\varepsilon} dx = 1 \tag{2.3}$$

To approximate the delta function, kernel function, which is a continuous function with properties similar to the delta function, is used.

$$f(r) = \int_{\Omega} f(r')W(r - r', h)dr' \tag{2.4}$$

By discretizing the above equation, the basic formulation of SPH particle approximation is derived.

$$f_i(r) = \sum_j f_j W(\boldsymbol{r}_i - \boldsymbol{r}_j, h) \frac{m_j}{\rho_j} \tag{2.5}$$

where the subscript i denotes the center particle, and j denotes the adjacent particles, respectively. $W(\boldsymbol{r}_i - \boldsymbol{r}_j, h)$ stands for the kernel function, which is a function of inter-particle distance of i and j particle, and smoothing length, h. As shown in Figure 2.2, smoothing length defines the effective range of the kernel function.

2.1.2 Smoothing Kernel Function

Smoothing kernel functions play important roles in the accuracy and stability of the SPH simulation. Since the kernel function is used to approximate the delta function, it should follow the mathematical properties of the delta function. (Table 2.1)

First, the kernel function should have the largest value at the center and gradually decrease smoothly as it moves away from it. Since the kernel function should be defined only in the support domain, it should converge to zero.

$$W(\boldsymbol{r}_i - \boldsymbol{r}_j, h) = 0, \qquad where \ \ |r_i - r_j| > \kappa h \tag{2.6}$$

κh means the support domain of the kernel function, where h is smoothing length and κ is a constant, which was set to be 2.0 in this study. This is so-called compact support condition. Also, the value of integrating the kernel function over the entire support domain should be 1.

$$\int_{\Omega} W(r-r')dr' = 1 \tag{2.7}$$

This is called unity condition, where Ω is a computational domain. Other conditions are listed in Table 2.1.

There are several kinds of kernel functions that satisfy the above conditions. In this study, the Wendland C2 kernel function was adopted, due to its advantages of high accuracy and low instability. (Dehnen and Aly, 2012) Kernel function and its derivative are as follows.

$$W(r,h) = \begin{cases} \frac{5}{4(2h)}\left(1-\frac{r}{2h}\right)^3\left(1+3\frac{r}{2h}\right) & for\ 1D \\ \frac{7}{\pi(2h)^2}\left(1-\frac{r}{2h}\right)^4\left(1+4\frac{r}{2h}\right) & for\ 2D \\ \frac{21}{2\pi(2h)^3}\left(1-\frac{r}{2h}\right)^4\left(1+4\frac{r}{2h}\right) & for\ 3D \end{cases} \tag{2.8}$$

$$\nabla W(r,h) = \begin{cases} \frac{5}{4(2h)^2}\left(-12\left(1-\frac{r}{2h}\right)^2\frac{r}{2h}\right) & for\ 1D \\ \frac{7}{\pi(2h)^3}\left(-20\left(1-\frac{r}{2h}\right)^3\frac{r}{2h}\right) & for\ 2D \\ \frac{21}{2\pi(2h)^4}\left(-20\left(1-\frac{r}{2h}\right)^3\frac{r}{2h}\right) & for\ 3D \end{cases} \tag{2.9}$$

2.1.3 SPH approximation of derivatives

Derivation of the spatial derivatives in SPH starts from taking differential operator to Equation (2.4). (Monaghan, 1992) Using integration by parts, the

derivative term can be expressed by follows.

$$\nabla \cdot f(r) = \int_{\Omega} \nabla \cdot \big(f(r')W(r-r',h)\big)dr' - \int_{\Omega} f(r')\nabla W(r-r',h)dr' \quad (2.10)$$

The first term of the RHS of the Equation 2.9 can be expressed by the surface integral from the divergence theorem.

$$\nabla \cdot f(r) = \int_{S} f(r')W(r-r',h)\vec{n}dS - \int_{\Omega} f(r')\nabla W(r-r',h)dr' \quad (2.11)$$

By the compact support condition previously mentioned, kernel function is only defined within the support domain, making the value of surface integral zero. Then, through a similar process as in the previous section, the derivative of the field function $f(r)$ can be expressed in SPH formulation as follows.

$$\nabla \cdot f_i(r) = \sum_j f_j \nabla W(\boldsymbol{r}_i - \boldsymbol{r}_j, h)\frac{m_j}{\rho_j} \quad (2.12)$$

From the above equation, various forms of the SPH differential operators can be derived as Table 2.2.

2.2 SPH governing equations

The basic governing equations for SPH are the continuity equation (mass conservation) and the Navier-Stokes equation (momentum conservation), and can

be expressed in Lagrangian form as follows.

$$\frac{d\rho}{dt} = -\rho\nabla \cdot \boldsymbol{u} \tag{2.13}$$

$$\frac{d\boldsymbol{u}}{dt} = -\frac{\nabla p}{\rho} + \nu\nabla^2\boldsymbol{u} + \boldsymbol{g} + \boldsymbol{f}_{ext} \tag{2.14}$$

where $\boldsymbol{u}, p, \nu, \boldsymbol{g}$ and $\boldsymbol{f}_{ext}$ denote velocity, pressure, kinematic viscosity, gravitational acceleration and external body force, respectively. In SPH method, above equations are approximated in SPH formulation. The governing equations in SPH formula are summarized in Table 2.3.

2.2.1 Mass conservation

In SPH, density of the fluid can be calculated in either two ways: By continuity equation (Equation 2.13), and direct mass summation. (Monaghan, 1992)

$$\left(\frac{d\rho}{dt}\right)_i = \rho_i \sum_j \frac{m_j}{\rho_j} (\boldsymbol{u_i} - \boldsymbol{u_j}) \cdot \nabla_i W_{ij} \tag{2.15}$$

$$\rho_i = \rho_{ref,i} \sum_j \frac{m_j}{(\rho_{ref})_j} W_{ij} \tag{2.16}$$

where $W_{ij} = W(\boldsymbol{r}_i - \boldsymbol{r}_j, h)$, and $\nabla_i W_{ij} = \partial W(\boldsymbol{r}_i - \boldsymbol{r}_j, h)/\partial \boldsymbol{r}_i$. ρ_{ref} denotes the reference density of the fluid. Equation (2.16) is a formulation based on normalized density, and was proposed to handle the discontinuous density field

in multi-phase flows. (Jo et al., 2019)

2.2.2 Momentum conservation

For the conservation of momentum, Navier-Stokes equation (Equation 2.14) can be expressed in form of SPH approximation. The first term of the equation represents the force induced by pressure gradient, which can be discretized as follows using the SPH gradient operator described in the previous section.

$$\left(\frac{d\boldsymbol{u}}{dt}\right)_{fp,i} = -\sum_j m_j \left(\frac{p_i + p_j}{\rho_i \rho_j}\right) \nabla_i W_{ij} \tag{2.17}$$

The pressure force model used in this study can handle discontinuous density field by calculating the pressure gradient based on the particle volume. (Liu et al., 2003)

The second term of the momentum equation represents the viscous force of the fluid and expressed in SPH formulation as below.

$$\left(\frac{d\boldsymbol{u}}{dt}\right)_{fv,i} = \sum_j \frac{4m_j}{\rho_i \rho_j} \frac{\mu_i \mu_j}{(\mu_i + \mu_j)} (\boldsymbol{u}_i - \boldsymbol{u}_j) \frac{\boldsymbol{r}_{ij} \cdot \nabla W_{ij}}{\left(|\boldsymbol{r}_{ij}|^2 + \eta^2\right)} \tag{2.18}$$

where μ denotes the dynamic viscosity of the fluid and $\boldsymbol{r}_{ij} = \boldsymbol{r}_i - \boldsymbol{r}_j$. This formulation of viscous force was proposed by Cleary (1998), and it is suitable for multi-phase simulation where the viscosity of fluids are different.

2.2.3 Equation of State

The SPH code used in this study, SOPHIA, is based on the weakly-compressible SPH(WCSPH), which assumes a weak compressibility of the fluid. In WCSPH method, the pressure is explicitly calculated as a function of density using the Tait equation below. (Monaghan, 1994)

$$p_i = \frac{c_0^2 \rho_{ref,i}}{\gamma}\left[\left(\frac{\rho_i}{\rho_{\mathrm{ref,i}}}\right)^{\gamma} - 1\right] \tag{2.19}$$

where c_0 and γ denote the sound speed and polytrophic constant, respectively. By adjusting the value of γ, which is recommended to set as $1 \leq \gamma \leq 7$, the sensitivity of the pressure calculation is adjusted.

2.2.4 Surface tension

This study used the surface tension force based on the Continuum Surface Force(CSF) model. The surface tension force is in CSF model estimated as the product of the surface curvature and surface normal. (Morris, 2000)

$$\left(\frac{d\boldsymbol{u}}{dt}\right)_i^{CSF} = -\frac{\sigma_i}{\rho_i}\kappa_i \boldsymbol{n}_i \tag{2.20}$$

$$\boldsymbol{n}_i = \sum_j \frac{1}{V_i}\left(V_i^2 + V_j^2\right) c_{ij} \nabla W_{ij} \tag{2.21}$$

$$\kappa_i = -d\frac{\sum_j V_j\left(\hat{\boldsymbol{n}}_\iota - \varphi_{ij}\hat{\boldsymbol{n}}_j\right)\cdot\nabla W_{ij}}{\sum_j V_j|r_{ij}||\nabla W_{ij}|} \tag{2.22}$$

$$c_{ij} = \begin{cases} 0 & for\ \ f_i = f_j \\ \dfrac{\rho_i}{\rho_i + \rho_j} & for\ \ f_i \neq f_j \end{cases}, \qquad \varphi_{ij} = \begin{cases} 1 & for\ \ f_i = f_j \\ -1 & for\ \ f_i \neq f_j \end{cases} \tag{2.23}$$

where $\sigma, \kappa, \boldsymbol{n}$ and d denote a surface tension coefficient, the curvature, surface normal vector, and dimension $(d = 1,2,3)$. $\hat{\boldsymbol{n}}_\iota$ denotes the unit surface normal vector. However, it was confirmed that the above formulation of surface normal vector causes unphysical curvature at the phase interface, and the newly proposed model was implemented. (Jo et al., 2019)

$$\hat{\boldsymbol{n}}_\iota = \sum_j \frac{m_j}{\rho_j}(c_j - c_j)\nabla W_{ij} \tag{2.24}$$

$$c_i = \frac{\sum_j \varphi_{ij} W_{ij}}{\sum_j W_{ij}}, \qquad \varphi_{ij} = \begin{cases} 1 & for\ \ f_i = f_j \\ 0 & for\ \ f_i \neq f_j \end{cases} \tag{2.25}$$

2.3 SPH Algorithm

Figure 2.3 shows a basic algorithm of the SPH code. First, the input file is read, and the initial conditions of the particles are stored. Then, based on the particle position, the Nearest-Neighboring Particle Search (NNPS) is conducted. Since the SPH model solves the governing equations for each particle using the properties of the neighboring particles, reducing the search range of each particle through the NNPS is an essential part of the overall algorithm, in terms of computational efficiency. (Figure 2.4) After the NNPS is done, the density,

pressure, and forces of each particle are estimated sequentially using the SPH governing equations described in the previous section. Finally, the velocity and position of each particle are updated using the calculated force, and the above process from NNPS to time update is repeated during the simulation time. For time integration, SOPHIA code uses a modified predictor-corrector scheme as below.

$$Predictor \quad \begin{cases} \boldsymbol{u}^{p}_{t+\frac{\Delta t}{2}} = \boldsymbol{u}_t + \dfrac{\Delta t}{2}\left(\dfrac{d\boldsymbol{u}}{dt}\right)_{t-\frac{\Delta t}{2}} \\ \boldsymbol{r}^{p}_{t+\frac{\Delta t}{2}} = \boldsymbol{r}_t + \dfrac{\Delta t}{2}\left(\dfrac{d\boldsymbol{r}}{dt}\right)_{t-\frac{\Delta t}{2}} \\ \rho^{p}_{t+\frac{\Delta t}{2}} = \rho_t + \dfrac{\Delta t}{2}\left(\dfrac{d\rho}{dt}\right)_{t-\frac{\Delta t}{2}} \end{cases} \tag{2.26}$$

$$Corrector \quad \begin{cases} \boldsymbol{u}^{c}_{t+\Delta t} = \boldsymbol{u}_t + \Delta t\left(\dfrac{d\boldsymbol{u}}{dt}\right)_{t+\frac{\Delta t}{2}} \\ \boldsymbol{r}^{c}_{t+\Delta t} = \boldsymbol{r}_t + \Delta t\left(\dfrac{d\boldsymbol{r}}{dt}\right)_{t+\frac{\Delta t}{2}} \\ \rho^{c}_{t+\Delta t} = \rho_t + \Delta t\left(\dfrac{d\rho}{dt}\right)_{t+\frac{\Delta t}{2}} \end{cases} \tag{2.27}$$

where t and Δt denote time and time step, respectively. In predictor step, physical variables are explicitly calculated using half time step. After solving governing equations, the variables are re-integrated over the full-time step using the updated time derivatives in correction step.

Table 2.1 Conditions of Kernel functions

Kernel Conditions	
Unity condition	$\int_{-\infty}^{\infty} W(\boldsymbol{r}_{ij},\, h)dr = 1$
Symmetry condition	$W(\boldsymbol{r}_{ij},\, h) = W(-\boldsymbol{r}_{ij},\, h)$
Delta function Property	$\lim_{h\to 0} W(\boldsymbol{r}_{ij},\, h) = \delta(\boldsymbol{r}_{ij})$
Compact support	$W(\boldsymbol{r}_{ij},\, h) = 0,\ for\ \lvert\boldsymbol{r}_{ij}\rvert > kh$
Monotonic Decrease	$\frac{\partial}{\partial \boldsymbol{r}} W(\boldsymbol{r},\, h) < 0$
Positive value	$W(r_{ij},\, h) \geq 0$
Sufficiently smooth	

Table 2.2 SPH gradient operators

Gradient operator	SPH formulation
Gradient	$(\nabla f)_i = \sum_j \frac{m_j}{\rho_j} f_i \nabla W_{ij}$
	$(\nabla f)_i = \sum_j \frac{m_j}{\rho_j} (f_i - f_j) \nabla W_{ij}$
	$(\nabla f)_i = \rho_i \sum_j m_j \left(\frac{f_i}{{\rho_i}^2} + \frac{f_j}{{\rho_j}^2} \right) \nabla W_{ij}$
Divergence	$(\nabla \cdot \boldsymbol{f})_i = \rho_i \sum_j m_j \left(\frac{\boldsymbol{f_i}}{{\rho_i}^2} + \frac{\boldsymbol{f}_j}{{\rho_j}^2} \right) \cdot \nabla W_{ij}$
Laplacian	$(\nabla^2 f)_i = \sum_j \frac{2m_j}{\rho_j} (f_i - f_j) \frac{\boldsymbol{r}_{ij}}{\lvert r_{ij} \rvert^2} \cdot \nabla W_{ij}$

Table 2.3 SPH Governing Equations

SPH governing equations	
Mass summation	$\rho_i = \rho_{ref,i} \sum_j \frac{m_j}{(\rho_{ref})_j} W_{ij}$
Continuity equation	$\left(\frac{d\rho}{dt}\right)_i = \rho_i \sum_j \frac{m_j}{\rho_j} (\boldsymbol{u}_i - \boldsymbol{u}_j) \cdot \nabla W_{ij}$
Pressure force	$\left(\frac{d\boldsymbol{u}}{dt}\right)_{fp,i} = -\sum_j m_j \left(\frac{p_i + p_j}{\rho_i \rho_j}\right) \nabla_i W_{ij}$
Laminar viscous	$\left(\frac{d\boldsymbol{u}}{dt}\right)_{fv,i} = \sum_j \frac{4m_j}{\rho_i \rho_j} \frac{\mu_i \mu_j}{(\mu_i + \mu_j)} (\boldsymbol{u}_i - \boldsymbol{u}_j) \frac{\boldsymbol{r}_{ij} \cdot \nabla W_{ij}}{\left(\lvert \boldsymbol{r}_{ij} \rvert^2 + \eta^2\right)}$
Surface tension	$\left(\frac{d\boldsymbol{u}}{dt}\right)_{fs,i} = -\frac{\sigma}{\rho_i} \kappa_i (\nabla c)_i$
Tait equation	$p_i = \frac{c_0^2 \rho_{\mathrm{ref,i}}}{\gamma} \left[\left(\frac{\rho_i}{\rho_{\mathrm{ref,i}}} \right)^{\gamma} - 1 \right]$

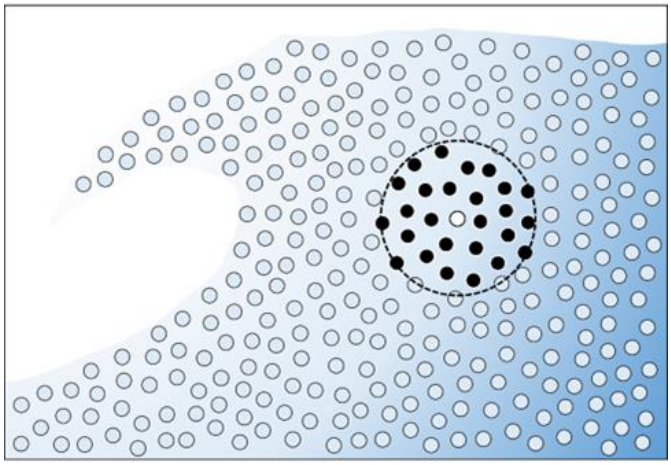


Figure 2.1 The basic concept of SPH

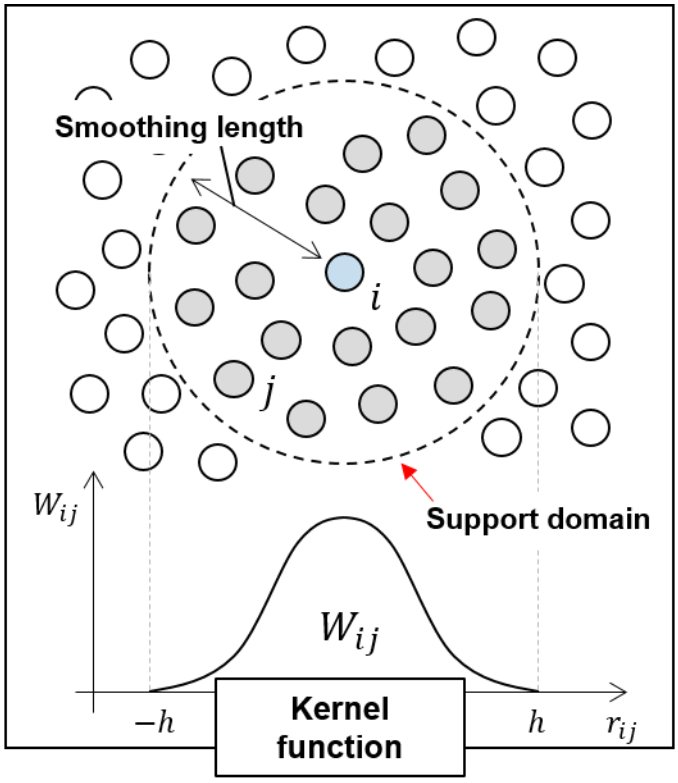


Figure 2.2 SPH particle approximation

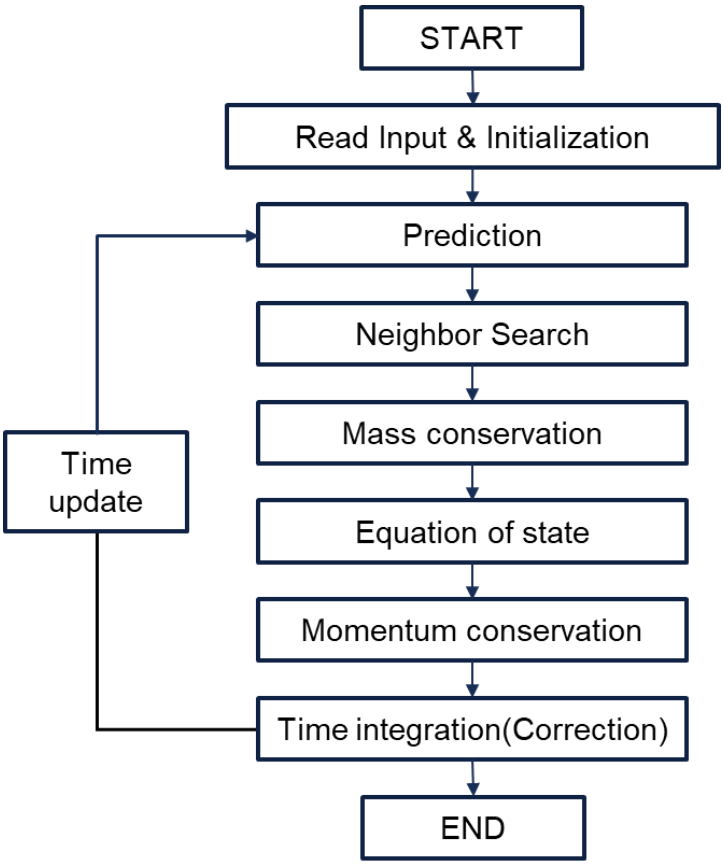


Figure 2.3 Basic algorithm of the SPH code

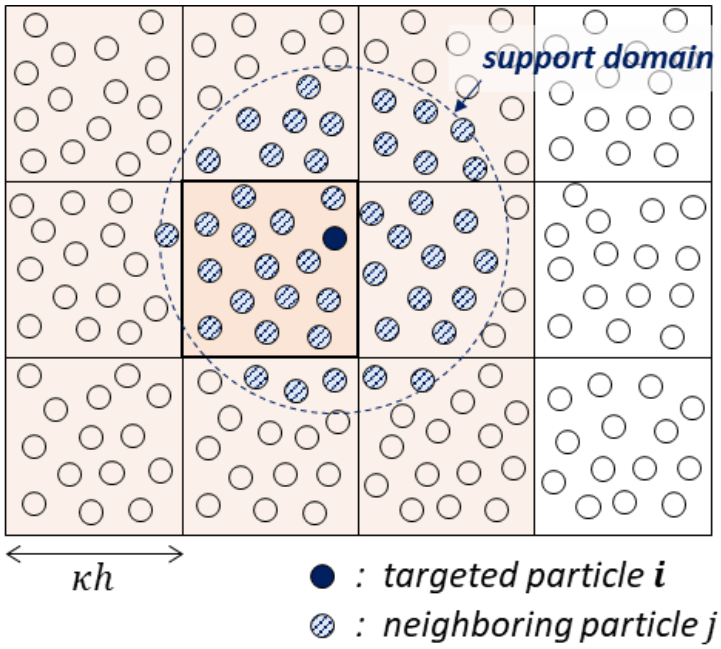


Figure 2.4 Nearest Neighboring Particle Search (NNPS)

Chapter 3

Adaptive Particle Refinement

3.1 Adaptive Particle Refinement (APR)

Conventional SPH simulations use the uniform size of particles in total computational domain, and higher particle resolution, which means using smaller size particles, generally guarantees high-accuracy results. However, when solving complex flows such as turbulence, shockwave, and multi-phase flows, different particle resolutions may be required within the computational domain. In this case, using constant particle size may cause unnecessary computational load or decrease the local accuracy. From this background, Adaptive Particle Refinement (APR), which is a method of controlling the local resolution by splitting or merging individual particles, has been developed in SPH field to enable the multi-resolution analysis. In this section, the basic concept and methodology of particle refinement are described, and new ideas to improve the model accuracy are presented.

3.1.1 Basic concept of APR

The basic concept of APR is to split and merge SPH particles in certain criteria, using different sizes of particles in a single computational domain. (Figure 3.1) In previous studies, various refinement patterns and APR criteria were presented and it is summarized in Table 3.1. In this study, square and cubic refinement patterns and particle position-based criteria were selected, which were commonly used in previous studies.

3.1.2 APR methodologies

In 2D splitting process, particles that satisfy the splitting condition are split into four fine particles. Fine particles are placed at each vertex of the square shape, and the inter-particle distance is reduced to be half of the initial particle distance. (Figure 3.2) To satisfy mass conservation, the mass and volume of each fine particle are defined to be 1/4 of the coarse particle. The relation between the variables of fine particles and their coarse particle is written as follows, defined to satisfy the conservation of momentum and kinetic energy.

$$
\begin{aligned}
x_f^i = x_c \pm \frac{\sqrt{V_c}}{4}; \qquad & y_f^i = y_c \pm \frac{\sqrt{V_c}}{4}; \\
\rho_f = \rho_c; \qquad & p_f = p_c; \qquad \boldsymbol{u_f} = \boldsymbol{u_c}; \\
m_f = \frac{1}{4}\, m_c; \qquad & V_f = \frac{1}{4}\, V_c; \qquad h_f = \gamma h_c;
\end{aligned}
\tag{3.1}
$$

Subscript f and c represent fine particle and daughter particle respectively, and superscript i stands for the number of the fine particle, from 1 to 4. γ denotes the smoothing length coefficient, which will be further covered later in chapter

3.2.3. In this study, γ was set to be 0.5 or 1.0.

In 3D, one coarse particle splits into 8 fine particles, forming a cubic arrangement. (Figure 3.3) Similar to 2D, properties of fine particles are determined as follows to satisfy the conservation of mass, momentum, and energy.

$$x_f^i = x_c \pm \frac{\sqrt[3]{V_c}}{4}; \qquad y_f^i = y_c \pm \frac{\sqrt[3]{V_c}}{4}; \qquad z_f^i = z_c \pm \frac{\sqrt[3]{V_c}}{4};$$
$$\rho_f = \rho_c; \qquad p_f = p_c; \qquad \boldsymbol{u_f} = \boldsymbol{u_c}; \tag{3.2}$$
$$m_f = \frac{1}{8} m_c; \qquad V_f = \frac{1}{8} V_c; \qquad h_f = \gamma\, h_c;$$

Particle merging in 2D is conducted exactly in the reverse order of splitting, except velocity calculation. (Figure 3.4) For the velocity of the coarse particle after merging, it is calculated as the vector sum of the four fine particles. The spatial position of the coarse particle is defined as the center of mass of the fine particles which are involved in the merging process. Coarse particle properties are determined to satisfy the mass and momentum conservation.

$$x_c = \frac{1}{4}\sum_{i=1}^{4} x_f^i; \qquad y_c = \frac{1}{4}\sum_{i=1}^{4} y_f^i;$$
$$\rho_c = \frac{1}{4}\sum_{i=1}^{4} \rho_f^i; \qquad p_c = \frac{1}{4}\sum_{i=1}^{4} p_f^i; \qquad \boldsymbol{u_c} = \frac{1}{4}\sum_{i=1}^{4} \boldsymbol{u_f^i}; \tag{3.3}$$
$$m_c = 4m_f; \qquad V_c = 4V_f; \qquad h_c = \frac{1}{\gamma} h_f;$$

In the case of merging, the particles to be merged must be located sufficiently

close for physical feasibility. Thus, merging occurs only when the following condition is satisfied.

$$\left|r_{ij}\right| \leq \sqrt{3V_i} \tag{3.4}$$

where $\left|r_{ij}\right|$ denotes the distance between center particle *i* and target particle *j*, and V_i is the volume of the center particle.

In 3D, since it is difficult for all eight particles to meet the inter-particle distance condition (Eq. 3.4), eight fine particles are sequentially merged in pairs, eventually producing one coarse particle. (Figure 3.5) Similar to 2D, properties of the coarse particle are determined as follows to satisfy the conservation of mass and momentum.

$$\begin{aligned} x_c &= \frac{1}{8}\sum_{i=1}^{8} x_f^i; \qquad y_c = \frac{1}{8}\sum_{i=1}^{8} y_f^i; \qquad z_c = \frac{1}{8}\sum_{i=1}^{8} z_f^i; \\ \rho_c &= \frac{1}{8}\sum_{i=1}^{8} \rho_f^i; \qquad p_c = \frac{1}{8}\sum_{i=1}^{8} p_f^i; \qquad \boldsymbol{u_c} = \frac{1}{8}\sum_{i=1}^{8} \boldsymbol{u_f^i}; \\ m_c &= 8m_f; \qquad V_c = 8V_f; \qquad h_c = \frac{1}{\gamma} h_f; \end{aligned} \tag{3.5}$$

3.1.3 Kinetic Energy Conservation

As described in the previous section, particle splitting process essentially conserves energy along with mass and momentum. However, in the case of merging, energy is not conserved because the physical quantity of the coarse

particle can be determined only by using the mass and momentum equations. Figure 3.6 shows a simple example in which kinetic energy is not conserved after merging.

In order to consider the conservation of energy, an additional particle must be added. For instance, in 2D, if two particles are generated after merging, total of four velocity variables should be determined (horizontal and vertical velocity of each particle). (Figure 3.7) Since the total mass must be conserved, the mass of the coarse particle is twice that of the fine particle. Momentum and energy equations are written as follows.

$$\begin{aligned} m(v_{1,x} + v_{2,x} + v_{3,x} + v_{4,x}) &= 2m(V_{1,x} + V_{2,x}) \\ m(v_{1,x} + v_{2,x} + v_{3,x} + v_{4,x}) &= 2m(V_{1,x} + V_{2,x}) \\ \frac{1}{2}m(v_1^2 + v_2^2 + v_3^2 + v_4^2) &= \frac{1}{2}(4m)(V_1^2 + V_2^2) \end{aligned} \tag{3.6}$$

where v and V denote the velocity of the fine particle and coarse particle, respectively.

Since one more equation is needed to determine the velocity of the coarse particles, an additional assumption was used that the magnitude of the velocity of the two coarse particles are the same.

$$|\boldsymbol{V_1}| = |\boldsymbol{V_2}| \tag{3.7}$$

Therefore, the coarse particle velocities are derived as follows.

$$
\begin{aligned}
V_{1,x} &= \frac{\sum v_x}{4} - \frac{\sum v_y}{4}\sqrt{\frac{4\sum\left(v_x^2+v_y^2\right)}{(\sum v_x)^2+\left(\sum v_y\right)^2}-1} \\
V_{1,y} &= \frac{\sum v_y}{4} + \frac{\sum v_x}{4}\sqrt{\frac{4\sum\left(v_x^2+v_y^2\right)}{(\sum v_x)^2+\left(\sum v_y\right)^2}-1} \\
V_{2,x} &= \frac{\sum v_x}{4} + \frac{\sum v_y}{4}\sqrt{\frac{4\sum\left(v_x^2+v_y^2\right)}{(\sum v_x)^2+\left(\sum v_y\right)^2}-1} \\
V_{2,y} &= \frac{\sum v_y}{4} - \frac{\sum v_x}{4}\sqrt{\frac{4\sum\left(v_x^2+v_y^2\right)}{(\sum v_x)^2+\left(\sum v_y\right)^2}-1}
\end{aligned}
\tag{3.8}
$$

The newly proposed merging model satisfies the conservation of energy, which can be seen in Figure 3.8.

3.1.4 Error Analysis

In previous studies, error analysis has been performed to find the optimum splitting and merging pattern and related parameters. Liu et al. (2017) have mentioned that γ, which refers to the smoothing length ratio between the coarse and fine particle, and β, the inter-particle distance ratio between the coarse and fine particles are important factors that determines the accuracy of the simulation. (Figure 3.9) In this study, a simple parametric study has been conducted to find the value of β and γ which minimizes the error.

Figure 3.10 shows the computational domain used in the parametric study. Simple square-patterned particle distribution was used, while the center of the square was pre-defined to be split. SPH approximation was performed for 1D and

2D function in the entire domain: (1) $f(x) = e^{-x^2}$ and (2) $f(x,y) = e^{-(x^2+y^2)}$ The results were compared for three different γ values. (Table 3.2) For uniform distribution of particles after splitting, the value of β was fixed to be 0.5. Figure 3.11 and Table 3.3 show the result of the test, and the case in which γ was set to be 1.0 (constant smoothing length for entire domain) showed the minimum error. However, it is preferable to use γ as 0.5 in terms of computational efficiency because the fine particles interact with a small number of particles appropriate for their size only when the smoothing length is reduced. The error shows a maximum value near the refinement interface, where the particle size changes abruptly. In this study, both values were used considering both computational efficiency and accuracy. Details are described in the following chapter.

For merging, the same domain was used to find the optimal merging pattern, but the particles were merged in the center region of the domain. Three patterns were tested, and it is described in Figure 3.12. Pattern A is the pattern that merging four particles into one coarse particle, which can't satisfy the conservation of energy. Coarse particle is placed at the center of mass of the fine particles. Pattern B is the newly presented merging pattern in this study, and two coarse particles are placed at a certain distance from the center of mass in a diagonal direction. Pattern C also produces two coarse particles, but the positions of particles are defined at the center of mass of two fine particle groups respectively. From Figure 3.13 and Table 3.5, it can be seen that pattern B showed the best results, which is the pattern used in this study.

3.1.5 Variable smoothing length

The smoothing ratio is an essential parameter that greatly affects the instability at the interface where the resolution is changed. In Figure 3.14, the left side of the figure, which is a result of using $\gamma = 0.5$, shows the large error of velocity profile near the resolution interface. On the other hand, the simulation results using constant smoothing length show a stable velocity profile, which is shown on the right side of the figure. In terms of accuracy only, it is ideal to use the same smoothing length for the entire computational domain, regardless of particle size. However, in terms of computational efficiency, if the smoothing length is maintained even the particle size is reduced, the number of particles to be included in the support domain increases, which leads to an increase in computational cost. This is maximized when refinement is applied into multiple stages. When a particle is split twice with constant smoothing length, the number of particles included in the support domain of the fine particle is 16 times the original.

Therefore, to maintain both computational accuracy and efficiency at a reasonable level, propose a continuous smoothing length model is proposed in this study. Using the model, smoothing length smoothly changes at the resolution interface, but eventually be adjusted according to the particle size. (Figure 3.15) The smoothing length of each particle can be estimated as below.

$$\widetilde{h_i} = \sum_j h_j W_{ij} \frac{m_j}{\rho_j} \tag{3.7}$$

where $\widetilde{h_i}$ denotes the averaged smoothing length of the center particle. Also, different smoothing lengths between the particles can lead to a violation of the action-reaction law. (Figure 3.16) Therefore, in the calculation between particles

with different smoothing lengths, it is necessary to include large particles should in the neighbor search of small particles.

3.2 GPU-Parallelization

SPH is a Lagrangian-based method and represents the entire fluid system with a finite number of particles. Since several thousand to billions of particles can be used in the simulation, large-scale calculations inevitably result in a massive computational cost. Therefore, to perform simulation within a reasonable time using SPH, an appropriate acceleration technique is required. The SPH code used in this study, SOPHIA, is an in-house code developed within the research group and is parallelized using Graphics Processing Unit (GPU) through previous studies.

3.2.1 GPU-based SPH Algorithm

Figure 3.17 shows GPU-parallelized SPH algorithm. All processes except input reading and initialization are performed in parallel by assigning one thread allocated to each particle in the computational domain. APR is performed after the time integration.

3.2.2 APR data management

The use of APR changes the total number of particles in the computational domain by creating or erasing particles. Thus, the particle array, which stores the

particle information, requires a buffer memory for newly generated particles. (Figure 3.18) The buffer memory is at the end of the particle array.

In splitting process, one coarse particle splits into four fine particles. Fine particles are produced at the front of the buffer memory space, and the coarse particle is treated as a dummy and excluded from the calculation. The particle array, in which the order is twisted by creating and deleting particles, is reordered through the GPU sorting process in the next time step. (Figure 3.19)

Similarly, produced coarse particles in merging process are stored at buffer memory, while merged fine particles are treated as dummy particles. However, to prevent overlapping with particles generated through splitting at the same time step, coarse particles are stored in the rearmost part of the buffer memory. (Figure 3.20)

3.2.3 Race Condition and Atomic

During the parallel computation of the SOPHIA code, each particle is assigned to one CUDA core thread. Each thread calculates the density and force of the particle assigned to it, which works fine in most cases. However, problems may occur in a situation when multiple threads access to same memory address simultaneously, which is called a race condition. A simple example is shown in figure 3.21. Local variable x is initialized to 7. Assuming that the operation of adding 1 to x is performed twice, the operation using the CPU does not cause any problems. However, if the calculation is performed in parallel using two threads A and B, the calculation can be started in the remaining thread before it is finished in the first one, which obviously gives the wrong answer.

Using APR can also cause the race condition, when splitting or merging occurs several times in a single time step. (Figure 3.22) For two particles to be split at the same time step, different memory addresses must be allocated to each particle, which is not possible using parallel computing. (Figure 3.23) In merging on the other hand, when the center particle searches for its neighbor particles to be merged, target particles can be simultaneously searched for different center particles. (Figure 3.24) In race condition, particles are not generated properly or more particles are generated.

To get rid of the race condition, it is necessary to release the parallelization in certain sections where the condition occurs so that only one thread can access the memory at a time. CUDA C provides atomic operations, which prevent other threads from accessing the memory during simple operations are being performed. When the atomic operation is used, if there is a thread that is performing the calculation, the remaining threads are forcibly delayed. (Figure 3.25) By utilizing atomics, it is possible to implement a mutex lock algorithm that can serialize only a specific part of a parallelized code. (Figure 3.26) The process of the mutex lock algorithm is as follows.

1) The first thread enters the lock function, which includes the mutex variable. The mutex is initialized as 0. Since the lock function is an atomic operation, access from other threads is not allowed.

2) The first thread changes the mutex value to 1 as it exits the lock function, and moves on to the next step. Since the operation at the lock function of the first thread is finished, the second thread can access the function.

However, since the lock is a function that can only be exited when the mutex value is 0, the second thread repeats the while loop in the function.

3) After all calculations of the first thread are finished, the mutex value is returned as 0 through the unlock function. The lock function that obtains the mutex value releases the second thread out of the function, and the third thread is locked in the function again.

The above process is repeated until all threads have completed the calculation, and the race condition can be resolved by isolating the calculation of each thread independently.

3.2.4 GPU-based APR Algorithm

The GPU-parallelized APR algorithm in which race condition is resolved is performed in the following order. First, in the case of splitting,

1) In the first step, particles satisfying the splitting condition are searched, and the number of particles to be split is counted. The reason for counting is to check whether the atomic operation is necessary for the subsequent process. If it is determined that atomic operation is not necessary, the following processes are omitted.

2) Each newly produced particle must receive a different memory address in which it will be stored. Each coarse particle is assigned a different number.

If there is more than one particle to be split, this process is done by using atomic operation.

3) New fine particles are formed and stored in the buffer memory based on the number assigned earlier. If the assigned number of the coarse particle is 1, the first four memory spaces are used to produce the fine particles.

The sequence of the processes are shown in Figure 3.27. The merging process is performed in the following order. (Figure 3.28)

1) Similar to splitting, particles satisfying the merging condition are searched. Particles must have other particles to merge within their search range.

2) The particles to be merged form a group. Each group is given a different number as in the splitting process, using atomic. One particle in the group is selected as a representative particle so that only the representative particle can participate in the merging function of the next step.

3) Each representative particle determines the memory address to be stored based on the group number, and new coarse particles are produced. If the assigned group number is 1, the last two memory spaces are used to produce the coarse particles.

As a result, the produced particles are stored in the buffer memory in the form shown in Figure 3.29.

Table 3.1 APR patterns and criteria of previous studies

Author (year)	APR criterion	APR pattern	Split/Merge
R.Vacondio (2013)	Position	Hexagon (2D)	Split + Merge
Barcarolo (2014)	Position	Square (2D)	Split + Merge
R.Vacondio (2016)	Position	Various patterns (3D)	Split
W.T.Liu (2017)	Position	Square (2D)	Split
P.N.Sun (2019)	Volume	Square (2D)	Split + Merge
L.Hu (2020) [17]	Position	Square (2D)	Split

Table 3.2 Conditions for particle splitting error analysis

Input function	Case	β	γ
$f(x) = e^{-x^2}$	Case 1)	0.5	0.5
	Case 2)	0.5	0.7
	Case 3)	0.5	1.0
$f(x) = e^{-(x^2+y^2)}$	Case 4)	0.5	0.5
	Case 5)	0.5	0.7
	Case 6)	0.5	1.0

Table 3.3 Error analysis result of particle splitting

Case	e_{total} [%]	e_{split} [%]
Case 1)	0.0525	0.1148
Case 2)	0.0522	0.1165
Case 3)	0.0502	0.1146
Case 4)	0.0503	0.0764
Case 5)	0.0533	0.0812
Case 6)	0.0481	0.0744

Table 3.4 Conditions for particle merging error analysis

Input function	Case	Positioning	γ
$f(x) = e^{-x^2}$	Case 1)	A	0.5
	Case 2)	B	0.5
	Case 3)	C	0.5
$f(x) = e^{-(x^2+y^2)}$	Case 4)	A	0.5
	Case 5)	B	0.5
	Case 6)	C	0.5

Table 3.5 Error analysis result of particle merging

Case	e_{total} [%]	e_{merge} [%]
Case 1)	0.0869	0.3879
Case 2)	0.0540	0.1538
Case 3)	0.0567	0.1579
Case 4)	0.1739	0.2254
Case 5)	0.1726	0.2079
Case 6)	0.1723	0.1911

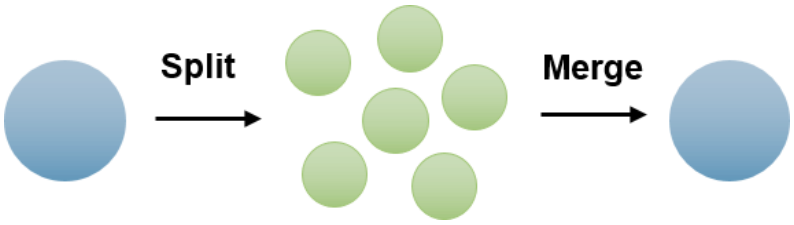


Figure 3.1 Schematic of the Adaptive Particle Refinement (APR)

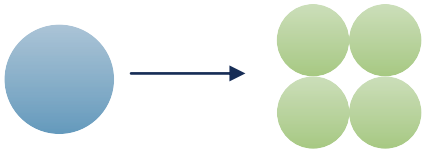


Figure 3.2 Particle splitting pattern in 2D

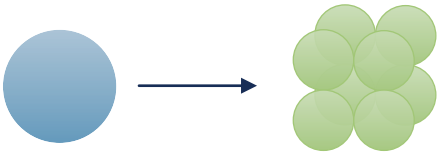


Figure 3.3 Particle splitting pattern in 3D

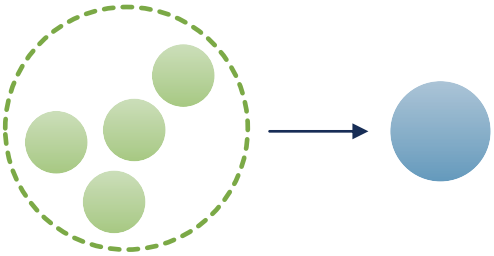


Figure 3.4 Particle merging pattern in 2D

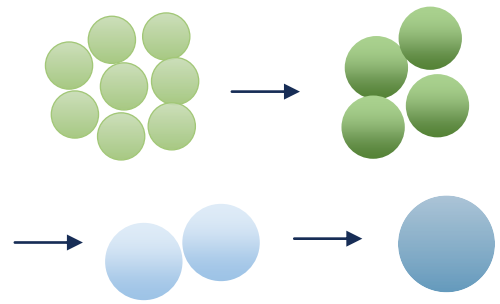
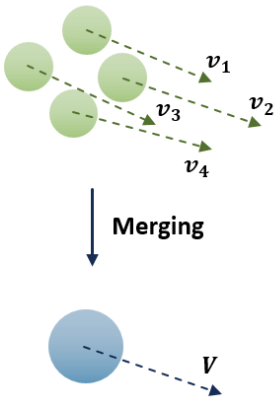


Figure 3.5 Particle merging pattern in 3D



$m = 1$	v_x	v_y	p_x	p_y	E_{kin}
Particle #1	1.0	-0.45	1.0	-0.45	0.6013
Particle #2	0.95	-0.52	0.95	-0.52	0.5865
Particle #3	1.04	-0.51	1.04	-0.51	0.6709
Particle #4	0.90	-0.5	0.90	-0.5	0.5300
Sum			**3.89**	**-1.98**	**2.3886**

$m = 4$	V_x	V_y	p_x	p_y	E_{kin}
Sum	0.9725	-0.4950	**3.89**	**-1.98**	**2.3816**

Figure 3.6 Merging case in which kinetic energy is not conserved

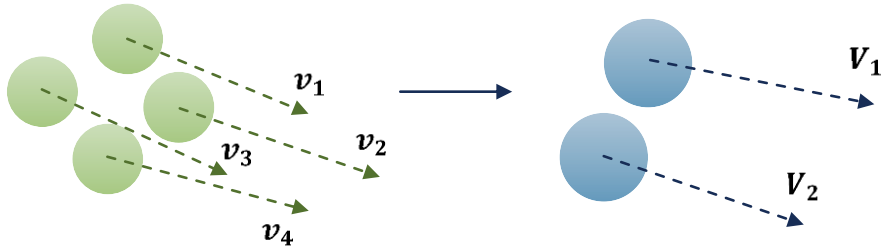
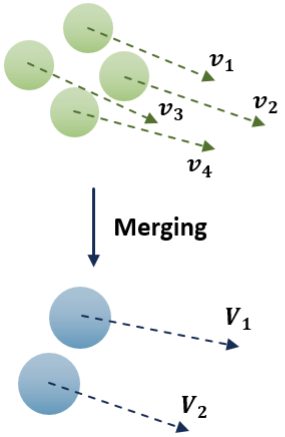


Figure 3.7 New concept of particle merging



$m = 1$	v_x	v_y	p_x	p_y	E_{kin}
Particle #1	1.0	-0.45	1.0	-0.45	0.6013
Particle #2	0.95	-0.52	0.95	-0.52	0.5865
Particle #3	1.04	-0.51	1.04	-0.51	0.6709
Particle #4	0.90	-0.5	0.90	-0.5	0.5300
Sum			**3.89**	**-1.98**	**2.3886**

$m = 2$	V_x	V_y	p_x	p_y	E_{kin}
Particle #1	0.9993	-0.4423	1.9986	-0.8846	1.1942
Particle #2	0.9457	-0.5477	1.8914	-1.0954	1.1944
Sum			**3.89**	**-1.98**	**2.3886**

Figure 3.8 New merging concept in which conserves kinetic energy

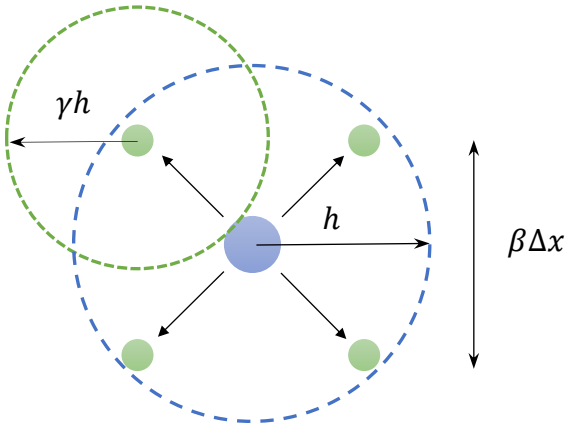


Figure 3.9 Smoothing length ratio γ and spacing ratio β

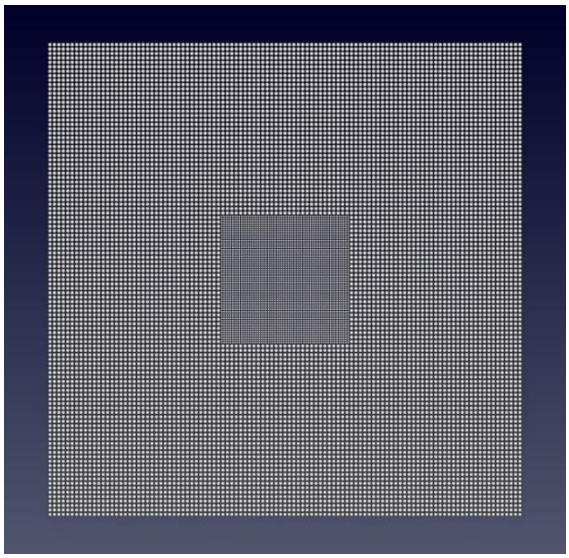


Figure 3.10 Computational domain of the Error analysis

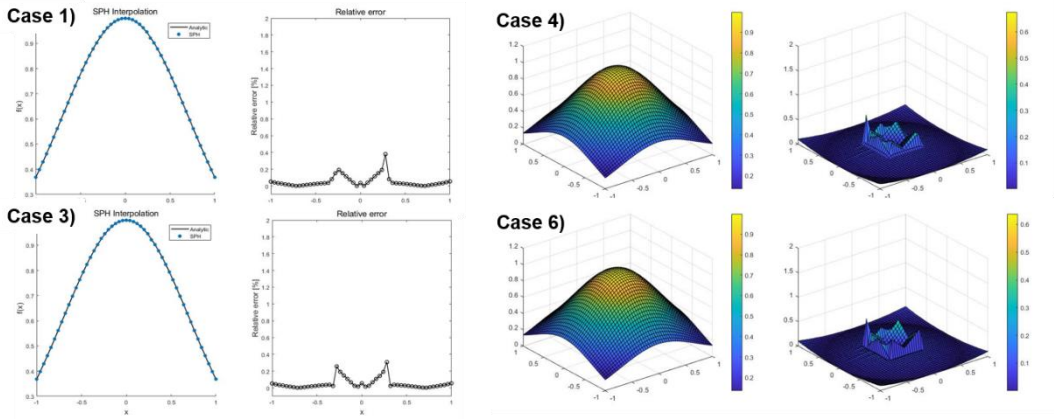


Figure 3.11 Error analysis results of particle splitting cases
(Case 1,3 : 1D function Case 4,6 : 2D function)

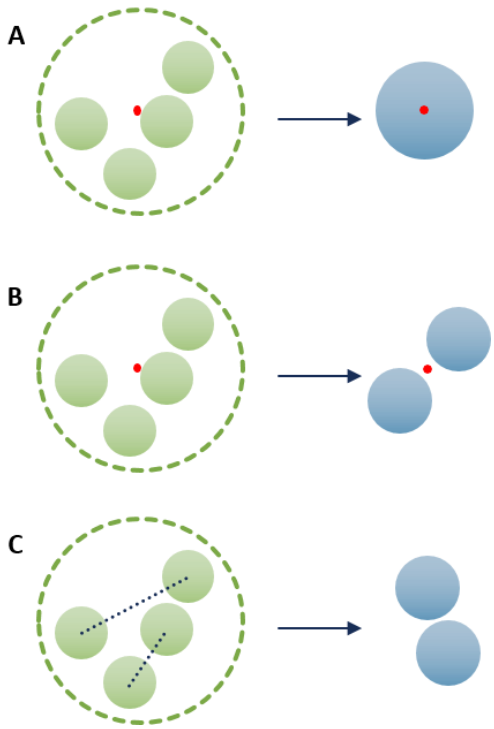


Figure 3.12 Positioning cases of the particle merging

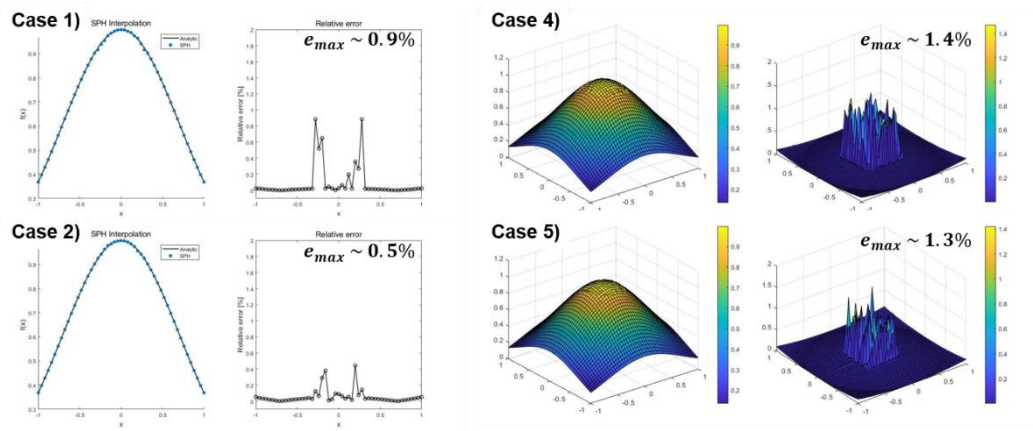


Figure 3.13 Error analysis results of particle merging cases
(Case 1,2 : 1D function Case 4,5 : 2D function)

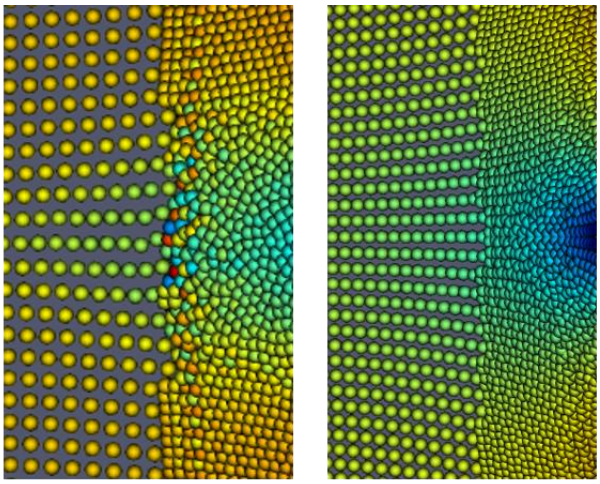


Figure 3.14 Interface error with different smoothing length ratio
(Left : $\gamma = 0.5$, Right : $\gamma = 1.0$)

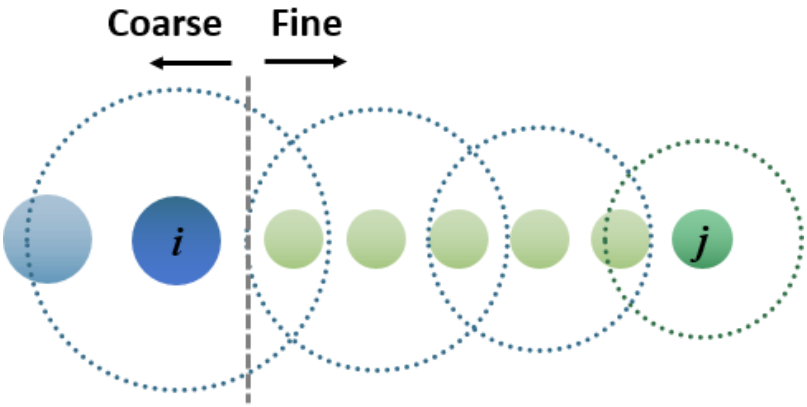


Figure 3.15 Continuous smoothing length model

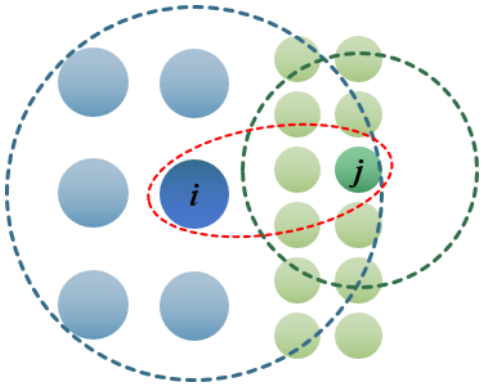


Figure 3.16 Particle distribution at resolution interface

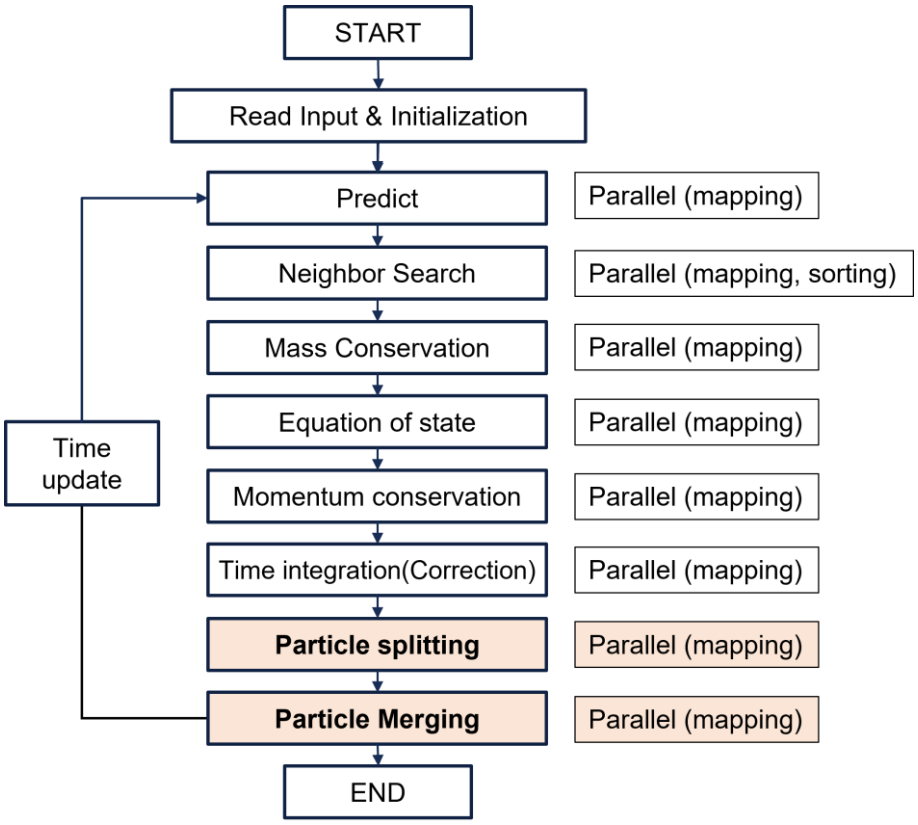


Figure 3.17 SOPHIA code algorithm (GPU-parallelized)

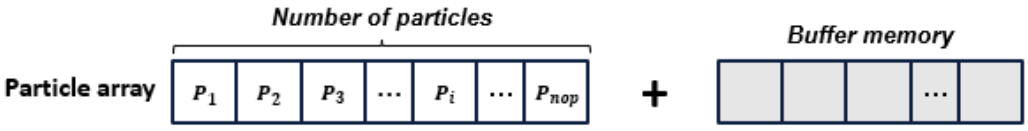


Figure 3.18 SOPHIA particle array and buffer memory

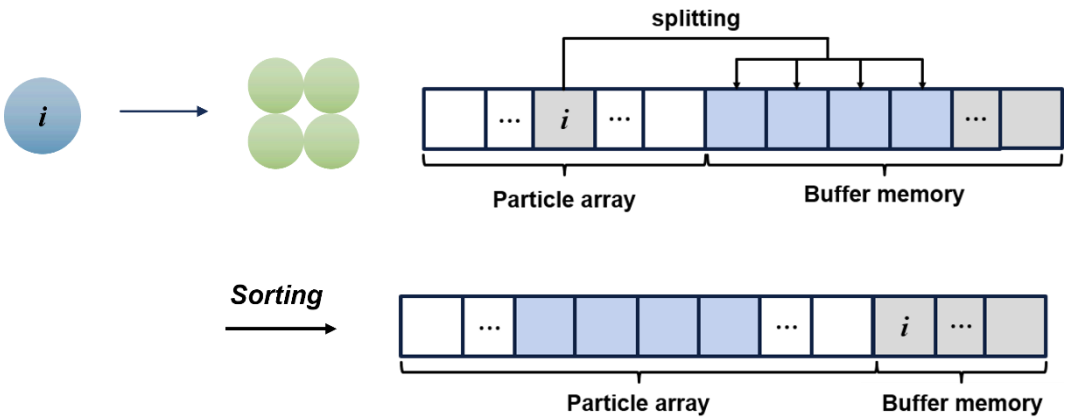


Figure 3.19 Data management in particle splitting

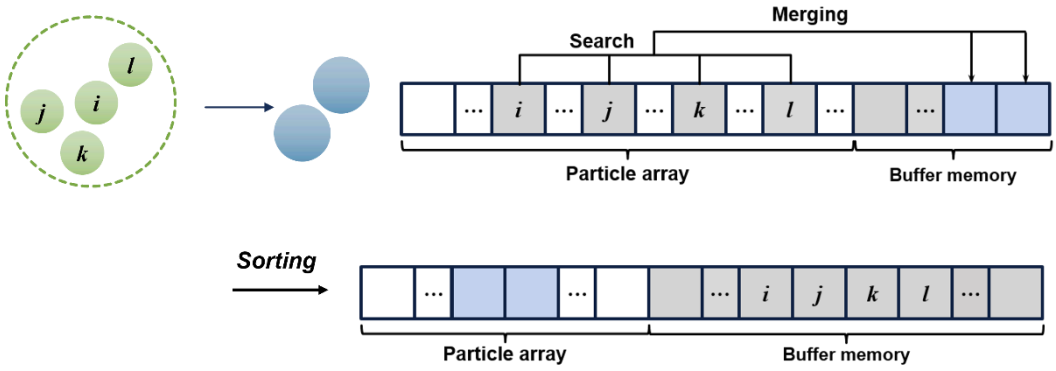


Figure 3.20 Data management in particle merging

STEP	Example
Thread A reads the value in x.	A reads 7 from x.
Thread B reads the value in x.	B reads 7 from x.
Thread A adds 1 to the value it read.	A computes 8.
Thread B adds 1 to the value it read.	B computes 8.
Thread A writes the result back to x.	**x ← 8.**
Thread B writes the result back to x.	**x ← 8.**

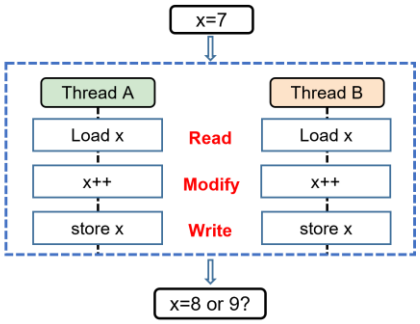


Figure 3.21 Example of the race condition in parallel computing process

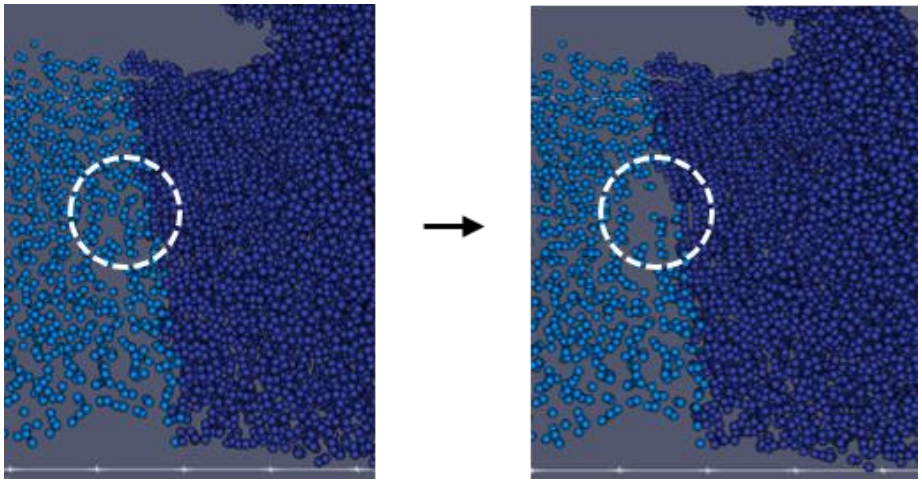


Figure 3.22 Race condition occurred while using APR

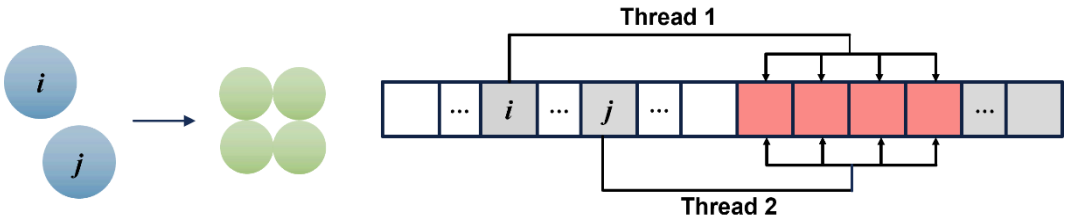


Figure 3.23 Race condition in particle splitting

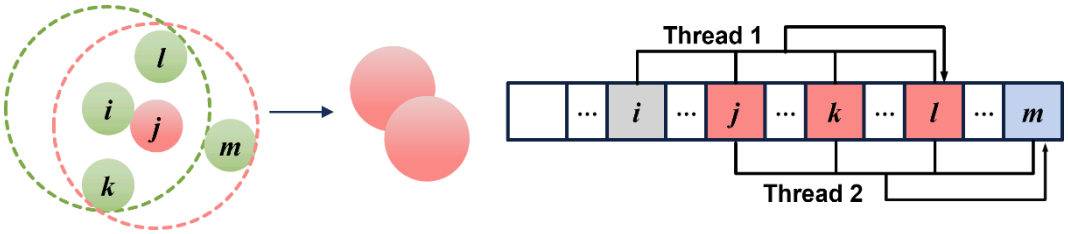


Figure 3.24 Race condition in particle merging

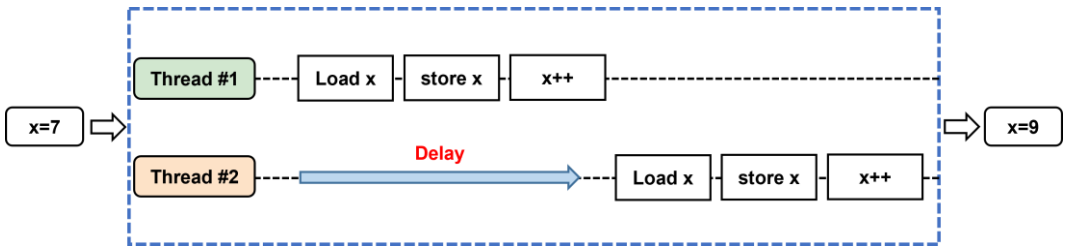


Figure 3.25 Parallel computing using atomic operation

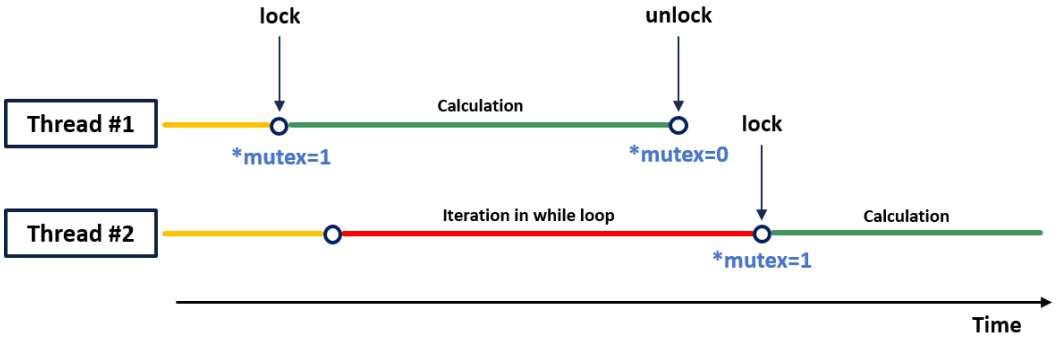


Figure 3.26 GPU mutex lock algorithm using atomic operation

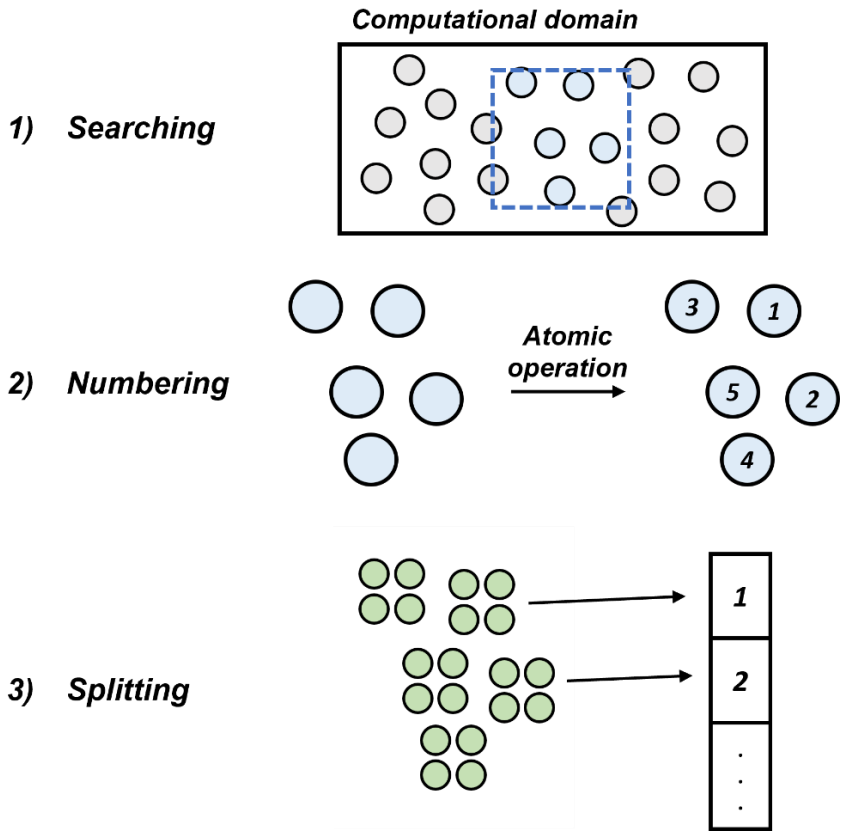


Figure 3.27 Particle splitting sequence using atomic

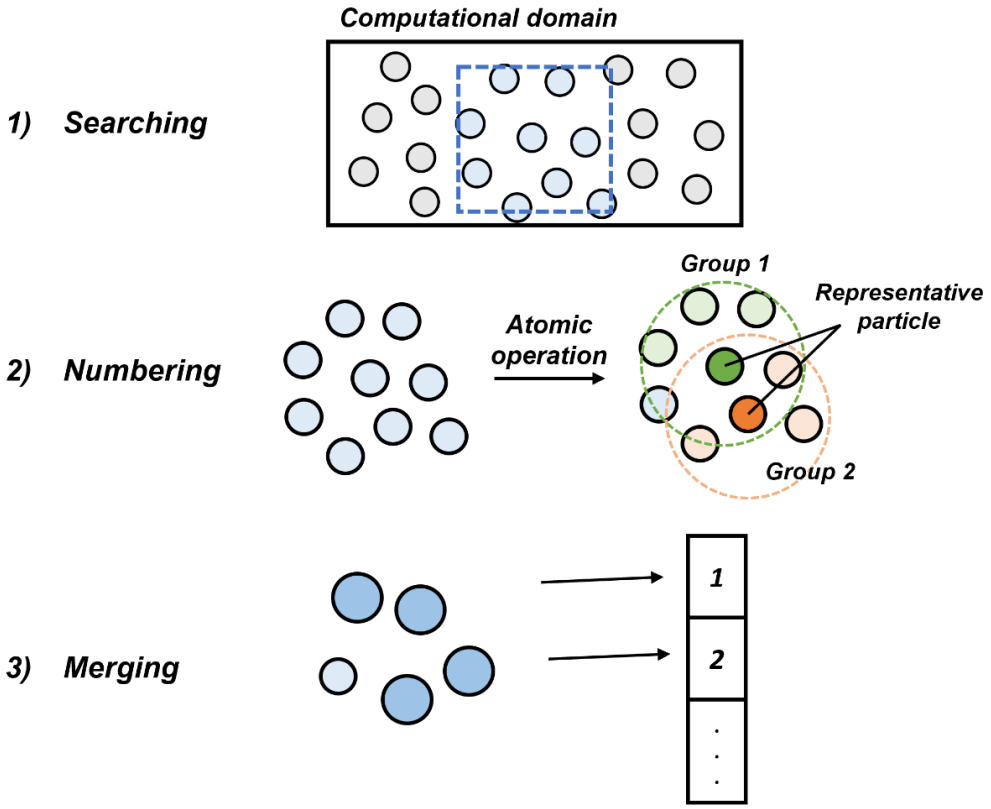


Figure 3.28 Particle merging sequence using atomic

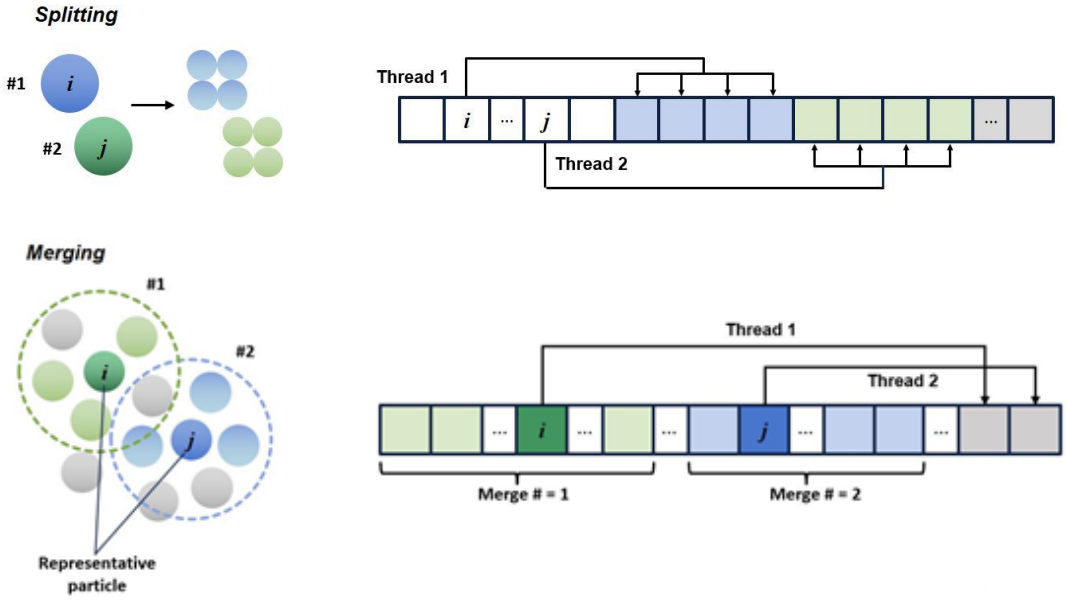


Figure 3.29 GPU-based APR particle production and date management

Chapter 4

Results & Discussions

4.1 Benchmark Simulation

For verification and validation of the developed APR model, various benchmark simulations were performed. Since the APR model is basically a method of increasing the local resolution, the results of high-resolution analysis and the results of applying APR to a lower resolution were compared in terms of accuracy and efficiency.

4.1.1 Hydrostatic Pressure

A simple hydrostatic test case was performed to test the ability of the APR model in maintaining the hydrostatic pressure. The initial condition of the hydrostatic test is shown in figure 4.1. The depth of the tank filled with water is 0.9 m and the width is 2 m. In order to confirm that the model works stably, refine zones were set in different heights of the tank in three test cases. Reference density was set to be the density of pure water, i.e. $\rho = 1000\ \mathrm{kg/m^3}$ and the

gravitational constant is $\mathrm{g} = 9.8\ \mathrm{m}/s^2$. Initial particle distance in the coarse zone is $\Delta x_c = 0.005\ \mathrm{m}$, which means that initial particle distance in the refine zone is $\Delta x_f = 0.0025\ \mathrm{m}$.

Pressure distribution is shown in Figure 4.2. The color bar means the pressure from 0 to 9000 pa, and it can be seen that the pressure field is well-formed along with the height. In Figure 4.3, the numerically calculated pressures for three cases are compared with the analytic solution defined as $p = \rho g h$. All three cases showed good results within the margin of error of 0.4%.

One more test was performed to apply multi-stage refinement. As shown in Figure 4.4, the computation domain was divided into three zones; Coarse zone at the top of the tank, first refine zone at the middle, and second refine zone at the bottom, which splits the particles twice so that one coarse particle is finally split into 16 fine particles. The wall boundary particles were split twice to match the size of the smallest particles. The result also confirms that the static pressure is well-formed, and it can be seen that the second stage refinement is well applied through the enlarged part on the right side of Figure 4.5. The maximum relative error in the entire domain was about 0.3%. (Figure 4.6)

4.1.2 Pipe Flow

The second validation case is a fully developed laminar flow between two infinite parallel plates. The initial setup of the simulation is shown in Figure 4.7 and Table 4.1. The first case assumes a constant velocity of the fluid at the inlet of the flow path. Assuming Newtonian fluid, the flow develops over time and shows a parabolic velocity distribution with the highest flow velocity at the center

of the channel. Second case assumes a constant velocity at the lower solid boundary, forming a linear distribution of velocity in the y direction. No-slip boundary condition was applied in both case, and a region near the y-direction centerline was set as a refine zone, where particle resolution was doubled. Inlet and wall velocity are 0.1 m/s, and Reynolds number was set to be 10 in both cases.

Figure 4.8 shows the velocity profile of the fully developed flow. In the left part of the figure, it can be seen that only the region near the center line has been simulated with higher resolution. From the velocity profile on the right side, it can be seen that the results of the entire domain including the interface of the refine zone agree well with the analytic solution, showing an insignificant error of up to 0.2%.

4.1.3 Dam Break

Dam break is a classic benchmark simulation that is widely adopted in SPH field, including previous researches on APR. Dam break simulation was performed by applying both particle splitting and merging. The initial simulation setup is shown in Figure 4.9. The region near the right wall was designated as the refine zone, and the line 0.5m away from the refine zone was set to be the merging line. As simulation progress, the water column at the left side collapse and propagates toward the right wall. If particles reach the refine zone, each particle splits into 4 fine particles. When the wave hits the wall and returns through the merging line, the fine particles are merged into coarse particles again. The simulation conditions are shown in Table 4.x, and simulation was performed for three different resolutions: 1) Low-resolution, 2) High-resolution, and 3) Multi-

resolution using APR

As a result, it was confirmed that the pre-defined refine and merging regions work well, controlling the particle resolution for each area. In Figure 4.10, it can be seen that the simulation results using APR are qualitatively and quantitatively in good agreement with the high-resolution results. Especially, it was confirmed that the computing time of the case applying APR was reduced by about 58% compared to the high-resolution case, showing very high computational efficiency. (Table 4.2) The number of particles used during the simulation is as shown in the graph of Figure 4.11, showing that using APR can dynamically control the number of particles within the simulation time.

4.1.4 Karman Vortex

In this section, the results of the Karman vortex simulation are described. As shown in Figure 4.12, viscous flow around a circular cylinder is tested using the SPH model with APR technique applied. The Reynolds number, defined as $Re = \rho UD/\mu$ was set as 100. U and D denote stream velocity and cylinder diameter, and μ is the viscosity of the fluid. The initial conditions of the simulation are listed in Table 4.3. Refine zone was arranged in a square shape with a width of $3D$ in the adjacent area including the cylinder. Distance between inlet boundary and outlet boundary is $24D$, and two free-slip boundaries were arranged at $y = \pm 6D$. The particles entering the refine zone were split into 4 fine particles, and particle merging was not used in this simulation. The particle shifting technique has been applied for the simulation, to prevent the formation of cavities in the area behind the cylinder. (Lind, 2012)

Figure 4.13 shows the simulation results over time. (a) is set of snaps colored by velocity magnitude, while the color of (b) refers to the vorticity. As shown in Figure 4.14, fine particles move along the vortex structure in the wake with a shape similar to the von Karman vortex street. By using the constant smoothing length model, no instabilities in velocity or vorticity are observed at the interface.

The drag and lift force coefficients were measured during the simulation and were compared with the reference simulations from the previous studies. (Figure 4.14, Table 4.4) The results show some small discrepancies, but this could be due to the different mesh/particle resolutions adopted in each simulation.

4.2 Application

As the final stage of the study, the developed APR model was applied to various hydrodynamic problems. In this section, the following two application cases are described: (1) The Jet break-up phenomenon at the pre-mixing stage of Fuel Coolant Interaction; (2) Single air bubble rising. Simulations were performed using various particle resolutions to show the performance of the applied model, and the results were compared qualitatively and quantitatively with the reference results.

4.2.1 Jet Break-up

Fuel Coolant Interaction (FCI) at light water reactor is an important phenomenon in nuclear safety, and it includes a variety of complex physical phenomena. At the premixing stage of FCI, the ejected molten corium jet first

encounters coolant in the lower cavity and breaks into small droplets, which is called jet breakup. Understanding the physics of the jet breakup phenomenon is important for more precise FCI analysis.

In this study, jet breakup phenomenon was simulated by SOPHIA code. For a preliminary test, the simulation was performed in 2D, and boiling of the coolant was not considered. The experiment performed by Manickam et al. (2017) was selected as the reference. Figure 4.x shows the schematic of the MISTEE-Jet facility, which was built to perform the small-scale jet breakup experiment. (Figure 4.16) At the upper part of the facility, the melt is prepared in a crucible heated by induction furnace. After the melt temperature reaches the experimental condition, the plug at the bottom side of the crucible is pulled out, and the melt is ejected into a rectangular tank filled with water. The experiment case adopted for simulation is one that using wood metal for the melt, and since wood metal has a melting point of about 90℃, boiling is not included in the experiment.

The simulation input created based on the experiment is shown in Figure 4.17. The wood metal jet is injected into the water pool, while the water tank has a long rectangular shape in the vertical direction. Since the region of interest is the area near the leading edge of the melt jet, the computational load may become excessively high if the entire domain including the lower part of the tank is calculated at high resolution. Using APR, by tracking the leading edge of the melt jet and only splitting particles around the melt jet, the total computing time can be reduced while performing high-resolution analysis only on the region of interest. For comparison in terms of accuracy and efficiency, three cases were simulated using three different resolutions: (1) Low resolution ($\Delta x = 0.4mm$); (2) High resolution ($\Delta x = 0.2mm$); (3) Multi-resolution with APR ($\Delta x =$

$0.2mm \sim 0.4mm$) Initial conditions of the melt jet are also described in Table 4.5.

The simulation results over time of the APR case are shown in Figure 4.18. As shown in Figure 4.19, the refine zone gradually expands downward as the melt jet penetrates the water pool. Since the melt jet breaks and only penetrates to the midpoint of the water tank until the total simulation time of 0.2 seconds, the lower area of the tank is not split, increasing overall computational efficiency. Comparing with other cases, results of the APR case show good agreement with the results of the high-resolution case, precisely simulating the fluid behavior near the leading edge of the melt jet. (Figure 4.20) Jet penetration depth was also compared, and the results using the APR model also matched well with the high-resolution analysis, showing a maximum error of 0.3%. (Figure 4.21) Computing time for each case was measured, and as shown in Table 4.6, the calculation was performed 39% faster than the high-resolution analysis when using APR.

4.2.2 Single Bubble Rising

The simulation of a single air bubble rising was performed. The simulation was selected to evaluate whether the APR model can be applied stably to two-phase flow problems. The reference simulation adopted in this study is the level set simulation of Sussman et al. (1995) Simulation setup is shown in Figure 4.22. The radius of the air bubble is 2.5cm and the square shape refine zone of 3R width was set around an air bubble. As the bubble rises, the refine zone moves along with the bubble and maintains the high-resolution area around the bubble. The particles excluded from the refine zone are merged to lower the spatial resolution. No-slip boundary conditions were used for the domain boundary. The detailed

initial conditions for the simulation are listed in Table 4.7. The Bond number ($Bo = 4g\rho_l R^2/\sigma$) and Reynolds number ($Re^* = (2R)^{3/2}\sqrt{g}\rho_l/\mu_l$) were set as same as the reference simulation. To evaluate the performance of the APR model, the simulation was done using three different resolutions: (1) Low resolution ($\Delta x = 1.0mm$); (2) High resolution ($\Delta x = 0.5mm$); (3) Multi-resolution with APR ($\Delta x = 0.5mm \sim 1.0mm$)

Figure 4.23 and Figure 4.24 show the snapshots of the simulation over time in two different Reynolds number cases. Each row contains low-resolution, high-resolution, and multi-resolution results using APR, respectively. Unlike the high-resolution result, the low-resolution result shows that the shape of the air bubble is distorted, whereas the multi-resolution result simulates the droplet shape stably. Figure 4.25 and Figure 4.26 compare the surface motion of the rising bubble with different Reynolds number. The black line represents the level set result from Sussman(1995), while the blue colored area shows the SPH simulation result using the APR model. As shown in the figures, SPH results agree well with the level set result, precisely expressing the shape of the bubble including the pinch-off of the bubble. The axial position of the bubble was tracked and compared with the level set result, and it was confirmed that SPH result followed the trend, both in high-resolution and multi-resolution cases.(Figure 4.27, Figure 4.28) Total computation time was also reduced by 46%, as can be seen in Table 4.8. Through the bubble rising simulation, it was confirmed that the APR model performed stably in two-phase conditions.

Table 4.1 Simulation conditions of pipe flows

Parameter	Case 1	Case 2
$\boldsymbol{Re}$	**10**	**10**
d	0.01	0.01
ρ_0	1,000	1,000
μ	0.1	0.1
$\boldsymbol{u_0}$	**0.0**	**0.1**
$\boldsymbol{u_w}$	**0.1**	**0.0**
Δx	$2.0e-4$ $\sim 1.0e-4$	$2.0e-4$ $\sim 1.0e-4$
N_p	6,436	6,436

Table 4.2 Total computing time of the Dam break simulation

Dam break	$\Delta \boldsymbol{x}$ $[\boldsymbol{m}]$	$\boldsymbol{N_p}$	Computing time [sec]
Low resolution	0.025	18,745	1,678.4
High resolution	0.0125	74,931	6,095.9
APR	**0.025 ~ 0.0125**	**18,745 ~ 35,521**	**2,608.5**

Table 4.3 Simulation conditions of Karman vortex

Parameter	Case
$\boldsymbol{Re_D}$	1**00**
ρ	1,000
d	0.1
u_0	0.1
μ	0.1
Δx	0.005 ~0.0025
$N_{p,i}$	78,727

Table 4.4 Drag and lift coefficient

Reference	$\boldsymbol{C_D}$	$\boldsymbol{C_L}$
SOPHIA	1.41 $\pm$ 0.034	$\pm$ 0.34
Tafuni et al. (2018)	1.474	$\pm$ 0.322
Marrone et al. (2013)	1.36 $\pm$ 0.01	$\pm$ 0.24
Braza et al. (1986)	1.364 $\pm$ 0.015	$\pm$ 0.25

Table 4.5 Simulation conditions of Jet break up

Parameter	**Case 1**	**Case 2**	**Case 3**
APR	***OFF***	***OFF***	***ON***
ρ_l	1,000	1,000	1,000
ρ_m	9,870	9,870	9,870
u_0	1.7	1.7	1.7
c_0	10	10	10
Δx	0.004	0.002	0.004
dt	$1.0e-6$	$1.0e-6$	$1.0e-6$
N_p	245,059	963,017	670,987

Table 4.6 Total computing time of the Jet break up simulation

Resolution	Δx $[m]$	N_p	T_{com} [sec]
Low	4.0e-4	241,422	5375.4
High	2.0e-4	963,017	20948.3
APR	**4.0e-4 ~ 2.0e-4**	**327,796 ~ 602,818**	**12790.8**

Table.4.7 Simulation conditions of single bubble rising

Parameter	Case 1	Case 2
$\boldsymbol{Bo}$	**200**	**200**
$\boldsymbol{Re^*}$	**100**	**1,000**
R	0.025	0.025
ρ_l/ρ_g	1,000	1,000
μ_l/μ_g	100	100
σ	0.1225	0.1225
g	9.8	9.8
$\Delta x\ [m]$	0.001	0.001
$dt\ [sec]$	$1.0e-7$	$1.0e-7$
N_p	51,840	51,840

Table 4.8 Total computing time of the bubble rising simulation

Resolution	$\Delta x\ [m]$	N_p	T_{com}
Low	1.0e-3	39,744	24,408
High	5.0e-4	154,444	86,857
APR	**1.0e-3 ~5.0e-4**	**51,840 ~ 66,501**	**47,480**

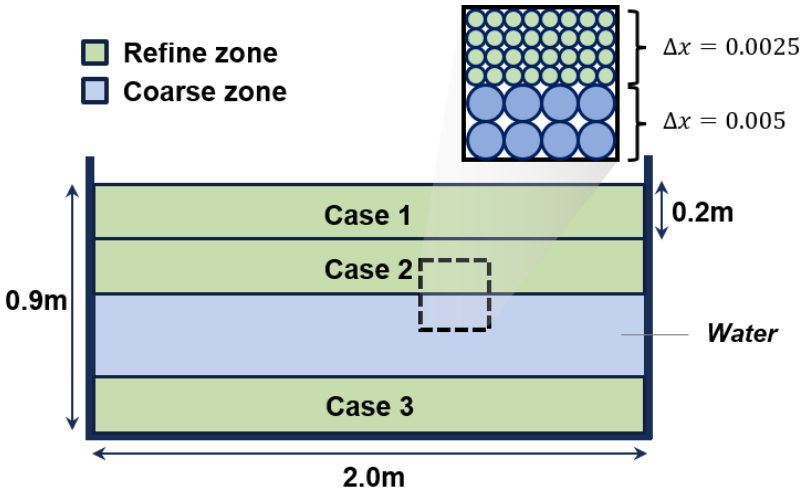


Figure 4.1 Initial setup of the hydrostatic pressure test

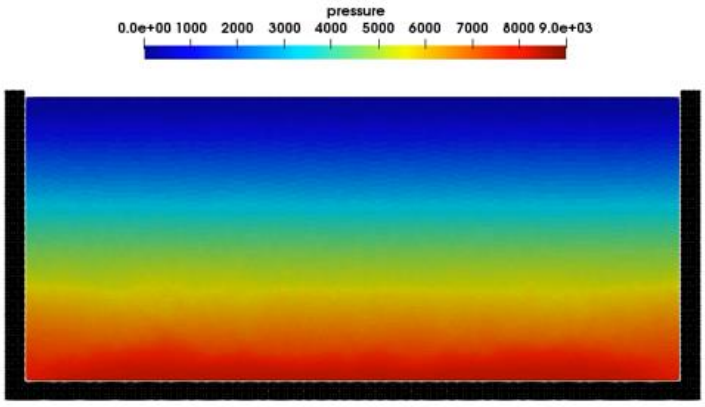


Figure 4.2 Pressure distribution of test case

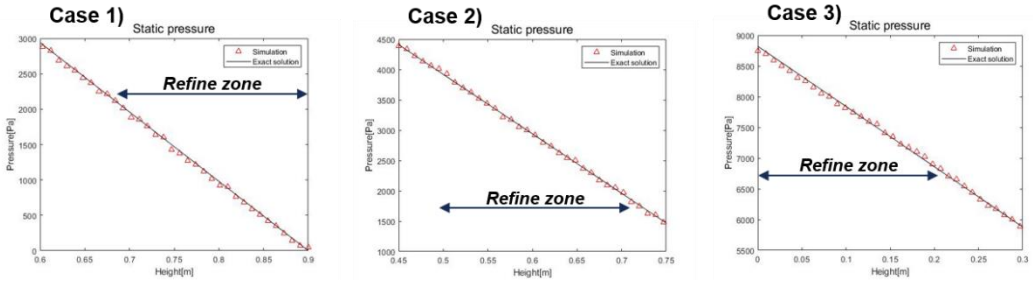


Figure 4.3 Pressure results of three test cases

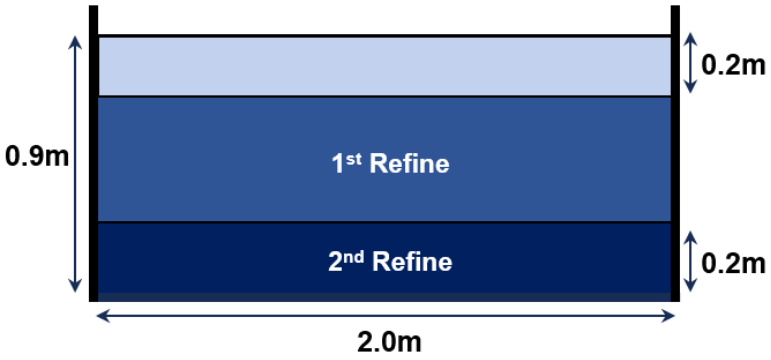


Figure 4.4 Initial setup for multi-stage refinement test

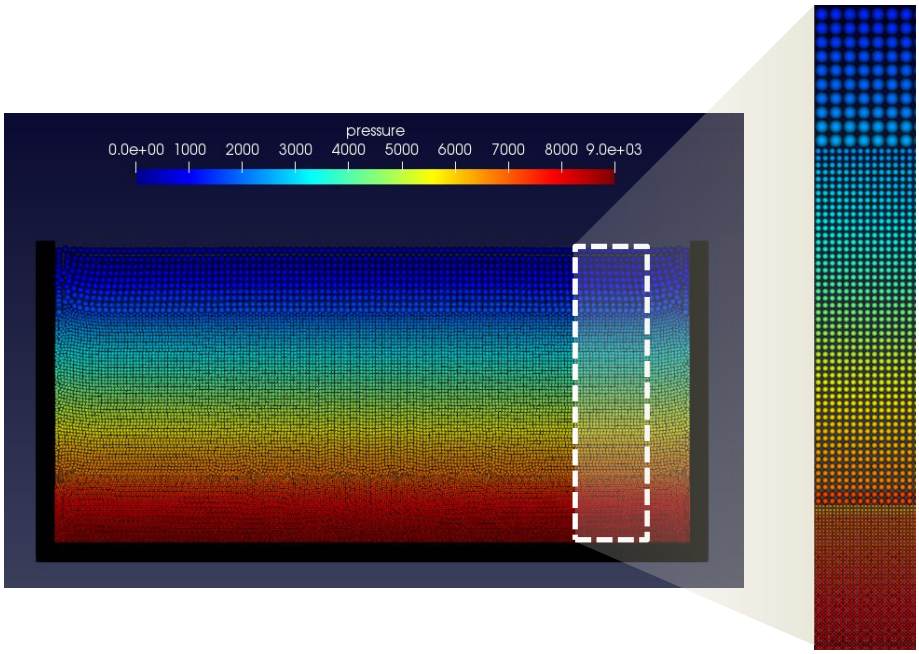


Figure 4.5 Pressure distribution of multi-stage refinement test

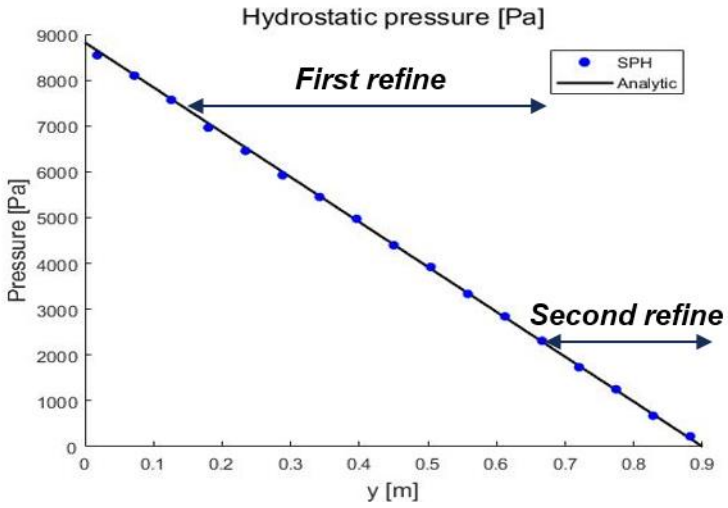


Figure 4.6 Pressure result of multi-stage refinement case

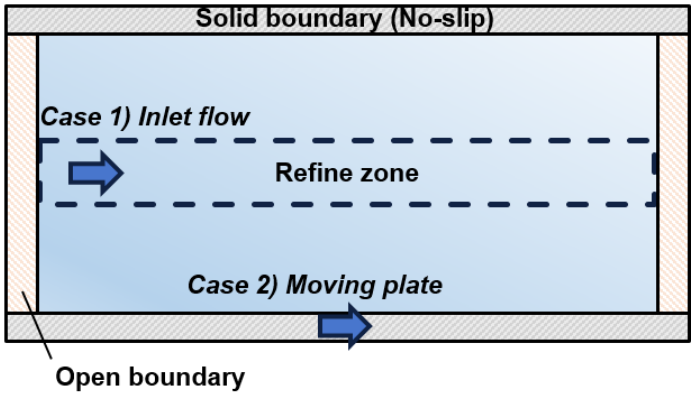


Figure 4.7 Initial setup of the pipe flow simulation

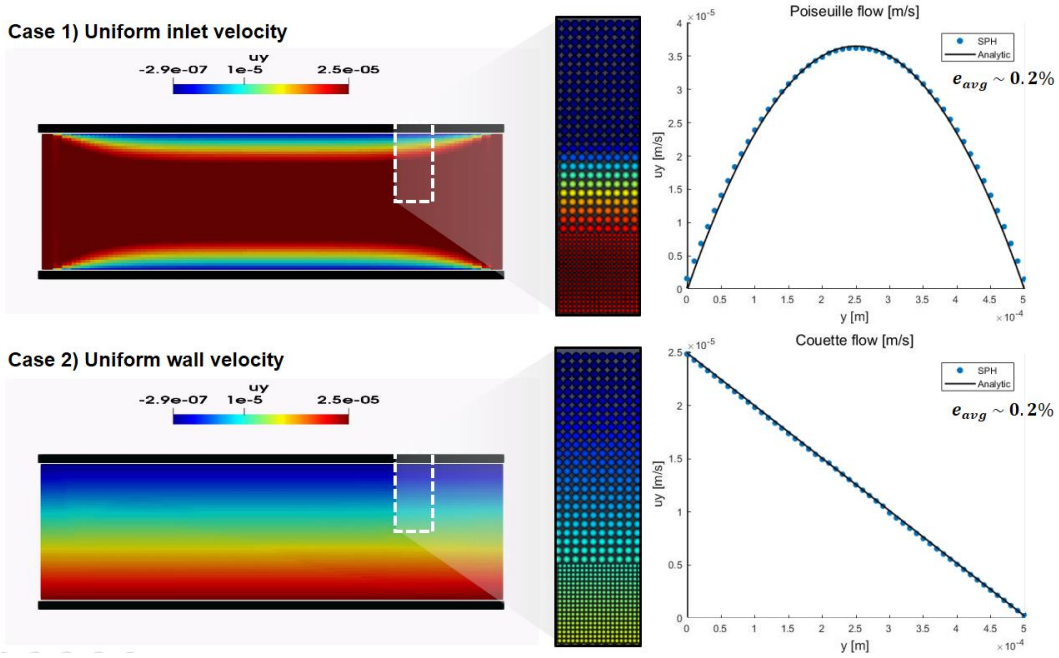


Figure 4.8 Velocity profile in fully-developed flow

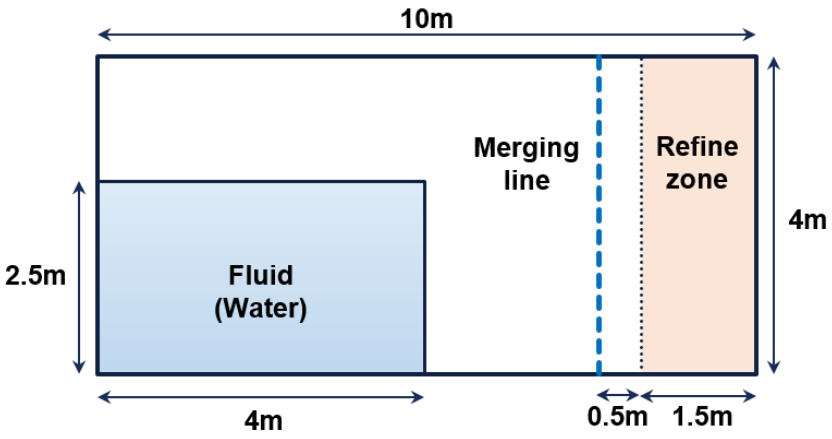


Figure 4.9 Initial setup for the Dam break simulation

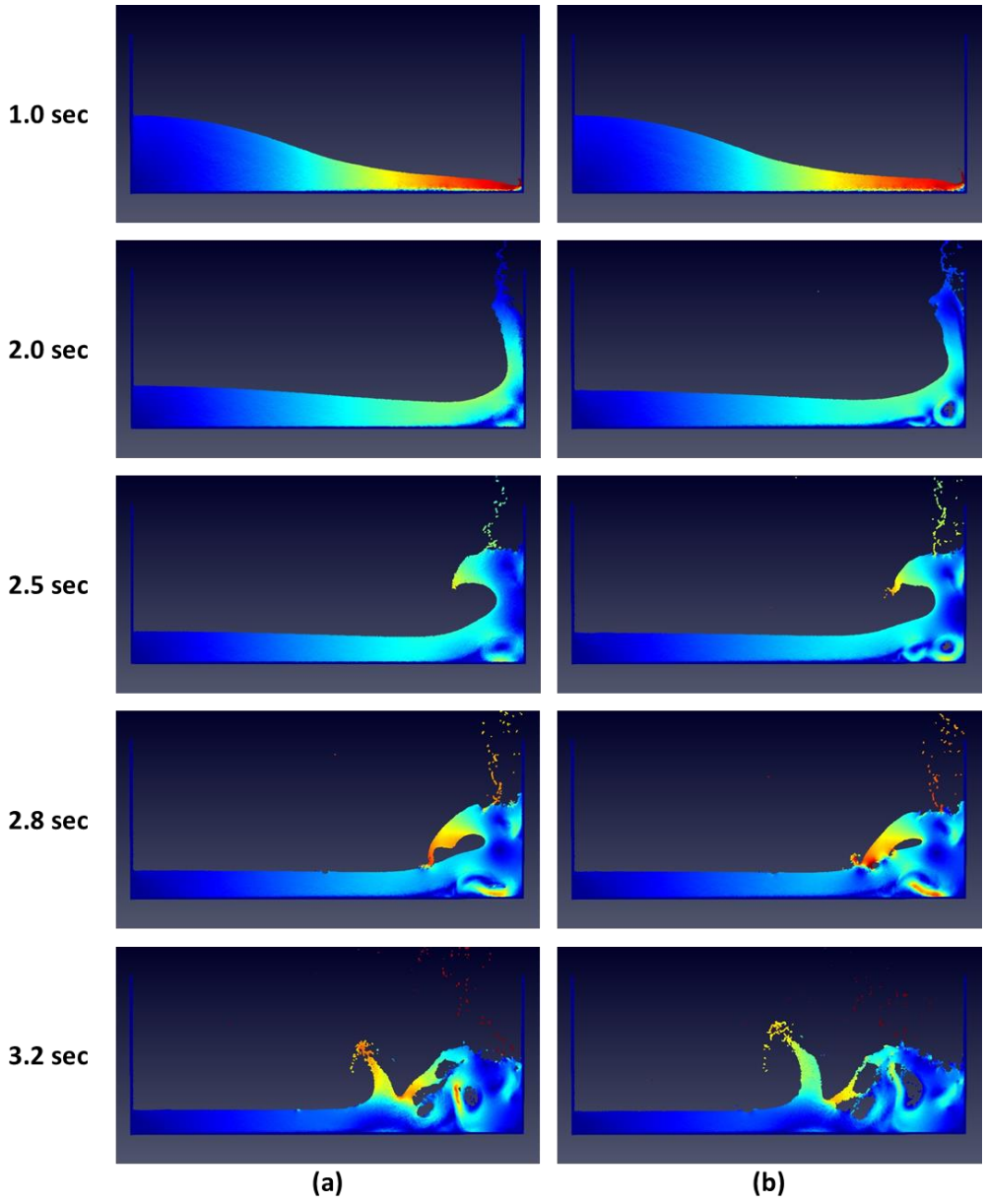


Figure 4.10 Snapshots of Dam break simulation with different resolution
(a) High-resolution (b) Multi-resolution

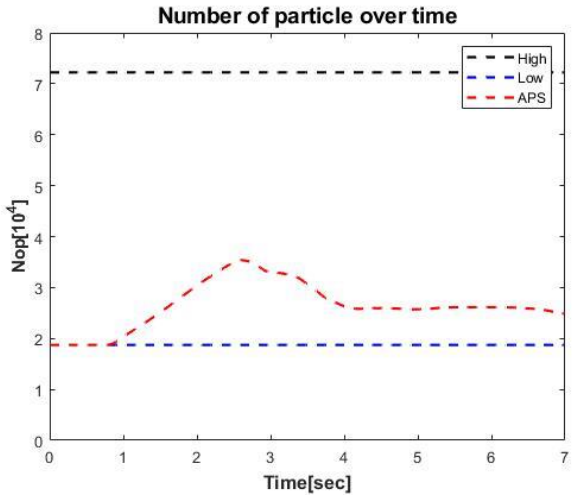


Figure 4.11 Number of particles used over simulation time

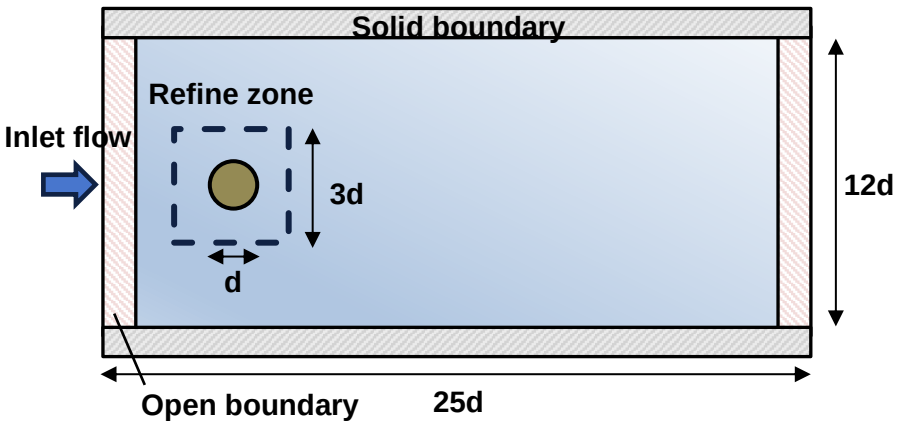


Figure 4.12 Initial setup of pipe flow simulation

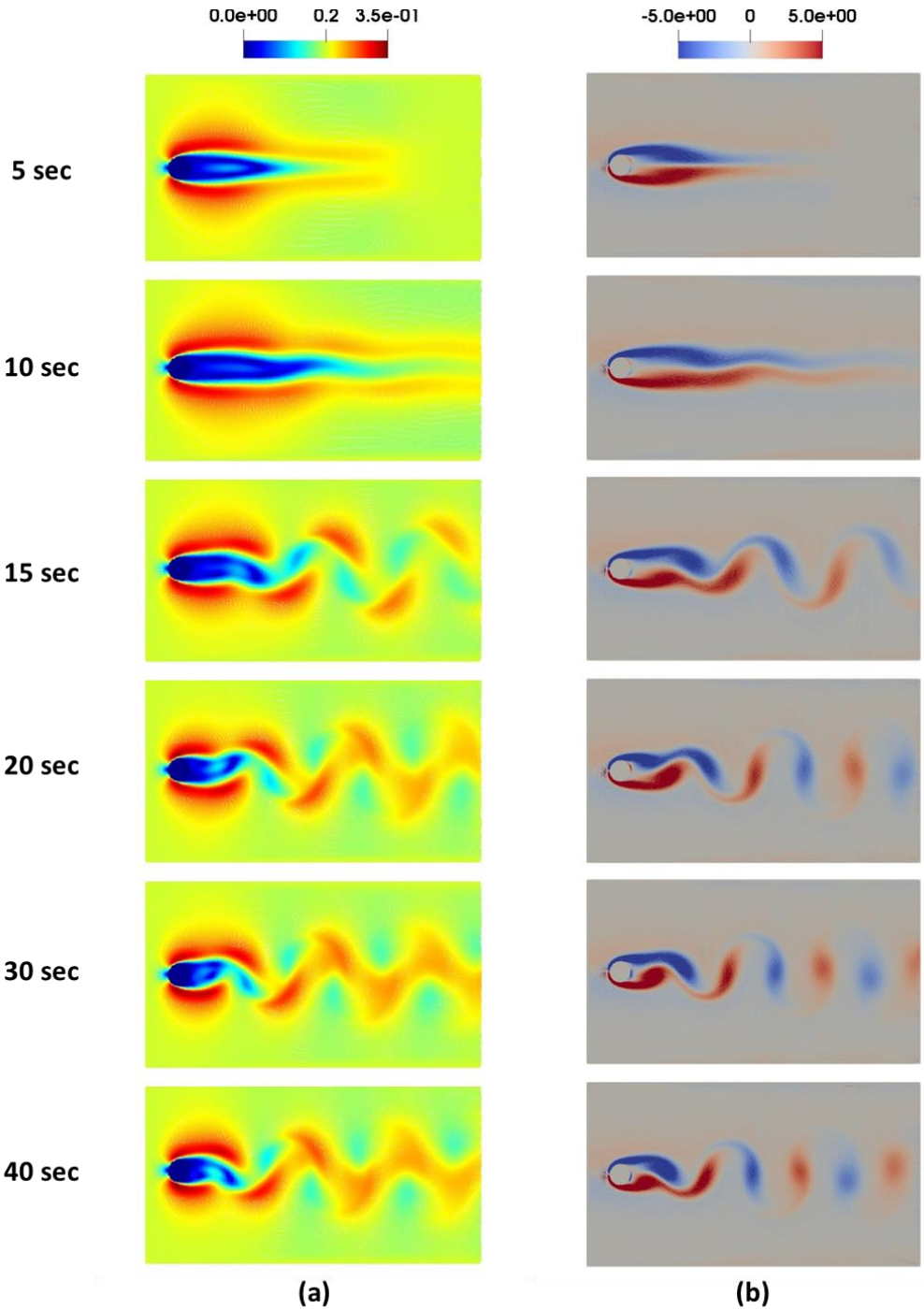


Figure 4.13 Snapshots of the Karman Vortex simulation
(a) Velocity (b) Vorticity

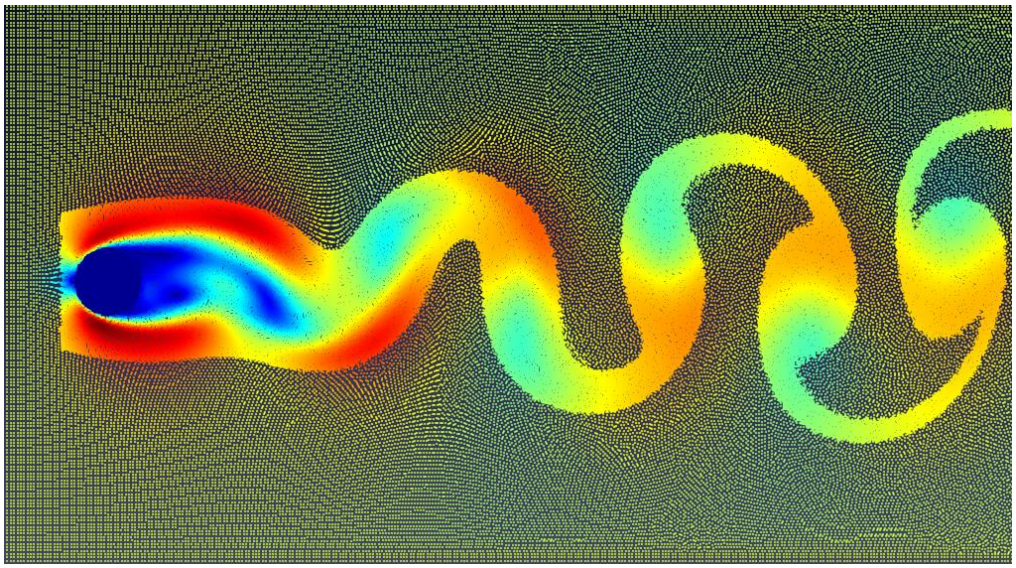


Figure 4.14 Multi-resolution analysis of Karman vortex

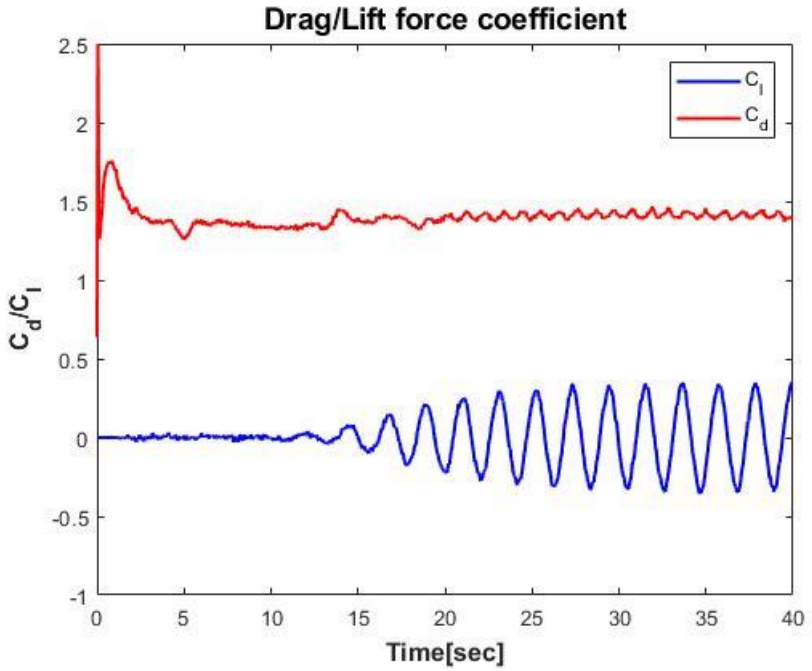
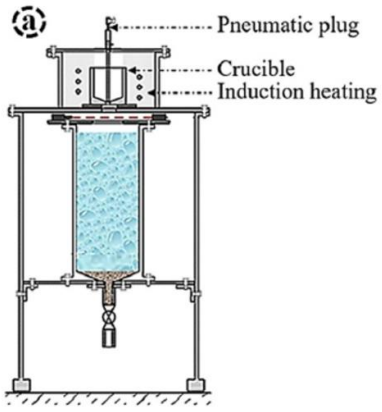


Figure 4.15 Time evolution of the drag and lift force coefficients



Manicam(2017)

Figure 4.16 Sketch of MISTEE-Jet facility

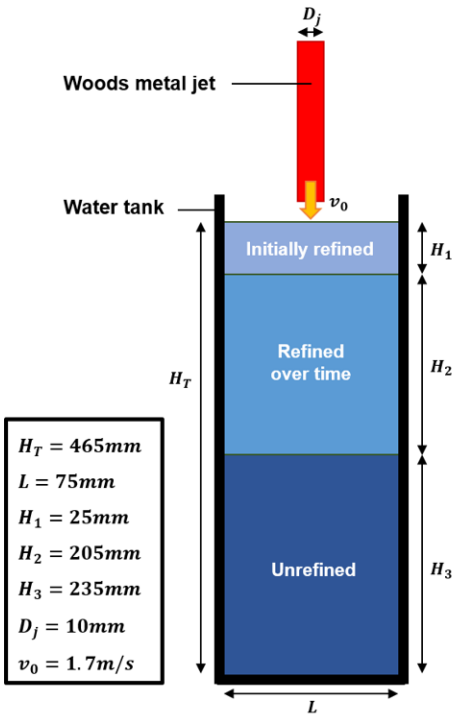


Figure 4.17 Initial setup for Jet breakup simulation

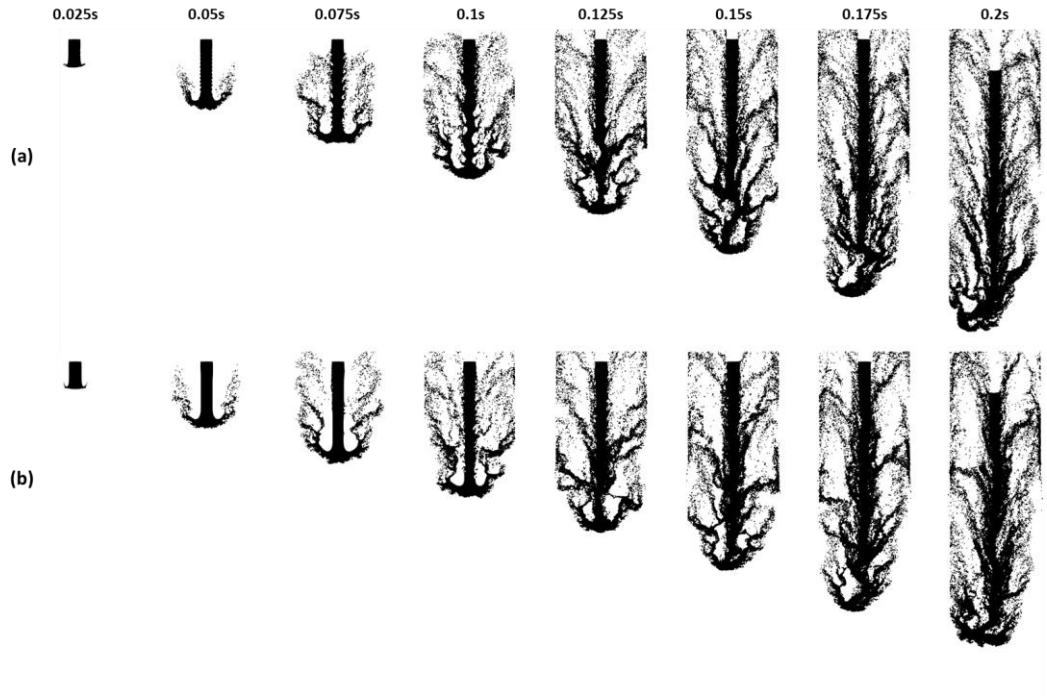


Figure 4.18 Snapshots of the jet break up simulation with different resolution
a) High-resolution b) Multi-resolution

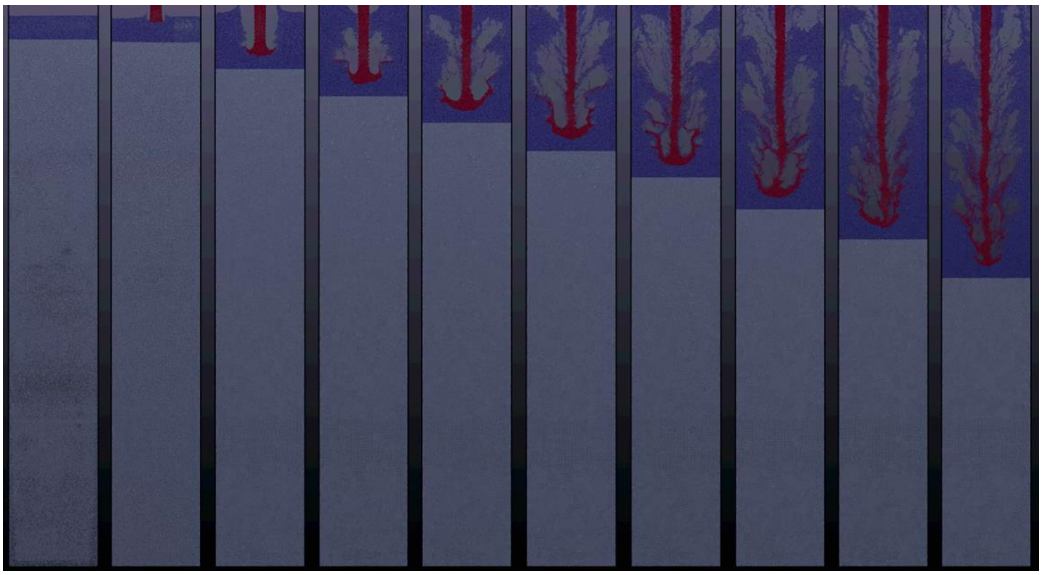


Figure 4.19 Snapshots of the jet break up simulation (colored with particle mass)

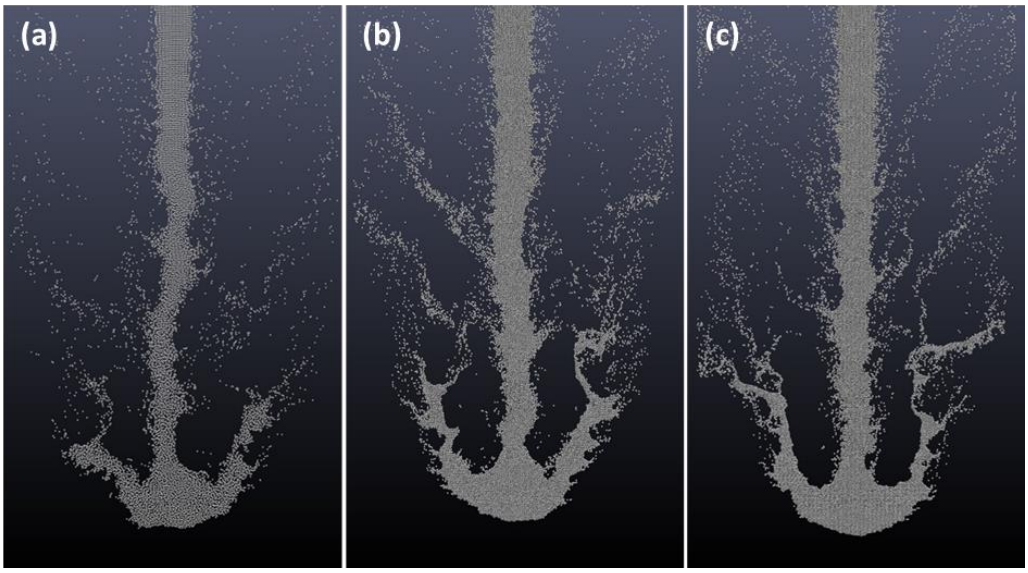


Figure 4.20 Leading edge of the jet in 0.1 sec
a) Low-resolution b) High-resolution c) Multi-resolution

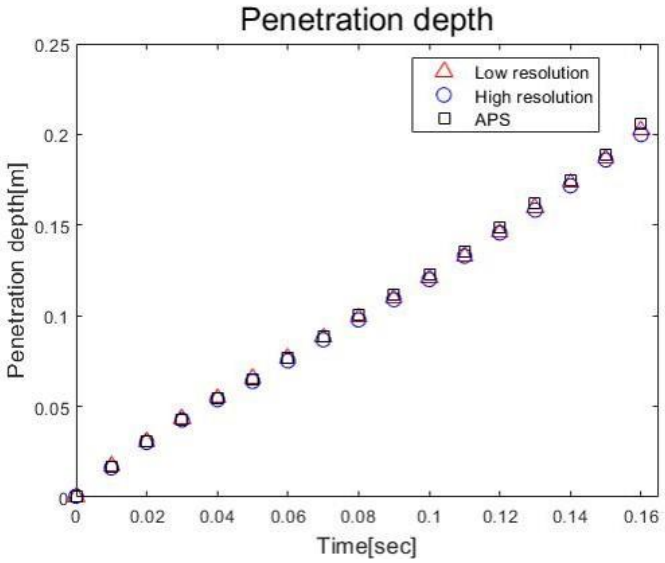


Figure 4.21 Comparison of the Jet penetration depth

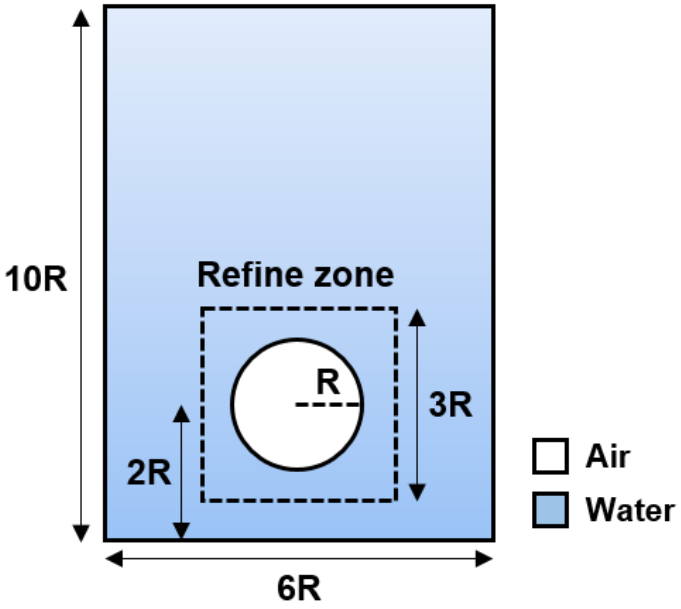


Figure 4.22 Initial setup for single bubble rising simulation

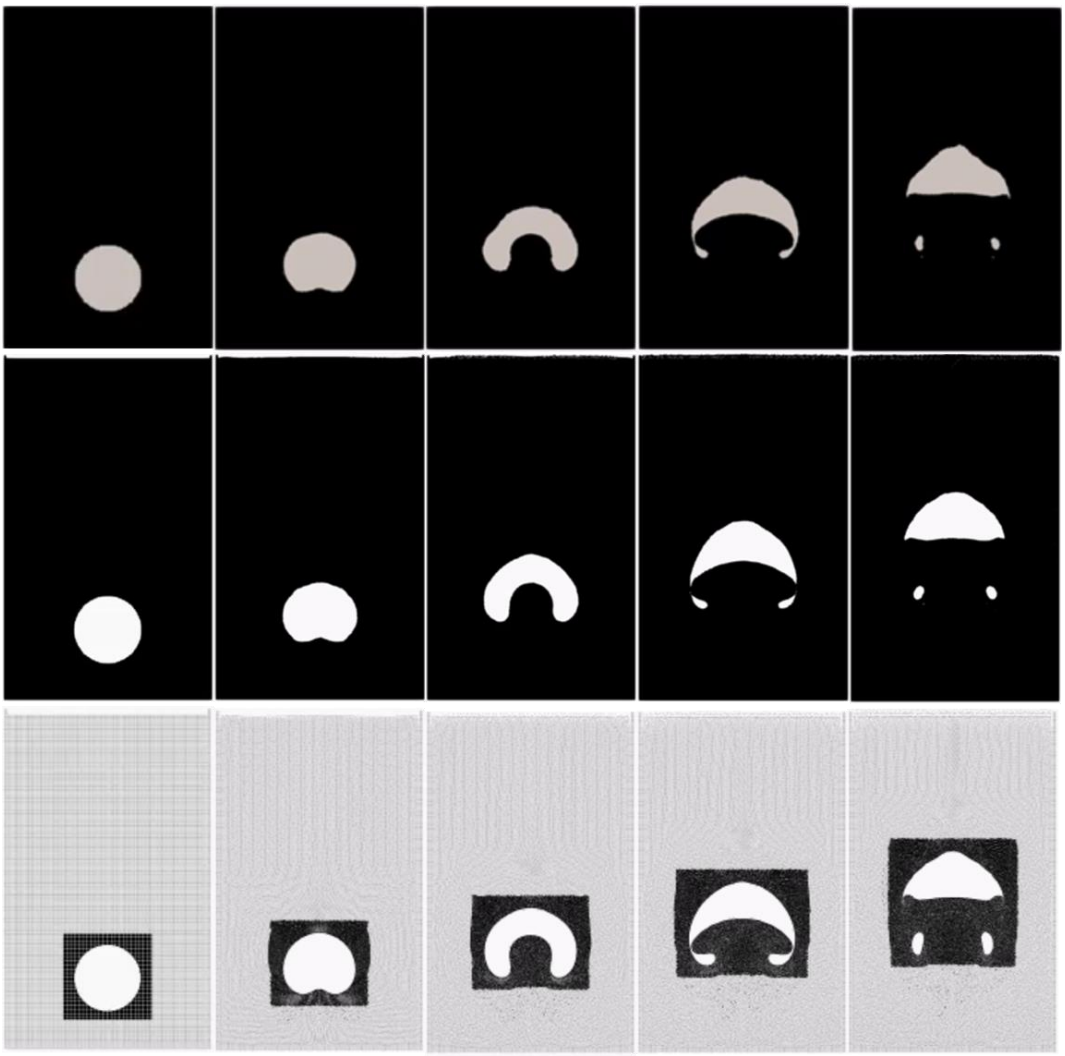


Figure 4.23 Snapshots of bubble rising simulation with different resolution (Re=100)

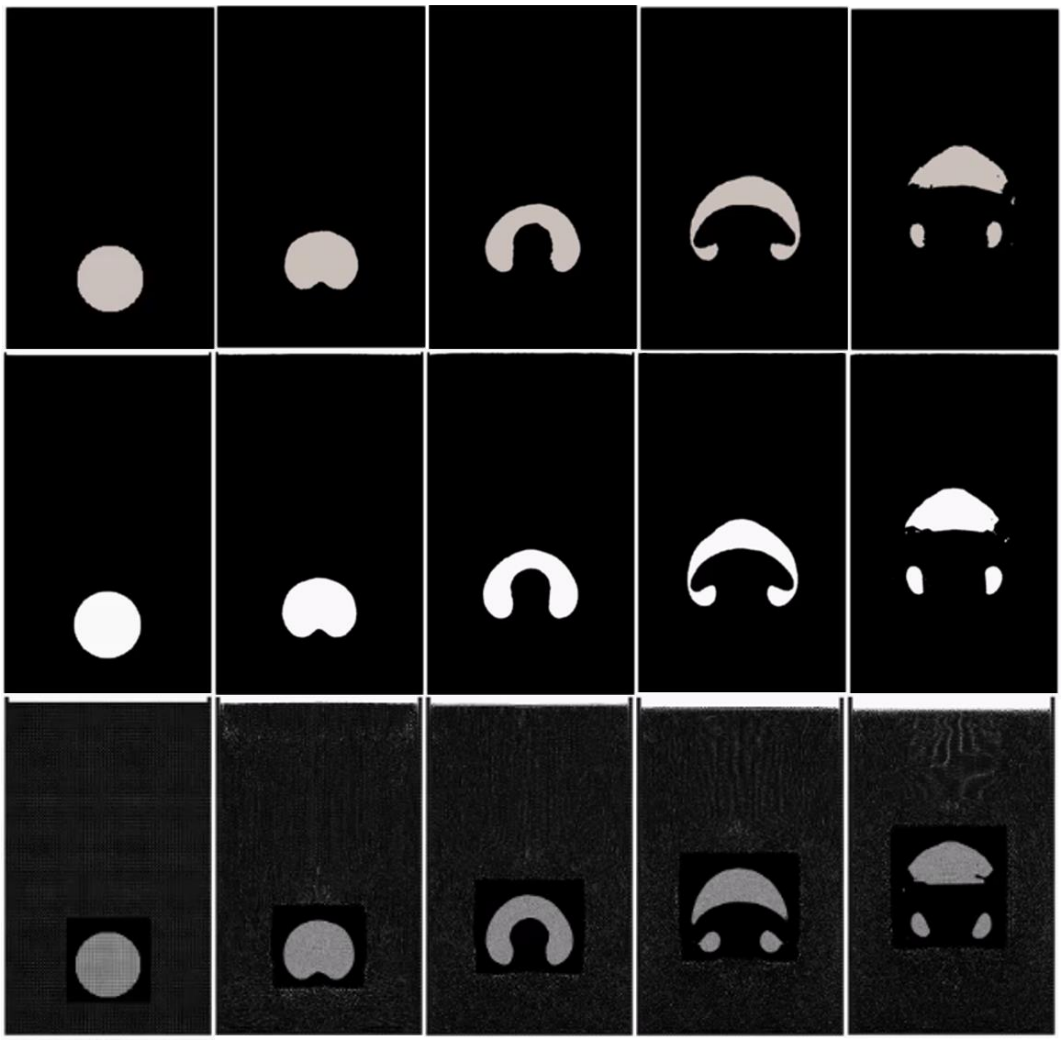


Figure 4.24 Snapshots of bubble rising simulation with different resolution (Re=1,000)

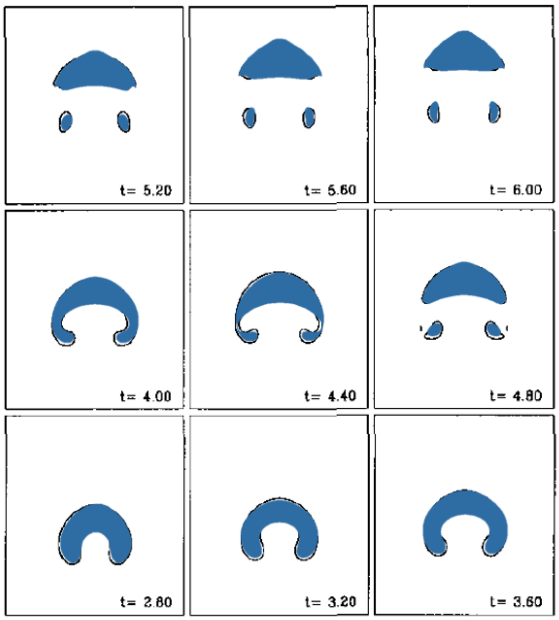


Figure 4.25 Single bubble rising simulation with level set results (Re=100)

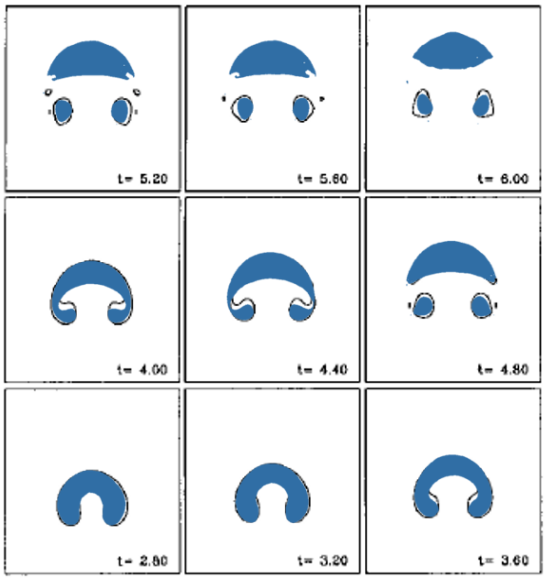


Figure 4.26 Single bubble rising simulation with level set results (Re=1,000)

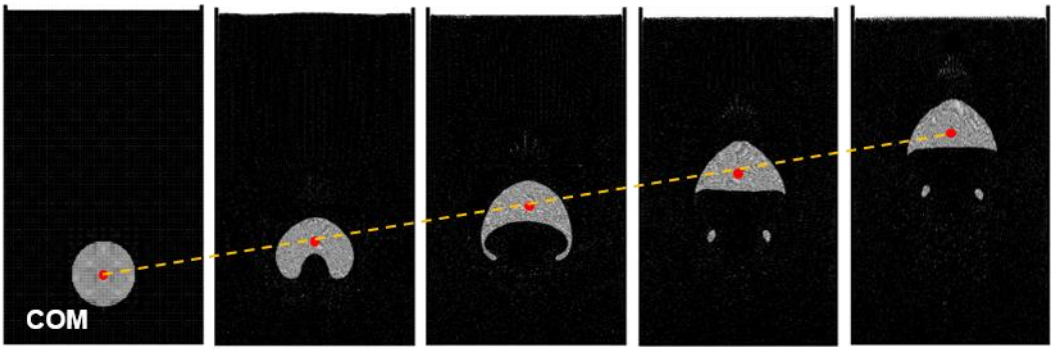


Figure 4.27 Axial position of bubble (Re=100)

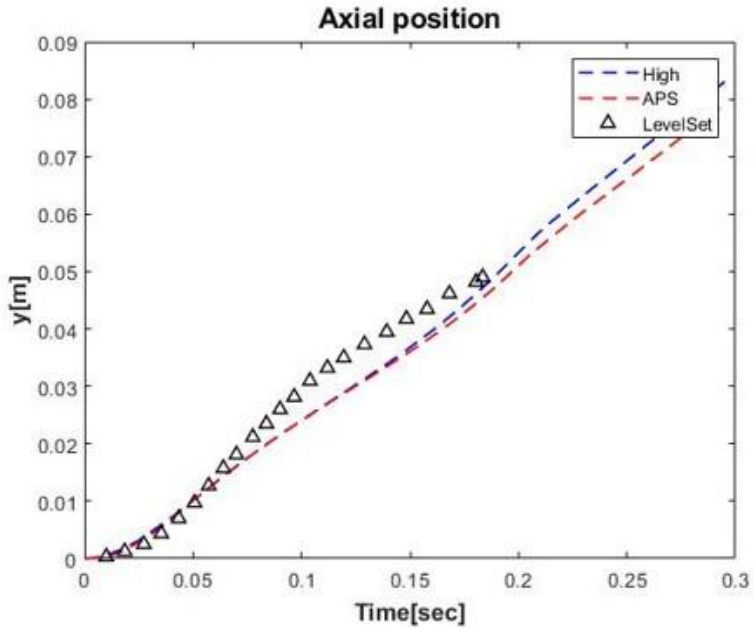


Figure 4.28 Axial position of bubble with level set result (Re=100)

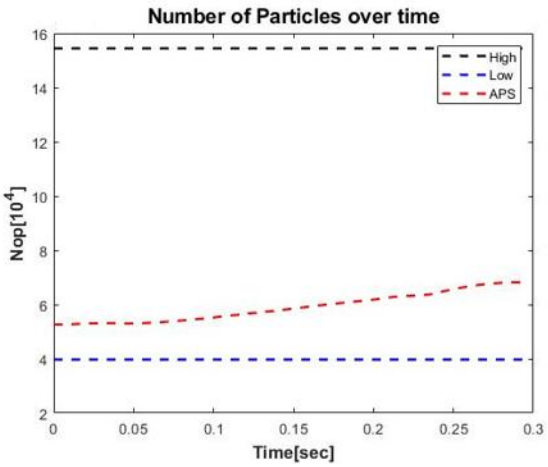


Figure 4.29 Number of particles used over simulation time

Chapter 5

Conclusion

5.1 Summary

In this study, the GPU-parallelized Adaptive Particle Refinement model was developed in SPH code and evaluated in terms of model stability, accuracy, and computational efficiency. The results, achievements, and findings of the study are summarized as follows.

1) Development of the APR model

Starting from the previous studies, the APR model was implemented in the SPH code and was improved for higher accuracy. Error analysis was performed to find the optimal refinement parameters. For model improvement, a new concept of merging, which can conserve the kinetic energy as well as mass and momentum, was presented and tested. To improve the accuracy at the interface between different resolutions, two methods have been proposed: 1) Continuous smoothing length model; 2) Stochastic refinement model.

2) GPU parallelization of APR model

Even if the APR model can improve the computational efficiency of the SPH code, the model itself should be accelerated using GPU to maximize the performance. In process of APR, there could arise the race condition, which is a problem that occurs when multiple GPU threads try to access the single memory address. To resolve the problem, a GPU mutex lock algorithm using CUDA atomic operation has been adopted. Using the algorithm, it was able to build a parallelized APR model that can avoid access collisions between threads even in parallel operations.

3) Application to various hydrodynamic problems

Using the developed APR model, various benchmark simulations were performed for validation of the model, as well as evaluation. The model showed good performance both in terms of computational accuracy and efficiency. For the application, three phenomena (Jet break up, air bubble rising, and gas combustion) were selected as the reference. Each simulation result was in good agreement with the reference data and showed that it is possible to build an accelerated multi-resolution scheme in SPH by overcoming the existing SPH methodologies based on the single particle resolution.

5.2 Recommendations

Through the present study, the following further studies are suggested.

1. The atomic operation used in the present study is contained in the basic library provided by CUDA C language, which cannot be modified by users.

Although the atomic operation is the most intuitive and reliable way to resolve the race conditions, there may be other algorithms that have better performance. Since using atomic operations slows the computational speed of the entire code, progress in the algorithm itself can greatly improve the overall computational efficiency.

2. APR model presented in this study uses discrete particle sizes. It can implement multiple resolutions within the computational domain, and maintain accuracy by adjusting the smoothing length, but when the initial particle size is determined, the particle size of other resolutions is inevitably predetermined. This method may have limitations in application depending on the complexity of the phenomenon, so a methodology that can smoothly change the resolution by continuously adjusting the particle size is required.

3. Although the APR model presented in this study includes a 3d refinement model, it has not been applied to the simulation in practice. For the simulation closer to the real phenomenon, it is necessary to introduce 3d analysis. Since the computational load of the SPH method rapidly increases with the number of particles, it is expected that the performance of the APR model can be maximized in 3d analysis.

Nomenclature

c_0	Sound speed (m/s)
d	Dimension (=1, 2, 3)
$\boldsymbol{f}_{ext}$	External body force acceleration (m/s^2)
$\boldsymbol{g}$	Gravitational acceleration (m/s^2)
h	Smoothing length (m)
m	Mass (kg)
$\boldsymbol{n}$	Surface normal vector
$\hat{\boldsymbol{n}}$	Unit surface normal vector
N	Number of particles
N_0	Initial number of particles
p	Pressure (Pa)
$\boldsymbol{r}$	Position vector
$\boldsymbol{r}_{ij}$	$\boldsymbol{r}_i - \boldsymbol{r}_j$
t	Time (sec)
Δt	Time step (sec)
$\boldsymbol{u}$	Velocity vector (m/s)
$\boldsymbol{u}_{ij}$	$\boldsymbol{u}_i - \boldsymbol{u}_j$
V	Volume (m^3)
V_0	Initial volume (m^3)
W	Kernel function

Greek

δ	Dirac delta function
γ	EOS stiffness parameter
κ	Curvature
μ	Dynamic viscosity (Pa·s)

ν	Kinematic viscosity (m^2/s)
Ω	Infinite volume domain
ρ	Density (kg/m^3)
ρ_{ref}	Reference density (kg/m^3)
σ	Surface tension coefficient (N/m)

Superscript

c	Corrector
p	Predictor

Subscript

c	Coarse particle
f	Fine particle
fp	Pressure force
fs	Surface tension force
fv	Viscous force
i	Center particle
j	Neighbor particle

Abbreviation

APR	Adaptive Particle Refinement
CPU	Central Processing Unit
CSF	Continuum Surface Force
EOS	Equation of State
FCI	Fuel Coolant Interaction
GPU	Graphics Processing Unit
NNPS	Nearest Neighbor Particle Search
SPH	Smoothed Particle Hydrodynamics
WCSPH	Weakly Compressible Smoothed Particle Hydrodynamics

References

Antuono, M., Colagrossi, M., Marrone, S. & Molteni, D. (2010). Free-surface flows solved by means of SPH schemes with numerical diffusive terms. Computer Physics Communications, 181(3), 532-549.

Barcarolo, D.A, Touze, D.Le, Oger, G., & Vuyst, F.De (2014). Adaptive particle refinement and derefinement applied to the smoothed particle hydrodynamics method, Journal of Computational Physics 273: 640 – 657

Braza, M., Chassaing, P. & Minh, H.H.(1986). Numerical study and physical analysis of the pressure and velocity fields in the near wake of a circular cylinder, J.Fluid Mech. 165: 79–130.

Cleary, P. W. (1998). Modelling confined multi-material heat and mass flows using SPH. Applied Mathematical Modelling, 22(12), 981-993.

Dehnen, W., & Aly, H. (2012). Improving convergence in smoothed particle hydrodynamics simulations without pairing instability. Monthly Notices of the Royal Astronomical Society, 425(2), 1068-1082.

Gingold, R.A., & Monaghan, J.J. (1977). Smoothed Particle Hydrodynamics: Theory and Application to Non-Spherical Stars, Monthly Notices of the Royal Astronomical Society 181: 375-389.

Jo, Y. B., Park, S.H., Choi, H.Y., Jung H.W., Kim Y.J. & Kim, E.S. (2019). SOPHIA: Development of Lagrangian-based CFD code for nuclear thermal-hydraulics and safety applications, Annals of Nuclear Energy, Vol. 124, 132-149.

Lin, L., Dinh, N.T., Prescott, S.R. & Bao, H. (2019). Assessment of smoothed particle hydrodynamics methods for simulating the external-flooding scenario, NURETH-18

Lind, S.J., Xu, R., Stansby, P.K. & Rogers, B.D. (2012). Incompressible smoothed particle hydrodynamics for free-surface flows: A generalized diffusion-based algorithm for stability and validations for impulsive flows and propagating waves, Journal of Computational Physics 231, 1499-1523

Liu, G.R., & Liu, M.B. (2003). Smoothed particle hydrodynamics: a meshfree particle method. World scientific.

Liu Hu, Q, HongFu, C, FuZhen, Shi Chao (2020). A particle refinement scheme with hybrid particle interacting technique for multi-resolution SPH, Engineering Analysis with Boundary Elements 118. 108-123

Liu, W.T., Sun, P.N., Ming, F.R. & Zhang, A.M.(2017). Application of particle splitting method for both hydrostatic and hydrodynamic cases in SPH, Acta Mechnica Sinica, vol.34: 601-613

Manickam, L., Bechta, S. & Ma, W. (2017). On the fragmentation

characteristics of melt jets quenched in water, International Journal of Multiphase Flow, Vol. 91, 262-275.

Marrone, S., et al. (2013). An accurate SPH modeling of viscous flows around bodies at low and moderate Reynolds numbers. Journal of Computational Physics **245**: 456-475.

Molteni, D., & Colagrossi, A. (2009). A simple procedure to improve the pressure evaluation in hydrodynamic context using the SPH. Computer Physics Communications, 180(6), 861-872.

Monaghan, J.J. (1992). Smoothed particle hydrodynamics. Annual review of astronomy and astrophysics, 30(1), 543-574.

Monaghan, J.J. (1994). Simulating free surface flows with SPH, Journal of computational physics 110, No.2, 399-406.

Morris, J. P. (2000). Simulating surface tension with smoothed particle hydrodynamics. International journal for numerical methods in fluids, 33(3), 333-353.

Park, S., Park, H. S., Jang, B. I. & Kim, H. J. (2016). 3-D simulation of plunging jet penetration into a denser liquid pool by the RD-MPS method, Nuclear Engineering and Design, Vol. 299, 154-162.

Sussman, M., Fatemi, E., Osher, S., & Smereka, P. (1995). A level set approach for computing solutions to incompressible two-phase flow II (No. UCRL-JC-121224; CONF-9509183-3). Lawrence Livermore National Lab., CA (United States).

Sun, P. N., Touzé, D. Le, Oger, G. & Zhang, A. M. (2019). Derivation and validation of a δ-SPH model for simulating strongly-compressible multiphase flows, The 14th SPHERIC International Workshop, 84-91.

Tafuni, A., et al. (2018). A versatile algorithm for the treatment of open boundary conditions in Smoothed particle hydrodynamics GPU models. Computer Methods in Applied Mechanics and Engineering **342**: 604-624.

Vacondio, R. et al. (2021). Grand challenges for Smoothed Particle Hydrodynamics numerical schemes, Computational Particle Mechanics, vol. 8, 575-588

Vacnodio, R., Rogers, B.D., Stansby, P.K., Mignosa, P. & Feldman, J. (2013). Variable resolution of SPH:A dynamic particle coalescing and splitting scheme, Computational Methods in Applied Mechanics and Engineering. 256. 132-148

Vacnodio, R., Rogers, B.D., Stansby, P.K. & Mignosa, P. (2013). Shallow water SPH for flooding with dynamic particle coalescing and splitting, Advances in Water Resources 58, 10-23

Vacnodio, R., Rogers, B.D., Stansby, P.K. & Mignosa, P. (2016). Variable resolution for SPH in three dimensions: Towards optimal splitting and coalescing for dynamic adaptivity, Computational Methods in Applied Mechanics and Engineering. 300. 442-460

Wei Hu, Gannan Guo, Xiaozhe hu, Dan Negrut, Zhijie Xu & Wenxiao Pan (2019). A consistent spatially adaptive smoothed particle hydrodynamics method for fluid-structure interactions, Computer Methods in Applied mechanics and Engineering. 347: 402-424

Xiufeng Yand & Song-Charng Kong (2017). Smoothed particle hydrodynamics method for evaporating multiphase flows, American Physical Society, E 96

Zhang, A., Sun, P.N., & Ming, F. (2015). SPH modeling of bubble rising and coalescing in three dimensions, Computational Methods in Applied Mechanics and Engineering 294: 189-209

국문 초록

최근 원자력 안전 관련 현안들은 열수력 현상 뿐만이 아닌 핵연료 용융, 구조, 재료, 화학반응, 다상 유동 등을 포함하는 매우 복잡한 현상들로 이루어진다. 전통적인 원자로 안전 해석은 주로 오일러리안 격자 기반의 수치해석 방법에 기반하지만, 최근에는 유체 시스템을 유한 개의 유체 입자의 집합으로 표현하는 라그랑지안 입자 기반 방법론 역시 활발하게 연구되고 있다. 대표적인 라그랑지안 기반 해석 기법인 완화입자유체동역학(Smoothed Particle Hydrodynamics : SPH)은 입자를 직접 추적하는 특성으로 인해 앞서 언급한 복잡한 물리 현상들이 포함하는 자유표면이나 다상 유동 등의 유동적인 계산 영역을 해석하는 데에 용이하다.

입자 기반의 유체 해석에서 높은 해상도는 일반적으로 높은 정확도의 결과를 보장하지만, 이는 해석 영역 내 입자 수의 증가에 따른 높은 계산 부하를 야기한다. 현존하는 대부분의 입자 기반 해석 코드는 계산 영역 전체에서 동일한 크기의 입자를 사용하는 단일 해상도(Single-resolution) 방식을 채택하고 있다. 이러한 방식은 난류 해석, 비등/응축 해석, 충격파 해석 등과 같이 유동 영역에 따라 다른 수준의 입자 해상도를 요구하는 해석의 경우 불필요한 계산 부하가 형성되거나, 오히려 계산의 정확도를 저하시킬 수 있다. 따라서 이를 개선하기 위해 계산 영역 내에서 국부적으로 입자의 크기를 조절할 수 있는 다중 해상도(Multi-resolution) 해석의 도입이 필요하다.

이에 따라 본 연구에서는 SPH 기법을 기반으로 한 입자 분할/병합 방법론(Adaptive Particle Refinement : APR)을 개발하고, 모델의 가속화를 위해 이를 GPU 병렬 계산에 적합한 형태로 구현하였다. APR 방법론의 기본 개념은 계산 중 특정 조건에서 입자를 분할하거나 병합함으로써, 계산 영역 내의 국부적인 영역에서 서로 다른 해상도로 해석을 수행하는 것이다. 입자는 특정 조건(입자의 위치, 부피, 속도 구배 등)에서 여러 개로 분할되거나, 또는 여러 개의 입자가 더 적은 수의 입자로 병합되는 과정을 통해 계산 내에서 다양한 입자 해상도를 구현할 수 있다. 하지만 기존 연구에서 사용된 방식들은 입자를 병합하는 과정에서 병합 입자의 속도가 기존 입자들의 운동량 보존식만으로 결정한다. 이러한 방식은 질량과 운동량을 잘 보존하지만 입자의 운동 에너지를 보존하지 못하기 때문에, 본 논문에서는 운동 에너지 보존을 위한 새로운 병합 모델이 제시되었다. 또한, APR 과정에서 입자의 완화 거리(Smoothing length)를 변화시키는 경우, 서로 다른 크기의 입자가 상호작용하는 해상도의 경계에서 계산의 정확도를 떨어트릴 수 있다. 이를 개선하기 위한 연속적 완화 거리 변화 모델 역시 제안되었다.

GPU 병렬 계산의 특성상, 하나의 메모리에 여러 스레드가 동시에 접근하여 연산을 수행할 경우 스레드 간 연산의 순서가 꼬여 기대하던 것과 다른 결과를 도출하는 경쟁 조건이 발생할 수 있다. APR 방법론을 병렬화 할 경우 새롭게 생성되는 입자들을 저장하는 과정에서 이러한 경쟁 조건이 발생하여 생성 입자의 메모리 주소가 충돌하는

현상이 발생한다. 이를 해결하기 위해 CUDA C 언어가 제공하는 원자 연산을 이용하여 스레드 간 계산의 간섭을 방지할 수 있는 잠금 알고리즘을 구현하였고, 과도한 직렬화로 인한 계산 속도 저하를 방지하기 위해 알고리즘을 최적화하였다.

적용된 APR 방법론을 검증하고 성능을 평가하기 위해 다양한 시뮬레이션에 대한 검증 해석이 수행되었다. 정수압 형성, 관내 유동, 댐 붕괴, 그리고 칼만 와류에 대한 해석을 통해 개발된 APR 모델이 안정적으로 다중 해상도를 구현할 수 있음을 확인하였고, 높은 정확도와 계산 효율을 보이는 것을 확인하였다. 또한 제트 파쇄 해석과 공기 방울 상승 해석을 통해 다유체, 다상 유동에의 적용을 수행하였고, 실험 데이터와의 정략적으로 비교하였다. 분석 결과, 시뮬레이션이 실제 현상을 잘 모사함이 확인되었으며, 입자 수 조절을 통해 계산 효율이 크게 향상되었다.

본 연구에서는 SPH 방법론을 기반으로 한 입자 분할/병합 모델을 개발하고 GPU를 이용하여 최적화함으로써, SPH 방법론 내에 다중 해상도 해석 체계를 구축하였다. 이는 기존 단일 해상도의 입자 기반 해석 체계가 필연적으로 가지고 있었던 해상도 증가에 따른 과도한 계산 부하 문제에 대한 해결책을 제시할 수 있다는 점에서 의의를 가지며, 원자로 중대 사고 해석과 같이 현상 내에서 여러 입자 해상도를 요구하는 복잡한 유동에 대한 해석에 기여할 수 있을 것으로 예상된다.

주요어 완화입자유체동역학, 입자 분할/병합 방법론, 다중 해상도 해석, GPU 병렬화, 경쟁 조건, 원자 연산

학번 : 2019-27978