論文 / 著書情報
Article / Book Information

題目(和文)	
Title(English)	Study on Large-scale 3-D Simulations for Driftwood Disasters
著者(和文)	SHEN DAWEI
Author(English)	Dawei Shen
出典(和文)	学位:博士(工学), 学位授与機関:東京科学大学, 報告番号:甲第262号, 授与年月日:2025年3月26日, 学位の種別:課程博士, 審査員:青木 尊之,肖 鋒,大西 領,伊井 仁志,門永 雅史
Citation(English)	Degree:Doctor (Engineering), Conferring organization: Institute of Science Tokyo, Report number:甲第262号, Conferred date:2025/3/26, Degree Type:Course doctor, Examiner:,,,,
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

Study on Large-scale 3-D Simulations for Driftwood Disasters

by

Dawei Shen

Supervisor: Professor Takayuki Aoki

Submitted in partial fulfillment
of the requirements for the degree of
Doctor of Engineering

Department of Mechanical Engineering
School of Engineering
Institute of Science Tokyo

March 2025

Abstract

Recently, severe damage to cities from river flooding and driftwood has become a critical problem. In order to better understand this problem and improve its study, we developed large-scale 3-D simulations for water flow which includes large amounts of driftwood. This code uses the cumulant lattice Boltzmann method (LBM) and the discrete element method (DEM) to simulate the fluid phase and the solid phase, respectively. The free surface of the river flooding is described by solving the conservative Allen-Cahn equation of the phase-field model. Adaptive mesh refinement (AMR) makes it possible to apply the code to large areas with a high-resolution mesh, and a GPU implementation greatly accelerates the computation.

For disaster prediction, we wanted to understand the driftwood accumulation locations and identify the trapping mechanism of the driftwood in an actual river flooding disaster. For this, simulations for a small-scale experiment and a real case of river flooding were conducted. The accuracy of the present method was validated by simulating the small-scale experiment and comparing the computational results, which were in good agreement with the experimental results. As an application, two large-scale computations were conducted as case studies. These cases were based on an actual river flooding disaster that occurred on the Omoto Rvier in Iwate Prefecture, Japan, including 1,000 pieces of driftwood and a maximum resolution of 7.8125 cm. The simulation results showed that the proposed model could generally replicate the inundation process and the driftwood accumulation positions like the real disaster. Moreover, three driftwood accumulation processes were found: trapping in narrow channels, hitting building corners, and stagnation in front of buildings.

For disaster mitigation, installing driftwood trap devices on the impervious concrete dam was proposed. To evaluate the effectiveness and contribute to the design of trap devices, two hydraulic model experiments and a real-scale dam with the driftwood trap device were simulated. Simulation results for hydraulic model experiments showed good agreement with the experimental results, thereby validating the feasibility of the proposed method in simulating the interactions between the driftwood, the trap device, and the water flow. For a real-scale dam with a driftwood trap device, we examined flows including 1,000 pieces of driftwood with different parameters. It was found that the number of trapped pieces of driftwood strongly depended on the driftwood length. Namely, more than 90% of driftwood

was captured when the length was greater than 75% of the trap device gap. We showed that the driftwood distribution under the water and the angle to the water surface was smaller for longer driftwood. Additionally, it was discovered that the dammed-up depth in front of the trap device had a strong correlation with the porosity and the width of the trapped area.

Key words: ***River flooding***, ***Driftwood***, ***Cumulant LBM***, ***DEM***, ***AMR***.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisor, Prof. Takayuki Aoki, for giving me the opportunity to pursue PhD degree in the Aoki Laboratory, for his continuous support and insightful guidance during my doctoral period. Your expertise, patience, and commitment to excellence have been invaluable to me.

I would also like to thank my lab-mates in Aoki Laboratory, especially Dr. Seiya Watanabe, who provides me with the LBM code and countless guidance, and Dr. Kai Yang, who gives me a lot of guidance in CFD study and life. I deeply appreciate Prof. Takao Nagasaki, Dr. Marlon Arce Acuna, Dr. Kenta Sugihara, Dr. Shintaro Matsushita, Dr. Yos Sitompul and Dr. Xuyang Chen, for advising me. Thank you Eric Tada de Souza, Jun Kawahara, Nishanth Baskaran, Tan Hong Guan, Uchida Haruki, Sho Kitagawa, Yuta Taki and Kazuki Shimohata for the discussions and for taking classes together. I want to say thanks to Tongda Lian, Haocong Wang, Yuwei Yin for the insightful discussions and friendship.

I want to thank Prof. Shinsuke Takase, for providing me with the experimental results of the drifwood capturing experiment. I also thank Prof. Moriguchi Shuji, for providing the geometry information of the Iwaizumi Town. I appreciate Prof. Changhong Hu, for admitting me as an exchange student at Kyushu University and giving me countless guidance.

I would also like to express my thanks to former secretary Akiko Ito, current secretary Ayako Kaneko and Satomi Sugimoto, for their continuous help in my study and life.

I want to thank my girlfriend Jenny Ji. Words cannot express my gratitude for your unconditional love and unwavering belief in me. Your understanding provided the foundation I needed to persevere through the difficult times. This achievement is as much yours as it is mine.

Last but not least, I want to express my special thanks to my parents. Thank you very much for raising me up and supporting me in my education. Thank you for your continuous financial and emotional support.

This research was partly supported by a Grant-in-Aid for Scientific Research (S) 19H05613, from the Japan Society for the Promotion of Science (JSPS), High Performance Computing Infrastructure (HPCI), hp240034 and hp210129, JST SPRING, grant number JPMJSP2106, and the Joint Usage/Research Center for Interdisciplinary Large-scale Information Infras-

tructures (JHPCN), jh200018 and jh210013. I want to thank the Supercomputing Research Center, Institute of Science Tokyo for the use of the computing resources of the TSUBAME 3.0 and TSUBAME 4.0 supercomputers and the Information Technology Center of Nagoya University for the use of the computing resources of the Flow Type II supercomputer.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

River flooding has long been a significant threat to human life and has recently wreaked havoc in Japan, with hundreds of deaths and billions of dollars in property losses. On August 30, 2016, Typhoon No. 10 caused extremely heavy rainfall in the Iwaizumi Town of Iwate Prefecture, leading to an immediate rise in the water level in the Omoto River. Several buildings, including an elderly care facility, were inundated, and nine lives were lost. A substantial amount of driftwood had piled around the affected buildings. It is considered that the damage to structures is greatly increased when driftwood or other solid materials are included in a flood flow [1], as illustrated in Fig. 1.1. Thus, proposing mitigation measures to reduce the damage from driftwood is crucial. Since the early Heisei era to the present, driftwood control measures have primarily involved the installation of driftwood trap devices at the forefront of impermeable dams. However, existing impermeable dams lack the capability to capture floating objects such as driftwood, making it necessary to incorporate this function into these dams.

Two kinds of driftwood trap devices are commonly employed based on the types of river channels. In river channels with primarily flat terrain and wide riverbeds, inlet-type driftwood capture devices are widely used. These devices utilize a bypass channel to effectively guide and capture the driftwood [2]. On the other hand, river channels located in mountainous areas lack the conditions necessary for establishing bypass channels due to their narrow

widths. In such cases, driftwood capturing facilities installed on the upstream face of the impervious closed-type concrete dam are commonly adopted to directly gather and trap the driftwood [3]. This research focuses on the second type of driftwood trap device.

Figure 1.1: The damage of driftwood to structures.

For a long time, experiments, theoretical analysis, and numerical computations have been the three primary research methods used in fluid mechanics. The experimental method is the most traditional approach for fluid analysis, and it plays a vital role in the development of fluid mechanics. Experimental methods continue to provide a foundation of data for analysis and calculations up to the present time. The second approach is theoretical analysis, which is constrained to idealized or simplified cases such as Poiseuille and Couette flows [4] because of the complexity of fluid flow equations. Based on the principles of continuity and conservation of mass, momentum, and energy, the Navier-Stokes equations [5, 6] are one of the representative theories to characterize fluid motion. With the rapid advancement of computer science in the 1950s, numerical computations gained increasing importance in the study of fluid dynamics. Consequently, computational fluid dynamics (CFD) [7] has emerged.

Numerous experimental studies have focused on river flooding disasters, the driftwood accumulation process caused by flooding, and the effectiveness of driftwood trap devices on

impervious dikes [8, 9]. Bocchiola et al. [10] observed two stopping modes of driftwood: "leaning against one obstacle" and "bridging two obstacles". Okamoto et al. [11] revealed the mechanism governing the sinking motion of driftwood accumulating at bridges. Shima et al. [3] conducted hydraulic model experiments to clarify the differences in driftwood capture between ponding and wash flow conditions, and driftwood capture devices were installed upstream of the flow passage of the dam.

Numerical simulations have recently become practical tools for investigating river flooding with driftwood and trap devices [12, 13, 14, 15, 16]. Shimizu and Osada [17] developed a 2-D depth-averaged flow motion model and a 2-D discrete element model to describe the behavior of driftwood motion and their accumulation process in channels with bridge piers. The simulation results presented the driftwood accumulation process in front of bridge piers but the vertical stack of driftwood could not be reproduced using 2-D model. Shibuya et al. [18] reproduced the movement and capture of driftwood under wash flow conditions using numerical simulations based on the discrete element method [19]. Nevertheless, since a flow velocity distribution model was used around the driftwood and driftwood capture devices, this study could only roughly simulate the movement of the water surface, making it difficult to be applied to precise large-scale driftwood disaster simulations. Kimura et al. [20] used a 3-D flow model, which solves Reynolds-averaged Navier-Stokes (RANS) equations [21] and continuity equation, and a 3-D driftwood model to simulate the performance of an inlet-type driftwood capture facility installed on a 400-meter-long river channel. However, due to the lack of mesh adaptation, it is reported that an insufficient grid resolution was used and the driftwood capture by the slit could not be reproduced.

Originated from the lattice gas automata, the lattice Boltzmann method (LBM) has gained increased acceptance over the past two decades [22, 23, 24, 25]. By assuming a finite sound velocity significantly greater than the flow velocity for an incompressible fluid and approximating it to a weakly compressible fluid, LBM can calculate the flow using a completely explicit approach without the need to solve Poisson's equation. This approach has made LBM a popular alternative simulation tool for large-scale computational fluid dynamics [26]. The computational process of LBM is split into collision and streaming steps on a rectangular mesh. In addition, LBM employs a single-step explicit time integration,

which is suitable for parallel computing. Compared with the SPH method [27], LBM is based on a simpler algorithm and allows for efficient memory access. However, LBM has certain drawbacks, such as its limitation in dealing with high density ratios of gas-liquid two-phase flows [28, 29, 30]. To address this limitation, a free-surface model [31] is adopted. Since the atmosphere does not strongly affect water flow dynamics, the free-surface model does not compute the airflow dynamics and only applies atmospheric pressure to the water surface.

Free-surface flows exhibit complex behaviors at the gas-liquid interface. To accurately track the interface, interface capturing methods such as the volume of fluid (VOF) method [32] and the phase field method [33] are frequently employed. While the VOF method excels in conserving volume, its accuracy in capturing the interface shape diminishes as the interface curvature approaches the grid resolution. To handle this limitation, the THINC (Tangent Hyperbola for Interface Capturing) method [34], which utilizes a hyperbolic tangent function for interpolation, has been proposed. However, in a turbulent gas-liquid two-phase flow, there will be minute VOF values floating in the gas phase or regions of low VOF values in the liquid phase, which will persist for an extended period and result in non-physical phenomena. Based on the free energy theory of non-equilibrium systems, the phase field method offers an alternative. In addition to the interfacial advection term in its time evolution equation, it incorporates interfacial diffusion and anti-diffusion terms, enabling it to resolve the non-physical artifacts in the VOF method. Furthermore, coupling the phase field method with the level set method [35] enhances the accuracy of calculating geometric information of the interface, such as the curvature and the normal vector.

In the simulation of river flooding with driftwood, the water surface deforms in time and space with complex shapes, and the driftwood diameter is typically less than several tens of centimeters. To accurately compute the interactions between the driftwood and water surfaces, as well as between the driftwood and trap devices, it is necessary to use a high-resolution mesh. However, real river disasters often spread over a wide area. If we were to use a uniform mesh, an unacceptably large amount of mesh would be needed. For free-surface flows including driftwood, the high-resolution mesh is only needed in specific regions: around the free surfaces of the water, driftwood surfaces, structures, trap devices, and river beds. In order to allocate high-resolution mesh to these regions and low-resolution mesh to regions

far from them, we employ the adaptive mesh refinement (AMR) method [36, 37, 38, 39, 40], which helps minimize memory usage and computational cost, and makes it possible to carry out simulations of large-scale free-surface flows including driftwood. Additionally, we leverage GPU computing to accelerate the computation because a single GPU processor has much higher floating-point operational performance and higher memory bandwidth compared with a CPU, making it more suitable for computational fluid dynamics applications [41, 42].

1.2 Research Purpose

Given the significant damage caused by driftwood during river flooding, it is crucial to employ numerical simulation methods to assess potential risks and evaluate the effectiveness of driftwood trapping devices. These simulations are vital for developing disaster prediction and mitigation strategies.

In this thesis, LBM is used for fluid dynamics analysis. The Single Relaxation Time (SRT) collision model [43] is commonly used due to its simplicity and ease of implementation. However, it is prone to numerical instability, particularly when handling high Reynolds number flows. To improve computational stability, the cumulant collision model [44] is adopted. Moreover, we couple the discrete element method (DEM) with LBM to simulate the interaction between driftwood pieces and their collisions with structures. In these simulations, the driftwood exhibits both rotational and translational motion. The original simulation code is developed in the Aoki laboratory. It is optimized to handle large-scale 3D river flooding simulations including driftwood, and new functions are developed to simulate driftwood trapping devices.

The first purpose, as an evaluation of our model for driftwood disaster prediction, is to apply the code to simulate a real river flooding disaster including driftwood, and elucidate the mechanism of driftwood accumulation during the disaster. Firstly, driftwood-capturing experiments were conducted in the water tank at the Hachinohe Institute of Technology. After obtaining the experimental settings and results, simulations were performed to assess the validity of our approaches in simulating driftwood disasters. Subsequently, we studied a real river flooding disaster involving driftwood and simulated the overflow and the inundation

processes. To identify the driftwood accumulation processes, we focused on specific areas in which a lot of driftwood accumulated. It should be noted that due to the lack of observational data, we made assumptions regarding the initial parameters, such as the water discharge at the inflow boundary, the duration of the disaster, shapes and sizes of the driftwood, the number and density of the driftwood, the location and timing of the driftwood entering the affected area, and the coefficient of friction between the driftwood and the river bed. The assumptions are deemed acceptable because the purpose of our simulation is not to accurately reproduce the real disaster situation, but to understand the trapping mechanism of driftwood in the real terrain with buildings.

The second objective, as a contribution of our model to the mitigation of driftwood disasters, is to use the simulation code to evaluate the effectiveness of driftwood trap devices installed on real-scale dams. Initially, we aimed to reproduce the interactions between pieces of driftwood and trap devices through 3-D simulations of two hydraulic model experiments. Following this, a full-scale simulation of a dam equipped with a driftwood trap device was conducted. Real-scale simulations can reveal issues that are difficult to assess in hydraulic model experiments, such as the influence of driftwood lengths on capture rates, the capture mechanism by driftwood interaction, and the movement of driftwood under the water surface. Furthermore, by developing a tool that can predict the effectiveness of driftwood traps without the need for physical experiments, we can contribute to the design of more efficient driftwood trap devices.

The remaining chapters of this thesis are organized as follows. Chapter 2 provides an overview of the numerical methods used in the simulation, LBM, DEM, phase field method, and their coupling. Details of the AMR method and its coupling with LBM are introduced in Chapter 3. Chapter 4 describes simulations conducted to verify our model, which were based on small-scale water flume experiments, and then introduces the simulations we performed on an actual river flooding event. Simulations of hydraulic model experiments for driftwood trap devices and a series of real-scale simulations for a driftwood trap device installed on an impervious concrete dam are performed and results are shown and discussed in Chapter 5. Finally, we make some concluding remarks and explain the originality in Chapter 6.

Chapter 2

Numerical Method

In order to investigate flows including driftwood, we compute the fluid dynamics of water and the motion of driftwood by taking into account of the interaction between the water and the driftwood. In this chapter, we introduce the numerical methods employed in this thesis. First, we describe the lattice Boltzmann method, which is used to calculate the fluid dynamics of water. Next, we outline the discrete element method for simulating the motion of solid objects. Following this, we explain the phase field method which is utilized for capturing the gas-liquid interface. Finally, we discuss the coupling of these methods.

2.1 Lattice Boltzmann Method

2.1.1 Lattice Boltzmann Equation

For the numerical simulations of incompressible flow, semi-implicit methods such as the finite difference method are commonly used to solve the Navier-Stokes equations. However, this approach requires iterative computation of the pressure Poisson equation. As the number of grid points increases, the convergence of the pressure Poisson equation deteriorates, leading to an increased number of iterations required for computation.

The lattice Boltzmann method (LBM) has become a popular alternative simulation tool for large-scale computational fluid dynamics. In 1988, McNamara and Zanetti [45] proposed converting the integer operations in lattice gas automata into real number operations, which

marked the inception of the lattice Boltzmann method. LBM models the fluid as a collection of virtual particles that propagate and collide at lattice points. The velocity distribution function, denoted as $f(\boldsymbol{x},\ \boldsymbol{\xi},\ t)$, is a function of the spatial position vector $\boldsymbol{x}(x,\ y,\ z)$, the molecular velocity vector $\boldsymbol{\xi}(\xi_x,\ \xi_y,\ \xi_z)$ and time t. The Boltzmann equation can be written as follows when the external force $\boldsymbol{F}$ is included:

$$\frac{\partial f}{\partial t} + \boldsymbol{\xi} \cdot \frac{\partial f}{\partial \boldsymbol{x}} + \boldsymbol{a} \cdot \frac{\partial f}{\partial \boldsymbol{\xi}} = \Omega, \tag{2.1}$$

where $\boldsymbol{a} = \boldsymbol{F}/m$ represents the acceleration of the fluid particle, m is the mass of the fluid particle and Ω denotes the collision term.

Solving the collision term of the Boltzmann equation is challenging due to its nonlinear nature and its dependence on molecular forces. To address this, Bhatnagar, Gross, and Krook proposed the BGK approximation [46], which simplifies the collision term to a linear operator. Consequently, the Boltzmann-BGK equation can be expressed as follows:

$$\frac{\partial f}{\partial t} + \boldsymbol{\xi} \cdot \frac{\partial f}{\partial \boldsymbol{x}} + \boldsymbol{a} \cdot \frac{\partial f}{\partial \boldsymbol{\xi}} = -\omega(f - f^{eq}) = -\frac{1}{\tau}(f - f^{eq}), \tag{2.2}$$

where $\omega = 1/\tau$ is the relaxation frequency and τ is the relaxation time. f^{eq} represents the local equilibrium function, defined by the Maxwell distribution function of macroscopic quantities

$$f^{eq} = \rho \frac{1}{(2\pi R_g T)^{D/2}} e^{-\frac{(\xi - u)^2}{2R_g T}}, \tag{2.3}$$

where ρ is the density, R_g is the gas constant, T is the temperature and D denotes the number of the spatial dimensions. The BGK approximation replaces the exact collision term with an approximate linear collision term, introducing some limitations. However, for small disturbances close to the equilibrium state, its accuracy is sufficient to meet the requirements.

The lattice Boltzmann equation is a special discrete form of the Boltzmann-BGK equation [47, 48]. By discretizing Eq. (2.2) in the velocity space into a finite number of discrete velocities $\boldsymbol{e}_i$ (i=0, 1, 2 $\cdots$ N) and expressing the left-hand side with discrete spatial intervals

Δx and discrete time steps Δt, we obtain the lattice Boltzmann equation

$$f_i(\boldsymbol{x}+\boldsymbol{e}_i\Delta t, t+\Delta t) = f_i(\boldsymbol{x},t) - \frac{1}{\tau}\left[f_i(\boldsymbol{x},t) - f_i^{eq}(\boldsymbol{x},t)\right] + G_i(\boldsymbol{x},t), \tag{2.4}$$

where $\Delta x = \boldsymbol{e}_i \Delta t$, G_i represents the discrete external force, and f_i^{eq} denotes the local equilibrium distribution function in the discrete velocity space. For a relatively low Mach number, f_i^{eq} can be expressed using the Taylor expansion as follows:

$$f_i^{eq} = \rho\omega_i\left[1 + \frac{\boldsymbol{e}_i\cdot\boldsymbol{u}}{c_s^2} + \frac{(\boldsymbol{e}_i\cdot\boldsymbol{u})^2}{2c_s^4} - \frac{\boldsymbol{u}^2}{2c_s^2}\right] + O(u^3), \tag{2.5}$$

where ω_i is the weight coefficient, $c_s = \sqrt{RT}$ is the lattice sound speed and $O(u^3)$ can be neglected.

2.1.2 Lattice Models

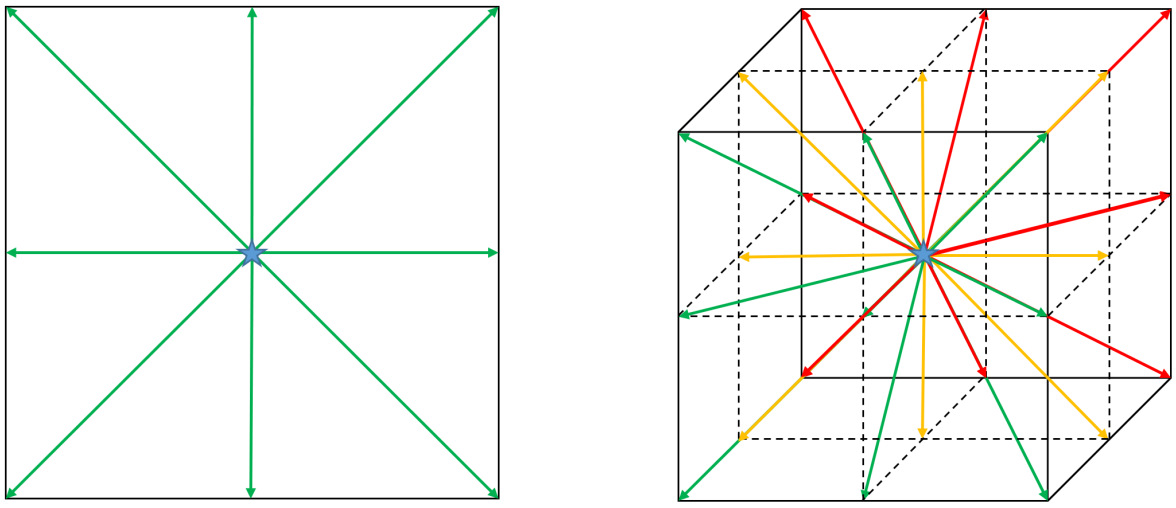
Lattice models are designated as D_dQ_m [49] , where d represents the number of spatial dimensions and m indicates the number of discrete velocities. These D_dQ_m models, first proposed by Qian et al. in 1992, form the foundation of the lattice Boltzmann method. Among the two-dimensional models, the $D2Q9$ model is the most commonly utilized, as illustrated in Fig. 2.1(a).

The $D2Q9$ model has the following velocity set

$$e_i = \begin{cases} (0,0), & i = 0 \\ (\pm c, 0), (0, \pm c), & i = 1,2,3,4 \\ (\pm c, \pm c), & i = 5,6,7,8 \end{cases}, \tag{2.6}$$

where $c = \Delta x/\Delta t = 1$ because $\Delta x = \Delta t = 1$ is the lattice unit in uniform mesh. The weight coefficients correspond to the discrete velocities, the sum of which is equal to 1

$$\omega_i = \begin{cases} 4/9, & i = 0 \\ 1/9, & i = 1,2,3,4 \\ 1/36, & i = 5,6,7,8 \end{cases}. \tag{2.7}$$



(a) D2Q9 (b) D3Q27

Figure 2.1: Lattice models in LBM.

For 3-dimensional models, $D3Q15$, $D3Q19$ and $D3Q27$ models are widely accepted. Due to the higher isotropy and accuracy, we adopted the $D3Q27$ model in this thesis. The detail of the $D3Q27$ model is shown in Fig. 2.1(b). The velocity set of the $D3Q27$ model is shown as follows:

$$e_i = \begin{cases} (0,0,0), & i = 0 \\ (\pm c,0,0),(0,\pm c,0),(0,0,\pm c), & 1 \le i \le 6 \\ (\pm c,\pm c,0),(\pm c,0,\pm c),(0,\pm c,\pm c), & 7 \le i \le 18 \\ (\pm c,\pm c,\pm c), & 19 \le i \le 26 \end{cases} . \tag{2.8}$$

The corresponding weight coefficients are

$$\omega_i = \begin{cases} 8/27, & i = 0 \\ 2/27, & 1 \le i \le 6 \\ 1/54, & 7 \le i \le 18 \\ 1/216, & 19 \le i \le 26 \end{cases} . \tag{2.9}$$

The macroscopic density and velocity are determined as follows

$$\rho = \sum^{i} f_i, \tag{2.10}$$

$$\boldsymbol{u} = \frac{1}{\rho}\sum^{i} f_i \boldsymbol{e}_i. \tag{2.11}$$

The macroscopic pressure can be given by

$$p = \rho c_s^2, \tag{2.12}$$

where $c_s = c/\sqrt{3}$ is the lattice sound speed.

By applying lattice models, the external force term in Eq. (2.4) can be calculated by

$$G_i = -3\omega_i \rho \frac{\boldsymbol{e}_i \cdot \boldsymbol{a}_i}{c^2}\Delta t, \tag{2.13}$$

where $\boldsymbol{a}_i$ is the acceleration generated by the external force.

Through the Chapman-Enskog analysis [23] for LBM, it can be demonstrated that the LBM can be recovered to the standard Navier-Stokes equations

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) = 0, \tag{2.14}$$

$$\frac{\partial (\rho \boldsymbol{u})}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}\boldsymbol{u}) = -\nabla p + \nabla \cdot \left[\rho\nu(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T) - \frac{\nu}{c_s^2}\nabla \cdot (\rho \boldsymbol{u}\boldsymbol{u}\boldsymbol{u})\right], \tag{2.15}$$

where ν is the kinematic viscosity, and it can be calculated by

$$\nu = c_s^2(\tau - \frac{1}{2})\Delta t. \tag{2.16}$$

The value of relaxation time τ is calculated from Eq. (2.16)

$$\tau = \frac{\nu}{c_s^2 \Delta t} + \frac{1}{2}. \tag{2.17}$$

The error term $\nabla \cdot (\rho \boldsymbol{u}\boldsymbol{u}\boldsymbol{u})$ in Eq. (2.15) is not included in the standard Navier-Stokes equations, however, for the low Mach number flow, this error term can be neglected.

After applying the lattice models above, the streaming and collision steps of LBM are

calculated using the following equations

$$f_i^*(\boldsymbol{x},t) = f_i(\boldsymbol{x},t) - \frac{1}{\tau}(f_i(\boldsymbol{x},t) - f_i^{eq}(\boldsymbol{x},t)) + G_i(\boldsymbol{x},t), \tag{2.18}$$

$$f_i(\boldsymbol{x} + \boldsymbol{e}_i\Delta t, t + \Delta t) = f_i^*(\boldsymbol{x},t), \tag{2.19}$$

where the asterisk $*$ denotes the post-collision state.

2.1.3 Collision Models

With the advancement of the lattice Boltzmann method, various collision models have been developed over time. The single relaxation time (SRT) collision model, governed by Eq. (2.4), assumes that all off-equilibrium velocity distribution functions relax simultaneously. This model is the simplest and requires the least computational effort among the various collision models. However, from a physical perspective, each distribution function should relax at different rates. When applied to high Reynolds number cases, the SRT collision model is prone to divergence. To enhance numerical stability in high Reynolds number simulations, the multiple relaxation time (MRT) [50], cascaded [51], and cumulant [44] collision models have been proposed. In this thesis, the cumulant collision model was adopted due to its superior stability. It is important to note that, compared to the SRT collision model, the cumulant model involves a significantly more complex algorithm.

According to the scale and flow velocity, floods including driftwood have very high Reynolds numbers and become unsteady turbulent flows. A large eddy simulation (LES) should be carried out to study the fluid dynamics of floodings. The cumulant collision model inherently introduces an implicit eddy viscosity due to its handling of higher-order moments and the dissipation mechanisms in the cumulant space. This implicit eddy viscosity acts to dissipate energy at smaller scales, which is similar to what explicit subgrid-scale models do in traditional LES. And this makes it possible to conduct LES without requiring explicit subgrid-scale models [44, 52].

The cumulant model transforms the velocity distribution function into a statistical quantity known as the cumulant, which is then used to calculate the collision process. This

approach increases the number of adjustable parameters, thereby enhancing the stability of the calculations. Since the $D3Q27$ lattice model is employed in this thesis, the velocity distribution function f_i is replaced by f_{ijk} hereafter and $i, j, k \in (-1, 0, 1)$. Assuming the cumulant is denoted by C, the order of cumulant is given by $\alpha + \beta + \gamma$. The cumulant is defined as follows

$$C_{\alpha\beta\gamma} = c^{-\alpha-\beta-\gamma} \frac{\partial^\alpha \partial^\beta \partial^\gamma}{\partial \Xi^\alpha \partial \Upsilon^\beta \partial Z^\gamma} \ln(F(\Xi, \Upsilon, Z))\Big|_{\Xi=\Upsilon=Z=0}, \tag{2.20}$$

$$F(\Xi, \Upsilon, Z) = \mathcal{L}[f(\boldsymbol{\xi} - \boldsymbol{u})] = e^{-\boldsymbol{u}\cdot\boldsymbol{\Xi}} \int_{-\infty}^{\infty} f(\boldsymbol{\xi}) e^{-\boldsymbol{\Xi}\cdot\boldsymbol{\xi}} d\boldsymbol{\xi}, \tag{2.21}$$

where $\boldsymbol{\Xi} = (\Xi, \Upsilon, Z)$ is the wave number.

The collision term of the cumulant model is expressed as follows:

$$C^*_{\alpha\beta\gamma} = \omega_{\alpha\beta\gamma} C^{eq}_{\alpha\beta\gamma} + (1 - \omega_{\alpha\beta\gamma}) C_{\alpha\beta\gamma}, \tag{2.22}$$

where the asterisk $*$ denotes the post-collision state, $\omega_{\alpha\beta\gamma}$ is the individual relaxation frequency for each cumulant and $C^{eq}_{\alpha\beta\gamma}$ is the equilibrium for each cumulant.

In this study, for the off-diagonal second-order cumulants, we set $\omega_{\alpha\beta\gamma} = 1/\tau$, while for the other cumulants, we set $\omega_{\alpha\beta\gamma} = 1$ to achieve relaxation to the local equilibrium state in a single step. After completing the collision process, the cumulant C^* is transformed back into the velocity distribution function f. The collision process acting on higher-order moments functions similarly to numerical viscosity, thereby stabilizing the computation even in high Reynolds number cases.

2.1.4 Large Eddy Simulation

Although the cumulant collision model inherently introduces an implicit eddy viscosity to the sub-grid scale, as we mentioned in the previous sub-section, it is risky to only use the cumulant collision model to stabilize the calculation of violent free-surface flows such as river flooding. In order to make the calculation more stable by suppressing the dispersion of tiny droplets, a coherent structure Smagorinsky LES model [53] is employed to couple with

cumulant LBM. The effect of sub-grid scale eddies acts similarly to molecular viscosity, and the eddy viscosity v_{SGS} is

$$v_{SGS} = C\overline{\Delta}^2 \left|\overline{S}\right|, \tag{2.23}$$

$$\left|\overline{S}\right| = \sqrt{2\overline{S}_{ij}\overline{S}_{ji}}, \tag{2.24}$$

where C is the Smagorinsky coefficient, $\overline{\Delta}$ is the filter width and we set $\overline{\Delta} = \Delta x$ in this thesis. The variables with an overline represent the grid-scale components, which are determined by the macro variables computed in LBM. $\overline{S}_{ij}$ is the strain rate tensor and it is computed by

$$\overline{S}_{ij} = \frac{1}{2}\left(\frac{\partial \overline{u}_j}{\partial x_i} + \frac{\partial \overline{u}_i}{\partial x_j}\right). \tag{2.25}$$

Since we consider the effects of the sub-grid scale, the kinematic viscosity v^* for each grid point is calculated by

$$v^* = v_0 + v_{SGS}, \tag{2.26}$$

where v_0 is the molecular viscosity. Thus, the relaxation time τ of the LBM is computed by

$$\tau = \frac{v^*}{{c_s}^2 \Delta t} + \frac{1}{2}. \tag{2.27}$$

The Smagorinsky coefficient C is computed using the coherent structure function F_{CS} and F_{CS} is calculated from the second invariant Q of the velocity gradient tensor and the magnitude E of the velocity gradient tensor.

$$C = C_{CSM} \left|F_{CS}\right|^{\frac{3}{2}}, \tag{2.28}$$

$$F_{CS} = \frac{Q}{E}, \tag{2.29}$$

$$Q = -\frac{1}{2}\frac{\partial \overline{u}_j}{\partial x_i}\frac{\partial \overline{u}_i}{\partial x_j}, \tag{2.30}$$

$$E = \frac{1}{2}\frac{\partial \overline{u}_j}{\partial x_i}\frac{\partial \overline{u}_j}{\partial x_i}, \tag{2.31}$$

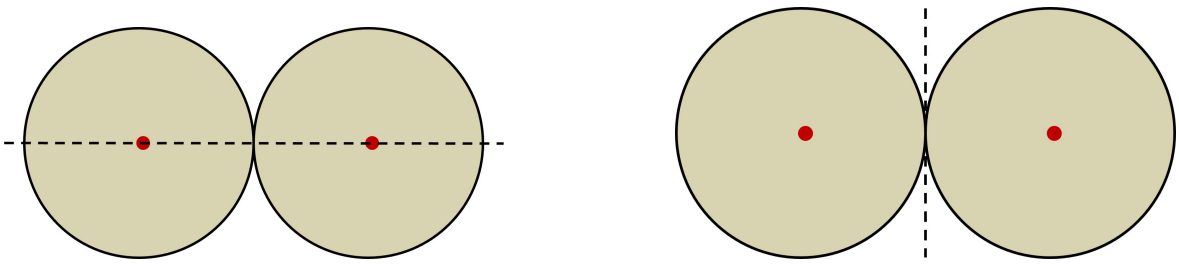
where C_{CSM} is the coefficient and we set $C_{CSM} = 0.1$.

In the commonly used Smagorinsky model, the model coefficient C is typically set as a constant for all the grid points. Still, this approach has a limitation in that it cannot accurately handle eddy viscosity near the wall boundary. However, the coherent structure Smagorinsky LES model applies the coefficient C at each grid point, so it can be used in turbulent flows with object boundaries without using any wall function.

2.2 Discrete Element Method

2.2.1 Calculation of Contact Force

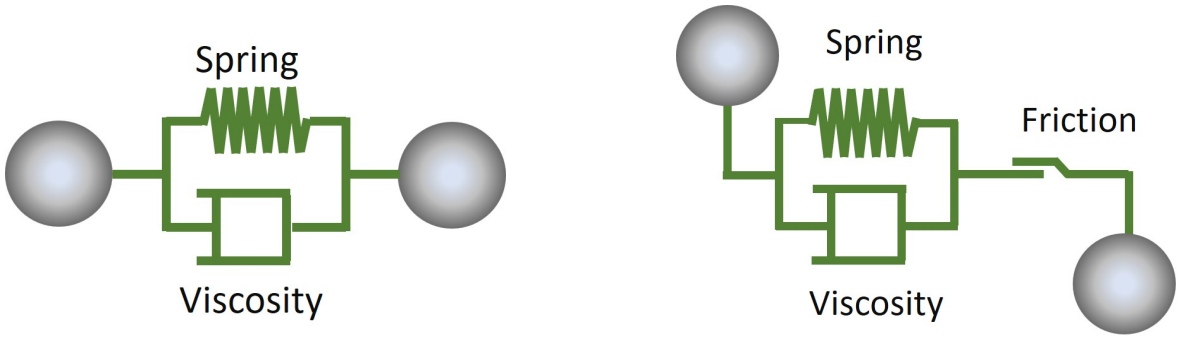
The Discrete Element Method (DEM) is a technique used to model the behavior of a group of particles by individually solving the motion of each particle.



(a) Normal Direction (b) Tangential Direction

Figure 2.2: Contact directions.

As illustrated in Fig. 2.2, the contact force is analyzed separately in the normal and tangential directions. The contact force between particles is modeled using springs, dashpots, and friction sliders. In the normal direction, as depicted in Fig. 2.3 (a), the repulsive force generated by the spring is proportional to the contact depth between the particles,



(a) Normal Direction (b) Tangential Direction

Figure 2.3: Contact force models.

while the damping force produced by the dashpot is proportional to the relative velocity of the particles. In the tangential direction, as shown in Fig. 2.3 (b), in addition to the aforementioned forces, the friction force arising from the relative velocity between particles is also taken into account.

First, we introduce the details of the calculation method for the normal direction. The particle is assumed to be i. The vectors of position, velocity, and angular velocity of particle i are set to $\boldsymbol{x}_i, \boldsymbol{v}_i, \boldsymbol{w}_i$ respectively. Suppose that particle j is in contact with particle i and the radius of particle i and particle j are r_i and r_j respectively. The contact depth d is calculated by

$$d = (r_i + r_j) - (\boldsymbol{x}_i - \boldsymbol{x}_j) \cdot \boldsymbol{n}, \tag{2.32}$$

where $\boldsymbol{n}$ is the unit normal vector between the two particles, which can be calculated by

$$\boldsymbol{n} = \frac{\boldsymbol{x}_i - \boldsymbol{x}_j}{|\boldsymbol{x}_i - \boldsymbol{x}_j|}. \tag{2.33}$$

Particle i and particle j are in contact with each other when $d > 0$. Thus, the relative velocity in the normal direction v_n can be computed by

$$v_n = -(\boldsymbol{v}_i - \boldsymbol{v}_j) \cdot \boldsymbol{n}. \tag{2.34}$$

After calculating Eq. (2.32) and Eq. (2.34), the contact force in the normal direction f_n

is calculated by

$$f_n = K_n d + C_n v_n, \tag{2.35}$$

where K_n is the coefficient of the spring in the normal direction and C_n is the coefficient of the dashpot in the normal direction. Therefore, the contact force vector $\boldsymbol{f}_n$ in the normal direction is

$$\boldsymbol{f}_n = f_n \boldsymbol{n}. \tag{2.36}$$

After introducing the calculation method for the normal direction, the details of the calculation method in the tangential direction are explained here, which are more complex than the former. Suppose the vector of relative velocity at the contact point is $\boldsymbol{v}_{ij}$. In the tangential direction, the rotation of particles must be considered, and thus $\boldsymbol{v}_{ij}$ is calculated by

$$\boldsymbol{v}_{ij} = \boldsymbol{v}_i - \boldsymbol{v}_j + r_i \boldsymbol{n} \times \boldsymbol{w}_i + r_j \boldsymbol{n} \times \boldsymbol{w}_j. \tag{2.37}$$

After solving Eq. (2.37), the vector of relative velocity in the tangential direction $\boldsymbol{v}_t$ is computed by

$$\boldsymbol{v}_t = \boldsymbol{v}_{ij} - \boldsymbol{n}(\boldsymbol{n} \cdot \boldsymbol{v}_{ij}). \tag{2.38}$$

After calculating $\boldsymbol{v}_t$, the vector of contact force in the tangential direction can be determined by

$$\boldsymbol{f}_t = -K_t \boldsymbol{\xi} - C_t \boldsymbol{v}_t, \tag{2.39}$$

where K_t is the spring coefficient in the tangential direction, C_t is the dashpot coefficient in the tangential direction, and $\boldsymbol{\xi}$ represents the spring compression vector in the tangential direction.. The spring compression vector $\boldsymbol{\xi}$ in the tangential direction can be calculated using the compression vector from the previous step, $\boldsymbol{\xi}^{'}$, and the relative velocity in the tangential direction $\boldsymbol{v}_t$ by

$$\boldsymbol{\xi} = \boldsymbol{\xi}^{'} + \boldsymbol{v}_t \Delta t, \tag{2.40}$$

where $\boldsymbol{\xi}^{'} = 0$ if the two particles were not in contact during the previous step. According to Coulomb's friction law, the actual value of $\boldsymbol{f}_t$ should be constrained by

$$\boldsymbol{f}_t = \begin{cases} \boldsymbol{f}_t & (|\boldsymbol{f}_t| < \mu\,|\boldsymbol{f}_n|) \\ \mu\frac{|\boldsymbol{f}_n|}{|\boldsymbol{f}_t|}\boldsymbol{f}_t & (|\boldsymbol{f}_t| \geq \mu\,|\boldsymbol{f}_n|) \end{cases}, \tag{2.41}$$

where μ is the friction coefficient.

After calculating $\boldsymbol{f}_n$ and $\boldsymbol{f}_t$, the force $\boldsymbol{F}_{ij}$ that particle i receives from particle j, and the moment $\boldsymbol{M}_{ij}$ can be computed by

$$\boldsymbol{F}_{ij} = \boldsymbol{f}_n + \boldsymbol{f}_t, \tag{2.42}$$

$$\boldsymbol{M}_{ij} = r_i \boldsymbol{n} \times \boldsymbol{f}_t. \tag{2.43}$$

When multiple particles are present in the calculation domain, a single particle will experience forces and moments from each particle it is in contact with. Therefore, the resultant force is equal to the sum of all the forces exerted on the particle by the contacting particles, and the resultant moment is determined similarly by summing all the individual moments.

$$\boldsymbol{F}_i = \sum_{i \neq j} \boldsymbol{F}_{ij}, \tag{2.44}$$

$$\boldsymbol{M}_i = \sum_{i \neq j} \boldsymbol{M}_{ij}. \tag{2.45}$$

For the other particles in the calculation domain, the resultant forces and resultant moments are calculated in the same manner. The velocities, positions, and angular velocities are then updated based on the obtained resultant forces and resultant moments using the following equations

$$\boldsymbol{v}_i^{n+1} = \boldsymbol{v}_i^n + \frac{\boldsymbol{F}_i + m_i \boldsymbol{g}}{m_i} \Delta t, \tag{2.46}$$

$$\boldsymbol{x}_i^{n+1} = \boldsymbol{x}_i^n + \boldsymbol{v}_i^n \Delta t, \tag{2.47}$$

$$\boldsymbol{w}_i^{n+1} = \boldsymbol{w}_i^n + \frac{\boldsymbol{M}_i}{I_i}\Delta t, \tag{2.48}$$

where I_i is the moment of inertia of particle i, and $\boldsymbol{g}$ is the gravitational acceleration vector. However, the update of the rotation angles is not fully addressed here. In the three-dimensional case, due to the gimbal lock problem associated with Euler angles [54], quaternions are used to represent the rotation angles, which will be introduced in Section 2.2.4. For the two-dimensional case, the rotation angles are computed by the following equation

$$\theta_i^{n+1} = \theta_i^n + \boldsymbol{w}_i^n \Delta t. \tag{2.49}$$

2.2.2 Parameter Setting

The spring coefficients K_n and K_t, as well as the dashpot coefficients C_n and C_t are calculated by

$$K_n = \frac{2E}{3(1-\nu^2)}\sqrt{\frac{r}{2}}, \tag{2.50}$$

$$K_t = \frac{K_n}{2(1+\nu)}, \tag{2.51}$$

$$C_n = 2\sqrt{\frac{mK_n}{1+(\pi/\ln e)^2}}, \tag{2.52}$$

$$C_t = 2\sqrt{\frac{mK_t}{1+(\pi/\ln e)^2}}, \tag{2.53}$$

where E is the Young's modulus of the solid particle, ν is the Poisson's ratio, r is the radius of the solid particle, m is the mass of the solid particle and e is the coefficient of restitution.

The time step Δt needs to be adjusted to ensure computational stability. The condition for Δt is given by

$$\Delta t < 2\pi\sqrt{\frac{m}{K_n}}. \tag{2.54}$$

However, in our simulations, the number of solid particles can reach several million, so the actual Δt used in the simulations is typically one-tenth of the value calculated above. When the actual Young's modulus is employed, the time step required for stable calculations becomes extremely small, leading to significantly increased total computation time and making the simulation very costly. To mitigate this, in many cases, Young's modulus is set lower than its actual value, and the time step is increased to the order of microseconds, provided that this adjustment does not significantly affect the overall particle dynamics.

2.2.3 Representation of Rigid Bodies Other than Spheres

The actual shape of a rigid body is rarely spherical; instead, it often exhibits a complex, non-spherical geometry. For instance, in the water tank simulation discussed in Chapter 4, the solid objects take the form of cylinders and cuboids. Furthermore, in the real disaster simulation of the O-tomo area, the actual terrain, with its intricate and irregular shape, must be accurately modeled using the Discrete Element Method (DEM). To address the challenge of handling non-spherical objects in DEM, two primary approaches have been proposed: one involves representing non-spherical shapes using polygons [55, 56], and the other involves approximating non-spherical shapes by rigidly connecting micro-spherical particles [57]. In this thesis, we adopt the latter method due to its ease of implementation, relatively low computational cost, and the ability to utilize the same algorithm as that used for spherical particles.

There are two methods for combining a group of micro-spherical particles into a non-spherical shape. The first method involves arranging the micro-spherical particles without overlapping. While this approach results in a smaller number of particles, it produces a non-spherical object with an uneven surface. The second method arranges the micro-spherical particles with overlapping to achieve a smoother surface. However, this method requires a larger number of particles and, consequently, incurs higher computational costs compared to the first method. In this thesis, as shown in Fig. 2.4, we adopt the second method to ensure a smooth surface for the non-spherical objects. We use the zero iso-surface of the level-set function to represent the shape of the driftwood, and we then place the spherical particles at the zero value of the level-set function and connect rigidly to their neighbor

particles. The sphere diameter is determined to describe the shape with a proper accuracy on the lattice and we set the diameter to be 1.5 times the finest lattice spacing of the fluid calculation. By calculating the resultant force on the micro-spherical particles that constitute the non-spherical object, the translation and rotation of the entire object can be determined. Additionally, only the particles located on the surface will experience contact forces from other particles, while the interior particles remain unaffected. To further reduce the number of particles and computational load, the interior of the non-spherical object is set to be hollow.

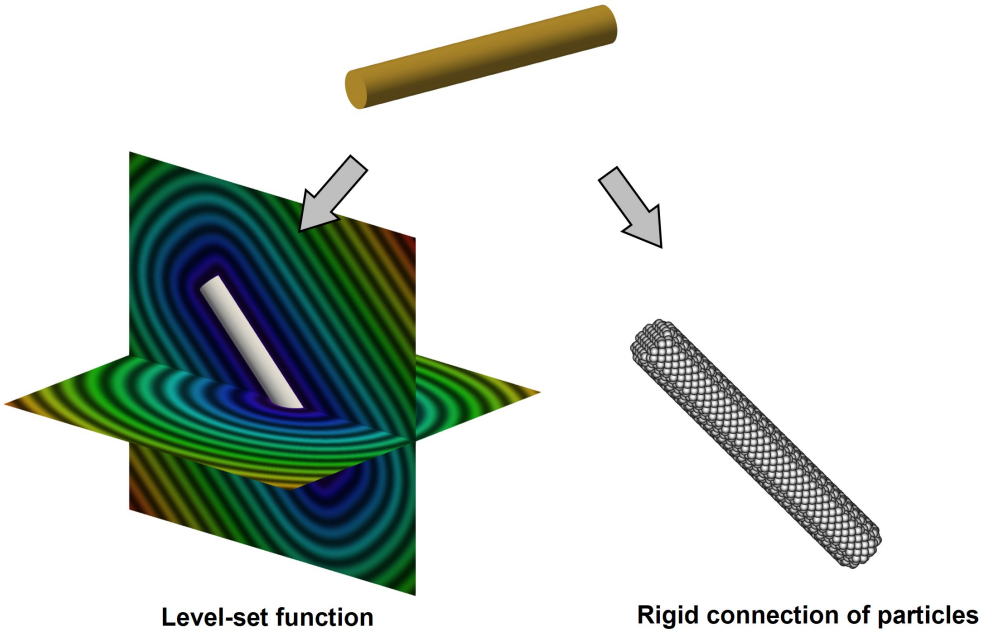


Figure 2.4: Object represented by small spherical particles with overlapping

2.2.4 Motion of Rigid Bodies Other than Spheres

The motion of a non-spherical object is computed by time-integrating the equations of motion for both the translation of the center of gravity and the rotation around the center of gravity. Let $\boldsymbol{F}_G$ represent the contact force acting on the non-spherical object as a whole, and $\boldsymbol{F}_i$ denote the contact force acting on each individual spherical particle that constitutes

the object.

$$\boldsymbol{F}_G = \sum^{n_G} \boldsymbol{F}_i, \tag{2.55}$$

where n_G is the number of spherical particles that constitute the object. According to Newton's second law, the following relationship can be established

$$m_G \frac{d\boldsymbol{v}_G}{dt} = \boldsymbol{F}_G + m_G \boldsymbol{g} + \boldsymbol{F}_f. \tag{2.56}$$

where m_G denotes the mass of the object, $\boldsymbol{v}_G$ is the translational velocity of the object, $\boldsymbol{g}$ is the acceleration due to gravity, and $\boldsymbol{F}_f$ denotes the force from the water pressure (described in the subsection 2.4.3).

By performing time integration of Eq. (2.56), the translational motion of the non-spherical object can be determined.

After computing the translational motion, the rotation of the object needs to be calculated.

$$\boldsymbol{M}_G = \sum^{n_G} [(\boldsymbol{x}_i - \boldsymbol{x}_G) \times \boldsymbol{F}_i)] + \boldsymbol{M}_f, \tag{2.57}$$

where $\boldsymbol{M}_G$ is the moment acting on the object, $\boldsymbol{x}_i$ is the position vector of a spherical particle, and $\boldsymbol{x}_G$ is the center of gravity of the non-spherical object, and $\boldsymbol{M}_f$ is the torque driven by the water (described in the subsection 2.4.3). Therefore, the angular momentum $\boldsymbol{L}_G$ can be calculated by

$$\frac{d\boldsymbol{L}_G}{dt} = \boldsymbol{M}_G. \tag{2.58}$$

After calculating $\boldsymbol{L}_G$, the angular velocity is calculated by

$$\boldsymbol{\omega}_G = \boldsymbol{I}_G(t)^{-1} \boldsymbol{L}_G, \tag{2.59}$$

where $\boldsymbol{I}_G(t)^{-1}$ is the inverse matrix of the moment of inertia tensor of the object at time t. Since the moment of inertia tensor varies with the rotation of the object, it must be recalculated at each time step. $\boldsymbol{I}_G(t)^{-1}$ can be calculated by

$$\boldsymbol{I}_G(t)^{-1} = \boldsymbol{R}(t)\boldsymbol{I}_G(0)^{-1}\boldsymbol{R}(t)^T, \tag{2.60}$$

where $\boldsymbol{R}$(t) is the rotation matrix that describes the orientation of the object from the initial time to time t. The rotation angle is updated using the angular velocity ω_G obtained from the previous calculations.

Euler angles are commonly used to express rotation in three-dimensional space. While they are intuitive and easy to understand, they are susceptible to the gimbal lock problem, which can compromise the accuracy of rotational calculations. To overcome this limitation, quaternions are introduced as an alternative representation for rotation angles.

A quaternion is a four-dimensional number consisting of one real part and three imaginary parts. SLet the components of the quaternion be q_s, q_t, q_u, q_v, with the imaginary units i, j, k. The quaternion $\boldsymbol{q}$ can be represented as

$$\boldsymbol{q} = q_s + q_t\mathrm{i} + \mathrm{q_u j} + \mathrm{q_v k}. \tag{2.61}$$

The quaternion $\boldsymbol{q}^*$, which is conjugate to the quaternion $\boldsymbol{q}$, is given by

$$\boldsymbol{q}^* = q_s - q_t\mathrm{i} - \mathrm{q_u j} - \mathrm{q_v k}. \tag{2.62}$$

Performing the rotation operation with quaternions requires calculating the product of quaternions. The imaginary units have the following multiplication properties

$$\mathrm{i}^2 = \mathrm{j}^2 = \mathrm{k}^2 = \mathrm{ijk} = -1, \tag{2.63}$$

$$\mathrm{ij} = -\mathrm{ji} = \mathrm{k}, \tag{2.64}$$

$$\mathrm{jk} = -\mathrm{kj} = \mathrm{i}, \tag{2.65}$$

$$\mathrm{ik} = -\mathrm{ki} = \mathrm{j}. \tag{2.66}$$

Using these properties, the product of quaternions $\boldsymbol{q}$ and $\boldsymbol{p}$ is represented by

$$\begin{aligned}\boldsymbol{qp} = &(q_s p_s - q_t p_t - q_u p_u - q_v p_v) \\ &+ (q_s p_t + q_t p_s + q_u p_v - q_v p_u)\mathrm{i} \\ &+ (q_s p_u - q_t p_v + q_u p_s + q_v p_t)\mathrm{j} \\ &+ (q_s p_v + q_t p_u - q_u p_t + q_v p_s)\mathrm{k}\end{aligned}. \tag{2.67}$$

For the sake of clarity, let the components of the imaginary part be represented by the vector $\boldsymbol{H}$ from this point onward. Consequently, the quaternion $\boldsymbol{q} = [q_s, \boldsymbol{H}]$.

From Eq. (2.59), the angular velocity ω_G can be determined, allowing the change in the quaternion $\Delta\boldsymbol{q}$ over the small time interval Δt to be expressed as

$$\Delta\boldsymbol{q} = \left[\cos\frac{\theta}{2}, \boldsymbol{x}\sin\frac{\theta}{2}\right], \tag{2.68}$$

where $\theta = |\omega_G \Delta t|$ is the rotation angle over the small time interval, and $\boldsymbol{x} = \omega_G/|\omega_G|$ is the unit vector representing the axis of rotation.

The time evolution of the quaternion is carried out by computing the quaternion product of $\Delta\boldsymbol{q}$ and the quaternion $\boldsymbol{q}$ at time t.

$$\boldsymbol{q}(t + \Delta t) = \Delta\boldsymbol{q}\,\boldsymbol{q}(t). \tag{2.69}$$

By calculating Eq. (2.69) at each time step, the orientation of the object can be determined. The rotation matrix $\boldsymbol{R}_t$ in the Eq. (2.60) can then be computed using the following expression

$$\boldsymbol{R}_t = \begin{pmatrix} 1 - 2q_u^2 - 2q_v^2 & 2q_t q_u - 2q_s q_v & 2q_t q_v + 2q_s q_u \\ 2q_t q_u + 2q_s q_v & 1 - 2q_t^2 - 2q_v^2 & 2q_u q_v - 2q_s q_t \\ 2q_t q_v - 2q_s q_u & 2q_u q_v + 2q_s q_t & 1 - 2q_t^2 - 2q_u^2 \end{pmatrix}. \tag{2.70}$$

Perform time integration of the translational and rotational motion equations for the center of gravity of the non-spherical object to update the object's motion after a time step. The position and velocity of the micro-spherical particles that make up the non-spherical object are then updated based on the translation and rotation of the object's center of

gravity. Let $\boldsymbol{r}_i(0)$ represent the relative position of micro-spherical particle i with respect to the center of gravity of the non-spherical object at the initial time. The position $\boldsymbol{r}_i(t)$ at time t is then calculated by

$$\boldsymbol{r}_i(t) = \boldsymbol{q}(t)\boldsymbol{r}_i(0)\boldsymbol{q}^*(t), \tag{2.71}$$

where $\boldsymbol{q}^*(t)$ is the conjugate quaternion of $\boldsymbol{q}(t)$, as defined in Eq. (2.62). Consequently, the position $\boldsymbol{x}_i$ of micro-spherical particle i can be computed by the following expression

$$\boldsymbol{x}_i = \boldsymbol{x}_G + \boldsymbol{r}_i, \tag{2.72}$$

where $\boldsymbol{x}_G$ is the position of the center of gravity of the non-spherical object. By utilizing the value of the velocity $\boldsymbol{v}_G$ and angular velocity $\boldsymbol{\omega_G}$ of the non-spherical object, the velocity of micro-spherical particle i can be calculated by

$$\boldsymbol{v}_i = \boldsymbol{v}_G + \boldsymbol{\omega}_G \times \boldsymbol{r}_i. \tag{2.73}$$

When forming a non-spherical object, the micro-spherical particles do not possess any rotational degrees of freedom. Therefore, the angular velocity of the micro-spherical particles is typically set to 0.

2.3 Interface Capturing Method

In gas-liquid two-phase flow, or its simplified version, free-surface flow, accurately capturing the gas-liquid interface is a critical challenge. Various interface capturing methods have been proposed to address this issue, including the volume of fluid (VOF) method [32], the level set method [35], and the conservative phase field method [33].

For wide-area flows such as tsunamis and river flows, the atmosphere does not enter as an important factor affecting water flow. Hence, we do not solve the fluid dynamics of the atmosphere. Instead, we apply the atmospheric pressure to the water surface and perform 3-D simulations of free-surface flow.

In the VOF method, the volume fraction of liquid within a cell is used to characterize

the cell's properties. Let the liquid volume fraction be denoted by χ, where $\chi = 0$ represents a gas cell, $\chi = 1$ represents a liquid cell, and $0 < \chi < 1$ represents an interface cell. To solve the VOF model numerically, several geometric approaches have been developed, such as the piecewise linear interface calculation (PLIC) method [58], the tangent hyperbola for interface capturing (THINC) method [34], and its extension, the THINC/WLIC method [59].

Although the VOF method is excellent in preserving volume, it can exhibit some non-physical phenomena during violent gas-liquid two-phase flows, as discussed in Section 1.1. Therefore, in this thesis, the conservative phase-field method coupled with the level-set method is adopted, and its details will be introduced in the following sections.

2.3.1 Interface Capturing by Phase Field Method

The phase field method captures the interface by defining an order parameter ϕ that implicitly represents the interface shape and solving its time evolution. This approach allows for the tracking of microstructure evolution in two or three dimensions [60]. As illustrated in Fig. 2.5, $\phi = 0$ represents the atmosphere/solid region, $\phi = 1$ represents the water region, and $0 < \phi < 1$ represents the interface region, which is conceptually similar to the volume fraction χ used in the VOF method.

Among various kinds of phase field models, models derived from the Cahn-Hilliard equation [61] and the Allen-Cahn equation [33] are the most widely accepted. The Cahn-Hilliard equation, which governs the evolution of the phase field function ϕ, is expressed as follows

$$\frac{\partial \phi}{\partial t} + \nabla \cdot (\phi \boldsymbol{u}) = M \nabla^2 (\phi^3 - \phi - \gamma \nabla^2 \phi), \tag{2.74}$$

where M is the mobility, and γ is a phenomenological parameter that scales with the interface width. Since the Cahn-Hilliard equation is a fourth-order, nonlinear partial differential equation, the calculation of the fourth-order term necessitates an extremely small time step Δt. To meet the stability condition when discretizing the Cahn-Hilliard equation using the Euler Forward method, Δt must satisfy the following condition, as described in [62]:

$$\Delta t < \frac{(\Delta x)^2}{4 + \frac{8\gamma}{(\Delta x)^2}}, \tag{2.75}$$

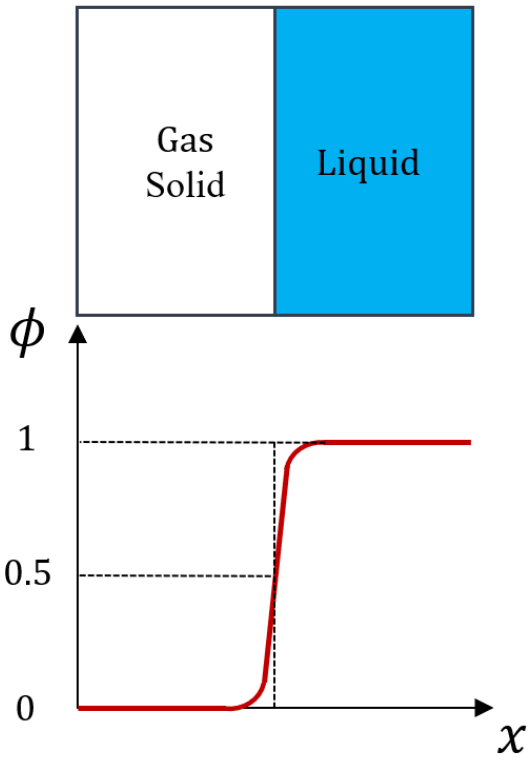


Figure 2.5: Free-surface represented by the phase field model

where Δx is the discretization grid step size.

Due to the aforementioned reasons, we solve the conservative Allen-Cahn equation for the time evolution of ϕ, as it can also accurately conserve mass in incompressible two-phase flows. The equation is given as follows

$$\frac{\partial \phi}{\partial t} + \nabla \cdot (\boldsymbol{u}\phi) = \nabla \cdot \left\{ M \left[\nabla \phi - \frac{1 - 4(\phi - \phi^{ave})^2}{W} \boldsymbol{n} \right] \right\}, \tag{2.76}$$

where M is the mobility, W is the thickness of interface, and $\boldsymbol{n}$ is the unit out-normal vector of the interface. In this study, we set $M = 0.05$ and $W = 3\Delta x$. ϕ^{ave} is the average value of ϕ for the liquid and gas phases, and we use $\phi^{ave} = 0.5$. The right-hand side of the Allen-Cahn equation consists of a diffusion term and an anti-diffusion term at the interface. The interplay between these terms allows the thickness of the interface to be maintained at a constant value.

Similar to the lattice Boltzmann method, Eq. (2.76) can be solved by introducing the

velocity distribution function h [63]. The relationship between the phase field value ϕ and the velocity distribution function h is expressed as follows

$$\phi(\boldsymbol{x},t)=\sum_{i=-1}^{1}\sum_{j=-1}^{1}\sum_{k=-1}^{1}h_{ijk}(\boldsymbol{x},t). \tag{2.77}$$

We also adopt the $D3Q27$ lattice model to solve the conservative Allen-Cahn equation. The time evolution equation of the velocity distribution function h is given as follows

$$h_{ijk}(\boldsymbol{x}+\boldsymbol{e}_{ijk}\Delta t,t+\Delta t)=\frac{1}{\tau}h_{ijk}^{eq}(\boldsymbol{x},t)+\left(1-\frac{1}{\tau_\phi}\right)h_{ijk}(\boldsymbol{x},t), \tag{2.78}$$

where τ_ϕ is the relaxation time, which can be calculated using the following expression

$$\tau_\phi=\frac{1}{2}+\frac{3M}{c^2\Delta t}. \tag{2.79}$$

The local equilibrium value of the discrete velocity distribution function h_{ijk}^{eq} is given by

$$h_{ijk}^{eq}=\phi\Gamma_{ijk}+\theta\omega_{ijk}\boldsymbol{e}_{ijk}\cdot\boldsymbol{n}, \tag{2.80}$$

$$\Gamma_{ijk}=\omega_{ijk}\left[1+\frac{\boldsymbol{e}_{ijk}\cdot\boldsymbol{u}}{c_s^2}+\frac{(\boldsymbol{e}_{ijk}\cdot\boldsymbol{u})^2}{2c_s^4}-\frac{|\boldsymbol{u}|^2}{2c_s^2}\right], \tag{2.81}$$

$$\theta=\frac{M}{c_s^2}\left[\frac{1-4(\phi-\phi^{ave})^2}{W}\right], \tag{2.82}$$

where $\boldsymbol{n}$ is the unit out-normal vector of the interface, the calculation method for which will be introduced in the next subsection.

2.3.2 Level Set Method and Its Coupling with Phase Field Method

Although the phase field method is effective for capturing the gas-liquid interface and maintaining mass conservation, the rapid changes in the contours of the phase field variables at the interface make it challenging to accurately calculate geometric information, such as the normal vector and curvature of the interface. To address this issue, the level set method [35] is introduced in this thesis to be coupled with the phase field method.

The level set method captures the interface by solving the time evolution equation of a signed distance function ψ. In this method, $\psi < 0$ represents the liquid phase, $\psi = 0$ denotes the interface, and $\psi > 0$ denotes the gas/solid phase. The time evolution equation of ψ is given as follows

$$\frac{\partial \psi}{\partial t} + \boldsymbol{u} \cdot \nabla \psi = 0. \tag{2.83}$$

The level set method allows for high-accuracy geometric calculations, but it suffers from poor conservation properties. By coupling the level set method with the phase field method, both accurate geometric calculations and relatively good conservation can be achieved. In the region near the interface, the level set function $\psi_0(\boldsymbol{x})$ can be derived from the phase field function $\phi(\boldsymbol{x})$ using the following relation

$$\psi_0(\boldsymbol{x}) = W\left(\frac{1}{2} - \phi\right). \tag{2.84}$$

After several steps of calculation, the level set function may become inconsistent with the phase field method, and the signed distance function may lose its property. In such cases, a re-initialization procedure [64] is necessary. The re-initialization equation is solved over several iterations (six iterations in this thesis) to correct $\psi_0(\boldsymbol{x})$

$$\frac{\partial \psi}{\partial t_\psi} + S(\psi)(|\nabla \psi| - 1) = 0, \tag{2.85}$$

where t_ψ is the time step used for iteration, in this thesis $t_\psi = 1/2\Delta x$ and $S(\psi)$ is a smoothed signed function, which is given by

$$S(\psi) = \frac{\psi}{\sqrt{\psi^2 + (|\nabla \psi|\, \Delta x)^2}}. \tag{2.86}$$

By calculating Eq. (2.84), (2.85), (2.86) at each time step of the fluid calculation, the level set function that is consistent with the phase field function is generated.

After obtaining the reinitialized value of the level set function, the normal vector in Eq. (2.80) can be calculated using the level set function as follows

$$\boldsymbol{n} = \frac{\nabla\psi}{|\nabla\psi|}, \tag{2.87}$$

where $|\nabla\psi| = 1$ is the property of the signed distance function, which simplifies the calculation of geometric information and enhances its accuracy.

The curvature of the interface can also be computed by the level set function

$$\begin{aligned}\kappa &= -\nabla \cdot \boldsymbol{n} \\ &= -\frac{\psi_x^2(\psi_{yy}+\psi_{zz}) + \psi_y^2(\psi_{xx}+\psi_{zz}) + \psi_z^2(\psi_{xx}+\psi_{yy}) - 2(\psi_x\psi_y\psi_{xy} + \psi_y\psi_z\psi_{yz} + \psi_x\psi_z\psi_{xz})}{|\nabla\psi|^3},\end{aligned} \tag{2.88}$$

where the subscript denotes the derivative, which is calculated using the second-order central difference method.

2.3.3 Velocity Extension Method

Under normal circumstances, solving the Allen-Cahn equation requires calculating the velocity fields for both the fluid and gas phases near the interface. However, in this study, the density of the liquid phase is much greater than that of the gas phase. To enhance the stability of the calculation, only the velocity field in the liquid phase where $\phi > 0.5$ is calculated. To address the need for velocity information near the interface, the velocity extension method [65] is employed. This method extrapolates the velocity field from the liquid phase into the gas phase, allowing the Allen-Cahn equation to be solved using this extended velocity field.

By iteratively calculating the following equation, the velocity $\boldsymbol{u}$ of the liquid phase is extrapolated into the gas phase

$$\frac{\partial \boldsymbol{u}}{\partial t_\alpha} = -S(\psi)\frac{\nabla\psi}{|\nabla\psi|} \cdot \boldsymbol{u}, \tag{2.89}$$

$$S(\psi) = \begin{cases} 0, & \psi < 0 \\ 1, & \psi \geq 0 \end{cases}. \tag{2.90}$$

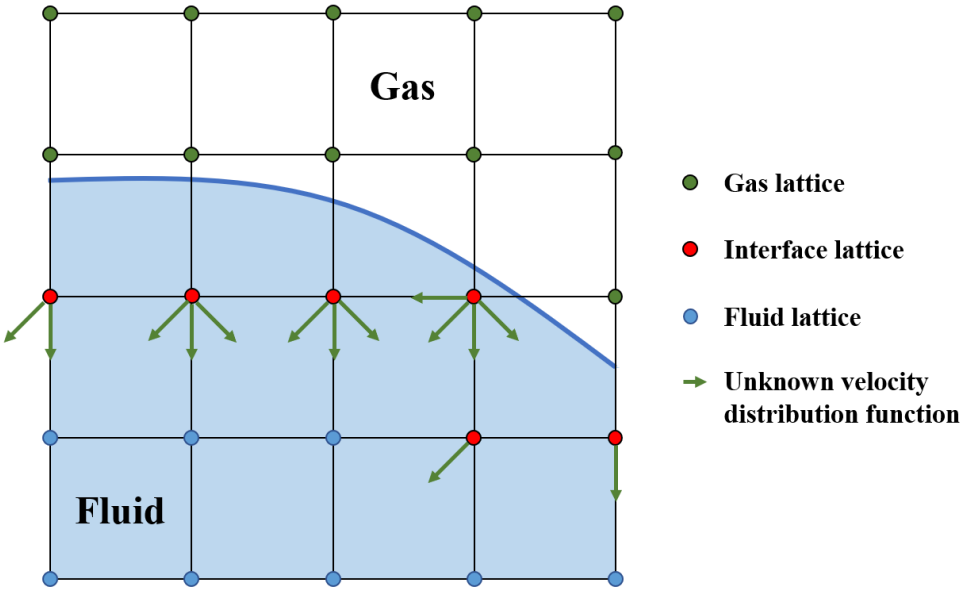


Figure 2.6: Boundary condition for free surface.

As a result, the extrapolated velocity u satisfies $\nabla u \cdot \boldsymbol{n} = 0$, meaning it becomes constant in the direction normal to the interface. The time step t_α for the velocity extension method is set to $1/2\Delta x$, which implies that with each time step, the velocity field is extrapolated by half a mesh into the gas phase. In this thesis, the number of iterations is set to 10, allowing the velocity to be extrapolated by 5 mesh cells from the interface into the gas phase.

2.4 Coupling Method

In this section, we introduce the methods that couple the lattice Boltzmann method, the discrete element method, and the interface capturing method.

2.4.1 Boundary Condition for Free Surface

As illustrated in Section 2.3.3, only the fluid phase where $\phi > 0.5$ is calculated in this research, while the gas phase is not directly calculated. This approach introduces a challenge. In Section 2.1.2, the streaming process of the lattice Boltzmann method (LBM) is described, where each lattice requires velocity distribution functions from adjacent lattices during the

streaming process. However, since only the liquid phase is calculated in this study, the velocity distribution functions for the gas lattices remain unknown, which poses a problem for interface lattices during the streaming process, as depicted in Fig. 2.6. To resolve this issue, the boundary condition for the free surface [31] is employed. The unknown velocity distribution functions that are streamed from the gas phase are calculated using the following approach

$$f_{ijk}(\boldsymbol{x}, t + \Delta t) = -f_{\overline{ijk}}(\boldsymbol{x}, t) + f^{eq}_{ijk}(\rho_f, \boldsymbol{u}) + f^{eq}_{\overline{ijk}}(\rho_f, \boldsymbol{u}), \tag{2.91}$$

where $\overline{ijk}$ denotes the opposite direction of the velocity distribution function, and ρ_f is the density at the free surface, which is calculated by the following equation

$$\rho_f = \rho_0 - 6\kappa\sigma, \tag{2.92}$$

where κ is the curvature of the free surface, σ is the surface tension of the fluid, and in this study, σ is set to $7.28 \times 10^{-2} N/m$.

2.4.2 Interpolated Bounce-back Scheme

To account for the coupling between the fluid and the solid object, the motion of the object must be incorporated into the LBM as a moving boundary condition. However, objects may have curved and inclined boundaries, making it challenging to describe these complex geometries accurately. To address this, the interpolated bounce-back scheme [66] is employed in this thesis. The interpolated bounce-back scheme is second-order accurate for arbitrary boundary shapes, and a conceptual diagram illustrating this approach is shown in Fig. 2.7. In what follows, $\boldsymbol{x}_b$ represents a boundary lattice point in the water and $\boldsymbol{x}_s$ denotes a solid lattice point in the solid, where $\boldsymbol{x}_b$ and $\boldsymbol{x}_s$ are neighboring lattices that are located on either side of the wall. $\boldsymbol{x}_f$ is the nearest fluid lattice beyond $\boldsymbol{x}_b$.

The interpolated bounce-back scheme is based on the assumption that the velocity distribution function toward the wall rebounds from the wall and returns to the original lattice point. If the distribution function moves into a solid node, the reflected distribution function is computed using the second-order interpolation scheme

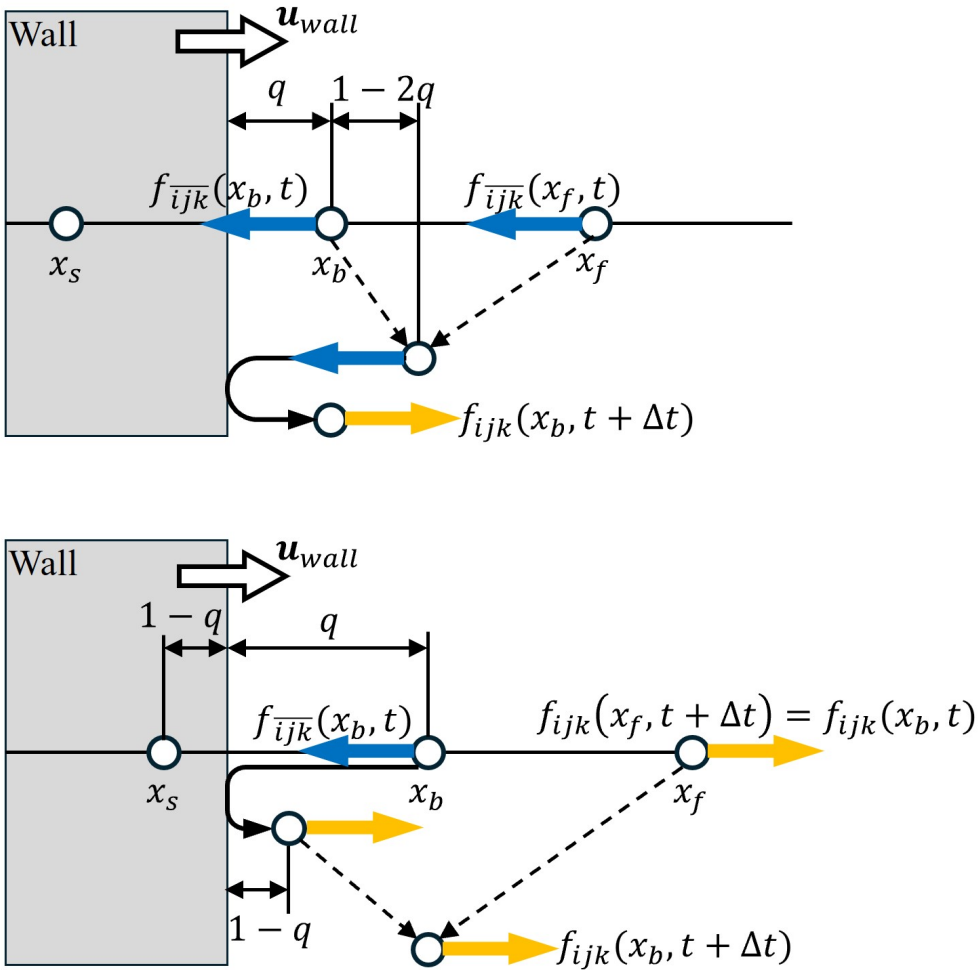


Figure 2.7: Interpolated bounce-back scheme(top: $0 < q \leq 0.5$, bottom: $0.5 \leq q \leq 1$).

$$f_{ijk}(\boldsymbol{x}_b, t+\Delta t) = \begin{cases} 2q f_{\overline{ijk}}(\boldsymbol{x}_b, t) + (1-2q) f_{\overline{ijk}}(\boldsymbol{x}_f, t) + \frac{6\omega_{ijk}\rho(\boldsymbol{e}_{ijk}\cdot\boldsymbol{u}_{\text{wall}})}{c^2}, & 0 < q \leq 0.5 \\ \frac{1}{2q}\left[f_{\overline{ijk}}(\boldsymbol{x}_b, t) + \frac{6\omega_{ijk}\rho(\boldsymbol{e}_{ijk}\cdot\boldsymbol{u}_{\text{wall}})}{c^2}\right] + \frac{2q-1}{2q} f_{ijk}(\boldsymbol{x}_b, t), & 0.5 \leq q \leq 1 \end{cases}, \tag{2.93}$$

where q is the ratio of the distance between the lattice points to the distance between the boundary lattice points and the wall, $\overline{ijk}$ denotes the direction of the distribution function towards the wall, ijk indicates the opposite direction of $\overline{ijk}$ and $\boldsymbol{u}_{\text{wall}}$ is the velocity of the moving wall. It should be noted that in the phase field method, the boundary condition for the velocity distribution function h is set to $q = 0.5$ in all cases to ensure conservation.

2.4.3 Hydrodynamic Force Acting on the Object

For calculating the hydrodynamic force acting on the object, the Galilean invariant momentum exchange method [67] is employed in this thesis. This method assumes that momentum exchange occurs between the wall and the fluid particles during the collision between the object and the particles. The force acting on the object is then determined by calculating the change in momentum before and after the particle collides with the object. Fig. 2.8 shows the mechanism of the momentum exchange method. The blue nodes denote fluid nodes, and the yellow dots show the locations where the momentum exchange happens. We apply the aforementioned interpolated bounce-back scheme at the yellow dots and the reflected distribution function $f_{ijk}(x, t + \Delta t)$ is calculated using Eq. (2.93).

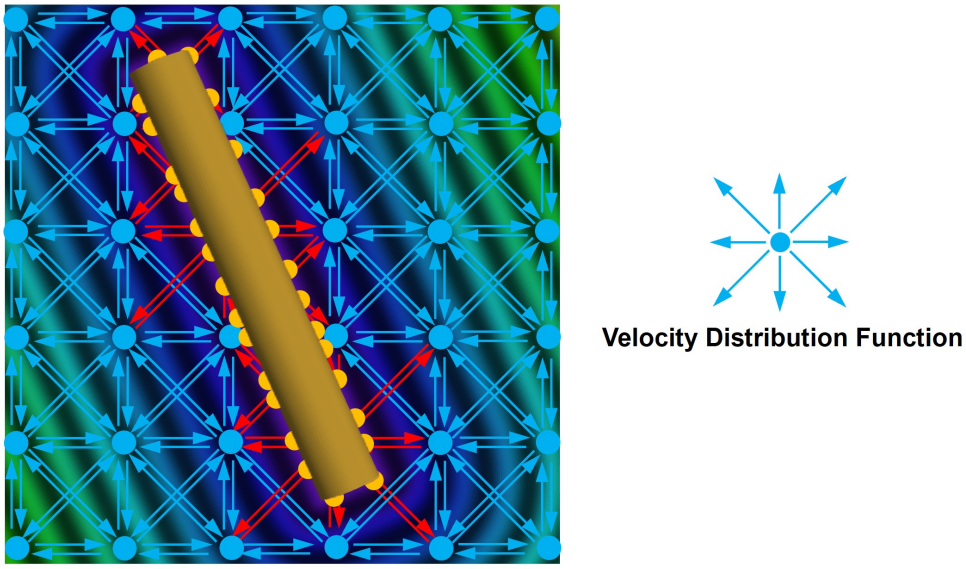


Figure 2.8: Mechanism of the momentum exchange method.

According to the theorem of momentum, the Galilean invariant momentum exchange method is defined by the following equation

$$\boldsymbol{F}_{ijk}(\boldsymbol{x}, t) = (\boldsymbol{e}_{\overline{ijk}} - \boldsymbol{u}_{\text{wall}}) f_{\overline{ijk}}(\boldsymbol{x}, t) - (\boldsymbol{e}_{ijk} - \boldsymbol{u}_{\text{wall}}) f_{ijk}(\boldsymbol{x}, t + \Delta t). \tag{2.94}$$

The total hydrodynamic force $\boldsymbol{F}_f$ and torque $\boldsymbol{T}_f$ acting on the object are evaluated by

$$\boldsymbol{F}_f = \sum_{x\in\Xi}\sum_{i=-1}^{1}\sum_{j=-1}^{1}\sum_{k=-1}^{1}\boldsymbol{F}_{ijk}(\boldsymbol{x},t), \tag{2.95}$$

$$\boldsymbol{T}_f = \sum_{x\in\Xi}\left[(\boldsymbol{x}-\boldsymbol{x}_G)\times\sum_{i=-1}^{1}\sum_{j=-1}^{1}\sum_{k=-1}^{1}\boldsymbol{F}_{ijk}(\boldsymbol{x},t)\right], \tag{2.96}$$

where Ξ denotes the wall area of the object, and $\boldsymbol{x}_G$ is the center of gravity of the object.

2.4.4 Refill Method for Wet-and-dry Cell Transition

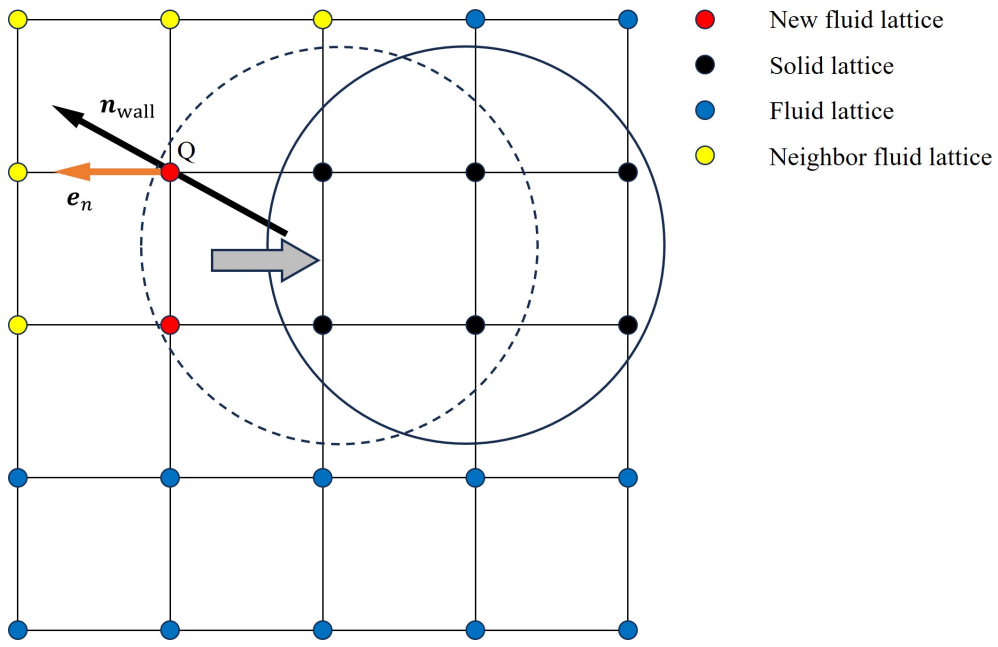


Figure 2.9: Refill of the velocity distribution function during wet-and-dry cell transition process.

When the solid object located at the dashed line circle in Fig. 2.9 moves to the position of the thick line, certain solid lattices within the object transform into fluid lattices, as indicated by the red dots in Fig. 2.9. In this study, the time evolution of the velocity distribution functions is calculated only at the lattices in the liquid phase, and velocity distribution functions are not defined at the lattices in the solid phase. Therefore, velocity distribution functions of the new fluid lattice [68] are determined using an equilibrium distribution plus a nonequilibrium correction by

$$
\begin{aligned}
f_{ijk}(\boldsymbol{x},t) &= f^{eq}_{ijk}(\boldsymbol{x};\boldsymbol{u}_{\text{wall}},\overline{\rho}) + f^{neq}_{ijk}(\boldsymbol{x}+\boldsymbol{e}_n\Delta t,t) \\
&= f^{eq}_{ijk}(\boldsymbol{x};\boldsymbol{u}_{\text{wall}},\overline{\rho}) + \left[f_{ijk}(\boldsymbol{x}+\boldsymbol{e}_n\Delta t,t) - f^{eq}_{ijk}(\boldsymbol{x}+\boldsymbol{e}_n\Delta t,t)\right]
\end{aligned}, \tag{2.97}
$$

where $\boldsymbol{u}_{\text{wall}}$ is the velocity of the solid object, $\overline{\rho}$ is the average density of neighbor fluid lattices (For lattice Q in Fig. 2.9, neighbor fluid lattice are yellow lattices), $\boldsymbol{e}_n$ is a specified discrete velocity along which direction the quantity $\boldsymbol{n}_{wall}\cdot\boldsymbol{e}_n$ takes the maximum value, $\boldsymbol{n}_{wall}$ is a unit normal vector of the object surface. The type of the new fluid lattice is changed from "SOLID" to "FLUID" which means the LBM calculations will be conducted for the new fluid lattices.

When a fluid lattice is transformed into a solid lattice, the lattice type is set to "SOLID" and LBM calculations are not conducted when the lattice type is "SOLID".

In addition to the transitions between solid and fluid lattices due to the movement of objects, gas lattices may also become liquid lattices as a result of the movement of the free surface. In such cases, the velocity distribution functions of the new liquid nodes that were previously gas nodes are computed by averaging the velocity distribution functions of the adjacent fluid nodes.

$$
f_{ijk}(\boldsymbol{x},t) = \frac{1}{N_f}\sum_{l=-1}^{1}\sum_{m=-1}^{1}\sum_{n=-1}^{1}\Omega_{lmn}(\boldsymbol{x},t)f_{ijk}(\boldsymbol{x}+\boldsymbol{e}_{lmn}\Delta t,t), \tag{2.98}
$$

where N_f is the number of the adjacent fluid nodes, Ω_{lmn} is a variable that identifies whether adjacent nodes are in the liquid phase, which is calculated by

$$
\Omega_{lmn}(\boldsymbol{x},t) = \begin{cases} 0, & \boldsymbol{x}+\boldsymbol{e}_{lmn}\Delta t \notin \text{liquid phase} \\ 1, & \boldsymbol{x}+\boldsymbol{e}_{lmn}\Delta t \in \text{liquid phase} \end{cases}. \tag{2.99}
$$

The type of the new fluid lattice is changed from "GAS" to "FLUID", which means the LBM calculations will be conducted for the new fluid lattices. When a fluid lattice is transformed into a gas lattice, the lattice type is set to "GAS" and LBM calculations are not conducted when the lattice type is "GAS".

2.4.5 Sub Time Step

In this thesis, the calculation of the lattice Boltzmann method needs to be coupled with the calculation of the discrete element method. However, a challenge arises due to the difference in time steps between the two methods. The time step for the lattice Boltzmann method, Δt_{LBM}, can be determined using Eq. (2.17).

$$\Delta t_{\mathrm{LBM}} = \frac{\nu}{c_s^2(\tau - \frac{1}{2})} = \frac{\Delta x^2}{3\nu}(\tau - \frac{1}{2}). \tag{2.100}$$

The time step condition for the discrete element method, Δt_{DEM}, is given in Eq. (2.54). Although, in many cases, Young's modulus is reduced from its actual value, allowing the time step to be increased to the order of microseconds without significantly affecting particle movement, Δt_{DEM} is still typically smaller than Δt_{LBM}.

There are two approaches to address the problem of differing time steps between the lattice Boltzmann method and the discrete element method. The first approach is to reduce Δt_{LBM} so that it matches Δt_{DEM}. However, since the computational load per time step for the lattice Boltzmann method is much greater than that for the discrete element method, this approach would significantly increase the overall computational cost. The second approach is to perform multiple time step calculations of the discrete element method within a single time step of the lattice Boltzmann method [69]. Therefore, the time step for the discrete element method, Δt_{DEM}, is set as follows to satisfy Eq. (2.54).

$$\Delta t_{\mathrm{DEM}} = \frac{1}{N}\Delta t_{\mathrm{LBM}}, \tag{2.101}$$

where N is the number of sub-time steps, and we set $N = 10$ in our approaches. For each time step of the lattice Boltzmann method, N time steps of the discrete element method are performed. During these N time steps, it is assumed that the force and torque exerted on the object by the fluid remain constant.

Chapter 3

Adaptive Mesh Refinement Method

In this chapter, the adaptive mesh refinement (AMR) method is introduced to improve computational efficiency. First, the development of the AMR method is explained. Next, the data structure and algorithms of the tree-typed block-structured AMR are introduced. Finally, the coupling of LBM and AMR is presented.

3.1 Development of AMR

The entire computational domain is divided into grids with uniform resolution when using orthogonal grids for calculations. However, for the free surface flow simulations in this thesis, only the interface areas require high resolution, causing problems that if low-resolution grids are assigned to the entire computational domain, calculations near the interface will become inaccurate. This will lead to some non-physical phenomena. On the other hand, if high-resolution grids are assigned to the whole computational domain, calculations of areas far away from the interface will cause a waste of computing resources, leading to difficulties in large-scale simulation.

A straightforward solution is to adjust the resolution according to the flow structure, which leads to the formation of the adaptive mesh refinement method. In 1984, Berger and Oliger [36] proposed the adaptive mesh refinement method for the first time. In their method, high-resolution grids are allocated to areas requiring high computational accuracy, while low-resolution grids are allocated to other areas. The method proposed by Berger and

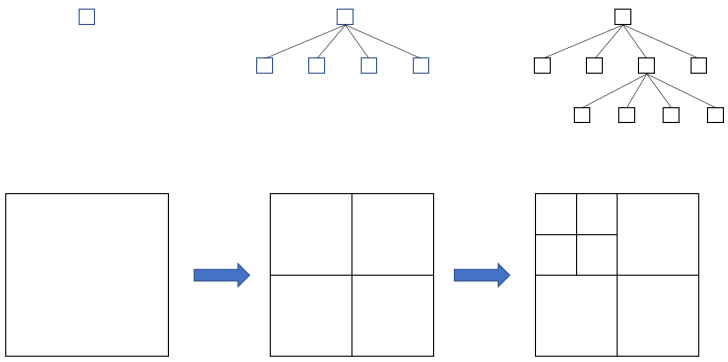


Figure 3.1: Recursive mesh generation based on tree-typed data structure.

Oliger, known as the patch-type method, involves arranging orthogonal grids with different resolutions at arbitrary positions.

With the advancement of the AMR method, the tree-typed [70] AMR method is developed. This method uses a tree-typed data structure to recursively divide the computational domain and changes the resolution according to the depth of the division. Compared with the patch-type approach, the tree-typed method allows for easier grid refinement. Additionally, due to the relatively simple arrangement rules of the computing units in the tree-typed block-structured AMR method [26], it is more suitable for GPU and parallel computing. Consequently, the tree-typed block-structured AMR method is adopted in this thesis.

3.2 Tree-typed Block-structured AMR

3.2.1 Tree-typed Data Structure

Fig. 3.1 shows the recursive mesh generation process using the AMR method with a tree-typed data structure. The node located at the top is referred to as the root node. It is the first to be created and represents the entire computational domain. The middle node connected by a solid line is called the parent node, which has a lower hierarchical level, and the lower node is called the children node, which has a higher hierarchical level. In two-dimensional computations, a quadtree data structure is employed, where each parent node has four children nodes, while in three-dimensional computations, an octree data structure

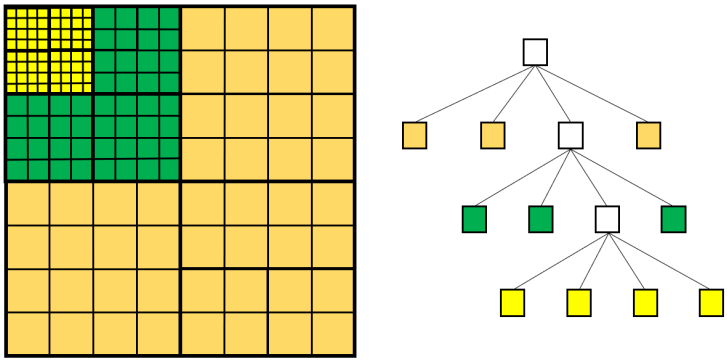


Figure 3.2: Block-structured meshes using quadtree.

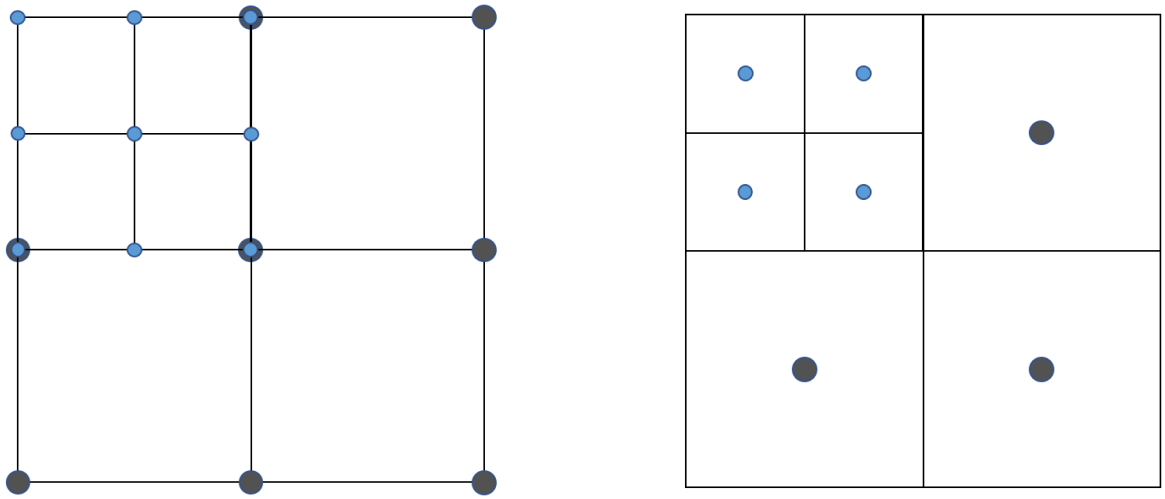
is used, with each parent node having eight children nodes. The computational domain is divided into four (2D) or eight (3D) equal regions, each assigned to a child node of the root node. These regions are then further recursively subdivided.

The decision of whether to divide a node into smaller children nodes depends on the physical characteristics of the region. In this thesis, the level set value is used to determine whether subdivision is necessary. To ensure computational stability, a 2:1 balance between neighboring blocks must be satisfied, which means the maximum allowable difference in level is one between adjacent blocks.

3.2.2 Block-structured Grid

Evenly spaced orthogonal grids are assigned to areas divided by the tree structure. As shown in Fig. 3.2, the leaves of the quadtree structure in the 2D domain contain 4×4 calculation cells. Leaves located farther from the root node are allocated higher-resolution grids. Each uniform grid assigned to a leaf is called a block. A smaller number of grids per block allows for more precise alignment with contours. Conversely, a larger number of grids per block results in fewer leaves and a more compact tree structure. Since the efficiency of the grid generation process depends on the size of the tree structure, this approach can significantly enhance computational performance.

There are two approaches for setting lattice points when integrating block-structured grids into the LBM. The first approach, as shown in Fig. 3.3(a), is to place lattice points at



(a) Node-centered lattice (b) Cell-centered lattice

Figure 3.3: Location of lattice points in block-structured meshes.

the cell vertices. This technique is known as the node-centered lattice. In this configuration, a block with 2×2 cells requires 3×3 lattice points. The advantage of using the node-centered lattice is that during mesh refinement or coarsening, the values within the current block can be used directly to interpolate the values of the new mesh. The second method is to position the lattice points at the cell centers, also known as the cell-centered lattice, as shown in Fig. 3.3(b). For a block with 2×2 cells, this approach uses 2×2 lattice points. The benefit of the cell-centered lattice is that compared with the node-centered lattice, it requires fewer lattice points for the same problem, thereby saving memory usage. However, during the refinement process, the values of the lattice adjacent to the refined mesh are needed, making this method more complex than the node-centered lattice. In this research, the cell-centered lattice is utilized to save memory usage.

3.3 Coupling of LBM and AMR

3.3.1 Multi-time Step Method

Multiple lattice resolutions within the computational domain are included when the LBM is coupled with the AMR. In that case, it is essential to maintain a constant speed of sound

throughout the whole domain.

$$c_s = \frac{\sqrt{3}}{3}\frac{\Delta x}{\Delta t}. \tag{3.1}$$

When subdividing the lattice based on the tree-typed AMR, the lattice width Δx^{d+1} at level $d+1$ is half of the lattice width Δx^d at level d.

$$\Delta x^{d+1} = \frac{1}{2}\Delta x^d. \tag{3.2}$$

To maintain a constant speed of sound, the time step interval for the fine grid at level $d+1$ is set as follows:

$$\Delta t^{d+1} = \frac{1}{2}\Delta t^d. \tag{3.3}$$

Thus, while the coarse grid advances by one time step, the finer grid advances by two time steps. This approach is known as the multi-time step method.

3.3.2 Interpolation of Velocity Distribution Function

In the calculation using the aforementioned multi-time step method, it is necessary to synchronize the boundaries between coarse and fine grids during the temporal overlap of these grids. In the synchronization process, as illustrated in Fig. 3.4, a coarse lattice is defined based on the fine lattices (blue points) near the boundary. The velocity distribution function f^c of the coarse lattice is then interpolated from velocity distribution functions f^f of the fine lattices using linear interpolation. The velocity distribution function $f^c_{I,J}$ of lattice (I, J) is calculated by

$$f^c_{I,J} = \frac{f^f_{i,j} + f^f_{i+1,j} + f^f_{i,j+1} + f^f_{i+1,j+1}}{4}. \tag{3.4}$$

In the multi-time-step approach, the fine grid performs two computational steps while the coarse grid advances by one step. Therefore, when interpolating the velocity distribution function f^f from the coarse grid to the fine grid, values are assigned to two grid points on the fine grid from the boundary of the resolution transition. For example, When interpolating

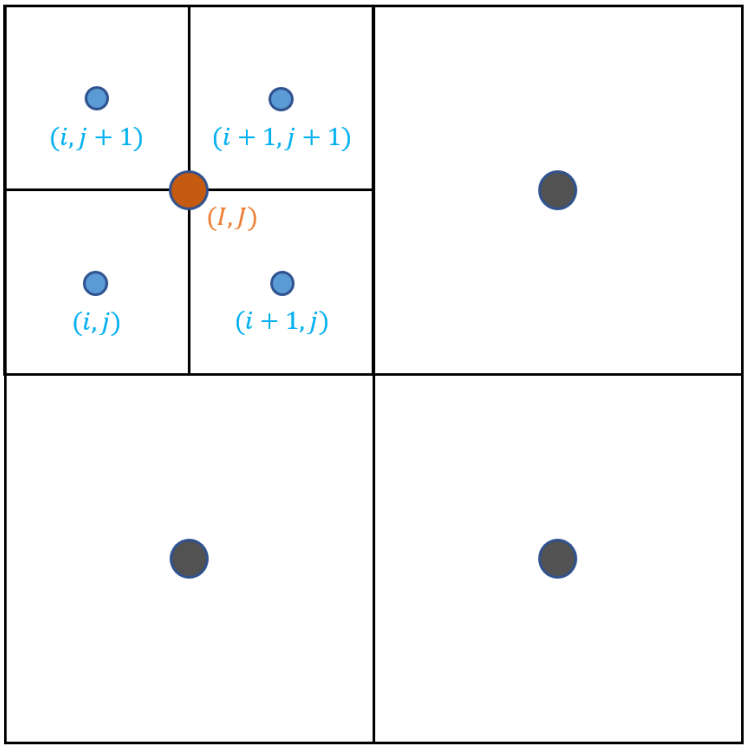


Figure 3.4: Interpolation from higher resolution lattices (blue) to a lower resolution lattice (brown) in two dimensional.

velocity distribution functions of the first fine grid point near the boundary, such as point (i, j) or $(i, j + 1)$ in Fig. 3.5, velocity distribution functions of the artificial coarse lattice point (brown in Fig. 3.5) as well as velocity distribution functions of actual coarse lattice points (black in Fig. 3.5) are utilized. The velocity distribution function $f^f_{i,j+1}$ for point $(i, j + 1)$ in Fig. 3.5 is calculated based on the non-dimensional distances α and β from the point (I, J) using linear interpolation.

$$f^f_{i,j+1} = (1-\alpha)(1-\beta)f^c_{I,J} + \alpha(1-\beta)f^c_{I+1,J} + (1-\alpha)\beta f^c_{I,J+1} + \alpha\beta f^c_{I+1,J+1}. \tag{3.5}$$

For the three dimensional calculation, the velocity distribution function $f^c_{I,J,K}$ of coarse lattice point (I, J, K) is calculated by

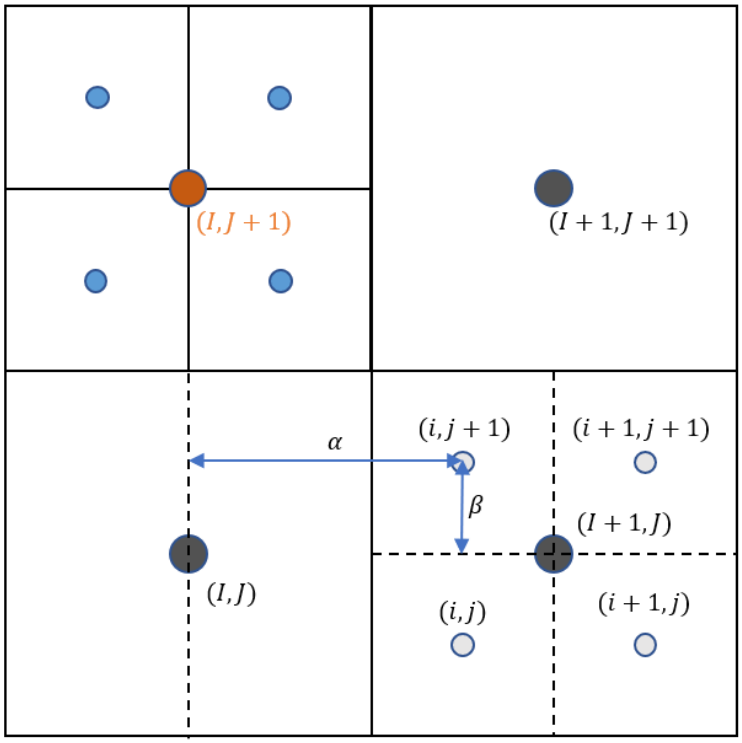


Figure 3.5: Interpolation from a lower resolution lattice (black) to a higher resolution lattice (grey) in two dimensional.

$$f^c_{I,J,k} = \frac{f^f_{i,j,k} + f^f_{i+1,j,k} + f^f_{i,j+1,k} + f^f_{i,j,k+1} + f^f_{i+1,j+1,k} + f^f_{i+1,j,k+1} + f^f_{i,j+1,k+1} + f^f_{i+1,j+1,k+1}}{8}. \tag{3.6}$$

The velocity distribution function $f^f_{i,j+1,k+1}$ for fine lattice point $(i, j+1, k+1)$ is calculated using the non-dimensional distances α, β and γ from the point (I, J, K).

$$\begin{aligned} f^f_{i,j+1,k+1} =& (1-\alpha)(1-\beta)(1-\gamma) f^c_{I,J,K} \\ &+ \alpha(1-\beta)(1-\gamma) f^c_{I+1,J,K} + (1-\alpha)\beta(1-\gamma) f^c_{I,J+1,K} + (1-\alpha)(1-\beta)\gamma f^c_{I,J,K+1} \\ &+ (1-\alpha)\beta\gamma f^c_{I,J+1,K+1} + \alpha(1-\beta)\gamma f^c_{I+1,J,K+1} + \alpha\beta(1-\gamma) f^c_{I+1,J+1,K} \\ &+ \alpha\beta\gamma f^c_{I+1,J+1,K+1}. \end{aligned} \tag{3.7}$$

Macroscopic fluid quantities, such as density and velocity, are derived from the distribu-

tion functions calculated by interpolation. Similarly, for the distribution function h used in the phase field method, values are computed using the same linear interpolation as for the velocity distribution function. The phase field variable ϕ is then determined by summing the distribution functions of the phase field method.

3.3.3 Dynamic Gird Coarsing and Refinement

As the computation progresses, the grid needs to be dynamically refined and coarsened in response to the movement of the interface. We use the value $\phi(1-\phi)$ computed from the phase field variable and the absolute value of the signed distance function $|s|$, which represents the distance between one fluid block and the nearest solid block as the refinement and coarsening criteria.

If the maximum value of $\phi(1-\phi)$ is larger than 0.09 or the minimum value of $|s|$ is smaller than $3\Delta x$, the block executes the refinement process and creates eight child blocks. On the other hand, if the eight child blocks do not satisfy the criteria, they are merged into a new parent block.

Chapter 4

3-D Simulation of Real River Flooding with Driftwood

In this chapter, with the assistance of Professor Shinsuke Takase from Hachinohe Institute of Technology, we conducted numerical simulations in a large water tank whose size is $0.6m \times 0.6m \times 7.8m$, as shown in Fig. 4.1, to validate our numerical simulation methods. We compared the simulation results with the experimental results and verified whether the calculation using the methods introduced above is appropriate.

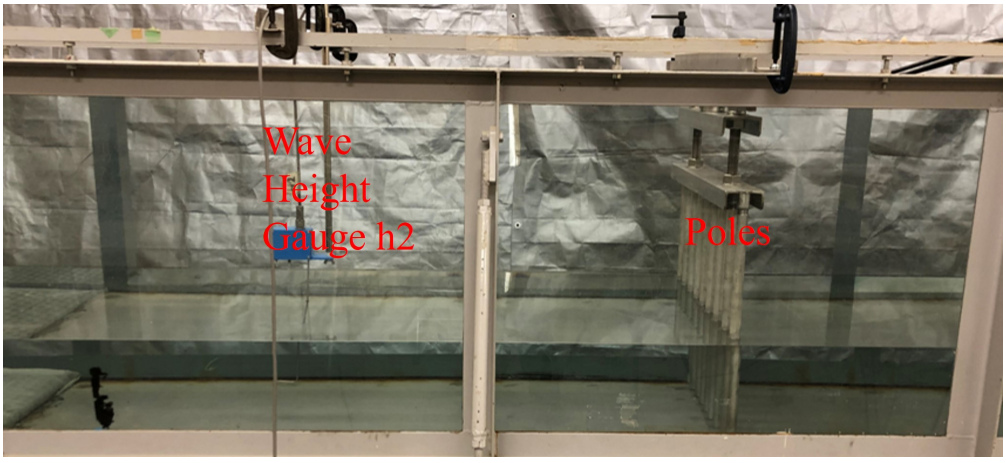


Figure 4.1: Side-view of water tank facility at Hachinohe Institute of Technology: picture of poles and h2 gauge.

Secondly, as an application, we used our method to simulate a real river-flood disaster

in Iwaizumi town in Iwate Prefecture in 2016 that was caused by Typhoon No. 10 with the assistance of Professor Shuji Moriguchi from Tohoku University. The flood on the Omoto River significantly damaged the area. Even more tragically, nine residents of an elderly care facility died. The computations generally reproduced the driftwood distribution and produced information with which to understand the process of driftwood accumulation.

4.1 Comparison with Laboratory Experiments

4.1.1 Water Flow Around Poles

First, we conducted an experiment on a water flow without driftwood. Fig. 4.2 shows an illustration of the whole experimental facility together with some sensor gauges when there is no driftwood included. The details of the poles are shown in Fig. 4.3. As a structure, we set nine poles 2.17 cm in diameter and 50 cm in vertical length at 5.5 m from the inflow area. There were 4 cm gaps between the poles and a 2 cm space from the bottom tank.

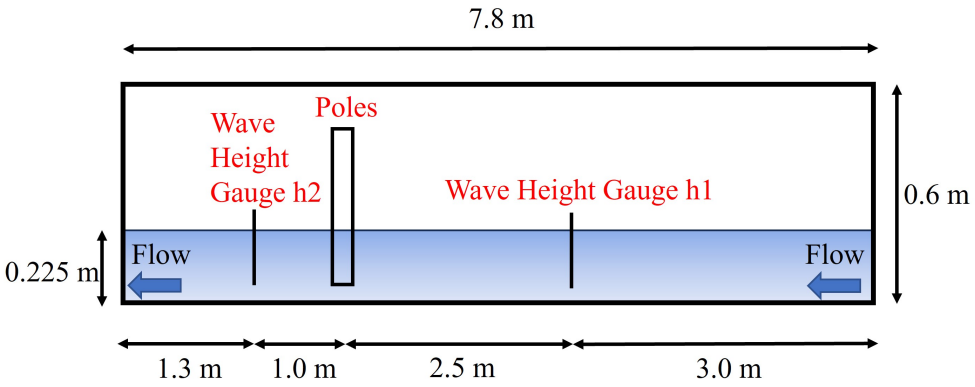


Figure 4.2: Simulation conditions when no driftwood is included.

To measure the change in water height before and after flowing through the poles, we set the first wave height gauge $h1$ at a position 2.5 m upstream from the poles and the second gauge $h2$ at 1.0 m behind the poles. A force gauge F was attached to each pole to measure the force acting on the poles due to the water pressure and the forces caused by collisions with the poles.

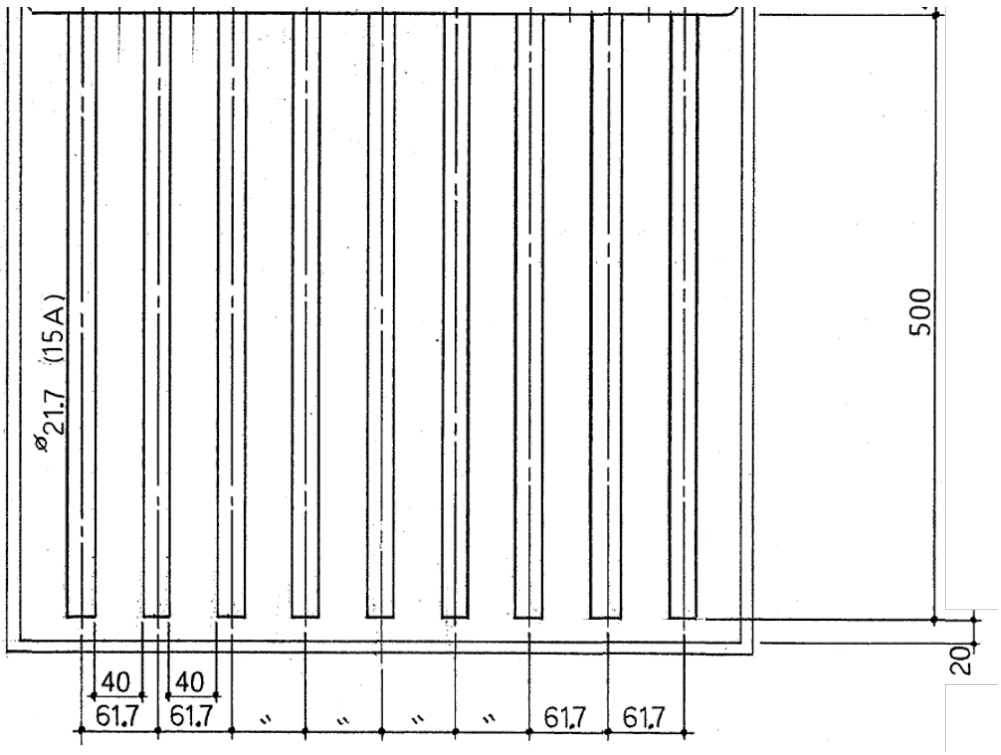


Figure 4.3: Details of poles.

The computational parameters were a water density of $1000\,kg/m^3$ and a kinematic viscosity of $1.004 \times 10^{-6}\,m^2/s$. In the experiment, we used a waterway with a circulation pump to make a steady flow after a long period of operation. In the simulation, we initially set a velocity of $0.6\,m/s$ measured in the experiment and a water height of $23.5\,cm$ to all the computational domains uniformly. We kept the constant velocity of $0.6\,m/s$ at the inflow boundary and applied the Neumann condition to the outflow boundary at the end of the computational domain.

We used the AMR method with five different levels for mesh refinement. The size of the coarsest mesh was $\Delta x_0 = 3.75\,cm$. The first refined mesh size was $\Delta x_1 = 0.5\Delta x_0 = 1.875\,cm$, the second $\Delta x_2 = 0.5\Delta x_1 = 9.375\,mm$, the third $\Delta x_3 = 0.5\Delta x_2 = 4.6875\,mm$. The finest mesh size was $\Delta x_4 = 0.5\Delta x_3 = 2.34375\,mm$. This setting was equivalent to $256 \times 3,328 \times 256$ grids in a uniform mesh. The total number of AMR grids was $75,682,020$, for a 65.3% saving in total memory and computational cost in comparison with a uniform mesh.

The calculation was performed on eight NVIDIA Tesla P100 GPUs on the TSUBAME3.0

supercomputer for 36 hours, resulting in a physical time of $30.0\,s$ with $850,000$ steps and Δt of $3.515625 \times 10^{-5}\,s$. Some specifications of the Tesla P100 GPU are shown in Table. 4.1.

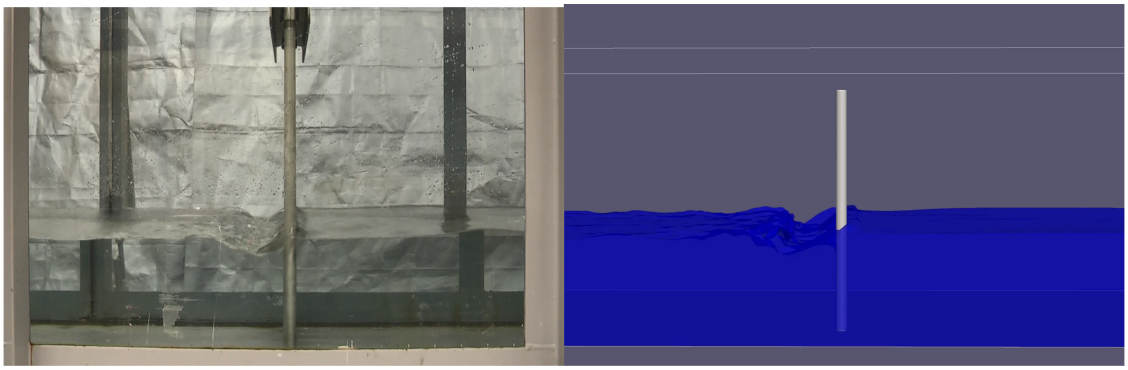
Table 4.1: Specifications of Tesla P100.

Nvidia Tesla P100	
Number of CUDA cores	3584
Peak single-precision floating point performance	9.3Tflops
Peak double-precision floating point performance	4.7Tflops
Memory size	16GB HBM2
Memory bandwidth	720 Gbytes/sec

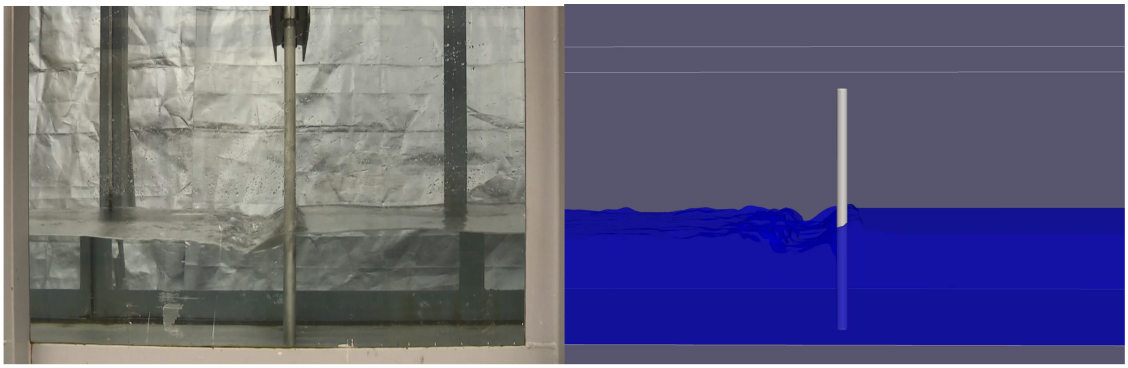
The time history of snapshots for the experimental and simulation results are shown in Fig. 4.4. By comparing the shape of the water surface after flowing through the pipes, it can be concluded that simulation results and experimental results are in good agreement. Fig. 4.5 shows a detailed side-view snapshot of the water flow including poles in the experiment when $t = 19.5\,s$. When the flow passed through the poles, the cross-sectional area became small and the flow velocity increased. After the poles, the water plunged and pushed down the water surface to make a shape like a waterfall basin. Fig. 4.6 is a snapshot of the simulation at 19.5 s, showing a similar shape to the water surface.

The history of the water heights measured at wave height gauges $h1$ and $h2$ is presented in Fig. 4.7. Since the Reynolds number of the flow was very high, the flow became turbulent immediately when it passed through the poles and driftwood was released into the flow. Before $8\,s$, the simulation result shows that the water heights at $h1$ and $h2$ gradually decrease because the Neumann boundary condition does not give the complete balance with the initial uniform flow. The gauge $h2$ has a shorter distance from the outflow boundary than $h1$, and the water height decrease at $h2$ starts faster than $h1$. Around $10\,s$ at $h1$ and $15\,s$ at $h2$, the flow reaches a steady state after interacting with the poles. Some fluctuations appear due to the poles' effect. The water level difference of $1.2\,cm$ between $h1$ and $h2$ is in good agreement with the experimental water height.

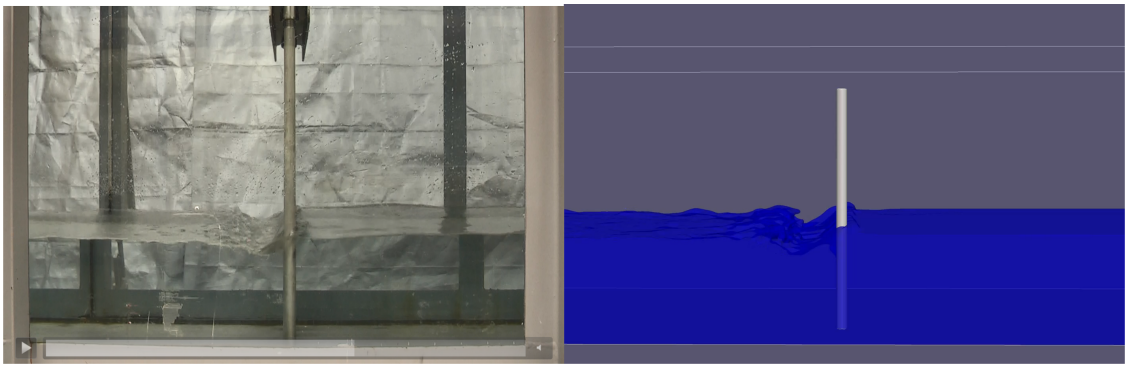
It is noted that the experimental data includes some fluctuations due to the noise of the water circulation pump. It is found that the simulation result shows more constant water



(a) t=5.0s



(b) t=10.0s



(c) t=15.0s

Figure 4.4: Time history of experimental and simulation snapshots when no driftwood is included.

Figure 4.5: Water surface around poles of the experiment without driftwood at 19.5 s.

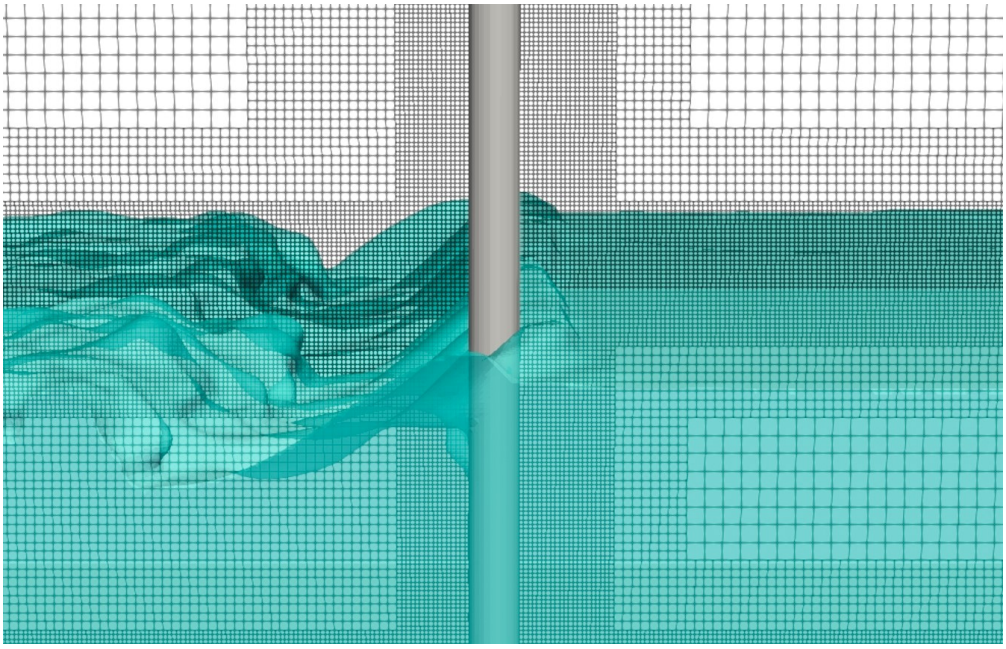


Figure 4.6: Water surface and AMR meshes around poles of the simulation without driftwood at 19.5 s.

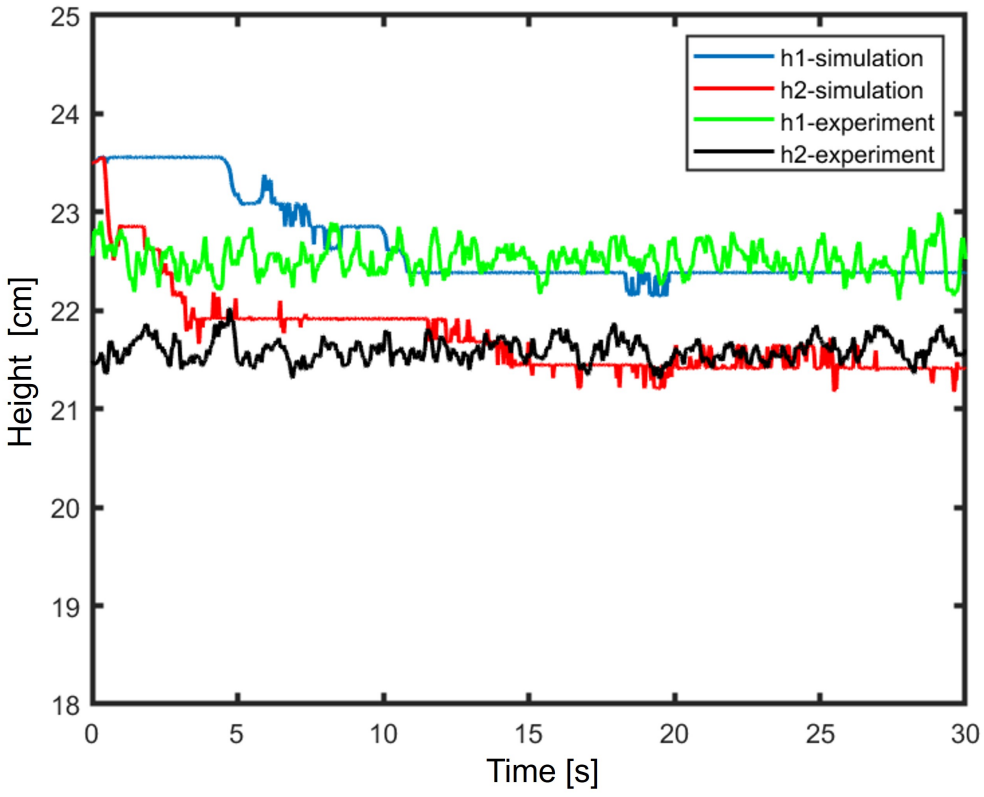


Figure 4.7: History of water heights at gauges $h1$ and $h2$ without launched driftwood.

heights than the experiment and the height at the $h1$ gauge is less affected by the poles than $h2$ because the $h1$ gauge is located at the upstream position.

4.1.2 Water Flow with 50 Pieces of Driftwood

Second, as a small-scale driftwood experiment, 50 pieces of chemical wood were launched onto a water flow under the same flow conditions as the above experiment. The driftwood model had a cuboid shape with a square bottom of $2\,cm \times 2\,cm$ and a length of $15\,cm$ and a density of $590\,kg/m^3$. The pieces entered the water at $6.0\,s$. Fig. 4.8 shows the launch position of 50 pieces of driftwood. Fig. 4.9 shows the top view of the initial arrangement of 50 pieces of driftwood.

The computational parameters were a water density of $1000\,kg/m^3$ and a kinematic viscosity of $1.004 \times 10^{-6}\,m^2/s$. The initial water level was set as $23.5\,cm$. Non-slip boundary conditions were applied to the side and bottom walls, the Dirichlet boundary condition was

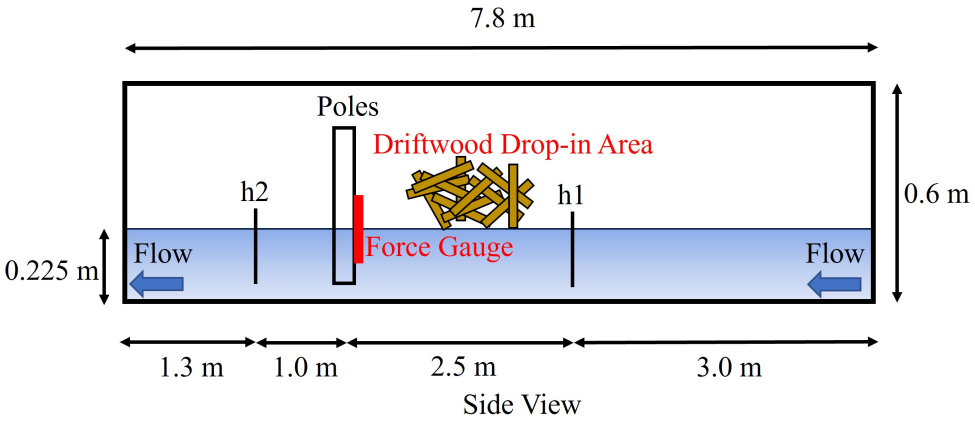


Figure 4.8: Launch position of 50 pieces of driftwood.

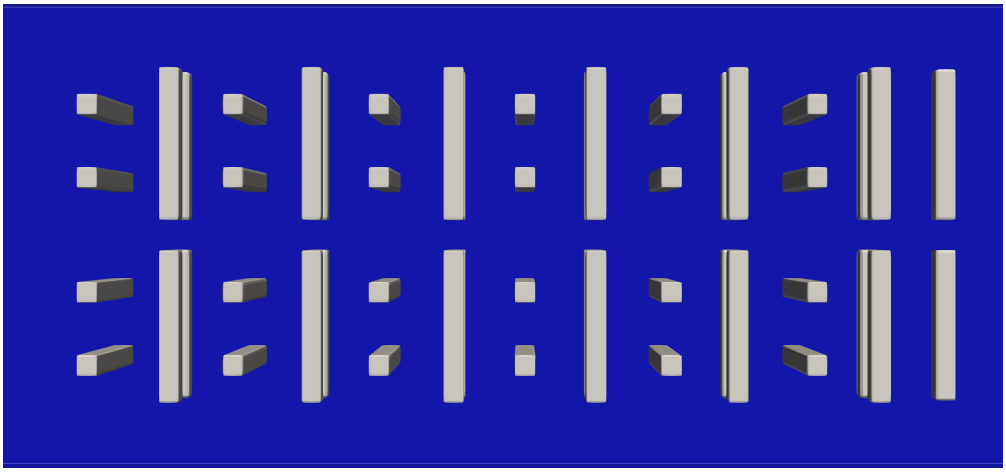


Figure 4.9: Initial arrangement of 50 pieces of driftwood (top view).

applied to the inflow boundary, and the inflow water had a velocity of $0.6\,m/s$, the Neumann boundary condition was adopted for the outflow boundary.

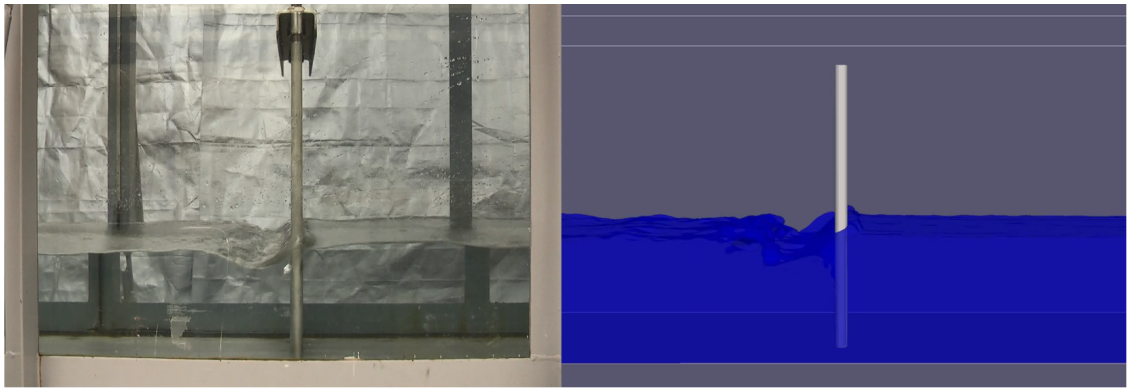
We assigned DEM spherical particles to pieces of driftwood and poles. 1,102 particles were assigned to one piece of driftwood and 3,474 particles were assigned to one pole. The total number of spherical particles in this simulation is 86,366. For the collisions between pieces of driftwood, DEM was applied with a Poisson's ratio of 0.33, a restitution coefficient of 0.25, and a friction coefficient of 0.3.

The calculation was performed up to a physical time of $14.8\,s$ on eight NVIDIA Tesla P100 GPUs of TSUBAME3.0 for 24 hours, amounting to $251,968$ steps in total.

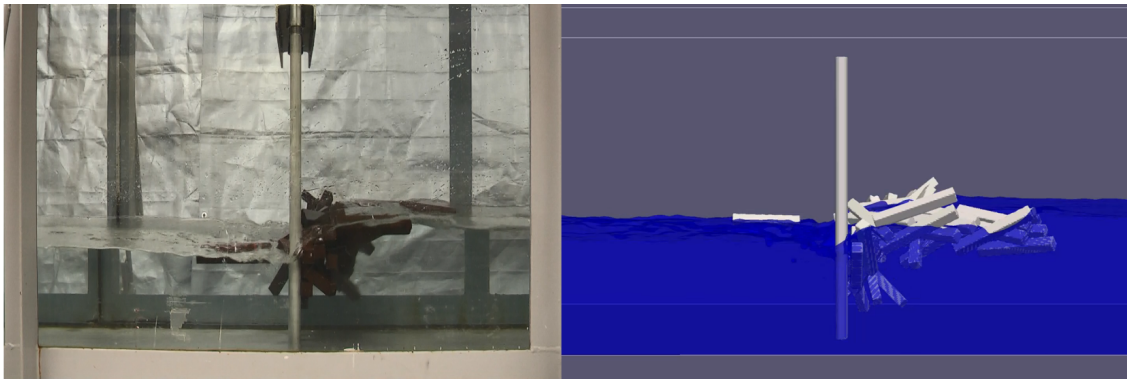
The time history of experimental and simulation snapshots is shown in Fig. 4.10. Fig. 4.11 and Fig. 4.12 respectively show snapshots at $14.8\,s$ of the water surface and the driftwood accumulation around poles in the experiment and simulation. By comparing the shapes of the water surface and positions of the driftwood blocked by the poles, it can be concluded that the simulation and experimental results are in good agreement. Some driftwood pieces sank in the water and overall, the driftwood accumulated into a triangular shape, viewed from the side. In addition, we observed an interesting phenomenon where one piece of driftwood became trapped in the depression area after the poles in both the experiment and simulation. This shows that a draw force occurs in the same manner as in a waterfall. The good agreement with the experiment indicated the high accuracy of the simulation result.

The history of water heights at $h1$ and $h2$ is shown in Fig. 4.13 for the case of 50 pieces of launched driftwood. The top piece of driftwood arrived at $8.0\,s$ and the driftwood gradually accumulated at the poles. The water height at $h1$ increased, whereas it decreased at $h2$. By comparing with Fig. 4.11, the driftwood accumulated at the poles in a way that the cross-sectional area which the water passed through becomes much smaller. The speed of the water flow increased after the poles and resulted in a lower water height. The simulation results of the water heights $h1$ and $h2$ were in good agreement with the experimental data.

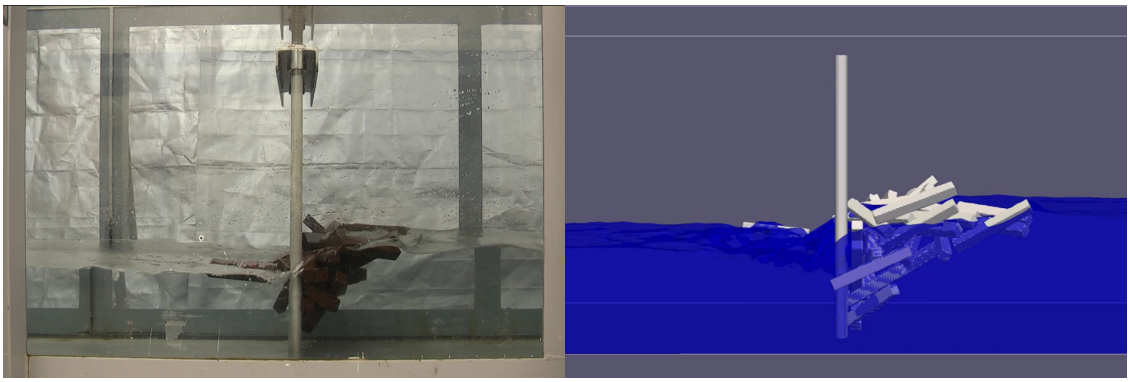
The force acting on the poles increased from $8.0\,s$ to $14.0\,s$, and the simulated force over the history was within 20% of the experimental values (Fig. 4.14). The deviations arose from the randomness of the driftwood launch. The force did not increase for $9-10\,s$ in the experiment and the simulation result shows a fall down at $9.5\,s$. After several top driftwood



(a) t=5.00s



(b) t=10.00s



(c) t=14.80s

Figure 4.10: Time history of experimental and simulation snapshots when 50 pieces of driftwood are included

Figure 4.11: Experimental snapshot of the water surface and driftwood accumulation around poles at 14.8 s.

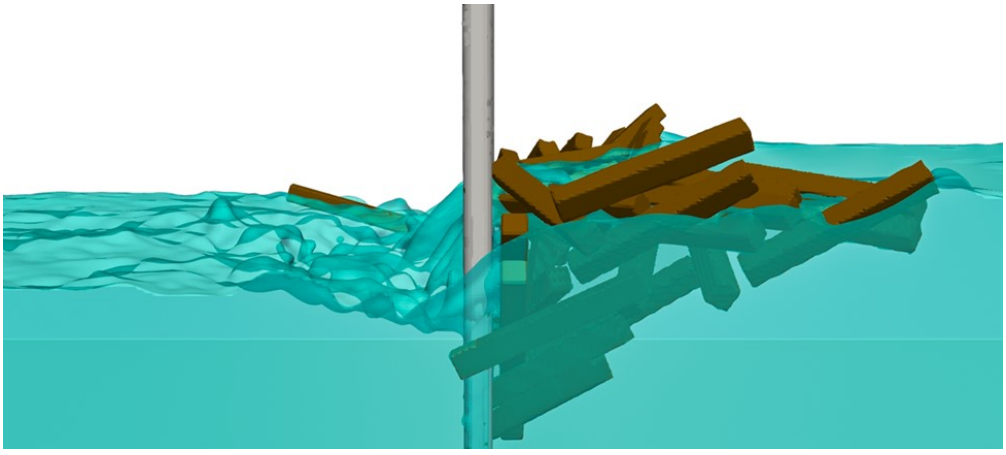


Figure 4.12: Simulation snapshot of the water surface and driftwood accumulation around poles at 14.8 s.

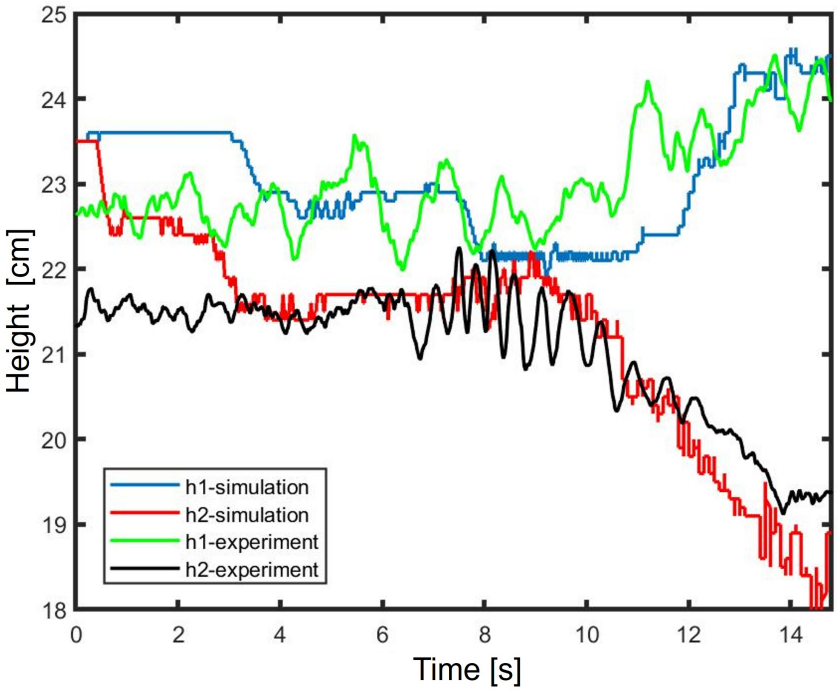


Figure 4.13: History of water heights at gauges $h1$ and $h2$ in the case of 50 pieces of launched driftwood.

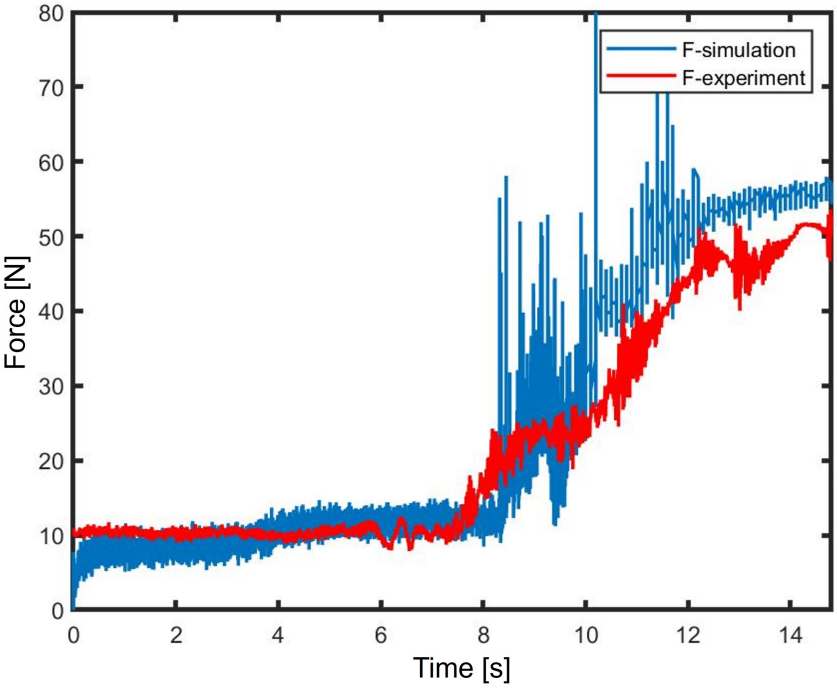


Figure 4.14: Total forces in simulation and experiment with 50 pieces of driftwood.

pieces became trapped at 8.0 s, the following driftwood flowed under the trapped driftwood.

Table 4.2: Number of driftwood pieces trapped by poles.

Experiment Results			
Experiment number	Total number	Trapped number	Trapping rate
1	50	44	88%
2	50	41	82%
3	50	48	96%
4	50	40	80%
5	50	40	80%
Average	50	43	85%
Simulation Results			
Simulation number	Total number	Trapped number	Trapping rate
1	50	48	96%
2	50	42	84%
3	50	45	90%
Average	50	45	90%

As shown in Fig. 4.11 and Fig. 4.12, in the experiments and simulations, some pieces of driftwood were trapped by the poles. On the other hand, some pieces of driftwood were carried through the pipes to the end of the water tank by the water flow. The number of trapped pieces is presented in Table 4.2. The experiments were conducted 5 times and the trapping ratios varied from 80% to 96%. These variations came from some uncertainties in the experiments, such as the drop angle of driftwood, and the timing of the drop because the driftwood was dropped manually within a specific area. The simulations were conducted 3 times and all pieces of driftwood were dropped at the same time when $t = 6.0\,s$ in simulation 1, pieces of driftwood were dropped randomly between 6.0 s and 6.5 s in simulation 2 and the driftwood was rotated with a random angle in simulation 3. The trapping rates were 96%, 84% and 90%. The average trapping rate was 43 out of 50 pieces in the experiment and 45 out of 50 in the simulation. Thus, we conclude that the simulation reproduced the flow phenomena including the free surface, structure, and driftwood, and that the simulation code is applicable to other flows including driftwood.

4.2 Real River Flooding with Driftwood

The ultimate goal in developing this method is to perform large-scale simulations for the real flooding disaster with driftwood, predict driftwood disasters at a relatively low cost and high efficiency, and thus provide some information with which to understand the driftwood accumulation process. In this section, we used our method to simulate a real river-flood disaster at Iwaizumi Town in Iwate Prefecture in 2016. The disaster was caused by Typhoon No. 10. The flood on the Omoto River significantly damaged the Iwaizumi Town. Fig. 4.15 (a) shows an aerial photograph and Fig. 4.15 (b) corresponding terrain data with a $2.5\,m$ resolution obtained by a drone, red circle 1 shows the location of the elderly care facility, where nine residents died during the river flood. After the disaster, it became apparent that a large number of logs had accumulated around the buildings.

We conducted two 3-D simulations to explore the process of inundation and driftwood accumulation through fluid dynamics. One was of a wide area; it had the goal of examining the flooding process affecting the river. The other was a high-resolution simulation of smaller areas around the buildings; it examined the process of driftwood accumulation.

4.2.1 Optimization for 3-D Large-scale Simulations

In order to conduct large-scale simulations, it was necessary to optimize the original code. The first step was to incorporate the level-set functions of the riverbed into the simulation domain. In the previous simulations, the level-set functions of solid objects were fully contained within the computational domain. However, when performing large-scale simulations involving the riverbed, part of the riverbed could extend beyond the domain boundaries if it was irrelevant to the simulation. To address this, we modified the index-setting process for the DEM particles accordingly.

The original index-setting function was defined as $dem.grid[i] = gridx[2] * nx[1] * nx[0] + gridx[1] * nx[0] + gridx[0]$. This code converted the particle position data into indices within a three-dimensional grid and stored them as one-dimensional indices. However, if any particles were positioned outside the calculation domain, their indices could not be correctly assigned, potentially causing the program to crash.

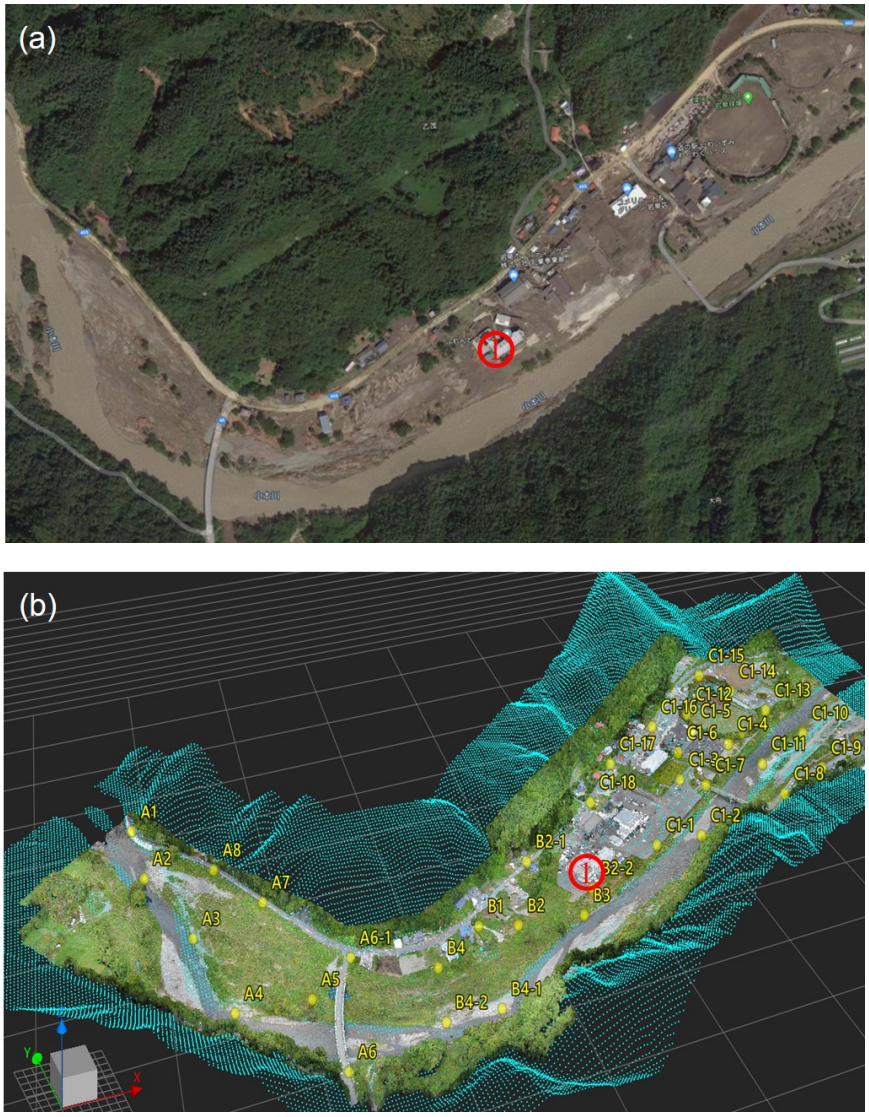


Figure 4.15: (a) Aerial photo and (b) terrain data of the Iwaizumi Town.

To solve this issue, we modified the index-setting function as $dem.grid[i] = MAX($ $MIN(gridx[2] * nx[1] * nx[0] + gridx[1] * nx[0] + gridx[0], nx[0] * nx[1] * nx[2] - 1), 0)$. The $MIN()$ function ensured that the computed indices did not exceed the maximum valid index, $nx[0]*nx[1]*nx[2]-1$, which represents the total number of grid cells minus one. Meanwhile, the $MAX()$ function prevented the indices from being less than 0, thereby avoiding negative index values. This new function constrained the particle index calculations, ensuring that the indices remained within valid bounds and preventing out-of-bounds memory access or computational errors.

After solving the index-setting problems, the next challenge was increasing the calculation speed. To accurately represent the shapes of riverbeds, terrain, and buildings, we used highly refined DEM particles. The number of DEM particles for the combined model of the riverbed, terrain, and buildings could reach several million. Fig. 4.16 shows the flowchart for coupling LBM and DEM calculations. It can be found that calculating fluid forces, torques and contact forces on particles of the combined model would be extremely time-consuming. However, these detailed force calculations were not needed for the combined model. Furthermore, during the initial stages of fluid flow, DEM calculations were unnecessary because all solid objects remained static.

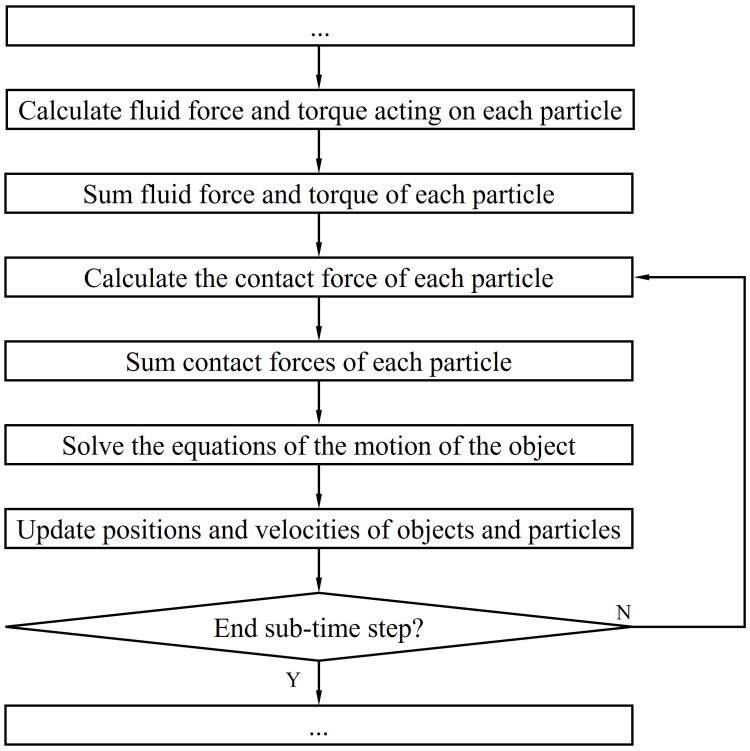


Figure 4.16: Flowchart of original coupling LBM and DEM calculations.

We optimized the flowchart to increase the calculation speed, as shown in Fig. 4.17. First, we determined whether DEM calculations were required. If they were required, we limited the calculations to the fluid force, torque, and contact force on the particles of driftwood only. This optimization reduced the calculation time for 100 time steps from 116 minutes to 23 minutes, demonstrating its effectiveness and suitability for large-scale computations.

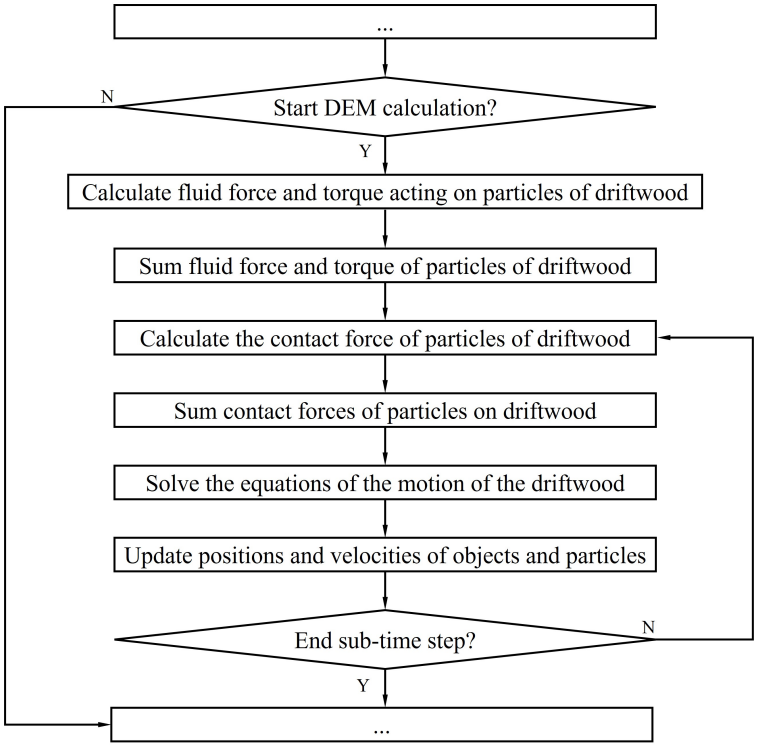


Figure 4.17: Flowchart of optimized coupling LBM and DEM calculations.

4.2.2 3-D Flooding Simulation of Omoto River

On the night of August 30, 2016, the water level of the Omoto River in the Iwaizumi Town of Iwate Prefecture increased dramatically due to Typhoon No. 10, flooding the elderly care facility, causing the death of nine elderly people as well as significant damage to the buildings due to the driftwood in the water.

To conduct a simulation of real river flooding with driftwood, the actual terrain was modeled first, as depicted in Fig. 4.18. This model was then converted using the conversion tool developed by Aoki Lab, which transforms STL files into level-set files. Consequently, the real terrain could be imported into the simulation program in the form of a level-set file. Importing models as level-set files offers two key advantages. First, when generating micro-spherical particles from the DEM, the particles can be positioned along the iso-surface where the level set value is zero. Second, when employing the AMR method, the level set value can be used to determine whether a specific region requires refinement.

After importing the model, we used the same code as in the previous section to conduct

Figure 4.18: STL model of Iwaizumi Town.

the simulation. The computational domain was $920\,m \times 280\,m$ and had a height of $40\,m$. The AMR mesh had five refinement levels: $\Delta x_0 = 2.5\,m$; $\Delta x_1 = 0.5\Delta x_0 = 1.25\,m$; $\Delta x_2 = 0.5\Delta x_1 = 62.5\,cm$; $\Delta x_3 = 0.5\Delta x_2 = 31.25\,cm$; and $\Delta x_4 = 15.625\,cm$. The total number of AMR meshes, at $567,237,883$, was equivalent to a $1,792 \times 5,888 \times 256$ uniform mesh, for a saving of 79.0% in total memory. The LBM computation needed a time step of $3.125 \times 10^{-4}\,s$ for the Δx_4 mesh. The calculation was executed on 36 NVIDIA Tesla V100 GPUs of the FLOW supercomputer. It took 200 hours of computational time to complete the simulation up to $450\,s$ on the real clock. Some specifications of the Tesla V100 GPU are shown in Table. 4.3.

Table 4.3: Specifications of Tesla V100.

Nvidia Tesla V100	
Number of CUDA cores	5120
Peak single-precision floating point performance	14.0Tflops
Peak double-precision floating point performance	7.0Tflops
Memory size	32GB HBM2
Memory bandwidth	900 Gbytes/sec

It should be noted that the number of micro-spherical particles in the model could exceed several million, making the DEM calculations for the bottom model highly time-consuming. However, in all simulations, the bottom model was a stationary rigid body and it was unnec-

essary to conduct the DEM calculations for the bottom model. To enhance computational efficiency, we chose not to perform DEM calculations on the bottom model. Nevertheless, the bottom model still needed to be converted into micro-spherical particles using DEM, as the interactions between pieces of driftwood and the bottom model required the calculation of collision forces and friction based on the DEM particles. This was essential for accurately evaluating the motion of the driftwood.

First, we had to ascertain the normal flow conditions of the Omoto River. The water was launched from the upstream boundary at the left-hand side into the computational domain, the whole area of which was initially dry. An open boundary condition was applied to the outflow at the right-hand side of the domain. As shown in Fig. 4.19, the whole water channel was almost inundated at 200 s and the red circle 1 also shows the location of the elderly care facility in the simulation model.

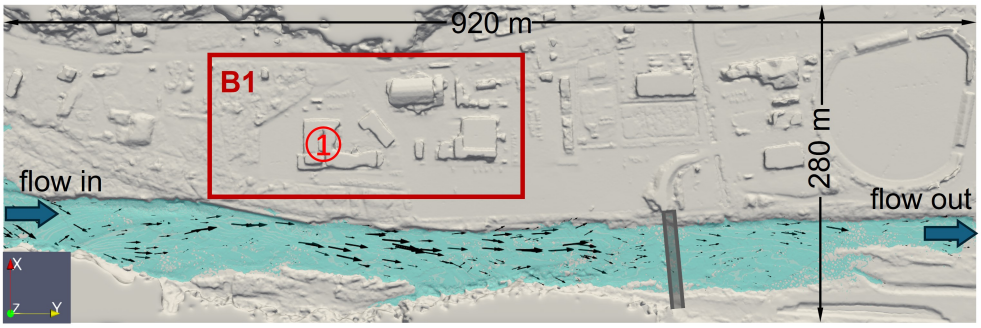


Figure 4.19: Snapshot of the simulated flow of the Omoto River when t=200 s.

Next, to compute the flooding, we widened the water inflow and increased the water height to 1.5 m and the velocity to 5.0 m/s, according to the investigation report [71].

Snapshots of the simulation result at 250.0 s, 350.0 s, and 450.0 s are shown in Fig. 4.20. Because of the widened water flow, inundation progressed from the upstream region in Fig. 4.20 (a). The water height gradually rose and overflowed the river bank. The overflow, in combination with the inundation from the left-hand side, accelerated the flooding (Fig. 4.20 (b)). The road bridge crossing the Omoto River is 5.3 m above the surrounding ground and it intercepted and separated the water flow. The separated flow inundated the inner dry area of the baseball stadium (Fig. 4.20 (c)). The simulation showed that almost the entire

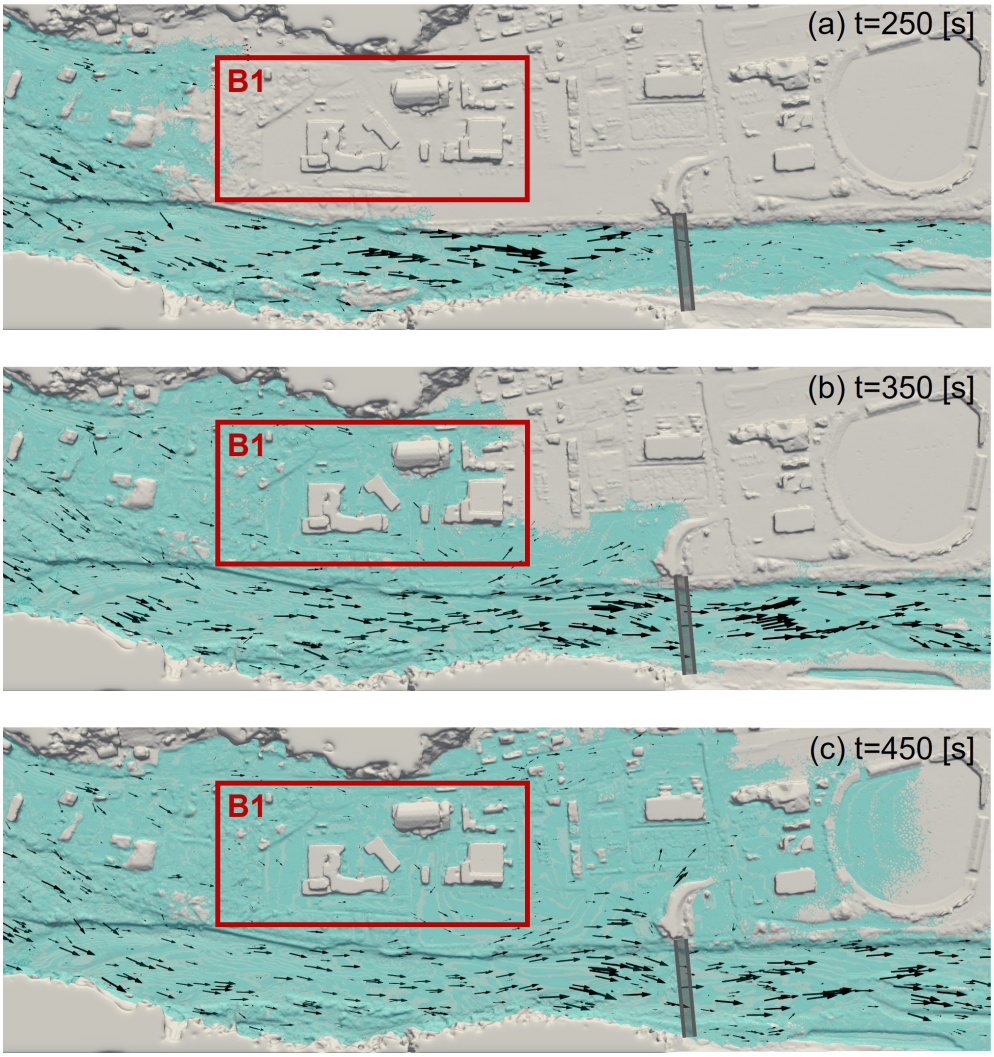


Figure 4.20: Snapshots of simulated flooding on the Omoto River.

Otomo area eventually became submerged. The overflow area was in good agreement with the investigation report [71].

In summary, the 3-D simulation reproduced the actual water flow of the flooding river; it could compute overflows of steep river banks and inundations of buildings.

4.2.3 Simulation of Driftwood in A Flooding River

We focused on the driftwood that accumulated around the elderly care facility: the computational domain was area **B1** of $360\,m \times 120\,m$ (Fig. 4.19 and Fig. 4.20). This domain had a height of $40\,m$. Since most driftwood is less than $0.5\,m$ in diameter, high-

resolution mesh had to be assigned around the simulated driftwood. The AMR mesh had six levels of refinement and the finest mesh size was $7.8125\,cm$ (corresponding to a domain with a uniform mesh of $1,536 \times 4,608 \times 512$). The x-axis was in the streamwise direction, the y-axis was normal to the x-axis, and the z-axis was in the vertical direction.

According to small-scale computations and the reference paper [38], we need to assign at least three meshes for the diameter of driftwood to compute the flow interaction with floating driftwood. We estimated the driftwood size from the on-site photo and we only focused on larger-diameter driftwood in the real situation. By applying the mesh limit and through the inspection of real driftwood pictures, two types of tree models were made (Fig. 4.21). Model **A** was $6.0\,m$ in length and $0.45\,m$ in diameter. It had four short branches $1.0\,m$ in length and $0.2\,m$ in diameter. Model **B** had a length of $4.0\,m$ and a diameter of $0.4\,m$ with four branches $0.5\,m$ in length and $0.2\,m$ in diameter. In total, 1000 pieces of driftwood were released ahead of the buildings, 500 of Model **A** and 500 of Model **B**. We also placed DEM spherical particles over the river bed and the interaction between pieces of driftwood and the river bed was evaluated using the same DEM collisions. When the water level is small, driftwood moves on the river bed. The friction force (shown in Eq. 2.41) in the DEM computation becomes dominant. We assigned 5,232,655 particles to the bottom floor and buildings, 814 particles to one piece of Model **A** and 527 particles to Model **B**. The total number of spherical particles in this simulation is 5,903,155.

The time step of LBM for the finest mesh $\Delta x = 7.8125\,cm$ was $\Delta t_{LBM} = 1.5625 \times 10^{-4}\,s$ and the time step of DEM was $\Delta t_{DEM} = 1/10 \Delta t_{LBM}$ [69]. The driftwood density was $850\,kg/m^3$, Poisson's ratio was 0.33, the restitution coefficient was 0.25, and the friction coefficient was 0.3. Non-slip boundary conditions were applied to the building walls, ground, and driftwood surfaces.

The water inundated the area from the left boundary. We derived the water levels from the reference paper [71], and assumed a linear decrease in water level from the peak value of $1.5\,m$ to $0\,m$. It also should be noted that the real disaster continued for several hours, however, we assumed that the water level changed in a short time of 350 [s]. The history of the water discharge and inflow water height is shown in Fig. 4.22. The initial inflow water height was set to $1.5\,m$ (inflow flux was $900\,m^3/s$) for $50.0\,s$. From $50.0\,s$ to $65.0\,s$, it was

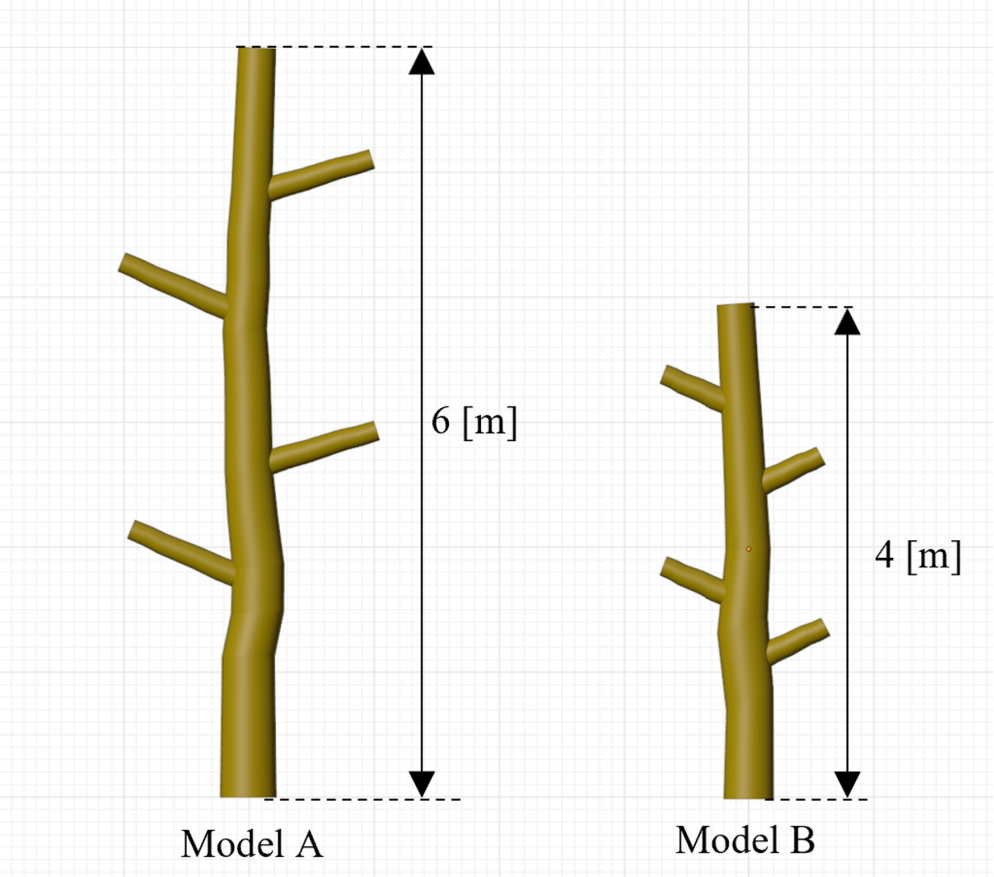


Figure 4.21: Two driftwood models.

decreased linearly to $0.6\,m$ (inflow flux was $360\,m^3/s$) and held at that value until $80.0\,s$. From $80.0\,s$ to $100.0\,s$, it was decreased linearly to $0.2\,m$ (inflow flux was $120\,m^3/s$) and held at that value until $115.0\,s$. Finally, the water height was decreased to $0\,m$ (inflow flux was $0\,m^3/s$) at $150.0\,s$, thereby ending the flooding in the upstream area.

Fig. 4.23 exhibits snapshots (top view) of the simulation at $t = 85\,s$, $175\,s$ and $350\,s$, where the water height is indicated by the color bar and the driftwood is in brown. Pieces of driftwood were discharged at the upstream area of the elderly care facility. We released 500 pieces of driftwood at $t = 50\,s$ and another 500 pieces at $t = 80\,s$, the release timing was estimated to ensure the driftwood was dropped when the inflow flux was relatively high. A comparison with an on-site photo corresponding to the computational domain (Fig. 4.24) and taken after the disaster indicates that although there are significant differences between the accumulation density and shape of the driftwood in the simulation compared with real

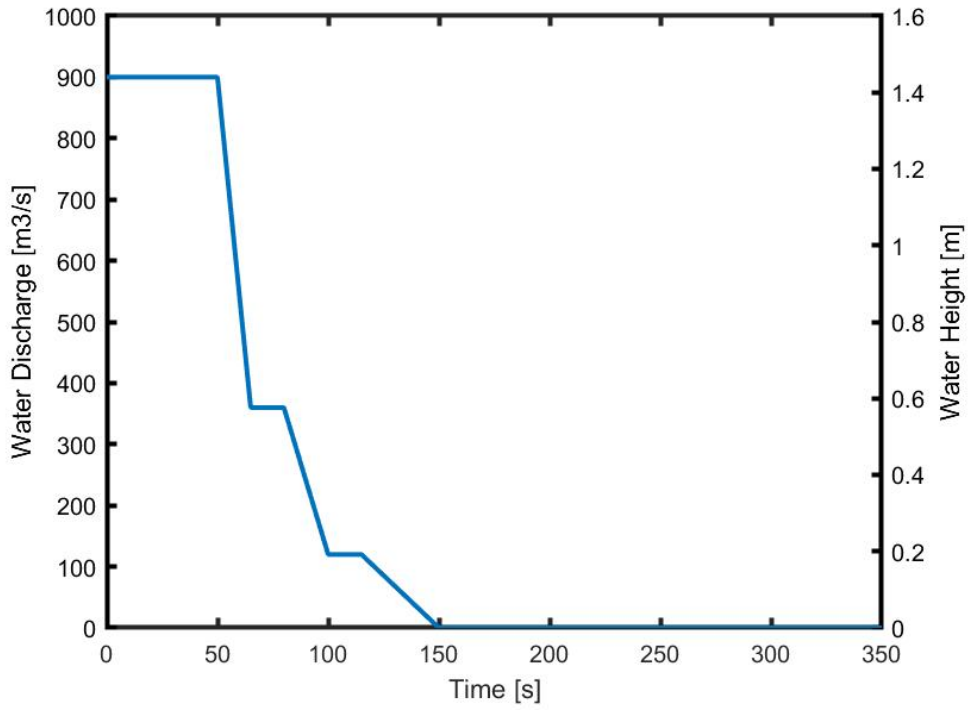


Figure 4.22: History of the water discharge and water height at the inflow boundary.

situations, the simulated pieces of driftwood generally accumulate in areas **B2**, **B3** and **B4**.

In **B2**, there was a narrow channel between the elderly care facility and the small cabin. The width was $12\,m$, twice the driftwood length. When one piece of driftwood collided with the facility wall or the small cabin, it was decelerated and made the channel narrower. The following pieces of driftwood were easily trapped. Consequently, a large amount of driftwood had become trapped by $175\,s$. Fig. 4.25 shows the driftwood accumulated in the simulation from camera angle **C1** indicated in Fig. 4.24. A comparison with the on-site photo (Fig. 4.26) taken from a similar camera angle reveals differences in the shape and density of driftwood accumulation. However, our simulation result indicated that driftwood tended to accumulate in this area.

In **B3**, driftwood had already accumulated around the corner of a house standing on a slant to the water flow by $175\,s$. The water flow forked in two directions at the obstacle. When the water height and flow speed decreased, some logs hit the corner of the house and lost speed, so that they easily became trapped near the house. When other trees collided

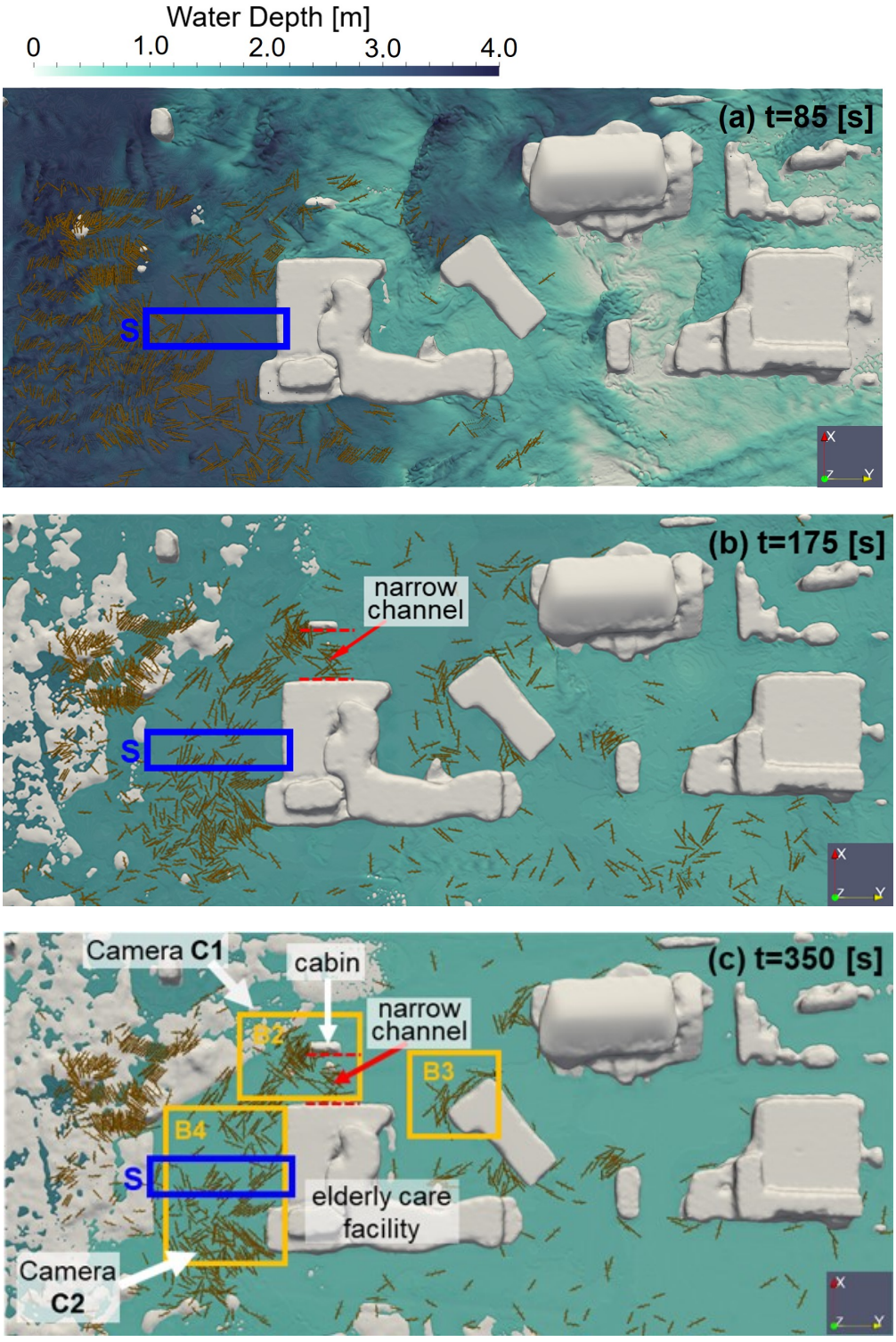


Figure 4.23: Simulation snapshots (top view).

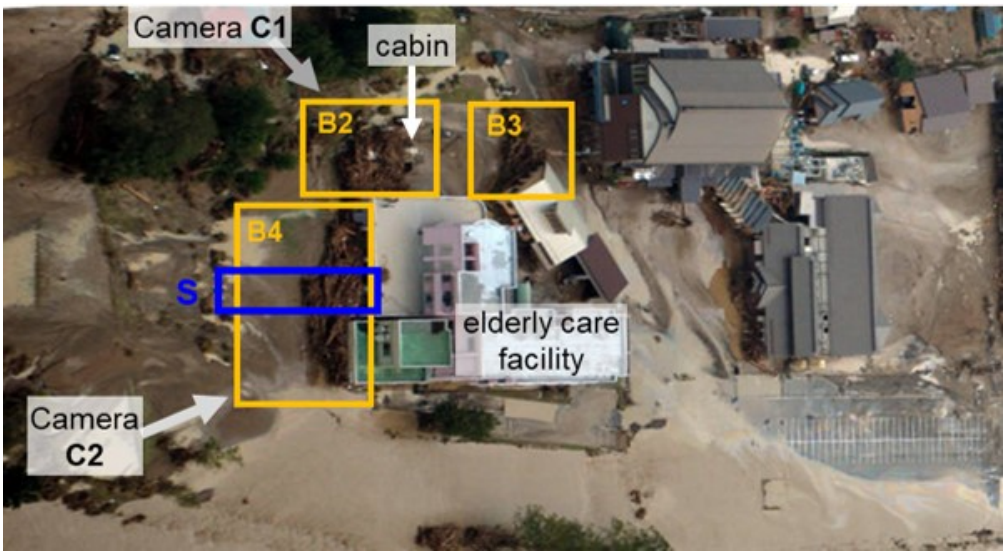


Figure 4.24: On-site photo of the same area after the disaster.

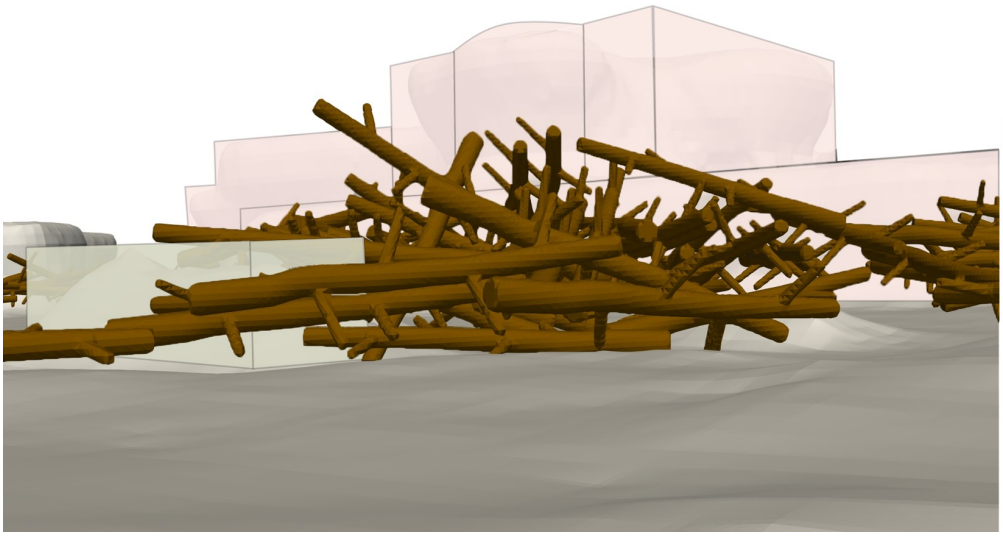


Figure 4.25: Snapshot of simulation from Camera **C1** at $t = 350.0\,s$.

Figure 4.26: On-site photo with a similar camera angle as Fig. 4.25.

with the trapped ones, their branches hooked the trapped trees and they became trapped as well.

In **B4**, the front of the building was free of driftwood until 175 s. The flow direction was almost normal to the building wall and it stagnated. The wood drifted at a certain water depth and did not reach the building wall because of the flow stagnation. Fig. 4.27 shows a series of cross-sectional views of Zone **S** in Fig. 4.23. At $t = 80\,s$, the water height was $3.2\,m$, and all of the pieces of driftwood were floating in the stagnation state. At $t = 136.5\,s$, the water height decreased to $2.0\,m$ and some of the pieces touched the ground and started gathering because the friction with the ground reduced the moving speed and the branches hooked the driftwood to each other. The accumulated driftwood weakened the stagnation effect and the floating pieces gradually began moving to the building, as shown in Fig. 4.27 (c). At $350\,s$, the water height decreased more and most of the deeper driftwood touched the ground. The shallower driftwood piled up and moved quite slowly toward the front of the building wall.

In the simulation result (Fig. 4.23 (c)), the driftwood pieces were distributed over a wide area and did not pile up in front of the building in area **B4**, whereas in the actual situation (Fig. 4.24), pieces of driftwood were densely clustered in this region, highlighting the discrepancy between the simulation and the real-world observation. A comparison of

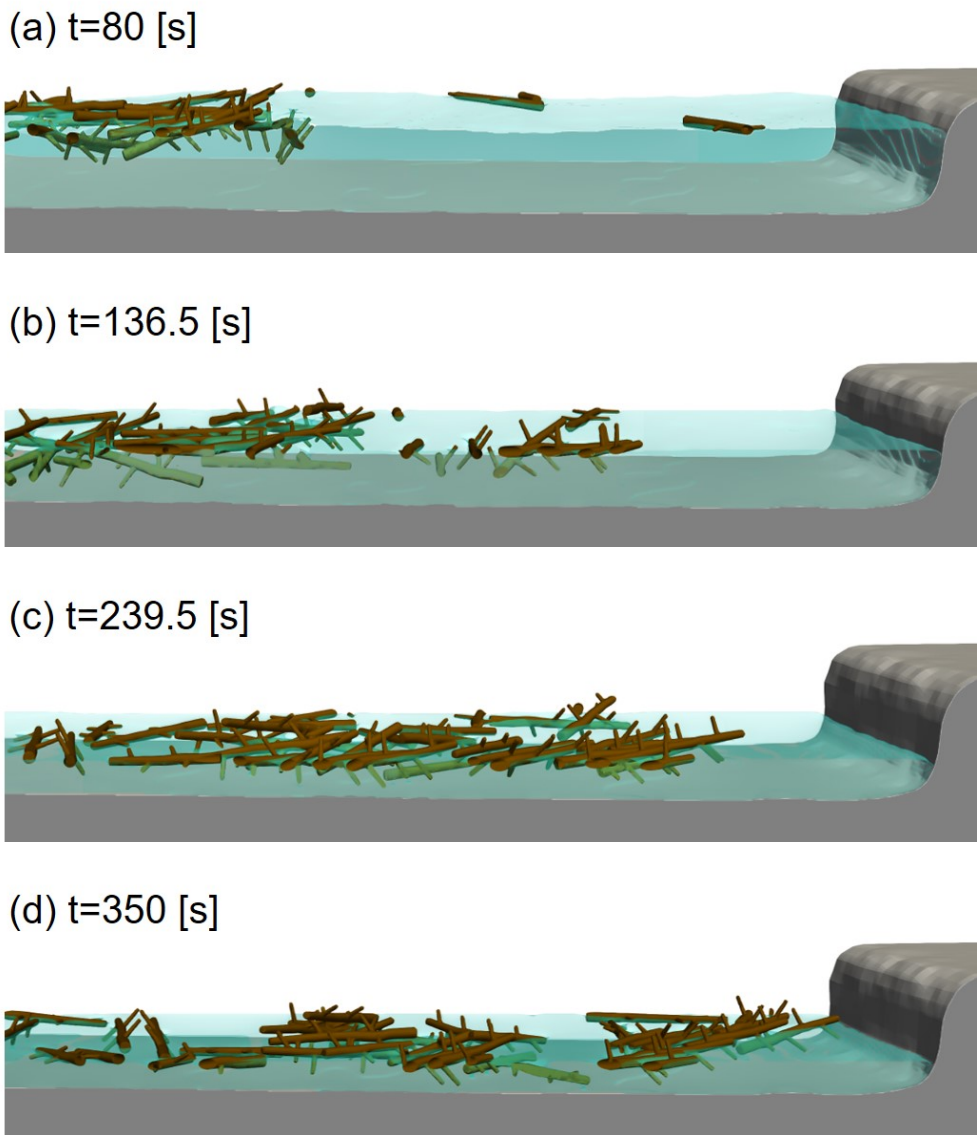


Figure 4.27: Snapshots of the cross-sectional view of Zone **S**.

Fig. 4.28 and Fig. 4.23, which depict driftwood accumulation in front of the building from camera angle **C2**, further demonstrates the difference in accumulation patterns. Although the simulation result generally reflects the inclination for driftwood to accumulate in front of the building in area **B4**, developing new methods to improve the accuracy of the simulation in this area will be a crucial task for future research.

Figure 4.28: Snapshot of simulation from Camera **C2** at $t = 350.0\,s$.

Figure 4.29: On-site photo with a similar camera angle as Fig. 4.28.

It should be noted that we can observe the presence of some small-scale debris in the on-site photos. We consider that small pieces of wood, glass and debris would enhance the accumulation of driftwood. These are a category of multi-scale and multi-physics and a new

model should be developed to take into consideration of small-scale debris in the future work.

The calculation was executed on 28 NVIDIA Tesla V100 GPUs of the FLOW supercomputer. It took 220 hours of computational time to complete a simulation up to 350 s on the real clock.

We conclude that a large-scale computation with high-resolution mesh can reproduce natural flooding disasters with driftwood and is useful for learning the details of such disasters. This knowledge will be useful for drawing up disaster prevention plans.

Chapter 5

3-D Simulation for Driftwood Trapping

According to the World Bank data, the forest coverage percentage of Japan in 2022 was 68.4%. While abundant forest resources provide numerous benefits for Japan, they are vulnerable to being uprooted by the strong winds when a typhoon strikes. The subsequent heavy rainfall can wash these fallen trees into river channels, turning them into driftwood. A large amount of driftwood destroys structures such as bridges or buildings, posing significant threats to human life and property, as we discussed in Chapter 4. To mitigate the damage of driftwood, the adoption of driftwood trap facilities should be considered. To contribute to the mitigation of driftwood disasters, we have applied a code for 3-D simulations that incorporates free-surface flow, multiple pieces of driftwood, and solid structures such as closed dams, driftwood trap devices, and river beds.

Two validation tests for small-scale hydraulic model experiments were conducted to assess the feasibility of our code for simulating driftwood trap devices. For a real closed dam with a driftwood trap device, we examined flows including 1,000 pieces of driftwood with different parameters. We aimed to identify the factors influencing the capture efficiency of trap devices and to reveal phenomena that are difficult to observe in experiments, such as the movement of driftwood underwater and the porosity of the trapped area.

5.1 Simulation of the Driftwood Trapping Experiment

In 2022, Dr. Shima conducted small-scale hydraulic model experiments [3] to evaluate the effectiveness of two kinds of driftwood trap devices (linear layout and convex layout) that were installed upstream of impervious concrete closed dams. We attempted to reproduce these experiments using numerical simulations.

Fig. 5.1 shows the waterway model used in both the experiments and simulations. The channel had a slope of 1/40. In Case 2 (case name in the reference paper), a linear layout driftwood trap facility was installed with a river width of 85.7 cm and a water passage width of 10.7 cm. The water passage width is 1/8 of the river width. In Case 10 (case name in the reference paper), a convex-layout capture device was used with the same river width, but the water passage width was set to 1/2 (42 cm) of the river width. A water height gauge was installed on the dam to measure the dammed-up depth. Both cases were simulated to compare with the experimental results.

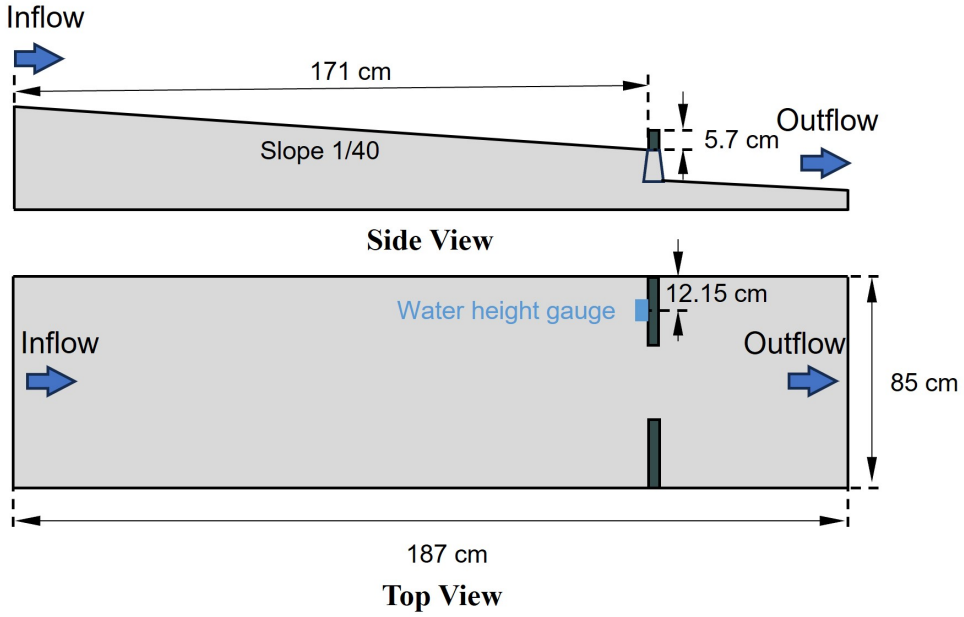


Figure 5.1: Waterway model for experiment and simulation.

We assigned AMR meshes to the vicinity of the water surface, the surface of the driftwood, and near the riverbed. The AMR meshes had five different levels for mesh refinement. The

size of the coarsest mesh was $\Delta x_0 = 1.0625\,cm$. The first refined mesh size was $\Delta x_1 = 0.5\Delta x_0 = 5.3125\,mm$, the second $\Delta x_2 = 0.5\Delta x_1 = 2.65625\,mm$, the third $\Delta x_3 = 0.5\Delta x_2 = 1.328125\,mm$. The finest mesh size was $\Delta x_4 = 0.5\Delta x_3 = 0.6640625\,mm$. The total number of computational cells reached a maximum of 443,060,224. If a uniform grid had been used, the number of cells would have been $1280 \times 2816 \times 512$, meaning that the computational cost and memory consumption were reduced to one-quarter of that required for a uniform grid. The time step for fluid calculations using the LBM was set to $\Delta t_{LBM} = 3.32 \times 10^{-5}\,s$ for the finest cells.

The simulation parameters were set to be as close as possible to those of the experiment. The computational parameters were a water density of $1000\,kg/m^3$ and a kinematic viscosity of $1.004 \times 10^{-6}\,m^2/s$. Non-slip boundary conditions were applied to the side and bottom walls, while the Dirichlet boundary condition was applied to the inflow boundary, and the Neumann boundary condition was adopted for the outflow boundary. In the experiments, driftwood models were introduced at a rate of 100 pieces per minute from the upstream channel. It would take more than 5 or 10 minutes to introduce 500 or 1000 pieces, respectively. However, simulating up to 5 to 10 minutes of physical time would result in extremely long computation times. Therefore, in the simulations, the only difference from the experimental setup was the release rate of driftwood. In Case 2, 100 pieces of driftwood were introduced every $3.5\,s$, and in Case 10, 100 pieces were released every $1.9\,s$. Although the driftwood number density increased with the release frequency, the effect of the number density increase was considered minimal because the length of the driftwood exceeds the spacing between components, making it more likely for the driftwood to become captured.

The driftwood model had a length of $7.0\,cm$ and a diameter of $0.2\,cm$. For collision calculations between solid objects, 568 micro-spherical particles were used for each driftwood model, and a total of 3,949,837 particles were placed on the riverbed and dam structures. The restitution coefficient between particles in the DEM was set to 0.25, with a Poisson's ratio of 0.33 and a friction coefficient of 0.3. The time step for DEM calculations was set to 1/10 of Δt_{LBM}, which means during each fluid calculation step, DEM computations were sub-cycled 10 times.

The simulation for Case 2 was performed on sixteen NVIDIA H100 GPUs on the TSUB-

AME4.0 supercomputer for 120 hours, resulting in a physical time of 35.1 s with $1,059,200$ steps. For Case 10, the calculation was conducted under the same conditions for 130 hours, yielding a physical time of 38.0 s with $1,145,600$ steps. Some specifications of the NVIDIA H100 GPU are shown in Table. 5.1.

Table 5.1: Specifications of NVIDIA H100.

Nvidia H100	
Number of CUDA cores	16,896
Peak single-precision floating point performance	60Tflops
Peak double-precision floating point performance	30Tflops
Memory size	94GB HBM3
Memory bandwidth	3.9 Tbytes/sec

5.1.1 Development of Random Release of Driftwood

In the experiment, pieces of driftwood were released randomly into the water channel. To address this, it is better to introduce the random driftwood release to the simulation as well.

The random release of driftwood can be divided into three aspects: random release position, random release angle, and random release timing. In our code, all solid objects must be initialized within the computational domain at the beginning, and the DEM method prohibits any overlap between solid objects. If the positions and rotation angles of the driftwood are set randomly, there is a high risk of overlap between the driftwood pieces. In that case, we decided to assign a fixed position to the center of mass of each driftwood and allow only random rotation around the vertical axis to avoid overlap between them. Fig. 5.2 shows the top view of the initial setting of randomly rotated driftwood pieces around the vertical axis.

Next, to implement the random release timing of the driftwood, we utilized the std : $random\_shuffle()$ function in C++ language. In the initial setup, each piece of driftwood is assigned an object ID. For example, if 1,000 pieces of driftwood are set in the computational domain, the object IDs range from 1 to 1,000, which is an ordered sequence and not suitable

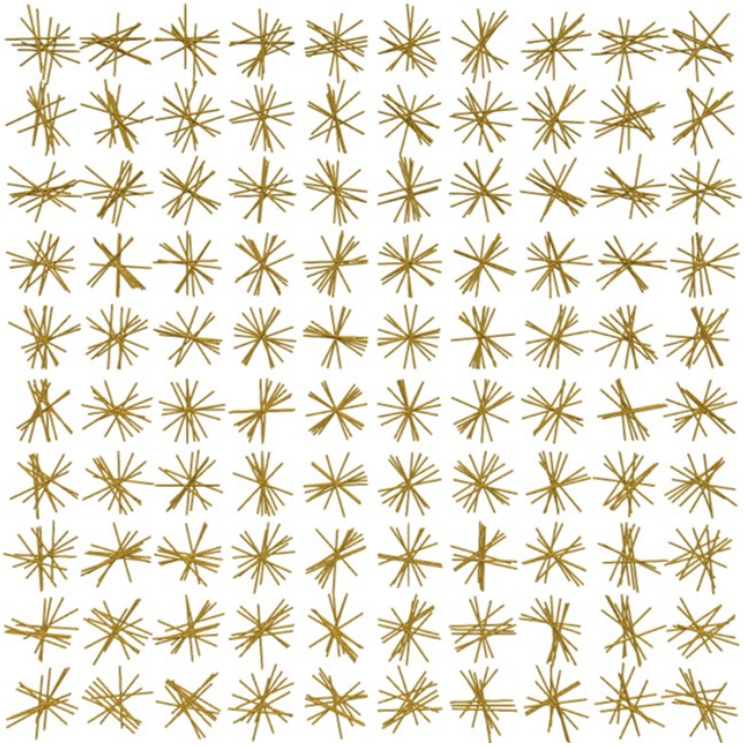


Figure 5.2: Top view of initial setting of driftwood.

for random release.

To create a random release sequence, we defined a new array $order[1000]$ in the code and assigned values from 1 to 1,000 to this array. Then, we used the $std : random\_shuffle()$ function to shuffle the numbers in the array $order$ to create a random sequence. Finally, the object IDs were matched with the numbers from $order[0]$ to $order[999]$, thereby determining the release timing for each piece of driftwood.

5.1.2 Simulation of the Linear Layout (Case 2) of the Driftwood Trap Device

First, we conducted a simulation for Case 2. As shown in Fig. 5.3, a driftwood capture device with a linear layout was installed $5\,cm$ upstream of an impervious concrete dam. The water passage width was 1/8 of the river width. The driftwood trap device consisted of a structure with 9 capture spans, as illustrated in Fig. 5.4. The width of each span was $3.0\,cm$ and the diameter of each pole was $0.5\,cm$. The total width of the trap device was $31.5\,cm$

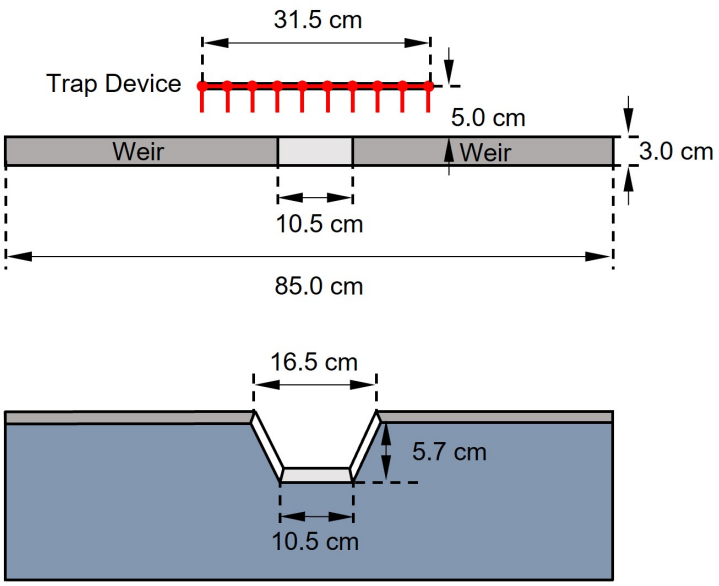


Figure 5.3: Location of the trap device and shape of the water passage in Case 2.

and the height of the device was 6.0 cm.

The simulation was conducted with an inflow flux of 1.5 L/s from the left end of the channel, as shown in Fig. 5.1. The time history of simulation snapshots is shown in Fig. 5.5, where the time located at the upper-left corner represents the elapsed time after the driftwood release. For the first 100 pieces of released driftwood, 60% were captured, while 90% of the subsequent 100 pieces were trapped. Since the length of the driftwood was twice the spacing between the components of the capture device, it was more likely to be trapped. The driftwood was captured almost horizontally by the device, blocking the flow near the water surface. As a result, the area near the surface became a ponding state, while in the central area, water flowed underneath the trapped driftwood and discharged through the water passage.

Fig. 5.6 compares the flow in the simulation and the experiment when approximately 190 pieces of driftwood are captured. The driftwood acts like tracer particles, and vortices rotating in opposite directions on the left and right sides were observed both in the simulation and the experiment. By the time 200 pieces of driftwood were released, the water surface near the driftwood trap device had been blocked, resulting in almost all subsequently introduced driftwood being captured. Fig. 5.7 shows the uncaptured driftwood ratio for every 100 pieces

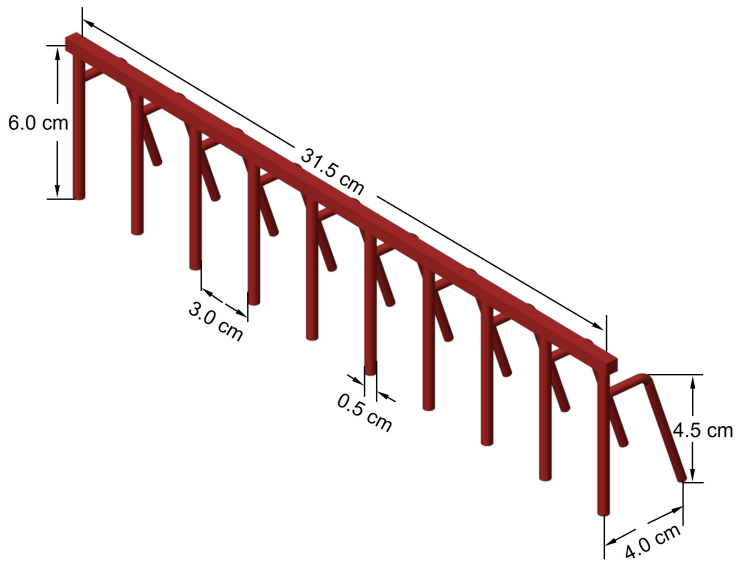


Figure 5.4: Trap device with linear layout.

of introduced driftwood. The simulation results agree well with the experimental results.

Fig. 5.8 illustrates the variation in the dammed-up depth caused by the accumulation of the captured driftwood. The numbers in the figure represent the total number of released driftwood pieces up to that point, and the blue circles indicate the dammed-up depths measured in the experiment. Due to the differences in the timing of driftwood release between the experiment and the simulation, a direct comparison of the time history of dammed-up depth was not possible. In the simulation, the water level began to rise when around 100 pieces of driftwood reached the capture device, and it stabilized at approximately 400 pieces ($20\,s$). Compared with the initial water depth of $4\,cm$, the maximum dammed-up depth in the simulation was $0.25\,cm$, while the experiment showed a final dammed-up depth of $0.2\,cm$. Although the trends in dammed-up depth were similar, the simulation exhibited a more gradual increase. It seemed reasonable that the water level rose progressively over time, considering the ponding effect.

Fig. 5.9 shows the distribution of driftwood near the trap device after 500 driftwood pieces were released. The distribution observed in the simulation agreed well with the experiment. Fig. 5.10 illustrates a view of the driftwood capture situation underwater, which was difficult to measure in the experiment. A considerable amount of driftwood pieces was captured

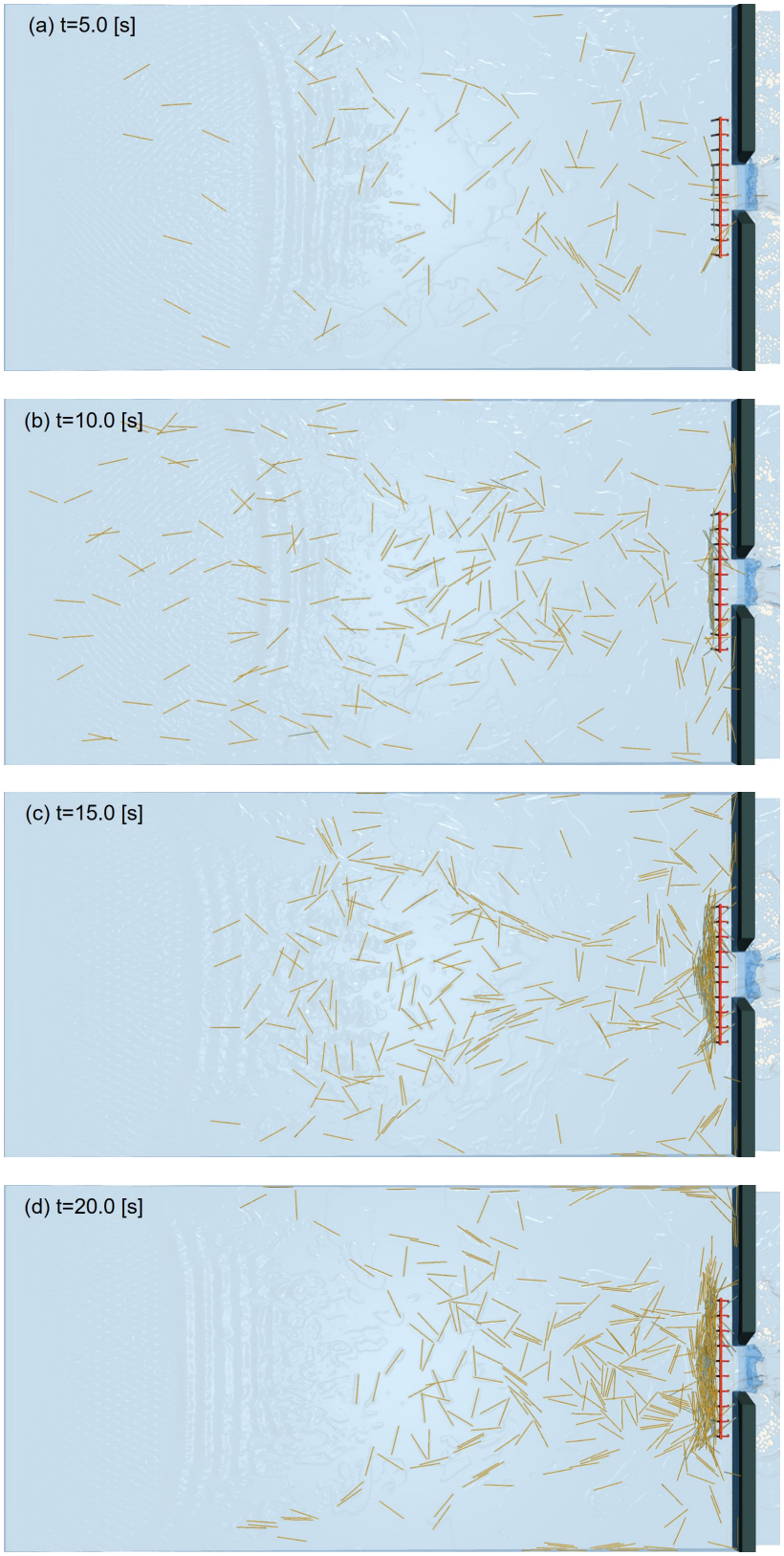


Figure 5.5: Time history of simulation snapshots for Case 2.

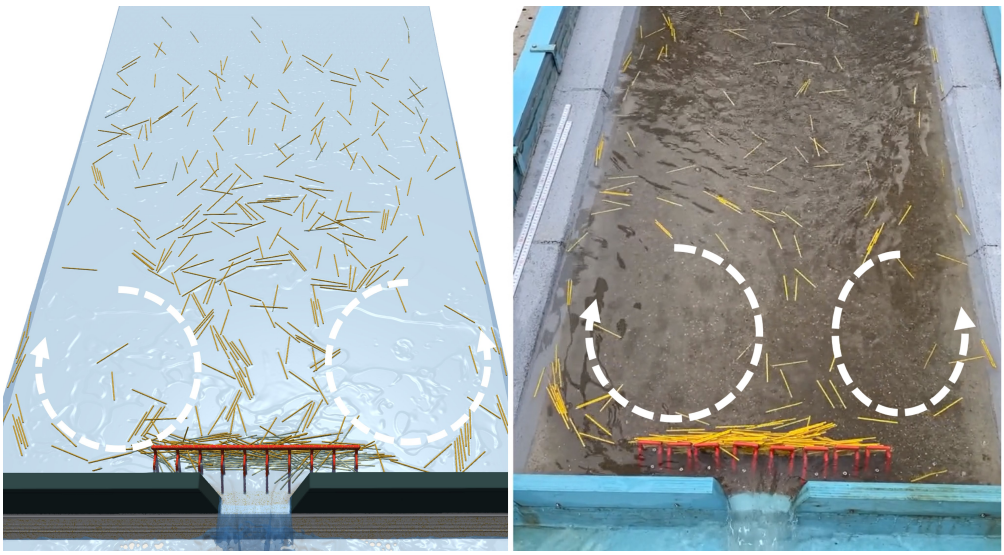


Figure 5.6: Snapshots of the simulation and the experiment at 190 pieces of trapped driftwood.

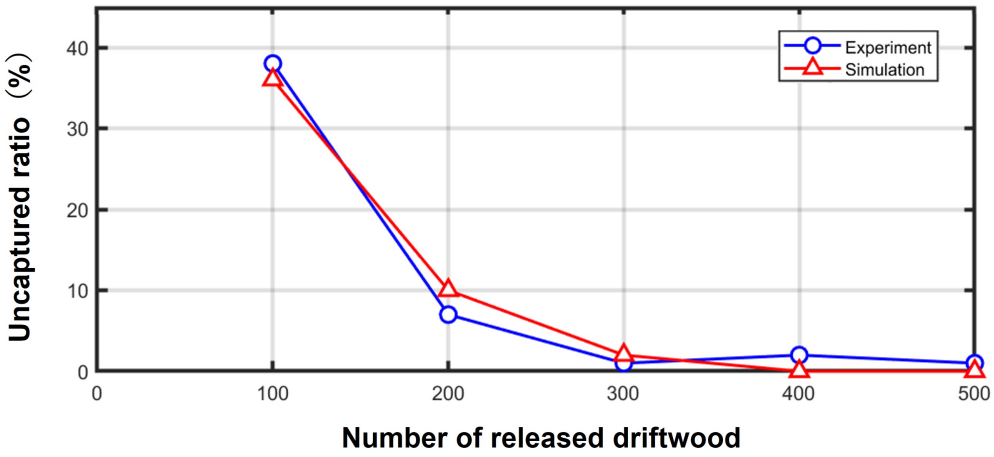


Figure 5.7: Uncaptured driftwood ratio – every 100 pieces of released driftwood in Case 2.

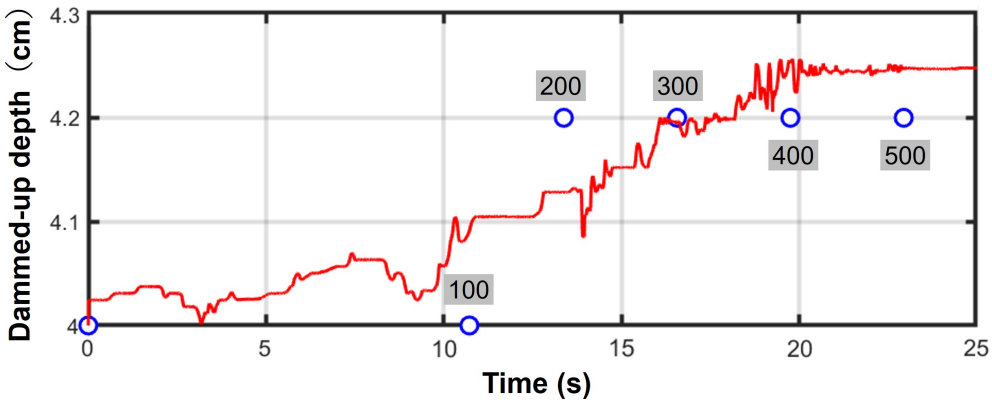


Figure 5.8: Dammed-up depth – time after driftwood release in Case 2.

below the water surface. The porosity of the trapped region in front of the capture device, measuring $31.5\,cm \times 2.7\,cm \times 4.25\,cm$ (outlined by the yellow dashed line in Fig. 5.9 and Fig. 5.10), was 34.73%, which indicated significant blockage that contributed to the damming effect upstream of the dam.

5.1.3 Simulation of the Convex Layout (Case 10) of the Driftwood Trap Device

Next, we conducted a simulation for Case 10. Fig. 5.11 shows the location of the trap devices and the shape of the water passage for Case 10. Seven driftwood capture devices were arranged in a convex layout upstream of the impervious concrete dam. The width of the water passage was 1/2 of the channel width. Each driftwood trap device had a structure with 3 capture spans, as depicted in Fig. 5.12. The width of each trap device was $10.5\,cm$ and the total width of the seven devices was $73.5\,cm$.

The flow was classified as a wash flow due to the wide water passage. Although the experiment used a mobile river bed, its influence on driftwood capture was considered negligible. Therefore, the bed was treated as fixed in the simulation. The number of driftwood pieces introduced in the simulation was 1,000, the same as the experiment. The inflow flux for Case 10 was $3.5\,L/s$. Other settings were the same as those described in Section 5.1.2.

The time history of simulation snapshots for Case 10 is shown in Fig. 5.13. The time

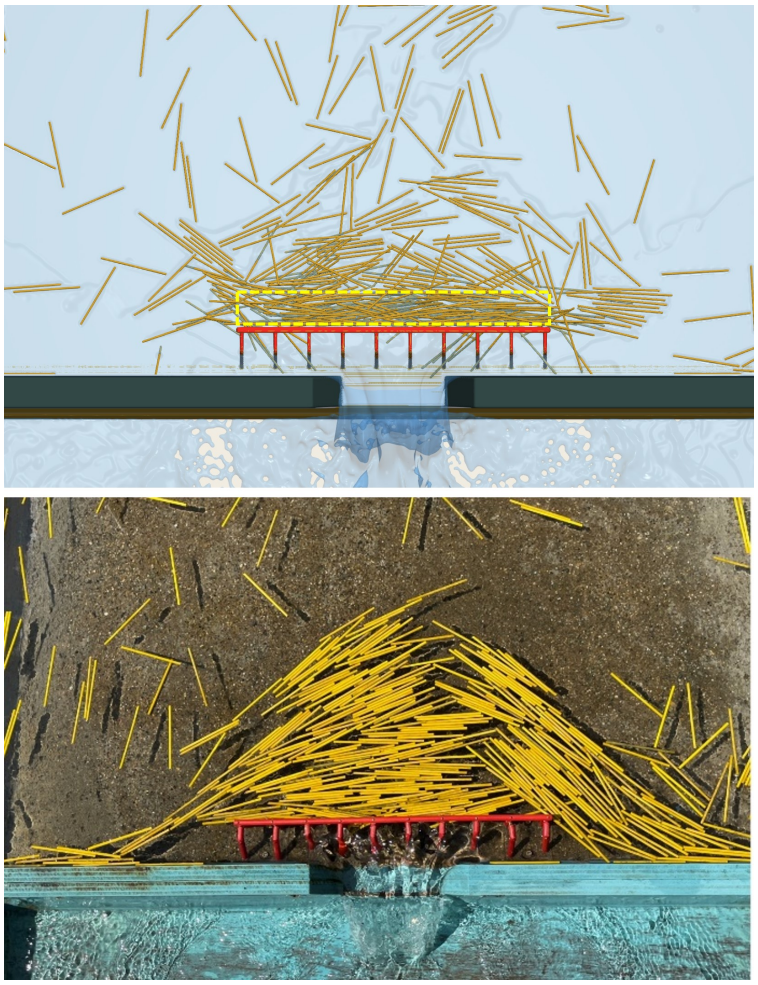


Figure 5.9: Driftwood distributions of simulation (upper) and experiment (lower) after 500 pieces of released driftwood in Case 2.

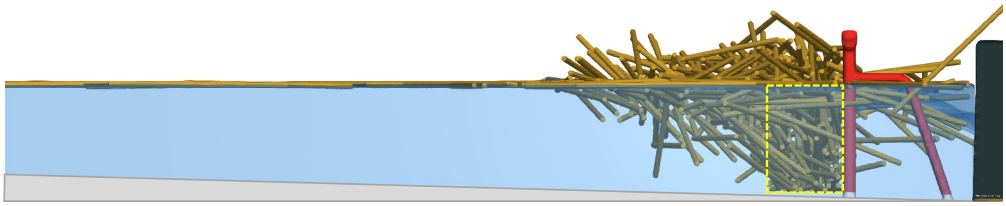


Figure 5.10: Side view of the simulation result for Case 2.

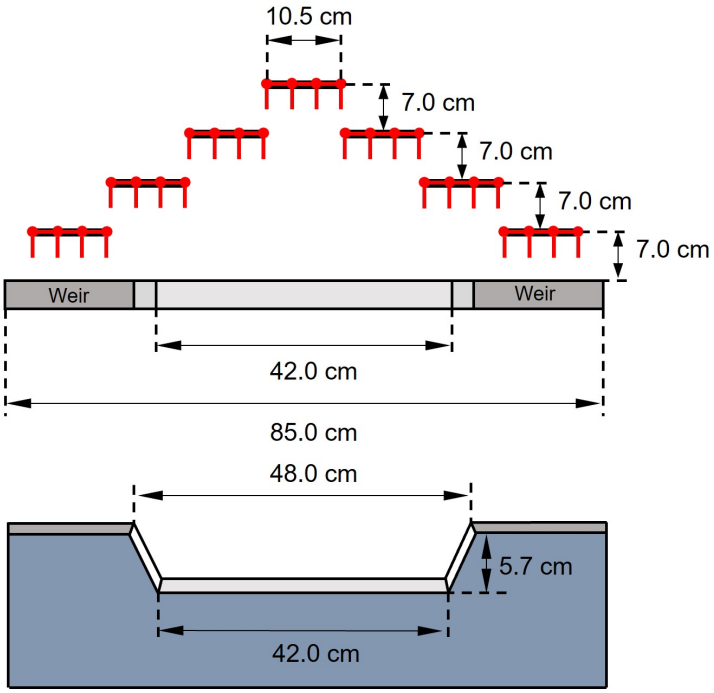


Figure 5.11: Location of the trap device and shape of the water passage in Case 10.

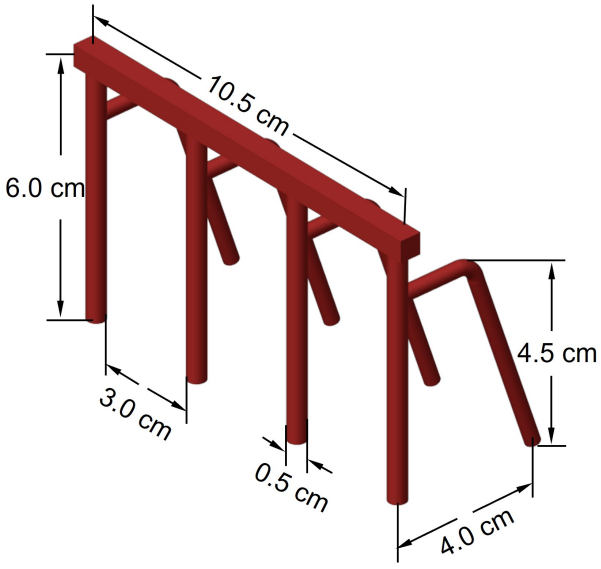


Figure 5.12: Unit of driftwood trap device with 3 spans.

displayed in the upper-left corner indicates the elapsed time since the driftwood release. As each trap device had only 3 spans for capturing driftwood, the capture efficiency of a single device was lower compared with that in Section 5.1.2. When driftwood collided with a single pole at the side of a trap device, it tended to rotate in the direction of the flow. Subsequently, the direction of the driftwood gradually became perpendicular to the capture device. This allowed the driftwood to pass more easily through the gaps between the downstream devices. As a result, the wash flow condition near the water surface persisted for an extended period, and the fast water flow speed further reduced the likelihood of driftwood being captured. Consequently, even after 400 pieces of driftwood were introduced, the uncaptured driftwood ratio remained above 10%. However, as more driftwood began to form "bridges" between trap devices, the water surface near the devices became almost completely obstructed after 700 pieces were introduced. This led to a substantial decrease in the uncaptured driftwood ratio. The relationship between the uncaptured ratio and the number of released driftwood pieces is shown in Fig. 5.14, which demonstrates a fair agreement between the simulation and the experimental results.

In the early stages of driftwood release, the ponding state occurred in the left and right side areas in front of the weirs. This caused the flow to concentrate in the center. As a result, the central trap devices captured the driftwood first. Pieces of driftwood gradually began to be captured by the devices on the sides as well. Fig. 5.15 shows the velocity vectors of the water surface when 750 pieces of driftwood were captured. Since the central area near the water surface was almost obstructed, the flow separated from the center and diverted toward the sides. This led to increased driftwood capture by the trap devices farther from the center. In Fig. 5.15, the flow velocity at the water surface was higher in the upstream region far from the trap device, but it decreased as the flow approached the captured driftwood. This indicated that the flow velocity increased beneath the accumulated driftwood, meaning the flow reached the water passage by passing under the trapped driftwood.

Fig. 5.16 shows the time history of the dammed-up depth in both the simulation and experiment for Case 10. The dammed-up depth began to increase after approximately 300 pieces of driftwood were introduced (with 224 pieces captured), and by the time 400 pieces were released (with 314 pieces captured), the depth had increased by $0.3\,cm$. After 800

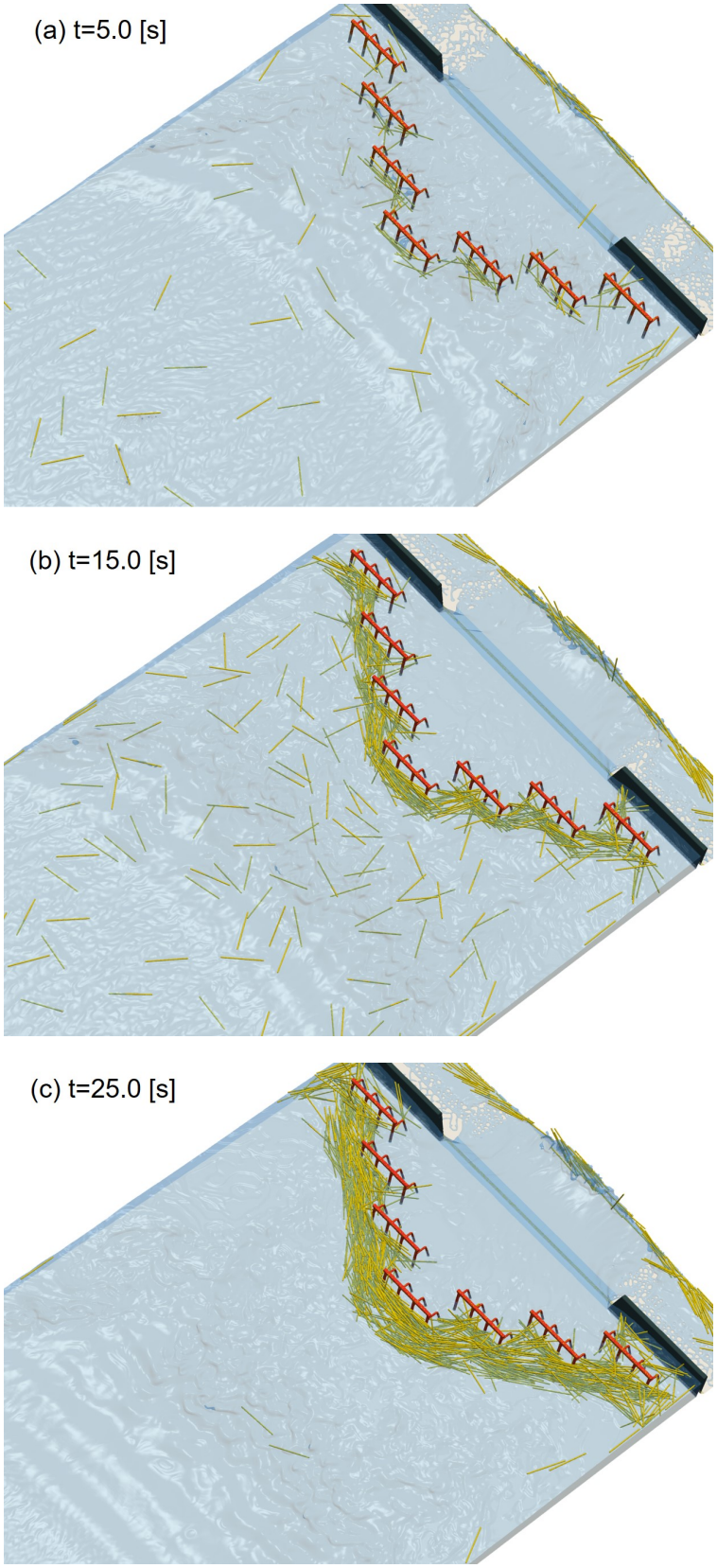


Figure 5.13: Time history of simulation snapshots for Case 10.

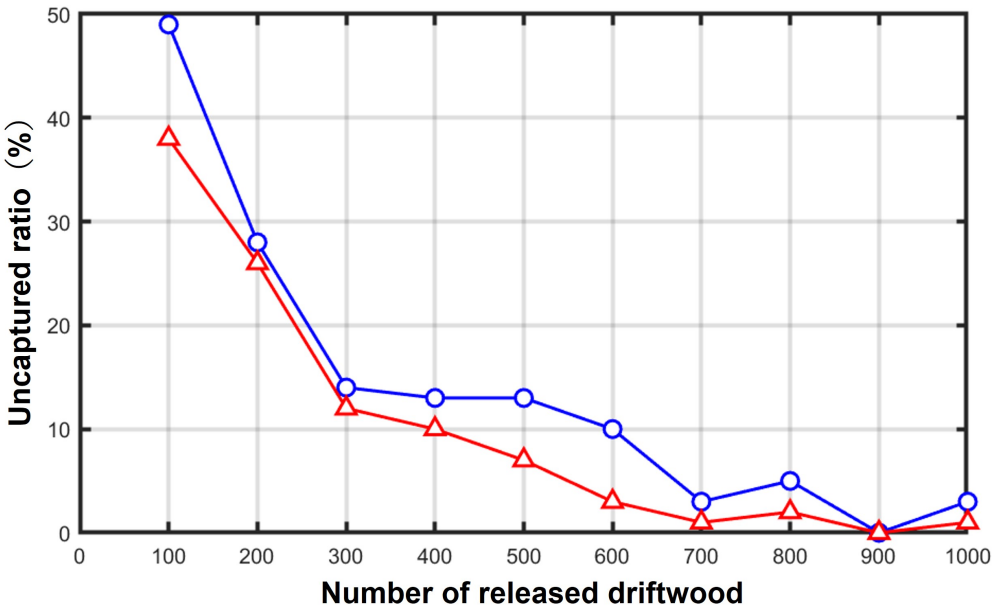


Figure 5.14: Uncaptured driftwood ratio – every 100 pieces of released driftwood in Case 10.

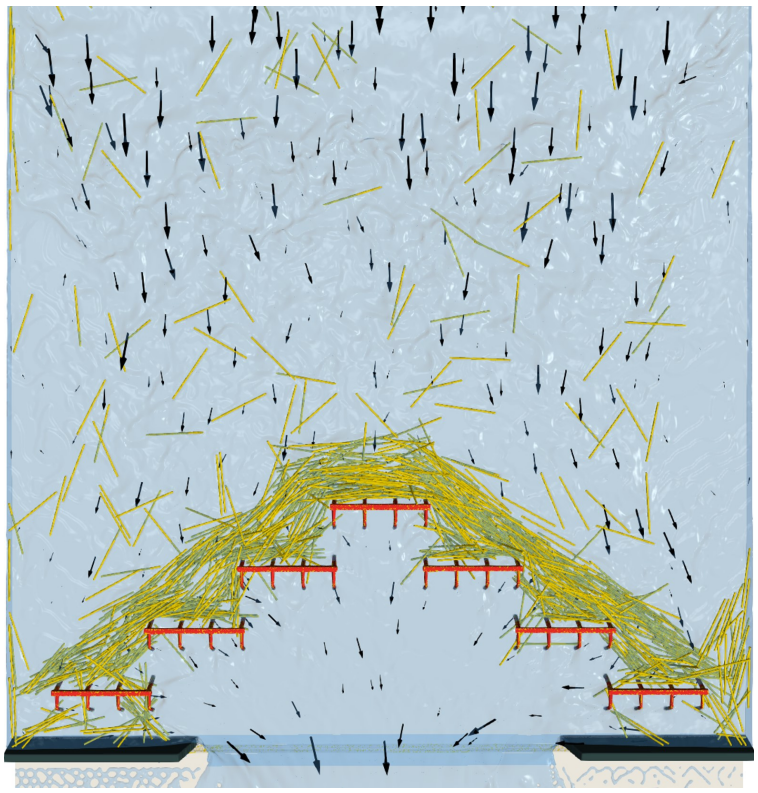


Figure 5.15: Flow vectors on the water surface at 750 pieces of trapped driftwood (the lengths indicate the absolute values).

pieces were dropped (with 701 pieces captured), the dammed-up depth remained nearly constant. The experiment showed a similar trend and the dammed-up depth stabilized at around 4.3 cm, which closely matched the simulation results.

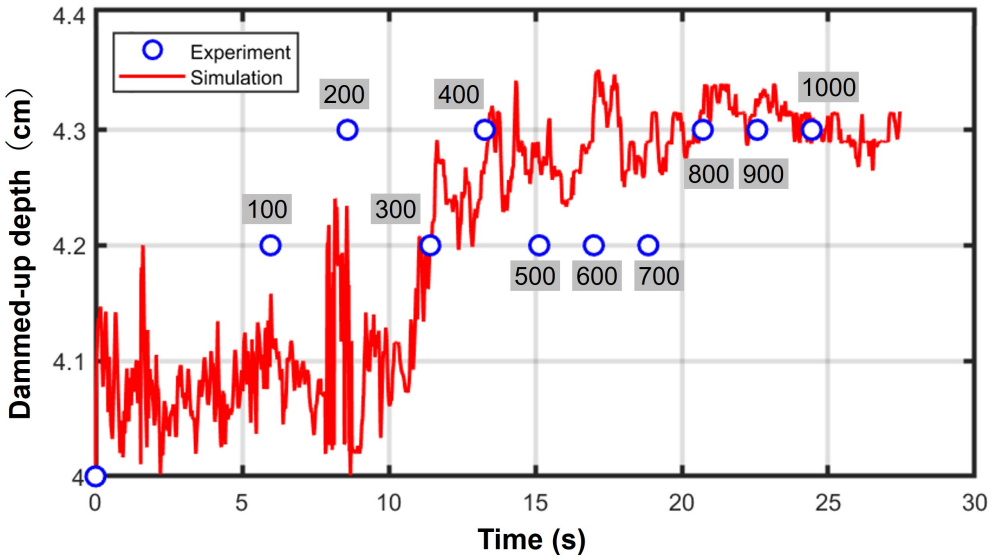


Figure 5.16: Dammed up depth – time after driftwood release for Case 10.

Fig. 5.17 displays the final distribution of driftwood after 1,000 pieces were released. Driftwood pieces were captured evenly across all seven trap devices, and the simulation results agreed well with the experimental snapshots. The way in which the driftwood bridged between the trap devices also closely resembled the experimental results. Furthermore, in the devices near the left and right banks, the amount of captured driftwood was initially low due to the ponding state. However, as the central area became blocked by captured driftwood, the flow separated to the two sides and then carried driftwood towards these devices. This enhanced the capturing process of the side trap devices. The distribution of driftwood captured near the side sections also showed a good agreement between the simulation and the experiment.

Fig. 5.18 shows a side-view snapshot of the left half of the 900 pieces of driftwood captured in the final state. Compared with Fig. 5.10, the average porosity of the trapped area was 63.8% because the driftwood pieces did not reach the riverbed in many areas. This suggested that the flow passed near the riverbed and continued toward the water passage.

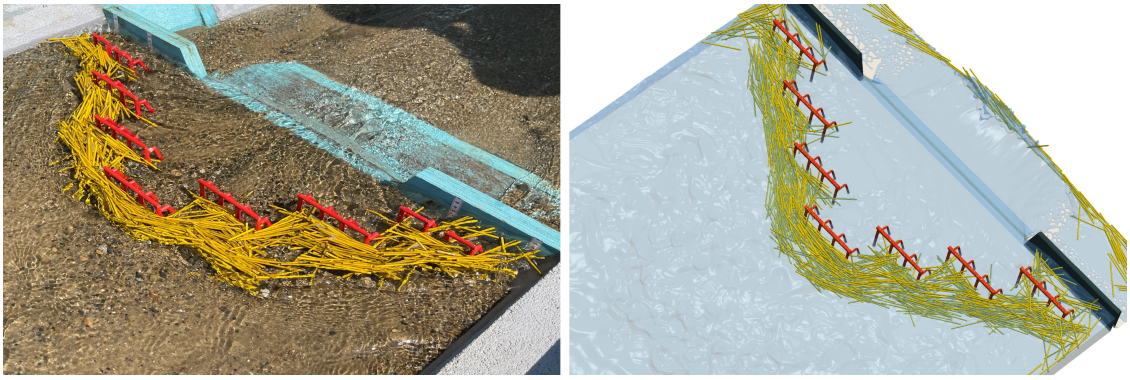


Figure 5.17: Driftwood distributions of simulation (right) and experiment (left) after 1,000 pieces of driftwood release.

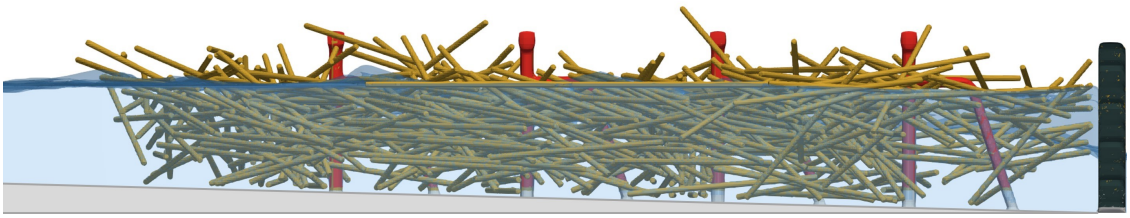


Figure 5.18: Side view of the simulation result for Case 10.

The relatively high porosity also explained why the increase of dammed-up depth remained almost unchanged in Case 10, despite twice the amount of released driftwood compared with Case 2.

We conclude that the simulation reproduced the driftwood capture processes and the flow phenomena in Case 2 and Case 10. The simulation code is applicable to driftwood capture simulations.

5.2 Real-scale Driftwood Capture Simulation

After confirming the feasibility of our code for driftwood capture simulations, we aimed to evaluate the capture effectiveness of a real-scale driftwood capture device installed on an actual impervious concrete dam. Fig. 5.19 illustrates the shape and size of the simulation model. The sediment deposition area in front of the dam was assumed to be fully filled with sediment (sand or rocks). The 10-meter-long area upstream of the dam was set to a horizontal plane, while the 35-meter-long area in front of the plane was set as a 1/20 slope. The channel width was $16.4\,m$, representing a relatively small impervious dam. The flow passing through the dam was discharged through an outflow boundary condition located 8.5 meters below the dike to prevent any impact on the upstream flow. Five-meter-high walls were set on both sides to prevent overflow from the riverbanks. The width of the water passage was set to $7.0\,m$, which was approximately half of the total channel width.

The shape and size of the real-scale driftwood trap device are shown in Fig. 5.20. Each pole had a diameter of $0.4\,m$, with a gap of $1.6\,m$ between adjacent poles. The total width of the trap device was $12.0\,m$. To prevent driftwood from escaping around the sides, two poles were installed on each side and the side length of the trap device was $3.2\,m$. Since we assumed a fully sediment-filled condition, the trap device was exposed $2.5\,m$ above the riverbed. The device was directly installed on the dike, and it enclosed the flow passage.

The inflow flux was set to $49.2\,m^3/s$, with water entering from the inflow area at a depth of $1.5\,m$ and a flow velocity of $2.0\,m/s$. Driftwood pieces were released at the slope region in the upstream area, where the flow velocity was approximately $2.6\,m/s$. We used AMR meshes for the simulations. The AMR meshes had four different levels for mesh

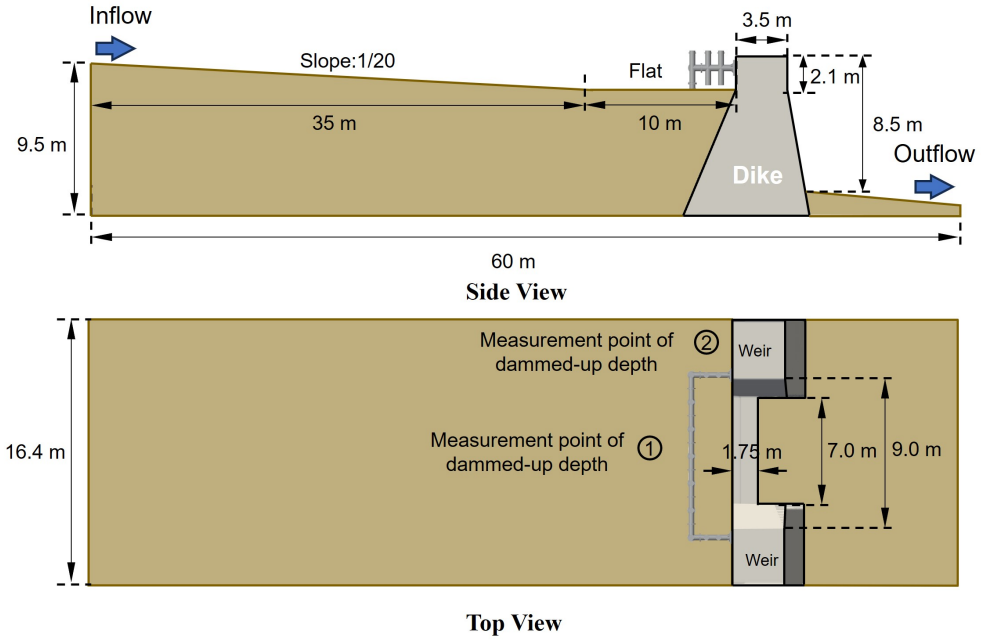


Figure 5.19: Simulation model for dike and river bed.

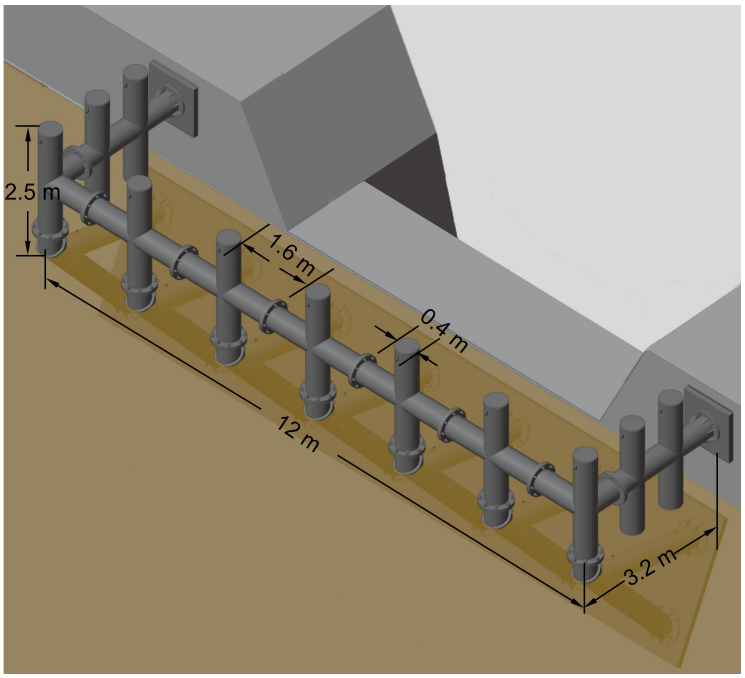


Figure 5.20: Shape of the real-scale driftwood trap device.

refinement. The size of the coarsest mesh was $\Delta x_0 = 28.0\,cm$. The first refined mesh size was $\Delta x_1 = 0.5\Delta x_0 = 14.0\,cm$, the second $\Delta x_2 = 0.5\Delta x_1 = 7.0\,cm$. The finest mesh size was $\Delta x_3 = 0.5\Delta x_2 = 3.5\,cm$. The total number of computational cells reached a maximum of 436,207,616. If a uniform grid had been used, the number of cells would have been $512 \times 1,664 \times 512$, meaning that the computational cost and memory consumption were reduced to one-quarter of that required for a uniform grid. The time step for fluid calculations using the LBM was set to $\Delta t_{LBM} = 2.1 \times 10^{-4}\,s$ for the finest cells.

The computational parameters were a water density of $1000\,kg/m^3$ and a kinematic viscosity of $1.004 \times 10^{-6}\,m^2/s$. Non-slip boundary conditions were applied to the side and bottom walls and the dike, the Dirichlet boundary condition was applied to the inflow boundary and the Neumann boundary condition was adopted for the outflow boundary.

Table 5.2: Parameters for each simulation case.

Case number	Driftwood length L [m]	Driftwood diameter D [cm]	Driftwood density $[kg/m^3]$	Number of released driftwood	Release frequency [pieces/s]
Case 0	1.8	20	850	1000	100/5.6
Case 1	1.8	16	850	1000	100/5.6
Case 2	1.5	16	850	1000	100/5.6
Case 3	1.2	16	850	1000	100/5.6
Case 4	0.8	16	850	1000	100/5.6
Case 5	0.5	16	850	1000	100/5.6
Case 6	1.2	16	750	1000	100/5.6
Case 7	1.2	16	950	1000	100/5.6
Case 8	0.8	16	850	500	100/30

The parameters for each simulation case are shown in Table 5.2. The driftwood models had two diameters, $20\,cm$ and $16\,cm$, with either 1,000 or 500 pieces introduced to the calculation domain. The capture efficiency was evaluated for various driftwood lengths ranging from $0.5\,m$ to $1.8\,m$. The mass density of the driftwood was set to $850\,kg/m^3$. For collision calculations between solid objects, a total of 11,668,273 particles were distributed on the riverbed and dam structures. Each piece of driftwood was assigned between 200 and 568 micro-spherical particles, depending on its size. The restitution coefficient between particles in the DEM was set to 0.25, with a Poisson's ratio of 0.33 and a friction coefficient of 0.3.

The time step for DEM calculations was also set to 1/10 of Δt_{LBM}.

Although the riverbed was a mobile bed in reality, it was treated as a fixed bed for the simulations due to the short duration of the simulated physical time, which was approximately 3 minutes. For Case 0 to Case 7, the simulations required 158 hours using eight NVIDIA H100 GPUs. This resulted in a physical time of $194.88\,s$ with $928,000$ steps. For Case 8, the calculation took 200 hours to simulate $258.72\,s$ of physical time with $1,232,000$ time steps.

5.2.1 The Effect of Driftwood Diameter on Driftwood Capture Efficiency

First, to investigate the influence of driftwood diameter on capture efficiency, simulations were conducted with driftwood length of $1.8\,m$ and diameters of $20\,cm$ and $16\,cm$ in Case 0 and Case 1, respectively. Both cases released 1,000 pieces of driftwood. Since the driftwood length exceeded the $1.6\,m$ spacing between adjacent poles of the capture device, the pieces were easily captured. Driftwood was released at a rate of 100 pieces every $5.6\,s$. This led to an average number density of $0.54\,pieces/m^2$ at the water surface. The flow reached a nearly steady state at around $80\,s$, after which driftwood pieces were released. In both Case 0 and Case 1, the first driftwood piece reached the capture device at around $92\,s$, and driftwood accumulation became stable by $194\,s$. In Case 1, 977 out of 1,000 pieces of driftwood were captured. A snapshot of the flow exiting the capture device from the downstream side at this point is shown in Fig. 5.21. Fig. 5.22 presents the driftwood outflow rate for every 100 pieces released for both cases. For the first 100 pieces of driftwood, 83 pieces were captured in Case 0, while 84 pieces were captured in Case 1. Almost all pieces of subsequently released driftwood were captured. Contrary to expectations, the thinner (diameter $16\,cm$) driftwood was captured slightly more effectively. However, the difference of just one piece between the $20\,cm$ and $16\,cm$ diameters was not statistically significant.

Fig. 5.23 demonstrates the time history of the dammed-up depth at measurement point ①, which is located at the center in front of the capture device, as shown in Fig. 5.19. In both Case 0 and Case 1, the dammed-up depth increased as the amount of captured driftwood

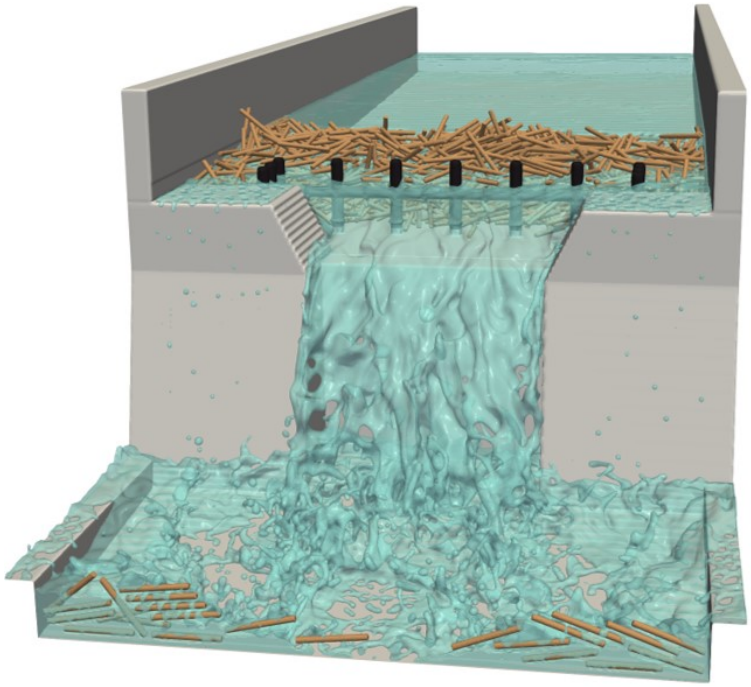


Figure 5.21: Snapshot of Case 1 simulation at 977 pieces of captured driftwood.

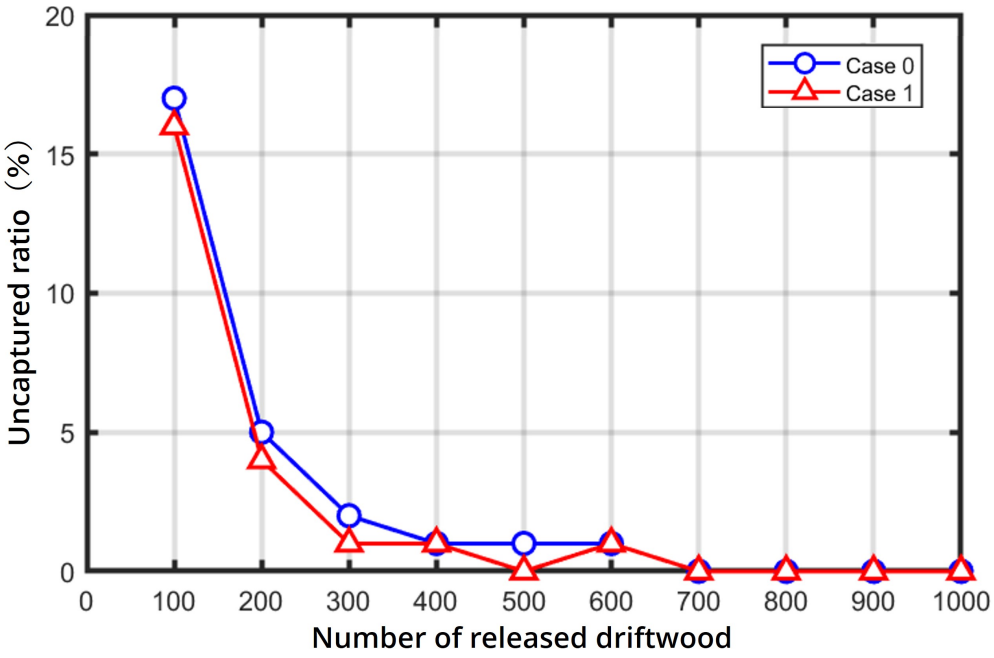


Figure 5.22: Uncaptured driftwood ratio – every 100 pieces of released driftwood for Case 0 and Case 1.

accumulated over time. After 185 s, when all 1,000 pieces of driftwood had reached the trap device, the dammed-up depth stabilized. The final depth was slightly higher in Case 1, where the driftwood diameter was 16 cm.

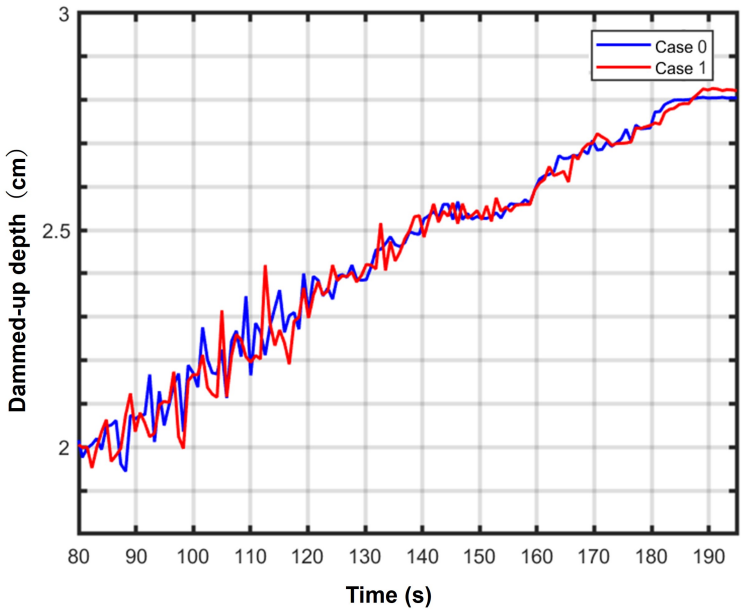


Figure 5.23: Time history of dammed-up depth for Case 0 and Case 1.

Fig. 5.24 presents cross-sectional images of the final capture state at the central section for Case 0 and Case 1. The underwater porosity caused by captured driftwood in front of the trap device was 37.4% for Case 0 and 25.0% for Case 1. The length of the low-porosity region in the flow direction (highlighted by the yellow dashed line in Fig. 5.24) is considered a reliable indicator for the degree of occlusion. Consequently, the degree of occlusion was calculated by

$$Degree\ of\ occlusion = (1 - porosity) \times passage\ distance, \tag{5.1}$$

The degrees of occlusion for Case 0 and Case 1 were 3.70 and 3.81, respectively. The occlusion degree of Case 1 was slightly higher than that of Case 0. This could explain why the dammed-up depth of Case 1 was slightly higher than that of Case 0. Although the passage distance in Case 0 was longer, the larger driftwood diameter resulted in a higher porosity. Overall, the degree of occlusion showed a strong correlation with the dammed-up

depth.

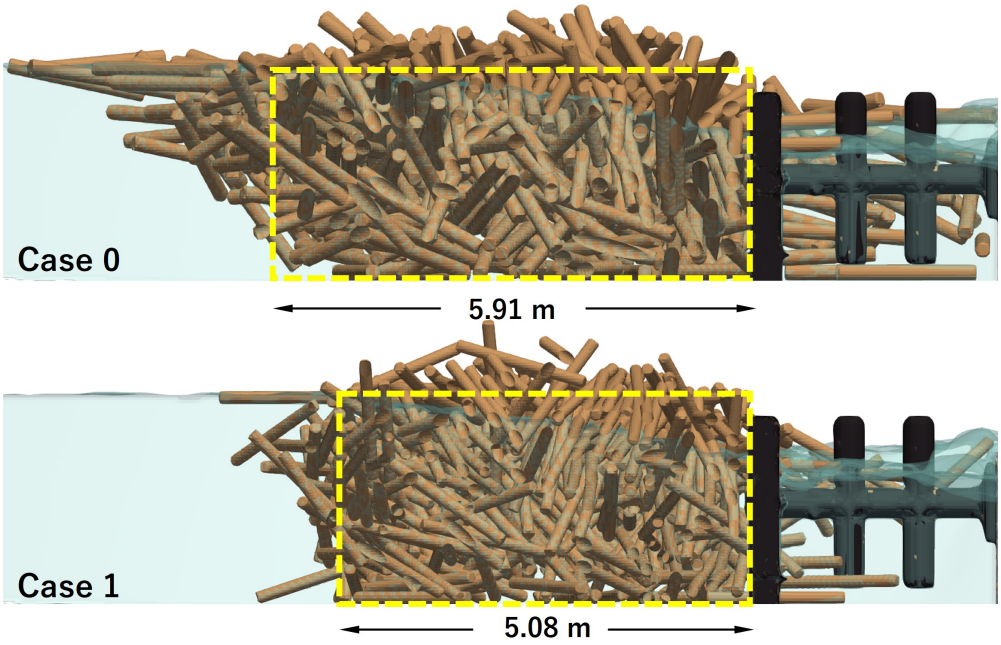


Figure 5.24: Cross-sectional view at the center in front of the trap device in the flow direction for Case 0 and Case 1.

Based on the above results, it was concluded that driftwood capture efficiency was not significantly influenced by driftwood diameter. Therefore, subsequent simulations were conducted with a driftwood diameter of 16 cm.

5.2.2 The Effect of Driftwood Length on Driftwood Capture Efficiency

Next, we examined the influence of driftwood length on capture efficiency, as it was expected to have the greatest impact. The following cases were simulated: 1.8 m (Case 1, previously calculated in Section 5.2.1) for the length longer than the 1.6 m spacing between the capture device poles, and shorter lengths of 1.5 m (Case 2), 1.2 m (Case 3), 0.8 m (Case 4), and 0.5 m (Case 5). Fig. 5.25 displays the distribution of captured driftwood pieces after 194 s. As the driftwood length decreased, it became noticeably more difficult for the trap device to capture the driftwood, and the area of driftwood accumulation in front of the device diminished.

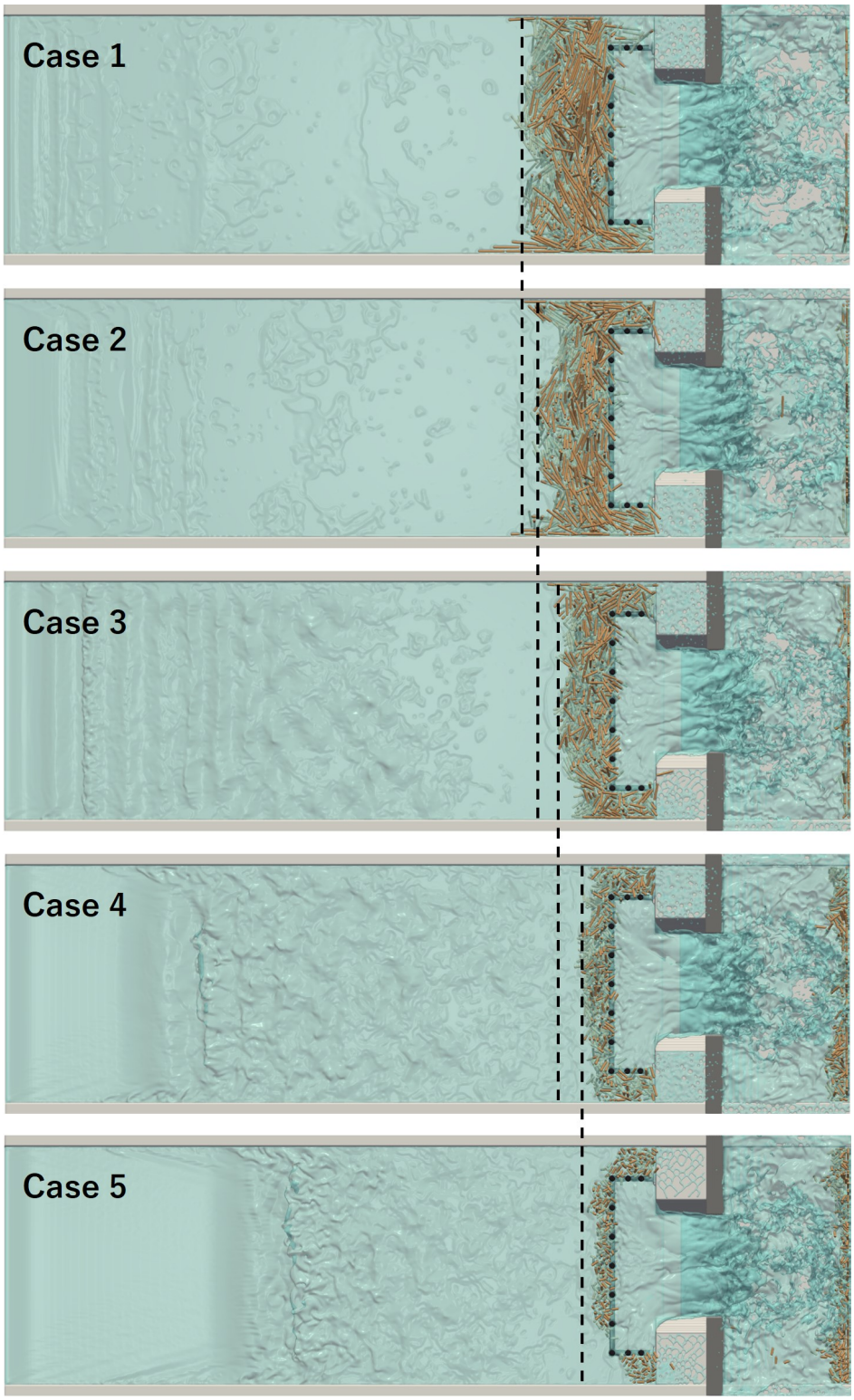


Figure 5.25: Top view of the finally trapped driftwood distribution for Case 1 to Case 5.

Fig. 5.26 shows the driftwood uncaptured ratio against the ratio of "driftwood length ÷ distance between trap device poles" on the horizontal axis. Even when the driftwood length was only 75% of the gap length, 90% of the driftwood was still captured. This was because any contact between the driftwood and a pole of the trap device caused the driftwood to rotate around the point of contact until it reached an angle where it could no longer move away from the device. During this rotation, the driftwood made the poles of the capture device appear thicker, which increased the likelihood of collisions with subsequent driftwood. The time required for this rotation depended on the angle of contact and the distance from the mass center of the driftwood. Longer driftwood had a larger rotational arc, which increased the duration it remained engaged with the trap device.

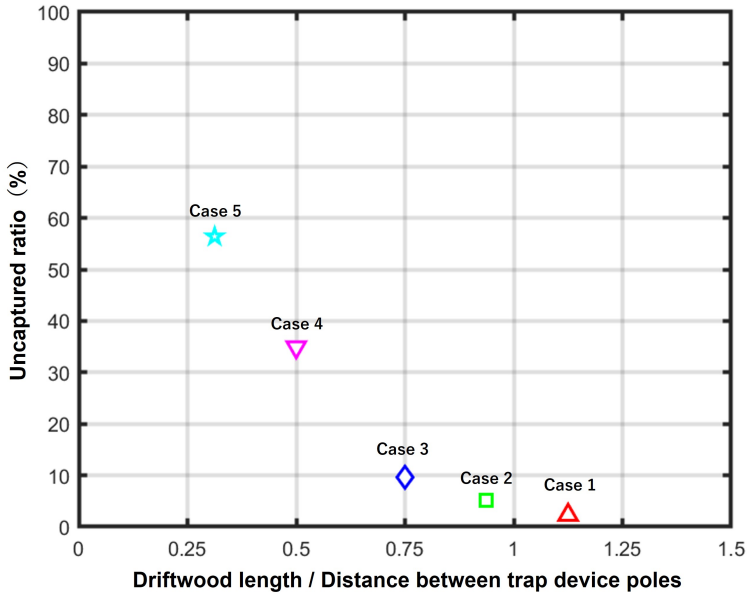


Figure 5.26: Uncaptured ratio – Driftwood length/Distance between trap-device poles for Case 1 to Case 5.

Fig. 5.27 demonstrates the uncaptured driftwood ratio for every 100 pieces of released driftwood. In the cases with shorter driftwood lengths, such as Case 4 and Case 5, a high uncaptured ratio persisted throughout the whole simulation. Fig. 5.28 provides a side view of the final captured driftwood distribution. The distribution of captured driftwood underwater is indicated by the red dashed line. This indicated that longer driftwood pieces formed a smaller angle with the water surface, while shorter pieces tended to approach a

nearly 90-degree angle. When the driftwood was submerged, the buoyancy force depended on the cross-sectional area of the driftwood because we used the same density for these cases. The downward flow appeared if the area near the water surface of the trap device became blocked. The upward buoyancy force proportional to the cross-sectional area of the driftwood acted against the downward flow. For the same diameter, longer driftwood had a larger cross-sectional area, which made it less likely to submerge. Consequently, longer driftwood tended to be captured closer to the water surface.

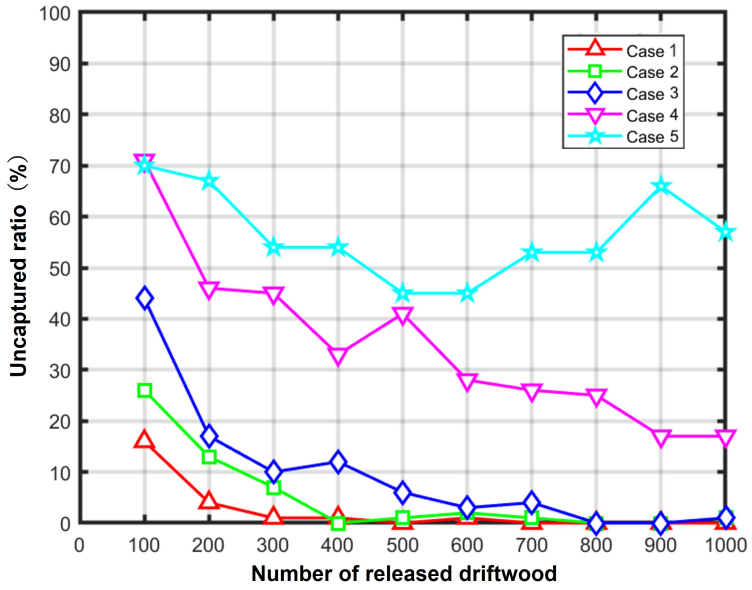


Figure 5.27: Uncaptured ratio – every 100 pieces of released driftwood for Case 1 to Case 5.

As shown in the Case 4 and Case 5 snapshots in Fig. 5.28, we can observe a gap between the captured driftwood and the riverbed. This gap indicated that some driftwood pieces had the potential to flow downstream through the gap. By reviewing the simulation video, it was clear that shorter driftwood pieces, which submerged more easily, were carried by the faster flow near the riverbed and passed through the trap device. This observation clarified why the uncaptured pieces of driftwood continued to increase even after 1,000 pieces had been released, as shown in Fig. 5.27. In Case 5, the shortest driftwood submerged more easily, and the faster flow in deeper water caused more driftwood to escape. This resulted in a shallower angle between the captured driftwood and the water surface.

Regarding the submersion of driftwood, a lower mass density resulted in relatively stronger

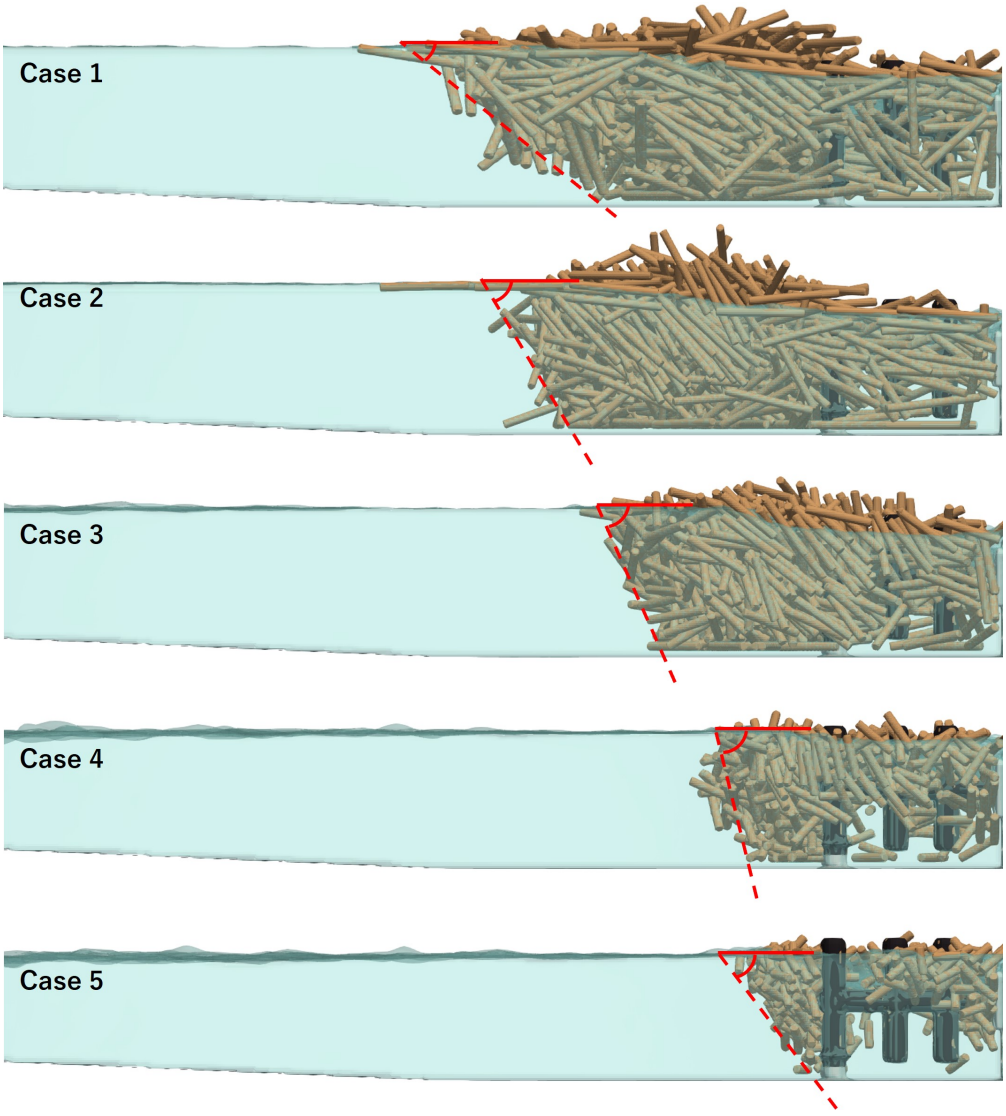


Figure 5.28: Side view of the simulation results for Case 1 to Case 5.

buoyancy forces. Consequently, additional simulations were conducted for Case 6, with the mass density reduced to $750\,kg/m^3$, and Case 7, with the mass density increased to $950\,kg/m^3$. As shown in Fig. 5.29, the angles between the captured driftwood and the water surface in Case 6 and Case 7 were almost the same as that observed in Case 3.

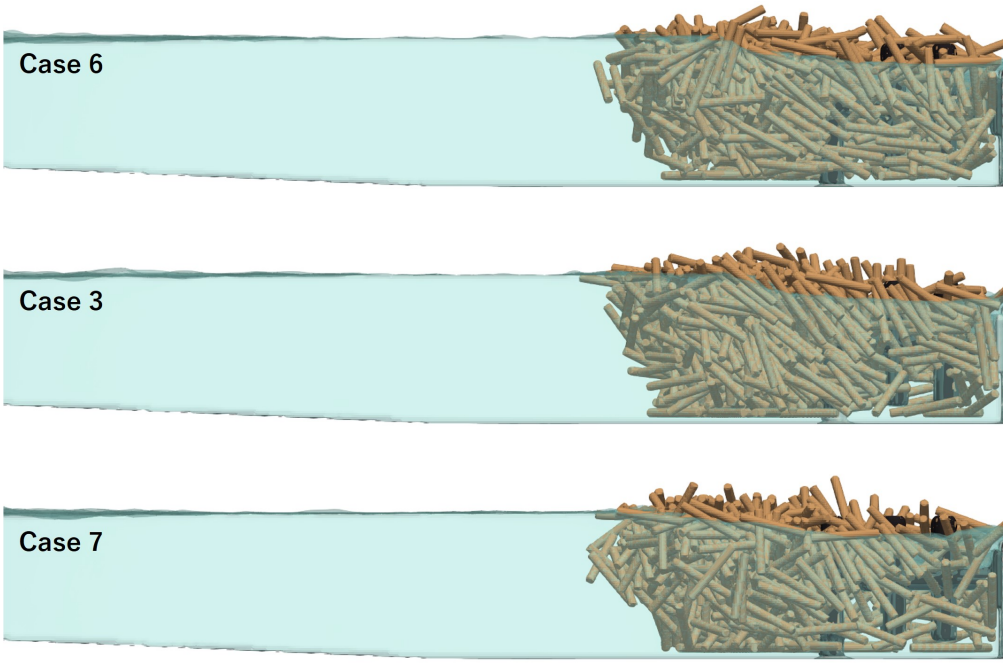


Figure 5.29: Side view of the simulation results for Case 6, Case 3 and Case 7.

5.2.3 The Effect of Driftwood Length on Dammed-up Depth

Then, we investigated the effect of driftwood length on the dammed-up depth. Once driftwood was captured, the flow became obstructed, and the damming effect occurred. The number of captured driftwood increased over time, and the width of the driftwood trapped area expanded upstream, as shown in Fig. 5.30.

Fig. 5.31 presents the simulated dammed-up depth at two measurement points. Table 5.3 shows the porosity in front of the capture device for Case 1 to Case 5. It can be observed that as the driftwood length decreased, the porosity increased. As more driftwood pieces were captured and the compression of captured driftwood intensified, the porosity decreased, and the distance that the flow passed through the highly accumulated driftwood area increased.

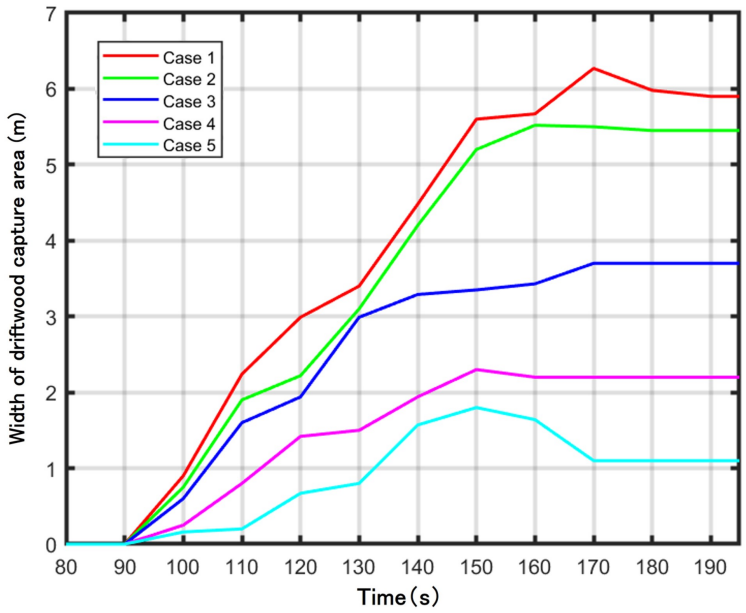


Figure 5.30: Width of driftwood capture area – time after driftwood release for Case 1 to Case 5.

The degree of occlusion (Eq. (5.1)) listed in Table 5.3 shows a good correlation with the dammed-up depth at measurement point ①.

Table 5.3: Porosity and occlusion for Case 1 to Case 5 at 140 s and 194 s.

	Case 1	Case 2	Case 3	Case 4	Case5
Porosity at 140 s	31.9%	34.5%	38.3%	48.2%	61.6%
Porosity at 190 s	25.0%	26.8%	30.7%	36.4%	62.9%
Occlusion	3.81	2.36	1.53	0.66	0.31

In Fig. 5.31, the damming effect at point ② is less obvious compared with point ①. This suggested that the flow was not redirected toward the side sections even when the central area was blocked. Instead, most of the flow still passed through the gaps between the captured driftwood in the central region. Fig. 5.25 shows that the damming effect propagated upstream. In Case 1, the influence of damming extended almost the entire 35-meter slope. Even in Case 5, compared with the condition when no driftwood was captured, the damming surface propagated 3.5 m upstream.

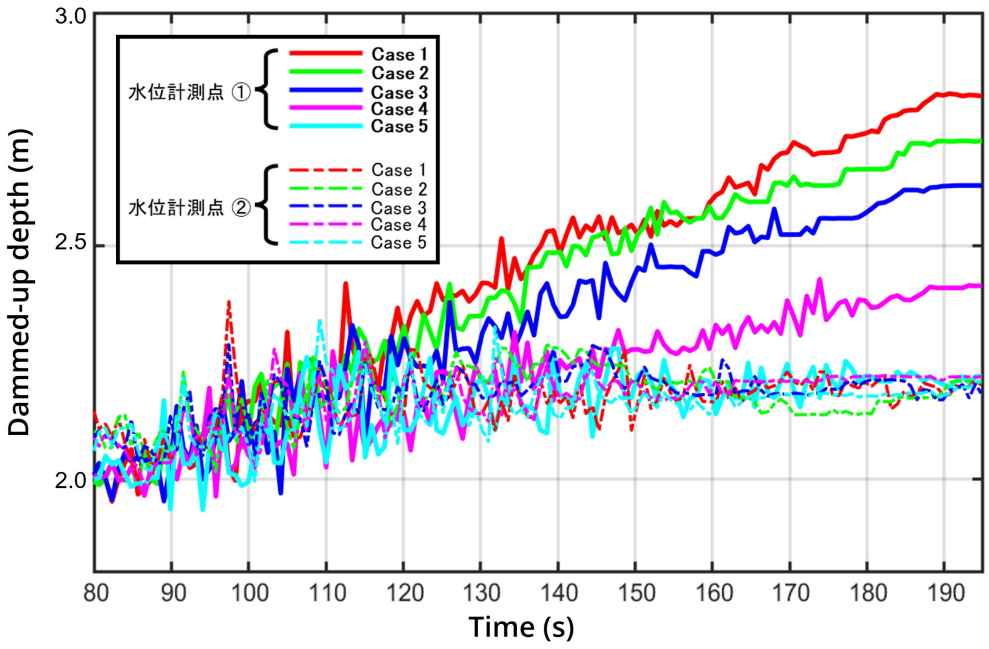


Figure 5.31: Dammed up depth – time after driftwood release for Case 1 to Case 5.

5.2.4 The Effect of Driftwood Number Density on Driftwood Capture Efficiency

Last but not least, we evaluated the effect of driftwood number density on trap efficiency. A higher driftwood density increased the likelihood of incoming driftwood colliding with driftwood already captured by the trap device, thereby enhancing capture efficiency. To investigate this effect, a simulation was conducted using the same driftwood length and diameter as in Case 4, but with a lower driftwood release frequency of 100 pieces every 30 s, instead of every 5.6 s. This simulation was designated as Case 8.

The average number density decreased from 0.54 $pieces/m^2$ in Case 4 to 0.13 $pieces/m^2$ in Case 8. During the period when the driftwood collided with the trap device and remained in the capture area, the lower number density in Case 8 indicated that fewer pieces arrived within the same time. This made it easier for the driftwood to pass through the device after colliding with the poles. Fig. 5.32 shows the distribution of captured driftwood after 500 released pieces. Compared with Case 4 in Fig. 5.25, the number of captured driftwood pieces in Case 8 significantly decreased.

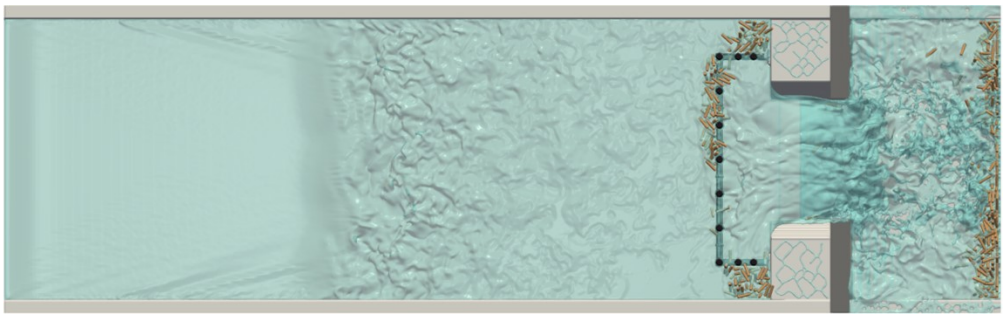


Figure 5.32: Top view of trapped driftwood distribution for Case 8.

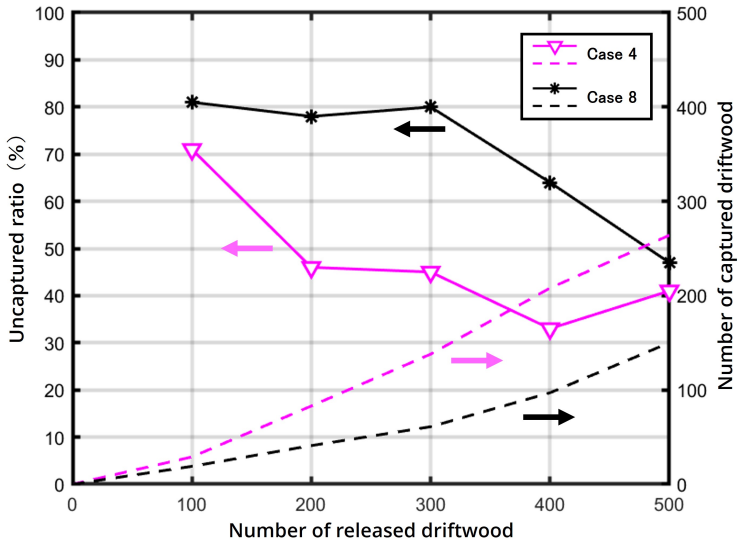


Figure 5.33: Uncaptured driftwood ratio and captured number of driftwood – every 100 pieces of released driftwood for Case4 and Case 8.

Fig. 5.33 shows the uncaptured driftwood ratio for every 100 released pieces and the cumulative number of captured driftwood. Although we introduced the same number of driftwood pieces in both Case 4 and Case 8, the number of pieces captured in Case 8 decreased substantially from 264 to 150. In Fig. 5.32, it is evident that no driftwood was captured on the perpendicular trapping surface of the lower half of the device (the left half in relation to the flow direction). In contrast, the side sections experienced an increased driftwood number density and a higher capture rate due to the ponding state.

In both Case 4 and Case 8, the driftwood length was shorter than the spacing between the adjacent poles of the trap device. However, when the driftwood length exceeded twice the spacing, the probability of the collision with two poles increased. Once the collision occurred, the likelihood of capture became much higher. As a result, driftwood number density had a limited impact on the capture efficiency of longer driftwood. In Section 5.1, although the driftwood release frequency (number density) differed significantly between the experiment and the simulation, its impact on capture efficiency remained minimal.

Chapter 6

Conclusion

6.1 Summary

With the development of CFD methods and computer hardware such as GPU, large-scale 3-D problems of high complexity have been allowed to be simulated and analyzed on supercomputers. In this thesis, we optimized and modified the original code developed by the Aoki Laboratory to make it more suitable for large-scale computations. The code employs the cumulant lattice Boltzmann method (LBM) with the $D3Q27$ lattice model, coupled with the conservative phase field (CPF) method, level-set method, discrete element method (DEM), and adaptive mesh refinement (AMR) method.

The forest resources are abundant in Japan. However, frequent typhoons and floods can easily uproot trees into river channels and turn them into driftwood pieces. Consequently, pieces of driftwood carried by river floodings can lead to secondary disasters, which pose a significant threat to human life and property. This thesis addressed the prediction and mitigation of large-scale driftwood disasters using the advanced computational methods described above.

Driftwood Disaster Prediction

For the 3-D large-scale simulations to predict a real driftwood disaster, the findings are summarized as follows:

(1) In small-scale simulations for water flume experiments, the results of the simulations

were in good agreement with those of experiments in terms of the shape of the water surface, water heights, the trapping rate of the driftwood, the angle formed between the trapped driftwood and the water surface, wave impact, and collision force. These results validated the robustness of our simulation approach.

(2) In the 3-D flooding simulation for a natural disaster that happened in 2016, our simulation demonstrated reasonable agreement with the real situation, particularly regarding the inundation process, the overflow in the central region, and the interception effect of the road bridge crossing the Omoto River.

(3) In the simulation with driftwood for the real disaster, higher resolution meshes were used. Although the simulation could not reproduce the exact driftwood distribution due to the use of many assumed parameters, it generally reflected similar driftwood accumulation locations as observed in the real situation. Furthermore, three different driftwood accumulation processes were identified, i.e., trapping in narrow channels, hitting building corners, and stagnation in front of buildings.

Driftwood Disaster Mitigation

For the simulation of driftwood capture devices installed on impervious dams as a driftwood disaster mitigation measure, we computed two driftwood capture experiments from a hydraulic model study published in the Journal of Erosion Control Engineering (Vol. 75, No. 4, pp. 14-24, 2022) at first. The simulations showed good agreement with the experimental results in terms of the uncaptured driftwood ratio, the shape of captured driftwood, the flow phenomenon, and the dammed-up depth, thus verifying the feasibility of the simulation code. We also obtained data that were difficult to measure in experiments, such as the movement of driftwood underwater and the porosity of driftwood accumulation areas. This highlighted the value of our simulation approach in designing more effective driftwood capture devices.

In simulations including 1,000 pieces of driftwood of multiple lengths interacting with a real-scale impervious dam with a capture device, the following findings were concluded:

(1) Driftwood capture efficiency is strongly influenced by driftwood length, and longer pieces are captured more easily. Even if the length is less than the spacing between

the adjacent poles of the trap device, driftwood with a length that is at least 3/4 of the spacing can be captured effectively.

(2) Driftwood captured by the trap device tends to submerge more easily when the length is shorter, resulting in a steeper angle between the captured driftwood and the water surface. However, when driftwood is too short, it is instead driven away by fast flow near the riverbed, leading to a shallower angle.

(3) For driftwood shorter than the spacing of the poles, higher driftwood number density increases the capture rate.

(4) The damming effect near the center of the capture device has a strong correlation with the degree of occlusion, and a higher degree of occlusion leads to a greater dammed-up depth.

(5) The damming effect near the side sections (weirs) is less sensitive to the state of the captured driftwood compared to the central area.

In summary, large-scale 3-D simulations of real river flooding with driftwood and the real-scale driftwood trap device installed on a concrete dam using GPU supercomputers demonstrate the capability of our numerical methods in simulating these scenarios with high accuracy. It is our deepest hope that this humble contribution to the prediction and mitigation of driftwood disasters will help reduce damage and better protect human lives.

Future Work

There remain several areas where the physical and numerical models can be further developed and refined to enhance the accuracy and applicability of the simulations. The following aspects should be addressed in future work.

- Incorporation of non-Newtonian fluid calculations: Real-world floods often contain large amounts of sediment, mud, and other particulate matter, which can change the rheological properties of the fluid and lead to non-Newtonian behavior. In high-sediment conditions, the floodwater may exhibit shear-thinning or Bingham plastic characteristics, depending on the sediment concentration and particle size distribution.

While my current work primarily focuses on Newtonian fluid assumptions, incorporating non-Newtonian fluid dynamics into the model is an important direction for future research.

- Inclusion of wind effects: The current study only calculates the velocity field in the liquid phase to maintain computational stability, while the influence of the velocity field in the gas phase is ignored. Thus, the inclusion of wind effects is crucial in future work, particularly in scenarios where atmospheric conditions play a dominant role in driftwood transportation, such as typhoons. Wind-induced surface stress can change driftwood trajectories, especially in wide floodplains and reservoirs.

- Introduction of solid fracture simulation: In the current research, all solid objects are treated as rigid bodies, which prevents the simulation of phenomena such as riverbed deformation and landslides caused by water flow, as well as structural damage to bridge piers and driftwood trap devices due to driftwood collisions. In the future, methods such as bonded DEM can be introduced to incorporate solid fracture modeling based on the current framework, making the simulation results more consistent with the real situation.

- Further enhancement of simulation efficiency: In the large-scale river flooding simulations, the presence of some small-scale debris was not considered due to the limitations of DEM and simulation efficiency. The small-scale debris may enhance the accumulation of driftwood near obstacles because it can create a more cohesive and stable structure with larger pieces. Future research could focus on enhancing the efficiency of simulations and exploring adaptive strategies for real-time disaster management.

- Improvement on modeling: In the simulations for driftwood trap devices, the geometry of the driftwood was relatively simple, and the effects of stones, sand, and other materials carried by the water had yet to be considered. Introducing more complex geometries of natural trees and investigating the effectiveness of devices in trapping debris flows containing driftwood will be crucial for future research.

6.2 Originality

This study significantly expands the application of the LBM method in the field of natural disaster research, particularly in the area of large-scale driftwood disasters. The introduction of the AMR method, GPU acceleration, and improvements in computational logic have enabled accurate and rapid simulations of large-scale driftwood disasters. Full-scale simulations can clarify issues that are difficult to verify in hydraulic model experiments, such as the actual underwater movement of driftwood and the porosity of driftwood accumulation areas. These issues help us gain a deeper understanding of driftwood accumulation processes during actual disasters and contribute to the more effective design of driftwood trap devices. My contributions are primarily reflected in the following aspects:

- Optimization for original code: Several optimizations were implemented for large-scale driftwood simulations based on the original code. The modifications included redefining the index of DEM particles on the computational grid to insert the riverbed into the calculation domain and optimizing the calculation logic between LBM and DEM to enhance the computational efficiency, as described in Section 4.2.1. In addition, new functions, such as realizing the random release of driftwood, were developed and integrated into the original code, which is discussed in Section 5.1.1.

- Clarification of the accumulation process of driftwood: To the best of our knowledge, we are the first team to use the LBM to simulate the real-time driftwood disaster that occurred in Iwaizumi Town. Our simulations successfully identified three different driftwood accumulation mechanisms: trapping in narrow channels, colliding with building corners, and stagnation in front of buildings. These findings enhance our understanding of driftwood accumulation processes.

- Contribution to the development of efficient driftwood trap devices: This study elucidated the detailed trapping process of driftwood capture devices with realistic geometries. We successfully found the factors that influence driftwood capture efficiency and dammed-up depth, which provide a theoretical foundation for the design of efficient driftwood capture devices.

Bibliography

[1] Hideo Matsutomi. Method for estimating collision force of driftwood accompanying tsunami inundation flow. *Journal of Disaster Research*, 4(6):435–440, 2009.

[2] Lukas Schmocker and Volker Weitbrecht. Driftwood: Risk analysis and engineering measures. *Journal of Hydraulic Engineering*, 139(7):683–695, 2013.

[3] Shima Joji and Yasutomi Kenichi. Driftwood trapping function of a driftwood trap installed upstream of concrete closed dam. *Journal of the Japan Society of Erosion Control Engineering*, 75(4):14–24, 2022.

[4] Robert M Terrill and Tracy Colgan. Some simple analytic solutions of the Navier-Stokes equations. *International Journal of Engineering Science*, 29(1):55–68, 1991.

[5] Roger Temam. *Navier-Stokes equations: theory and numerical analysis*, volume 343. American Mathematical Society, 2024.

[6] Charles R Doering and John D Gibbon. *Applied analysis of the Navier-Stokes equations*. Number 12. Cambridge university press, 1995.

[7] M Ragheb. Computational fluid dynamics. *mragheb Website, Available Online at https://mragheb. com/NPRE*, 20475, 1976.

[8] Yoshiharu Ishikawa, Takahisa Mizuyama, and Makoto Fukuzawa. Generation and flow mechanisms of floating logs associated with debris flow. *Journal of the Japan Society of Erosion Control Engineering*, 42(5):11–20, 1989.

[9] Noriyuki Minami, Takashi Yamada, and Doi Yasuhiro. Woody debris trapping by impermeable type sabo dam. *Journal of the Japan Society of Erosion Control Engineering*, 53(3):62–63, 2000.

[10] Daniele Bocchiola, MARIA CRISTINA Rulli, and Renzo Rosso. Transport of large woody debris in the presence of obstacles. *Geomorphology*, 76(1-2):166–178, 2006.

[11] Taka-aki Okamoto, Kenta Tanaka, Kazumasa Matsumoto, and Tomohiro Someya. Influence of velocity field on driftwood accumulation at a bridge with a single pier. *Environmental Fluid Mechanics*, 21(3):693–711, 2021.

[12] E Mignot, André Paquier, and S Haider. Modeling floods in a dense urban area using 2D shallow water equations. *Journal of Hydrology*, 327(1-2):186–199, 2006.

[13] Owen T Ransom and Bassam A Younis. Explicit gpu based second-order finite-difference modeling on a high resolution surface, Feather River, California. *Water Resources Management*, 30:261–277, 2016.

[14] Ting Zhang, Ping Feng, Čedo Maksimović, and Paul D Bates. Application of a three-dimensional unstructured-mesh finite-element flooding model and comparison with two-dimensional approaches. *Water Resources Management*, 30:823–841, 2016.

[15] Reza Marsooli and Weiming Wu. 3-D finite-volume model of dam-break flow over uneven beds based on VOF method. *Advances in Water Resources*, 70:104–117, 2014.

[16] Zhangping Wei, Robert A Dalrymple, Eugenio Rustico, Alexis Hérault, and Giuseppe Bilotta. Simulation of nearshore tsunami breaking by smoothed particle hydrodynamics method. *Journal of Waterway, Port, Coastal, and Ocean Engineering*, 142(4):05016001, 2016.

[17] Yoshihiko Shimizu and Kengo Osada. Numerical experiments on accumulation process of driftwoods around piers by using a DEM-flow coupling model. *Proceedings of Hydraulic Engineering*, 51:829–834, 2007.

[18] Hajime Shibuya, Satoshi Katsuki, Hisashi Ohsum, and Nobutaka Ishikawa. 3D-DEM Simulation on trap performance of driftwood capturing structure. *Journal of the Japan Society of Erosion Control Engineering*, 63(6):13–22, 2011.

[19] Peter A Cundall and Otto DL Strack. A discrete numerical model for granular assemblies. *Geotechnique*, 29(1):47–65, 1979.

[20] Ichiro Kimura, Taeun Kang, and Kazuo Kato. 3D–3D computations on submerged-driftwood motions in water flows with large wood density around driftwood capture facility. *Water*, 13(10):1406, 2021.

[21] Giancarlo Alfonsi. Reynolds-averaged Navier–Stokes equations for turbulence modeling. 2009.

[22] Sauro Succi. *The lattice Boltzmann equation: for fluid dynamics and beyond*. Oxford university press, 2001.

[23] Dieter A Wolf-Gladrow. *Lattice-gas cellular automata and lattice Boltzmann models: an introduction*. Springer, 2004.

[24] Cyrus K Aidun and Jonathan R Clausen. Lattice-Boltzmann method for complex flows. *Annual Review of Fluid Mechanics*, 42:439–472, 2010.

[25] Yos Panagaman Sitompul and Takayuki Aoki. A filtered cumulant lattice Boltzmann method for violent two-phase flows. *Journal of Computational Physics*, 390:93–120, 2019.

[26] Seiya Watanabe and Takayuki Aoki. Large-scale flow simulations using lattice Boltzmann method with AMR following free-surface on multiple GPUs. *Computer Physics Communications*, 264:107871, 2021.

[27] Ting Ye, Dingyi Pan, Can Huang, and Moubin Liu. Smoothed particle hydrodynamics (SPH) for complex fluid flows: Recent developments in methodology and applications. *Physics of Fluids*, 31(1), 2019.

[28] Yu Wang, Chang Shu, and LM Yang. An improved multiphase lattice Boltzmann flux solver for three-dimensional flows with large density ratio and high Reynolds number. *Journal of Computational Physics*, 302:41–58, 2015.

[29] Takaji Inamuro, Takaaki Yokoyama, Kentaro Tanaka, and Motoki Taniguchi. An improved lattice Boltzmann method for incompressible two-phase flows with large density differences. *Computers & Fluids*, 137:55–69, 2016.

[30] Abbas Fakhari and Diogo Bolster. Diffuse interface modeling of three-phase contact line dynamics on curved boundaries: A lattice Boltzmann model for large density and viscosity ratios. *Journal of Computational Physics*, 334:620–638, 2017.

[31] Carolin Körner, Michael Thies, Torsten Hofmann, Nils Thürey, and Ulrich Rüde. Lattice Boltzmann model for free surface flow for modeling foaming. *Journal of Statistical Physics*, 121(1):179–196, 2005.

[32] Cyril W Hirt and Billy D Nichols. Volume of fluid (VOF) method for the dynamics of free boundaries. *Journal of Computational Physics*, 39(1):201–225, 1981.

[33] Pao-Hsiung Chiu and Yan-Ting Lin. A conservative phase field method for solving incompressible two-phase flows. *Journal of Computational Physics*, 230(1):185–204, 2011.

[34] F Xiao, Y Honma, and T Kono. A simple algebraic interface capturing scheme using hyperbolic tangent function. *International Journal for Numerical Methods in Fluids*, 48(9):1023–1040, 2005.

[35] Mark Sussman, Peter Smereka, and Stanley Osher. A level set approach for computing solutions to incompressible two-phase flow. *Journal of Computational Physics*, 114(1):146–159, 1994.

[36] Marsha J Berger and Joseph Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of Computational Physics*, 53(3):484–512, 1984.

[37] Seiya Watanabe, Takayuki Aoki, and Tomohiro Takaki. A domain partitioning method using a multi-phase-field model for block-based AMR applications. *Parallel Computing*, 97:102647, 2020.

[38] Seiya Watanabe, Jun Kawahara, Takayuki Aoki, Kenta Sugihara, Shinsuke Takase, Shuji Moriguchi, and Hirotada Hashimoto. Free-surface flow simulations with floating objects using lattice Boltzmann method. *Engineering Applications of Computational Fluid Mechanics*, 17(1):2211143, 2023.

[39] Tongda Lian, Shintaro Matsushita, and Takayuki Aoki. An AMR-based liquid film simulation with surfactant transport using PLIC-HF method. *Applied Sciences*, 13(3):1955, 2023.

[40] Shintaro Matsushita and Takayuki Aoki. Gas-liquid two-phase flows simulation based on weakly compressible scheme with interface-adapted AMR method. *Journal of Computational Physics*, 445:110605, 2021.

[41] Pedro Valero-Lara. Leveraging the performance of LBM-HPC for large sizes on GPUs using ghost cells. In *International Conference on Algorithms and Architectures for Parallel Processing*, pages 417–430. Springer, 2016.

[42] Kai Yang and Takayuki Aoki. Multi-GPU scaling of a conservative weakly compressible solver for large-scale two-phase flow simulation. In *International Conference on Parallel and Distributed Computing: Applications and Technologies*, pages 16–27. Springer, 2022.

[43] Shiyi Chen, Hudong Chen, Daniel Martnez, and William Matthaeus. Lattice Boltzmann model for simulation of magnetohydrodynamics. *Physical Review Letters*, 67(27):3776, 1991.

[44] Martin Geier, Martin Schönherr, Andrea Pasquali, and Manfred Krafczyk. The cumulant lattice Boltzmann equation in three dimensions: Theory and validation. *Computers & Mathematics with Applications*, 70(4):507–547, 2015.

[45] Guy R McNamara and Gianluigi Zanetti. Use of the Boltzmann equation to simulate lattice-gas automata. *Physical Review Letters*, 61(20):2332, 1988.

[46] Prabhu Lal Bhatnagar, Eugene P Gross, and Max Krook. A model for collision processes in gases. I. Small amplitude processes in charged and neutral one-component systems. *Physical Review*, 94(3):511, 1954.

[47] Xiaoyi He and Li-Shi Luo. A priori derivation of the lattice Boltzmann equation. *Physical Review E*, 55(6):R6333, 1997.

[48] Xiaoyi He and Li-Shi Luo. Theory of the lattice Boltzmann method: From the Boltzmann equation to the lattice Boltzmann equation. *Physical Review E*, 56(6):6811, 1997.

[49] Yue-Hong Qian, Dominique d'Humières, and Pierre Lallemand. Lattice BGK models for Navier-Stokes equation. *EPL (Europhysics Letters)*, 17(6):479, 1992.

[50] Dominique d'Humieres. Multiple–relaxation–time lattice Boltzmann models in three dimensions. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 360(1792):437–451, 2002.

[51] Martin Geier, Andreas Greiner, and Jan G Korvink. Cascaded digital lattice Boltzmann automata for high Reynolds number flow. *Physical Review E*, 73(6):066705, 2006.

[52] Martin Geier, Stephan Lenz, Martin Schönherr, and Manfred Krafczyk. Under-resolved and large eddy simulations of a decaying taylor–green vortex with the cumulant lattice boltzmann method. *Theoretical and Computational Fluid Dynamics*, 35:169–208, 2021.

[53] Hiromichi Kobayashi. The subgrid-scale models based on coherent structures for rotating homogeneous turbulence and turbulent channel flow. *Physics of Fluids*, 17(4), 2005.

[54] Evan G Hemingway and Oliver M O'Reilly. Perspectives on euler angle singularities, gimbal lock, and the orthogonality of applied forces and applied moments. *Multibody System Dynamics*, 44(1):31–56, 2018.

[55] Nicolin Govender, Daniel N Wilke, Schalk Kok, and Rosanne Els. Development of a convex polyhedral discrete element simulation framework for NVIDIA Kepler based GPUs. *Journal of Computational and Applied Mathematics*, 270:386–400, 2014.

[56] Benjamin Nassauer and Meinhard Kuna. Contact forces of polyhedral particles in discrete element method. *Granular Matter*, 15(3):349–355, 2013.

[57] Darius Markauskas and Rimantas Kačianauskas. Investigation of rice grain flow by multi-sphere particle model with rolling resistance. *Granular Matter*, 13(2):143–148, 2011.

[58] Denis Gueyffier, Jie Li, Ali Nadim, Ruben Scardovelli, and Stéphane Zaleski. Volume-of-fluid interface tracking with smoothed surface stress methods for three-dimensional flows. *Journal of Computational Physics*, 152(2):423–456, 1999.

[59] Kensuke Yokoi. Efficient implementation of THINC scheme: a simple and practical smoothed VOF algorithm. *Journal of Computational Physics*, 226(2):1985–2002, 2007.

[60] Erika O Avila-Davila, Victor M Lopez-Hirata, and Maribel L Saucedo-Muñoz. Application of phase-field method to the analysis of phase decomposition of alloys. *Modeling and Simulation in Engineering Sciences*, page 221, 2016.

[61] John W Cahn and John E Hilliard. Free energy of a nonuniform system. I. Interfacial free energy. *The Journal of Chemical Physics*, 28(2):258–267, 1958.

[62] ZR Sibbing. Numerical methods for the implementation of the Cahn-Hilliard equation in one dimension and dynamic boundary condition in two dimensions. 2015.

[63] Abbas Fakhari, Martin Geier, and Taehun Lee. A mass-conserving lattice Boltzmann method with dynamic grid refinement for immiscible two-phase flows. *Journal of Computational Physics*, 315:434–457, 2016.

[64] Jeremiah U Brackbill, Douglas B Kothe, and Charles Zemach. A continuum method for modeling surface tension. *Journal of Computational Physics*, 100(2):335–354, 1992.

[65] Danping Peng, Barry Merriman, Stanley Osher, Hongkai Zhao, and Myungjoo Kang. A PDE-based fast local level set method. *Journal of Computational Physics*, 155(2):410–438, 1999.

[66] M'hamed Bouzidi, Mouaouia Firdaouss, and Pierre Lallemand. Momentum transfer of a Boltzmann-lattice fluid with boundaries. *Physics of fluids*, 13(11):3452–3459, 2001.

[67] Binghai Wen, Chaoying Zhang, Yusong Tu, Chunlei Wang, and Haiping Fang. Galilean invariant fluid-solid interfacial dynamics in lattice Boltzmann simulations. *Journal of Computational Physics*, 266:161–170, 2014.

[68] Hui Gao, Hui Li, and Lian-Ping Wang. Lattice Boltzmann simulation of turbulent flow laden with finite-size particles. *Computers & Mathematics with Applications*, 65(2):194–210, 2013.

[69] YT Feng, K Han, and DRJ Owen. Coupled lattice Boltzmann method and discrete element modelling of particle transport in turbulent fluid flows: Computational issues. *International Journal for Numerical Methods in Engineering*, 72(9):1111–1134, 2007.

[70] Tiankai Tu, David R O'Hallaron, and Omar Ghattas. Scalable parallel octree meshing for terascale applications. In *SC'05: Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, pages 4–4. IEEE, 2005.

[71] Moriguchi Shuji, Okawara Masafumi, and Kure Shiichi. Analysis of damage caused by 2016 typhoon no.10 in Iwate Prefecture - From viewpoints of geotechnical engineering and river engineering -. *Japanese Geotechnical Journal*, 13(2):149–158, 2017.