https://doi.org/10.48048/tis.2023.5539

Eulerian vs Lagrangian Method for Low Computational Resources: A Comparison of 2D Dam Break Case

Rida Siti Nuraini Mahmudah[1,*] and Koji Morita[2]

[1]Department of Physics Education, Universitas Negeri Yogyakarta, Yogyakarta, Indonesia
[2]Department of Quantum Physics and Nuclear Engineering, Kyushu University, Fukuoka, Japan

([*]Corresponding author's e-mail: rida@uny.ac.id)

Received: 8 August 2022, Revised: 16 August 2022, Accepted: 23 August 2022, Published: 16 March 2023

Abstract

It is well known that computational fluid dynamics has been challenging for low-computational-resourced students and researchers. In this study, we performed a 2D dam break simulation by Eulerian approach-finite volume method of OpenFOAM and Lagrangian approach-finite volume particle method (FVP) in a low-specification personal computer. We compared those approaches' results qualitatively and quantitatively against experimental data and measured their simulation time for 6 different grid sizes. We compared the visualization results and their pressure and velocity using ParaView and VisIt. At the same time, a quantitative comparison was made by determining the waterfront position each time using the Tracker video analyzer. It was found that OpenFOAM resulted in better visualization results, lower error by 28 - 68 %, and more reasonable pressure and velocity profiles. It can also simulate smaller grid sizes and 10 to 200 times faster than FVP, but it can't produce the air-liquid interface's sharpness as good as FVP. Thus, to simulate cases where interfaces are not important, Eulerian-based OpenFOAM is best suited to perform in a low-specification computer.

Keywords: Eulerian method, Lagrangian, Low computational resources, 2D dam break, OpenFOAM, Finite volume particle method

Introduction

Computational fluid dynamics (CFD) is one of the branches of physics that has essential roles and applications in many fields, such as urban physics [1], health [2], food industry [3], wastewater [4], nuclear reactor analysis [5], aeronautical industry [6] and many more. There are 2 kinds of approaches in computational fluid dynamics: Eulerian and Lagrangian. The former is a mesh method-the simulation domain is divided into meshes, while the latest is a meshless method-the simulation domain is represented with moving particles [7].

Despite its importance, CFD is well-known for being expensive. One must pay a significant amount of money to buy commercial CFD software [8]-mostly Eulerian-based, and need to have huge computational resources to get acceptable results-especially for Lagrangian-based. Therefore, for students and researchers with limited budgets, studying CFD becomes challenging whichever approaches they choose.

Luckily, in terms of software availability, OpenFOAM [9] offers a free CFD tool. Its solvers are based on finite volume methods [10], one of the Eulerian approaches. For Lagrangian, a lot of researchers have developed their code, such as smoothed particle hydrodynamics [11], moving-particle semi-implicit method [12], and finite volume particle (FVP) method [13]. Among them, FVP is being enhanced continuously [14] and has been used to simulate essential phenomena such as heat transfer with phase change [15].

Now that we can freely have the tool for Eulerian and Lagrangian, the remaining question is: Which approach is best suitable for a-low-computational-resourced students/researchers? While many studies have been performed using those approaches separately [16-18], as well as combined in 1 simulation [19-21], only one study so far compares these 2 [22]. They simulate free surface flow and fluid-solid interaction problems using smoothed particle hydrodynamics and finite element methods. Both methods were compared in terms of robustness, model setup easiness, and versatility. However, they performed each approach on different computational resources with relatively high specifications.

In this study, we compared the finite volume method of OpenFOAM and FVP in a low-specification personal computer: Ubuntu 20.04 LTS operating system, 4 GB RAM, Intel® CoreTM i7-7500 CPU @ 2.70GHz×4. Both methods were used to simulate an available dam break experimental case performed by another study. We analyzed both approaches regarding qualitative and quantitative results (compared to the experiment) and the simulation time. We highlighted the advantages and disadvantages of each so that a-low-budgeted students/researchers can choose the most appropriate approach for their needs.

Methods

In this study, we propose to determine fluid movements, i.e., positions, velocities, and pressures at each time. Navier-Stokes equations (Eqs. (1) - (3)), [23] are widely used to do this, both in Lagrangian and Eulerian methods.

$$\nabla \cdot \vec{u} = 0 \tag{1}$$

$$\frac{\partial \rho \vec{u}}{\partial t} + \nabla \cdot (\rho \vec{u}\vec{u}) = \nabla \cdot (\mu \nabla \vec{u} + \rho\tau) - \nabla p + \rho g + \sigma \kappa_\alpha \nabla \alpha \tag{2}$$

$$\frac{\partial \rho \varphi}{\partial t} + \nabla \cdot (\rho \varphi \vec{u}) = \nabla \cdot (D \nabla \varphi) + q_\varphi \tag{3}$$

Where t is the time, $\vec{u}$ is the velocity vector, ρ is the fluid density, μ is the dynamic viscosity, p is the pressure, g is the gravitational acceleration, φ is a scalar quantity, D is the diffusion coefficient, κ_α is the surface curvature, and σ is the surface tension. Eqs. (1) - (3) are the mass, momentum, and scalar quantities conservation, respectively. Unfortunately, those equations are difficult to be solved analytically, thus CFD exist to solved those numerically.

In the dam break simulation, there are 2 fluids involved: Water and air inside the dam. OpenFOAM uses the Volume of Fluids (VoF) method to simulate this by solving the following equation [24];

$$\frac{\partial \alpha}{\partial t} + \nabla \cdot (\alpha \vec{u}) + \nabla \cdot \left(\vec{u}_r \alpha (1 - \alpha)\right) = 0 \tag{4}$$

Where α is the volume fraction of fluids ($\alpha = 1$ for liquid, and $\alpha = 0$ for air), $\vec{u}_r$ is the relative velocity, $\vec{u}_r = \vec{u}_l - \vec{u}_g$, defined as velocity difference of 2 phases (subscripts l for liquid and g for gas). By solving Eq. (4), we can get the phase of each grid in every time steps. The density and viscosity of all grids are then updated based on their resulting α by using Eqs. (5) - (6), respectively.

$$\rho = \alpha \rho_l + (1 - \alpha)\rho_g \tag{5}$$

$$\mu = \alpha \mu_l + (1 - \alpha)\mu_g \tag{6}$$

Phase differentiation by VoF is essential to determine both phases' surface and interface. Meanwhile, in FVP, we can simulate dam break only by simulating the water and the surrounding wall representing the dam. It doesn't need special treatment to capture the interface, as it can be tracked automatically. This is one of the advantages of FVP and other Lagrangian methods.

To solve those governing equations, each approach uses a different discretization method. OpenFOAM uses finite volume discretization methods to discretize Eqs. (1) - (4), which describe completely in [25,26]. In FVP, Eqs. (1) - (3) are solved by assigning each simulation domain as a particle possessing its physical properties. The methods then track down the particle's position, and its physical properties change. Detailed information about FVP can be found in our previous studies [27,28].

In this study, we use the PISO algorithm for FVP dan OpenFOAM. PISO algorithm is well known for its accuracy in solving Navier-Stokes equations due to its predictor-corrector step. Application of the PISO algorithm in OpenFOAM can be found in [10,26], while in FVP, it can be found in the previous studies we mentioned above [27,28]. **Table 1** summarizes both approaches in detail used in this study.

Table 1 Lagrangian and eulerian methods summary.

	Eulerian	**Lagrangian**
Discretization method	Finite volume method	Finite volume particle method
Algorithms	PISO	PISO
Programming language	C++ (OpenFOAM)	Fortran
Solver/compiler	interFoam	gfortran
Modelled phase	Liquid and air	Only liquid
Interface tracking method	Volume of Fluid	None (done automatically)
Results visualization	ParaView [29]	VisIt [30]

We simulate the dam break experimented by [31]. In 2D, their initial experiment setup is depicted in **Figure 1**. The blue-coloured box is the water, the black line is the dam wall, and the white is the air. The physical properties used in this study for FVP and OpenFOAM are listed in **Table 2**.

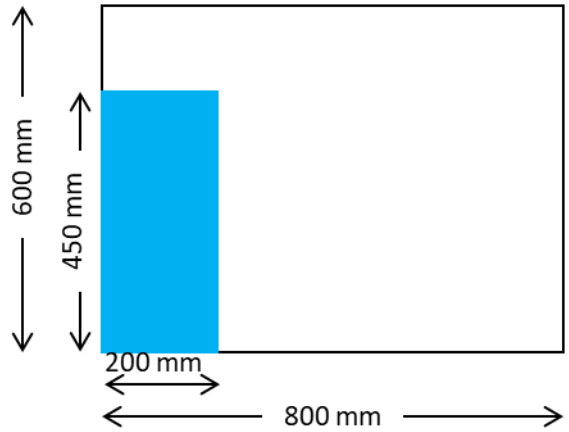


Figure 1 Dam break case geometry.

Table 2 Fluid properties.

	Density (kg/m^3)	**Kinematic viscosity (m^2/s)**
Water	1,000	10^{-6}
Air	1	10^{-5}

We simulated both approaches with different grid sizes to investigate our computer capability, from 1 cm to 0.5 mm. Here, we use a homogenous grid in the horizontal and vertical axis, i.e., grid size $= \Delta x = \Delta y$. The number of mesh and/or particle involved in this simulation for each grid size is shown in **Table 3**. The mesh number in OpenFOAM is more than 4 times larger than FVP because it simulates both water and air phases, as mentioned above. The finer the grid, the larger the mesh and/or particle number for the same-sized geometry.

Table 3 Mesh and particle numbers used in the simulation.

Grid size	FVP (particle number)	OpenFOAM (mesh number)
1 cm	1,180	4,800
5 mm	4,180	19,200
2 mm	23,900	120,000
1 mm	92,800	480,000
0.8 mm	-	750,000
0.5 mm	-	1,920,000

Results from both methods were then compared to the experiment results qualitatively-by comparing visual results of simulations and experiment pictures and quantitatively by determining the waterfront position each time. To do this, the simulation results were saved as videos by ParaView dan VisIt for OpenFOAM and FVP, respectively. Then, we analyzed those videos using Tracker [32]. A Screenshot of this process is shown in **Figure 2**. First, we load the simulation video into the Tracker. Then, we set the calibration stick and point to be the same size and position as the dam break geometry. After that, we determined the waterfront mass to be tracked, and Tracker will do the rest.

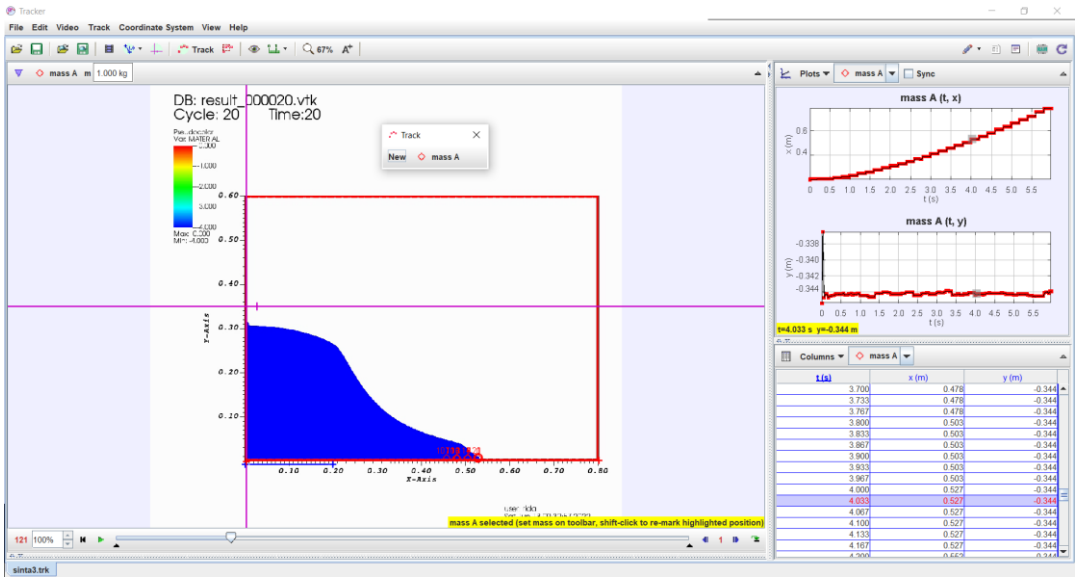


Figure 2 Waterfront analysis by Tracker.

Data of waterfront positions are recorded automatically by Tracker in the right-bottom corner of **Figure 2**. We copied this data, compared it with the experiment's waterfront, and calculated their mean absolute percentage error (MAPE).

$$MAPE(\%) = \frac{100}{N}\sum_{i=1}^{N}\left|\frac{\widehat{x_i}-x_i}{\widehat{x_i}}\right| \quad (7)$$

Where $\hat{x}_i$ is the waterfront of simulation results at the i-th time step, x_i is the waterfront of simulation results of the same time step, and N is the amount of data. We also analyze simulation results' velocity and pressure profile to understand how both approaches behave.

Lastly, we measured the simulation time for each case. In FVP, we checked the time of the first and last simulation results and then subtracted it. While in the OpenFOAM, we made a log file when we ran the solver by typing this command: InterFoam > log & in the terminal. After the simulation, we can extract the run time in the bottom line of the log file.

Results and discussion

Geometry building

FVP method has a different architecture with OpenFOAM. In FVP, we set the geometry, phases, physical properties, time step and boundary condition setting, and the discretization and algorithms in 1 fortran90 file. Thus, the geometry building is done simultaneously with running the simulation in 1 command. Meanwhile, in the OpenFOAM®, we must build the whole calculation domain first, then set some parts of the domain as water and others as air. The flowchart of this process is described in **Figure 3**. It takes 2 commands to build the geometry and another command to run the solver.

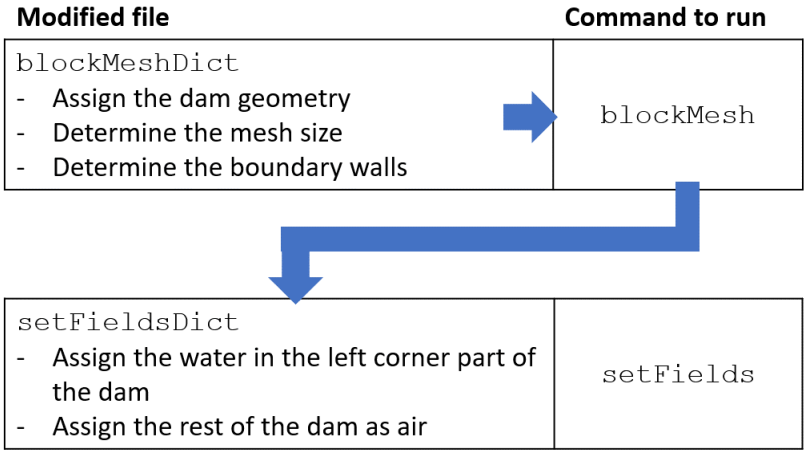


Figure 3 Running flowchart of OpenFOAM.

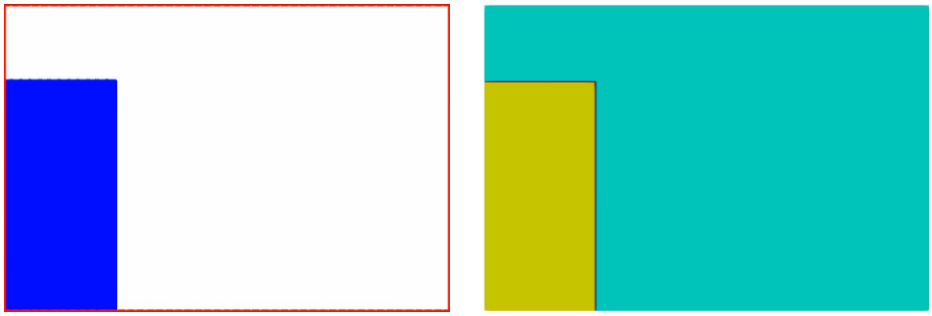


Figure 4 Simulation geometry (a) FVP geometry (blue = water, red = walls) and (b) OpenFOAM geometry (yellow = water, green = air).

Geometry results for both methods are shown in **Figure 4**. Particles in FVP are actualy sphere-shaped. Depending on the grid size, it will appear as tiny "balls" building up a specific geometry. The finer the grid, the "balls" will also be tinier; they almost look like a dot. For comparison purposes, we set the "balls" size in the visualization tool VisIt such that the water part in **Figure 4**. (a) seems like a unity-not contain thousands of tiny "balls".

Grid size analysis

In most CFD simulations, finer mesh generally results in more minor errors. It is often desirable to perform a simulation with the smaller mesh possible, even though it costs more computational resources. In this study, we investigated the smallest mesh that our modest personal computer can handle. We simulated FVP and OpenFOAM simulations using different grids, i.e., 1 cm, 5, 2, 1, 0.8 and 0.5 mm.

Surprisingly, the computer can simulate all the grid sizes with OpenFOAM (up to 1.9 million of mesh) but failed to simulate a 1 mm grid size of FVP. The error message was: "Cannot allocate memory. Allocation would exceed memory limit", meaning that the computer has not enough capability to run smaller than 2 mm for Lagrangian methods-despite the number of particles in FVP being way smaller than

OpenFOAM (**Table 3**). This is due to the working principles of the FVP and other Lagrangian methods, that its interaction with neighboring particles determines a particle's movement and physical properties changes. On each time step, its neighboring particle could change-due to the particle's movement; thus, it must be determined every time. That is why it needs enormous computational resources that our computer cannot provide for a 1 mm grid size.

Since 2 mm is the smallest grid our computer can simulate for both methods, we compare its results and their respective experiment results [31] in **Figure 5**.

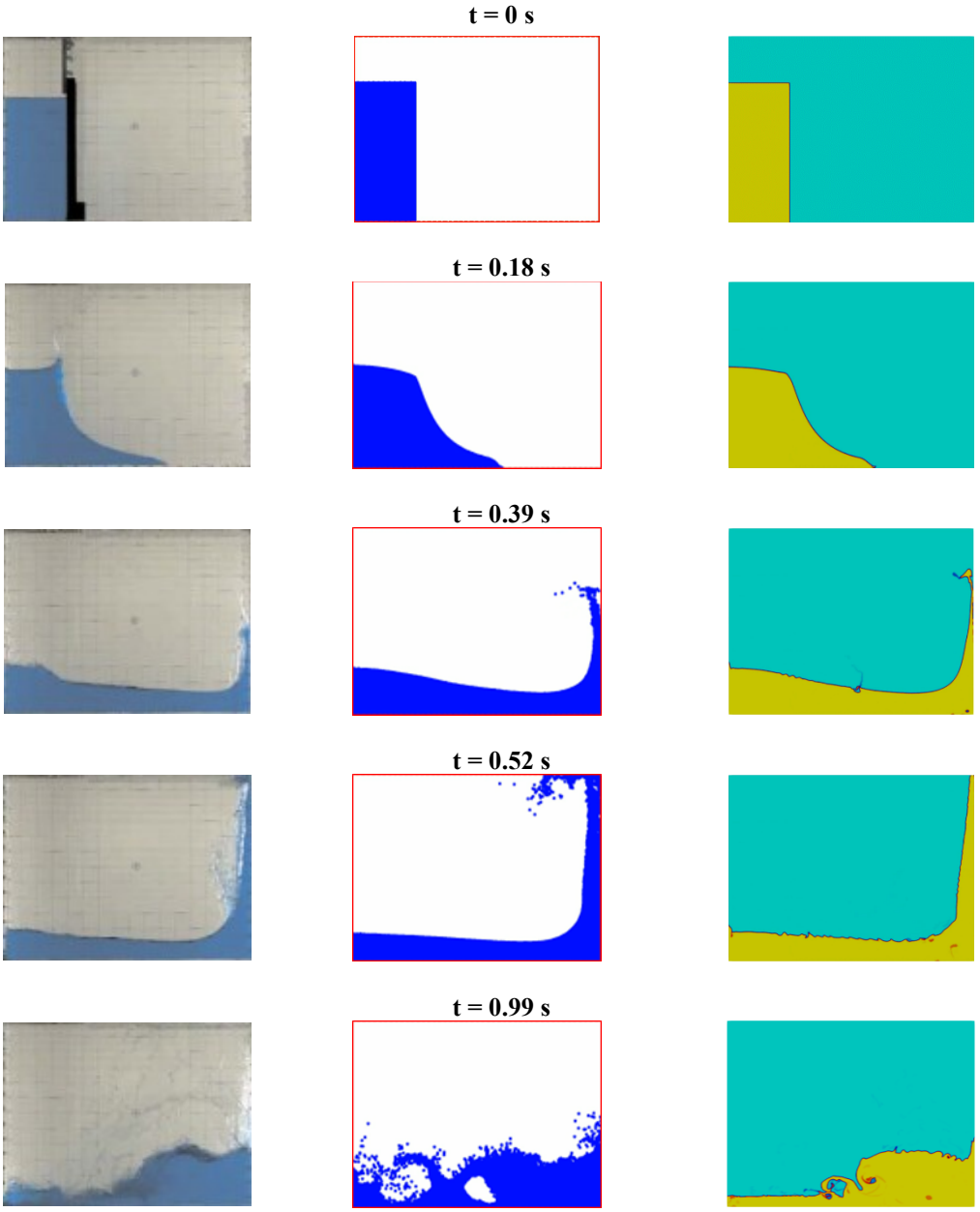


Figure 5 Visualization comparison between experiment (left), FVP (middle) and OpenFOAM (right).

In the dam break experiments, water was held together by a wall (black line in the top left of **Figure 5**) and released from the top of the tank at $t = 0$. Water was then breaking from the bottom and moving along the tank's floor until it reached the right wall of the tank. Depending on the velocity and the pressure of the water, it will move along the right wall as high as it can reach. Then, the gravity will "pull" it back down and collide with other water in the bottom of the tank, resulting in a wave-like bouncing phenomenon.

Generally, both methods can produce the same sequence behavior but differ in detail. Both methods can accurately produce the same visual results with experiments approximately until $t = 0.30$ s before the water reaches the right wall. From $t = 0.39$ s, both approaches seem to overestimate the water movements, where they go further/higher than that of the experiment's visual results. Regarding the visual comparison, the water shape of OpenFOAM in every time step is more similar to the experiment results than that of FVP. However, we observed small oscillations in the bottom water-air interface of OpenFOAM results at $t > 0.39$ s. These oscillations do not exist in the experiments and simulation results by FVP and can be considered as "flaws" of the interFoam solver. A study by [33] discussed these flaws and gave several considerations to eliminate those.

As for the interface, in the FVP methods, we can see splashing phenomena, where "balls" of water move separately from its bulk. This confirms one of the advantages of Lagrangian methods, i.e., it can clearly differentiate between 2 separate phases and interface without any special treatment. In the OpenFOAM results, this splashing is not being reproduced. Instead, we can always see "diffusion" between the water-air interface, shown as a blue-ish line that always exists between the yellow and green bulk of the phase. This line is even bolder in a larger grid size. An example of the interface of both methods with a grid size of 1 cm is shown in **Figure 6**.

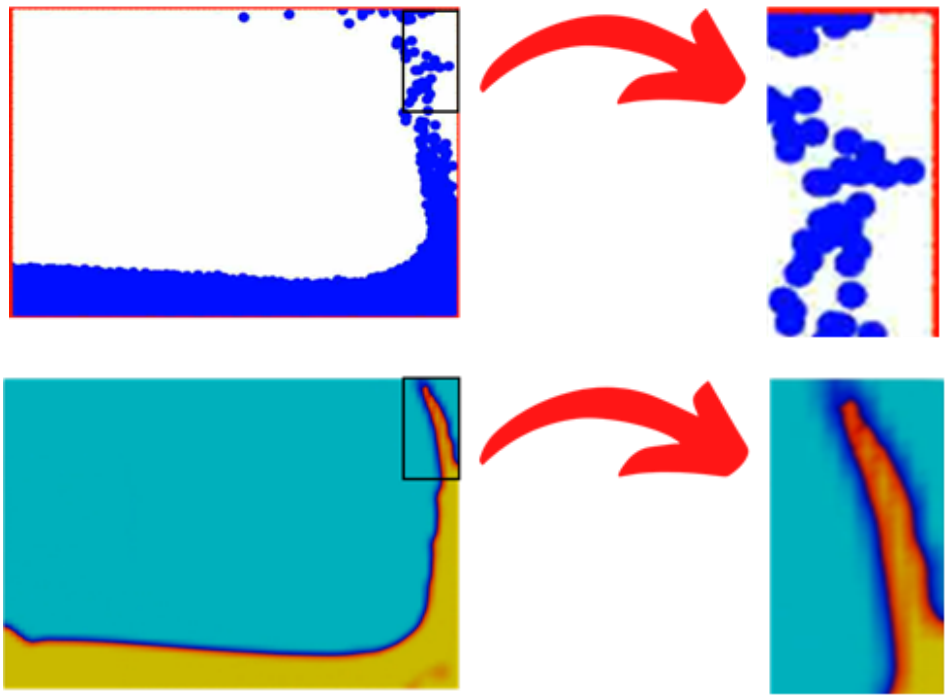


Figure 6 Interface with a grid size of 1 cm: Sharp interface in FVP (top) and diffused interface in OpenFOAM (bottom).

As can be seen from **Figure 6**, FVP results produce a clear blue-white interface, as the representation of water in the air. The bigger the grid size only affect the size of the "balls" and the smoothness of the water interface. Meanwhile, in the OpenFOAM, the bigger grid size the bigger the blue-ish line. Due to the strong relationship between the diffusion with grid size, we can minimize the diffusion effect by decreasing it, with the consequences of additional computational resources. **Figure 7** shows that the diffusion line is reduced considerably by decreasing the grid size from 1 cm to 1 mm, but still visible. The line become almost invisible with a grid size of 0.5 mm.

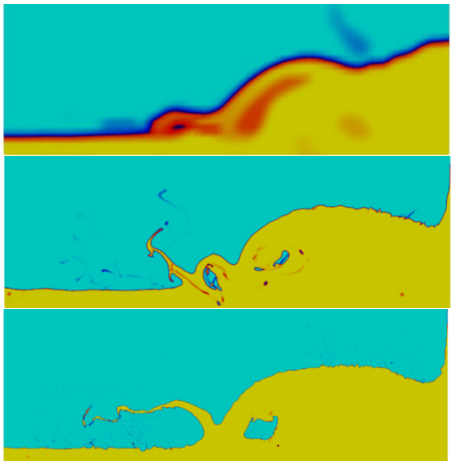
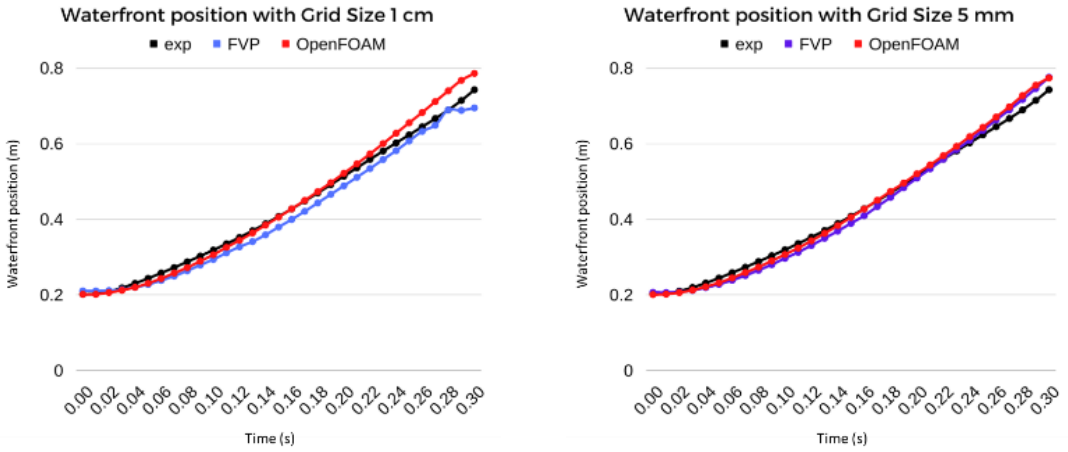


Figure 7 "Diffusion" line with different grid size: 1 cm (top), 1 mm (middle), and 0.5 mm (bottom).

To further compare both simulation results with the experiment quantitatively, we plotted the waterfront position of each time. We analyze the waterfronts of simulation results using Tracker software, as mentioned in the methods section, while the experiment waterfront's data are extracted from the graph by the same study [31]. As the waterfront can be measured only before the water reaches the tank's right wall, we only plot the waterfront position until t = 0.3 s. The comparable results of grid sizes in the experiment, FVP, and OpenFOAM are shown in **Figure 8**. **Figure 9** shows the waterfront position of the smaller grids with OpenFOAM, while **Figure 10** shows the respective MAPE for all grid sizes. It is worth mentioning that the ParaView failed to load all the 0.5 mm simulation results due to the memory shortage for this grid size, thus the waterfront analysis was done manually by extracting the results of each time step, combining those to a video, and then analyzed it with the Tracker.



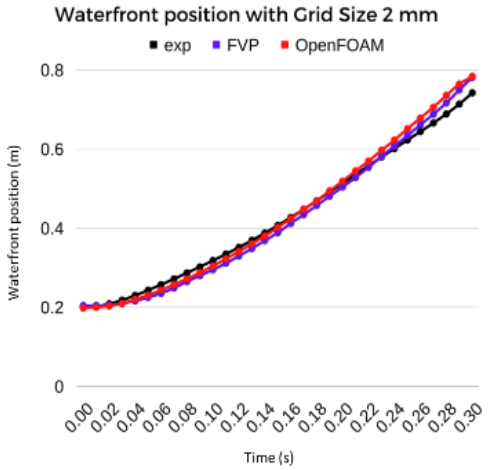


Figure 8 Waterfront comparison with different grid sizes.

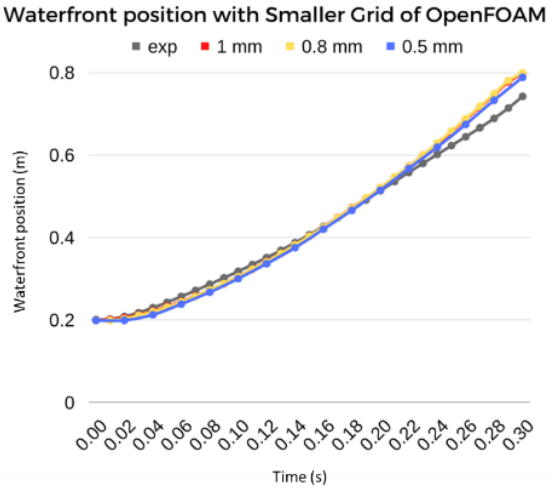


Figure 9 Waterfront position for smaller grid of OpenFOAM.

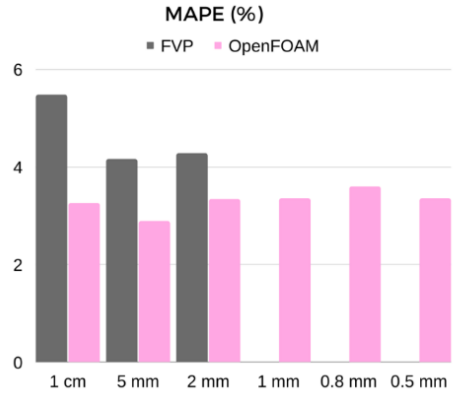


Figure 10 MAPE of waterfront's position.

Figure 8 confirms slight differences in waterfront between experiment and simulation results of both methods that are not shown in qualitative comparison (**Figure 5**) for all grid sizes. Even before the water reaches the right wall of the dam, the simulation results overestimate the experiments. This difference can be due to the water and the dam's wall friction in the experiment that is not being incorporated in the simulations. The friction will hamper the water's movement, resulting in smaller velocity and smaller waterfront position changes each time.

In **Figure 9**, we can see that the overestimate still exist even with the smaller grid size of OpenFOAM. Nonetheless, all the differences between experiment and simulation results are considered to be small, as can be seen from the MAPE chart in **Figure 10**.

Regarding waterfront position, FVP methods show the most significant error with a grid size of 1 cm. However, OpenFOAM results seem not dependent on the grid size; its MAPE ranged from 2.89 - 3.35 %, all of which are lower than the FVP method. OpenFOAM obtains the lowest MAPE with a 5 mm grid size-but again, visually, this grid size still produces noticeable diffusion at the interface. It is worth mentioning that the experiment was 3D, so 2D simplification might also affect the simulation results. In general, both simulations can produce the waterfront position with MAPE of less than 6 %. Lewis [34] interprets MAPE values lower than 10 % as highly accurate in the forecasting world.

Pressure and velocity analysis

Apart from the water-dam's wall friction of the experiment, the overestimate in waterfront position must have been related to the pressure and velocity of the water. Hence, we plot both pressure and velocities in several time steps for both simulation methods. Unfortunately, there are no experimental data available for these. Thus, we only compared those simulation results. For brevity, we show the selected time step of significant interest.

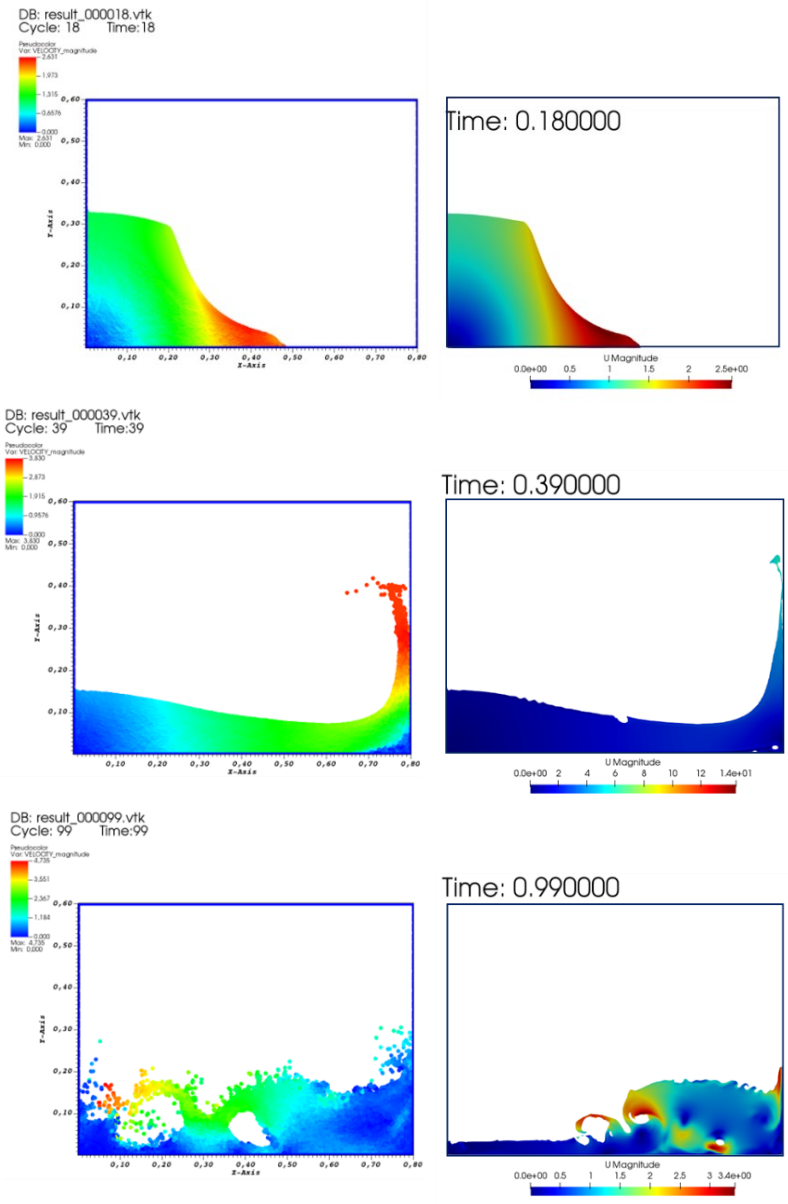
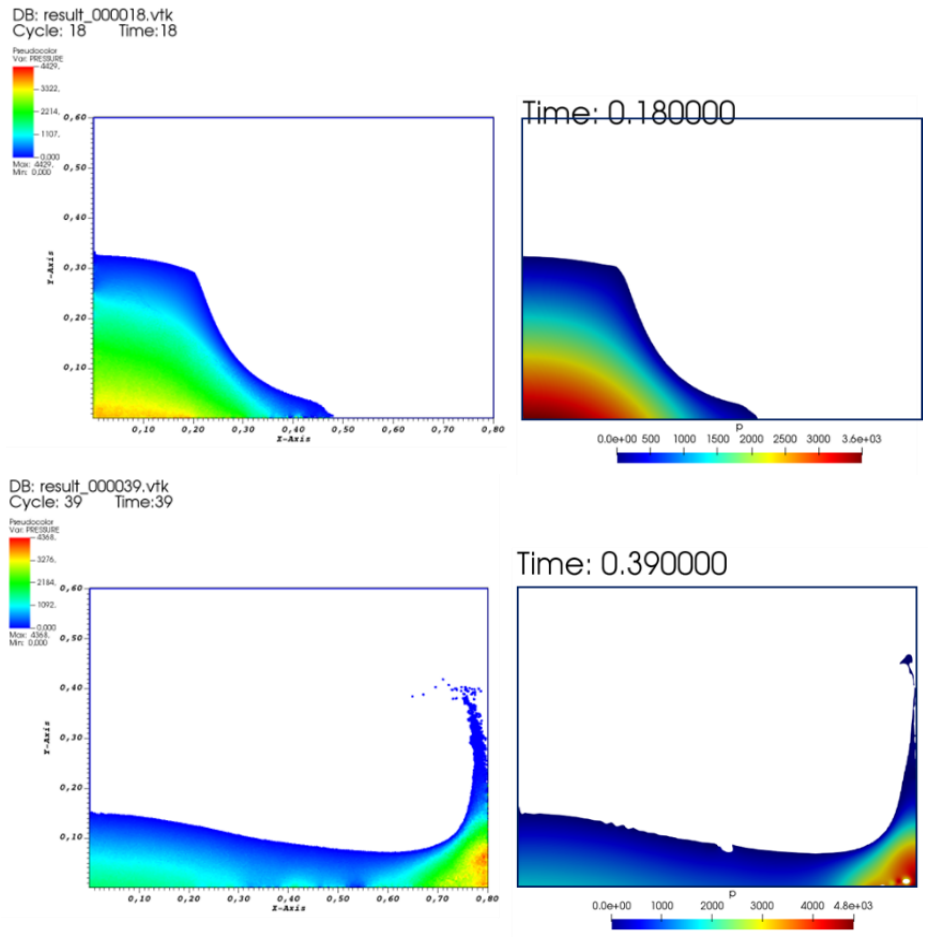


Figure 11 Velocity profile of FVP (left) and OpenFOAM (right).

Figure 11 shows velocity profile of simulation results at t = 0.18 s (top), t = 0.39 s (middle) and t = 0.99 s (bottom). Each color range represents magnitude, defined by a color bar-from blue (lowest) to red (biggest), at the top left for FVP and at the center bottom for OpenFOAM. At t = 18 s, both methods show a good agreement in velocity contour. Water at the bottom left has the lowest velocity (still not moving), and the water at the waterfronts has the highest velocity with a magnitude of 2.6 m/s in FVP and 2.5 m/s in OpenFOAM. Both methods show a relatively big difference after the water hit the right wall of the tank, i.e., t > 0.35 s. In all time steps, velocities in FVP are larger than that of OpenFOAM, shown by a larger green-yellow-red colored area. With larger velocity, water in the FVP methods moves faster-its "front" almost reaches the left wall at t = 0.99 s, while in the experiments and OpenFOAM results, it is just halfway to the left wall in that time step.

Similarly, we can see the same behavior in pressure (**Figure 12**). As opposed to velocities (**Figure 11**), the highest pressure is in the bottom part of the water. Until t = 0.18 s, FVP shows larger maximum pressure than the OpenFOAM, but only a small number of particles have that large pressure-showed by less "red" compared to OpenFOAM. After that, OpenFOAM consistently has more considerable pressure than FVP. In the OpenFOAM results, the pressure seems to be proportionally distributed, with a large portion being 0 at the upper part of the water. In general, in terms of pressure, OpenFOAM results seem more reasonable.



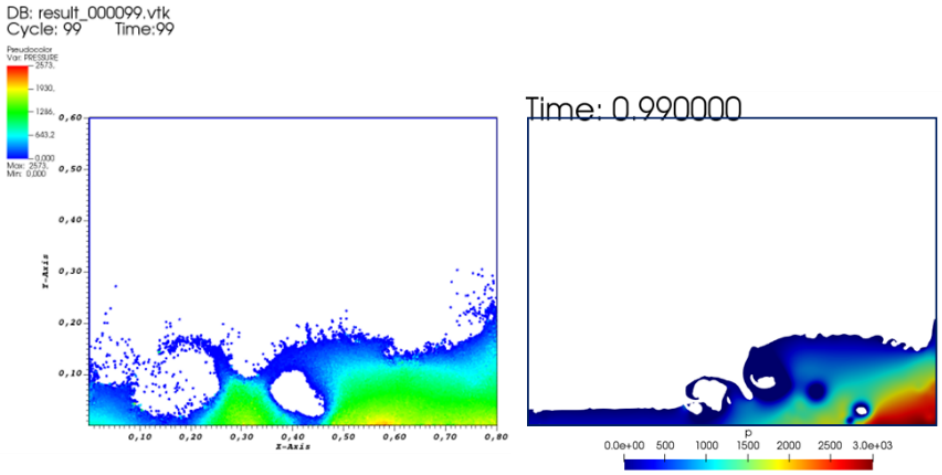


Figure 12 Pressure profile of FVP (left) and OpenFOAM (right).

CPU time

Finally, we compare the running time of both methods in **Table 4**. As expected, the finer grid needs more CPU time, and OpenFOAM simulation time was way faster than FVP simulations-as can be expected for Eulerian methods. For larger grid sizes, OpenFOAM can be 200 times faster than FVP. The finer the mesh, the ratio decreases. Unfortunately, the computer cannot perform a 1 mm grid size of FVP, so we cannot see the CPU time comparison for this case. Meanwhile, OpenFOAM can successfully simulate the 0.5 mm grid size with approximately 2 weeks of running time and failed to be performed when the grid size was decreased to 0.4 mm. Thus, the 0.5 mm is the smallest grid size our computer can handle.

Table 4 Running time comparison (in seconds).

Grid size	FVP	OpenFOAM	OpenFOAM ratio
1 cm	2,460	12	$\frac{1}{205}$
5 mm	8,160	135	$\approx \frac{1}{60}$
2 mm	42,420	4,359	$\approx \frac{1}{10}$
1 mm	-	78,029	NA
0.8 mm	-	186,702	NA
0.5 mm	-	1,194,268	NA

Conclusions

In this study, we compare the Lagrangian and Eulerian approaches: Finite volume particle (FVP) method and OpenFOAM-finite volume method-based solver in simulating dam break cases with limited computational resources. We performed the simulations with 6 different grid sizes, ranging from 1 cm to 0.5 mm. The computer can simulate up to 2 mm grid size for FVP method, and 0.5 mm for OpenFOAM. From visualization results, error calculation of MAPE, pressure and velocity analysis, and CPU time, it can be concluded that OpenFOAM is more robust and effective to be used by a single computer. OpenFOAM was faster, lower in MAPE, and more reasonable in pressure and velocity profile, as well as the water visualization. But, if I prefers the interface area's sharpness over other things, then the FVP method is the best choice. Although to get smaller MAPE results of FVP, it is strongly recommended to use a more sophisticated computational resource.

References

[1] B Blocken. Computational Fluid dynamics for urban physics: Importance, scales, possibilities, limitations and ten tips and tricks towards accurate and reliable simulations. *Build. Environ.* 2015; **91**, 219-245.

[2] S Peng, Q Chen and E Liu. The role of computational fluid dynamics tools on investigation of pathogen transmission: Prevention and control. *Sci. Total Environ.* 2020; **746**, 142090.

[3] B Xia and DW Sun. Applications of computational fluid dynamics (cfd) in the food industry: A review. *Comput. Electron. Agr.* 2002; **34**, 5-24.

[4] E Wicklein, DJ Batstone, J Ducoste, J Laurent, A Griborio, J Wicks, S Saunders, R Samstag, O Potier and I Nopens. Good modelling practice in applying computational fluid dynamics for WWTP modelling. *Water Sci. Tech.* 2016; **73**, 969-82.

[5] M Wang, Y Wang, W Tian, S Qiu and GH Su. Recent progress of CFD applications in PWR thermal hydraulics study and future directions. *Ann. Nucl. Energ.* 2021; **150**, 107836.

[6] A Bahatmaka, DJ Kim and Y Zhang. Verification of CFD method for meshing analysis on the propeller performance with OpenFOAM. *In*: Proceedings of the 2018 International Conference on Computing, Electronics & Communications Engineering, Southend. 2018, p. 302-6.

[7] DJ Benson. Computational methods in lagrangian and eulerian hydrocodes. *Comput. Meth. Appl. Mech. Eng.* 1992; **99**, 235-394.

[8] P Kopyt and W Gwarek. A comparison of commercial CFD software capable of coupling to external electromagnetic software for modeling of microwave heating process. *In*: Proceedings of the 6th Seminar Computer Modeling & Microwave Power Engineering, Texas. 2004, p. 33-9.

[9] OpenFOAM, Available at: https://openfoam.org, accessed July 2022.

[10] HK Versteeg and W Malalasekera. *An introduction to computational fluid dynamics*. Longman, London, 1995.

[11] JJ Monaghan. Smoothed particle hydrodynamics. *Rep. Progr. Phys.* 2005; **68**, 1703.

[12] S Koshizuka and Y Oka. Moving-particle semi-implicit method for fragmentation of incompressible fluid. *Nucl. Sci. Eng.* 2017; **123**, 421-34.

[13] S Hibi, K Yabushita and T Tsutsui. Study on incompressible fluid analysis by three-dimensional particle method with finite volume techniques. *EPJ Web Conferences* 2022; **269**, 01020.

[14] X Liu, R Ogawa, M Kato, K Morita and S Zhang. Accuracy and stability enhancements in the incompressible finite-volume-particle method for multiphase flow simulations. *Comput. Phys. Comm.* 2018; **230**, 59-69.

[15] T Zhang, K Funakoshi, X Liu, W Liu, K Morita and K Kamiyama. Numerical simulation of heat transfer behavior in EAGLE ID1 in-pile test using finite volume particle method. *Ann. Nucl. Energ.* 2021; **150**, 107856.

[16] Q Zhang, J Zhang and Z Sun. Optimal convergence rate of the explicit Euler method for convection-diffusion equations. *Appl. Math. Lett.* 2022; **131**, 108048.

[17] M Babanezhad, AT Nakhjiri, M Rezakazemi and S Shirazian. Developing intelligent algorithm as a machine learning overview over the big data generated by euler-euler method to simulate bubble column reactor hydrodynamics. *ACS Omega* 2020; **5**, 20558-66.

[18] L Bertolotti, R Jefferson-Loveday, S Ambrose and E Korsukova. A comparison of VOF and Euler-Euler approaches in CFD modelling of two-phase flows with a sharp interface. *In*: Proceedings of the ASME Turbo Expo 2020: Turbomachinery Technical Conference and Exposition, London. 2020.

[19] EJ Ching, SR Brill, M Barnhardt and M Ihme. A two-way coupled Euler-Lagrange method for simulating multiphase flows with discontinuous Galerkin schemes on arbitrary curved elements. *J. Comput. Phys.* 2020; **405**, 109096.

[20] A Legay, J Chessa and T Belytschko. An eulerian-lagrangian method for fluid-structure interaction based on level sets. *Comput. Meth. Appl. Mech. Eng.* 2006; **195**, 2070-87.

[21] CW Hirt, AA Amsden and JL Cook. An arbitrary lagrangian-eulerian computing method for all flow speeds. *J. Comput. Phys.* 1974; **14**, 227-53.

[22] M Rakhsha, CE Kees and D Negrut. Lagrangian vs. Eulerian: An analysis of two solution methods for free-surface flows and fluid solid interaction problems. *Fluids* 2021; **6**, 460.

[23] PE Rodríguez-Ocampo, M Ring, JV Hernández-Fontes, JC Alcérreca-Huerta, E Mendoza and R Silva. CFD simulations of multiphase flows: Interaction of miscible liquids with different temperatures. *Water* 2020; **12**, 2581.

[24] E Sánchez-Cordero, M Gómez and E Bladé. Three-dimensional numerical analysis of a dam-break using OpenFOAM. *Proc. Inst. Syst. Program. RAS* 2017; **29**, 311-20.

[25] R Eymard, T Gallouët and RE Herbin. *Finite volume methods*. HAL open science, Lyon, France, 2019, p. 1-236.
[26] X Su, J Zhang, MY Zhao and H Zhang. Numerical and Experimental Investigations on the Hydrodynamic Performance of a Tidal Current Turbine. *IOP Conf. Mater. Sci. Eng.* 2017; 280, 012001.
[27] L Guo, Y Kawano, S Zhang, T Suzuki, K Morita and K Fukuda. Numerical simulation of rheological behavior in melting metal using finite volume particle method. *J. Nucl. Sci. Tech.* 2010; **47**, 1011-22.
[28] RS Mahmudah, M Kumabe, T Suzuki, L Guo and K Morita. 3D simulation of solid-melt mixture flow with melt solidification using a finite volume particle method. *J. Nucl. Sci. Tech.* 2011; **48**, 1300-12.
[29] ParaView, Available at: https://www.paraview.org/, accessed July 2022.
[30] VisIt, Available at: https://wci.llnl.gov/simulation/computer-codes/visit, accessed July 2022.
[31] C Hu and M Sueyoshi. Numerical simulation and experiment on dam break problem. *J. Mar. Sci. Appl.* 2010; **9**, 109-14.
[32] Tracker Video Analysis and Modeling Tool for Physics Education, Available at: https://physlets.org/tracker, accessed Jun 2022.
[33] BE Larsen, DR Fuhrman and J Roenby. Performance of interFoam on the simulation of progressive waves. *Coast. Eng. J.* 2019; **61**, 380-400.
[34] CD Lewis. *Industrial and business forecasting methods*. Butterworth-Heinemann, Oxford, 1982.