Computational Science and Engineering
(International Master's Program)

Technische Universität München

Master's Thesis

A flexible approach to 2D-3D coupling of a Shallow-Water Equation solver to OpenFOAM

Francisco Javier Espinosa Pelaez

Computational Science and Engineering
(International Master's Program)

Technische Universität München

Master's Thesis

A flexible approach to 2D-3D coupling of a Shallow-Water Equation solver to OpenFOAM

Author:	Francisco Javier Espinosa Pelaez
1st examiner:	Univ.-Prof. Dr. Hans-Joachim Bungartz
Assistant advisor:	M.Sc. Gerasimos Chourdakis
Submission Date:	September 30th, 2020

I hereby declare that this thesis is entirely the result of my own work except where otherwise indicated. I have only used the resources given in the list of references.

September 30th, 2020 Francisco Javier Espinosa Pelaez

Acknowledgments

First of all, I would like to express my gratitude to my advisor, Gerasimos, for his constant and invaluable support throughout the realization of my thesis. His suggestions, his patience, our late discussions, and his constant interest kept my motivation high at all times, making this process a very enriching experience. I also want to thank the members at the TUM Scientific Computing in Computer Science Chair, for expressing their interest and for their support during my thesis.

My master's program was possible thanks to the partnership between Mexico and Germany, through the CONACYT-DAAD (Consejo Nacional de Ciencia y Tecnologa - Deutsche Akademischer Austauschdienst) program. During my master's program, CONACYT-DAAD provided me with full financial support.

Lastly, as an international student being far away from home, I want to deeply thank my family. In spite of the distance and the rough times, they constantly supported me and were always looking after my academic progress. I'm eternally grateful to my grandmother who, until the last moment, was always very close to me and was proud that I had the opportunity to do my graduate studies abroad. Finally, I give special thanks to my family of old and new friends; without them, my master's program would have been only that.

Abstract

Simulating 3D waves crashing against large 3D solid structures, such as ship hulls, piers, or large coastlines in the case of a tsunami, require a large number of cells even for a low-accuracy simulation. Traditional methods using the Navier-Stokes equations with a high number of cells require considerable amount of computational resources, and are, often, unfeasible. The 2D Shallow Water Equation (SWE) model delivers satisfactory simulations for the same scenarios, using a much lower number of cells, and therefore, at a lower cost. However, as a 2D model, it fails to resolve the 3D effects (e.g. near obstacles). This leads us to think of a solver in which we could combine the low cost of SWE with the capability of the Navier Stokes Equations to capture 3D effects.

Previous research [14] proposed a solver in which the domain is partitioned into 2D and 3D subdomains solved by OpenFOAM and coupled using the same framework to construct the global solution. In this thesis, we build upon this idea and develop a similar environment, moving the SWE computations outside of the OpenFOAM framework to a solver developed at the TUM SCCS [17], written in C++. On the 3D side, we continue to use *interFoam* as the driving solver. We implement the coupling of the solvers using the preCICE library by developing a new preCICE adapter for the SWE solver and by extending the preCICE adapter for OpenFOAM.

This thesis builds the foundations of 2D-3D fluid-fluid simulations using preCICE, focusing on breaking-dam and open-channel-flow scenarios as validation cases. We compare partitioned to monolithic simulations, and observe qualitatively good results, with the error depending on the direction of the coupling, which in turn depends on the characterization of the flow (supercritical or subcritical).

Contents

1. Introduction

1.1. Introduction

Multi-physics problems have been a research field of great interest in the past decade, offering numerous visions, formulations and approaches. The development of tools for multi-physics simulations aims to bring these simulations into a simpler implementation for the user, without compromising computational resources and precision. In the literature we can find a spectrum of approaches for multi-physics simulations, ranging from fully partitioned to fully monolithic simulations. A literature review can be found in the dissertation[21] of Benjamin Uekermann. On the one hand, the monolithic approach aims to solve a coupled system of equations on a single solver, and on the other hand, the partitioned approach aims to decouple the system of equations into their single-physics properties and equations, to an extent of data exchange between specialized software for each of the physical domains of a multi-physics application.

The partitioned approach allows us to treat each domain of the multi-physics problem independently; therefore, it is possible to have independent simplifications, assumptions and even solution methods for each domain with minimum invasion on the remaining domain.

In regard to fluid dynamics applications, Florian Mintgen's dissertation[14] takes advantage of the partitioned approach by reducing (computational) complexity on solutions for environmental free-surface flows by coupling a simple, yet general, 2-D model to a more exact, yet expensive and more complicated, 3-D model within critical sections across the whole domain. Continuing with this idea, this thesis will seek to extend such a partitioned approach for coupling a free-surface flow solution for specifically aimed solvers between a 2D and 3D domain. Furthermore it will serve as an investigation and collaboration line to *preCICE* as the multi-physics open-source coupling tool.

1.2. Free-Surface Flow

Free-surface flow refers to the flow of liquids open to the atmosphere or in partially filled conduits, and is characterized by the presence of a liquid-gas interface called the *free-surface*. Most natural free-sufcace flows include flow around ships, floods, river flows, aqueducts, hydroelectric dam spillways, interaction of waves with piers, soil erosion, etc. Human-made free-surface flow systems include irrigation systems, sewer lines, drainage ditches, and gutters, and the design of such systems is an important application area of engineering. [4].

Simulations for 3D free-surface flows are characterized not only by their degree of complexity for its derivation and implementation, but they also demand considerable computational power for large domains. The are also subjected to a number of considerations, that dictate how the flow should be treated. Among these considerations we find:

- domain's spatial dimensions
- interaction with irregularities upstream and downstream, such as obstacles along the flow or narrowing of the flume's edges
- interaction with external forces
- energy exchange
- sea-floor conditions, etc.

In general, to the interaction of the fluid with the surroundings is referred as *boundary conditions*, and in some cases they could be unknown.

The mathematical model that describes 3D free-surface flows, or any fluid motion, is constituted by the *Navier-Stokes equations*, a set of 3D, non-linear, partial differential equations (PDEs). By the complex nature of the 3-dimensional PDE system, an analytical solution is not available, so domain decomposition and space and time discretization methods constitute an alternative numerical solution. The accuracy of the numerical solution relies on the order of the discretization scheme, and more significantly, on the number of cell/nodes/elements of the mesh , which grows exponentially with the number of dimensions. In the case of large 3D domain scales, the computational complexity would make it infeasible to obtain a numerical solution.

1.3. Shallow Water Equations

Aiming to decrease the computational complexity concerning the number of nodes, the *Shallow Water Equations (SWE)* model reduces the 3D Navier-Stokes equations to a 2 dimensional set of equations. The SWE are a set of partial differential equations, derived from the physical conservation laws for mass and momentum, that provide an accurate solution under the assumption that the longitudinal dimensions are much greater than the vertical dimension. The model detaches the calculation of the vertical velocity from the equations system, and focuses on the longitudinal velocities instead, resulting in a 2D equations system. Mintgen provides a description for the properties, characteristics and limitations of SWE in a detailed manner, some of which will be further addressed and extended for a deeper analysis in Section 2.2. For now, it suffices to remark that the aforementioned assumption about longitudinal and vertical scales will not be valid for flow across critical sections (such as irregularities or obstacles on the stream), and in this case, the 3D Navier-Stokes equations would need to be solved at their full extension for obtaining meaningful simulation results.

1.4. Motivation for this thesis

Being aware of the limitations and advantages of free-surface flow simulations under both the *3D-Navier Stokes Equations* model and the *Shallow Water Equations* model, it follows to explore the possibility of coupling both approaches, so the simulation for larger 3D free-surface flows becomes feasible.

1.4.1. Previous work

In his dissertation [14], Florian Mintgen developed *shallowInterFoam* as a free-surface flow solver, that makes use of the advantages of *SWE* and the *Reynolds Averaged Navier Stokes*[1] *(RANS)* models in a partitioned fashion. The solver's implementation uses the 2D SWE model, where the spatial assumptions for SWE hold and therefore allowing minimal computational demand, and exchanges the calculated variables, usually the pressure and velocity, with the 3D RANS solver when the assumption of SWE model does not hold anymore. Such a Fluid-Fluid (FF) coupling scheme[2] used in *shallowInterFoam* has been developed in OpenFOAM (Open-source Field Operation And Manipulation), a numerical solver toolbox developed mainly for Computational Fluid Dynamics (CFD) applications, further discussed in Section 3.3 .

The implementation of *shallowInterFoam* uses its own coupling techniques, making the solver efficient, accurate and robust under the OpenFOAM framework with a Finite Volume Method discretization scheme and with unstructured meshes. This means, however, a lack of flexibility for exporting this particular Fluid-Fluid coupling scheme for free-surface flows to different solvers, toolboxes, engines, or even new implementation techniques outside from the OpenFOAM framework.

The work done by Mintgen serves as a good reference, from the mathematical and conceptualization point of view, to implement such a coupling scheme to different frameworks, other than OpenFOAM, offering the 2D - 3D free-surface solution more extension and flexibility.

[1]Navier-Stokes equations for modelling turbulent flows.

[2]In contrast to the usually used terms of Conjugate Heat Transfer (CHT) or Fluid Structure Interaction (FSI), where temperature and deformation are the variables of interest respectively in a multi-physics problem, Fluid-Fluid coupling refers usually to the pressure and velocity as the variables of interest.

In order to be able to use solvers from different frameworks as a partitioned approach for Fluid-Fluid coupling, a third-party coupling tool between the solvers ought to be implemented. Such is the case of the work [7] done by Christian Osse, where the *shallow-InterFoam* solver was de-coupled into their base-solvers, *shallowFoam* and *interFoam*, both under the OpenFOAM framework . In this case, the coupling is carried out externally with *preCICE*, a multi-physics coupling library developed at the Technical University of Munich, the University of Stuttgart and the Eindhoven University of Technology. A further implementation[6] of preCICE as the driving coupling tool for Fluid-Fluid applications is done by Chourdakis et al., where flow from 3D and 1D domains were coupled together, also under the OpenFOAM framework.

1.4.2. Objectives of this thesis

The previously mentioned works demonstrate the feasibility of coupling Fluid-Fluid applications from different dimensional domains. They also show the feasibility of using *preCICE* as the driving coupler, which has proven its usefulness under minimum invasion implementation and showing its flexibility. The aforementioned implementations are, however, under the OpenFOAM framework, and so far there is no implementation that has been carried out on a different framework. Therefore, this thesis will take one step further and show that Fluid-Fluid coupling across different dimensions using *preCICE* as the driving coupling tool can be carried out in different frameworks, other than OpenFOAM solely, hence providing a more flexible approach.

The implementation of this thesis will rely mostly on some of the formulations and results found in Mintgen's dissertation, and will focus on coupling the 2-dimensional SWE model with the 3-dimensional Navier-Stokes equations model. In case of the 3D domain, it makes sense to continue using OpenFOAM's free-surface 3D solver, *interFoam*, since it has proven accuracy and robustness as will be seen in Section 3.3. For the 2D domain case, an SWE solver written in C++ will be used. More about the 2D SWE solver will be mentioned in Section 3.2

Using preCICE as the driving coupling tool for this topic has considerable benefits. To this date, preCICE has ready-to-use adapters for OpenFOAM, a high-level API for C++, and it is rapidly increasing not only its bindings to different languages, but also its capability for addressing diverse multi-physics problems, thus making it a more flexible viable tool for this research field. Therefore, choosing preCICE as the driving coupler for Fluid-Fluid coupling will contribute considerably to the development of *preCICE*, and consequently to the multi-physics research field. A formal introduction to preCICE is presented in Section 3.1 .

2. Theory

This chapter will introduce the theoretical foundations for obtaining solutions for surface flow, previously introduced in Section 1.2 . The chapter is divided into two main sections: Section 2.1 , introducing the Navier Stokes equations, the foundation basis for fluid mechanics, and Section 2.2 , that presents a basic derivation of the Shallow Water Equations model from the Navier-Stokes equations. For the former, a detailed description of the terms of the equations is presented, together with the description of the classification of the fluids that shall be used for this work, and lastly a description of the method used for solving such system of equations. For the latter, the specific steps and assumptions that hold the SWE model as valid will be described, including a description of the solving method.

2.1. Navier-Stokes and 3D free-surface flows

The 3D Navier-Stokes equations are mainly conformed by the mass conservation equation,

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \tag{2.1}$$

and the momentum conservation equations,

$$\frac{\partial (\rho \mathbf{u})}{\partial t} + \nabla \cdot (\rho \mathbf{u}\mathbf{u}) = -\nabla p + \mu \nabla^2 \mathbf{u} + \frac{1}{3}\mu \nabla (\nabla \cdot \mathbf{u}) + \rho \mathbf{g} \tag{2.2}$$

where $\mathbf{u}$ is the vector velocities, p is the pressure, ρ is the density, ∇ is the gradient operator, Δ is the Laplacian operator, μ dynamic viscosity and $\mathbf{g}$ represents the gravity as a volume force only in the z direction. In this work, an incompressible Newtonian fluid, and a transient and laminar flow will be considered. Thus, Equation 2.1 simplifies to

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0 \tag{2.3}$$

and Equation 2.2 simplifies to

$$
\begin{aligned}
&\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} + \frac{1}{\rho}\frac{\partial p}{\partial x} = \nu\left[\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right] \\
&\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + w\frac{\partial v}{\partial z} + \frac{1}{\rho}\frac{\partial p}{\partial y} = \nu\left[\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial z^2}\right] \\
&\frac{\partial w}{\partial t} + u\frac{\partial w}{\partial x} + v\frac{\partial w}{\partial y} + w\frac{\partial w}{\partial z} + \frac{1}{\rho}\frac{\partial p}{\partial z} = \nu\left[\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} + \frac{\partial^2 w}{\partial z^2}\right] + g
\end{aligned}
\tag{2.4}
$$

where u v w the velocity in the x y z directions respectively and $\nu = \mu/\rho$ is the kinematic viscosity.

2.1.1. Navier-Stokes Free-Surface Multi-phase Solver

The Navier-Stokes equations model all kinds of fluid motion, and for every kind there is a method that is derived based the most remarkable features of the flow type we deal with. In the case of free-surface flow, the **Volume-of-Fluid Method** (VOF) is chosen for this thesis, and corresponds to a *volume tracking method*. Further references to different approaches for solving free-surface flows, and also different types of flow, can be found at Mintgen's dissertation [14].

The Volume of Fluid (VOF) Method was presented by Hirt & Nichols [10], and started a new trend in multi-phase flow simulation. It relies on the definition of an indicator function, that allows us to know whether the cell is occupied by one fluid or another, or a mix of both. Let us consider a function f that can be defined as having a value of 1 at any cell that is occupied by a fluid, and 0 otherwise. The average value of f would then represent the fractional volume of the cell occupied by a fluid. Particularly, a value of 1 would indicate a cell full of fluid, and a value of 0 would mean no fluid on that cell at all. Therefore, values ranging from 0 to 1 shall represent a free-surface [19]. Such function f is referred to as a volume indicator function α, representing the volume fraction between two immiscible fluids, *Fluid A* and *Fluid B*, and can take the following values

$$
\alpha(\mathbf{x}, t) = \begin{cases} 0, & \text{Fluid A} \\ 1, & \text{Fluid B} \\ 0 < \alpha < 1, & \text{at the interface between Fluid A and Fluid B} \end{cases}
$$

Fluid A and Fluid B are typically water(liquid) and air(gas) respectively, hence the solver is considered to be a *multi-phase* solver. The VOF method includes a transport equation for calculating the volume indicator α, expressed as

$$
\frac{\partial \alpha}{\partial t} + \nabla \cdot (\alpha \mathbf{u}) + \nabla \cdot (\alpha(1-\alpha)\mathbf{u_r}) = 0
$$

where the third term is considered to be an artificial compression term that acts only in the interface ($0 < \alpha < 1$), counteracting the diffusion of α and leading to a well defined representation of the interface. The compression velocity u_r acts perpendicular to the interface, and is obtained by multiplying the velocity magnitude $|u|$ at the interface with the normal vector n^* of the interface

$$u_r = C_\alpha n^* |u|$$

where C_α is a compression coefficient that allows adjustment of the compression magnitude [14]. The remaining step is coupling the value of α to the momentum conservation, Equation 2.1 . This is done through the next constitutive equations for the density and the dynamic viscosity as a function of the volume indicator α as

$$\rho(\alpha) = \alpha\rho_l + (1-\alpha)\rho_g \tag{2.5}$$

$$\nu(\alpha) = \alpha\nu_l + (1-\alpha)\nu_g \tag{2.6}$$

where the subscripts l and g refer to liquid and gas respectively.

2.2. Shallow Water Equations in 2D free-surface flows

As mentioned in Section 1.3 , the Shallow Water Equations (SEW) model aims to reduce 3D Navier-Stokes equations to a 2D system of equations model by following the derivation discussed in this section in this section.

The SWE are a set of PDEs that describe flow problems, and are derived from the physical conservation laws for mass and momentum. The SWE hold valid for problems in which vertical dynamics can be neglected, compared with horizontal effects. The SWE also constitute a 2D model that is derived from considering ratios between spatial dimensions, depth averaging techniques and special boundary conditions, all of which will be shown in Section 2.2.1.

Some of the most common applications of SWE include:

- Tsunamis prediction
- Atmospheric flows
- Storm surges
- Flow around structures

On the other hand, the SWE model would not hold as valid when

- 3D effects become essential, such as rotational flow
- viscous effects become considerable
- waves become either too big or too small

2.2.1. Shallow Water Equation derivation - Hyperbolic PDEs

This section shows the derivation of the SWE from the 3D Navier-Stokes equations. The constitutive equations of the SWE model form a system of Hyperbolic partial differential equations, and as such they provide an expression for calculating the flow velocity in the form of waves (also referred to as *celerity*) that will have an influence on the open channel flow regime, which is essential for the implementation of this work.

We outline the most representative steps and assumptions from the full derivation [8] [15] of the SWE model. In the full derivation, an incompressible Newtonian fluid and a transient turbulent flow is considered, in addition to Coriolis forces and sea-bed frictional and wind frictional effects. For this thesis, however, a simpler model shall be addressed, as the SWE solver that will be implemented (Section 3.2) does not take into account viscous effects, turbulence nor Coriolis forces effects. Therefore, an inviscid incompressible fluid under a transient and laminar flow will be considered. From these assumptions Equation 2.1 simplifies to

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0 \tag{2.7}$$

and Equation 2.2 simplifies to

$$\begin{aligned} \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} &= -\frac{1}{\rho}\frac{\partial p}{\partial x} \\ \frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + w\frac{\partial v}{\partial z} &= -\frac{1}{\rho}\frac{\partial p}{\partial y} \\ \frac{\partial w}{\partial t} + u\frac{\partial w}{\partial x} + v\frac{\partial w}{\partial y} + w\frac{\partial w}{\partial z} &= -\frac{1}{\rho}\frac{\partial p}{\partial z} + g \end{aligned} \tag{2.8}$$

Equation 2.7 can be approximated as $\frac{U}{L} + \frac{V}{L} + \frac{W}{H} \approx 0$, where U, V, W are the averaged horizontal and vertical velocities, and L, H are the averaged horizontal and vertical dimensions. We approximate that $U \approx V$, hence obtaining

$$\frac{2U}{L} + \frac{W}{H} \approx 0$$

The underlying assumption for SWE model is considering that the horizontal dimensions, x and y, are much greater than the vertical dimension, z thus $L \gg H$. From these assumptions it then follows

$$W \approx -2U\frac{H}{L}$$

so W becomes negligible. This assumption allows us to vanish the terms that include the velocity w from Equation 2.8 as follows

$$\begin{aligned}\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} &= -\frac{1}{\rho}\frac{\partial p}{\partial x} \\ \frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} &= -\frac{1}{\rho}\frac{\partial p}{\partial y} \\ 0 &= -\frac{1}{\rho}\frac{\partial p}{\partial z} + g\end{aligned} \tag{2.9}$$

The z component of the momentum conservation equation provides a direct solution for the pressure p as a function of the water height z, including sea floor elevation b (Figure 2.1), and the gravity g. This expression leads to the fundamental formulation of the hydrostatic pressure. Moreover, the vertical velocity component w is not considered anymore, consequently a 3D equation system becomes a 2D equations system.

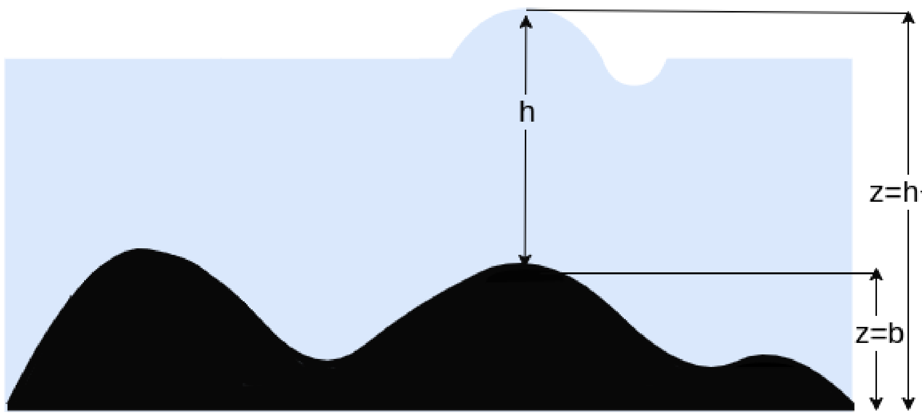


Figure 2.1.: Shallow water equations variables. Black surface refers to the sea floor elevation.

Based on the previous derivation, we observe that the Shallow Water Equations model holds only where horizontal dimensions are much greater than the vertical dimensions. An appropriate relation for holding the SWE model as valid is given by $H/L = 0.05$ [20].

Mass Conservation Equation Depth-Averaging

Integrating Equation 2.7 over the depth, following the Leibniz Theorem, the Fundamental Rule of Integration, the kinematic boundary conditions at the free surface (see [15]), and assuming that there is no flow across the normal direction of the sea floor, the mass continuity equation results in

$$\int_b^{b+h} \left(\frac{\partial hu}{\partial x} + \frac{\partial hv}{\partial x} + \frac{\partial w}{\partial z}\right)\partial z = \frac{\partial hu}{\partial x} + \frac{\partial hv}{\partial y} + \frac{\partial h}{\partial t} = 0 \tag{2.10}$$

Two new variables hu and hv result from depth-integrating the continuity equation, and they are referred to as *discharges* on the x and y directions respectively, since it measures the flow rate of water past a point. Equation 2.10 is referred as the depth-integrated continuity equation and it shows that the difference between the flow into and out of a volume-control of water happens with a change of the water depth.

Pressure Gradient Depth-Averaging

From Equations 2.9, a differential solution for the pressure was obtained. By integration from z to H, the following expression for the pressure results as

$$p = \rho g(H - z) \tag{2.11}$$

where $H = b + h$ (see Figure 2.1). From the RHS of Equation 2.9, the pressure gradient shall be obtained. For the x component, it follows as

$$\frac{\partial\, p}{\partial x} = \frac{\partial\, \rho g(H - z)}{\partial x}$$

Performing a depth-averaged integration on the RHS of the obtained expression, and by means of the chain rule we get

$$\begin{aligned}\int_b^H -\frac{1}{\rho}\frac{\partial}{\partial x}\rho g(H - z)\,\partial z &= -g\frac{\partial\, H}{\partial x}(H - b)\\ &= -g\frac{\partial\, b}{\partial x} - g\frac{\partial\, h}{\partial x}h\\ &= -g\frac{\partial\, b}{\partial x} - g\frac{\partial}{\partial x}\left(\frac{1}{2}h^2\right)\end{aligned} \tag{2.12}$$

The similar process is followed for the y component. Equation 2.12 shows that the pressure gradient is independent of the variable z, and further on the momentum equation it will be noted that such gradient is responsible for the horizontal accelerations.

Momentum Conservation Equation Depth-Averaging

The modification for the momentum conservation is similar as for the mass continuity equation, as previously seen. Once more depth-averaged integrating Equation 2.9, and substituting Equation 2.12, it yields

$$\frac{\partial\, hu}{\partial t} + \frac{\partial}{\partial x}\left(hu^2 + \frac{1}{2}gh^2\right) + \frac{\partial\, huv}{\partial y} = -gh\frac{\partial\, b}{\partial x} \tag{2.13}$$

$$\frac{\partial\, hv}{\partial t} + \frac{\partial\, huv}{\partial x} + \frac{\partial}{\partial y}\left(hv^2 + \frac{1}{2}gh^2\right) = -gh\frac{\partial\, b}{\partial y} \tag{2.14}$$

where b is the sea floor elevation, also known as *bathymetry*. The previous equations show that the momentum is dependent on the height of the flow, plus some source terms from the bathymetry. It can be found in literature[15] a more detailed derivation for these expressions.

Hyperbolic PDEs System

Hyperbolic PDEs are known to model conservative laws (also denoted as the advection equation) in the form of

$$q_t + f(q)_x = 0 \tag{2.15}$$

for 1-dimensional problems. The derivation of the equations from the previous section constitute the Shallow Water Equations model, and they represent a conservative law. Equations 2.10, 2.13 and 2.14 can be organized in a matrix form as in

$$\begin{bmatrix} h \\ hu \\ hv \end{bmatrix}_t + \begin{bmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \\ huv \end{bmatrix}_x + \begin{bmatrix} hv \\ huv \\ hv^2 + \frac{1}{2}gh^2 \end{bmatrix}_y = \begin{bmatrix} 0 \\ -gh\, b_x \\ -gh\, b_y \end{bmatrix} \tag{2.16}$$

resembling the conservation form from Equation 2.2.1 with one extra dimension in the form of

$$q_t + f_x + g_y = s \tag{2.17}$$

Note the subscripts that indicate partial derivatives. We can observe a non-linear dependence of the state variables h, hu, hv on the fluxes f and g.
Non-linear hyperbolic systems are typical of producing shockwaves and rarefaction waves, which under the nature of the SWE, shockwaves are rather denoted as *hydraulic jumps*, discussed in Section 2.4. For a better insight of the behaviour and properties of the system, the quasi-linear form is prefered. This means representing the equations system from Equations 2.16 and 2.17 in the form of $q_t + Aq_x + Bq_y = 0$, where $A = f'(q)$ and $B = g'(q)$ are known as *flux jacobians*.
Calculating the eigenvalues of A and B, we obtain

$$\lambda^{x1} = u - c, \qquad \lambda^{x2} = u, \qquad \lambda^{x3} = u + c$$

and

$$\lambda^{y1} = v - c, \qquad \lambda^{y2} = v, \qquad \lambda^{y3} = v + c$$

respectively. Both sets of eigenvalues are real numbers, in accordance to the definition of hyperbolic PDEs, and they represent the propagation speed of the waves that compose the system given by the eigenvectors of matrices A and B. The variable c in this case is referred to as the celerity, which is the speed of the gravity waves, and is defined as $c = \sqrt{gh}$. For further reference on previous derivation refer to literature [12].

It is worth mentioning that Equation 2.16 includes source terms on the RHS denoted by b. These source terms describe the sea floor elevation. In this thesis, however, such source terms will be omitted, and consequently omitted as well in the derivation of the solving methods for the SWE model in the following section.

2.3. Solving Methods for Shallow Water Equations

This section provides an introduction and derivation of the solving methods used for the Shallow Water Equations model as a hyperbolic PDE system. The content is mainly taken from the literature from Leveque et al. [1] [12], and only the most fundamental concepts for a proper understanding of the problem are mentioned. Should specific details on the derivation want to be revised, refer to the literature.

The Shallow Water Equations represent a conservation model that is constructed from the basis of the Euler Equations (mass, momentum, and energy conservation). As such, it has its foundation on the integral form of conservation laws, namely

$$\frac{d}{dt}\int_{x_1}^{x_2} q(x,t)dx = f(q(x_1,t)) - f(q(x_1,t)) \tag{2.18}$$

Equation 2.18 states that total quantity q between any two points can change only due to the flux past the endpoints, plus some source and sink terms (not shown) which will not be considered for this thesis. Classical solutions for decoupled equations in a differential form are well known and easily derived, provided that the flux functions $f(q)$ are smooth. Coupled and non-linear conservation laws, however, are typical of shockwaves (compression) and rarefaction (expansion) waves, hence the smoothness is lost and one must resort to numerical methods.

2.3.1. Finite Volume Methods

Discontinuities lead to computational difficulties. Classical finite difference methods, in which derivatives are approximated by finite differences, can be expected to break down near discontinuities in the solution where the differential equation does not hold. Finite Volume Methods (FVM) are based on the integral form of conservative laws(Equation 2.18) instead of differential forms (in the case of finite differences). Rather than pointwise approximations at grid points, the domain is broken into grid cells and the total integral of q is approximated over the cell average of q, which is this integral divided by the volume of the cell. These values are modified in each time step by the flux through the edges of the grid cells, and the primary focus is to determine good numerical flux functions that approximate the correct fluxes reasonably well, based on the approximate cell averages. [12]. Essentially, a finite volume method evaluates exact expressions for the average value of the solution over some cell volume, and it uses this data to construct approximations of the solution within cells. After averaged integration of Equation 2.18, the FVM is obtained in the form yields

$$Q_i^{n+1} = Q_i + \frac{\Delta t}{\Delta x}\left[F_{i+1/2}^n - F_{i-1/2}^n\right] \tag{2.19}$$

where

$$Q_i = \frac{1}{\Delta x}\int_{x_1}^{x_2} q(x,t)dx \tag{2.20}$$

$$F_{i-1/2} = \frac{1}{\Delta t}\int_{t_n}^{t_{n+1}} f(q(x_{i-1/2},t))dt \tag{2.21}$$

The FVM consists on somehow determining the value of the integrals of Equations 2.20 and 2.21. If Equation 2.21 can be approximated with the values of Q^n, then a fully discretized model can be implemented. See Figure 2.2 for a schematic of this process.

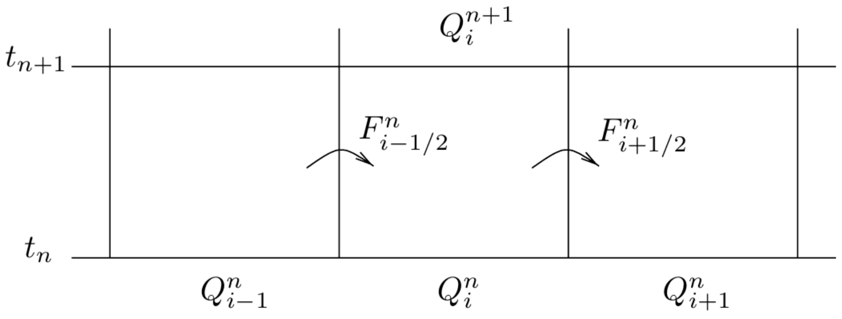


Figure 2.2.: Finite volume method for updating the cell average Q_n i by fluxes at the cell edges [12]

.

The Finite Volume method together with the Riemann Problem approach allows us to account discontinuities across cells. Such discontinuities across cells are typical of non-linear conservation laws. Therefore, choosing FVM for the solving the Shallow Water Equations provides a reliable solution.

2.3.2. The Riemann Problem and F-Wave method

Finite Volume Methods provide a solving scheme that considers cell averages calculated through flux functions, in contrast to cell points in finite differences methods though discretized derivatives. Whether the FVM delivers a reliable solution, depends on finding reliable solution for Equation 2.21. To this end, FVM resorts to the Riemann Problem formulation. An overview of the Riemann problem and the way to treat the flux function will be treated in this section.

The Riemann Problem is a fundamental tool in the development of finite volume methods that solves the hyperbolic equation conservation law with special piecewise constant initial data at some point $x = 0$ as

$$q(x,t) = \begin{cases} q_l, & \text{if } x < 0 \\ q_r, & \text{if } x > 0 \end{cases} \tag{2.22}$$

Essentially, if Q_{i-1} and Q_i are the cell averages in two neighbouring grid cells on a finite volume grid, then by solving the Riemann problem with $q_l = Q_{i-1}$ and $q_r = Q_i$, we can obtain information that can be used to compute a numerical flux and update the cell averages over a time step. For hyperbolic problems, the solution to the Riemann problem typically consists of a finite set of waves that propagate away from the origin at certain wave speeds. For linear hyperbolic systems, finding a solution for q_l and q_m, can be found by calculating the eigenvalues and eigenvectors of the constant matrix A. This method also holds for non-linear systems of equations (with some transformation to a quasi-linear state, see Section 2.2.1), and the exact solution (or good approximations) to the Riemann problem can be constructed. Therefore, we aim to express Equation 2.21 as $F_{i-1/2} = F(Q^n_{i-1/2}, Q_i)$ for obtaining a discretized solving method. In this case, Equation 2.19 becomes

$$Q_i^{n+1} = Q_i + \frac{\Delta t}{\Delta x}\left[F(Q_i^n,\, Q_{i+1}^n) - F(Q_{i-1/2}^n,\, Q_i^n)\right] \tag{2.23}$$

This represents the essence of the Riemann problem, and the method we choose for solving it depends on how we choose to treat $F_{i-1/2}$ based on type of hyperbolic system we are dealing with.

For the classical Riemann problem solution, $q \in m \times 1$ is usually expressed as a linear combination of its eigenvectors multiplied by a function w, referred sometimes in literature as *eigen-coefficients*. For obtaining w, Equation 2.15 can be expressed in its quasi-linear form and, provided that matrix A is diagonalizable, w can be expressed as

$$\begin{aligned} q_t + Aq_x &= 0 \\ w_t + \Lambda w_x &= 0 \end{aligned} \tag{2.24}$$

where

$$\begin{aligned} A &= R\Lambda R^{-1} & &\in m \times m \\ w &= R^{-1} q & &\in m \times 1 \end{aligned} \tag{2.25}$$

Equation 2.24 decouples the system into m waves given by the eigenvalues and eigenvectors as

$$w_t^p + \lambda^p \, w_x^p = 0$$

allowing us to find the trivial solution for the waves as a typical decoupled advection equation. The solution for w^p is then

$$w^p(x,t) = w^p(x - \lambda^p t, 0) \tag{2.26}$$

Equation 2.26 indicates that w^p will remain constant along its characteristic line given by $x - \lambda^p t$, and will be equal to the initial condition at time $t = 0$, see Figure 2.3. Having computed all components $w^p(x,t)$, we can combine them into the vector $w(x,t)$ and, according to Equation 2.25, we can multiply it by the right eigenvectors R for obtaining q

$$q(x,t) = Rw(x,t)$$

This yields a solution for the state variables q as a linear combination of m superposed *simple* waves that constitute the system. This solution is a weighted average of the state variables in the vicinity of x, denoted as

$$q^*(x,t) = \sum_{p=1}^{m} w^p(x,t) r^p \tag{2.27}$$

The solution of $q(x,t)$ provided by Equation 2.27 is calculated depending on the range of influence of each $p-$characteristic, together with the condition from Equation 2.22. Special care must be taken into account in regard of the regions that the characteristics delimit in order to calculate discontinuities appropriately to Figure 2.4. It can be shown[12] that the value of q^* across the cell edges corresponds to the difference of the left and right values of q at the initial state

$$q^* = W^p = \alpha^p r^p \tag{2.28}$$

where α can be obtained from

$$R\alpha = q_r - q_l$$

Lastly a final expression for q can be derived as

$$q(x,t) = q_l + \sum_{p:\ \lambda^p < x/t} W^p \tag{2.29}$$

Equation 2.29 represents the solution to Equation 2.21 as the flux across the interface at x_{i-1}, namely

$$F_{i-1/2} = \frac{1}{\Delta t} \int_{t_n}^{t_{n+1}} f(q(x_{i-1/2}, t))dt = f(q_{i-1/2})$$

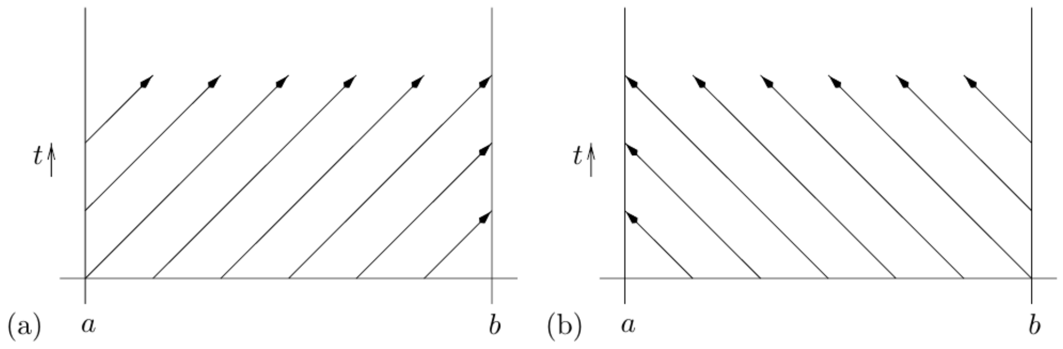


Figure 2.3.: The solution to the advection equation is constant along characteristics given by the equation $x - \lambda^p t$, taking the initial value at $t = 0$ [12].

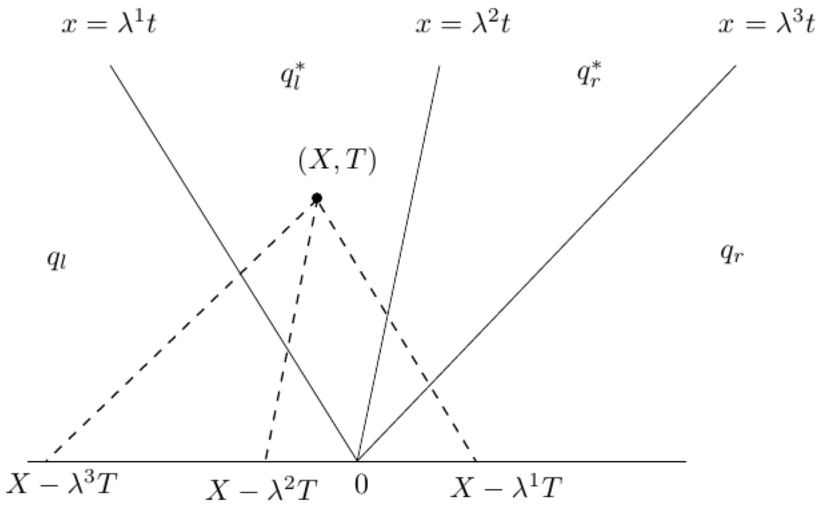


Figure 2.4.: The pth characteristic is traced back to determine the value of w^p from the initial data. The value of q is constant in each wedge of the xt plane: $q_l = w_l^1 r^1 + w_l^2 r^2 + w_l^3 r^3$ $\quad q_l^* = w_r^1 r^1 + w_l^2 r^2 + w_l^3 r^3$ $\quad q_r^* = w_r^1 r^1 + w_r^2 r^2 + w_l^3 r^3$ $\quad q_r = w_r^1 r^1 + w_r^2 r^2 + w_r^3 r^3$. Note that the jump across each discontinuity in the solution is an eigenvector of matrix A [12].

F-Wave Method

The flux function $f(q, x)$ from Equation 2.15 can be discretized with respect to x in some manner consistent with a finite-volume interpretation. For a given grid, two possible discretizations can be considered: *cell-centered flux* functions or *edge-centered flux functions*[1].

In the previous subsection, the derivation for solving the Riemann problem followed a *cell-centered flux functions* approach. By this approach, the derivation yielded a flux function $f_i(q)$ that holds throughout the ith grid cell. In this case, it was a function $f_i(q) = f_i(q, x_i)$ that, if the variation of f is sufficiently smooth, this methods holds sufficiently good. The derivation, together with Equation 2.22, can be summarized as

$$q_t + F_i - 1/2(q, x)_x = 0$$

where

$$F_{i-1/2}(q, x) = \begin{cases} f_{i-1}(q), & \text{if } x < x_{i-1/2} \\ f_i(q), & \text{if } x > x_{i-1/2} \end{cases}$$

An alternative approach is to assume that a distinct flux function $f_{i-1/2}(q)$ is associated with each cell interface $x_{i-1/2}$, rather than with each cell center. This is referred to as *cell-edge flux functions* approach. In this approach, it is the flux at the cell interface that is ultimately required to implement a finite-volume method based on flux differencing, and so associating flux functions with interfaces often makes sense. Under this approach, the Riemann problem at $x_{i-1/2}$ is now a classical Riemann problem for the single equation $q_t + f_{i-1/2}(q)_x = 0$ with the data from Equation 2.22. Note the disctiction on the subscript with the cell-centered flux approach. In cell-edge flux approach, it is the flux difference between cells that is decomposed into waves. This algorithm is based on an approximate Jacobian matrix $A_{i-1/2}$ (e.g. Roe Average[1]) that must be defined at the cell edge. The original non-linear Riemann problem is then replaced by the linear Riemann problem

$$q_t + A_{i-1/2} q_x = 0 \tag{2.30}$$

The classical Riemann problem for a constant-coefficient system can be applied to Equation 2.30, and apply the same procedure as shown in Equation 2.29. However, using waves W from Equation 2.28 will not yield algorithm in general, unless the condition

$$A_{i-1/2}(q_i - q_{i-1}) = f_i(q_i) - f_{i-1}(q_{i-1})$$

is satisfied. In essence, the novel feature of this cell-edge flux functions algorithm is not solving the classical Riemann problem by performing a classical decomposition in the form depicted in Equation 2.29, instead a *flux based decomposition* is used. With this approach, the flux difference $f_i(q_i) - f_{i-1}(q_{i-1})$ is directly decomposed in a linear combination of the eigenvectors $r^p_{i-1/2}$ as

$$f_i(q_i) - f_{i-1}(q_{i-1}) = \sum_{p=1}^{m} \beta^p_{i-1/2} r^p_{i-1/2} = \sum_{p=1}^{m} Z^p_{i-1/2} \tag{2.31}$$

[1]Roe Average Jacobian matrix refers to a matrix such that represents an avergage flux across the edge between two grid cells[18] [11]

where

$$\beta = R^{-1}_{i-1/2}(f_i(q_i) - f_{i-1}(q_{i-1}))$$

For spatially varying fluxes this represents a more natural decomposition. The vectors $Z^p = \beta^p r^p$ are called *f-waves*, as they are analogous to the waves W^p from Equation 2.28, but carry flux increments rather than increments in q.

The cell-edge flux functions, or F-Wave method, has some desirable features and hence implemented on this thesis. The method is conservative and second-order accurate regardless of what approximate Jacobian matrix is used. This may be useful where a Roe average cannot easily be computed, and a simpler expression such as the arithmetic average could instead be used. Another advantage of the F-Wave approach is that it is not necessary to determine the jumps in the conserved variables q that typically arise across the interface in solving the Riemann problem with a spatially varying $f(q, x)$. Since the flux is assumed to be continuous across the interface, decomposition of the flux difference into eigen-components immediately yields the propagating waves that are needed for the high-resolution wave-propagation algorithm[13].

2.4. Free-surface flow

In Section 4, we addressed the concept of free-surface flow. Theory for free-surface flow is quite extensive, as there exist a vast number of engineering applications that deal with free-surface flows. One of the most basic concepts within free-surface flow theory, is the flow characterization, depending on the *Froud number*.
In free-surface flows, the Froud Number Fr is an important dimensionless parameter that, depending on its value, characterizes the flow into three categories

$Fr < 1$	Subcritical
$Fr = 1$	Critical
$Fr > 1$	Supercritical

The Froud Number is expressed as

$$Fr = \frac{V}{c} = \frac{V}{\sqrt{gh}}$$

where V is the flow speed, g is the gravity, and h is the representative height of the free surface. The Froud number expresses the ratio between the flow speed to the wave speed, as an analogy to the Mach number that expresses the ratio between the flow speed to the sound speed. The Froud number can also be expressed of as the square root of the ratio of inertia (or dynamic) force to gravity force (or weight) as

$$Fr = \frac{2(\frac{1}{2}\rho^2 A)}{mg} \propto \frac{inertia\, force}{gravity\, force}$$

It follows that in subcritical flow, i.e. at low flow velocities ($Fr < 1$), a small disturbance travels upstream (with a velocity $c_0 - V$ relative to a stationary observer) and affects the upstream conditions. In supercritical flow, i.e. at high flow velocities ($Fr > 1$), a small disturbance cannot travel upstream (in fact, the wave is washed downstream at a velocity of $V - c_0$ relative to a stationary observer) and thus the upstream conditions cannot be influenced by the downstream conditions, and the flow in this case is controlled by the upstream conditions. Therefore, a surface wave travels upstream when $Fr < 1$, is swept down stream when $Fr > 1$, and appears frozen on the surface when $Fr = 1$ [4]. As will be seen in Section 4, the Froud number is of great importance for determining boundary conditions, as it acts as an indicator of flow conditions across a (coupled) domain.

As in compressible flows, a liquid can accelerate from subcritical to supercritical flow. Certainly, it can also decelerate from supercritical to subcritical, but it can do so by undergoing a shock. The shock in this case is called a *hydraulic jump*, Figure 2.5, which corresponds to a normal shock in compressible flow. Shockwaves are typical of non-linear hyperbolic systems and they represent abrupt changes in state, or discontinuities. In case of the Shallow Water Equations as non-linear hyperbolic systems, the discontinuity is manifested as a hydraulic jump. The steepness of the hydraulic jump is proportional to the discontinues between states. A typical scenario with strong discontinuities is the breaking of a dam, which will be part of the implementation of this work.

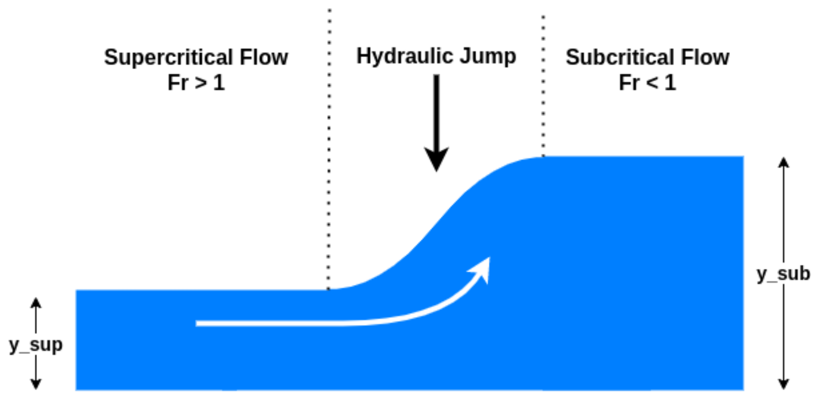


Figure 2.5.: Hydraulic jump and regions of flow type with sub/supercritical heights

2.5. Summary

This section introduced Navier-Stokes equations and the Shallow Water Equations models, as well as their solving methods. For solving the Navier-Stokes equations the Volume-Of-Fluid method will be implemented and considered an incompressible Newtonian fluid.

For solving the SWE model, the Finite Volume Method will be used and solved as a Riemann problem with the F-wave method without accounting for the source terms. The model considers the fluid to be an incompressible inviscid fluid, and holds valid as long as the horizontal dimensions are much greater than the vertical ones. Note that the SWE model does not account for the viscous effects, as the solver that will be used and described in Section 3.2 disregards such effects.

The influence of the Froud number is of great importance in the sense of being able to predict results and behaviours in accordance to free-surface flow conditions. Because of this predictions, we can approach and model the problem accordingly, and construct an appropriate formulation of the problem, specially when setting the boundary conditions on the domain's interface, as will be seen in chapter 4.

Both, the SWE and NS model reach a solution for free-surface problems. In case of the SWE, it reaches a solution under a 2-dimensional formulation, leading to a lower number domain cells which is always a desirable advantage. However, it fails on representing the 3D effects. If we choose the NS formulation, the 3D effects will certainly be represented, at a higher computational cost, however. Consequently, paradox leads us to thinking of having a solver which could combine the advantages of both solvers. By partitioning the domain into regions where the 3D effects(e.g. waves colliding against solids) could or could not be neglected, we could assign a particular methodology for each region, and eventually coupling their solutions. This inter-dimensional coupling allows more flexibility to the overall problem, and it is the centerpiece of this thesis. The upcoming sections will describe the coupling in detail.

3. Tools

In Section 2, we discussed the derivation of two methods for solving free surface problems. We reviewed the characteristics of each method, and we highlighted their advantages and limitations; based on these, we addressed domain partitioning as a flexible approach for free-surface problems. In this section we describe the solvers that shall implement the methodologies from Section 2, and shortly introduce the necessary tool needed for coupling these solvers as a partitioned simulation.

3.1. preCICE

preCICE (Precise Code Interaction Coupling Environment) is a coupling library for partitioned multi-physics simulations, where partitioned refears to puple solvers, capable of simulating a subpart of the complete physics within a simulation [3]. preCICE is designed to be used on cartesian grid solvers, offering an application programming interface (API) over a wide range of solvers, including C++ and frameworks such as OpenFOAM[9].

As a general overview, we can couple two solvers by calling the preCICE API through the adapters. preCICE takes care of the communication and data mapping between the participants (solvers). By calling preCICE through our solvers, we let preCICE orchestrate the coupling by only setting the parameters in the preCICE configuration file. Some of these parameters include the variables we want to exchange, the coupling scheme (explicit or implicit), the time stepping, the end time of simulation, etc. Figure 3.1 shows an overview of preCICE as the coupling tool. A deep analysis of the capabilities and development of preCICE is available in literature[21] [9], as well as in the preCICE repository[1].

This approach allows the user a high degree of flexibility for building partitioned simulations, as it makes it possible to couple solvers from different frameworks, fields in physics, and even solvers formulated in different dimensions.

[1]https://github.com/precice

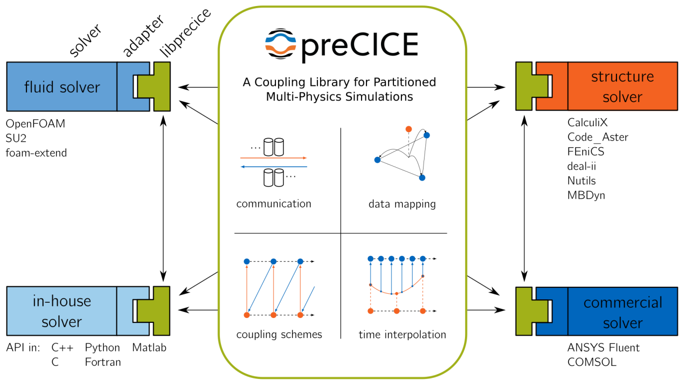


Figure 3.1.: An overview of preCICE as a coupling tool. The rectangles on the sides represent the coupling solvers. The adapter (green "T") is implemented in the solver, letting preCICE drive the simulation, and manage the communication and the data mapping between solvers.

3.2. SWE Solver

The Shallow Water Equations solver "is an education-oriented code that implements simple Finite Volumes models that solve the Shallow Water Equations" [16]. It was developed at the TUM Scientific Computing and Computer Science Chair for teaching on different parallel programming models for computational science and engineering.

An article [2] about the SWE solver design and its content was published by IEEE[2]. The article contains a detailed description of the solver's implementation and capabilities. The article also contains short but concise theory about different Riemann Problem methods, given that the solver's implementation allows the user to choose a particular method for approximating the fluxes. For this thesis, we chose the *f-wave* as the Riemann Problem solver, as discussed in Section 2.3.2.

The code for SWE solver can be found on GitHub[3]. A Doxygen documentation[4] is also available; note that the documentation does not contain the implementation done in this thesis.

[2]Institute of Electrical and Electronic Engineers
[3]https://github.com/TUM-I5/SWE/tree/e80170a445e8d1896a8b59bc6d5669ac7ce7d465. The link points to the the commit from which we implemented upon.
[4]https://www5.in.tum.de/SWE/doxy/

3.2.1. SWE implementation

The solver provides an algorithm for computing the solution on a discretized mesh on specified time steps for the height h, and the horizontal and vertical discharges hu and hv, as seen in Equation 2.17. The solver also includes a collection of scenarios, from which the *RadialDamBreakScenario* was chosen as the base scenario class for some of the case scenarios that will be simulated in this thesis. The other case scenario, open-channel flow, was implemented for this solver. Both scenarios are discussed in Section 4.

In order to exchange data across the solver, we implemented an adapater with the preCICE library. In a very general overview, the adatpter will control the communication, the data exhange and the data mapping. The exchange details can be set in the preCICE configuration files. We will address a closer view to the data exchange in Section 4

3.3. OpenFOAM

OpenFOAM[5] (Open-source Field Operation And Manipulation) is a toolbox for the development of numerical solvers, and pre- and post-processing utilities for the solution of (prominently) Computational Fluid Dynamics (CFD) problems [19]. OpenFOAM is distributed under an open-source licence, thus allowing modifications and adaptation to the collection solvers and utilities that comprise the toolbox[14].

3.3.1. interFoam

Within the collection of solvers for CFD applications, OpenFOAM supports incompressible and compressible flows, laminar and turbulent flows, multi-phase flows, and buoyancy-driven flows [5]. For this thesis, we focus on multi-phase flow solvers, particularly in *interFoam*.

As a multi-phase solver, interFoam is a solver for two incompressible, isothermal immiscible fluids, that utilizes the Volume-Of-Fluid (VOF) method. As discussed in Section 2.1.1. VOF's most recognizable feature is the implementation of a volume fraction indicator, that distinguishes between two different fluids, usually water and air, for solving the Navier-Stokes equations. Consequently, it results convenient to use interFoam as the solver for free-surface problems.

For free-surface problems, the variables of most interest are, generally, the pressure, the velocity and the flow depth. Similarly for this thesis, we focus our attention on these variables, as they become fundamental for coupling solutions between the SWE and the

[5] `www.openfoam.org`

Navier-Stokes models. interFoam provides a solution for the pressure, the velocity and the volume fraction indicator. It is possible to calculate the flow depth from the values of the volume fraction indicator, as will be further discussed in Section 4.

As an open-source framework, we can adapt and interface to OpenFOAM's implementation in accordance to a particular goal: in this case, we aim to exchange data from interFoam to other solvers. Therefore, it is possible to add the preCICE library to OpenFOAM, and let preCICE drive the exchange of information and the algorithm for the computation. For this, we resort to the OpenFOAM adapter for preCICE, allowing us to easily set the variables we want to exchange, previously indicated in the preCICE configuration file. A full description[5] of the OpenFOAM adapter is available, and it can be downloaded from the preCICE repository[6].

[6] `https://github.com/precice/openfoam-adapter/tree/FF-OF7`. Commit *383a18a* is the commit from which we build our implementation.

4. Implementation

Coupling multi-dimensional domains has fundamental considerations that must be taken into account, and they will be addressed in this chapter. In this work, two domains will be considered for coupling. They are referred to as Left and Right domain, and each of them solves either a 2-dimensional or 3-dimensional system.
As a first step for the implementation on this work, a combination of 2D and 3D systems on each domain must be assigned. The setups and case scenarios of this combination will be described in Section 4.2. Whether the coupling is from higher to lower dimension, lower to higher or it is in the same dimension, has different treatments in regard of the exchange of variables. Information across dimensions must be consistent, so as a second step, proper *data mapping* has to be established. Once data has been mapped to the boundaries of a neighbouring domain, one must know how to treat it. As a third step, this is done by establishing appropriate *boundary conditions* that are consistent with the phenomena that we are dealing with, based upon the flow characterization, i.e. whether the flow is supercritical or subcritical. This is discussed in section 4.1. Data mapping and the treatment of boundary conditions depending on the flow characterizations will be discussed in Section 4.3. We can find the implementation and the upcoming test cases in GitHub[1], and general instructions for reproducing the cases in Appendix A.1.

It is worth mentioning that this approach allows us an extended flexibility for coupling more than two domains across different dimensions, e.g. we can add an intermediate domain. As long as we can set the appropriate values on the boundaries and have a consistent mapping across the domains, the coupling is possible.

4.1. Boundary conditions for free-surface flow

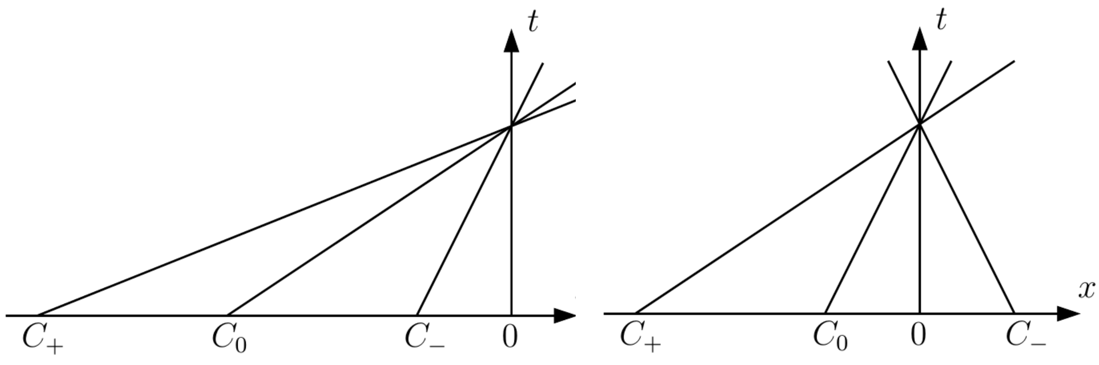
The theory and deduction of the Shallow Water Equations model, as a hyperbolic PDE system, is closely related to the free-surface flow description, seen in Section 2.4. Recall from Section 2.2.1 the eigenvalues of the SWE system,

$$\lambda^{x1} = u - c, \qquad \lambda^{x2} = u, \qquad \lambda^{x3} = u + c$$

where $c = \sqrt{gh}$ represents the wave-speed (celerity) of the m-waves that compose the system given by its eigenvectors. Figure 4.1 depicts a picture on how each of the m-waves

[1] https://github.com/pachesp/2d3dcoupling or
https://github.com/precice/openfoam-adapter/tree/SWE-interFoam

would look like depending on the flow characterization. Figure 4.1a shows a supercritical flow, while Figure 4.1b depicts a subcritical flow. For the supercritical case, it was discussed that no information from downstream can travel upstream and any disturbance would be washed by the flow velocity u. This means that the flow conditions are entirely determined upstream. Figure 4.1a shows that all of the m-waves come from the left side of the origin, i.e. upstream, and as the Riemann problem states, q is determined entirely from the q_l side. For the subcritical case, waves downstream do propagate upstream, meaning that the flow conditions are influenced from the downstream conditions. Figure 4.1b depicts one of the m-waves taking information from the right side of the origin, that is from the q_r side, therefore the actual state variable q is determined from both sides.



(a) Supercritical characteristics

(b) Subcritical characteristics

Figure 4.1.: Supercritical and subcritical characteristics [14]. Values C_+, C_0, C_- correspond to $\lambda^{x1}, \lambda^{x2}, \lambda^{x3}$ respectively. Each characteristic has a different slope that corresponds to their speed of propagation (given by eigenvalues). Positive eigenvalues mean positive slopes, and the flow speed u is greater than c and waves can not travel upstream. This is supercritical flow (Figure (a)). If an eigenvalue is negative, then the slope is negative, meaning that c is bigger than u, and waves can travel upstream. This is a subcritical case (Figure (b)).

One can see the relation of the Froud number to the speed of the waves given by the eigenvalues. In the case where the flow velocity $u > c$ the Froud number $Fr > 1$, hence resulting in supercritical flow. In case where $u < c$, the flow is subcritical according to the Froud number $Fr < 1$. We can set two types of boundary conditions across domains according to the flow characterization [14], which is based on the direction of its characteristics (Figure 4.1):

1. Supercritical flow ($Fr > 1$) - composed only by outward characteristics, i.e. information leaving the domain. In this case, *outflow* boundary conditions are considered, i.e. *Neumann-zero* boundary conditions.

2. Subcritical flow ($Fr < 1$) - includes an inward characteristic, i.e. information entering the domain. In this case, *inflow* boundary conditions are considered, and they are treated as *Dirichlet* boundary conditions.

4.2. Cases Setup

In this section, the test cases will be proposed by assigning a combination of 2D and 3D systems to the left and right domains. These combinations can be found in Table 4.1. Solutions of the 2D domain will be carried out by the SWE solver, while solutions of the 3D domain will be performed by interFoam, as seen in Section 3. In this section, "interFoam" and "OpenFOAM" will be used interchangeably as the solver for the 3D domain.

Left / Right	**2D**	**3D**
2D	SWE $\rightarrow$ SWE	SWE $\rightarrow$ OF
3D	OF $\rightarrow$ SWE	OF $\rightarrow$ OF

Table 4.1.: Solver used for data exchange across domains.

The entries of Table 4.1 show the direction in which the domains shall be ordered. *SWE* and *OF* refer to the 2D Shallow Water Equations Solver and to the 3D Navier-Stokes solver driven by OpenFOAM, respectively. Each case will be implemented under subcritical and supercritical conditions, resulting in a total of 8 cases to simulate. Tables 4.2 and 4.3 show a collection of scenarios for the *supercritical* and *subcritical* cases, respectively.

The following bullet points briefly explain why the cases from Tables 4.2 and 4.3 were chosen according to the flow characterization we want to simulate.

- Supercritical cases - The fluid is expected to travel from the left to the right domain, without any information or wave travelling back to the left domain. This is a typical feature of supercritical flow, as seen in the previous section. In case of SWE $\rightarrow$ SWE, the scenario is constituted by a column of water over fluid at rest at time $t = 0$. For $t > 0$, the perturbations waves are expected to travel to all directions, crossing to the right domain. In case of SWE $\rightarrow$ OF, the scenario is similar to the previous case, just adding a second water column at the original position, with the intention to see more clearly the wave crossing to the right domain. For the OF $\rightarrow$ SWE case, we have a similar scenario as the last two. The disturbances are expected to cross to the right domain and continue their way without any disturbances travelling upstream. Lastly for the OF $\rightarrow$ OF case, a modified breaking-of-a-dam example from

Case / Domain	**Left domain**	**Right domaing**
SWE → SWE	Radial breaking dam	Surface at rest
SWE → OF	Radial breaking dam x2	Surface at rest
OF → SWE	Radial breaking dam	Surface at rest
OF → OF	Breaking dam	Empty domain

Table 4.2.: Scenarios for each setup on supercritical flow.

Case / Domain	**Left domain**	**Right domaing**
SWE → SWE	Radial breaking dam	Radial breaking dam
SWE → OF	Flow to the right	Surface at rest with wall
OF → SWE	Flow to the right	Surface at rest with wall
OF → OF	Breaking dam	Empty domain with wall

Table 4.3.: Scenarios for each setup on subcritical flow.

OpenFOAM tutorials[2] is used, with an additional second domain coupled on the right side.

- Subcritical cases - In this case the setup becomes a bit more elaborated, since the intention is that perturbations travel back upstream from the right domain to the left domain. For the SWE → SWE case, this is done by setting a second column of water on the right domain at time $t = 0$. The perturbations from the right domain are expected to interact with the perturbations of the left domain. For the SWE → OF and OF → SWE cases, an inflow velocity on the left boundary is set. The flow is intended to cross to the right domain, and reflect back after reaching a wall on the right boundary on the right domain, travelling upstream crossing once more

[2] https://cfd.direct/openfoam/user-guide/v6-damBreak/

the interface between domains with an opposite direction. For the OF $\rightarrow$ OF case, the setup is the same as in the supercritical case with an additional wall boundary condition on the right boundary of the right domain.

4.3. Mapping and Dirichlet boundary conditions

As seen in Section 3, the Shallow Water Equations solver obtains a solution for the height h and discharges hu and hv. On the other hand, interFoam provides a solution for the velocity $\mathbf{u}$, the pressure p and the volume indicator α. In order to exchange this information across solvers in a consistent way, an appropriate mapping has to be implemented. Mintgen proposes mapping methods for the previously mentioned variables depending on the test case. These methods, some with slight modifications, will be discussed in the upcoming sections.

4.3.1. 2D $\rightarrow$ 3D mapping

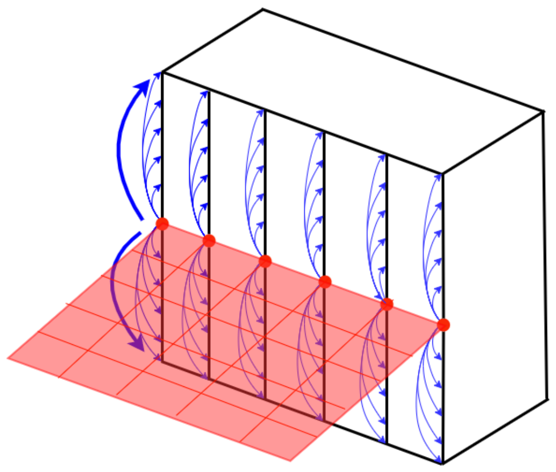
Referring to Table 4.1, a description of the 2D $\rightarrow$ 3D case will be addressed in this subsection. This implies having SWE as the left domain and OF as the right domain, and the flow going from the left to the right domain. Depending on the flow characterization (supercritical or subcritical), data will be exchanged either uni-directionally or bi-directionally. In a uni-directional coupling, information flows only from one solver to another, e.g. from the left to the right domain. In a bi-directional coupling, information is exchanged in both directions. Figure 4.2 depicts an overview on how data is exchanged between domains. The exchange happens from the cell edge from one domain on the boundary to cell edge on the interface on the neighbouring domain. In this case, we set the same resolution on the z direction (pointing outside of the screen) for both domains.

Mapping height h and volume indicator α

As seen in Section 2.1.1, α indicates the cells that occupy liquid; a cell full of liquid is set to 1, a cell full of gas is set to 0, and a cell with both liquid and gas indicates the free surface, and it is set to a value between 1 and 0. Consequently, there is a direct relationship between α and h. Both variables relate differently depending on the flow characterization we are dealing with:

1. Supercritical case

   Height exchange is happening from the left domain to the right domain. The indicator function α is expressed as a function of h, and it is set as a Dirichlet boundary condition on the boundary of the 3D domain Γ_{3D}. α is set to 0 for the cell faces on the 3D domain, whose lower edge is above the water level from the 2D domain $h_{\Gamma_{2D}}$. α is set to 1 in case the cell's lower edge is below $h_{\Gamma_{2D}}$. For cell faces that lie on the


(a) Mapping 2D to 3D domain

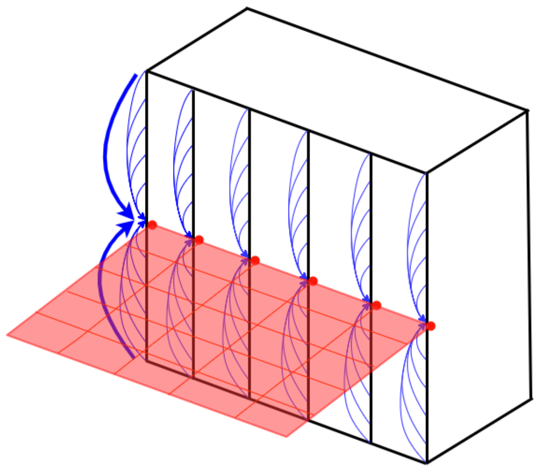

(b) Mapping 2D to 3D domain

Figure 4.2.: 2D $\leftrightarrow$ 3D mapping

interface between water and air, a linearly interpolated value is assigned to α. Thus, an expression for α can be expressed as

$$\alpha(\Gamma_{3D}) = f(h)$$

$$\alpha(\Gamma_{3D}) = \begin{cases} 0, & \text{if } h_{\Gamma_{2D}} \leq h_{\Gamma_{3D}} - 0.5\Delta h_{\Gamma_{3D}} \\ 1, & \text{if } h_{\Gamma_{2D}} \leq h_{\Gamma_{3D}} - 0.5\Delta h_{\Gamma_{3D}} \\ \dfrac{h_{\Gamma_{2D}} - h_{\Gamma_{3D}}}{\Delta h_{\Gamma_{3D}}} + 0.5, & \text{otherwise} \end{cases}$$

2. Subcritical case

   In this case, the height from the 3D domain is directly mapped to the height on the 2D domain as Dirichlet boundary condition.

$$h_{\Gamma_{2D}} = h_{\Gamma_{3D}}(\alpha)$$

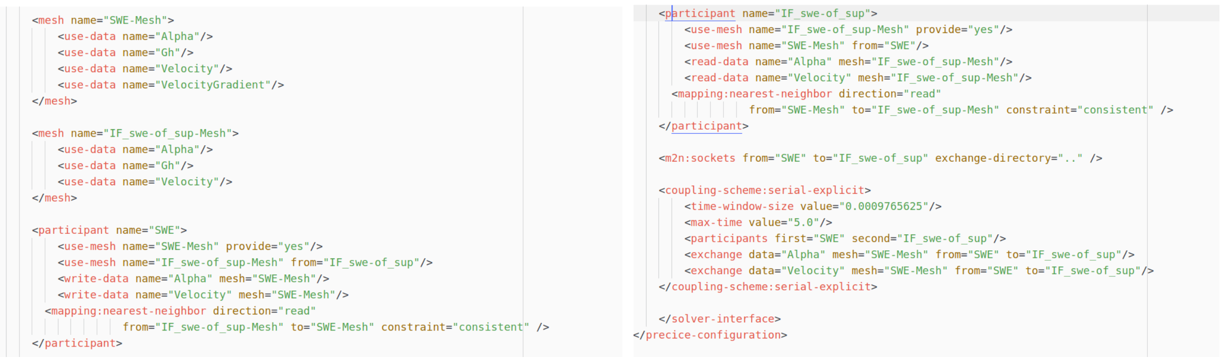
Velocity Mapping

The velocity profile on the 3D domain $h_{\Gamma_{3D}}$ is set as a Dirichlet condition on Γ_{3D} in subcritical and supercritical flow conditions. As a simplistic approach, $h_{\Gamma_{3D}}$ is set as a constant value over the flow depth based, on the velocity values of the 2D domain $\mathbf{u}_{\Gamma_{2D}}$. This is expressed as

$$\mathbf{u}_{\Gamma_{3D}} = \mathbf{u}_{\Gamma_{2D}}\, \alpha_{\Gamma_{3D}}$$

where $\mathbf{u}_{\Gamma_{2D}} = h\mathbf{u}/h$. This approach provides an acceptable mapping between domains as will be seen in Section 5, however, such profile implies a high velocity gradient near the bottom, leading to overestimation of the wall shear stress. Consequently, more energy would be necessary to mantain the discharge, eventually leading to an increase in water level [14], thus, breaking the conservation of mass principle. Mintgen proposes mapping the velocity profile by adding a logarithmic term to counteract the previously mentioned effects. Such an implementation can be found in his dissertation. Essentially, the logarithmic profile depends on the friction velocity at the bottom of the wall, and it is a function of the wall shear stress τ_b at the bottom. In the case of interFoam, τ_b is calculated from the 3D solver. Given the capabilities of the 2D solver, calculating a shear stress for the 2D domain, from the velocity gradients on the vertical direction, resulted in a cumbersome implementation, and the results were not satisfactory. Therefore, the simplistic approach was preferred.

Mapping of the SWE $\rightarrow$ OF case has been addressed for subcritical and supercritical flow. Figures 4.4 and 4.5 show the preCICE configuration files[3]. Figure 4.3 shows an example of the preCICE configuration file. In the configuration file, the user specifies the details to carry out the coupling. Some of the settings include choosing participants (solvers) and the exchanged variables. Within these fields, we specify the direction in which the coupling should be. In this case, we set the SWE and interFoam as the first and second participants, respectively, and the *Velocity* and *Alpha* as the exchanged variables from SWE to interFOAM using a nearest neighbour mapping. The end time and, the time stepping and coupling scheme (implicit or explicit) must be also defined in the configuration file.



```xml
  <mesh name="SWE-Mesh">
      <use-data name="Alpha"/>
      <use-data name="Gh"/>
      <use-data name="Velocity"/>
      <use-data name="VelocityGradient"/>
  </mesh>

  <mesh name="IF_swe-of_sup-Mesh">
      <use-data name="Alpha"/>
      <use-data name="Gh"/>
      <use-data name="Velocity"/>
  </mesh>

  <participant name="SWE">
      <use-mesh name="SWE-Mesh" provide="yes"/>
      <use-mesh name="IF_swe-of_sup-Mesh" from="IF_swe-of_sup"/>
      <write-data name="Alpha" mesh="SWE-Mesh"/>
      <write-data name="Velocity" mesh="SWE-Mesh"/>
    <mapping:nearest-neighbor direction="read"
            from="IF_swe-of_sup-Mesh" to="SWE-Mesh" constraint="consistent" />
  </participant>
```

```xml
  <participant name="IF_swe-of_sup">
      <use-mesh name="IF_swe-of_sup-Mesh" provide="yes"/>
      <use-mesh name="SWE-Mesh" from="SWE"/>
      <read-data name="Alpha" mesh="IF_swe-of_sup-Mesh"/>
      <read-data name="Velocity" mesh="IF_swe-of_sup-Mesh"/>
    <mapping:nearest-neighbor direction="read"
            from="SWE-Mesh" to="IF_swe-of_sup-Mesh" constraint="consistent" />
  </participant>

  <m2n:sockets from="SWE" to="IF_swe-of_sup" exchange-directory=".." />

  <coupling-scheme:serial-explicit>
      <time-window-size value="0.0009765625"/>
      <max-time value="5.0"/>
      <participants first="SWE" second="IF_swe-of_sup"/>
      <exchange data="Alpha" mesh="SWE-Mesh" from="SWE" to="IF_swe-of_sup"/>
      <exchange data="Velocity" mesh="SWE-Mesh" from="SWE" to="IF_swe-of_sup"/>
  </coupling-scheme:serial-explicit>

  </solver-interface>
</precice-configuration>
```

Figure 4.3.: preCICE configuration file for supercritical case. This setup corresponds to Figure 4.4

[3]The naming on the preCICE configuration files on the images has been modified for presentation purposes

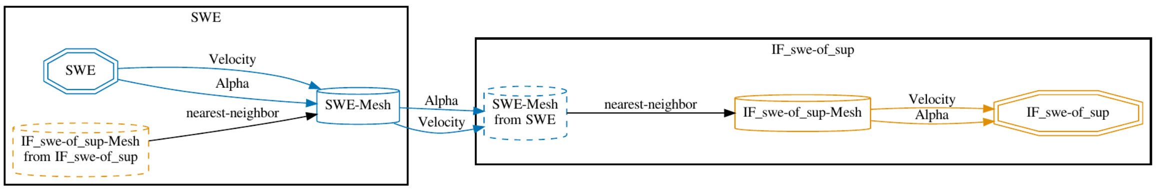


Figure 4.4.: SWE-OF supercritical preCICE configuration file. As information does not need to be sent back to the SWE, the coupling is uni-directional.

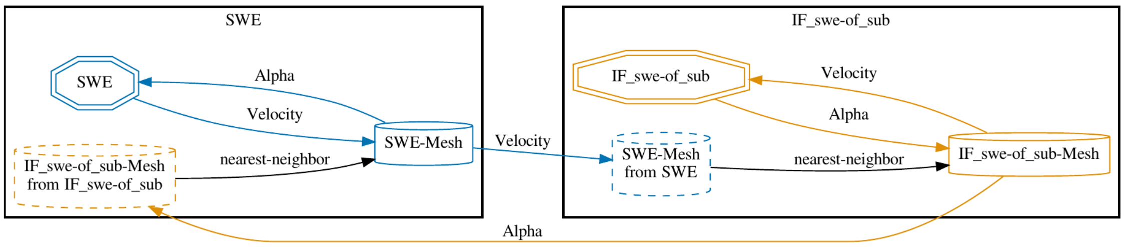


Figure 4.5.: SWE-OF subcritical preCICE configuration file. The volume fraction indicator α is sent back to SWE.

4.3.2. 3D $\rightarrow$ 2D mapping

Continuing with the cases from Table 4.1, the mapping from the 3D to the 2D domain is now addressed in this subsection. In this case, the 3D domain will be on the left side, with flow going from the left to the right domain. Similarly as in the 2D $\rightarrow$ 3D case, data will be exchanged uni-directionally or bi-directionally depending on the flow characterization (see, Figure 4.2b). The implementation for data mapping on each case is shown next.

Mapping height h and pressure p

In this case, the treatment of the pressure for interFoam must be taken into consideration to achieve a well-posed set of initial conditions for the Volume-of-Fluid method. Additionally, this serves as means for calculating the water level on the 3D domain, as further described. Once again, supercritical and subcritical flows are considered.

1. Supercritical case

   Mapping the height for this case is done exactly as in 2D $\rightarrow$ 3D for subcritical flow in Section 4.3.1, i.e.

$$h_{\Gamma_{2D}} = h_{\Gamma_{3D}}(\alpha)$$

2. Subcritical case

The constitutive equation for pressure for SWE is given by Equation 2.11 as a function of height. Thus, pressure can be calculated from the height downstream (from the 2D domain) and can be set as a Dirichlet boundary condition on Γ_{3D}. This is done via

$$p_{\Gamma_{3D}} = \rho_{\Gamma_{3D}}\, g\, h_{\Gamma_{2D}} \tag{4.1}$$

where $\rho_{\Gamma_{3D}}$ is given by Equation 2.5 as a function of α , g is gravity and $h_{\Gamma_{2D}}$ is the water level on the 2D domain.

By setting the pressure on Γ_{3D} from the downstream pressure, interFoam automatically adjusts the water level [14]. On this thesis, the quantity $gh_{\Gamma_{2D}}$ from Equation 4.1 is firstly calculated and later exchanged with the OF solver, where it is finally multiplied by the density term $\rho_{\Gamma_{3D}}$. See Figure 4.7.

Mapping discharge $h\mathbf{u}$

The discharge $h\mathbf{u}$ on the SWE domain is directly mapped from the OF domain and set as a Dirichlet boundary condition on Γ_{2D} for both supercritical and subcritical flow. That is

$$q_{\Gamma_{2D}} = q_{\Gamma_{3D}}$$

where $q_{\Gamma_{3D}} = \mathbf{u}_{\Gamma_{3D}} h_{\Gamma_{3D}}(\alpha)$.

Mapping for the state variables for subcritical and supercritical flow for the 3D $\rightarrow$ 2D case has been addressed. Figures 4.6 and 4.7 show the preCICE configuration files, where the details of the coupling between both solvers are summarized.

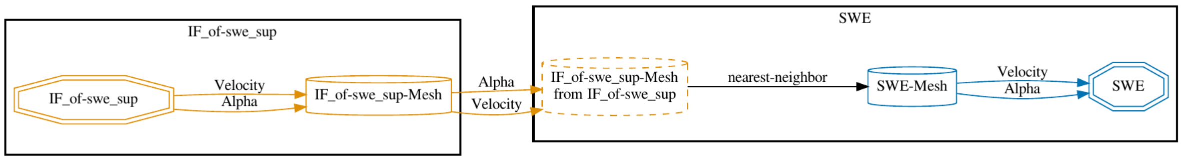


Figure 4.6.: OF-SWE supercritical preCICE configuration file. In supercritical flow only information from upstream (left domain) is exchanged.

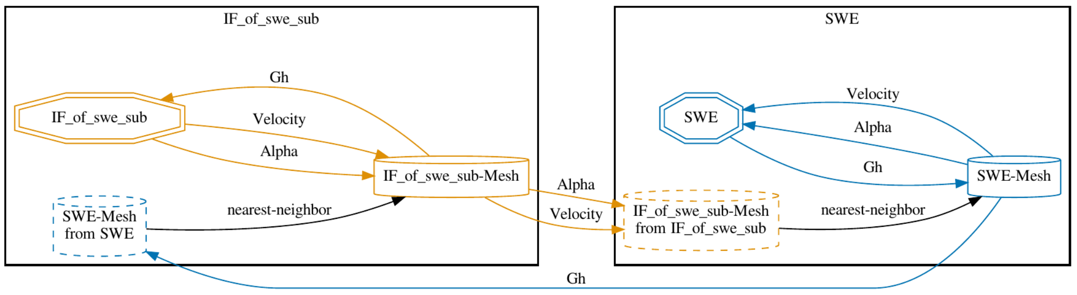


Figure 4.7.: OF-SWE subcritical preCICE configuration file. The term $g\,h_{\Gamma_{2D}}$ from Equation 4.1 is exchanged from the right to the left domain for setting the pressure on the OpenFOAM adapter.

4.3.3. SWE $\rightarrow$ SWE

In this case, data exchange happens across domains on the same dimension: 2D. Therefore, data exchange can be performed directly. The problem is solved across two 2D domains. For the supercritical case, data on the left domain is exchanged to the boundaries on the right domain, and set as Dirichlet boundary condition. For the subcritical case, additional to the information from the left domain, information on the right domain must be exchanged with the left domain. For the subcritical case, this is done as a gradient that has to be firstly calculated on the right domain with previously sent data from the left domain. Once the gradient has been calculated, it is sent back to the left domain and added to its state. This implementation implies bi-directional coupling. The gradient is calculated as follows

$$\begin{aligned}\frac{\partial\, h}{\partial n} &= h_{left} - h_{right}\\ \frac{\partial\, hu}{\partial n} &= hu_{left} - hu_{right}\\ \frac{\partial\, hv}{\partial n} &= hv_{left} - hv_{right}\end{aligned}$$

This implementation resulted to work as well for the supercritical case and therefore implemented. It could be possible, however, to only exchange data uni-directionally from the left to the right domain in order to save computations, however, this was not implemented. The preCICE configuration file, shown in Figure 4.8, is the same for both cases, and it summarizes the details of the coupling.

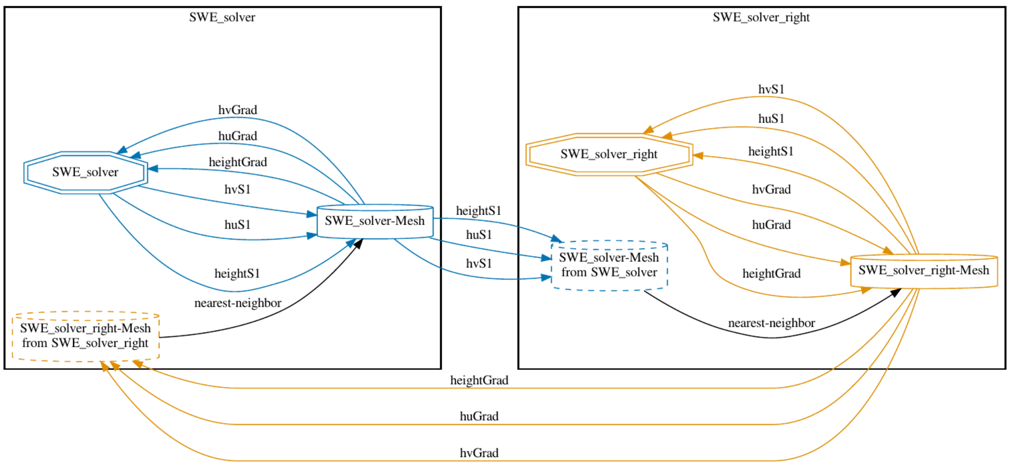


Figure 4.8.: OF-SWE supercritical preCICE configuration file.

4.3.4. OpenFOAM → OpenFOAM

Supercritical Flow

In this case, the problem is solved within two 3D domains. For the supercritical case, the variables α and $\mathbf{u}$ are set as Dirichelt boundary conditions on the right domain, while the pressure is set as Dirichlet condition on the left domain.

Figure 4.9 shows the preCICE configuration file for the supercritical case.

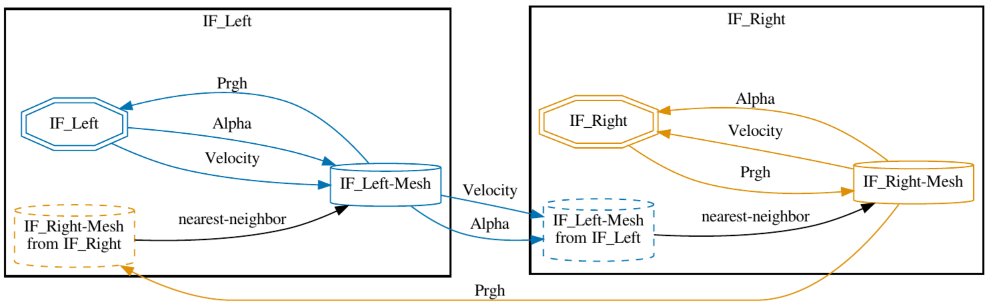


Figure 4.9.: OF-SWE supercritical preCICE configuration file.

Subcritical Case

For the subcritical case, we could not reach a configuration that yields satisfactory results. We experimented with Dirichlet and Neumann boundary conditions, and still obtained meaningless results. We tried similar settings to the subcritical case, shown in Figure 4.9, and we observed results from which we can do a better analysis, yet still unsuccessful. In Section 5.4 we show a precise configuration of this case and present our conclusions.

4.3.5. Neumann boundary conditions

The previous section provided mapping functions and Dirichelt boundary conditions for the inward characteristics for supercritical and subcritical flow (refer to Figure 4.1b). For having a well-posed solution, the outward characteristics also need to be treated. As discussed in Section 4.1, the outward characteristics shall be treated as Neumann boundary conditions, as this represents information leaving the domain. The gradient of the variables on the Neumann boundaries with respect to the normal direction, $\partial/\partial n$ is set to 0, meaning that the value at the boundary is equal to the one on the cell face in the inner domain. According to Mintgen [14], "it would be possible to calculate the actual gradient between the solutions on the 2D and 3D domain, resulting in higher accuracy, but it has been found that setting a gradient equal to 0 results in a stable choice". Table 4.4 shows a review of the boundary condition across domains for supercritical and subcritical flow.

	h and q on Γ_{2D}	α, p and u on Γ_{3D}
Supercritical & Subcritical SWE $\rightarrow$ SWE	$h_{left} = h_{left} + \partial h/\partial n$ $hu_{left} = hu_{left} + \partial hu/\partial n$ $hv_{left} = hv_{left} + \partial hv/\partial n$	
Supercritical SWE $\rightarrow$ OF	$\partial h/\partial n = 0$ $\partial q/\partial n = 0$	$\alpha = f(h)$ $\partial p/\partial n = 0$ $u = f(q, u)$
Subcritical SWE $\rightarrow$ OF	$h = h_{\Gamma_{3D}}$ $\partial q/\partial n = 0$	$\partial \alpha/\partial n = 0$ $\partial p/\partial n = 0$ $u = f(q, u)$
Supercritical OF $\rightarrow$ OF	$h = h_{\Gamma_{3D}}$ $q = q_{\Gamma_{3D}}$	$\partial \alpha/\partial n = 0$ $\partial p/\partial n = 0$ $\partial u/\partial n = 0$
Subcritical OF $\rightarrow$ SWE	$\partial h/\partial n = 0$ $q = q_{\Gamma_{3D}}$	$\partial \alpha/\partial n = 0$ $p = f(h)$ $\partial u/\partial n = 0$
Supercritical OF $\rightarrow$ OF		$\alpha_{right} = \alpha_{left}$ $u_{right} = u_{left}$ $p_{left} = p_{right}$
Subcritical OF $\rightarrow$ OF		*discussion in Section 5.4*

Table 4.4.: Dirichlet and Neumann boundary conditions for the inter-dimensional combinations and flow characterization [14], and for the cases on the same dimension.

5. Results

In Section 4 we discussed the details of exchanging information and of the boundary conditions across domains, depending on the cases shown in tables Tables 4.2 and 4.3. In this section, we present the specific configuration and parameters that were used for simulating each case, as well as an analysis of the results, including our most important observations. Keep in mind, we used an *explicit coupling* for all of the cases.

5.1. SWE $\rightarrow$ SWE

As the first case, two 2D SWE solvers are coupled together for supercritical and subcritical flow. Figure 5.1 shows a simple schematic of the domains at time $t = 0$, and Table 5.1 shows the parameters used for the simulation of the left and right domains for both types of flow. The types of boundary conditions used for this configuration, also discussed in Section 4.3.1, are shown in Table 5.2.

For the supercritical case, a column of water at $t = 0$ is set on the left domain, surrounded by water at rest. At $t > 0$, the waves are expected to cross to the right domain in a continuous way. For the subcritical case at time $t = 0$, a water column is set in the left and right domains. At $t > 0$, waves from both domains are expected to interact continuously with each other across domains. The positions of the columns of water are chosen to be close to the interface, such that the perturbations cross the interface in a relatively short execution time.

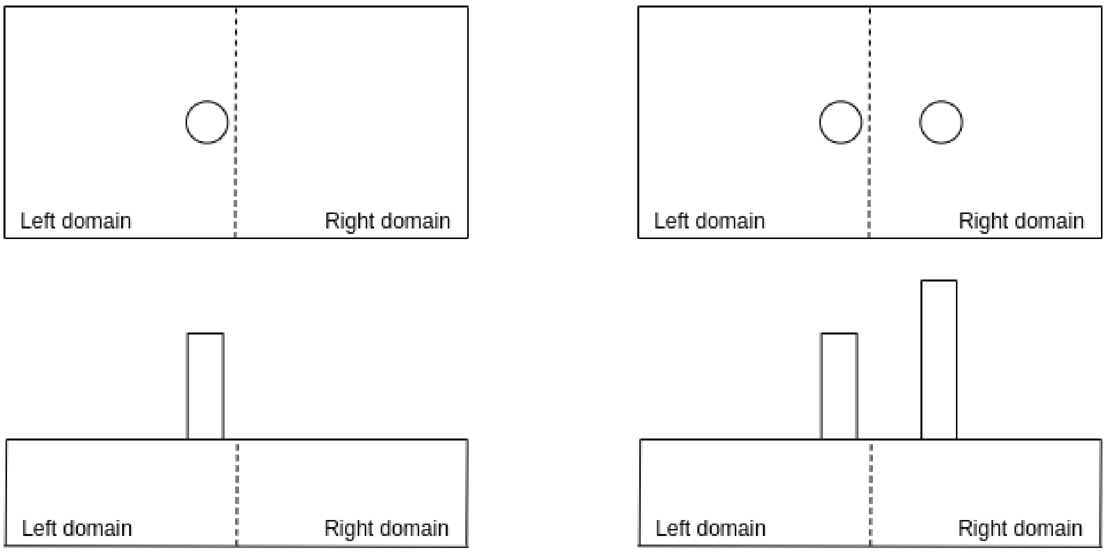


Figure 5.1.: Schematic for the SWE $\rightarrow$ SWE case at $t = 0$. The top-left and bottom-left figures show the top and side views, respectively, for supercritical flow. The right side shows the views for subcritical flow.

SWE $\rightarrow$ SWE	**Supercritical**		**Subcritical**	
	Left Domain 2D	**Right Domain 2D**	**Left Domain 2D**	**Right Domain 2D**
x length [m]	1000			
z length [m]	1000			
x resolution	120			
z resolution	120			
water at rest height [m] @$t = 0$	10			
water column height [m] @$t = 0$	15	-	15	20

Table 5.1.: Parameters for the left and right domains for supercritical and subcritical flow for SWE $\rightarrow$ SWE. *Water at rest height* refers to the level of water surrounding the water column.

Edge	Supercritical & subcritical			
	2D left		**2D right**	
	h		h	
Left	hu	`outflow`	hu	`inflow_couple`
	hv		hv	
	h		h	
Right	hu	`out_inflow_couple`	hu	`outflow`
	hv		hv	
	h		h	
Top	hu	`outflow`	hu	`outflow`
	hv		hv	
	h		h	
Bottom	hu	`outflow`	hu	`outflow`
	hv		hv	

Table 5.2.: Boundary conditions for SWE $\rightarrow$ SWE supercritical and subcritical flow. The naming of the boundary conditions (b.c.) corresponds to the naming convention for each solver. In this case, both scenarios use the same type of b.c, based on the way the implementation was done; *out_inflow_couple* b.c. is a special implementation, so that the solver automatically handles outflow or inflow boundary conditions depending on the flow characterization.

Figure 5.2 shows the results for the height h for the SWE $\rightarrow$ SWE supercritical case. The graphs show a comparison between the partitioned approach (two domains coupled) and the monolithic approach. On the middle graph at $t = 8.4$, a difference between both solutions is observed. The relative error is plotted in Figure 5.4. This discrepancy is probably due to an implementation error that could not be tracked, rather than a coupling error. This assumption is based on observing that the difference is still within the left domain, while a smooth transition of the solution across the interface can be observed. Later on $t = 36s$ this difference vanishes.

Figure 5.3 shows the results for the height for the SWE $\rightarrow$ SWE subcritical case. Similarly as in the supercritical case, the graphs on the right show a slight difference between the monolithic and partitioned approaches. The relative error is shown in Figure 5.4. At later times (not shown) both differences vanish, as in the supercritical case. The assumption for this difference is probably, again, an untracked error in the implementation as the solution across the domains behaves smoothly.

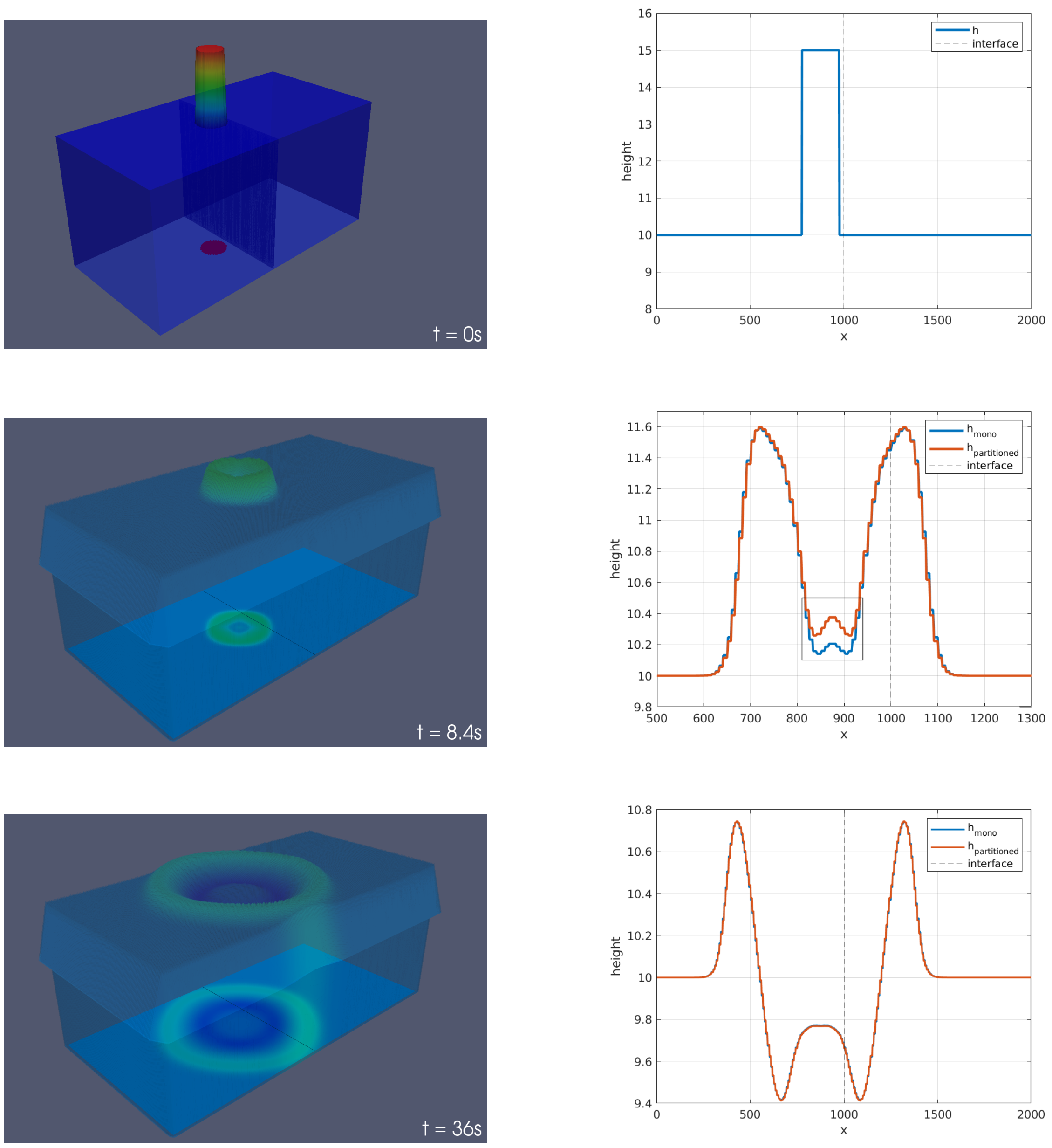


Figure 5.2.: SWE $\rightarrow$ SWE supercritical. Figures on the left show the results for the height h from the 2D solution, with a vertical projection. Notice a line and a plane delimiting the left and right domains. Top-most left figure shows simulation at time $t = 0$, and t increases downwards. The graphs compare monolithic and partitioned approaches. on the right side correspond to the images on the left, and include a comparison between the monolithic and partitioned approach. A square shows a difference between both approaches. The vertical projection is done with glyphs (the tip of the glyph is displayed, do not confuse as part of the solution).

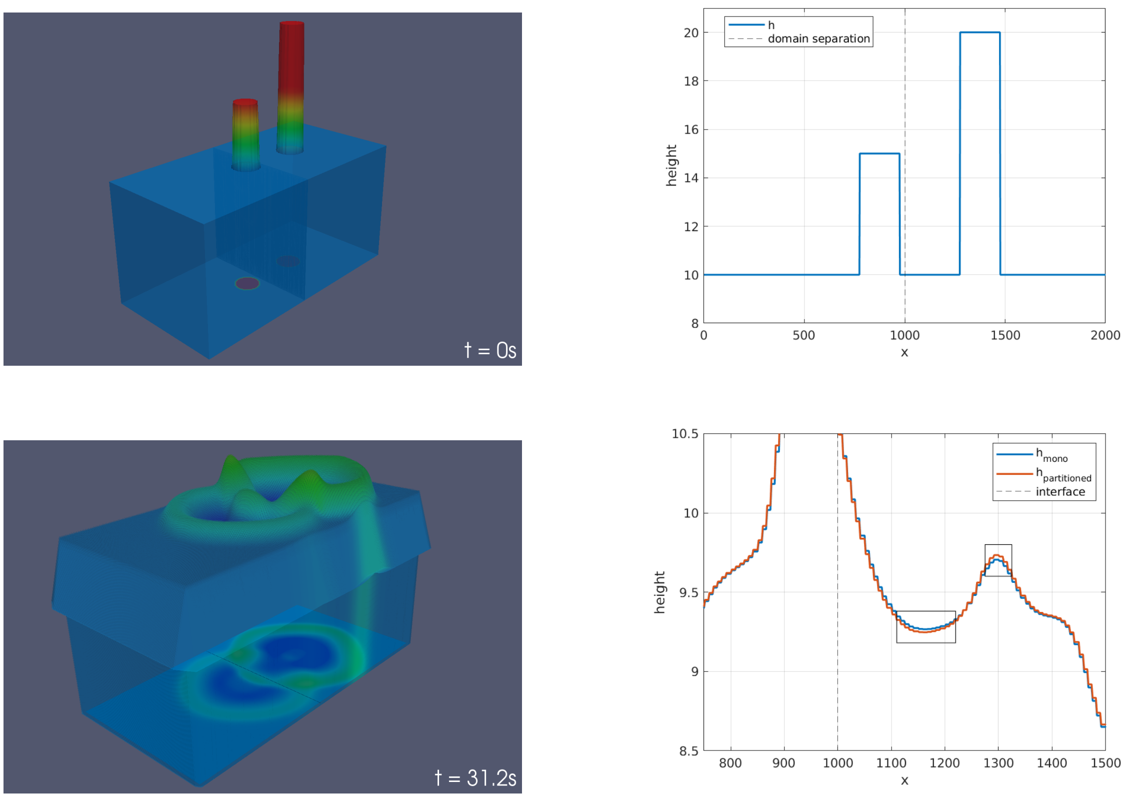


Figure 5.3.: SWE $\rightarrow$ SWE subcritical. Left figures show the 2D solution for the height h and a vertical projection at different times. Graphs on the right side show the corresponding comparisons between the monolithic and partitioned approach with a squares highlighting differences between both approaches. Note the interface line defining the left and right domains. The vertical projection is done with glyphs (the tip of the glyph is displayed, do not confuse as part of the solution).

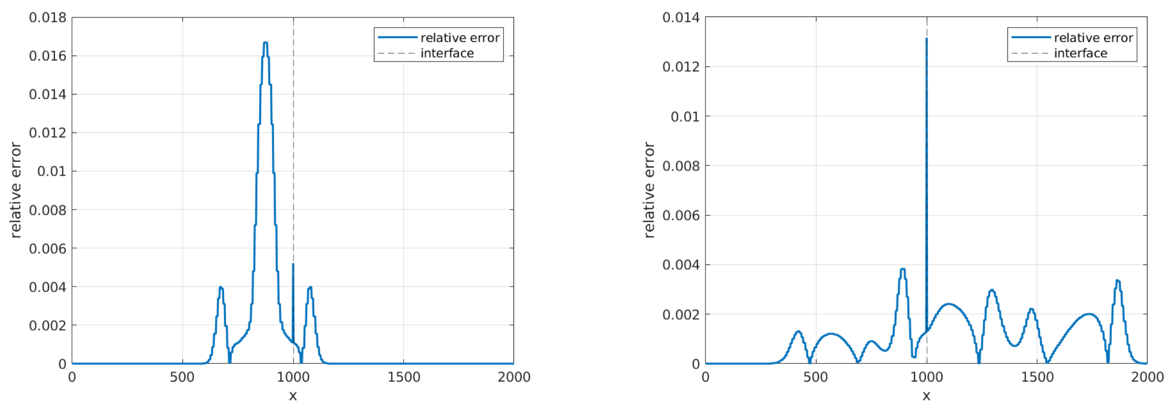


Figure 5.4.: Relative error, $e = \frac{|partitioned - mono|}{mono}$, of the supercritical and subcritical cases. Left: supercritical case at $t = 8.4s$. Right: supercritical case at $t = 31.2$.

5.2. SWE $\rightarrow$ OpenFOAM

For the next case, we address inter-dimensional coupling from a 2D to a 3D domain for supercritical and subcritical cases. The schematic for both cases is shown in Figure 5.5. In this case, the left domain corresponds to the 2D solver, while the right domain corresponds to the 3D solver. For the supercritical case, we set a water column at the middle of the left domain at $t = 0$, where the waves are expected to travel in all directions, crossing to the 3D domain. For the subcritical case, we simulate open-channel flow by setting inflow velocity and height on the left boundary of the 2D domain. We expect the flow to travel to the 3D domain, where wall-boundary conditions are set downstream. The flow is expected to reach the wall and flow back as a left-moving hydraulic jump, increasing the water level and crossing back to the 2D domain. Notice the two walls on the right domain: the intention for this is to create an area where the incoming fluid would not be affected in case of spilling occurs. This situation would not add significant information to the behaviour, but would rather add computation time. An alternative would be to implement outflow conditions, however, results show that no spilling occurred, so for future simulations this can be ignored, and we could consider the 3D domain until the first wall. The parameters for each solver are shown in Table 5.3, and Tables 5.4 and 5.5 show the boundary conditions for supercritical and subcritical flow, respectively.

SWE $\rightarrow$ OF	**Supercritical**		**Subcritical**	
	Left Domain 2D	**Right Domain 3D**	**Left Domain 2D**	**Right Domain 3D**
x length [m]	10			
y length [m]	-	10	-	10
z length [m]	10			
x resolution	30			
y resolution	-	30	-	30
z resolution	30			
water at rest height [m] @$t = 0$	5			
water column height [m] @$t = 0$	20	-	-	-

Table 5.3.: Parameters for the left and right domains for supercritical and subcritical flow for SWE $\rightarrow$ OF.

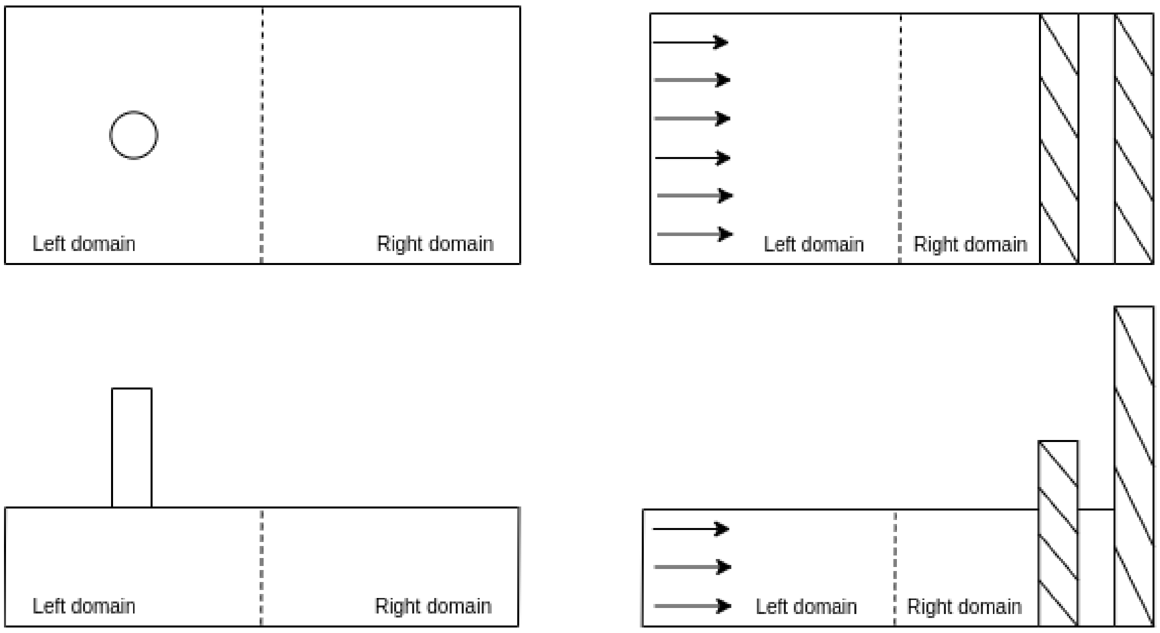


Figure 5.5.: Schematic for the SWE $\rightarrow$ SWE case at $t = 0$. The top-left and bottom-left figures show the top and side views, respectively, for supercritical flow. The right side shows the views for subcritical flow, where the horizontal arrows represent an initial velocity.

	Supercritical			
Edge	**2D**		**3D**	
Left	h	outflow	α	`fixedValue`*`: 0`
	hu		u	`fixedValue`*`: 0`
	hv		p_rgh	`fixedFluxPressure: 0`
Right	h	outflow	α	`zeroGradient`
	hu		u	`fixedGradient: 0`
	hv		p_rgh	`fixedFluxPressure: 0`
Top	h	outflow	α	`inletOutlet: 0`
	hu		u	`pressureInletOutletVelocity: 0`
	hv		p_rgh	`totalPressure: 0`
Bottom	h	outflow	α	`zeroGradient`
	hu		u	`noSlip`
	hv		p_rgh	`fixedFluxPressure: 0`
Front	-		α, u, p_rgh	`empty`
Back	-		α, u, p_rgh	`empty`

Table 5.4.: Boundary conditions for SWE $\rightarrow$ OF supercritical flow. The naming of the b.c. corresponds to the naming for each solver. Boundary conditions marked with (*) will be replaced by the exchanged data from preCICE.

Subcritical

Edge	**2D**		**3D**	
Left	h	`inflow: 5`	α	`zeroGradient`
	hu	`inflow: 2`	u	`fixedValue`*`: 0`
	hv	`inflow: 0`	p_rgh	`fixedFluxPressure: 0`
Right	h	`passive`	α	`zeroGradient`
	hu	`outflow`	u	`noSlip`
	hv	`outflow`	p_rgh	`fixedFluxPressure: 0`
Top	h		α	`inletOutlet: 0`
	hu	`wall`	u	`pressureInletOutletVelocity: 0`
	hv		p_rgh	`totalPressure: 0`
Bottom	h		α	`zeroGradient`
	hu	`wall`	u	`noSlip`
	hv		p_rgh	`fixedFluxPressure: 0`
Front		-	α	
			u	`empty`
			p_rgh	
Back		-	α	
			u	`empty`
			p_rgh	

Table 5.5.: Boundary conditions (b.c) for SWE → OF subcritical flow. The naming of the b.c. correspond to the naming for each solver. *passive* b.c. means that the variable will be updated when the exchange from the preCICE adapter takes place, rather than update them through the solver globally. Boundary conditions marked with a a (*) will be replaced by the exchanged data from preCICE.

Figure 5.6 shows the results for supercritical flow case. At $t = 0.45s$, we can see the perturbations reaching the 3D domain. The graph shows the height profiles for both domains, along the coupling interface (z axis pointing out of the screen). We observe that the height profiles are the same before and after the interface. At $t = 2.35s$, a second water column has fallen, and again perturbations are crossing again to the 3D domain, while the perturbations from the first water column continue their way downstream.

For the subcritical case, Figure 5.7 shows the perturbations from the initial conditions from the inlet travelling to the 3D domain. The grading of colour indicates the increment in water level as t increases. At $t = 1.7s$, we observe the water travelling back to the 2D domain, as it got reflected from the wall on the 3D domain. The left-moving-hydraulic jump can be observed at $t = 2.1$ (better depicted in Figure 5.8b, enclosed by the red square). The hydraulic jump continues to the left, as seen in Figure 5.8c, however, on its right side, the water level slightly decreases, and for $t > 2.5$ the water level oscillates around a certain height. These behaviours are not expected to happen, and they possibly mean some faults in the implementation.

Figure 5.8 shows the height across the interface between dimensions (along the x axis) in the middle of the domains. Black rectangles enclose discontinuities on the interface. On figure 5.8a the gap is small compared to those at later times, shown on Figures 5.8b and 5.8c. This behaviour certainly represents some flaws in the implementation for this case. However, aside from the discontinuities, we can observe a consistent concavity of the curve across the interface. This means that the computations of each solver could have present some sort of lagging.

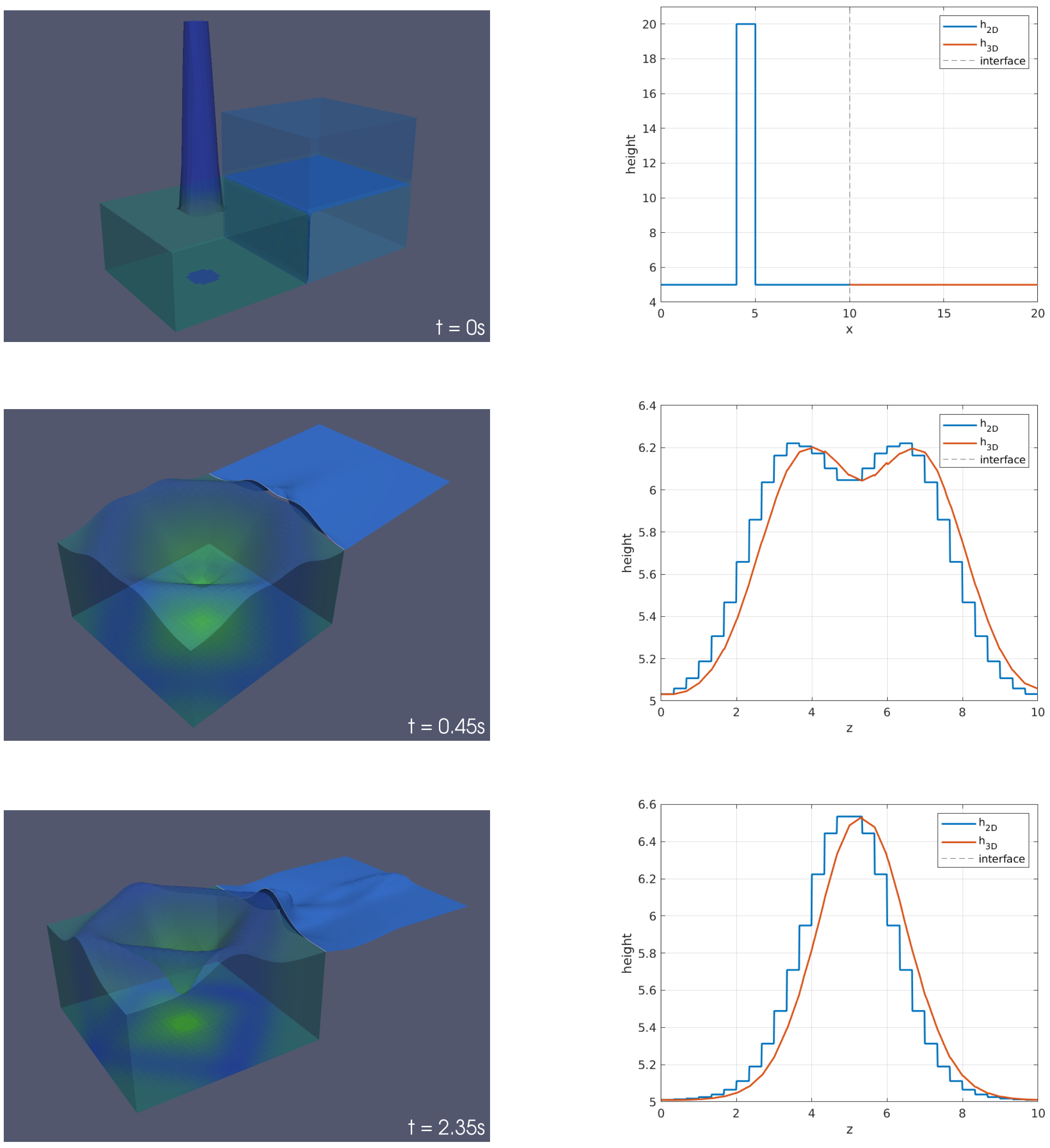


Figure 5.6.: SWE → OF supercritical. Images on the left show the solution for the height travelling to the 3D domain. The 2D domain is on the left side, with a vertical projection of the solution. Image at $t = 0s$ shows the 3D enclosed in a cube, while for the remaining images, only the free-surface is depicted. Graphs on the right correspond to the images on the left, and depict the height profile on the 2D and 3D domains along their interface on the z axis (out of the screen).

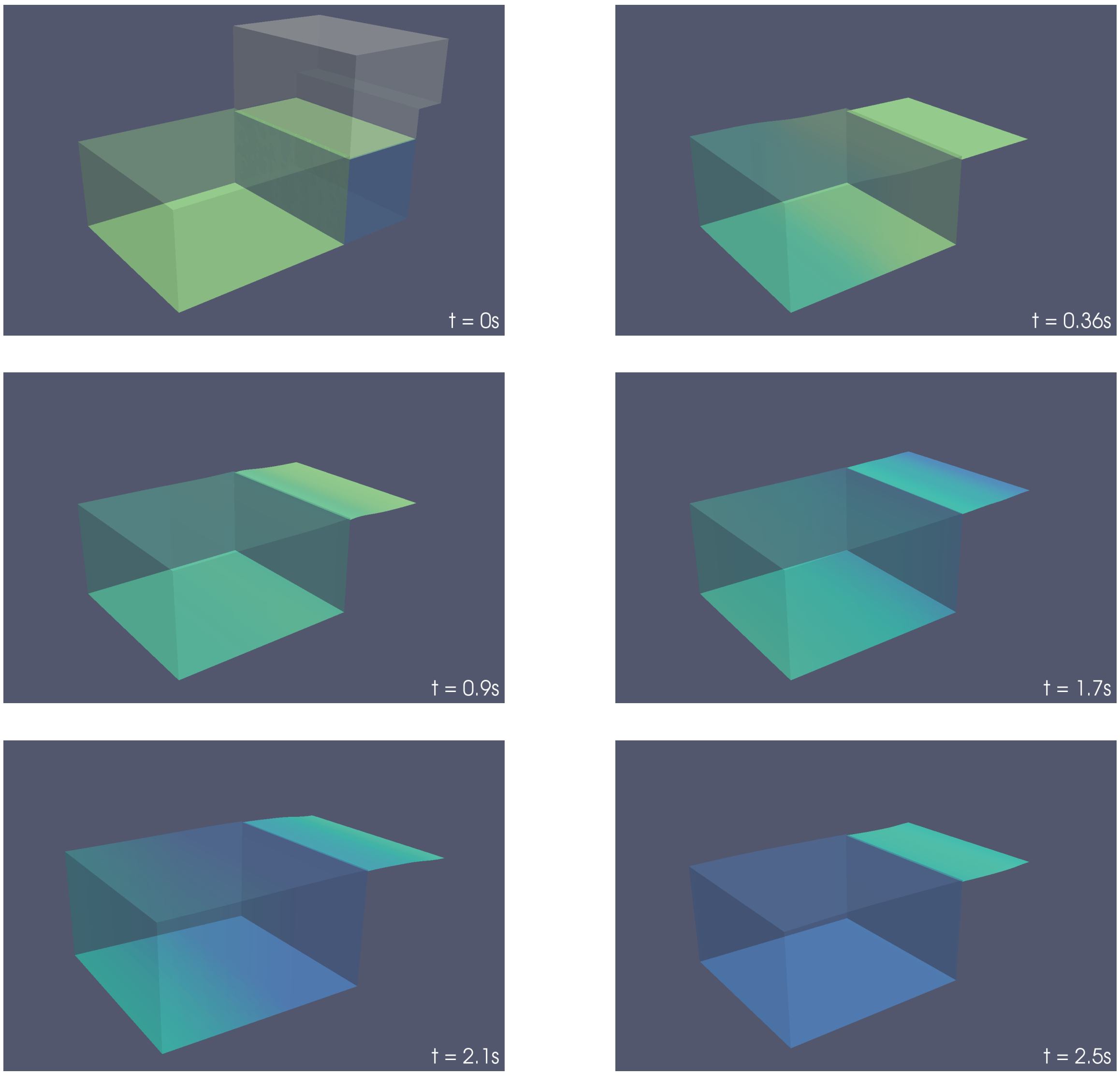
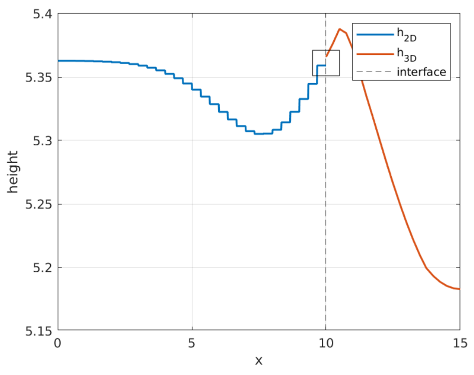
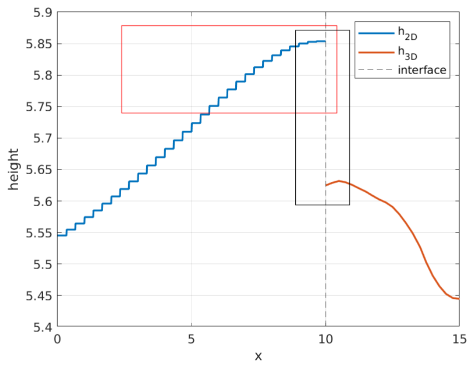


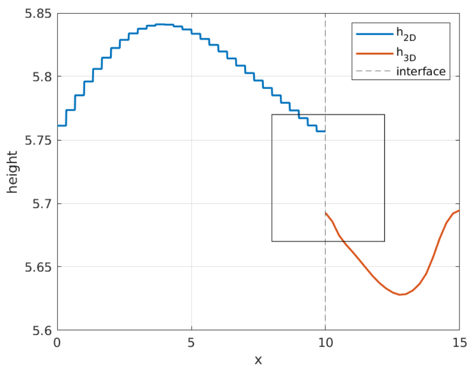
Figure 5.7.: SWE $\rightarrow$ OF subcritical. Sequence of the solution for the height. The 2D domain is on the left side of the images, and a vertical projection of the solution is shown. The 3D domain is on the right side. On the top-left figure we can see gray planes defining the 3D domain; for the resting images, only a contour indicating the free-surface is shown. At $t = 0$ flow starts from the left on the 2D domain towards the 3D domain on the right side. At $t = 0.36s$, we observe a color grading towards the 3D domain, indicating an increase of the water level. At $t = 1.7s$, the color grading occurs towards the left side, back to the 2D domain after reaching the wall at the right end.



(a) Height at $t = 0.9s$



(b) Height at $t = 2.1s$



(c) Height at $t = 2.5s$

Figure 5.8.: SWE $\rightarrow$ OF subcritical. Graphs show the height values across the 2D and 3D dimensions at different times. Black rectangles enclose discontinuities on the interface. The red rectangle points the developing left-moving hydraulic jump.

5.3. OpenFOAM $\rightarrow$ SWE

The last inter-dimensional case is coupling a 3D domain with a 2D domain. In this case, the 3D domain is on the left side, while the 2D domain on the right side. The setup for supercritical and subcritical flow cases are the same as shown in Figure 5.5, with the difference that in this case the 3D domain is on the left side and we different height for the water column. Details about these parameters are shown in Table 5.6, and the boundary conditions for both cases are shown in Tables 5.7 and 5.8.

OF $\rightarrow$ SWE	**Supercritical**		**Subcritical**	
	Left Domain 3D	**Right Domain 2D**	**Left Domain 3D**	**Right Domain 2D**
x length [m]	10		7	
y length [m]	-	10	7	-
z length [m]	10		5	
x resolution	30		30	20
y resolution	30	-	-	15
z resolution	30		40	
water at rest height [m] @$t = 0$	5		2	
water column height [m] @$t = 0$	10	-	-	-

Table 5.6.: Parameters for the left and right domains for supercritical and subcritical flow cases for OF $\rightarrow$ SWE.

	Supercritical			
Edge	**2D**		**3D**	

Left	h
	hu
	hv
		`passive`	α
			u
			p_rgh
				`fixedValue: 0`
				`noSlip`
				`fixedFluxPressure: 0`

Right	h
	hu
	hv
		`outflow`	α
			u
			p_rgh
				`zeroGradient`
				`fixedGradient: 0`
				`fixedFluxPressure: 0`

Top	h
	hu
	hv
		`outflow`	α
			u
			p_rgh
				`inletOutlet: 0`
				`pressureInletOutletVelocity: 0`
				`totalPressure: 0`

Bottom	h
	hu
	hv
		`outflow`	α
			u
			p_rgh
				`zeroGradient`
				`noSlip`
				`fixedFluxPressure: 0`

Front	-		α u p_rgh	`empty`
Back	-		α u p_rgh	`empty`

Table 5.7.: Boundary conditions for OF → SWE supercritical flow. The naming of the b.c. corresponds to the naming for each solver.

	Subcritical			
Edge	**2D**		**3D**	
Left	h hu hv	`passive`	see Table 5.9	

Right	h
	hu
	hv
		`wall`	α
			u
			p_rgh
				`zeroGradient`
				`fixedGradient: 0`
				`fixedValue: 0`

Top	h
	hu
	hv
		`wall`	α
			u
			p_rgh
				`inletOutlet: 0`
				`pressureInletOutletVelocity: 0`
				`totalPressure: 0`

Bottom	h
	hu
	hv
		`wall`	α
			u
			p_rgh
				`zeroGradient`
				`noSlip`
				`fixedFluxPressure: 0`

Front	-		α u p_rgh	`empty`
Back	-		α u p_rgh	`empty`

Table 5.8.: Boundary conditions for OF → SWE subcritical flow. The naming of the b.c. correspond to the naming for each solver. The left wall for the 3D case is divided on two sections: *left_air* and *left_water*, so as to have an open-channel scenario with constant inflow velocity.

left_air	α
	u
	p_rgh
		`zeroGradient`
		`fixedValue: 0`
		`fixedFluxPressure: 0`

left_water	α
	u
	p_rgh
		`fixedValue: 1`
		`fixedValue: (2.6,0,0)`
		`fixedFluxPressure: 0`

Table 5.9.: Boundary conditions for the left side on the 3D domain for OF → SWE subcritical flow. The left side is divided in two sections: *left_air* and *left_water* in order to have an open-channel scenario with constant velocity.

Figure 5.9 shows the result for the supercritical case. The scenario initially starts from a water column surrounded by water at rest. After one second, the waves from the water column crossed to the 2D domain. Later at $t = 7.2s$, the reflection of the waves on the left boundary of the 3D domain travel across the interface. From the graphs, we can observe some discontinuities in the solution. We also observe a different concavity on the solutions.

For the subcritical case, Figure 5.10 shows a sequence of the flow crossing to the 2D domain, and later reflected back to the 3D domain, similarly as in the SWE $\rightarrow$ OF subcritical case. We can observe some discontinuities from the graphs on the right side of the figure. Initially, when the fluid goes from the 3D domain to the 2D domain, the jump in the discontinuity seems relatively small. Later, after being reflected as the fluid crosses back to the 3D domain, the jump increases notably, and for later times the gap seems to reduce. We can therefore assume that the boundary conditions and mapping from the 2D to the 3D domain might contain some flaws on the implementation.

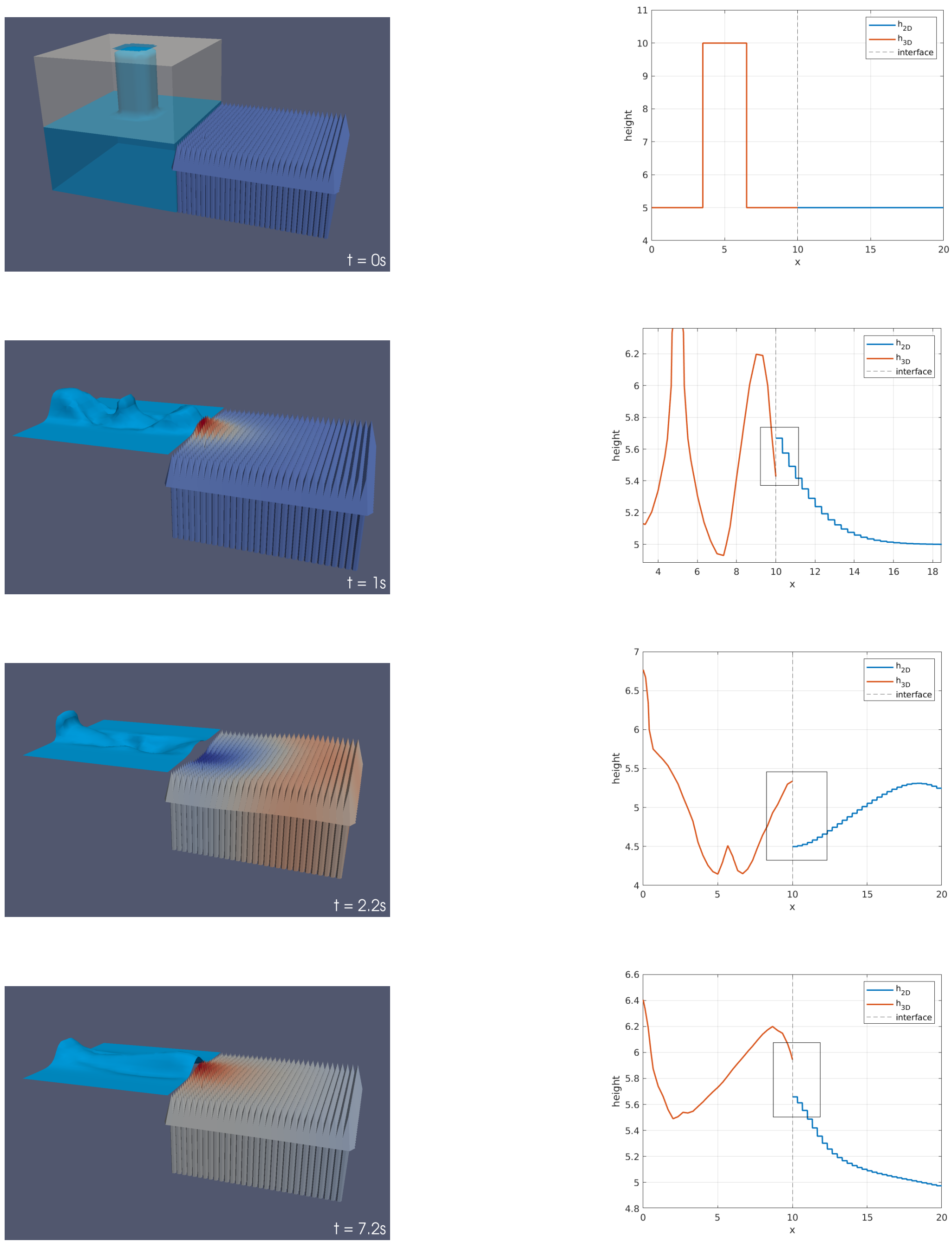


Figure 5.9.: OF → SWE supercritical. Solution of the height at different times. Graphs on the right show the water level across the 3D and 2D domains, and correspond to figure on the left. Black rectangles show discontinuities of the height level on the interface. Note that image at $t = 0s$ shows the 3D domain, while the rest of the images depict the free-surface. The solution on the 2D domain (right) uses glyphs for better visualization (the tip of the glyph is displayed. Do not confuse as part of the solution).

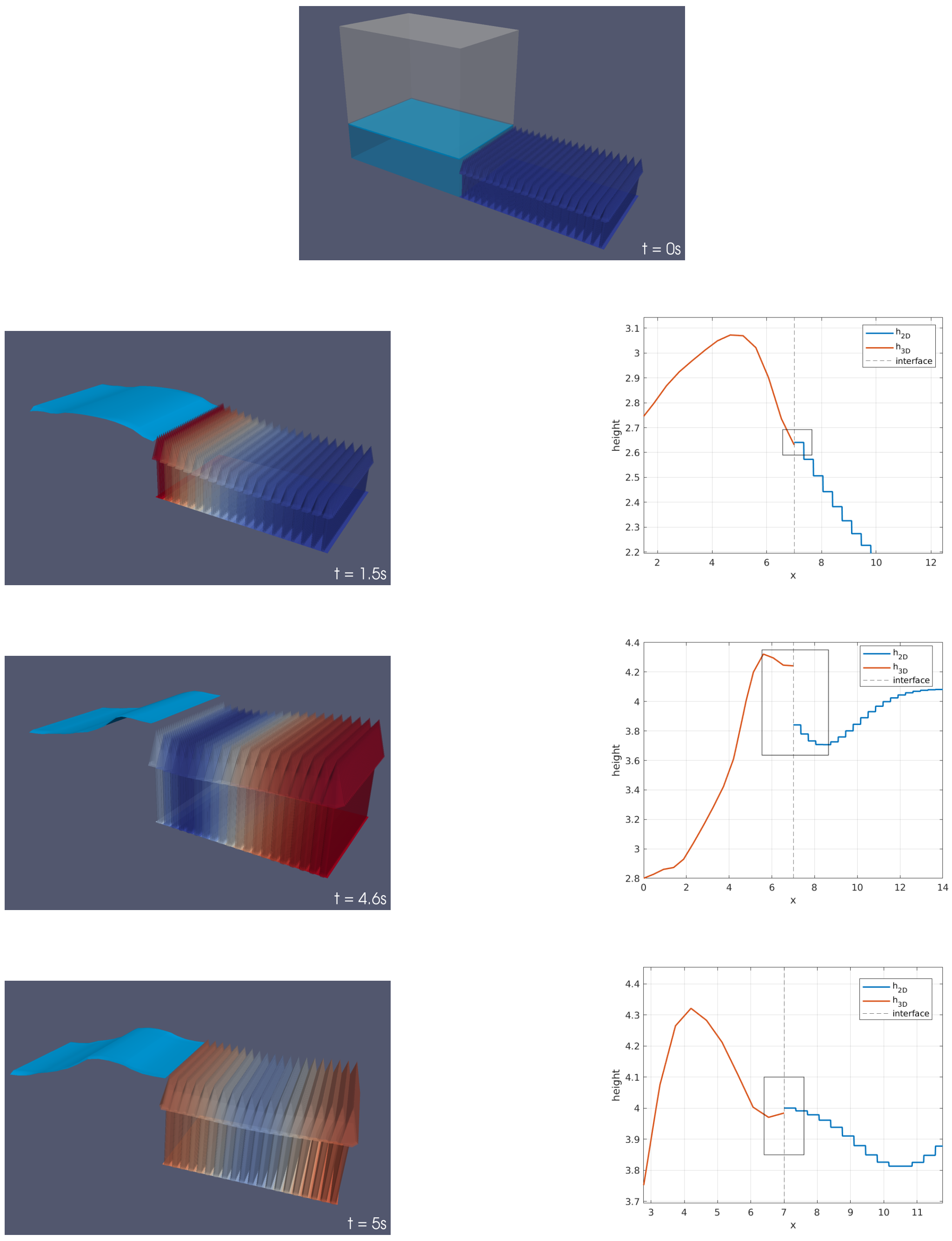


Figure 5.10.: OF → SWE subcritical case. Solution of the height. Image at $t = 0$ shows the 3D dimension on the left side, displaying the boundaries in gray color. An open-channel scenario is considered in this case. Flow going from the left(3D) to the right(2D) domain. At $t = 1.5s$ the flow crosses to the 2D domain. At time $t = 4.6s$, the flow travels back to the 3D domain after being reflected. At time $t = 5s$, the flow has crossed back to the 3D domain. Black rectangles in the graphs on the right side depict the discontinuities on the height. For better visualization we use glyphs (the tips of the glyphs are displayed. Do not confuse as part of the solution).

5.4. OpenFOAM → OpenFOAM

5.4.1. Supercritical

As a final case, we address the OF → OF case for supercritical flow. For this scenario, we consider a modified breaking-dam case scenario by adding a water column on top. The schematic for the initial configuration is shown in Figure 5.11. There is no particular reason for choosing this setup, other than slightly modifying the setup from the OpenFOAM tutorial[1] with preCICE as the coupling resource. We aim to compare this partitioned approach with monolithic approach. The parameters and boundary conditions for the partitioned configuration are shown in Tables 5.10 and 5.11, respectively.

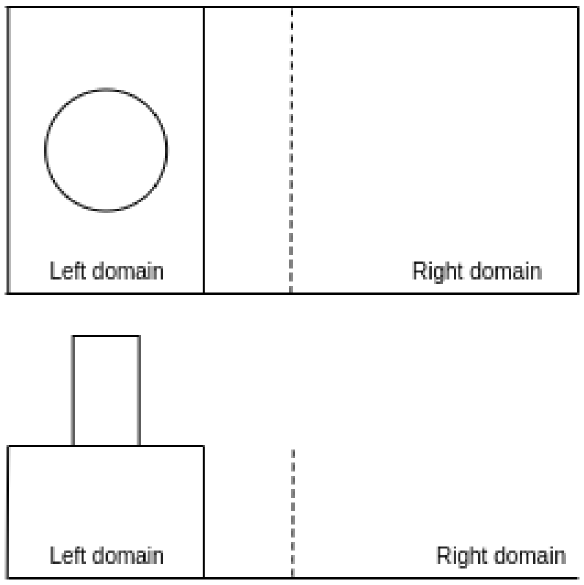


Figure 5.11.: Schematic for the OF → OF case at $t = 0$. The top and bottom figures show the top and side views respectively for supercritical flow.

[1] https://cfd.direct/openfoam/user-guide/v6-damBreak/

OF → OF	**Supercritical**	
	Left Domain 2D	**Right Domain 3D**
x length [m]	10	
z length [m]	10	
x resolution	20	
y resolution	20	
z resolution	20	
water at rest height [m] @$t = 0$	10	
water column height [m] @$t = 0$	10	-

Table 5.10.: Parameters for the left and right domains for supercritical and subcritical flow for SWE → SWE. *Water at rest height* refers to the level of water surrounding the water column.

	Supercritical			
Edge	**3D left**		**3D right**	
Left	α	`zeroGradient`	α	`fixedValue: 0`
	u	`noSlip`	u	`fixedValue: 0`
	p_rgh	`fixedFluxPressure: 0`	p_rgh	`fixedFluxPressure: 0`
Right	α	`zeroGradient`	α	`zeroGradient`
	u	`fixedGradient: 0`	u	`fixedGradient: 0`
	p_rgh	`fixedValue: 0`	p_rgh	`fixedValue: 0`
Top	α	`inletOutlet: 0`	α	`inletOutlet: 0`
	u	`pressInOutVelocity: 0`	u	`pressInOutVelocity: 0`
	p_rgh	`totalPressure: 0`	p_rgh	`totalPressure: 0`
Bottom	α	`zeroGradient`	α	`zeroGradient`
	u	`noSlip`	u	`noSlip`
	p_rgh	`fixedFluxPressure: 0`	p_rgh	`fixedFluxPressure: 0`
Front	α		α	`fixedValue: 0`
	u	`empty`	u	`fixedValue: 0`
	p_rgh		p_rgh	`fixedFluxPressure: 0`
Back	α		α	`fixedValue: 0`
	u	`empty`	u	`fixedValue: 0`
	p_rgh		p_rgh	`fixedFluxPressure: 0`

Table 5.11.: Boundary conditions for OF $\rightarrow$ OF supercritical flow. The naming of the b.c. correspond to the naming for each solver.

Figure 5.12 shows the results for the supercritical case. At $t = 1.7s$, we see that the fluid has already crossed to the right domain. Figure 5.13 shows a comparison of the state variables between the monolithic and partitioned approaches at $t = 1.7s$. We observe that the solutions are slightly different, and they keep the same shape most of the time.

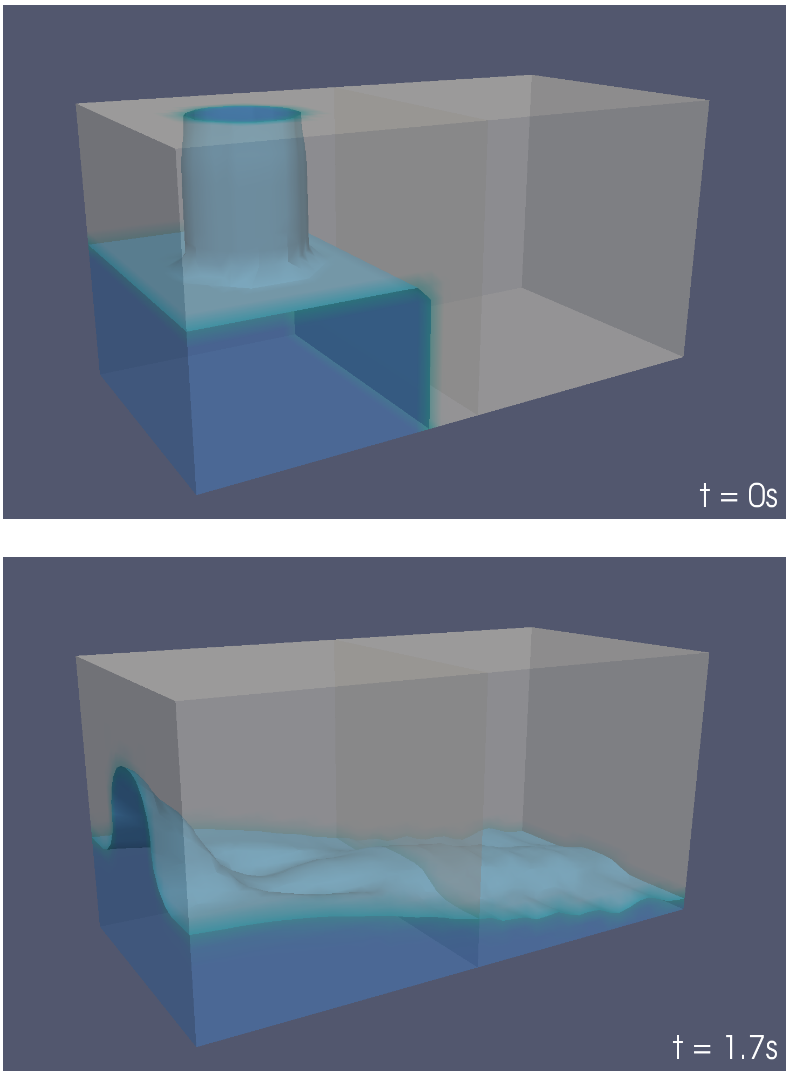


Figure 5.12.: OF $\rightarrow$ OF supercritical. Simulation results at $t = 0s$ and $t = 1.7s$ of the partitioned approach.

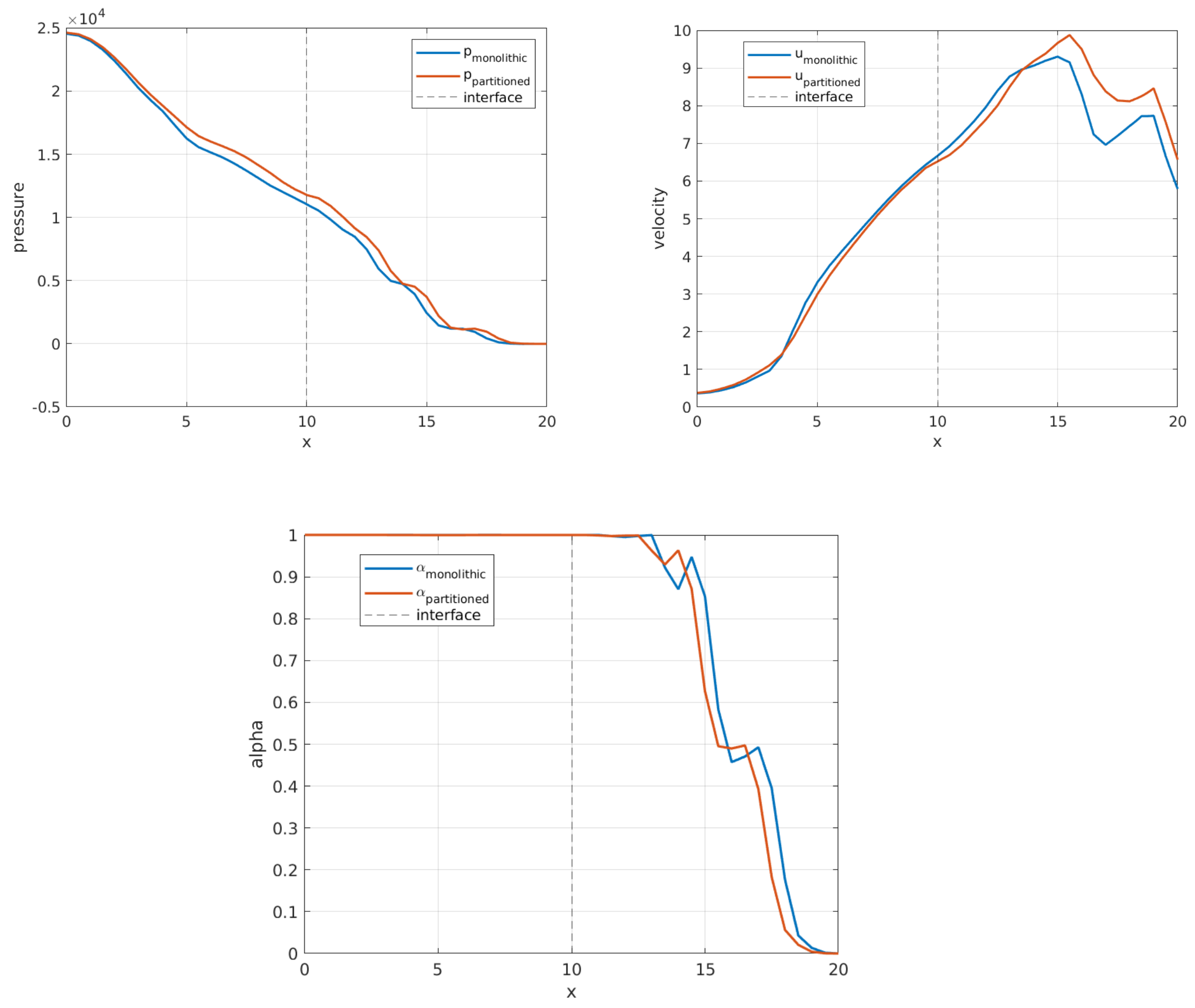
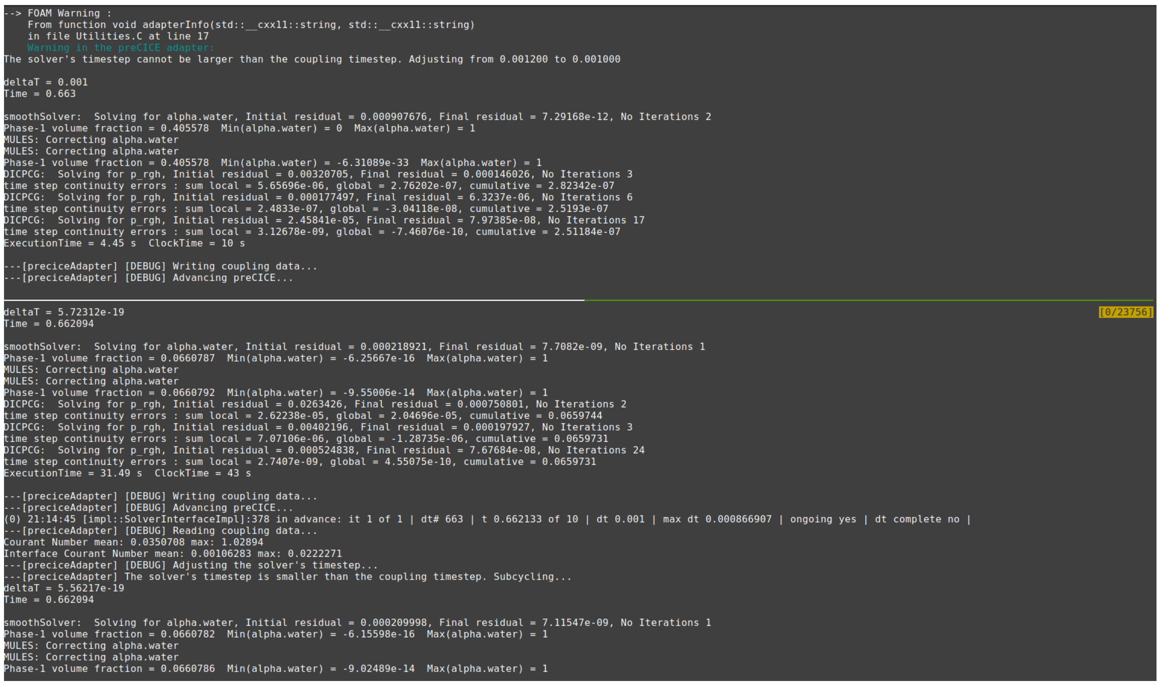


Figure 5.13.: OF $\rightarrow$ OF supercritical graphs. Comparison of the state variables p, u, α between the monolithic and partitioned approach at $t = 1.7s$.

5.4.2. Subcritical

For this case, we selected the same scenario as for the supercritical case, shown in Figure 5.11. For enforcing subcritical flow, this time we imposed wall boundary conditions on the right side of the right domain. In this way, the incoming flow would be reflected back towards the interface. In this case, we lack of a solid theoretical foundation for setting the correct boundary conditions on the interface. We tried a number of combinations, among Dirichlet and Neumann boundary conditions, resulting in unphysical results. We also tried setting the boundary conditions based on the configuration of the already-working OF-SWE and SWE-OF subcritical cases, however, we mostly ran into incompatibilities for the boundary conditions in the OpenFOAMs adapter. Lastly, we tried the same boundary conditions configuration as in the supercritical case. In this case, we observed that the solvers would continue to execute normally, delivering consistent results as the simulation was executing. However, after the flow was reflected back and reaching the interface, the adaptive time stepping from OpenFOAM would decrease to the order of 10^{-19}, resulting unfeasible to obtain any conclusive results. Figure 5.14 shows a screenshot of the output of the solver.



```
--> FOAM Warning :
    From function void adapterInfo(std::__cxx11::string, std::__cxx11::string)
    in file Utilities.C at line 17
    Warning in the preCICE adapter:
The solver's timestep cannot be larger than the coupling timestep. Adjusting from 0.001200 to 0.001000

deltaT = 0.001
Time = 0.663

smoothSolver:  Solving for alpha.water, Initial residual = 0.000907676, Final residual = 7.29168e-12, No Iterations 2
Phase-1 volume fraction = 0.405578  Min(alpha.water) = 0  Max(alpha.water) = 1
MULES: Correcting alpha.water
MULES: Correcting alpha.water
Phase-1 volume fraction = 0.405578  Min(alpha.water) = -6.31089e-33  Max(alpha.water) = 1
DICPCG:  Solving for p_rgh, Initial residual = 0.00320705, Final residual = 0.000146026, No Iterations 3
time step continuity errors : sum local = 5.65696e-06, global = 2.76202e-07, cumulative = 2.82342e-07
DICPCG:  Solving for p_rgh, Initial residual = 0.000177497, Final residual = 6.3237e-06, No Iterations 6
time step continuity errors : sum local = 2.4833e-07, global = -3.04118e-08, cumulative = 2.5193e-07
DICPCG:  Solving for p_rgh, Initial residual = 2.45841e-05, Final residual = 7.97385e-08, No Iterations 17
time step continuity errors : sum local = 3.12678e-09, global = -7.46076e-10, cumulative = 2.51184e-07
ExecutionTime = 4.45 s  ClockTime = 10 s

---[preciceAdapter] [DEBUG] Writing coupling data...
---[preciceAdapter] [DEBUG] Advancing preCICE...

deltaT = 5.72312e-19                                                                                                          [0/23756]
Time = 0.662094

smoothSolver:  Solving for alpha.water, Initial residual = 0.000218921, Final residual = 7.7082e-09, No Iterations 1
Phase-1 volume fraction = 0.0660787  Min(alpha.water) = -6.25667e-16  Max(alpha.water) = 1
MULES: Correcting alpha.water
MULES: Correcting alpha.water
Phase-1 volume fraction = 0.0660792  Min(alpha.water) = -9.55006e-14  Max(alpha.water) = 1
DICPCG:  Solving for p_rgh, Initial residual = 0.0263426, Final residual = 0.000750801, No Iterations 2
time step continuity errors : sum local = 2.62238e-05, global = 2.04696e-05, cumulative = 0.0659744
DICPCG:  Solving for p_rgh, Initial residual = 0.00402196, Final residual = 0.000197927, No Iterations 3
time step continuity errors : sum local = 7.07106e-06, global = -1.28735e-06, cumulative = 0.0659731
DICPCG:  Solving for p_rgh, Initial residual = 0.000524838, Final residual = 7.67684e-08, No Iterations 24
time step continuity errors : sum local = 2.7407e-09, global = 4.55075e-10, cumulative = 0.0659731
ExecutionTime = 31.49 s  ClockTime = 43 s

---[preciceAdapter] [DEBUG] Writing coupling data...
---[preciceAdapter] [DEBUG] Advancing preCICE...
(0) 21:14:45 [impl::SolverInterfaceImpl]:378 in advance: it 1 of 1 | dt# 663 | t 0.662133 of 10 | dt 0.001 | max dt 0.000866907 | ongoing yes | dt complete no |
---[preciceAdapter] [DEBUG] Reading coupling data...
Courant Number mean: 0.0350708 max: 1.02894
Interface Courant Number mean: 0.00106283 max: 0.0222271
---[preciceAdapter] [DEBUG] Adjusting the solver's timestep...
---[preciceAdapter] The solver's timestep is smaller than the coupling timestep. Subcycling...
deltaT = 5.56217e-19
Time = 0.662094

smoothSolver:  Solving for alpha.water, Initial residual = 0.000209998, Final residual = 7.11547e-09, No Iterations 1
Phase-1 volume fraction = 0.0660782  Min(alpha.water) = -6.15598e-16  Max(alpha.water) = 1
MULES: Correcting alpha.water
MULES: Correcting alpha.water
Phase-1 volume fraction = 0.0660786  Min(alpha.water) = -9.02489e-14  Max(alpha.water) = 1
```

Figure 5.14.: Output for OF → OF case for both solvers. Time step in the order of 10^{-19} for the right solver (below the green line). Other parameters, like the Courant number, are displayed.

6. Conclusions

In this thesis, we have coupled a 2-dimensional Shallow Water Equations (SWE) model with a 3-dimensional Navier-Stokes model for free-surface flow. The 2D model was solved using a Riemann Problem solver written in C++ and developed at the TUM Scientific Computing Chair, while the 3D model used Volume-Of-Fluid method implemented by OpenFOAM, a CFD open-source framework. Both models were coupled with the preCICE coupling library.

The methods and experimentation discussed and developed in this thesis, contributed to the multi-physics simulation field, particularly for Fluid-Fluid interaction problems. Traditional Fluid-Fluid interaction problems are solved as a domain-partitioned approach, and are performed within the same dimension. Our simulations make use of the domain-partitioned approach, however, in this case, the domain-partitioning is done across dimensions.

In this thesis, we designed simulations for 4 cases that resulted from a combination of two domains of different dimensions: 2D-2D, 2D-3D, 3D-2D, 3D-3D. Because of the nature of free-surface flow, each of the simulations was performed for supercritical and subcritical flow, giving a total of 8 scenarios to simulate. We prepared an appropriate environment for coupling the 2D and 3D solvers together. On the 3D side, we extended the preCICE adapter for OpenFOAM, by adding new variables for data mapping across dimensions. On the 2D side, we developed a preCICE adapter in such a way that the methods from the adapter are decoupled from the general solver implementation. This feature allows us for future research to treat each case independently, without affecting the implementation of the remaining cases, and also allows us to add more cases relatively easily.

One of the main variables of interest in free-surface flow is the fluid level. Our experiments focused on obtaining a solution for the free-surface height in each domain. In the 2D-2D, 2D-3D and 3D-2D, results are qualitatively good and overall consistent with expected physical behaviour, with potential for improvement. In some cases, we encountered discontinuities at the interface between the domains, and some of these show a consistent behaviour before and after the interface, that is, the height curves show the same concavity across the interface. We observe that discontinuities are more prominent depending on the flow characterization; in this case, subcritical cases represent the bigger challenge.
For the 3D-3D case, we implemented a number of configurations for coupling the domains.

In some cases, the results would show unphysical results. In other cases, OpenFOAM would adapt the time step to the order of 10^{-19}, making the simulation unfeasible, thus lacking of conclusive results. In none of the attempts we could obtain a successful simulation for this case. For future research, taking a deeper look into appropriate boundary conditions on the the interface might take us a step closer to a successful simulation.

From our results and observations we can conclude that it is possible to successfully couple at least two different solvers across different dimensions, therefore adding more simulation options and flexibility for Fluid-Fluid interaction problems. From the presented methodology, and from the resorted tools, we also conclude that it is possible to add even more flexibility, by adding adding more than two domains, with more than two solvers on more than two different dimensions.

6.1. Future research

The aforementioned discontinuities are certainly undesirable and some propositions can be made for resolving them for future research. For instance, the SWE model omitted the viscosity, thus disregarding bottom and wall roughness. OpenFOAM, on the other hand, does consider it. We see a clear inconsistency across models that could be addressed.

In terms of mapping data across dimensions, we only considered exchanging information along a common edge with the same resolution. Developing mapping techniques across edges with different resolutions, or even across different elements (e.g. surface to edge), would add more flexibility for coupling a broader set of scenarios.

In the direction of making full use of the SWE capabilities, it is possible to add source terms such as sea-floor elevation as new scenarios. In this work, the floor elevation was set to 0. Proposing interesting sea-floor elevation profiles can be another direction for future research.

Appendix

A. Detailed Descriptions

A.1. Running the Code

In this section, we describe the general steps for building and running the simulations.

Once OpenFOAM, a GCC compiler and SCons(as the building interface), have been installed, we can download the implementation repository[1], and follow the next steps. For this thesis, we used OpenFOAM7.

1. Install the preCICE library[2]
2. Build the OpenFOAM adapter [3]
3. In the repository, go to tutorials/FF/dambreak. We will find a list of directories containing the cases. In the cases for running 2D-3D and 3D-2D scenarios, we will find the *runSWE* and *runIF* scripts for running the cases. Run each of them on separate terminals. For running cases for 2D-2D, the scripts *runSWE1* and *runSWE2*, and for the 3D-3D cases, the scripts *runIFLeft* and *runIFRight*. Once the simulation has finished, we can find the results under the *IF<case>* and *SWE_output<case>* subdirectories. We used *Paraview*[4] to visualize the results.

[1] `https://github.com/pachesp/2d3dcoupling` or `https://github.com/precice/openfoam-adapter/tree/SWE-interFoam`
[2] `https://github.com/precice/precice`
[3] `https://github.com/precice/openfoam-adapter/wiki/Building`
[4] `https://www.paraview.org/`

Bibliography

[1] D.S. Bale, R.J. LeVeque, S. Mitran, and J.A. Rossmanith. A wave propagation method for conservation laws and balance laws with spatially varying flux functions. *SIAM Journal on Scientific Computing*, 24(3):955–978, 2002.

[2] Alexander Breuer and Michael Bader. Teaching Parallel Programming Models on a Shallow-Water Code, Proceedings - 2012 11th International Symposium on Parallel and Distributed Computing, ISPDC 2012. pages 301–308, 06 2012.

[3] Hans-Joachim Bungartz, Florian Lindner, Bernhard Gatzhammer, Miriam Mehl, Klaudius Scheufele, Alexander Shukaev, and Benjamin Uekermann. preCICE – A fully parallel library for multi-physics surface coupling. *Computers and Fluids*, 141:250–258, 2016. Advances in Fluid-Structure Interaction.

[4] Y.A. Cengel and J.M. Cimbala. *Fluid mechanics: Fundamentals and applications*. McGraw-HillHigher Education, 2006.

[5] Gerasimos Chourdakis. A general OpenFOAM adapter for the coupling library preCICE. Master's thesis, Technische Universitaet Muenchen, 2017.

[6] Gerasimos Chourdakis, Benjamin Uekermann, Gertjan van Zwieten, and Harald van Brummelen. Coupling openfoam to different solvers, physics, models, and dimensions using precice. In *14th OpenFOAM Workshop*, Duisburg, Germany, Jul 2019.

[7] Osse Christiaan. Geometric Multi-Scale Flooding Simulations, 2020.

[8] Flood Manager E-Learning. Depth-averaged shallow water equations. http://daad.wb.tu-harburg.de/?id=1492. Accessed: 2020-06-14.

[9] Bernhard Gatzhammer. *Efficient and Flexible Partitioned Simulation of Fluid-Structure Interactions*. Dissertation, Technische Universitaet Muenchen, 09 2014.

[10] C W Hirt and B D Nichols. Volume of fluid (VOF) method for the dynamics of free boundaries. *J. Comput. Phys.; (United States)*, 39:1, 1 1981.

[11] Culbert B. Laney. *Computational Gasdynamics*. Cambridge University Press, 1998.

[12] Randall J. LeVeque. *Finite Volume Methods for Hyperbolic Problems*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 2002.

[13] Randall J Leveque. A Well-Balanced Path-Integral f-Wave Method for Hyperbolic Problems with Source Terms. *Journal of scientific computing*, 48(1-3):209226, July 2011.

[14] Goeran Florian Mintgen. *Coupling of Shallow and Non-Shallow Flow Solvers An Open Source Framework*. Dissertation, Technische Universitaet Muenchen, Muenchen, 2018.

[15] Shahbudin BS Muhammad SBAH, Mohd ZBM. Complete derivation of 2D Shallow-Water model from the primitive equations governing geophysical flows. Technical report, Institute of Oceanography and Maritime Studies, International Islamic University Malaysia, 2222.

[16] Michael Bader CSCS-FoMICS-USI Summer School on Computer Simulations in Science and Engineering. Shallow water equations - sccs. https://www5.in.tum.de/wiki/index.php/SWE. Accessed: 2020-09-28.

[17] Michael Bader CSCS-FoMICS-USI Summer School on Computer Simulations in Science and Engineering. Swe anatomy of a parallel shallow water code. https://www5.in.tum.de/SWE/doxy/index.html. Accessed: 2020-05-14.

[18] P. L. Roe. Approximate riemann solvers, parameter vectors, and difference schemes. *J. COMP. PHYS*, 43:357–372, 1981.

[19] Santiago Mrquez Damin. Computational Fluid Dynamics. http://infofich.unl.edu.ar/upload/3be0e16065026527477b4b948c4caa7523c8ea52.pdf.

[20] R.M. Sorensen. *Basic Wave Mechanics: For Coastal and Ocean Engineers*. A Wiley-Interscience publication. Wiley, 1993.

[21] Benjamin Uekermann. *Partitioned Fluid-Structure Interaction on Massively Parallel Systems*. PhD thesis, Technische Universitaet Muenchen, 10 2016.