SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Master's Thesis in Informatics

Coupling Spectral Shallow-Water Equations and a 3D Finite Volume Solver to Simulate Complex Geometry

Armin Ettenhofer

SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Master's Thesis in Informatics

Coupling Spectral Shallow-Water Equations and a 3D Finite Volume Solver to Simulate Complex Geometry

Kopplung spektraler Flachwassergleichungen mit einem 3D Finite Volumen Solver zur Simulation komplexer Hindernisse

Author:	Armin Ettenhofer
Examiner:	Prof. Dr. Hans-Joachim Bungartz
Supervisors:	M.Sc. Keerthi Gaddameedi, M.Sc. (hons) Benjamin Rodenberg
Submission Date:	16.02.2025

I confirm that this master's thesis is my own work and I have documented all sources and material used.

[illegible]

Munich, 16.02.2025 Armin Ettenhofer

Abstract

This thesis couples the spectral Shallow-Water Equations (SWE) solver SWEET with the 3D finite volume solver OpenFOAM. An atmosphere-ocean and an ocean-ocean-coupled solver are constructed using the preCICE coupling library. In this context, the challenges of 2D-3D coupling two solvers using different simulation schemes and the challenges arising from SWEET's spectral discretization are examined. The atmosphere-ocean solver is briefly analyzed in a proof-of-concept scenario, though limited to one-way coupling. The main part of this work then focuses on the two-way ocean-ocean coupling scenario. OpenFOAM is used to improve simulation accuracy in a part of the SWEET domain, adding the ability to simulate obstacles and complex flows. This includes flows that violate the assumption of hydrostatic vertical pressure gradients and would not be possible with SWE alone. It is shown that with this coupling setup, the capabilities of SWEET can be significantly extended to adapt to realistic use cases. The results achieved by the coupled system are evaluated qualitatively and quantitatively, using a monolithic benchmark where possible.

Contents

Abstract **iii**

1. Introduction **1**

2. Background **3**
- 2.1. Computational Fluid Dynamics . . . 3
  - 2.1.1. Navier-Stokes Equations . . . 3
  - 2.1.2. Shallow Water Equations . . . 5
  - 2.1.3. Discretization Methods . . . 6
  - 2.1.4. SWEET . . . 8
  - 2.1.5. OpenFOAM . . . 11
- 2.2. Coupling Methods and Software . . . 11
  - 2.2.1. Domain Decomposition . . . 11
  - 2.2.2. preCICE . . . 13
  - 2.2.3. Coupling an SWE Solver to OpenFOAM . . . 15
- 2.3. Summary . . . 16

3. Implementation **17**
- 3.1. Atmosphere-Ocean Solver . . . 17
  - 3.1.1. Overview . . . 17
  - 3.1.2. preCICE Configuration . . . 20
  - 3.1.3. OpenFOAM Participant . . . 20
  - 3.1.4. SWEET Participant . . . 20
  - 3.1.5. Evaluation . . . 21
- 3.2. Ocean-Ocean Solver . . . 21
  - 3.2.1. Overview . . . 22
  - 3.2.2. preCICE Configuration . . . 27
  - 3.2.3. OpenFOAM Participant . . . 28
  - 3.2.4. SWEET Participant . . . 35
  - 3.2.5. Evaluation . . . 38
- 3.3. Summary . . . 42

4. Results **43**
4.1. Atmosphere-Ocean Solver . . . 43
4.2. Ocean-Ocean Solver . . . 44
4.2.1. Cart2D Domain . . . 45
4.2.2. Sphere2D Domain . . . 53
4.2.3. Complex Geometry . . . 60
4.3. Summary . . . 62

5. Discussion **63**
5.1. Viability of Coupling SWEET to OpenFOAM . . . 63
5.2. Coupling Between a Spherical and Flat Domain . . . 65
5.3. Outlook: Parallel-in-Time . . . 65

6. Conclusion **67**

Abbreviations **68**

List of Figures **69**

List of Tables **71**

Bibliography **72**

A. preCICE Configurations **76**

1. Introduction

Modern numerical simulations greatly influence our daily lives. With advanced algorithms and the ever-growing computational power of modern computers, they can simulate and predict physical processes with unprecedented accuracy. Regarding the field of Computational Fluid Dynamics (CFD), these processes can range from small-scale experiments like simulating the flow of fluids in an engine [1] to large domains like the entirety of the global climate [39]. However, the overall quality of the simulation is limited by how well the model approximates physical relationships. In the case of global climate models, using only an atmospheric solver will limit the quality of the results. Many scenarios feature interactions between different systems that can not be approximated by simple Boundary Condition (BC). Even if all of the equations governing the atmosphere are correctly simulated, it is not a closed system. For example, the global oceanic currents and the energy and warmth they transport significantly impact the climate. Be that directly like the Gulf Stream, which makes the coastal Western European climate more moderate [28], or indirectly, like warm oceans that form hurricanes [30].

Domain decomposition offers an elegant approach to handle this without the need to construct a new solver. The idea is to split the domain into subsystems, each of which can be solved by its own numerical solver. Special numerical coupling schemes handle the exchange of information between the domains. A benefit is that no new solvers are required as it is enough to analyze the system's relationship at the boundaries. When using a good and flexible coupling library, little implementation work or expert knowledge of either solver is needed, and the average user can do this. It also means that all solvers could be exchanged with newer or more accurate versions without changes to the rest of the system.

This thesis focuses on using coupling methods to enhance the capabilities of SWEET [34]. SWEET is a research framework for the numerical approximation of SWE using spectral methods. Its purpose is to aid in developing and evaluating numerical methods. By integrating SWEET with another solver, we examine the possibilities of using SWE solvers in coupled models. Additionally, this allows for evaluating methods implemented in SWEET in conjunction with models implemented in other numerical solvers.

As the second part of our coupled solver, we use the open source CFD solver

OpenFOAM [27]. OpenFOAM allows for 3D free surface simulation around complex geometries. We use the coupling library preCICE to handle the communication and numerical aspects of coupling. The necessary functionality to support the exchange of coupling data is implemented in SWEET. Additionally, the preexisting preCICE adapter for OpenFOAM [6] is extended by logic to facilitate the conversion between conflicting information representations and dimensions. The coupled solvers are evaluated for their functionality qualitatively and quantitatively using a benchmark simulation and multiple test configurations, including complex geometries in the OpenFOAM domain.

We construct solvers for two scenarios. The first is a layered atmosphere-ocean solver using SWEET for atmospheric simulation and OpenFOAM for oceanic currents. Even though this setup is found to be potentially valuable and works in a proof-of-concept scenario, it is limited by the characteristics of SWEET. Secondly, we implement an ocean-ocean coupled solver, where the simulation accuracy and capability in a part of the SWEET domain are enhanced by delegating it to OpenFOAM. The proposed approach also allows for using complex geometries by including them in the OpenFOAM domain. This solver performs well and can simulate complex two-way interactions between SWEET and OpenFOAM.

In the course of this work, we investigate a central research question: Are there viable options for coupling SWEET, a SWE solver with spherical discretization, to a 3D configuration of OpenFOAM, a finite volume solver? SWEET can utilize two different domain types. A periodic Cartesian domain and a spherical domain. As a secondary research question, we investigate if it is possible to create an accurate locally coupled solver between a spherical and flat domain.

We start by introducing the necessary background information in Chapter 2, briefly explaining the basics of CFD and numerical coupling. We proceed with Chapter 3, where the methodology for our atmosphere-ocean and ocean-ocean solver is explained in detail. Finally, Chapter 4 tests and evaluates various scenarios and configurations for these solvers, and Chapter 5 interprets these results regarding the two research questions we formulate and provides an outlook into possible further developments.

2. Background

In this work, we couple two solvers. To accurately describe the methods used, some background information needs to be introduced. This section will first introduce the necessary theory for CFD, the important governing equations and numerical simulation methods, and the tools used to approximate these equations in Section 2.1. Next, it will introduce coupling in Section 2.2, describing the theoretical foundation and methods as well as introducing the coupling library we use in our implementation.

2.1. Computational Fluid Dynamics

CFD is a large field that deals with the numerical simulation of fluids. This section introduces the physical equations governing those fluids and the numerical methods used to approximate them. It starts with the Navier-Stokes Equations (NSE) in Section 2.1.1, the foundation of fluid dynamics. Further, it introduces the SWE in Section 2.1.2, a depth-averaged version of the NSE. After introducing the governing equations, we briefly describe the discretization methods used in this work in Section 2.1.3. Finally, we introduce SWEET (Section 2.1.4) and OpenFOAM (Section 2.1.5), the two solvers we use to build our coupled solver.

2.1.1. Navier-Stokes Equations

From a mathematical perspective, the movement of fluids is governed by the NSE, a set of Partial Differental Equations (PDE). They consist of the continuity equation, which establishes the conservation of mass, and the momentum equation, which establishes the conservation of momentum.

For incompressible flow, which is the kind of flow used for the simulations in this thesis, the continuity equation can be written as

$$\nabla \cdot \mathbf{u} = 0, \tag{2.1}$$

with $\mathbf{u}$ as the velocity of the fluid. This is equivalent to setting the velocity divergence to 0, which ensures that the volume is constant, a central implication of incompressible flow. The momentum equation is

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

where p is the pressure, ρ the density, and ν the kinematic viscosity of the fluid. Its terms model the different factors in the momentum equilibrium. $(\mathbf{u} \cdot \nabla)\,\mathbf{u}$ is the convective term and describes how the momentum is transported by the motion of the fluid. $\nu\nabla^2\mathbf{u}$ models the diffusion of momentum. The other two terms model surface and volume forces, respectively, where volume forces refer to the gravitational acceleration $\mathbf{g}$ that affects the fluid.

Solving these equations is the main focus of CFD. In order to be solved numerically, they need to be discretized. Numerous methods exist for discretization in space and time, e.g., Finite Differences, Finite Elements, Finite Volumes. More information about these topics and on the NSE and their derivation can be found in [15].

Boundary Conditions

How solvers deal with edges of the simulation domain is one of the most important issues of CFD and crucial for coupling. A BC provides a function to define these values. There are two elemental types of BCs. Let $\mathbf{u}_\Omega$ be the velocity and pressure at the boundary. Then

$$u_\Omega = c \tag{2.3}$$

is a Dirichlet BC. Dirichlet BCs directly prescribe a field's value at the boundary. A common use-case for Dirichlet BCs is for the velocity field at a wall, which is set to 0. An example of a non-zero value is an inlet, where a certain inflow into the domain is defined. Instead of directly defining the value, defining the field's gradient is possible too. This is called a Neumann BC and is defined as

$$\frac{\partial u}{\partial \mathbf{n}} = c, \tag{2.4}$$

where $\mathbf{n}$ is the direction of the normal of the boundary. It is often used for the pressure field at walls, establishing a zero-gradient BC. A third often used BC is the Robin BC. Instead of just a fixed value or gradient, it is a linear combination of the Dirichlet and Neumann BCs. It can be defined as

$$\alpha u + \beta \frac{\partial u}{\partial \mathbf{n}} = c, \tag{2.5}$$

where α and β are the weights for both parts. While these common BCs are sufficient for many scenarios and domains, special cases might require different behavior. There are also more complex possibilities for defining BCs. For example, a periodic BC reflects the values from the other side of the domain, which is needed for simulating domains like the Earth's oceans. More information about the detailed theory on BCs can be found in [11], and more details on their use in CFD in [12].

Multiphase simulations

With the methods described above, simulating a fluid in a confined domain is possible. However, for accurate simulation of oceans, it is necessary to simulate the ocean surface. This can be done using free-surface methods or multiphase simulations, where the interaction of two or more (incompressible) fluids is simulated in the same domain. There exist several methods to achieve this. One of these methods is the Volume of Fluid method [19], which is also employed by the solver used in this thesis. The key idea of the method is to establish a field $\alpha \in [0, 1]$, which signals the fluid ratio in a cell. The movement of the surface is tracked by solving a transport equation for α:

$$\frac{\partial \alpha}{\partial t} + \nabla \cdot (\alpha \mathbf{u}) = 0 \tag{2.6}$$

The ocean boundary can then be reconstructed from the α field for usage in the NSE.

2.1.2. Shallow Water Equations

An issue with the simulation of the NSE is the computational complexity of achieving a high-accuracy solution in a large domain. SWE are a method to solve this problem, as they can be used under certain assumptions to simplify the NSE. The central assumption is that the simulated flow only has a minimal depth compared to its width, making it nearly two-dimensional. This allows the assumption that vertical velocity gradients are close to zero and pressure is hydrostatic.

$$p = \rho g \left(h - z_b\right) \tag{2.7}$$

In this equation, z_b is the height of the ocean floor, h is the total height of the water column, and $h_w = (h - z_b)$ is the depth of the fluid layer. Since all SWE usages in this thesis assume a flat bottom, we will assume $h_w = h$ in the following formulas.

Because of the assumption of no vertical velocity gradients, the NSE can be depth-averaged. Combining this with Equation 2.7 yields the SWE. Using a simplified version with no lateral stresses and bottom stress yields the continuity equation

$$\frac{\partial h}{\partial t} + \frac{\partial \left(hu\right)}{\partial x} + \frac{\partial \left(hv\right)}{\partial y} = 0. \tag{2.8}$$

As well as the momentum equations for the two remaining dimensions:

$$\frac{\partial (hu)}{\partial t} + \frac{\partial}{\partial x}\left(hu^2 + \frac{1}{2}gh^2\right) + \frac{\partial (huv)}{\partial y} - fhv = 0 \tag{2.9}$$

$$\frac{\partial (hv)}{\partial t} + \frac{\partial}{\partial y}\left(hv^2 + \frac{1}{2}gh^2\right) + \frac{\partial (huv)}{\partial x} + fhu = 0 \tag{2.10}$$

where f is the Coriolis force. A more detailed explanation of the SWE, their derivation, and approximations can be found in [37].

The key advantage of the SWE over the standard NSE is that they can be computed much faster while obtaining accurate results. They are beneficial for simulating the global atmospheric or oceanic flow, as this is an inherently flat domain relative to its size. [8] However, the required assumptions limit the scenarios in which the SWE can be used. As such, they can not be applied in domains where the width-to-depth ratio is insufficient or where ground geometry or obstacles would cause significant vertical velocity gradients.

2.1.3. Discretization Methods

The purpose of a solver is to calculate the solution $y(t)$ to some kind of problem. In order to compute solutions to the NSE or SWE, the equations need to be discretized. This discretization is necessary both in time and space. This section introduces the important time- and space-discretization methods used in this work.

Explicit Runge-Kutta

In order to approximate the equations numerically, they need to be time-discretized. Usually, their use cases present as initial value problems, where an initial state is given and its time evolution is of interest. There are multiple methods for this, but a common one is the Explicit Runge-Kutta (ERK) method, particularly the fourth order ERK. Given an initial value problem

$$\frac{dy}{dt} = f(t, y) \qquad\qquad y(t_0) = y_0. \tag{2.11}$$

It is defined as

$$y_{n+1} = y_n + \frac{h}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right), \tag{2.12}$$

where h is the timestep size, and k_1, k_2, k_3, k_4 are defined as

$$k_1 = f\left(t_n, y_n\right), \tag{2.13}$$
$$k_2 = f\left(t_n + h/2, y_n + hk_1/2\right), \tag{2.14}$$
$$k_3 = f\left(t_n + h/2, y_n + hk_2/2\right), \tag{2.15}$$
$$k_4 = f\left(t_n + h, y_n + hk_3\right). \tag{2.16}$$

There are also many other variations, as the number of terms and the coefficients used inside can be varied along certain restrictions to form the family of Runge-Kutta methods. [3] A notable other method of the family is the Euler method, which is its simplest version containing only one term.

Spectral Deferred Corrections

The higher the order of a timestepping algorithm, the smaller the error is expected to be. This allows larger timesteps while maintaining the same accuracy. A high order is beneficial for stiff problems that are challenging to approximate otherwise. Instead of increasing the order of approximation by choosing a method with an intrinsically higher order like fourth-order ERK, Spectral Deferred Corrections (SDC) allows one to reach a higher order by iteratively improving a solution using low-order approximations.

In SDC, each timestep $[t_n, t_{n+1}]$ is divided into multiple substeps $s_0, s_1, ..., s_m$ where $s_0 = t_n$ and $s_m = t_{n+1}$. A first rough approximation of the solution $y^0(t)$ is obtained using a low-order scheme like the explicit Euler method. The initial value problem from Equation 2.11 can be rewritten as a Picard integral. Inserting the initial solution obtains the residual $\epsilon(t)$:

$$\epsilon(t) = y_0 + \int_a^t f\left(s, y^0\left(s\right)\right) ds - y^0\left(t\right) \tag{2.17}$$

The integral in Equation 2.17 can be approximated using spectral quadrature. The substeps s_m are chosen as the required nodes. Typical examples are Gauss-Lobatto or Gauss-Legendre quadrature. The approximated function can be iteratively refined using this residual. One such refinement is called a sweep in the context of SDC. [9] Mathematically, the higher the number of sweeps and nodes, the lower the expected error and effective order of the method.

The structure of the SDC algorithm also lends itself to more improvements like the parallel-in-time method PFASST that allows calculating multiple timesteps in parallel. [10] For this, the method defines coarse and fine grids, the result of which can be exchanged between timesteps after each sweep. This permits efficient parallelization even for problems that might prohibit spatial parallelization.

Spherical Harmonics

Spherical harmonics are a set of functions $Y_l^m(\theta,\phi)$ that build an orthonormal basis for representing any square-integrable function defined on a sphere. θ and ϕ are a point's polar and azimuthal angle on the sphere. Using the appropriate coefficients f_l^m, a function $f(\theta,\phi)$ on the sphere can be represented as

$$f(\theta,\phi) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} f_l^m Y_l^m(\theta,\phi). \tag{2.18}$$

In practice, l is restricted to l_{max} to select the spectral resolution and accuracy. A grid of points in physical space is needed to discretize a spherical domain. The integrals to calculate the coefficients a_l^m can then be approximated using quadrature. A typical grid is a latitude-longitude grid. An efficient implementation of this method for use in numerical simulation is done by [32] and is used for the spatial discretization of the SWE on the sphere in this thesis.

2.1.4. SWEET

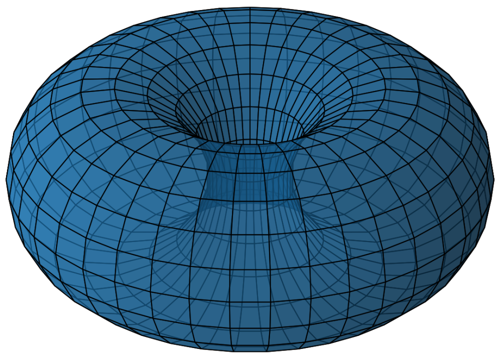
SWEET (Shallow Water Equation Environment for Tests) [34] is a CFD solver using SWE for numerical simulation. It is not built to be a production solver used to simulate real-world scenarios. Rather, it is intended as a test bench to develop and evaluate numerical methods for solving the SWEs and PDEs in general. In line with its research purpose, SWEET has many configuration options, utilities, and timestepping schemes. Among them are ERK, SDC [9], and PFASST [10]. A key characteristic of the solver is that the SWE are solved in the spectral domain. However, a grid representation of the domain and utilities to convert between physical and spectral space still exists.

Spectral solving is also the cause of some of SWEET's limitations. All of its available domains are periodic and do not have the possibility of defining obstacles or holes in the domain. There are also no other possibilities for boundary conditions. Consequently, any SWEET scenarios are solely defined by providing suitable initial values to its field, severely limiting the number of applications. SWEET also uses the simplified version of the SWE described in Section 2.1.2, which assumes a flat bottom. However, SWEET is intended to simulate atmospheric flows, where all these limitations are not meaningful. In this context, SWEET also allows outside forces by including gravity and Coriolis forces in its formulas.

In line with SWEET's intended purpose as a research framework, it is structured in an extensible way. Most key SWEET features, such as data structures, mathematical methods, and timestepping schemes, are implemented in a library structure. Additionally, there are several `Programs` using the library to implement solvers in different domains and extensions. Each `Program` specifies a list of scenarios, which can be selected at runtime to initialize the solver's fields with the necessary values. The main programs are `PDE_SWECart2D` and `PDE_SWESphere2D`. They implement the main solvers for the `Cart2D` and `Sphere2D` domains described in the following. Programs for more specific cases also exist, such as a program implementing the PFASST algorithm for timestepping.

Cart2D

The first domain implemented in SWEET is the `Cart2D` domain. It is a simple Cartesian grid with equal spacing and periodic boundaries at the sides, resulting in a two-dimensional torus. A visualization of the domain both in 3D can be seen in Figure 2.1a. However, the 3D portrayal is not entirely accurate as not all cells have equal area, which is the case for this domain. In addition to initial values, the horizontal and vertical size of the domain can be configured. The number of cells and frequency components can be controlled by setting either parameter, with the other one automatically being adjusted accordingly. The multidimensional Fourier Transform is used to convert the grid to spectral space. `Cart2D` uses the velocity version of the SWE described in Equation 2.9. Because of its simplicity, we choose this domain as the starting point for our work.


(a) Cart2D Domain

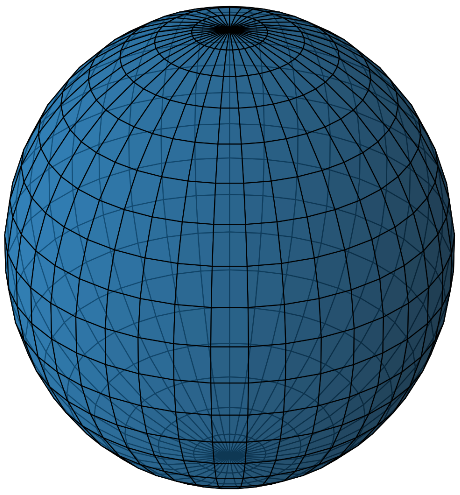

(b) Sphere2D Domain

Figure 2.1.: 3D visualizations of both SWEET domains. While the Sphere2D figure matches the model, the Cart2D figure is only a visualization, as in the model, all cells have the same size.

Sphere2D

The second domain offered by SWEET is a sphere. In physical space, it is defined as a latitude-longitude grid. A visualization of the domain can be seen in Figure 2.1b. When comparing the gridlines to the latitude-longitude grid in Figure 2.2, this also shows a complication with this domain. Due to the use of equirectangular projection for the grid, the points at very high or low latitudes are considerably closer to each other than at the equator. This has to be considered when implementing any scenarios and initial values in this domain. It also means that any output in physical space has a higher resolution at the poles. The domain size can again be configured using either the number of latitudinal and longitudinal cells or the spectral resolution. The size of the sphere can be controlled by setting the radius. For conversion into spectral space, this `Program` uses the spherical harmonics method described in Section 2.1.3. Instead of using the velocity-based SWE introduced in Section 2.1.2, it uses a version of the SWE defined in terms of vorticity $\zeta = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}$ and divergence $\delta = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}$. Additionally, instead of the perturbation of water height h', the geopotential height $\Phi = gh'$ is used. We use this `Program` in addition to the `Cart2D` program to test a more realistic application scenario on a sphere and because there exist implementations of more advanced timestepping schemes like parallel SDC [4] and PFASST due to previous work. We also use this domain to answer our secondary research question.

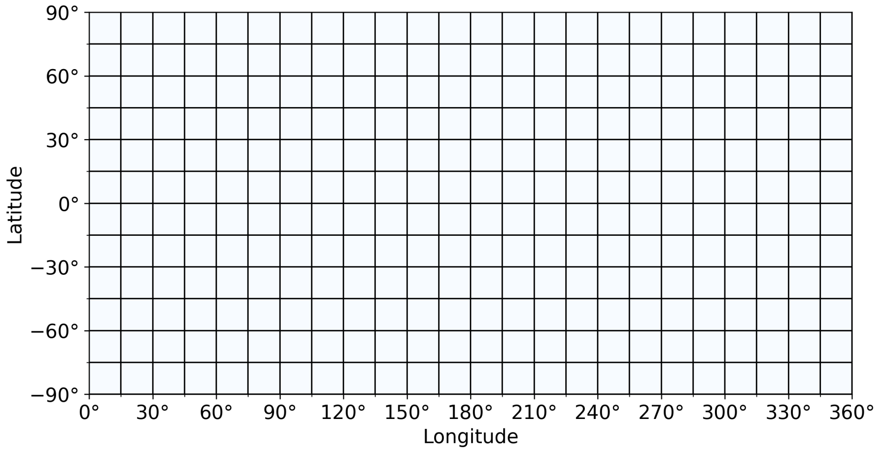


Figure 2.2.: Example of a latitude-longitude grid used to represent the Sphere2D domain. The dimensions of cells on the grid do not match their dimensions on the sphere.

2.1.5. OpenFOAM

OpenFOAM [27] is a large and popular open source CFD framework with many different solvers and methods. It uses the finite volume method and an unstructured grid for its simulations. For this work, we use two of the solvers provided with OpenFOAM. The first solver we use is the `icoFoam` solver, which is a solver for the incompressible NSE. The second solver we use is `interFoam`, a solver for incompressible multiphase flow. An OpenFOAM scenario is defined by a so-called `Case` directory containing the necessary configuration files. Here, all parameters concerning the algorithm, timestepping, boundary conditions, and other related aspects can be configured.

2.2. Coupling Methods and Software

This thesis works with coupled models. There are two essential ways to simulate a coupled system. The first way is to create a new solver to simulate both subdomains and their coupling. This is called the monolithic approach. The second way is to couple two existing solvers using either a coupling library or other methods. Both solvers operate independently in this approach, so they can rely on the preexisting implementations. This is the approach used in our work.

The following Section 2.2.1 provides an introduction to the theory and methods used to split a system into parts and to couple simulations of the individual parts to obtain a coupled solver. Section 2.2.2 introduces the coupling library we rely on for the implementation of our coupled solvers.

2.2.1. Domain Decomposition

Many applications of numerical simulation involve more than one system. While these systems may have different governing equations, their interactions can still be of interest. Examples of this can be the interaction between fluid dynamics and structural deformation in the cardiovascular system [18] or the interaction between the ocean and the atmosphere [24, 17]. This process of splitting up the domain is called domain decomposition [25]. The two subsystems do not necessarily need to have different governing equations for domain decomposition to be desirable. Different methods of approximating the same governing equations can have varying trade-offs, making it beneficial to simulate parts of the domain differently. In the aforementioned example, both the ocean and atmosphere are subdomains of a larger climate model. The following contains the most important information about techniques for coupling the subdomains to create a coupled simulation, but domain decomposition is an extensive topic, and more information on the topic can be found in [25].

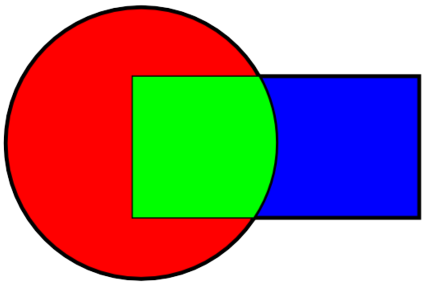


Figure 2.3.: Visualization of domain decomposition of a domain formed by overlapping circular and rectangular domains. Representation of the problem originally considered by H. A. Schwarz.[1]

Coupling Schemes

A coupled solver iteratively calculates timesteps. There are two different coupling schemes: loose (explicit) and tight (implicit) coupling. [21] For loose coupling, each system of equations is solved independently for each timestep. In tight coupling, both systems are solved simultaneously. The literature is not consistent with the terminology used. This work will use explicit and implicit coupling as defined in preCICE [5], the coupling library we use, to avoid confusion regarding the implementation.

With explicit coupling, both solvers calculate the solution to a timestep and exchange information. For the boundary, the values exchanged in the previous timestep are used. After exchanging values, both solvers immediately resume with the next timestep. This way, each solver calculates each timestep exactly once.

In contrast, in implicit coupling, each timestep is calculated multiple times. For the boundary conditions, the solvers combine the exchanged values at the beginning of the timestep and the last set of exchanged values at the end. This way, both solvers iteratively calculate the timestep and exchange information until a convergence metric is satisfied on the exchanged values. Usually, this means the exchanged values do not change anymore.

Because each timestep is only calculated once, explicit coupling is less computationally expensive than implicit coupling. However, it is less accurate and will cause instability in many scenarios. Implicit coupling is more accurate and can achieve convergence and high accuracy even for stiff scenarios.

[1]Pichon22, CC BY-SA 3.0 `https://creativecommons.org/licenses/by-sa/3.0`, via Wikimedia Commons

Interface and Volume Coupling

The type of coupling used can be characterized by how and where values are exchanged between the solvers. There are two options for the exchange location. The first option is interface coupling, where values are exchanged only at the boundaries of each domain. This can be the same boundary if both domains are adjacent without overlap, but it could also be in different locations if there is an overlap between the subdomains (e.g., Figure 2.3). Atmosphere-ocean models are an example of interface coupling, with the exchange of quantities at the sea surface (e.g., [2]). The second option is volume coupling. In this case, values are exchanged over a larger part of the domain. This type of coupling necessarily requires overlap. An example where this can be used is modeling deformation and fluid flow in porous media [38].

Coupling can further be categorized by the combination of used boundary conditions. Possible combinations are Dirichlet-Neumann coupling, Robin-Robin coupling, and Dirichlet-Dirichlet coupling. However, for Dirichlet-Dirichlet coupling, an overlap between the domains is required. Otherwise, the values would only be switched back and forth. An example of Dirichlet-Dirichlet coupling is the multiplicative Schwarz method. [25]

2.2.2. preCICE

preCICE [5] is a versatile coupling library designed for domain decomposition. Its goal is to simplify the coupling of numerical solvers by providing an interface that manages all communication and calculations required for this coupling. This functionality is made accessible through an easy-to-use API. It implements the coupling flow itself, i.e., it handles the synchronization of coupling windows and, in the case of implicit coupling, also tells the coupled solvers when they have to store or recover a checkpoint. Furthermore, there is no need for direct communication between solvers, as preCICE handles all aspects of data exchange and mesh interpolation. The API is accessible, with bindings implemented in commonly used programming languages like C++, Python, and Fortran. An overview of the preCICE ecosystem can be seen in Figure 2.4.

The vision preCICE follows is to be as modular and noninvasive as possible. It should be possible to exchange one of the solvers without changing anything with the other solvers. Thus, the programs are started as usual, standalone versions. Coupling is achieved using calls to the preCICE library, which implements peer-to-peer communication between the program processes. All solvers use the same preCICE configuration file to establish communication and other coupling parameters. The configuration assigns the coupling scheme, exchanged data names, solver names, and which meshes must be defined by which solver. In addition to options for implicit

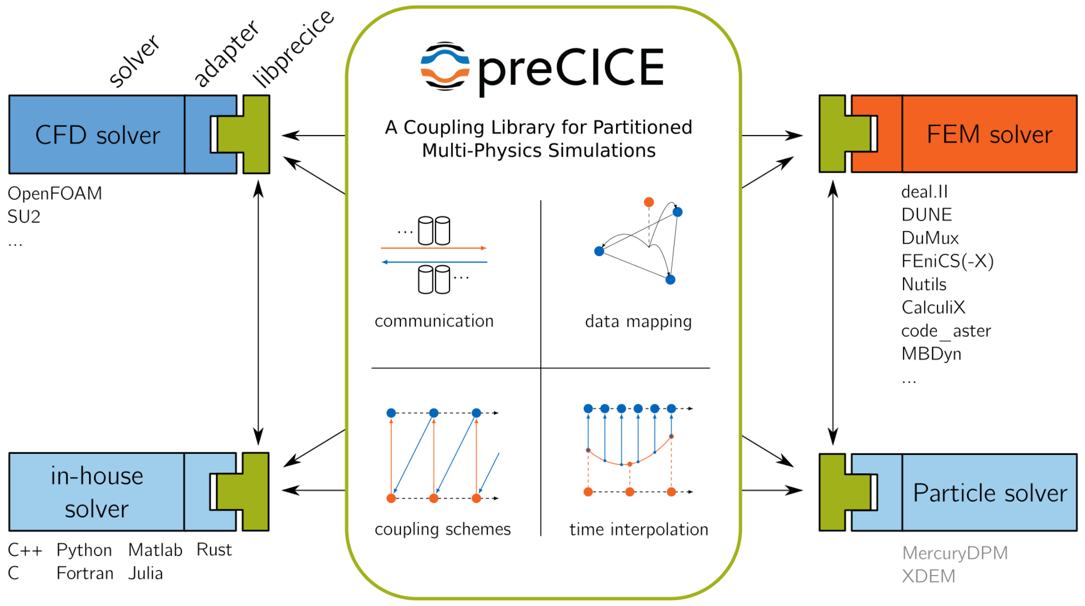


Figure 2.4.: Overview of the functionality provided by preCICE.[2]

and explicit coupling, there is also a choice of serial or parallel coupling. In serial coupling, solvers take turns solving timesteps and wait for the other's results. In parallel coupling, both solve timesteps and exchange values concurrently.

Solver meshes are defined using vertices with positions. Optionally, connectivity information can be added to the mesh, which is necessary for some interpolation modes. The simplest interpolation setting is nearest-neighbor interpolation. However, this can result in problems should there be larger holes in the domain. A more advanced mode is the nearest-projection method, a second-order method that uses surface projection and linear interpolation to achieve a higher accuracy mapping but requires connectivity information. Alternatively, there are also the RBF kernel methods, which are more computationally expensive but do not require information about mesh connectivity.

A key feature of preCICE is its modularity. It relies solely on the interface defined by the shared configuration file. The Solvers do not need to exchange implementation details, such as discretization methods or meshes. This approach facilitates the exchange of coupled participants without the need to make adjustments to the implementation.

[2]`preCICE.org`, CC BY 4.0 `https://creativecommons.org/licenses/by/4.0/`

Connecting a Participant to preCICE

The term preCICE uses for a solver that is part of a coupled model is "Participant". There are two options to connect a participant to preCICE: The first option is to use a preexisting adapter for the solver. An adapter handles all the calls to the preCICE API and integrates directly into the solver. It reads the preCICE configuration file, sets up the meshes, and handles the reading and writing of data. Depending on the solver, the adapter integrates directly into the solver, and the solver's code does not need to be changed. There exist "official" adapters for several popular numerical solvers. Some of them are OpenFOAM, FEniCS, and CalculiX.

In this work, as we use OpenFOAM, we make use of the OpenFOAM adapter [6]. It supports the exchange of most quantities in an interface coupling context. In addition, it supports volume coupling for temperature, pressure, and velocity. The adapter is structurally organized into three modules: fluid-fluid (FF), fluid-structure interaction (FSI), and conjugate heat transfer (CHT). In our work, we use the FF module. The adapter integrates into OpenFOAM as a function object called in every iteration and can be configured using the same files as OpenFOAM.

Should no adapter exist, or the existing adapter not support the necessary functionality, the calls to the preCICE API need to be manually added to the simulation loop. To use the API, only basic access to the timestep and data fields of the simulation is needed. Usually, no significant changes to the code are necessary. This is what we do for SWEET, as it is an in-house solver.

2.2.3. Coupling an SWE Solver to OpenFOAM

Using domain decomposition to simulate a flat water domain with a combination of an SWE solver and a 3D-NSE solver is not an inherently new idea. There are two notable works that use setups similar to this thesis.

The first one is [26]. They achieve good results in coupling an SWE domain and a 3D-RANS model. Similar to this work, they place their SWE domain inside of the 3D domain. However, contrary to this work, they do not couple two preexisting solvers but follow the monolithic approach, creating a new solver incorporating both models. Consequently, they also place stricter requirements on their meshes, requiring the cells of both the 2D and 3D solvers to align at the interfaces.

The other related work to this thesis is [29]. They also couple SWE to OpenFOAM, achieving good results. This work also utilizes two preexisting solvers and utilizes preCICE to couple them. Contrary to this work, [29] did not use obstacles and only coupled on one boundary. Furthermore, the setup of boundary conditions differs from this work.

Neither of them used SWEET or a similar SWE solver. A further distinction from the previous research is that this work deals in detail with SWEET's periodic domain with spectral discretization and applies the SWE-3D coupling to the sphere.

2.3. Summary

In this chapter, we introduced the theoretical foundation of the methods and tools used in this work. We presented the NSE and the SWE, a depth-averaged 2D version of them. We also went over the two individual solvers used in this work, OpenFOAM and SWEET. Lastly, we provided the necessary background information about domain decomposition and coupling as well as preCICE, the coupling library used for our coupled solver.

3. Implementation

In this chapter, we describe the implementation details for the two coupling setups we investigate. The actual code created and used in this thesis can be found on GitHub[1]. The chapter starts by introducing the one-way atmosphere-ocean solver in Section 3.1. As this solver only allows for one-way coupling, it is implemented solely in a proof-of-concept setup. In Section 3.2, we continue by describing the ocean-ocean solver that is the main focus of this work. For each solver, we start by providing an overview of the coupled model. Then, we explain the implementation details by presenting the preCICE configuration as well as the configurations of both the OpenFOAM and SWEET participants. Lastly, we introduce the scenarios used to test our solvers.

3.1. Atmosphere-Ocean Solver

This section describes the first solver developed in this work. The atmosphere-ocean solver is a layered solver supporting one-way coupling. The atmosphere is simulated by SWEET using SWE, and the ocean is simulated by OpenFOAM using the finite volume method. The solver supports one-way coupling from SWEET to OpenFOAM. We start by providing an overview of the motivation and model of this coupling setup in Section 3.1.1. Then, we discuss the preCICE configuration used to implement this model in Section 3.1.2. We continue with showing how we connect OpenFOAM (Section 3.1.3) and SWEET (Section 3.1.4) to preCICE. Finally, we explain the test scenario we use to validate our proof-of-concept in Section 3.1.5.

3.1.1. Overview

As a first step, we examine the possibilities of coupling SWEET in its intended role as a solver for atmospheric simulation. One of the usual coupling setups using an atmospheric solver is a coupled atmosphere-ocean model. [2, 13] Therein, the atmospheric and ocean solvers simulate different climate layers in a local or global context. Usually, this involves two-way communication along the boundary between these two layers, where information about velocity, temperature, and possibly other values is

[1] `https://github.com/arminettenhofer/thesis_sweet_coupling`

exchanged. The temperature is coupled bi-directional, with the ocean solver providing the Sea Surface Temperature (SST) to the atmosphere solver, while the atmosphere solver can give the heat flux. Additionally, the velocity is coupled unidirectional, with the atmosphere exerting a shear force on the ocean surface, which is described by the wind stress τ_{wind}.

$$\tau_{wind} = C_d \rho_{air} U_h^2 \tag{3.1}$$

$C_d = 0.0015$ is the drag coefficient for wind over the sea, and U_h is the wind velocity at some height over the sea surface, usually $h = 10\,\text{m}$. [8]

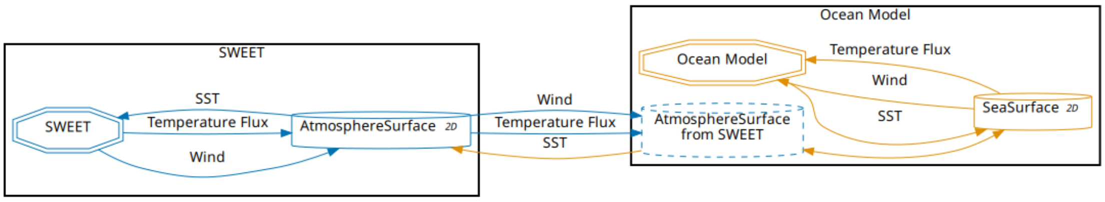


Figure 3.1.: preCICE configuration for atmosphere-ocean solver with the exchange of momentum and heat.

A preCICE configuration implementing this scheme can be seen in Figure 3.1. However, because SWEET simulates the atmospheric movements using SWE, there is no inherent energy transport, which would enable the coupling of SST and temperature flux, and the implementation suggested in Figure 3.1 is not feasible with SWEET.[2] Therefore, the coupling can only be implemented in a reduced form, which does not use the heat coupling and only uses the velocity coupling in the form of wind stress. This results in one-way coupling as seen in the revised preCICE configuration in Figure 3.2.

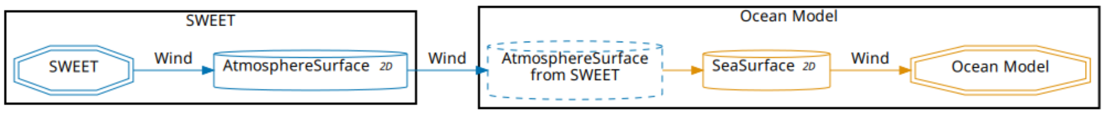


Figure 3.2.: preCICE configuration used for atmosphere-ocean solver with only the exchange of momentum. The full configuration file can be seen in the appendix in Listing A.1.

The focus of the thesis is to research the coupling scenarios available to SWEET and their viability. One-way coupling only allows very restricted interactions and scenarios.

[2]Adding energy transport to the SWE in SWEET is generally possible but would require major modifications that are out of scope for this work

These scenarios only have limited use cases, especially as SWEET can only be controlled with the initial conditions. Therefore, this solver is implemented as a proof-of-concept version to show general viability should heat transport or BCs be added to SWE at some point in the future. Instead, the work will focus mainly on developing a more promising ocean-ocean solver, which is detailed in Section 3.2. Nonetheless, a small prototype implementing this reduced atmosphere-ocean coupling is set up as an initial test using OpenFOAM as the ocean solver and simplifying the implementation of wind stress.

In the solver, the domains are placed on top of each other, with the intention of OpenFOAM simulating the ocean and SWEET simulating atmospheric currents. They are chosen to be the same size. The size of the SWEET domain is left at its default size, which is calculated using the Earth's radius $r = 6.37122 \times 10^6$ m. This results in a domain size of approximately 40000 km × 40000 km matching the circumference of the Earth. We use the `Cart2D` program for SWEET, resulting in a square domain with periodic boundaries representing a 2D torus. The OpenFOAM domain is set to match the SWEET domain with an added third dimension, which is also set to 40000 km, creating a cubic domain. From a physical perspective, this domain may not seem logical. However, this test case is only used as a proof of concept.

Boundary Conditions

	Velocity	Pressure
Top	$v = v_c$	$\partial p / \partial n = 0$
Bottom	$v = 0$	$\partial p / \partial n = 0$
Sides	$v = 0$	$\partial p / \partial n = 0$

Table 3.1.: OpenFOAM BCs in the atmosphere-ocean solver.

To complete the definition of the domain, the BCs and the equation governing the exchange of information need to be defined. For SWEET, no BCs are required. SWEET receives no data in the coupling, so no BC needs to be set at the boundary with OpenFOAM. Additionally, the boundaries on the sides of the domain are periodic.

The BCs of the OpenFOAM domain can be seen in Table 3.1. For the velocity, Dirichlet BCs are used for all domain edges. The value used is 0 for the bottom and sides and equal to the coupled velocity v_c received from SWEET for the top of the domain. A zero-gradient Neumann BC is used on all sides for the pressure.

This configuration effectively means that OpenFOAM has walls at the sides of its domain, while SWEET uses periodic BCs. We used this BC because of its more straightforward configuration. However, if the coupling were to be used in a correct physical context, the boundary behavior of SWEET and OpenFOAM should match on the sides of the domain. Furthermore, instead of calculating the wind stress described in Equation 3.1, it directly transfers the velocity from SWEET to OpenFOAM without applying any calculations. This is because implementing the effect of wind stress is not straightforward, as it depends on the relative velocity. Both the coupled velocity from SWEET and the current velocity of OpenFOAM are needed for calculation. Implementing this calculation would require a modification of the preCICE adapter for OpenFOAM. This solver is only used as a proof of concept, so we simplify the BC to only copy the SWEET value.

3.1.2. preCICE Configuration

Figure 3.2 shows a visualization of the preCICE configuration used to couple this solver. The original configuration file can be seen in the appendix in Listing A.1. There is only one value that is exchanged by preCICE. This is the two-dimensional wind velocity. However, as the OpenFOAM mesh is three-dimensional, the wind needs to be represented by a three-dimensional vector. The mapping between the meshes of the atmosphere and ocean solver is handled on the ocean side, as the resolution of the OpenFOAM domain is higher. This theoretically reduces the amount of values that need to be transferred between the processes by transferring the smaller mesh.

3.1.3. OpenFOAM Participant

We use a standard configuration of OpenFOAM without any significant changes since this thesis focuses only on the coupling itself. We use the icoFOAM solver, which simulates incompressible single-phase flow. The BCs are configured as detailed in Section 3.1.1. OpenFOAM is configured to use the preCICE adapter, which is configured to read the velocity from preCICE and write it to the top part of the domain. In the configuration, the top boundary's velocity BC is set to `fixedValue` with the value overwritten by the adapter.

3.1.4. SWEET Participant

There is no preexisting preCICE adapter for SWEET, which makes changes necessary to add the calls to the preCICE API. Only one-way communication is required, so these changes are relatively simple. All changes are based on the `SWE_Cart2D` program of SWEET. As these changes are also included in the two-way coupling scenario that is examined later, a detailed description can be found in Section 3.2.4 and Listing 3.6. As an overview, SWEET creates a mesh with the coordinates of all the grid values to be exchanged and writes the velocity at these locations to preCICE in every timestep.

3.1.5. Evaluation

We use one of the already existing scenarios for this SWEET program as a test scenario. The stable initial state can be seen in Figure 3.3. It models two jet streams in the northern and southern hemispheres. It was initially intended to be used with the addition of a Gaussian bump in the stream to break symmetry and test numerical solvers using developing instabilities. [20] We use it as a simple scenario with a constant influence on the ocean to test if the coupling has the desired effect. The atmosphere depth in this scenario is 10000 m.

In conjunction with the BCs for the OpenFOAM domain, which model walls at the sides and bottom and essentially a moving wall with varying speeds at the top, the scenario is similar to a lid-driven cavity. The difference is that the lid does not have a consistent speed but consists of moving and non-moving strips.

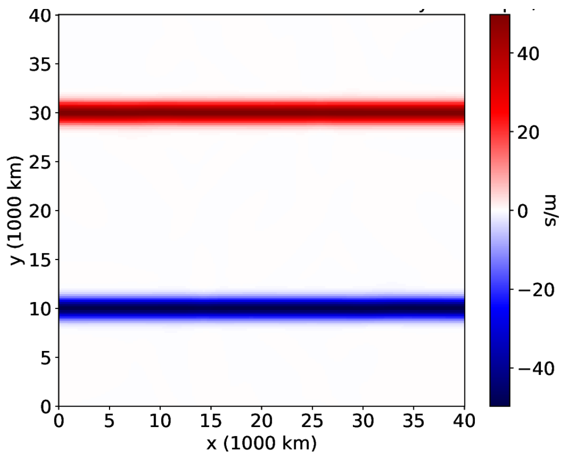


Figure 3.3.: Two jet streams in the SWEET domain for the atmosphere-ocean scenario.

3.2. Ocean-Ocean Solver

This section discusses the ocean-ocean solver, which is the main focus of this work. In contrast to the atmosphere-ocean solver, this solver allows for two-way coupling. Section 3.2.1 starts by giving an overview of the solver. It introduces the domain decomposition and the motivations behind it. It also introduces the BCs and the necessary data conversion necessary because of the different data representations in the two solvers. Section 3.2.2 shows and discusses the preCICE configuration used for this solver. We continue by describing the implementations necessary to connect

OpenFOAM (Section 3.2.3) and SWEET (Section 3.2.4) to preCICE. Finally, Section 3.2.5 introduces the scenarios used to evaluate this solver.

3.2.1. Overview

The atmosphere-ocean solver only allowed for one-way coupling. Therefore, other viable options were examined. One of them is to couple SWEET with a different solver horizontally instead of vertically. A similar setup was previously used by [29] to couple a SWE ocean solver and OpenFOAM. However, there are two issues with applying the same coupling scheme to SWEET. It was created for atmospheric simulation, and because of its periodic domain, there are no obvious horizontal boundaries at which a coupling would be possible. As described in Section 2.1.4, it does not contain any functionality for boundaries at all. The governing equations do not change for ocean simulation, so the first issue can be fixed by changing the viscosity from zero to that of water. The other problem is addressed by placing the OpenFOAM domain inside the SWEET domain instead of next to it.

Besides enabling two-way coupling, this scenario is also helpful in showing a possible use case of coupling SWE solvers like SWEET to 3D-fluid solvers. The low computational effort SWE can be used for large parts of the Earth's ocean system or a large body of fluid in any other periodic domain. Meanwhile, the high-precision 3D NSE can be used in high-interest regions, regions with obstacles, or regions where the assumptions necessary for the SWE do not hold.

SWEET uses spectral discretization methods. Consequently, its domain needs to be periodic and cannot have any holes. When the OpenFOAM domain is placed in the SWEET domain, SWEET still simulates the entire region of the OpenFOAM domain. For this reason, we also perform volume coupling for SWEET in that region, writing values from OpenFOAM to SWEET for the entirety of the overlap. In the other direction, we perform interface coupling along the sides of the overlap from SWEET to OpenFOAM. Therefore, while this results in two-way coupling, the coupling uses different meshes and a different number of data points for each direction. While this is straightforward for an empty OpenFOAM domain, some complications arise when handling the coupling of obstacles in the OpenFOAM domain. Furthermore, OpenFOAM and SWEET handle water height differently, with SWEET defining it as a scalar height value and OpenFOAM with a scalar value per cell assigning fluid content. These issues are discussed in detail in Section 3.2.1.

The domain decomposition is complicated because of the differences between the solvers. The SWE domain is a periodic plane with 100 km $\times$ 100 km dimensions for the `Cart2D` domain, and a sphere with 100 km circumference for the `Sphere2D` domain. The fluid solver has a domain of 10 km $\times$ 10 km $\times$ 2 km dimensions. water height is

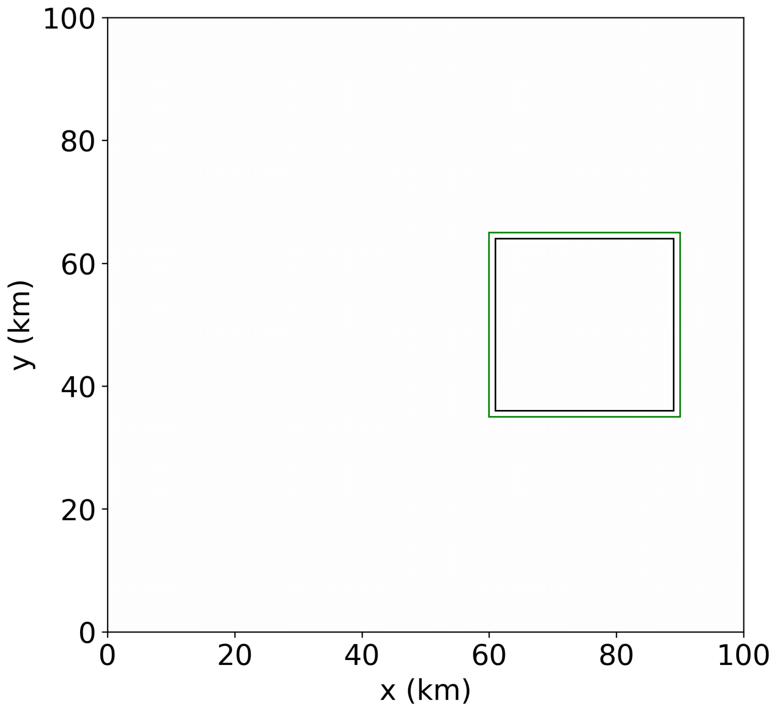


Figure 3.4.: Domain decomposition of the `Cart2D` domain in the ocean-ocean solver. The outer green line indicates the border of the OpenFOAM domain. The inner black line indicates the border of the SWEET domain.

represented as a scalar value in the SWE, so they can theoretically represent waves of any height. The vertical domain size of the OpenFOAM domain is thus chosen in such a way that it is larger than the maximum water height reached in any of the scenarios. In the SWEET to OpenFOAM direction, there is boundary coupling along the boundaries of the OpenFOAM domain. With the possibility of feedback loops and the necessity of overlap for Dirichlet-Dirichlet coupling [35], an offset is needed between the coupling regions. Therefore, the coupling from OpenFOAM to SWEET occurs as volume coupling not over the entire OpenFOAM domain but with offset from the edges of the OpenFOAM domain. This distance between the boundaries is set to $d_\Omega = 2\ \texttt{cells} = 1\,\text{km}$. The domain boundaries for the `Cart2D` domain are shown in Figure 3.4, and for the `Sphere2D` domain in Figure 3.5. Values are transferred from SWEET to OpenFOAM along the outer green line, while the transfer of values from OpenFOAM to SWEET takes place within the area of the black square.

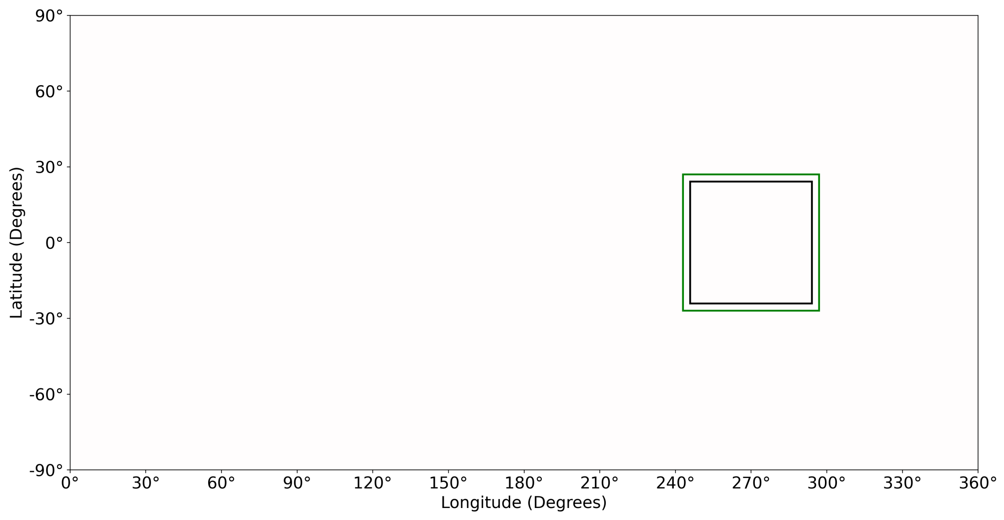


Figure 3.5.: Domain decomposition of the `Sphere2D` domain for the ocean-ocean solver. The outer green line indicates the border of the OpenFOAM domain. The inner black line indicates the border of the SWEET domain.

Boundary Conditions

	Velocity	Pressure	Alpha
Top	$pressureInOut = 0$	$totalPressure = 0$	In: $\alpha = 0$; Out: $\partial\alpha/\partial n = 0$
Bottom	$noSlip$	$fixedFluxPressure = 0$	$\partial\alpha/\partial n = 0$
Obstacle	$noSlip$	$\partial p/\partial n = 0$	$\partial\alpha/\partial n = 0$
Sides	$v = v_{SWEET}$[3]	$\partial p/\partial n = 0$	$\alpha = f_{2Dto3D}\left(h_{SWEET}\right)$[3]

Table 3.2.: OpenFOAM BCs in the ocean-ocean solver.

The BCs used for OpenFOAM can be seen in Table 3.2. For the bottom, which is the seafloor, and obstacles, they are modeled as walls. For alpha, a zero gradient BC is used, and for the velocity, a no-slip BC disabling movement in the obstacle adjacent fluid layer. The pressure uses a zero gradient BC on obstacles. However, for the seafloor, it uses the `fixedFluxPressure` BC, which sets the pressure in such a way that the flux is the specified value, in this case 0. This is useful to achieve an accurate solution and is taken from the OpenFOAM tutorial for multiphase flow.

The top is the atmosphere, meaning an unrestricted in and outflow of air should be possible. Therefore, for alpha, we use an inlet-outlet BC that changes depending

[3]For a detailed explanation of how and where these BCs are applied, see Section 3.2.3

on the velocity flux. For inflow, alpha is set to 0, corresponding to pure air, and for outflow, it is a zero gradient Neumann BC. For pressure, a *totalPressure* BC is used. The total pressure p_0 is a sum of the static and dynamic pressure, which depends on the velocity. Using the *totalPressure* BC allows the static pressure to vary depending on the convective flux: $p = p_0 - 0.5|u|^2$.

For the sides, the coupled values received by preCICE need to be used. The pressure is not coupled since it does not exist in the SWE. Instead, it is set to use a zero gradient Neumann BC. For the velocity, the velocity received from preCICE is used to set the water cells of the boundary. However, no OpenFOAM BC exists to achieve this. Therefore, this is manually implemented in the preCICE adapter. The same problem is encountered for the water height as the value cannot be directly used in OpenFOAM, which only uses alpha values for tracking the cell phase. Consequently, this conversion is also handled manually. Information on how this is done can be found in Section 3.2.1.

	Velocity	Height Perturbation
Coupling Region	$v = v_{FOAM}$	$h_p = h_{FOAM} - h_0$
Edges	*periodic*	*periodic*

Table 3.3.: SWEET BCs in the ocean-ocean solver.

The BCs used by SWEET can be seen in Table 3.3. SWEET has a periodic domain as explained in Section 2.1.4. As such, it uses periodic BCs for the velocity and water height perturbation at the sides of the domain. The velocity received from OpenFOAM is directly taken as the velocity value for the coupling region. The same is true for the height perturbation, although the initial water height h_0 needs to be subtracted because SWEET uses the height perturbation for its calculations.

Data Conversion

SWEET and OpenFOAM use equations with different dimensions. Therefore, for a successful coupling, a dimensionality conversion is necessary. The dimensionality conversion is implicit for the direction from SWEET to OpenFOAM. It is automatically performed by the nearest-neighbor interpolation done by preCICE. This extends the 2D values to cover each vertical column of the OpenFOAM boundary with the same value. For the direction from OpenFOAM to SWEET, the dimensionality reduction must be defined manually, as no functionality exists to handle this automatically. Additionally, different abstractions are used for the same physical circumstances. This is the case for the water height, which is modeled as a per-cell alpha value tracking the water-to-air ratio in OpenFOAM. For SWEET, this is handled merely as a scalar value giving the perturbation of water height to the normal water level h_0.

Velocity When applying the coupled velocity from SWEET to OpenFOAM, we only want to write the velocity to the water cells as SWEET does not model the air velocity at all, and setting it to be the same velocity as the water would significantly overestimate air velocity. Nonetheless, air velocity is likely not 0, so only setting water velocity and assuming air velocity to be 0 is a slight inaccuracy that is necessary because of the unavailable information. Let b be a cell on the OpenFOAM boundary and v^{SWEET} the velocities received from SWEET. Then

$$v_b = \alpha_b \cdot v_b^{SWEET} \tag{3.2}$$

In the other direction, an explicit dimensionality reduction is needed. For the reduced velocity, only the water is considered. Therefore, for every column, the cross-product between alpha and the velocity is computed and then divided by the sum of alpha to obtain the average water velocity in that column. Let C be a set of one column of cells of the OpenFOAM domain, n the number of vertical cells, and v^{FOAM} the velocities OpenFOAM writes to preCICE. Then

$$v_b^{FOAM} = \frac{1}{\sum_{c=C_1}^{C_n} \alpha_c} \cdot \sum_{c=C_1}^{C_n} \alpha_c \cdot v_c \tag{3.3}$$

Water Height For the direction from SWEET to OpenFOAM, the water height has to be converted to the matching alpha value for each cell in order to use it. In order to do this, the top and bottom of the cell must be checked. If the bottom is lower than the water height, alpha is set to the corresponding value. Let b be a cell on the OpenFOAM side boundary, h_{bottom} and h_{top} the height of the bottom and top of the cell, and h the water height for this column. Then

$$\alpha_b = \begin{cases} 0 & \text{if } h < h_{bottom} \\ \min\left(1, \frac{h - h_{bottom}}{h_{top} - h_{bottom}}\right) & \text{if } h \geq h_{bottom} \end{cases} \tag{3.4}$$

For the other coupling direction, one column of alpha values must be reduced to a single scalar height value. It would be possible to take the height of the first cell containing a value $\alpha > 0$. However, a reasonable threshold would need to be chosen as alpha is rarely exactly 0. Furthermore, OpenFOAM also allows wave barrels. Therefore, it is not necessarily true that all cells below have $\alpha = 0$. Therefore, the water height is calculated by the sum of all alpha values multiplied by the cell height δh. Let C be a set of one column of cells of the OpenFOAM domain, n the number of vertical cells, and h^{FOAM} the water height OpenFOAM writes to the interface. Then

$$h_b^{FOAM} = \delta h \cdot \sum_{c=C_1}^{C_n} \alpha_c \tag{3.5}$$

3.2.2. preCICE Configuration

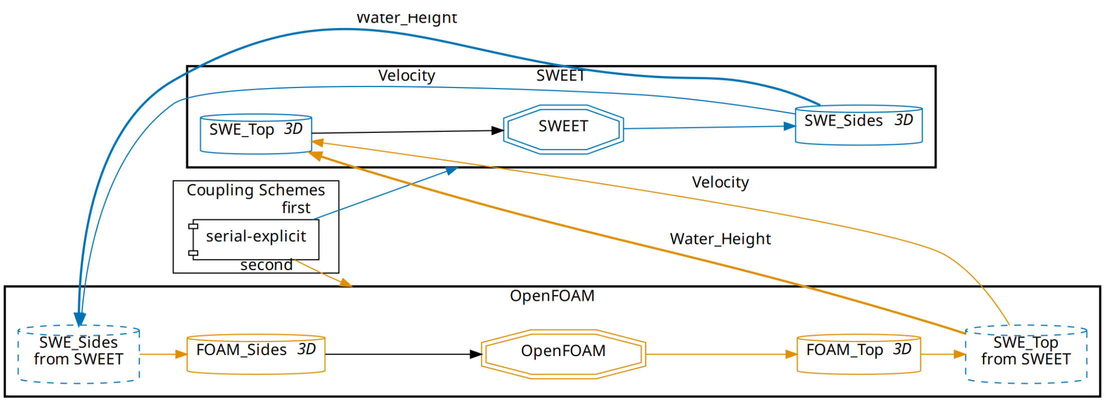


Figure 3.6.: preCICE configuration for the two-way-coupled ocean-ocean solver. The full configuration file can be seen in the appendix in Listing A.2.

A simplified visualization of the preCICE configuration we use for the ocean-ocean solver can be seen in Figure 3.6. The original configuration is also available in the appendix in Listing A.2. The configuration is considerably more complicated because more data needs to be exchanged compared to the one-way coupling case. As explained above, the respective data exchanges between SWEET and OpenFOAM do not occur on the same mesh in both directions. Instead, each solver has two meshes. One for the sides of the domain and one for the area over which the coupling occurs. They form a loop, with each solver writing and reading to and from the other solver. This also allows for more complex coupling schemes.

The configuration defines four data types, although they are only two pairs of the same data types for both mesh types. The water height is exchanged as a scalar value containing the entire water height from the surface to the ground in m. The velocities are transmitted as a 3D vector of x, y, z-direction velocities in m/s. Theoretically, it would be sufficient to only use a 2D vector for the velocities as SWEET is 2D, and the z-value will always be 0 or unused, thus saving bandwidth. However, because of the dimension of the mesh in preCICE and the necessity for 3D locations for the mesh cells, the vector automatically defaults to a 3D vector.

As mentioned above, SWEET is a 2D solver, and OpenFOAM is a 3D solver. Therefore, at some point, interpolation has to be done from 2D to 3D. In the other direction, a reduction from 3D to 2D is needed. As the communication between the processes can be one of the significant bottlenecks of coupling, it is in the best interest of performance to minimize the number of transmitted values. Therefore, the mapping between the meshes on the sides is done on the OpenFOAM side because the SWEET mesh for the sides consists of four 1D lines while OpenFOAM has four 2D rectangles. The mapping from 3D to 2D is done in the OpenFOAM preCICE adapter as explained in Section 3.2.3, which results in OpenFOAM already providing a 2D mesh to preCICE there. For mapping this mesh, while the mesh shape is the same as in SWEET, OpenFOAM still uses a finer grid, resulting in a larger number of values in the mesh. Therefore, the mapping on the top is also performed on the OpenFOAM side.

In contrast to the atmosphere-ocean solver, which only supports one-way coupling, this solver could also utilize implicit coupling. It can be turned on in case the scenario is not stable for explicit coupling, but we choose to use serial explicit coupling to prioritize runtime.

3.2.3. OpenFOAM Participant

We use the `interFOAM` solver for OpenFOAM, which simulates multiphase incompressible flow. We configure the two phases, water and air. The kinematic viscosity is set to $\nu_w = 1 \times 10^{-6}\,\mathrm{m}^2/\mathrm{s}$ and $\nu_a = 1.48 \times 10^{-5}\,\mathrm{m}^2/\mathrm{s}$ respecitvely. The density is set to $\rho_w = 1000\,\mathrm{kg}/\mathrm{m}^3$ and $\rho_a = 1\,\mathrm{kg}/\mathrm{m}^3$. Gravity is set to use the same value as SWEET. The simulation type is set to laminar flow, as turbulence is not the focus of our simulated scenarios. For the solvers, we do not use any special settings but standard settings from one of the OpenFOAM tutorials.

One of the advantages of preCICE is its API, which allows adapters to exist for many common solvers. This is also the case for OpenFOAM, which already has an existing adapter [6]. We use this adapter as the basis for connecting OpenFOAM to preCICE. However, the 2D-3D conversion needs to be implemented on the OpenFOAM side, as described in Section 3.2.2, and the adapter does not support this data processing. Therefore, it is necessary to make some additions to the adapter to work with this coupling configuration.

The conversion from a 3D mesh to a 2D mesh and vice-versa would not necessarily need to have been made in the OpenFOAM adapter itself. Another possibility would be to do it in preCICE before exchanging values to the other process. There was already some work on geometric multiscale coupling [7], and preCICE has a feature for geometric multiscale mapping[4]. However, it only supports 1D-3D coupling on a circular interface. For our manual implementation of the dimensionality mapping, we

[4]`https://precice.org/configuration-mapping#geometric-multiscale-mapping` Accessed: 2025-01-12

choose the adapter instead of preCICE because it has direct access to the OpenFOAM API, which provides valuable functions.

As explained in Section 2.2.2, the preCICE adapter uses a class for every data type that can be exchanged. Therefore, we only have to modify the corresponding classes `Velocity` and `Alpha` to add our changes.

preCICE to OpenFOAM

In Listing 3.1, the code for converting water height to alpha can be seen. It iterates over all patches which the adapter is configured to deal with. In our case, this is just the single `Sides` patch containing the four sides of the OpenFOAM domain. Then, it iterates over every cell in the patch, calculating the alpha value. The nearest-neighbor mapping in preCICE automatically handles the 2D to 3D mapping. However, to check which alpha value should be chosen for the water height received from preCICE, the location of the cell is needed. For this, the `cell.box()` function from the OpenFOAM API is used to get a bounding box of the cell in the form of a point-tuple. With these two points, the minimum and maximum height of the cell are determined and used to implement Equation 3.4. The `snappyHexMesh` algorithm used for mesh generation does not solely produce cuboids, so equating the ratio of water height to the ratio of volume is not always correct. However, because the splitting and snapping of cells only occurs near obstacles, we can safely assume that all cells at the boundary are cuboids or very close to cuboids.

```
void preciceAdapter::FF::Alpha::read(double *buffer) {
    int bufferIndex = 0;

    // For every boundary patch of the interface
    for (uint j = 0; j < patchIDs_.size(); j++) {
        int patchID = patchIDs_.at(j);
        auto &mesh = Alpha_->mesh();
        auto &patch_cells = mesh.boundary()[patchID].faceCells();
        auto &cells = mesh.cells();

        // For every
        for (auto &c_label: patch_cells) {
            auto water_height = buffer[bufferIndex++];
            auto c = cells[c_label];

            // Calculate cell height with bounding box
            auto b_box = c.box(mesh);
            double top = max(b_box.first().z(), b_box.second().z());
            double bottom = min(b_box.first().z(), b_box.second().z());

            if (top < height) {
                Alpha[c_label] = 1;
            } else if (bottom >= height) {
                Alpha[c_label] = 0;
            } else {
                Alpha[c_label] = (water_height - bottom) / (top - bottom);
            }
        }
    }
}
```

Listing 3.1: Reading of alpha values from preCICE to OpenFOAM

The writing of the velocity values from preCICE to the OpenFOAM mesh can be seen in Listing 3.2. Overall, the implementation is very similar to that of the alpha values. The difference is that since only the velocity for water cells is supposed to be set, the alpha value of each cell is also needed. Therefore, the `Velocity` class is extended with a reference to the alpha field. For this reason, it is also necessary that the alpha field is written before the velocity field in each timestep. Otherwise, outdated values would be used for the calculation.

```
void preciceAdapter::FF::Velocity::read(double* buffer) {
    int bufferIndex = 0;

    // For every boundary patch of the interface
    for (int j = 0; j < patchIDs_.size(); j++) {
        int patchID = patchIDs_.at(j);
        auto &patch_cells = U_->mesh().boundary()[patchID].faceCells();

        // For every cell of the patch
        for (int i = 0; i < patch_cells.size(); i++) {
            auto cell = patch_cells[i];
            auto alpha = Alpha[cell];
            U[cell].x() = buffer[bufferIndex++] * alpha;
            U[cell].y() = buffer[bufferIndex++] * alpha;
            U[cell].z() = buffer[bufferIndex++] * alpha;
        }
    }
}
```

Listing 3.2: Reading of velocity values from preCICE to OpenFOAM

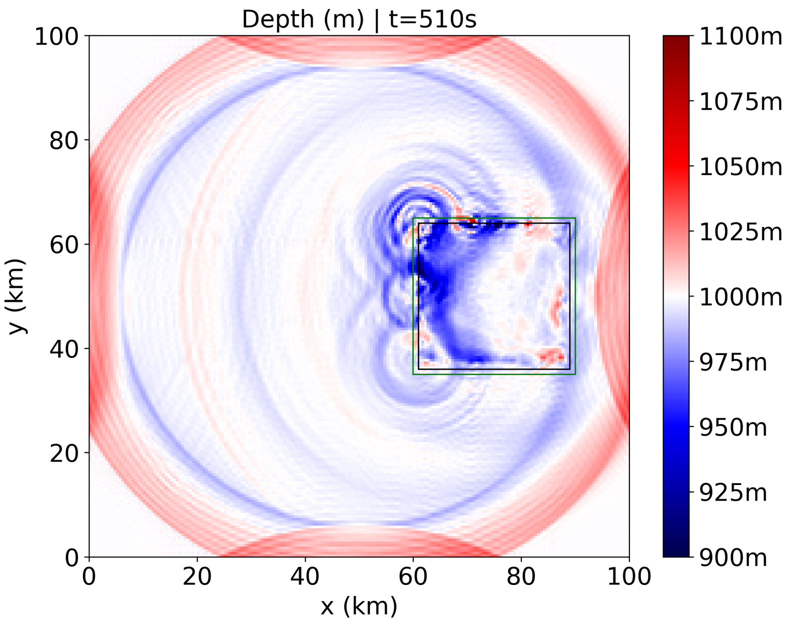


Figure 3.7.: Artifacts appearing in the coupling region when values are written to the boundary instead of the cell in OpenFOAM.

A detail that can be observed in the two code snippets above is that the values of alpha and velocity are not written to the boundaries, i.e., the value field of the `fixedValue` BCs, but to the value of the cells directly adjacent to these boundaries. This is because when values were written to the boundaries and `fixedValue` boundaries were used in early versions, anomalies appeared at the boundaries that prevented accurate simulations. An example of these artifacts can be seen in Figure 3.7. When using the current method of writing the values to the cells, these artifacts stopped appearing. We are not entirely sure of the cause of these artifacts, but we suspect they are related to the time at which the BCs were applied. Consequently to this implementation, the

OpenFOAM configuration does not use `fixedValue` BC as described in Section 3.2.1, but `zeroGradient` Neumann BCs, since the cell value is already set manually.

OpenFOAM to preCICE

The changes to the adapter for the other coupling direction are more extensive as the 2D-3D mapping is not handled automatically by preCICE. The same calculation has to be performed for alpha and the velocity, and it would be inefficient to iterate over the entire OpenFOAM domain twice. However, because of the preCICE adapter's structure, it is impossible to unify these methods in one of the classes without significant changes. For this reason, we introduce a secondary class that calculates the mapping of the alpha and velocity fields, which can be queried using the respective methods `get_v()` and `get_alpha()`. We call this class `DimensionReduction`, and both `Velocity` and `Alpha` receive a reference to it. It does not need to be called separately but automatically calculates the mapping again after both data types have been read. How the adapter writes these velocity values from OpenFOAM to preCICE using this reduction can be seen in Listing 3.3. The `Alpha` class works analogously. All values are already prepared and simply passed to the buffer.

```cpp
std::size_t preciceAdapter::FF::Velocity::write(double* buffer) {
    int bufferIndex = 0;

    for (auto v : reduction.getV()) {
        buffer[bufferIndex++] = v;
    }
    return bufferIndex;
}
```

Listing 3.3: Writing of velocity values from OpenFOAM to preCICE

The purpose of the `DimensionReduction` class is to implement Equation 3.3 and Equation 3.5. It would ideally iterate over all cells in a vertical column, starting at each of the faces forming the `Top` boundary. The issue with this is that OpenFOAM uses an unstructured grid. The grid is not defined by three-dimensional indexes but rather by an unordered list of cells, each with a list of faces and lists of owners and neighbors, which gives information about which cells border a given face. One option to implement this would be to iterate over all faces of the boundary cell, select the face at the bottom, find out which cell is on the other side, and repeat this process until reaching the ground. However, while this provides a good solution for a structured mesh that is saved as an unstructured mesh, it still assumes a particular structure and that there are no tetrahedrons or similar. However, when introducing obstacles and `snappyHexMesh`, these assumptions do not hold.

To facilitate an accurate implementation of the mapping, we make an approximation. Instead of checking all cells in a vertical column, we use the sampling utility of the OpenFOAM API to create a uniformly sampled line from each boundary face to the ground. We assume that given that the sample size is large enough, this yields a close enough result to the truth, no matter the actual shape of the mesh. How this `uniformSet` class in OpenFOAM is used in the `DimensionReduction` is shown in Listing 3.4.

```
preciceAdapter::FF::DimensionReduction::DimensionReduction(...) : ... {
    meshSearch searchEngine(mesh);
    int counter = 0;

    // For every boundary patch of the interface
    for (int patchID : patchIDs) {
        auto &temp_mesh = Alpha->mesh();
        auto &face_centers = temp_mesh.boundary()[patchID].Cf();

        // For every water column construct a vertical sample set
        for (const auto &center: face_centers) {
            sample_size = center.z() / n_samples;
            double offset = sample_size / 2;
            point start(center.x(), center.y(), center.z() - offset);
            point end(center.x(), center.y(), offset);

            sampleSets.emplace_back(temp_mesh, searchEngine, "z", start,
                end, n_samples);
        }
    }
}
```

Listing 3.4: Initialization of `DimensionReduction` class

In the constructor of the `DimensionReduction`, for each boundary patch and each cell on this patch, a `uniformSet` is created starting at the point and going to the bottom. The `uniformSet` works by selecting `n_samples` points on the line between the two points, with the first sample being the start and the last sample being the endpoint. It is then possible to access the corresponding cells each point lies in using the `cells()` function. To make this approximation as accurate as possible, we set `n_samples` to be equal to the number of vertical cells in the `blockMesh` that `snappyHexMesh` bases its mesh upon. Additionally, we offset both the start and end by the height of half a cell. This results in the samples corresponding to the cell centers in the `blockMesh`. Far away from the obstacles, at the edges of the domain, this corresponds exactly to the mesh, and we obtain the same result there using this method as the formula describes. Therefore, the only approximations happen in the inner parts of the OpenFOAM domain, which is not as important since it does not affect the SWEET domain for the most part and is mainly used to fill the hole in the SWEET domain.

How these sets are used can be seen in the calculation method in Listing 3.5, which is executed once every timestep. With the work of collecting the necessary cells already done, it simply needs to iterate over the list of `uniformSets` and apply Equation 3.3 and Equation 3.5 to the cell lists they contain. The creation of the sample sets is only done once in the constructor, and the sampling process is never repeated. Therefore, calculating all necessary cells only incurs a runtime cost during setup, and there is no sampling cost at runtime.

```cpp
void preciceAdapter::FF::DimensionReduction::calculateFields() {
    velocities.clear();
    heights.clear();

    for (uniformSet &set : sampleSets) {
        double alpha_sum, u_sum, v_sum;

        for (auto &cell : set.cells()) {
            alpha_sum += Alpha[cell];
            u_sum += V[cell].x() * Alpha[cell];
            v_sum += V[cell].y() * Alpha[cell];
        }

        // If there are obstacles in this column
        if (set.cells().size() < n_samples) {
            heights.push_back(DEFAULT_HEIGHT); // Prevents Extreme values
        } heights.push_back(alpha_sum * sample_size);

        velocities.push_back(u_sum ? u_sum / alpha_sum : 0)
        velocities.push_back(v_sum ? v_sum / alpha_sum : 0)
        velocities.push_back(0); // preCICE expects 3D vector
    }

    calculated = true;
    read_H = false;
    read_V = false;
}
```

Listing 3.5: Calculation of 2D values in `DimensionReduction` class

A special edge case needs to be handled with a check if there are obstacles in a water column. If there are obstacles, the sample set will be smaller than expected. In the case of an obstacle that goes from the bottom to the surface but has air above, this set would only contain only air cells. The calculated water height would thus be 0, causing spectral artifacts in the SWEET domain. Implementing a perfect calculation of water height for this case would require more complicated interactions with the OpenFOAM mesh. Therefore, as obstacles do not appear on boundaries and the values in the center are only needed to keep the SWEET domain consistent and prevent artifacts, the water height at these locations is set to h_0.

3.2.4. SWEET Participant

Contrary to OpenFOAM, there is no preCICE adapter yet for SWEET. Therefore, the necessary preCICE API calls must be manually added to the program. However, because of the simple API preCICE uses, this only results in limited changes. As outlined in Section 2.1.4, SWEET uses program classes to implement different scenarios using the SWEET framework's functionality. Our implementation is constructed based on the `PDE_SWECart2D` and `PDE_SWESphere2D` programs.

A simplified version of the main program can be seen in Listing 3.6. It is split into two parts: the initialization and the simulation loop. First, SWEET is initialized, and preCICE is set up with the configuration in Figure 3.6. Before beginning the simulation, both meshes need to be defined. No connectivity is needed, so this is simply an iteration over the coordinates of all relevant SWEET cells. For the `Cart2D` domain, the coordinates can be taken directly from the cells of SWEET's Cartesian domain. For the `Sphere2D` domain, this is not possible as it uses a latitude-longitude grid. No projection exists that preserves distance, area, and angles. For simplicity, we choose to implement equirectangular projection, where latitude and longitude are directly converted to the x- and y-component of a point after being appropriately scaled to the circumference of the sphere. Also, as preCICE assumes the initial coupling values to be 0, the correct water height for $t = 0\,\mathrm{s}$ must be written to preCICE. Afterward, preCICE is initialized, and simulation can begin.

The simulation loop has two criteria for continuing execution. On the one hand, it lets preCICE decide if the simulation should continue. On the other hand, it checks if any events on the SWEET side indicate that execution should stop, e.g., a failed instability check. As a first action in each iteration, SWEET simulates the timestep. As subcycling is disabled, preCICE dictates the timestep size. After each timestep is calculated, the velocity and water height are converted from the spectral domain, where the simulation runs, to the physical domain. The current values are then written to preCICE, and the new values are read from preCICE and written to the data grids. These can now be loaded into the simulation by returning them to the spectral domain. For the `Sphere2D` domain, an additional step not shown in Listing 3.6 is necessary to convert phi, vorticity, and divergence to the water height and velocities.

```
int main(int i_argc, char *i_argv[]) {
    PDE_SWECart2D::Program simulation(i_argc, i_argv);
    precice::Participant precice{"SWEET", "precice-config.xml", 0, 1};

    <...> // Populate coordinates and IDs for sides
    precice.setMeshVertices("SWE_Sides", coordinates, vertexIDs_write);
    <...> // Populate coordinates and IDs for top
    precice.setMeshVertices("SWE_Top", coordinates, vertexIDs_read);

    if (precice.requiresInitialData()) precice.writeData("SWE_Sides","
        Alpha_SWE_FOAM", ...);

    precice.initialize();

    while (!simulation.should_quit() && precice.isCouplingOngoing()) {
        simulation.precice_max_dt = precice.getMaxTimeStepSize();
        simulation.runTimestep();

        auto h = simulation.dataConfigOps.prog.h_pert.toGrid();
        auto u = simulation.dataConfigOps.prog.u.toGrid();
        auto v = simulation.dataConfigOps.prog.v.toGrid();
        <...> // Write data to buffer

        precice.writeData("SWE_Sides", "Velocity_SWE_FOAM", ...);
        precice.writeData("SWE_Sides", "Alpha_SWE_FOAM", ...);
        precice.readData("SWE_Top", "Velocity_FOAM_SWE", ...);
        precice.readData("SWE_Top", "Alpha_FOAM_SWE", ...);
        <...> // Write data to grid

        simulation.dataConfigOps.prog.u.loadCart2DDataGrid(u);
        simulation.dataConfigOps.prog.v.loadCart2DDataGrid(v);
        simulation.dataConfigOps.prog.h_pert.loadCart2DDataGrid(h);

        precice.advance(simulation.shackTimestepControl->
            currentTimestepSize);
    }
    precice.finalize();
}
```

Listing 3.6: Simplified structure of the SWEET program including calls to preCICE API

Configuration

The tests use the default build configuration of the `PDE_SWECart2D` and `PDE_SWESphere2D` programs in SWEET. As a timestepping method, we select fourth order ERK as described in section Section 2.1.3 for all parts of the right-hand side of the SWE equations. It is not a very complex time integration scheme, but it is widely used and appropriate since the actual performance of the solver is not the focus of this thesis. Furthermore, SWEET also requires setting either the number of modes in spectral space or the number of cells in physical space. The appropriate value for the respective other parameter is automatically determined. To make the configuration of the coupling and placement of coupling boundaries in the scenarios easier, we select a physical grid size of 200×200 cells for the `Cart2D` domain. For the `Sphere2D` domain, we use 200 spectral modes, as SWEET did not allow us to determine the domain size using the physical grid. Some natural constants concerning the fluids also need to be set.

- $g = 9.80161\,\mathrm{m^2/s^2}$ Constant for gravitational acceleration
- $f_0 = 0\,\mathrm{rad/s}$ Coriolis parameter at equator
- $\nu = 1 \times 10^{-6}\,\mathrm{m^2/s}$ Kinematic viscosity

Both g and ν are selected to match the scenario we need to simulate. Being equal to the Earth's gravity and water's kinematic viscosity. However, ν needs to be changed in the source code, as SWEET was not initially meant to simulate oceanic flows. It might be interesting to include Coriolis forces in the context of an oceanic flow. However, because OpenFOAM does not support them, they would be unevenly applied between the domains and complicate the coupling. Therefore, f_0 is set to be equal to no rotation.

3.2.5. Evaluation

To test the functionality and accuracy of the implemented coupling, we run multiple test scenarios with our coupled solver. All scenarios take place in the same decomposed domain described in Section 3.2.1. The general idea of the scenarios is to simulate bi-directional interactions between OpenFOAM and SWEET.

SWEET

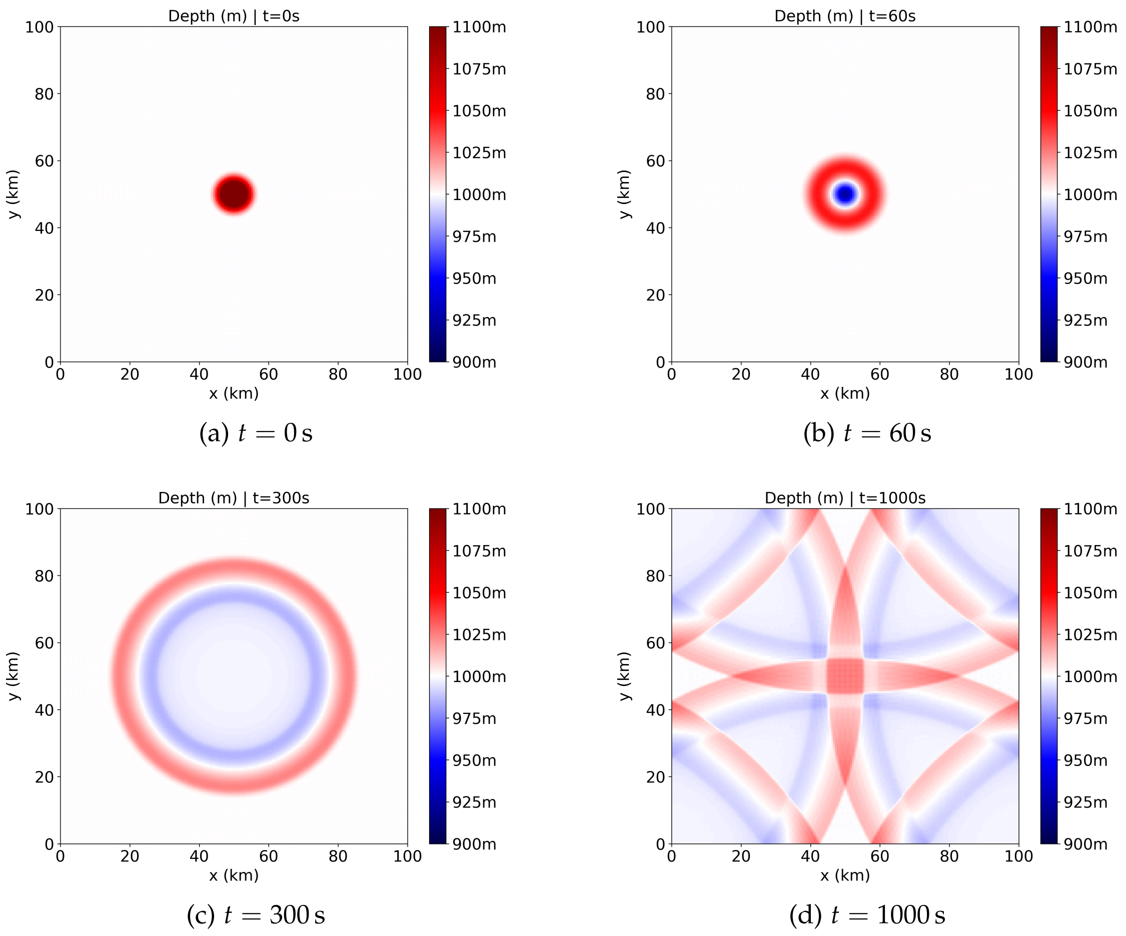
In line with a possible use-case for this sort of coupling, tsunami simulation, we want to simulate some excitement in the SWEET domain that travels to and interacts with the OpenFOAM domain. Although for some early tests, different setups were evaluated, we used the same setup for SWEET in all final tests. At the initial state, at the center of the SWEET domain, we place a circular column of water. Because of the spectral SWEET domain, we do not implement this column using a step function but rather use the cosine function. If we used a discontinuous step function, the Fourier Transform would cause oscillations near the step because of the Gibbs phenomenon. [36] For any point in a circle with radius d_{max}, the water height at a distance d from the center is calculated using this equation:

$$h(d) = 0.5 * \left(\cos\left(\frac{d}{d_{max}} * \pi\right) + 1\right) * h_{center} + h_0, \forall d < d_{max} \tag{3.6}$$

In order for the initial condition to be accurately represented and not include distortions, we do not use equirectangular projection to calculate the distance from the center in the `Sphere2D` domain. Otherwise, it would be hard to tell if any distortions in the circular wave are caused by the initial projection or problems with the coupling. To accurately calculate the distance from the central point, we calculate the great circle distance using the haversine function. [14] Let (ϕ_1, λ_1) and (ϕ_2, λ_2) be two points defined by latitude and longitude, and r be the radius of the sphere. Then, the great-circle distance is given by

$$d = 2r \cdot \arcsin\left(\sqrt{\frac{1 - \cos(\Delta\phi) + \cos(\phi_1) \cdot \cos(\phi_2) \cdot (1 - \cos(\Delta\lambda))}{2}}\right) \tag{3.7}$$

This column scenario is similar to the popular dam break scenario, which would not be able to work because SWEET does not support walls. The water height at the center is increased by $h_{center} = 200\,\text{m}$, and the outer boundary d_{max} is at 7.5 km from the origin. The ground of the domain is left entirely flat. This generates a water "hill" that collapses and forms a singular wave.



(a) $t = 0\,\text{s}$

(b) $t = 60\,\text{s}$

(c) $t = 300\,\text{s}$

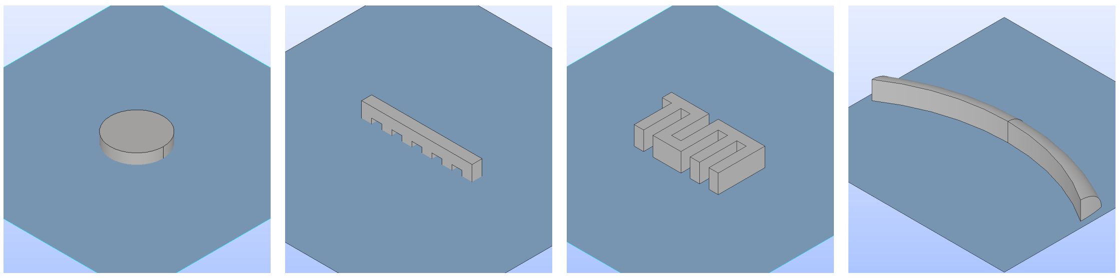
(d) $t = 1000\,\text{s}$

Figure 3.8.: Water height at different timesteps in an uncoupled simulation of the test `Cart2D` test scenario by SWEET.

This scenario only consists of an initial excitement, and no continued force is applied. It is a time-dependent scenario expected to converge to a flat, undisturbed ocean. An uncoupled simulation of this scenario by SWEET can be seen in Figure 3.8. While a stable state scenario would be interesting, it is challenging to find such a scenario. SWEET's periodic domain cannot contain holes, inlets, and outlets, so they would only be possible in the OpenFOAM domain. However, this does not resemble any use case, and if both inlet and outlet lie adjacent in the OpenFOAM domain, the scenario might converge to a stable state where the SWEET domain is not disturbed much at all. An interesting stable state scenario might be two different coupled instances of OpenFOAM lying in different parts of the SWEET domain, one containing an inlet and the other containing the outlet. However, this is an advanced coupling setup and is not helpful for testing the basic coupling functionality.

OpenFOAM

The simplest scenario we use for OpenFOAM is a water domain without obstacles at rest that is initialized to the same water height as the SWEET domain. This scenario is beneficial visually and empirically since its results can be easily verified by simulating the SWEET scenario without any coupling. However, the motivation of coupling SWEET to OpenFOAM or a solver like it is to add simulation capabilities in these areas that are not available to SWEET or any SWE solver by itself. Therefore, while this empty scenario is helpful for testing, it does not show the correct functionality of the primary use case for this thesis. To examine this, we implement four scenarios, each placing a different obstacle in the center of the OpenFOAM domain. These four obstacles can be seen in Figure 3.9.



(a) Cylinder (b) Bridge (c) TUM Logo (d) Reflector

Figure 3.9.: The four obstacles placed in the OpenFOAM domain during the tests.

The cylinder implements a simple obstacle, which cannot usually exist in SWEET scenarios. The TUM logo represents a complex obstacle to check if this has an influence on accuracy. The bridge obstacle goes a step further. It is not the product of a single extrusion of a horizontal face, i.e., it can not be reduced to the 2D plane. Therefore, it represents an obstacle that, by definition, cannot be simulated by an SWE solver since they implement 2D equations. Lastly, we construct a reflector obstacle that models a circle segment. Theoretically, this should reflect the wave and create a focal point at the wave origin. We use this obstacle to check objectively if wave direction and behavior are consistent across the boundary.

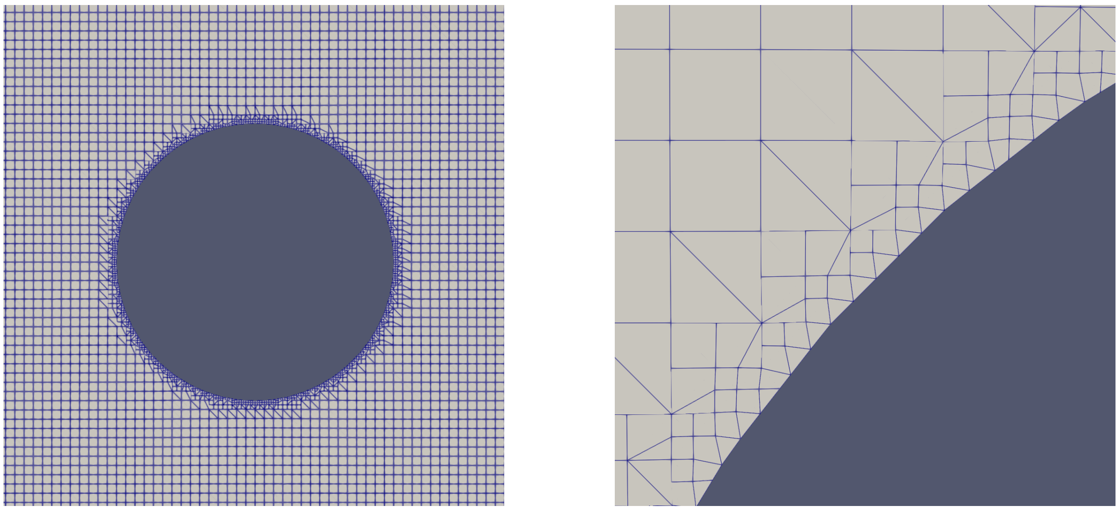

(a) Overview (b) Detailed view

Figure 3.10.: Top-down view of the mesh constructed around the cylinder obstacle.

To achieve good simulation accuracy and prevent the appearance of confusing artifacts, a good mesh is essential, though it can depend on the scenario what "good" means. [22] To create the meshes for the OpenFOAM scenarios, we use two tools. First, to create the general geometry, we use SALOME, an easy-to-use open-source CAD program from which we export the geometry as `.stl` files. This automatically triangulates the analytically defined objects. As a second step, we use the `snappyHexMesh`[5] tool, which is included in OpenFOAM. The basic concept is to improve a simple input mesh using stepwise refinements with different methods. We provide the initial mesh using the `blockMesh` utility, also used to create the mesh for the simulations without obstacles. The `snappyHexMesh` algorithm first subdivides the mesh near obstacle edges. Then, all cells lying inside the obstacle with more than 50% of their volume are removed. In a snapping step, the vertices on the mesh boundary are then displaced to the obstacle edges. The resulting mesh is then iteratively refined using relaxation to satisfy mesh quality parameters. A top-down view of the mesh for the cylinder cylinder scenario can be seen in Figure 3.10.

[5]`https://www.openfoam.com/documentation/user-guide/4-mesh-generation-and-conversion/4.4-mesh-generation-with-the-snappyHexMesh-utility` | Accessed: 11.02.2025

3.3. Summary

In this chapter, we implemented two solvers. The first one is an atmosphere-ocean solver, which, because of limitations by SWEET, is only able to be coupled unidirectional and is thus only used in a proof-of-concept scenario. The second one is a two-way-coupled ocean-ocean solver that we implement both for a Cartesian and spherical SWEET domain. The solver can be used to simulate parts of the SWEET domain using OpenFOAM in order to improve accuracy and simulate complex geometry. This second solver is tested for its viability using a benchmark scenario and multiple obstacles.

4. Results

We run comprehensive tests to evaluate the performance of the solvers implemented in Chapter 3. We primarily investigate the viability of using SWEET in a coupled solver. This chapter presents the results of these tests and is structured in two parts. The first part Section 4.1 briefly goes over the proof-of-concept tests on the atmosphere-ocean solver. As this solver only allows for one-way coupling due to the reasons outlined in Section 3.1, the remainder of the chapter focuses on the ocean-ocean solver. Section 4.2 provides a detailed evaluation of the ocean-ocean solver. First, it evaluates coupling with both available SWEET domain types, then tests the functionality of obstacles in the OpenFOAM domain.

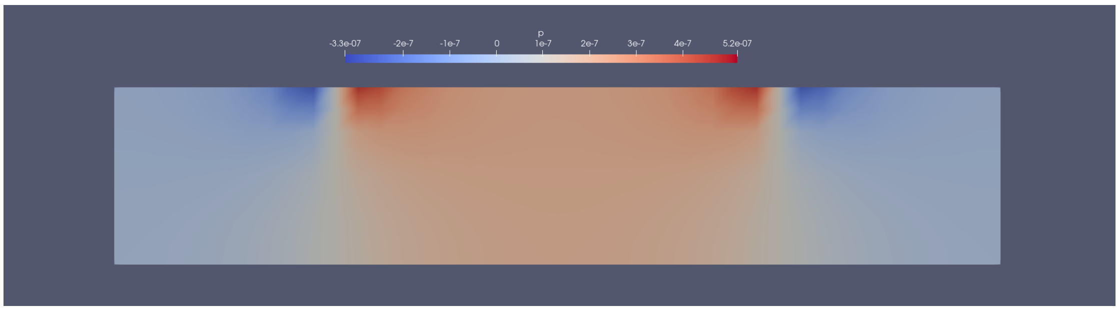
4.1. Atmosphere-Ocean Solver

The first solver we analyze is the atmosphere-ocean solver, with SWEET simulating the atmosphere part of the domain and OpenFOAM simulating a three-dimensional ocean domain. As described in Section 3.1, the atmosphere-ocean solver has some limitations that arise from the limitations imposed by using the SWEET solver for the atmosphere. Because of these limitations, the tests in this section do not evaluate any metrics or try to simulate a specific scenario but rather a proof-of-concept scenario to test the viability of the coupling should the aforementioned limitations be alleviated in the future.

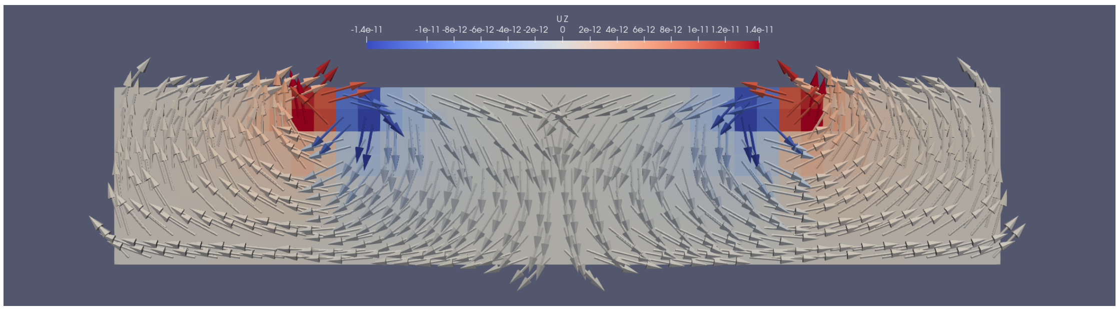
Our test scenario is described in Section 3.1.5. It starts with two jet streams in the SWEET domain and an ocean at rest in the OpenFOAM domain. The initial state of the SWEET domain can be seen in Figure 3.3. As this is a stable state and the atmosphere-ocean solver only supports one-way coupling from SWEET to OpenFOAM, this continues to be the state of SWEET throughout the simulation.

The simulation results in the OpenFOAM domain can be seen in Figure 4.1. The figure shows a cross-section of the domain. At the top, the two jet streams from Figure 3.3 can be seen. Through their influence on the top layer of the domain, they cause similar streams in the OpenFOAM domain. Additionally, as shown in Figure 4.1b, a rotational current forms along each jet stream, and a high-pressure area forms between the streams.

These results show that despite the restrictions of the atmosphere-ocean solver explained in Section 3.1.1, the coupling works in principle. The coupled values from



(a) Pressure



(b) Vertical Velocity

Figure 4.1.: Side-view of the results from the atmosphere-ocean solver. The image is taken in positive-X direction.

SWEET are successfully transmitted and influence the OpenFOAM domain.

4.2. Ocean-Ocean Solver

As a second and more viable possibility of coupling SWEET to a three-dimensional solver, we constructed the ocean-ocean solver described in Section 3.2. This section presents the results obtained when testing this solver.

The section is divided into three parts. First, it deals with simulations in the `Cart2D` domain, the simpler of the two SWEET domains. It performs both a qualitative and quantitative analysis of the results and examines the influence of parameters such as cell size of the OpenFOAM domain. Next, the other SWEET domain, `Sphere2D`, is analyzed in the same way as the `Cart2D` domain. Here, we pay attention to the inaccuracies caused by the projection between spherical and Cartesian domains. Lastly, the solver's behavior in one of its use cases is analyzed by placing obstacles inside the OpenFOAM domain. Tests are conducted using several obstacles. Though the inability of SWEET to simulate obstacles on its own prevents quantitative comparisons, correct functionality is analyzed using a specifically designed reflector obstacle.

4.2.1. Cart2D Domain

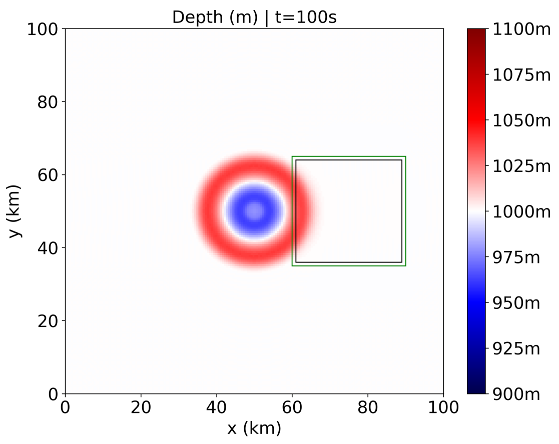
The `Cart2D` domain is the focus of the first tests we conduct. We start with the simplest scenario introduced in Section 3.2.5, where the OpenFOAM domain is simply an empty domain at rest. The SWEET domain starts with a smooth column of water at its center. Upon collapsing, this results in a large wave traveling across the domain. In this scenario, OpenFOAM employs no capabilities that are not available to SWEET. Therefore, we can simulate a benchmark where SWEET simulates the entire scenario without domain decomposition. The results of this can be seen in Figure 3.8. The scenario is simulated for a duration of 1000 s with a timestep of 0.5 s. This time frame is approximately equal to the time it takes the gravity wave caused by the water column to travel once around the entire domain. This means that a wave crosses the coupled region twice, once from the left and once from the right.

A comparison of the SWEET simulation without domain decomposition and the simulation with ocean-ocean coupling can be seen in Figure 4.2 and Figure 4.3. The data shown in the graphs is the data representation of the SWEET domain. For the OpenFOAM domain, this means that the values shown in the graphs result from the preCICE interpolation at the locations of the SWEET cells. Since, for our simulations, OpenFOAM uses a higher grid resolution, this representation is downsampled, and a higher resolution of the OpenFOAM domain is available by inspecting the OpenFOAM output files. However, this representation is sufficient for a global analysis as it provides an overview of the entire domain.

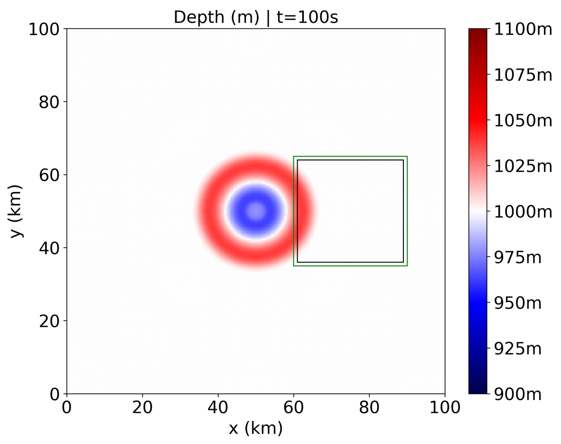
The simulation state is shown for four different points in time. The simulation state at $t = 300$ s and $t = 1000$ s is also shown in the benchmark in Figure 3.8. Overall, each of the renderings shows an important moment regarding the coupling. Figure 4.2a shows the moment the wave first travels between the SWEET and OpenFOAM domains. Figure 4.2c is the moment the eastbound wave crosses the OpenFOAM domain. Figure 4.3a is the moment the westbound wave passes the SWEET boundary and crosses the OpenFOAM domain from the east. Figure 4.3c shows the final state of the simulation and when all waves meet again in the middle. Notably, the north- and southbound waves also meet inside the OpenFOAM domain at this time.

There are some things to note about the results of the simulation. The overall quality looks very good: The wave travels as expected through the OpenFOAM domain, keeping the overall development of the scenario close to the benchmark. The wave also does not lose or gain speed when crossing between the domains, keeping a circular shape over both domains.

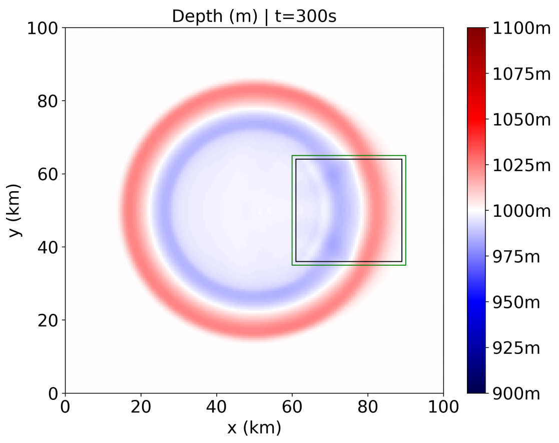
However, the edge of the wave is slightly smoothed when entering the OpenFOAM



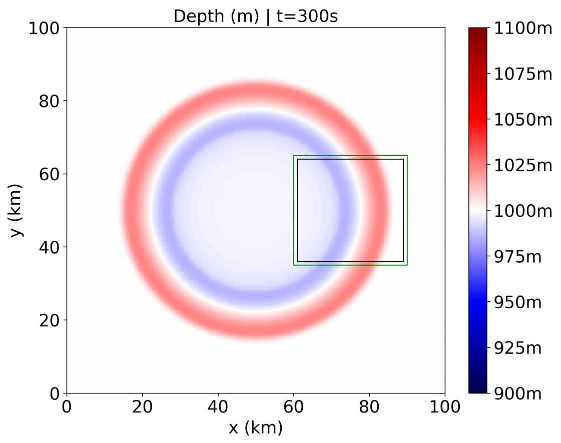
(a) Wave first enters SWEET domain.



(b) Benchmark $t = 100\,\mathrm{s}$



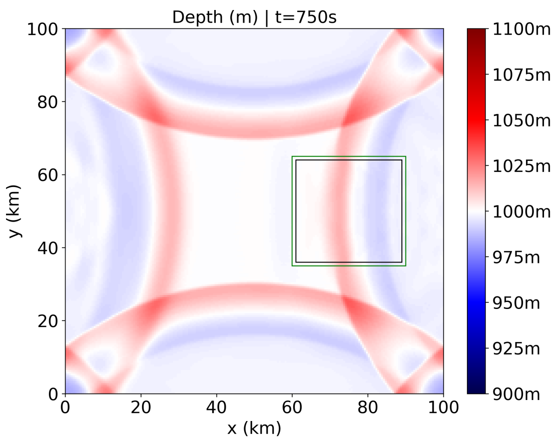
(c) Eastbound wave crosses SWEET domain.



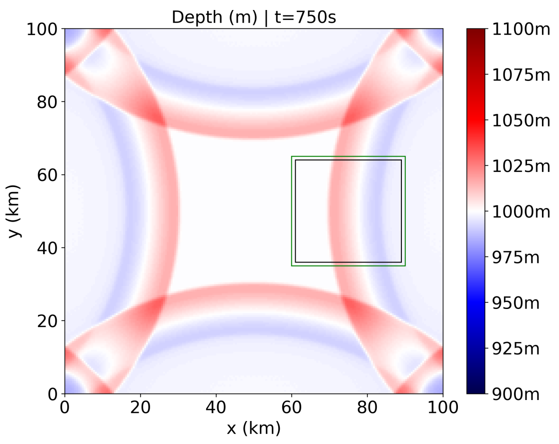
(d) Benchmark $t = 300\,\mathrm{s}$

Figure 4.2.: Comparison of water height at an early stage in the coupled `Cart2D` solver (left) and the benchmark (right).

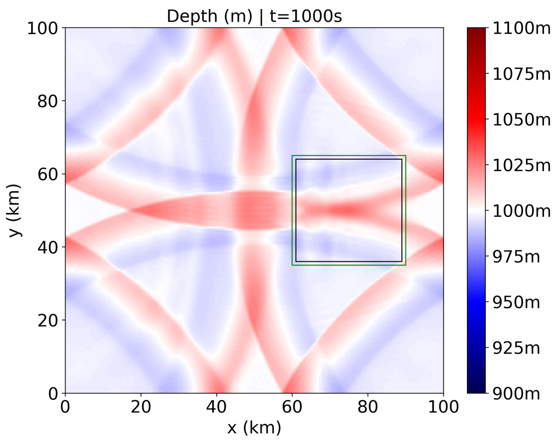
domain. There are two likely causes for this. The smoothing could be caused by the higher cell resolution in the OpenFOAM domain, which leads to fewer sharp edges in the wave. The other cause could be the intrinsic difference between the SWE and 3D-NSE. OpenFOAM can simulate vertical velocity gradients. Assuming that these velocity gradients are not zero in the OpenFOAM domain when being transformed back to the SWEET domain, the velocity is averaged again, which could cause smoothing. Also, OpenFOAM could model more accurate diffusive effects than SWEET, which was originally intended for atmosphere simulation.



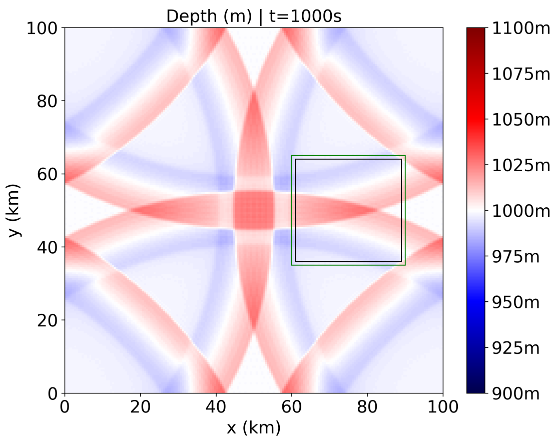
(a) Westbound wave crosses SWEET domain.



(b) Benchmark $t = 750\,\mathrm{s}$



(c) Wave meets again at the center.



(d) Benchmark $t = 1000\,\mathrm{s}$

Figure 4.3.: Comparison of water height at a late stage in the coupled `Cart2D` solver (left) and the benchmark (right).

Numerical Evaluation

While a visual comparison of results between the constructed ocean-ocean solver and our benchmark can be helpful, a quantitative measure of simulation accuracy can provide valuable insights. This is especially true when comparing different results.

One such metric we use for our evaluation is to check the time evolution of the water height at a specific point against the benchmark. This can either be plotted in a two-dimensional graph or summarized as a single scalar using the L_2 norm.

We use three different points to validate our model. The location of these points can

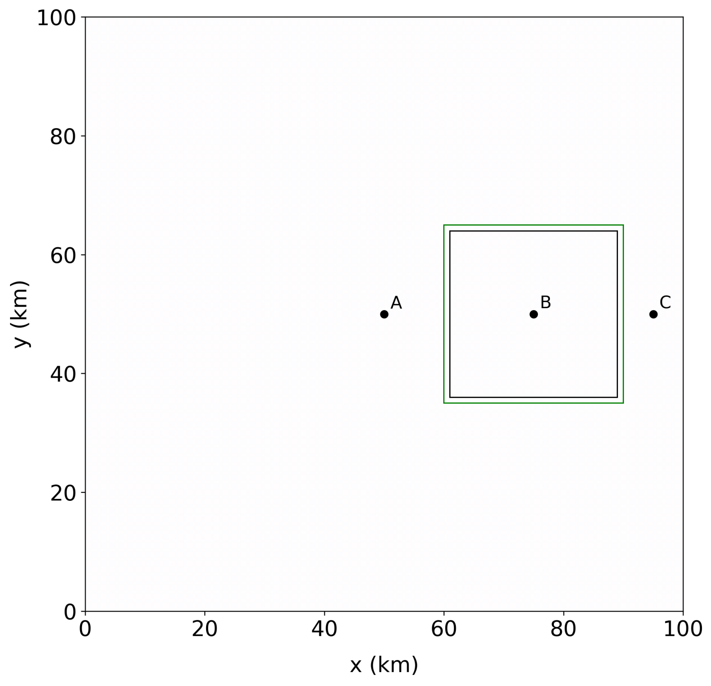


Figure 4.4.: Location of evaluation points in the decomposed `Cart2D` domain. The outer green line indicates the border of the OpenFOAM domain. The inner black line indicates the border of the SWEET domain.

be seen in Figure 4.4. Two of the points are in the SWEET domain. One of the points is inside the OpenFOAM domain for the coupled solver while still lying in the SWEET domain in the benchmark. They are strategically placed before, while, and after the coupling takes place to evaluate all regions of the coupling. The data of the points is captured at all timesteps, resulting in an interval of 0.5 s and 2001 values for each point.

Figure 4.5 shows the first of the evaluation points placed at the domain's center. This point is also the center of the water column at the beginning, leading to a prominent spike in the first 60 seconds as the column collapses and forms the wave central to the scenario. Afterward, there are no significant developments until $t = 950\,\text{s}$, when the waves meet again after traversing the entire domain once.

Mostly, the values provided by the coupled solver match the benchmark closely. The waves take the same time to traverse the domain and reach close to the same height after 950 s. However, the wavefront at $t = 950\,\text{s}$ is less steep than the benchmark, with an earlier start in elevation. This is consistent with the smoothing effect, which is also

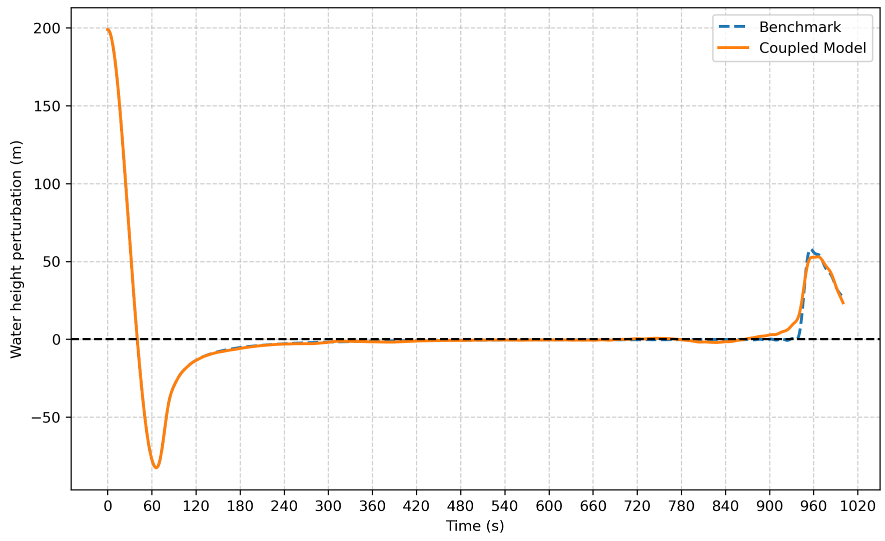


Figure 4.5.: Graph of the water level at Point A in the `Cart2D` domain.

observed visually.

Figure 4.6 shows the water height at Point B, which lies at the center of the OpenFOAM domain. When analyzing the Benchmark curve, there are three prominent structures to relate to Figure 4.2 and Figure 4.3. The first one is the wave peak and valley at $180\,\mathrm{s} \leq t \leq 360\,\mathrm{s}$. They correspond to the eastward wave traversing the OpenFOAM domain. Similarly, the peak at $700 \leq t \leq 840$ corresponds to the westward wave traversing the OpenFOAM domain. Lastly, the peak forming at $t = 960\,\mathrm{s}$ corresponds to the north- and southbound waves meeting inside the OpenFOAM domain, combining their peaks.

The coupled solver mostly reflects this, as all three features are present. They also appear at the same points in time. The same disparity as with Point A can be observed, where the wavefronts are less steep, and the waves begin building earlier. Similarly, the peaks are slightly later. Also, it can be observed that the wave peaks are up to $\approx 6\,\mathrm{m}$ higher for the single waves and even higher for the combined last wave. However, as both Point A and Point C show that the waves still have the same or slightly less height than in the benchmark, this might be attributed to a difference in wave shape between SWE and 3D-NSE, not an error in the energy conservation at the coupling. There is also a series of smaller waves between the main features in the coupled solver that do not appear in the benchmark. It is unclear whether this is a result of coupling inaccuracies or can be attributed to the different solver types.

The graph for Point C can be seen in Figure 4.7. It contains only one main feature. In

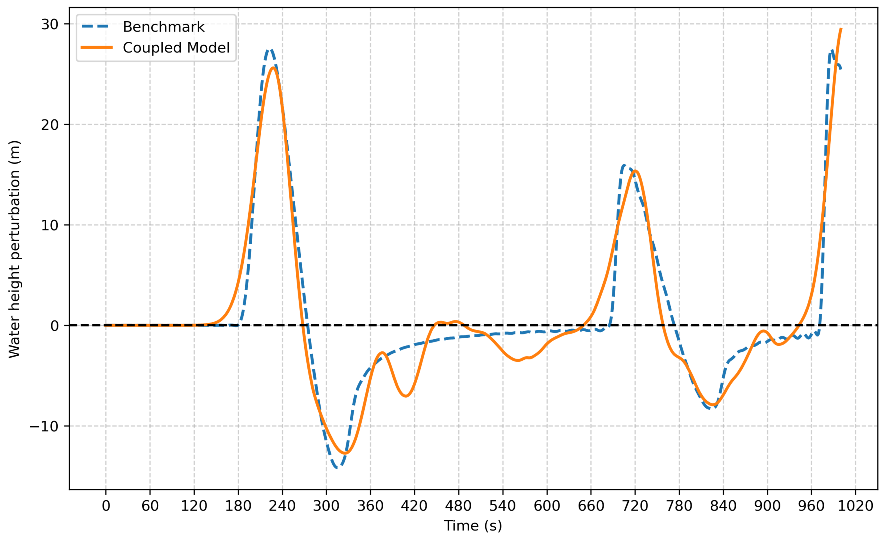


Figure 4.6.: Graph of the water level at Point B in the `Cart2D` domain.

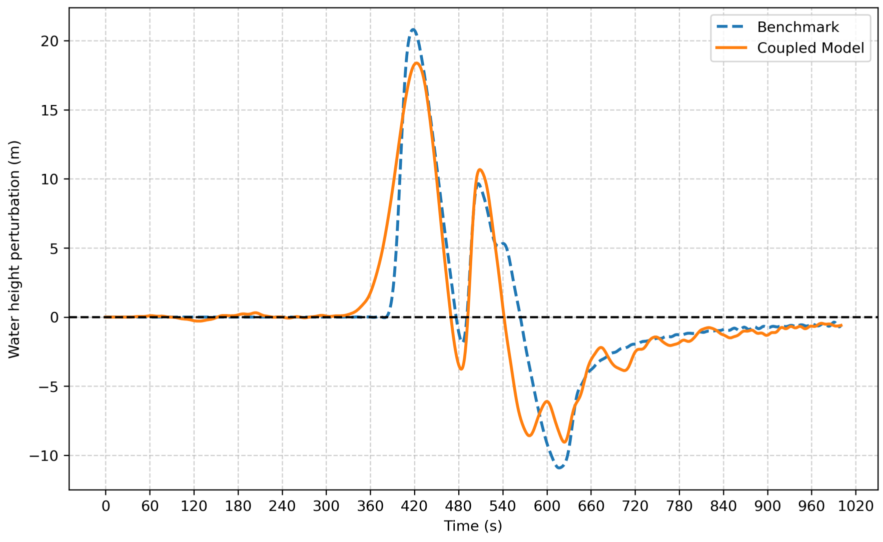


Figure 4.7.: Graph of the water level at Point C in the `Cart2D` domain.

the benchmark between $t = 360\,\mathrm{s}$ and $t = 720\,\mathrm{s}$, both the east- and westbound waves pass the point, partially overlaying one another. This results in multiple peaks and valleys. There are two prominent peaks at $t = 420\,\mathrm{s}$ and $t = 500\,\mathrm{s}$ and a valley at $t = 620\,\mathrm{s}$. These three features also appear in the coupled solver. The peaks and valleys are slightly less intense than the benchmark, which could result from the slightly more smoothed wave shape, causing the wave to be stretched out and reach less height. This is visible at $t = 330\,\text{s}$, where the wave starts building a whole minute earlier. More minor discrepancies also appear in small waves following the passing of the large waves.

With the results from the numerical analysis, our next step is to find out what influences the accuracy of the coupled solver. One such attribute we examine is the resolution of the OpenFOAM domain. For the use case of simulating a high-interest region in a global domain, the three-dimensional domain is typically much smaller relative to the entire domain. For our tests above, we use a relatively large OpenFOAM domain for better visualization of results and to amplify any adverse effects of the coupling in both domains. However, for tests increasing the granularity of the OpenFOAM domain, we choose to reduce the overall size as the cell count would otherwise increase the runtime of our experiments significantly. The smaller domain we use for the following test and the points used for evaluation can be seen in Figure 4.8. The location of the points does not change from the earlier analysis.

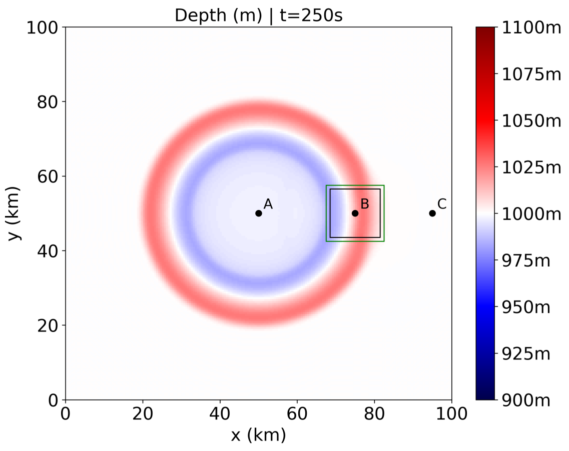


Figure 4.8.: Excerpt of the simulation with an OpenFOAM domain containing 150 × 150 × 20 cells in a smaller area. The outer green line indicates the border of the OpenFOAM domain. The inner black line indicates the border of the SWEET domain.

With this reduced-size domain, we run tests with three different numbers of cells in the OpenFOAM domain. We start with the configuration of 150 × 150 × 10 cells we used up until now. For our second configuration, we decrease the number of horizontal cells for a domain of 75 × 75 × 10. For our last test, we increase the number of vertical cells to reach 150 × 150 × 20 cells.

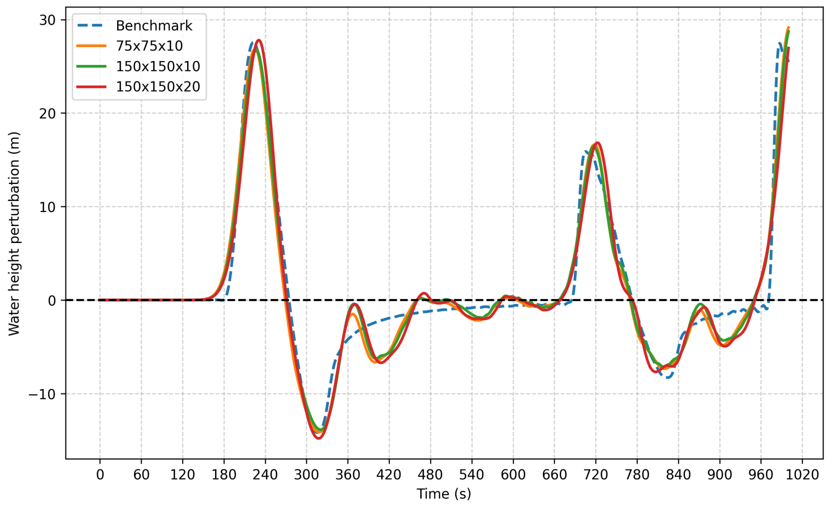


Figure 4.9.: Time evolution of the water depth at Point B with different configurations for the OpenFOAM domain.

The time evolution of the water height at Point B can be seen in Figure 4.9. Overall, there is only a tiny difference between the configurations. The main differences to the benchmark remain essentially unchanged, and it is difficult to determine the best of the three purely visually. For this reason, we use the L_2 norm to quantify the difference between each simulation and the benchmark. The L_2 norm is not ideal as it does not consider time shifting, i.e., a slightly shifted peak might incur a large norm despite the shape being the same. Nonetheless, it is a helpful indicator for comparing the configurations.

OpenFOAM Mesh	Point A	Point B	Point C	Avg.
$75 \times 75 \times 10$	**42.17**	58.62	36.45	45.75
$150 \times 150 \times 10$	42.18	**57.44**	**33.96**	**44.53**
$150 \times 150 \times 20$	54.87	68.17	41.15	54.73

Table 4.1.: L_2 norm of the difference between coupled simulation and benchmark for three different configurations of the OpenFOAM mesh. These experiments use the smaller domain configuration shown in Figure 4.8.

The quantitative evaluation results are in Table 4.1. Overall, the configuration with a $150 \times 150 \times 10$ cell count has the most accurate results. However, the difference to the configuration with $75 \times 75 \times 10$ cells is only very small and likely not significant. This indicates that horizontal cell count in the OpenFOAM domain has no significant influence on the accuracy of the coupling, which is a desirable outcome. The difference between these two configurations and the one with a larger vertical cell count is more significant. With the larger vertical cell count, the accuracy decreases. A possible explanation for this is that at least part of the discrepancy from the benchmark is related to the difference in the governing equations. As the number of vertical cells increases, the NSE might provide results less like the depth-averaged SWE.

Artifacts in the Discontinuous Step Scenario

As discussed in Section 3.2.5, because of the spectral discretization used in SWEET and the Gibbs phenomenon, using discontinuous initial conditions may lead to issues. For this reason, we use a smooth column for our tests. However, during earlier tests, we used a 100 m water column with a radius of 5 km as initial condition. The results of this experiment are presented in Figure 4.10.

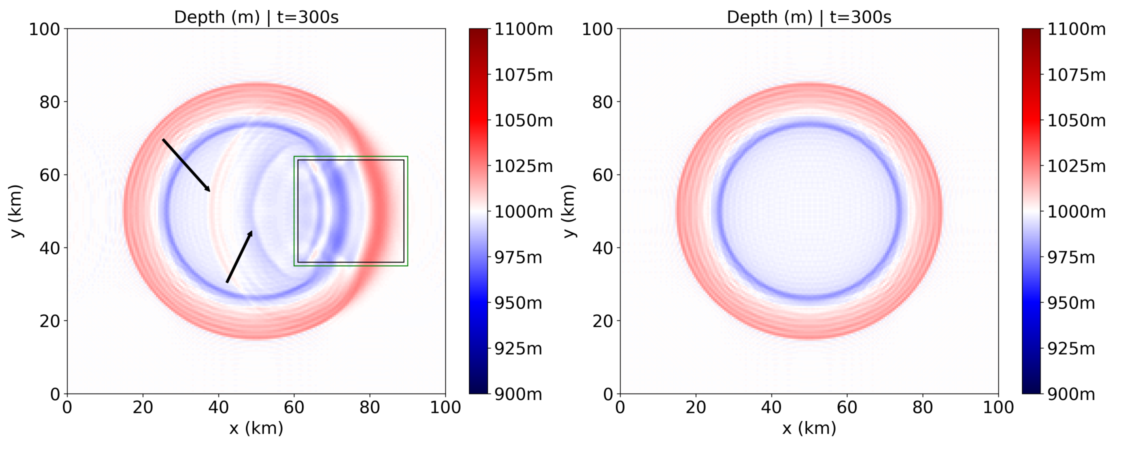
As expected, in Figure 4.10c, there are visible oscillations throughout the simulation that are likely initially caused by the Gibbs phenomenon. They can be best seen at $t = 40\,\text{s}$. The oscillations also appear at Point B and Point C and are of the magnitude of about 1 m. Also visible in Figure 4.10c is an additional peak and valley that is not present in the other simulations in Figure 4.5. The peak and valley can be seen at $t = 180\,\text{s}$ and $t = 280\,\text{s}$, respectively.

The source of this discrepancy can be seen when visually analyzing the results: A small portion of the wave is reflected instead of transferred whenever the wave enters the SWEET domain. This is especially obvious in Figure 4.2c, where the reflected valley of the wave is currently at $x = 50\,\text{km}$. These reflected waves are only small, but at the later timesteps, they cause persistent noise to the overall simulation. Notably, such reflections either do not appear or are too small to notice in Figure 4.2c.

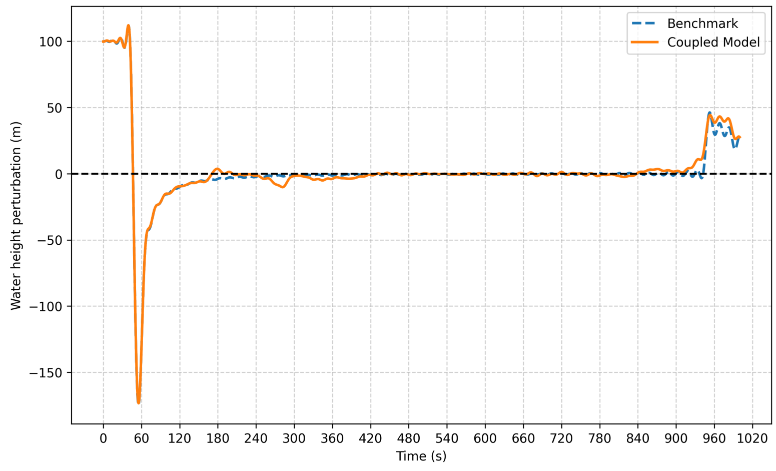
A possible explanation for these small oscillations could lie in the criteria for the SWE. Their accuracy is directly related to if the wavelength is large compared to the depth. [37] With their small oscillations, they could break this criterion and not be simulated correctly, leading to a simulation difference between the two solvers that results in these waves. As the other experiment only contains waves of much higher wavelength, this effect does not appear there.

4.2.2. Sphere2D Domain

In addition to the `Cart2D` domain, we also evaluate the `Sphere2D` domain. Because of the different domains, the solver has some implementation differences, which are



(a) Coupled Model (b) Benchmark

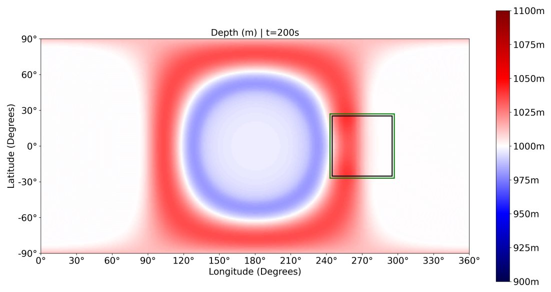


(c) Point A

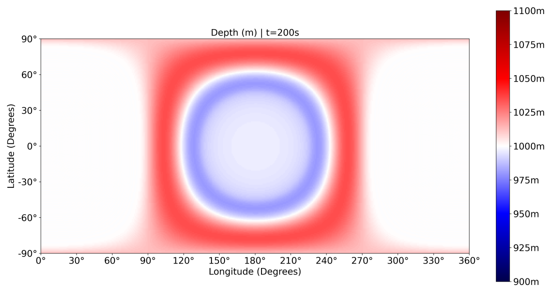
Figure 4.10.: Results of simulations using a discontinuous step as initial water column. Black arrows mark small reflective waves originating from the coupling boundary.

discussed in Section 3.2.4. However, regarding the evaluation, we use the same setup. For the OpenFOAM domain, we use the empty domain at rest, and for the SWEET domain, we use the `Sphere2D` version of the column scenario.

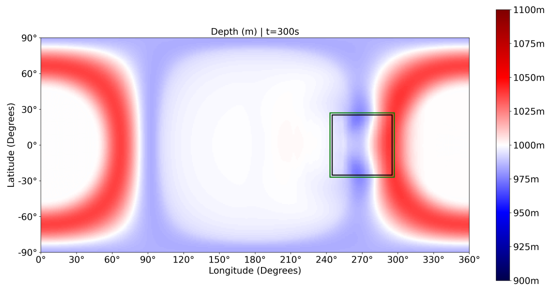
A key difference in the simulation is the size of the OpenFOAM domain, whose side length is halved compared to that in the `Cart2D` evaluation. This is because the equirectangular projection causes problems at higher latitudes. These effects are detailed in Section 4.2.2.



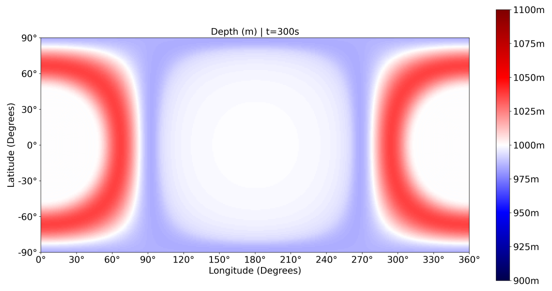
(a) Wave first enters SWEET domain.



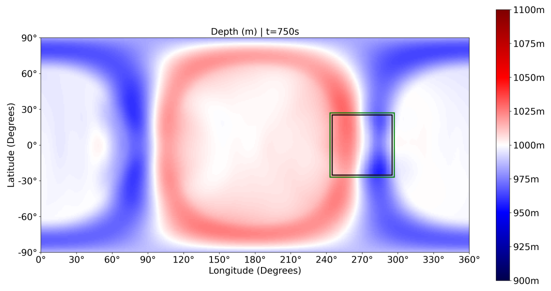
(b) Benchmark $t = 200\,\mathrm{s}$



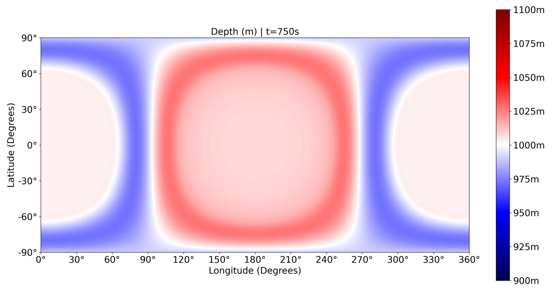
(c) Eastbound wave crosses SWEET domain.



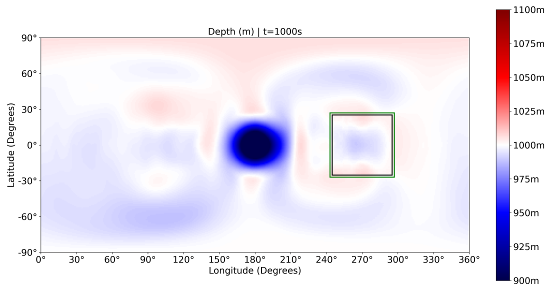
(d) Benchmark $t = 300\,\mathrm{s}$



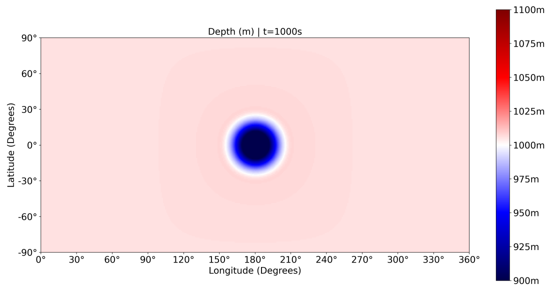
(e) Westbound wave crosses SWEET domain.



(f) Benchmark $t = 750\,\mathrm{s}$



(g) Wave meets again at the center.



(h) Benchmark $t = 1000\,\mathrm{s}$

Figure 4.11.: Comparison of water height in the coupled `Sphere2D` solver (left) and the benchmark (right).

The coupled solver's results for the `Sphere2D` domain can be seen in Figure 4.11. As for the `Cart2D` domain, four key timesteps are shown. When the wave first enters the OpenFOAM domain, the wave crosses the domain in each direction, and the final state at $t = 1000\,\mathrm{s}$. From a purely visual inspection, the results are similar to those with the `Cart2D` domain. The coupling works well, allowing the wave to cross into and out of the coupling area without issues. The same kind of smoothing previously observed in the other domain can also be seen here. There seems to be no obvious reflection of waves at the domain's boundary. In the last timestep in Figure 4.11g, it can be seen that the simulation achieves close to the same state as in the benchmark. There are only very minimal perturbations in the domain, which suggests a well-working coupling.

An additional difference between the benchmark and the coupled solver is a bend in the wave when it should be straight. This is visible in Figure 4.11a and Figure 4.11c. The bent wave is likely due to the distortions introduced by equirectangular projection. We investigate this further in Section 4.2.2.

Numerical Evaluation

To better understand the accuracy of our solver compared to a benchmark run only using SWEET without domain decomposition, we analyze the time evolution of the water height. For this, we use the same method as for the `Cart2D` domain, selecting specific points and comparing their water height, obtaining scalar measures of accuracy using the L_2 norm.

In contrast to the `Cart2D` domain, this domain is spherical and represented by a latitude-longitude grid. Nonetheless, we use the same locations for the evaluation points. Their locations in the `Sphere2D` domain can be seen in Figure 4.12. We place them along the equator. One point each is at the center, inside the coupled region, and after it.

The time evolution of the water depth at these three points can be seen in Figure 4.13. The results are mostly the same as for the `Cart2D` domain. A high overall accuracy is achieved. The peaks and valleys stay the same and happen roughly simultaneously. In Figure 4.13b, some peaks and valleys are significantly lower than in the benchmark. This is likely attributed to the 3D-NSE simulating a different wave shape, with less sharp edges and some small trailing waves. When crossing back into the SWEET domain, the wave mostly readopts the original shape, as evidenced by the matching shape and peak heights in Figure 4.13c. The absence of reflections at the boundary observed visually is also present in the graphs. Overall, the solver achieved very good accuracy.

A benefit of the `Sphere2D` domain is that it contains implementations for more advanced timestepping methods, namely SDC. We also conducted tests using this

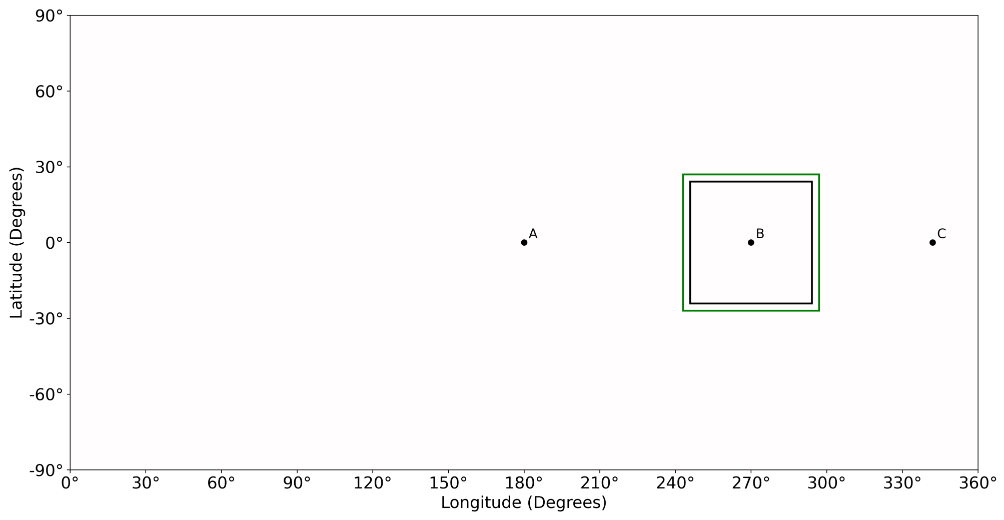


Figure 4.12.: Location of evaluation points in the `Sphere2D` domain. The outer green line indicates the border of the OpenFOAM domain. The inner black line indicates the border of the SWEET domain.

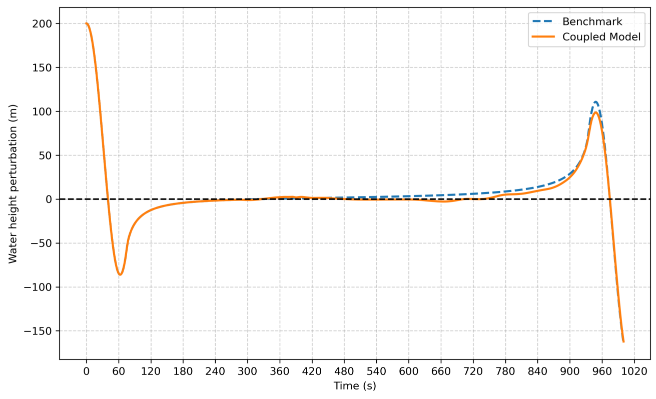
timestepping method. The results were the same as the ones obtained from the other simulations with the ERK method. This shows the possibility of using methods similar to SDC in the future to improve efficiency and/or accuracy. An example would be parallel-in-time simulation.

Projection

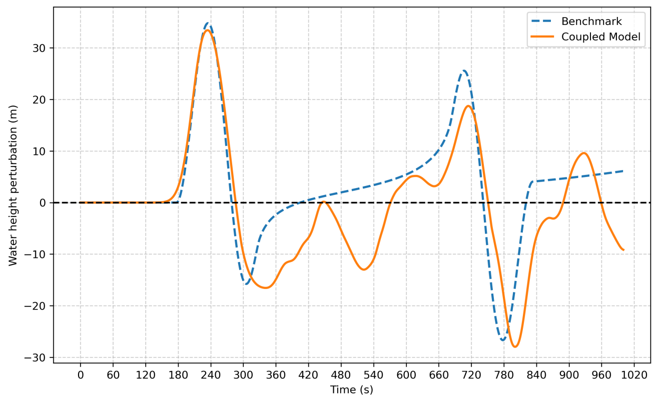
The `Sphere2D` domain is a sphere represented by a latitude-longitude grid. Coupling this domain with a rectangular domain requires some projection. Due to the fundamental problem of mapping a curved surface onto a flat plane, no mapping perfectly preserves both distance and area. The equirectangular projection we chose for our model is easy to implement but has the significant drawback of large distortions at higher latitudes. The further a cell is from the equator, the larger its area in the grid is compared to its actual area.

To examine how this affects the results, we conducted a test where the OpenFOAM domain is moved further away from the equator. We use two configurations, moving the OpenFOAM domain south by half its width for each one.

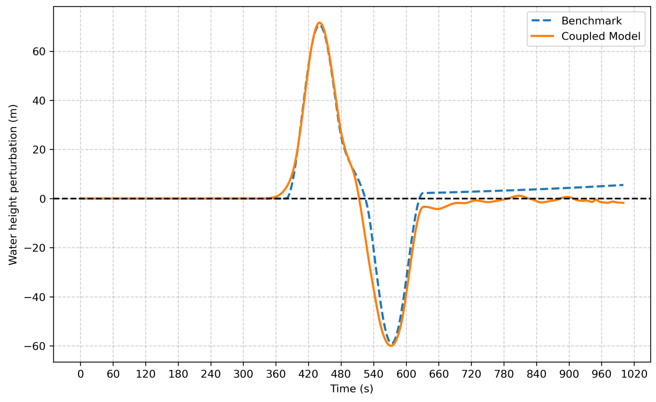
The results of these tests are visualized in Figure 4.14. As can be seen, both cases show significant distortions that decrease the accuracy of the simulation. In the case of Figure 4.14a, a significant disconnect between the OpenFOAM wave and the SWEET



(a) Point A

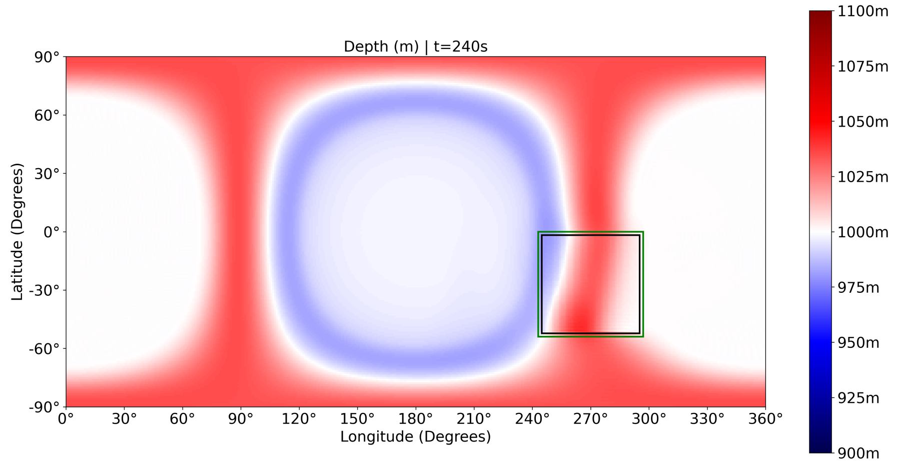


(b) Point B

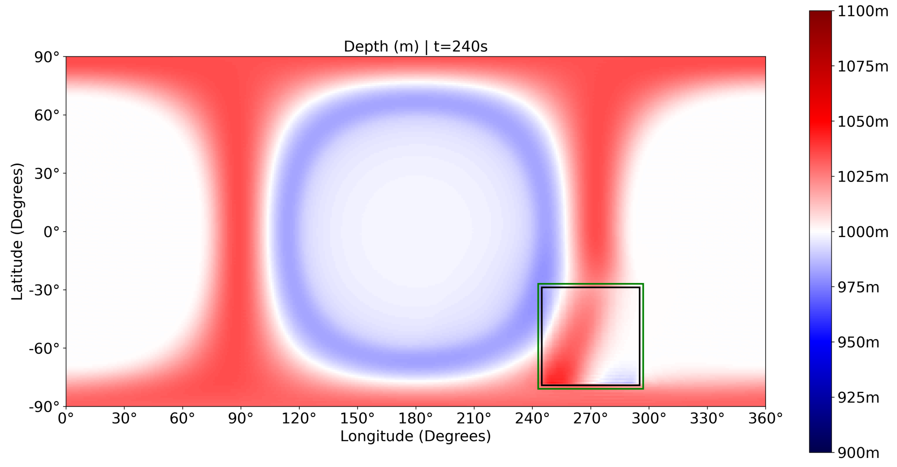


(c) Point C

Figure 4.13.: Evolution of the water level at three different points in the `Sphere2D` domain.



(a) Latitude $-27°$

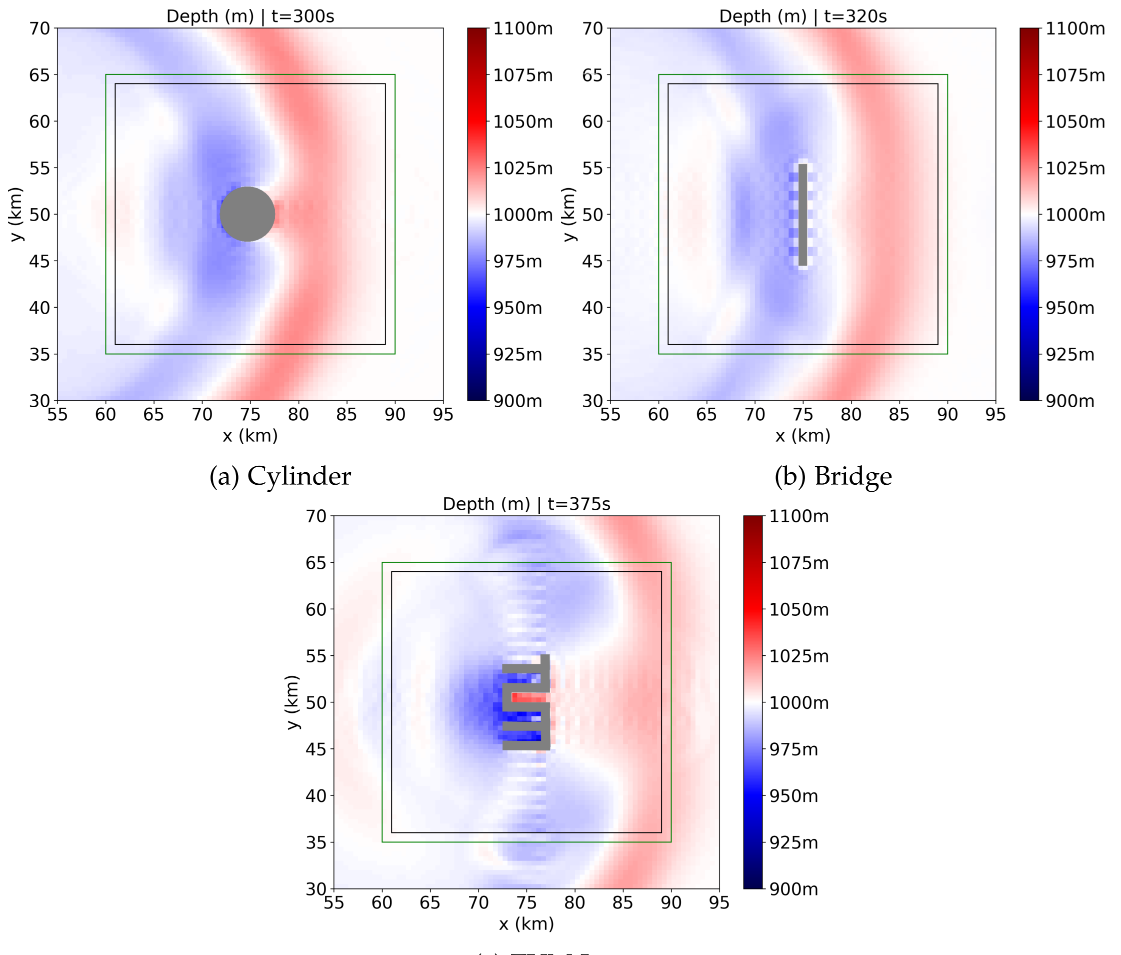


(b) Latitude $-54°$

Figure 4.14.: Simulation inaccuracies caused by equirectangular projection for non-equator domains.

wave can be seen at the lower edge of the domain. This distortion is already dominant even though the domain is only moved 7.5 km south. As shown in Figure 4.14b, the distortion quickly increases as the domain is moved even further. The same distortion effect can also be seen when leaving the OpenFOAM domain at the equator but increasing its size. This is why a smaller domain was chosen for earlier tests in the `Sphere2D` domain. This would also influence the usability of any model using this kind of coupling.

4.2.3. Complex Geometry



(a) Cylinder (b) Bridge

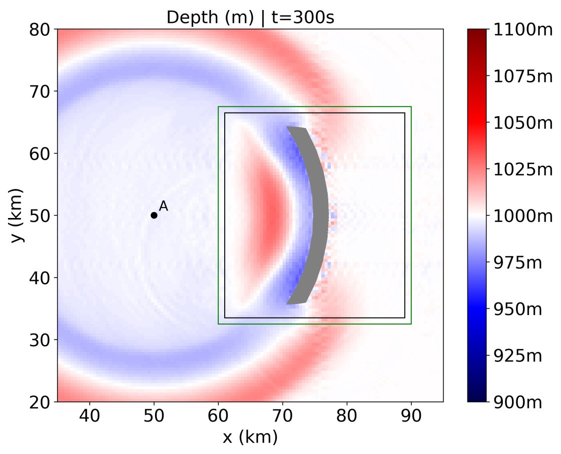
(c) TUM Logo

Figure 4.15.: Snapshots showing the results of simulations using several types of obstacles.

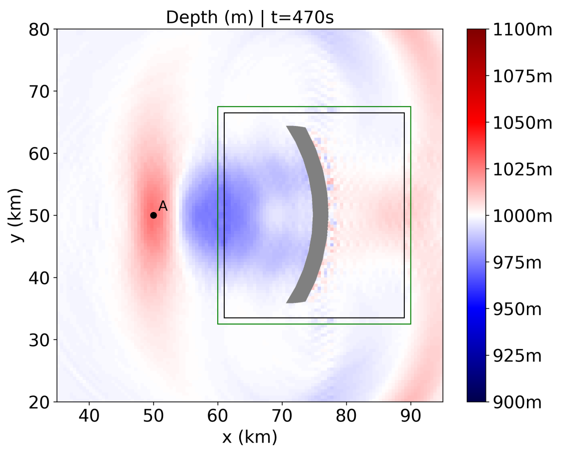
Simulating an empty domain is not the purpose of our coupled solver. The main reason for using coupling is that it can expand the capabilities of SWEET to simulate scenarios with complex geometry. This is impossible in SWEET and is limited in SWE as they only allow for two-dimensional obstacles. To test whether our implementations work and simulations with obstacles are possible and accurate, we conduct several tests with the obstacles introduced in Section 3.2.5. For the following tests, we use the `Cart2D` variant of SWEET, so there is no additional noise introduced into the results because of the projection effects discussed in Section 4.2.2. As the obstacles are simulated on the OpenFOAM side, they can just as well be used with the `Sphere2D` domain.

Some excerpts from the results for three of the obstacles can be seen in Figure 4.15. All obstacle simulations run without problems, converging as expected and with no difference from those with an empty OpenFOAM domain. As seen in the simulation for the cylindrical obstacle in Figure 4.15a, the interaction with the obstacle works correctly. The wave is parted by the cylinder and reconnects after it. These influences are then transferred back into the SWEET domain. As can be seen in the bridge scenario in Figure 4.15b, the simulation can also deal with obstacles that have vertical variations in their structure.

While the results look good intuitively, a better method is needed to evaluate their accuracy objectively. For the empty OpenFOAM domain, we conducted benchmark runs using an uncoupled version of SWEET. As SWEET cannot simulate obstacles, this is impossible for these scenarios. For this reason, we designed an obstacle with a predictable collision behavior to check if the coupled solver behaves as expected. The shape of the obstacle is that of a circle segment with the wave origin at its center. The wave-facing wall is entirely vertical. This way, any part of the waves collides with the obstacle at a right angle, causing it to be reflected directly back. Theoretically, this results in a focal point at the center of the domain.



(a) $t = 300\,\text{s}$



(b) $t = 470\,\text{s}$

Figure 4.16.: Two different timesteps in the simulation of the wave hitting a reflector obstacle in the form of a circle segment. Point A is the center of the domain and the focal point of the obstacle.

The results from the simulation with this reflector obstacle can be seen in Figure 4.16. Figure 4.16a shows the moment the wave impacts the obstacle is reflected, and Figure 4.16b shows the moment the reflected wave reaches the focal point. As the figure shows, the obstacle works as expected, and the wave is focused with a high peak at the focal point. There is some spread of the wave beside the peak, but this is likely the cause of dissipation and not any coupling-related side effect.

4.3. Summary

We conducted tests with both our coupled solvers for several setups. The atmosphere-ocean solver shows promising results, though there are few use cases with the current limitations. For our ocean-ocean solver, we tested both the `Cart2D` and `Sphere2D` domains to validate the implementation. The simulations yielded excellent results, both considering visual examination and numerical evaluation. However, we identify an issue in the `Sphere2D` domain considering the usage of equirectangular projection in the domain decomposition. We show that this leads to inaccuracies for OpenFOAM domains at higher latitudes. Placing obstacles in the OpenFOAM domain, we show that our implementation can effectively expand the capabilities of SWEET. We further validate this claim by designing a reflector obstacle to focus a wave on a focal point in the SWEET domain.

5. Discussion

The results from Chapter 4 show that our coupled solvers perform well in several test cases. This section discusses the implications of these results and their interpretation in terms of our research questions posed in Chapter 1. Section 5.1 starts by answering our primary research question, going over the general meaning of the results regarding the viability of coupling between SWEET and OpenFOAM. Section 5.2 continues with the secondary research question, analyzing the functionality of a coupling between a spherical and flat domain, considering the projection inaccuracies, and introducing possible remedies. Lastly, Section 5.3 goes over possible extensions to our ocean-ocean solver to improve accuracy and make it valuable in realistic use cases.

5.1. Viability of Coupling SWEET to OpenFOAM

First, we answer the main research question: Are there viable options for using SWEET, a SWE solver with spherical discretization, to a 3D configuration of OpenFOAM, a finite volume solver?

Atmosphere-Ocean Solver

Our work presents two solvers, the first being a one-way coupled atmosphere-ocean solver. This solver is severely limited by the absence of heat transport and many other quantities relevant to climate models, such as humidity. Therefore, only a proof-of-concept was provided. Nonetheless, the tests conducted with this solver showed that wind transmission from SWEET to OpenFOAM works well. Theoretically, this implementation could be extended to accurately represent wind stress. Some minor logic similar to that used for the ocean-ocean solver would need to be added to the preCICE adapter for OpenFOAM to handle the correct calculation and conversion from absolute to relative velocity and stress. Time-dependant influences to PDE that don't depend on other variables are called forcing functions. [16] As the SWEET solver only depends on the initial state and has no other input variables, it acts as a forcing function for the OpenFOAM domain in the atmosphere-ocean solver.

However, since this model only allows for one-way coupling, it might have few use cases. The accurate simulation of wind stress in complex wind patterns could improve OpenFOAM results for oceanic simulation. However, considering the absence of two-way interaction, similar results might be achieved by a simpler calculation of wind stress instead of a full-scale solver. Regardless, this proof-of-concept shows the general viability of such a setup, particularly if SWEET should, at any point in the future, be extended with the capabilities for heat transport.

Ocean-Ocean Solver

The ocean-ocean solver shows excellent results. All tested scenarios work as expected, showing only minor discrepancies from the benchmark. There are two main types of differences. The first one is the small reflections of waves at the OpenFOAM-SWEET boundary observed in the discontinuous column scenario. It is possible that the Gibbs phenomenon causes this, as explained in Section 4.2.1. However, if the coupling causes it, a possible cause for this could be the implementation of pressure BCs in OpenFOAM. As SWEET does not contain information about pressure, this is implemented as a zero gradient Neumann BC. A possible solution could be manually recovering the pressure from the SWEET velocity and water height. This should be straightforward because of the hydrostatic pressure assumption from Section 2.1.2. The other discrepancy is the slightly different wave shape in the OpenFOAM domain compared to the SWEET domain. Theoretically, assuming hydrostatic pressure and a large width-to-height ratio, the simulation should yield equivalent results. However, there is no explicit limit to the ratio needed for the SWE. [37] A maximum of 0.05 for the ratio of depth to wavelength is given by [23], but this is smaller than the ratio in our scenario. For good visualization, we choose a domain with a relatively low ratio (100:1), which could be close to violating the hydrostatic pressure assumption and cause slight inaccuracies. This would not be an issue in most actual use cases, as the width would be orders of magnitude larger while the height would be similar.

Tests with obstacles in the OpenFOAM domain show that the solver is well-suited for use in realistic scenarios. The most obvious scenario is the simulation of large bodies of water or the entire ocean system of the Earth by SWEET. OpenFOAM could then simulate islands and continents. This setup is useful for simulating events like tsunamis or similar large waves and their effects on coastlines. Although not used in the tests, nothing theoretically prevents the disjoint OpenFOAM regions. Without many changes, these could be simulated by the same solver or, with some more changes, even be simulated by different solvers, leveraging multiprocessing. Regarding efficiency, there is also a significant possibility for optimization. Much of the runtime is spent computing the 3D to 2D reduction, which essentially iterates all cells. However, in theory, only the values at the sides of the OpenFOAM domain are necessary for the coupling. The runtime could be significantly decreased by calculating the reduction for only a few of the center cells. As this still provides sensible values for the center cells, this does not lead to sharp transitions and the subsequent spectral artifacts.

5.2. Coupling Between a Spherical and Flat Domain

Our secondary research question is whether creating an accurate locally coupled solver between a spherical and flat domain is possible. This research question is particularly interesting because many actual use cases involve spherical domains. We encounter projection-related distortions when conducting tests of our ocean-ocean solver on the sphere. These distortions result from the equirectangular projection used in our coupling, a relatively inaccurate albeit simple projection method. Distortions are practically inevitable as it is a well-known problem that a flat plane cannot be perfectly wrapped around a sphere. Nonetheless, there are ways to mitigate these problems. The first option is the one also chosen in most of the tests: Distortions are minimized by simply keeping the domain small and at the equator. In an actual use case, the OpenFOAM domain is expected to be much smaller in relation to the overall domain size, so this is a viable option. However, moving the domain might not be an option, and with many domains, keeping all of them at the equator can be impossible. Another option is to use different projection methods that do not depend on the location of the domain. Many projection types of varying complexity exist, e.g., Gnomonic Projection, Stereographic Projection, or Lambert Azimuthal Equal-Area Projection. [33] Nonetheless, as there is no perfect projection, they all come with different trade-offs for preserving distances, area, and angles. With momentum and energy correlated to the area and the physical behavior of waves being related to distances and angles, these trade-offs carry over to simulation accuracy. Which projection method is best suited for this solver would need to be the focus of further research and could depend on the specific scenario.

5.3. Outlook: Parallel-in-Time

One of the incentives for using the ocean-ocean solver is that a considerable domain can be simulated using the low-complexity SWE. At the same time, only specific high-interest areas need to be calculated using the computationally expensive 3D-NSE. Nonetheless, if the domain gets large enough, the SWE solver can still be the bottleneck. Because of SWEET's structure, it is possible to use many different timestepping schemes and thus improve runtime. An implementation for SDC and even parallel-in-time method PFASST is already implemented. With the implementation of our solver, using such a method is as simple as changing a configuration parameter.

However, a key point of parallel-in-time schemes such as PFASST is to split large timesteps into small parts and calculate them concurrently. [10] Therefore, there will be no benefit in runtime unless the timestep size can be increased compared to the previous timestepping method. As OpenFOAM largely dictates the timestep because of its fine grid, the solver must thus start using different timesteps for both solvers. This so-called subcycling is possible in explicit coupling by reusing a constant boundary value until the next data exchange. However, this can quickly lead to inaccuracies. [12] Therefore, implicit coupling needs to be enabled to retain good accuracy. To further improve the results, this can be combined with waveform iteration [31], where instead of a static boundary value over the entire coupling timestep, the boundary value is interpolated using the values from the other solver's subcycling timesteps. Because all these features are handled by preCICE, this improvement could be added with minimal implementations, potentially improving efficiency substantially in use cases where SWEET is a runtime bottleneck.

6. Conclusion

This work explores the question of whether there are viable options for coupling SWEET to a 3D configuration of the finite volume solver OpenFOAM.

Using the coupling library preCICE, two distinct coupling approaches are developed: A layered atmosphere-ocean solver (Section 3.1) and a two-way coupled ocean-ocean solver (Section 3.2). We show that while the atmosphere-ocean approach can be successful in principle, it is held back by the limitations of the SWE, particularly the absence of heat transport. Nonetheless, a proof-of-concept setup is presented that shows promising results for inflicting wind stress on the OpenFOAM domain (Section 4.1). On the other hand, we implement a well-functioning ocean-ocean solver leveraging two-way coupling. The solver achieves excellent results in a benchmark scenario, showing only minimal inaccuracies at the coupling boundaries, with no significant consequences to the rest of the simulation (Section 4.2). Additionally, it can simulate any obstacle reaching from a simple cylindrical column to complex geometries (Section 4.2.3).

In a secondary research question, we investigate the possibility of coupling a spherical SWE domain to a flat 3D-NSE domain. The results show that our ocean-ocean solver performs well even under the challenges imposed by inaccurate projection methods (Section 4.2.2).

As an outlook, these results lay the foundation for further improvements to the computational efficiency of our solver. We show how the solver could easily be extended to incorporate advanced timestepping schemes and improvements to the coupling setup to achieve a significant speed-up suitable for more realistic use cases (Section 5.3).

Overall, we provide a viable option for efficiently and accurately simulating global flows with local complex geometry and high-interest areas in SWEET.

Abbreviations

SDC	Spectral Deferred Corrections
SWE	Shallow-Water Equations
NSE	Navier-Stokes Equations
SST	Sea Surface Temperature
CFD	Computational Fluid Dynamics
PDE	Partial Differental Equations
ERK	Explicit Runge-Kutta
BC	Boundary Condition

List of Figures

2.1. 3D visualizations of both SWEET domains. 9
2.2. Example of a latitude-longitude grid used to represent the Sphere2D domain. 10
2.3. Visualization of domain decomposition of a domain formed by overlapping circular and rectangular domains. Representation of the problem originally considered by H. A. Schwarz. 12
2.4. Overview of the functionality provided by preCICE. 14

3.1. preCICE configuration for atmosphere-ocean solver with the exchange of momentum and heat. 18
3.2. preCICE configuration used for atmosphere-ocean solver with only the exchange of momentum. 18
3.3. Two jet streams in the SWEET domain for the atmosphere-ocean scenario. 21
3.4. Domain decomposition of the `Cart2D` domain in the ocean-ocean solver. 23
3.5. Domain decomposition of the `Sphere2D` domain for the ocean-ocean solver. 24
3.6. preCICE configuration for the two-way-coupled ocean-ocean solver. . . 27
3.7. Artifacts appearing in the coupling region when values are written to the boundary instead of the cell in OpenFOAM. 31
3.8. Water height at different timesteps in an uncoupled simulation of the test `Cart2D` test scenario by SWEET. 39
3.9. The four obstacles placed in the OpenFOAM domain during the tests. . 40
3.10. Top-down view of the mesh constructed around the cylinder obstacle. . 41

4.1. Side-view of the results from the atmosphere-ocean solver. 44
4.2. Comparison of water height at an early stage in the coupled `Cart2D` solver (left) and the benchmark (right). 46
4.3. Comparison of water height at a late stage in the coupled `Cart2D` solver (left) and the benchmark (right). 47
4.4. Location of evaluation points in the decomposed `Cart2D` domain. 48
4.5. Graph of the water level at Point A in the `Cart2D` domain. 49
4.6. Graph of the water level at Point B in the `Cart2D` domain. 50
4.7. Graph of the water level at Point C in the `Cart2D` domain. 50

4.8. Excerpt of the simulation with an OpenFOAM domain containing 150 $\times$ 150 $\times$ 20 cells in a smaller area. 51
4.9. Time evolution of the water depth at Point B with different configurations for the OpenFOAM domain. 52
4.10. Results of simulations using a discontinuous step as initial water column. 54
4.11. Comparison of water height in the coupled `Sphere2D` solver (left) and the benchmark (right). 55
4.12. Location of evaluation points in the `Sphere2D` domain. 57
4.13. Evolution of the water level at three different points in the `Sphere2D` domain. 58
4.14. Simulation inaccuracies caused by equirectangular projection for non-equator domains. 59
4.15. Snapshots showing the results of simulations using several types of obstacles. 60
4.16. Two different timesteps in the simulation of the wave hitting a reflector obstacle in the form of a circle segment. 61

List of Tables

3.1. OpenFOAM BCs in the atmosphere-ocean solver. 19
3.2. OpenFOAM BCs in the ocean-ocean solver. 24
3.3. SWEET BCs in the ocean-ocean solver. 25

4.1. L_2 norm of the difference between coupled simulation and benchmark for three different configurations of the OpenFOAM mesh. 52

Bibliography

[1] H. J. Aguerre, P. H. Pedreira, P. J. Orbaiz, and N. M. Nigro. "Validation and Enhancement of a Supermesh Strategy for the CFD Simulation of Four-Stroke Internal Combustion Engines." In: *Fluids* 7.3 (2022). ISSN: 2311-5521. DOI: 10.3390/fluids7030104. URL: https://www.mdpi.com/2311-5521/7/3/104.

[2] V. Artale, S. Calmanti, A. Carillo, A. Dell'Aquila, M. Herrmann, G. Pisacane, P. M. Ruti, G. Sannino, M. V. Struglia, F. Giorgi, X. Bi, J. S. Pal, and S. Rauscher. "An atmosphere-ocean regional climate model for the Mediterranean area: assessment of a present climate simulation." In: *Climate Dynamics* 35.5 (Oct. 2010), pp. 721–740. DOI: 10.1007/s00382-009-0691-8.

[3] J. C. Butcher. "Runge–Kutta Methods." In: *Numerical Methods for Ordinary Differential Equations*. John Wiley & Sons, Ltd, 2016. Chap. 3, pp. 143–331. ISBN: 9781119121534. DOI: https://doi.org/10.1002/9781119121534.ch3.

[4] G. Čaklović, T. Lunet, S. Götschel, and D. Ruprecht. *Improving Efficiency of Parallel Across the Method Spectral Deferred Corrections*. 2024. arXiv: 2403.18641 [math.NA]. URL: https://arxiv.org/abs/2403.18641.

[5] G. Chourdakis, K. Davis, B. Rodenberg, et al. *preCICE v2: A Sustainable and User-Friendly Coupling Library*. Sept. 2021.

[6] G. Chourdakis, D. Schneider, and B. Uekermann. "OpenFOAM-preCICE: Coupling OpenFOAM with External Solvers for Multi-Physics Simulations." In: *OpenFOAM® Journal* 3 (Feb. 2023), pp. 1–25. DOI: 10.51560/ofj.v3.88. URL: https://journal.openfoam.com/index.php/ofj/article/view/88.

[7] G. Chourdakis and B. Uekermann. "Towards geometric multi-scale coupling in preCICE." In: *ECCOMAS Young Investigators Congress 2021*. Accessed: 2025-01-12. 2021. URL: https://mediatum.ub.tum.de/doc/1616787/1616787.pdf.

[8] B. Cushman-Roisin and J.-M. Beckers. "Introduction to Geophysical Fluid Dynamics." In: ed. by B. Cushman-Roisin and J.-M. Beckers. Vol. 101. International Geophysics. Academic Press, 2011. DOI: https://doi.org/10.1016/B978-0-12-088759-0.00001-8.

[9] A. Dutt, L. Greengard, and V. Rokhlin. "Spectral Deferred Correction Methods for Ordinary Differential Equations." In: *BIT* 40 (June 2000), pp. 241–266. DOI: https://doi.org/10.1023/A:1022338906936.

[10] M. Emmett and M. Minion. "Toward an efficient parallel in time method for partial differential equations." In: *Communications in Applied Mathematics and Computational Science* 7.1 (2012), pp. 105–132. DOI: 10.2140/camcos.2012.7.105. URL: https://doi.org/10.2140/camcos.2012.7.105.

[11] K. Eriksson, D. Estep, P. Hansbo, and C. Johnson. *Computational Differential Equations*. June 1997.

[12] J. H. Ferziger, M. Perić, and R. L. Street. *Computational Methods for Fluid Dynamics*. 2020.

[13] X. Fu and B. Wang. "Differences of Boreal Summer Intraseasonal Oscillations Simulated in an Atmosphere–Ocean Coupled Model and an Atmosphere-Only Model." In: *Journal of Climate* 17.6 (2004), pp. 1263–1271. DOI: 10.1175/1520-0442(2004)017<1263:DOBSIO>2.0.CO;2. URL: https://journals.ametsoc.org/view/journals/clim/17/6/1520-0442_2004_017_1263_dobsio_2.0.co_2.xml.

[14] K. Gade. "A Non-singular Horizontal Position Representation." In: *Journal of Navigation* 63.3 (2010), pp. 395–417. DOI: 10.1017/S0373463309990415.

[15] M. Griebel, T. Dornseifer, and T. Neunhoeffer. *Numerical Simulation in Fluid Dynamics*. Society for Industrial and Applied Mathematics, 1998. DOI: 10.1137/1.9780898719703. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9780898719703. URL: https://epubs.siam.org/doi/abs/10.1137/1.9780898719703.

[16] R. Haberman. *Applied Partial Differential Equations with Fourier Series and Boundary Value Problems*. 4th ed. Pearson, 2005.

[17] K. Hartung, G. Svensson, H. Struthers, A.-L. Deppenmeier, and W. Hazeleger. "An EC-Earth coupled atmosphere–ocean single-column model (AOSCM.v1_EC-Earth3) for studying coupled marine and polar processes." In: *Geoscientific Model Development* 11 (Oct. 2018), pp. 4117–4137. DOI: 10.5194/gmd-11-4117-2018.

[18] M. Hirschhorn, V. Tchantchaleishvili, R. Stevens, J. Rossano, and A. Throckmorton. "Fluid–structure interaction modeling in cardiovascular medicine – A systematic review 2017–2019." In: *Medical Engineering I& Physics* 78 (2020), pp. 1–13. ISSN: 1350-4533. DOI: https://doi.org/10.1016/j.medengphy.2020.01.008. URL: https://www.sciencedirect.com/science/article/pii/S1350453320300199.

[19] C. Hirt and B. Nichols. "Volume of fluid (VOF) method for the dynamics of free boundaries." In: *Journal of Computational Physics* 39.1 (1981), pp. 201–225. ISSN: 0021-9991. DOI: `https://doi.org/10.1016/0021-9991(81)90145-5`. URL: `https://www.sciencedirect.com/science/article/pii/0021999181901455`.

[20] R. K. S. Joseph Galewsky and L. M. Polvani. "An initial-value problem for testing numerical models of the global shallow-water equations." In: *Tellus A: Dynamic Meteorology and Oceanography* 56.5 (2004), pp. 429–440. DOI: `10.3402/tellusa.v56i5.14436`. eprint: `https://doi.org/10.3402/tellusa.v56i5.14436`. URL: `https://doi.org/10.3402/tellusa.v56i5.14436`.

[21] D. E. Keyes, L. C. McInnes, C. Woodward, et al. "Multiphysics simulations: Challenges and opportunities." In: *The International Journal of High Performance Computing Applications* 27.1 (2013), pp. 4–83. DOI: `10.1177/1094342012468181`. eprint: `https://doi.org/10.1177/1094342012468181`.

[22] P. Knupp. *Remarks on Mesh Quality*. Tech. rep. Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), 2007.

[23] B. Le Méhauté. *Linear Small Amplitude Wave Theories*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1976, pp. 212–238. ISBN: 978-3-642-85567-2. DOI: `10.1007/978-3-642-85567-2_16`. URL: `https://doi.org/10.1007/978-3-642-85567-2_16`.

[24] O. Marti, S. Nguyen, P. Braconnot, S. Valcke, F. Lemarié, and E. Blayo. "A Schwarz iterative method to evaluate ocean–atmosphere coupling schemes: implementation and diagnostics in IPSL-CM6-SW-VLR." In: *Geoscientific Model Development* 14.5 (2021), pp. 2959–2975. DOI: `10.5194/gmd-14-2959-2021`. URL: `https://gmd.copernicus.org/articles/14/2959/2021/`.

[25] T. Mathew. *Domain Decomposition Methods for the Numerical Solution of Partial Differential Equations*. Vol. 61. Springer Berlin, Heidelberg, Jan. 2008. ISBN: 978-3-540-77205-7. DOI: `10.1007/978-3-540-77209-5`.

[26] F. Mintgen and M. Manhart. "A bi-directional coupling of 2D shallow water and 3D Reynolds-averaged Navier–Stokes models." In: *Journal of Hydraulic Research* 56.6 (2018), pp. 771–785. DOI: `10.1080/00221686.2017.1419989`.

[27] *OpenFOAM*. URL: `https://www.openfoam.com/`.

[28] J. B. Palter. "The Role of the Gulf Stream in European Climate." In: *Annual Review of Marine Science* 7.Volume 7, 2015 (2015), pp. 113–137. ISSN: 1941-0611. DOI: `https://doi.org/10.1146/annurev-marine-010814-015656`. URL: `https://www.annualreviews.org/content/journals/10.1146/annurev-marine-010814-015656`.

[29] F. Pelaez. "A flexible approach to 2D-3D coupling of a Shallow-Water Equation solver to OpenFOAM." MA thesis. Technical University of Munich, 2020.

[30] R. A. Pielke Jr and R. A. Pielke Sr. *Hurricanes: Their nature and impacts on society*. Wiley, 1997.

[31] B. Rüth, B. Uekermann, M. Mehl, P. Birken, A. Monge, and H.-J. Bungartz. "Quasi-Newton waveform iteration for partitioned surface-coupled multiphysics applications." In: *International Journal for Numerical Methods in Engineering* 122.19 (2021), pp. 5236–5257. DOI: `https://doi.org/10.1002/nme.6443`. URL: `https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.6443`.

[32] N. Schaeffer. "Efficient spherical harmonic transforms aimed at pseudospectral numerical simulations." In: *Geochemistry, Geophysics, Geosystems* 14.3 (2013), pp. 751–758. DOI: `https://doi.org/10.1002/ggge.20071`. URL: `https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/ggge.20071`.

[33] J. P. Snyder. "Map projections: A working manual." In: (1987). DOI: `10.3133/pp1395`.

[34] *SWEET - Shallow Water Equation Environment for Tests*. URL: `https://gitlab.inria.fr/sweet/sweet`.

[35] A. Toselli and O. Widlund. *Domain Decomposition Methods - Algorithms and Theory*. Vol. 34. Springer, Jan. 2005. ISBN: 978-3-540-20696-5. DOI: `10.1007/b137868`.

[36] A. Vretblad. "Fourier series." In: *Fourier Analysis and Its Applications*. Ed. by S. Axler, F. W. Gehring, and K. A. Ribet. New York, NY: Springer New York, 2003, pp. 73–103. ISBN: 978-0-387-21723-9. DOI: `10.1007/0-387-21723-1_4`. URL: `https://doi.org/10.1007/0-387-21723-1_4`.

[37] C. Vreugdenhil. "Numerical Methods for Shallow-Water Flow." In: vol. 13. Water Science and Technology Library. 1994. DOI: `https://doi.org/10.1007/978-94-015-8354-1`.

[38] A.-T. Vuong, L. Yoshihara, and W. Wall. "A general approach for modeling interacting flow through porous media under finite deformations." In: *Computer Methods in Applied Mechanics and Engineering* 283 (2015), pp. 1240–1259. ISSN: 0045-7825. DOI: `https://doi.org/10.1016/j.cma.2014.08.018`. URL: `https://www.sciencedirect.com/science/article/pii/S0045782514002886`.

[39] M. Zhao, J.-C. Golaz, I. M. Held, et al. "The GFDL Global Atmosphere and Land Model AM4.0/LM4.0: 1. Simulation Characteristics With Prescribed SSTs." In: *Journal of Advances in Modeling Earth Systems* 10.3 (2018), pp. 691–734. DOI: `https://doi.org/10.1002/2017MS001208`. URL: `https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2017MS001208`.

A. preCICE Configurations

This section contains the detailed preCICE configuration files used for the atmosphere_ocean solver (Listing A.1) and the ocean-ocean solver (Listing A.2).

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<precice-configuration>
    <data:vector name="Velocity"/>

    <mesh name="SeaSurface" dimensions="2">
        <use-data name="Velocity"/>
    </mesh>

    <mesh name="AtmosphereSurface" dimensions="2">
        <use-data name="Velocity"/>
    </mesh>

    <participant name="SWEET">
        <provide-mesh name="AtmosphereSurface"/>
        <write-data name="Velocity" mesh="AtmosphereSurface"/>
    </participant>

    <participant name="Ocean">
        <provide-mesh name="SeaSurface"/>
        <read-data name="Velocity" mesh="SeaSurface"/>
        <receive-mesh name="AtmosphereSurface" from="SWEET"/>
        <mapping:nearest-neighbor direction="read" from="AtmosphereSurface
            " to="SeaSurface" constraint="consistent"/>
    </participant>

    <m2n:sockets acceptor="Ocean" connector="SWEET"/>

    <coupling-scheme:serial-explicit>
        <participants first="SWEET" second="Ocean"/>
        <max-time value="100"/>
        <time-window-size value="0.3"/>
        <exchange data="Velocity" mesh="AtmosphereSurface" from="SWEET" to
            ="Ocean"/>
    </coupling-scheme:serial-explicit>
</precice-configuration>
```

Listing A.1: preCICE configuration for the atmosphere-solver.

In the configuration file for the ocean-ocean solver in Listing A.2, the water height values are named "Alpha". This is caused by a specific part in the OpenFOAM adapter that expects certain names for the fields that interact with the OpenFOAM alpha field. However, as the conversion between water height and alpha takes place in the adapter, the value sent over preCICE under the name "Alpha" are actually the corresponding water heights.

```
<?xml version="1.0" encoding="UTF-8" ?>
<precice-configuration>
    <data:vector name="Velocity_SWE_FOAM"/>
    <data:vector name="Velocity_FOAM_SWE"/>
    <data:scalar name="Alpha_FOAM_SWE"/>
    <data:scalar name="Alpha_SWE_FOAM"/>

    <mesh name="SWE_Sides" dimensions="3">
        <use-data name="Velocity_SWE_FOAM"/>
        <use-data name="Alpha_SWE_FOAM"/>
    </mesh>

    <mesh name="FOAM_Sides" dimensions="3">
        <use-data name="Velocity_SWE_FOAM"/>
        <use-data name="Alpha_SWE_FOAM"/>
    </mesh>

    <mesh name="SWE_Top" dimensions="3">
        <use-data name="Velocity_FOAM_SWE"/>
        <use-data name="Alpha_FOAM_SWE"/>
    </mesh>
```

```
<mesh name="FOAM_Top" dimensions="3">
    <use-data name="Velocity_FOAM_SWE"/>
    <use-data name="Alpha_FOAM_SWE"/>
</mesh>

<participant name="SWEET">
    <provide-mesh name="SWE_Sides"/>
    <provide-mesh name="SWE_Top"/>

    <read-data name="Velocity_FOAM_SWE" mesh="SWE_Top"/>
    <read-data name="Alpha_FOAM_SWE" mesh="SWE_Top"/>
    <write-data name="Velocity_SWE_FOAM" mesh="SWE_Sides"/>
    <write-data name="Alpha_SWE_FOAM" mesh="SWE_Sides"/>
</participant>

<participant name="OpenFOAM">
    <provide-mesh name="FOAM_Sides"/>
    <receive-mesh name="SWE_Sides" from="SWEET"/>
    <provide-mesh name="FOAM_Top"/>
    <receive-mesh name="SWE_Top" from="SWEET"/>

    <read-data name="Velocity_SWE_FOAM" mesh="FOAM_Sides"/>
    <read-data name="Alpha_SWE_FOAM" mesh="FOAM_Sides"/>
    <write-data name="Velocity_FOAM_SWE" mesh="FOAM_Top"/>
    <write-data name="Alpha_FOAM_SWE" mesh="FOAM_Top"/>

    <mapping:nearest-neighbor direction="read" from="SWE_Sides" to="
        FOAM_Sides" constraint="consistent"/>
    <mapping:nearest-neighbor direction="write" from="FOAM_Top" to="
        SWE_Top" constraint="consistent"/>
</participant>

<m2n:sockets acceptor="OpenFOAM" connector="SWEET"
            exchange-directory="/home/armin/Documents/
                sweet_precice_coupling/benchmarks_plane/coupling"/>

<coupling-scheme:serial-explicit>
    <participants first="SWEET" second="OpenFOAM"/>
    <max-time value="1000"/>
```

```
        <time-window-size value="0.5"/>
        <exchange data="Velocity_SWE_FOAM" mesh="SWE_Sides" from="SWEET"
            to="OpenFOAM"/>
        <exchange data="Alpha_SWE_FOAM" mesh="SWE_Sides" from="SWEET" to="
            OpenFOAM" initialize="yes"/>
        <exchange data="Velocity_FOAM_SWE" mesh="SWE_Top" from="OpenFOAM"
            to="SWEET"/>
        <exchange data="Alpha_FOAM_SWE" mesh="SWE_Top" from="OpenFOAM" to=
            "SWEET" initialize="yes"/>
    </coupling-scheme:serial-explicit>
</precice-configuration>
```

Listing A.2: preCICE configuration for the ocean-ocean solver.