X International Conference on Computational Methods for Coupled Problems in Science and Engineering
COUPLED PROBLEMS 2023
M. Papadrakakis, B. Schrefler and E. Oñate (Eds)

PARTITIONED FLOW SIMULATIONS WITH PRECICE AND OPENFOAM

Markus Mühlhäußer[1,*], Gerasimos Chourdakis[1], and Benjamin Uekermann[2]

[1] School of Computation, Information and Technology
Technical University of Munich, Germany
e-mail: markus.muehlhaeusser@tum.de

[2] Institute for Parallel and Distributed Systems
University of Stuttgart, Germany

Key words: preCICE, OpenFOAM, coupling, flow partitioning

Abstract. Large and heterogeneous flow simulations could benefit from partitioning, where each subdomain is solved by an individual, dedicated code. In this paper, we investigate and validate the fluid-fluid module of the preCICE-OpenFOAM adapter by coupling two incompressible OpenFOAM fluid solvers via a surface Dirichlet-Neumann coupling approach. The results of various simple test cases are compared to monolithic OpenFOAM simulations. By utilizing a special pressure boundary condition, the coupled results show only a small error around the coupling interface. The magnitude of this error depends on the velocity gradient, the mesh width, as well as the cell orthogonality. Looking into the OpenFOAM source code reveals that higher accuracy is only possible by manipulating the solvers themselves and thereby violating the black-box approach of preCICE. The fluid-fluid module is extended to couple temperature with sufficient accuracy for one validation case. Furthermore, support for multiphase flow solvers is added to the adapter. Lastly, custom inlet-outlet boundary conditions are implemented for pressure and velocity, which switch their behavior dynamically, depending on the flow direction. The results shown in this presentation provide a basis for future fluid-fluid coupling applications with preCICE.

1 INTRODUCTION

Simulation of fluid flow is an extremely complex and computationally intensive task. The complete Navier-Stokes equations comprise multiple flow phenomena such as convection, diffusion, turbulence, or buoyancy. Numerous methods and discretization schemes exist to numerically solve these partial differential equations. A user planning a fluid simulation has the choice from a wide range of available open-source and commercial solvers. These include general solvers employing the finite volume or finite element method, specialized solvers for shallow water equations, Lattice-Boltzmann solvers , or even solvers of different dimensionality. Selecting the correct solver for a large and heterogeneous flow domain can be a difficult task. In a monolithic simulation approach, one must either invest extensive resources developing a new solver specific to the problem, or find a single solver that provides the best compromise for accuracy and simulation time over all regions of the domain.

Alternatively, a coupled approach can be chosen, which is already used for multiphysics problems such as fluid-structure interactions or conjugate heat transfer. This approach allows for choosing specific fluid solvers for different regions of the domain based on the required resolution and accuracy. For instance, it might be sufficient to use a simplified one-dimensional solver for a lengthy pipe segment, whereas a sophisticated three-dimensional solver is required around the main regions of interest. However, besides the advantages of flexibility and reusability of existing solvers, coupled simulations present the challenge of finding a numerically stable coupling scheme.

The objective of this paper is to demonstrate the capabilities and limitations of coupling OpenFOAM solvers with preCICE, and to validate the obtained results. For this, we will couple identical or similar OpenFOAM solvers and compare the results to the corresponding monolithic OpenFOAM simulation. In contrast to commonly employed domain decomposition approaches, we keep the coupled solvers separate from one another, meaning that they cannot interact with each other directly.

This paper presents the most important findings of the master thesis from Mühlhäußer [3] with some additional numerical examples. In Section 2, we first look at possible approaches to fluid-fluid coupling before briefly introducing the technical aspects of the preCICE-OpenFOAM adapter in Section 3. In Section 4, we summarize the theoretical findings from [3] with the existing adapter and Section 5 demonstrates the additional features that were implemented in the context of this work. Lastly, Section 6 showcases the results for various test scenarios.

2 COUPLING OF FLOW SIMULATIONS

There exist a few approaches when it comes to coupling two physics solvers [5]. From a numerical perspective, Dirichlet-Dirichlet coupling with an overlapping domain is usually the easiest. After the first solver (solver one) solves the current time step on its domain, values of exchanged quantities are taken from within the overlapping region and set as boundary conditions for the second solver (solver two), and vice versa. Such an algorithm is also known as a Schwarz iteration and was, for example, used by Santasmasas to couple OpenFOAM with a Lattice Boltzmann solver for wind simulations [6].

An alternative approach is Dirichlet-Neumann coupling, where not only the values of quantities of interest are exchanged but also its gradients in the opposite direction. Dirichlet-Neumann coupling does not require an overlapping region. Instead, the values are exchanged over a common interface. If solver one has a Neumann boundary condition at the interface, the values of the variable are taken from solver one and set as Dirichlet boundary values for solver two. After solving the second domain, the gradients are taken from solver two and set as the Neumann condition for solver one. This process is repeated until convergence has been reached. Grinberg et al. [7] used a Dirichlet-Neumann coupling approach to decompose arterial flow simulations.

A third method is Robin-Robin coupling, which is again done over a common interface, where both solvers use Robin boundary conditions, a combination of Dirichlet and Neumann conditions. This approach was successfully implemented by Fernandez et al. [8], who coupled finite element solvers using Nitsche's interface method.

One core idea of the preCICE library is that it should be minimally invasive into the coupled solvers. As such, coupling methods across a common interface are preferred, as we only need access to the domain boundaries. Even though, Robin-Robin coupling promises greater stability than Dirichlet-Neumann coupling, implementing such a scheme is rather complex. Therefore, we use Dirichlet-Neumann coupling in our presented work, which translates naturally into the Finite Volume code of OpenFOAM, where we use Dirichlet or Neumann boundary conditions for velocity and pressure at domain inlets and outlets.

Coupling methods can further be divided into explicit and implicit coupling schemes. In explicit schemes, each participant solves its system exactly once per time step. This is desirable from a performance perspective. However, this often leads to instabilities around the coupling interface for strongly coupled problems. Therefore, an implicit coupling scheme is necessary, where both participants repeatedly solve their systems for the same time step until the values at the coupling interface are converged. Depending on the convergence criteria, several iterations might be needed per time step. preCICE provides acceleration methods for implicit coupling schemes (such as quasi-Newton methods), which significantly improve the convergence rate.

3 THE OPENFOAM-PRECICE ADAPTER

The fluid solvers used in this work are all part of OpenFOAM, in particular, version 2112 maintained by ESI-OpenCFD[1]. The solvers we looked into all use the finite volume Method for discretization and a PIMPLE solution algorithm.

Coupling the OpenFOAM fluid solvers is done with preCICE [1], an open-source library for coupling partitioned multiphysics simulations. It follows a minimally invasive approach by considering participating single-physics solvers as black boxes. It handles communication and data mapping and provides numerical schemes for explicit, implicit, parallel, or serial coupling. Together with the core library come dedicated adapters for different solvers, which, in the case of surface coupling, adjust the boundary conditions at the interface.

The OpenFOAM adapter [4] is implemented as an OpenFOAM function object and, as such, can be loaded at runtime without any modifications to the solvers' source code. Figure 1 shows the execution flow of a PIMPLE solver with the enabled adapter. It is important to note that the calls to OpenFOAM function objects are limited to once per time step as well as in the initialization and the end phase of the simulation.

The OpenFOAM adapter is divided into multiple "modules", which provide the functionality for coupling the quantities of interest for the respective physical problem. The fluid-fluid module used in this work allows coupling of pressure and velocity as well as pressure gradient and velocity gradient. For every physical variable, the adapter provides methods to write or read the physical values obtained from OpenFOAM boundary patches to or from preCICE buffers.

The OpenFOAM adapter stores references to the fields of physical values and uses methods from the OpenFOAM framework to retrieve values and face-normal gradients from the boundary regardless of the underlying boundary condition. However, overwriting the boundaries with values from preCICE is specific to the OpenFOAM patch, as can be seen in the implementation for reading the pressure gradient:

```
scalarField& gradientPatch = refCast<fixedGradientFvPatchScalarField>(
                                p_->boundaryFieldRef()[patchID]).gradient();
forAll(gradientPatch, i){
    gradientPatch[i] = buffer[bufferIndex++];
}
```

[1] OpenFOAM v2112: https://develop.openfoam.com/Development/openfoam/-/releases/OpenFOAM-v2112

It is therefore required from the user to set `fixedValue` or `fixedGradient` boundary conditions where the respective values are supposed to be read from preCICE.

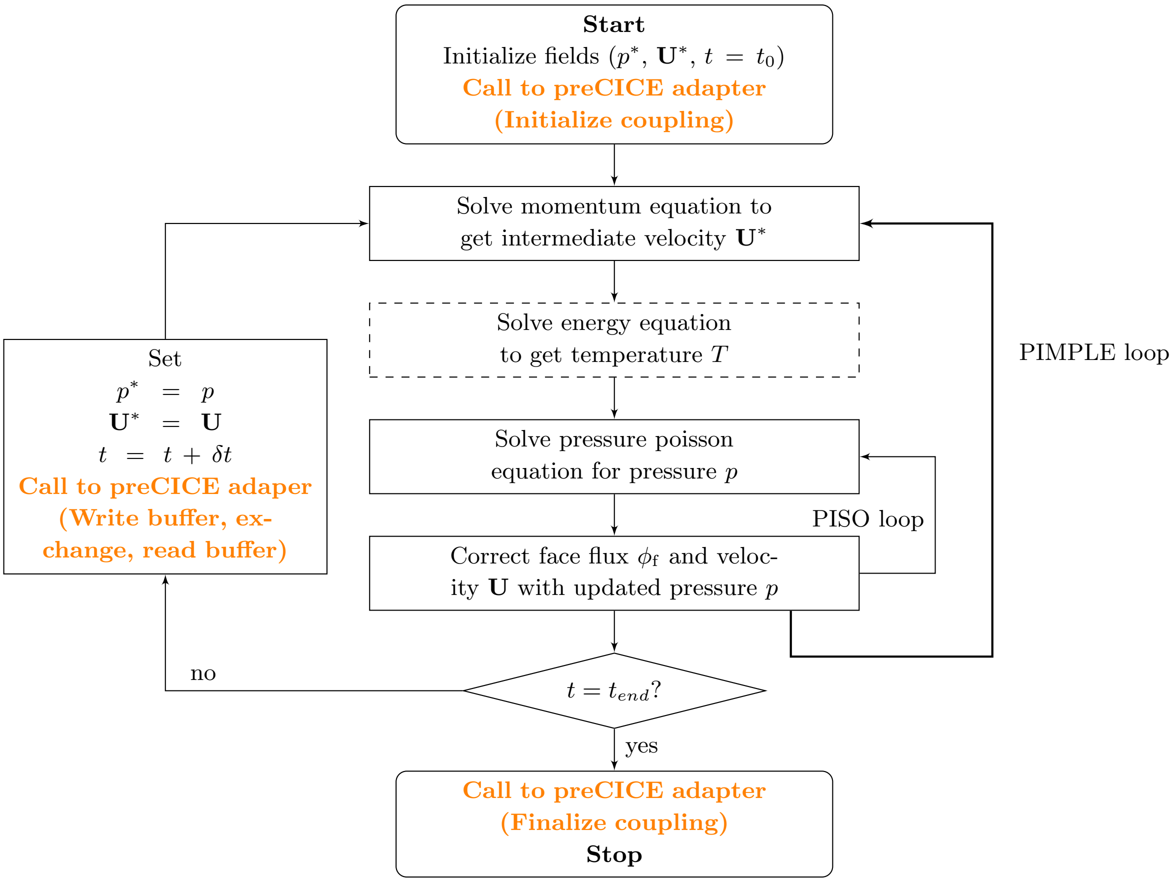

Start
Initialize fields (p^*, $\mathbf{U}^*$, $t = t_0$)
Call to preCICE adapter
(Initialize coupling)
Solve momentum equation to get intermediate velocity $\mathbf{U}^*$
Solve energy equation to get temperature T
Solve pressure poisson equation for pressure p
Correct face flux ϕ_f and velocity $\mathbf{U}$ with updated pressure p
PIMPLE loop
PISO loop
Set
$p^* = p$
$\mathbf{U}^* = \mathbf{U}$
$t = t + \delta t$
Call to preCICE adaper
(Write buffer, exchange, read buffer)
no
$t = t_{end}$?
yes
Call to preCICE adapter
(Finalize coupling)
Stop

Figure 1: The OpenFOAM PIMPLE algorithm, including calls to the preCICE adapter.

4 INSIGHTS ON FLUID-FLUID COUPLING WITH OPENFOAM

Previous work [2] already showed a validation example for partitioned flows with the OpenFOAM adapter, simulating laminar flow through a straight, three-dimensional pipe. While the general coupling worked, the velocity and pressure graphs across the coupling interface showed either oscillations or a jump, depending on the combination of the coupled quantities. The error was suspected to stem from the calculations of the gradients. Further investigation in the context of this work shows, however, that the error originates from the OpenFOAM internal implementation of the boundary values.

To understand the underlying issue, we have to look into how the underlying equations are solved in OpenFOAM. In a laminar, incompressible flow setting, the continuity and momentum equations read:

$$\frac{\partial}{\partial t}\rho = -\nabla \cdot (\rho \mathbf{U}) \tag{1}$$

$$\frac{\partial}{\partial t}\mathbf{U} = -\nabla \cdot (\mathbf{U} \otimes \mathbf{U}) + \nu \nabla^2 \mathbf{U} - \nabla p + \mathbf{g} \tag{2}$$

where $\mathbf{U}$ is the velocity, p the pressure and ν the kinematic viscosity. The linearized momentum equation can be written in matrix formulation as:

$$\mathbf{MU} - b = -\nabla p \tag{3}$$

For the PIMPLE algorithm, the equation is written as:

$$\mathbf{AU} - \mathbf{H} = -\nabla p \tag{4}$$

where the matrix $\mathbf{M}$ is split up into a diagonal matrix $\mathbf{A}$ and a matrix $\mathbf{H}$, containing the off-diagonal values of $\mathbf{M}$, as well as the source vector b. The matrix $\mathbf{H} = \mathbf{H}(\mathbf{U}_{old})$ is calculated explicitly from the velocity values of the previous time step. Solving Equation (4) for velocity results in the following representation:

$$\mathbf{U} = \mathbf{A^{-1}H} - \mathbf{A^{-1}}\nabla p \tag{5}$$

For an efficient implementation of the equation above, OpenFOAM creates new temporary fields for $A^{-1}H$ and A^{-1}, called `HbyA` and `rAU`, respectively. These fields also have boundary conditions, which are not explicitly given by the user, but are instead derived from the other fields, such as pressure and velocity. For patches that use a fixed velocity, OpenFOAM adjusts the boundary values for the field `HbyA` as:[2]

$$\mathbf{A^{-1}H} = \mathbf{U} \tag{6}$$

This assures that Equation (5) is fulfilled at patches that model inlets or walls, where a `zeroGradient` boundary condition for pressure is expected to go together with a fixed velocity. However, it cannot be assumed that the pressure gradient will be zero at the interface of a coupled simulation. OpenFOAM provides a boundary condition for pressure that prevents the wrong assignment of the `HbyA` boundary field: Instead of a `fixedGradient` boundary condition, the `fixedFluxExtrapolatedPressure` boundary condition can be used. Here, the boundary field of `HbyA` is not explicitly assigned, but instead, the pressure gradient at the boundary is calculated to

$$\nabla p_b = \mathbf{A_b}\left(\left(\mathbf{A^{-1}H}\right)_b - \mathbf{U}_b\right) \tag{7}$$

such that the momentum Equation (5) is fulfilled. The subscript b indicates that these terms refer only to the boundary field values. Therefore, we can expect better results at our interface using the `fixedFluxExtrapolatedPressure` boundary condition.

Despite this better choice of boundary conditions, the calculated ∇p is still an approximate extrapolation. The correct boundary values for `HbyA` and `rAU` needed in Equation (7) are not known. Instead, the values from the neighboring cells are used. Because the values of `HbyA` and `rAU` depend mainly on the velocity $\mathbf{U}$, we expect the extrapolation to be correct at the boundary only when the velocity gradient is zero normal to the boundary. As an OpenFOAM function object, the adapter is not able to access temporarily created fields such as `HbyA` or `rAU` to provide better extrapolation in cases of a non-zero velocity gradient. Therefore, the usage of the `fixedFluxExtrapolatedPressure` boundary condition is still recommended for all fluid-fluid coupling scenarios, as it provides superior results to a `fixedGradient` boundary condition.

[2] `constrainHbyA.C` (line 67): `https://develop.openfoam.com/Development/openfoam/blob/OpenFOAM-v2112/src/finiteVolume/cfdTools/general/constrainHbyA/constrainHbyA.C`

5 EXTENSIONS TO THE ADAPTER

Compared to the state of the OpenFOAM adapter presented in previous work [2], extensions have been implemented regarding inlet-outlet boundary conditions, mesh orthogonality, and additional coupled fields.

For scenarios where the flow direction at the interface might not be known beforehand or might change during the simulation, the adapter now provides custom coupling boundary conditions for pressure and velocity, which build upon OpenFOAM's `inletOutlet` boundary conditions. These boundary conditions change their behavior for in- and outflow for each cell dynamically, based on the sign of the mass flux values of the OpenFOAM field `phi`. These boundary conditions, called `coupledPressure` and `coupledVelocity`, are automatically available after adding the preCICE adapter library to the `controlDict`.

In scenarios in which the participants have skewed meshes close to the coupling interface, such non-orthogonality increases the error at the coupling interface up to a point where mass flux consistency cannot be guaranteed across the interface. For such scenarios, the adapter now provides a setting to adjust the retrieved velocity values by the mass flux as:

$$\mathbf{U} = \mathbf{U} - \mathbf{n}(\mathbf{n} \cdot \mathbf{U}) + \frac{\mathbf{n}\phi}{|\mathbf{S}|} \tag{8}$$

where ϕ are the face flux values, $\mathbf{n}$ are the face normal vectors, and $|\mathbf{S}|$ is the face area. Enabling this setting should, at least for matching meshes, ensure that mass is conserved across the coupling interface.

The features of the adapter presented so far are sufficient to couple simple laminar, single-phase flow solvers. To further expand the range of compatible solvers, support for coupling more physical quantities has been added. These include temperature (T) values and gradients at the fluid-fluid interface for heat transfer solvers such as `buoyantPimpleFoam`, as well as `alpha` (α) values and gradients for two phase solvers such as `interFoam`, where α is the phase variable in a volume of fluids method.

6 NUMERICAL EXAMPLES

This chapter showcases the results of several numerical examples that demonstrate the capabilities and limitations of the preCICE-OpenFOAM adapter. All described cases use a serial-implicit coupling scheme, which involves the participants running in an alternating fashion, with each time step being iterated until convergence of the coupled values is reached. All presented cases and corresponding code are available on Zenodo[3].

6.1 Laminar flow through a pipe

As a first example, we revisit the validation test case of a three-dimensional partitioned pipe from Chourdakis et al. [2]. The pipe points to the z-direction, has a diameter of 10m, length of 40m, and is split into two parts at $z = 20$m. The kinematic viscosity and density of the fluid are $\nu = 10\text{m}^2/\text{s}$ and $\rho = 1\text{kg}/\text{m}^3$ respectively. For the left pipe, a fixed (0.1m/s) velocity value and fixed (zero) pressure gradient are set at the inlet and a fixed velocity gradient and fixed (zero) pressure value at the outlet. The right pipe uses the same configuration for the outlet

[3] Zenodo archive: `10.5281/zenodo.8192959`

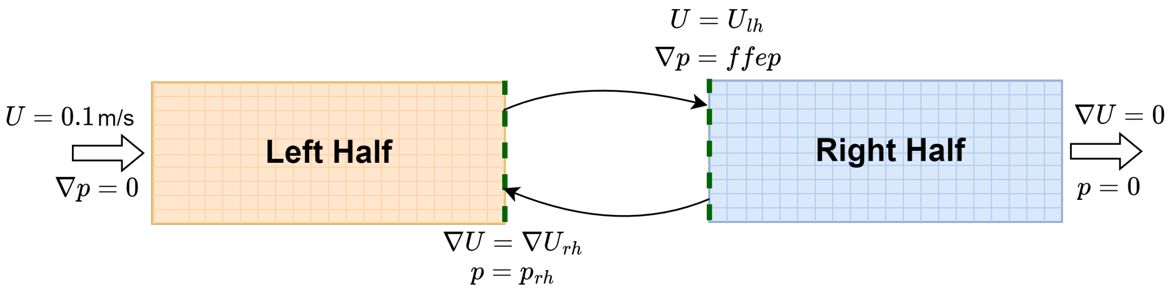


Figure 2: Setup of the partitioned pipe. *ffep* stands for the OpenFOAM boundary condition `fixedFluxExtrapolatedPressure`.

and a fixed value (zero) velocity with a `fixedFluxExtrapolatedPressure` boundary condition at the inlet. In terms of coupling configuration, the left pipe writes only velocity and reads pressure and the velocity gradient, while the right pipe complements it by reversing the write and read actions. The complete setup can be seen in Figure 2.

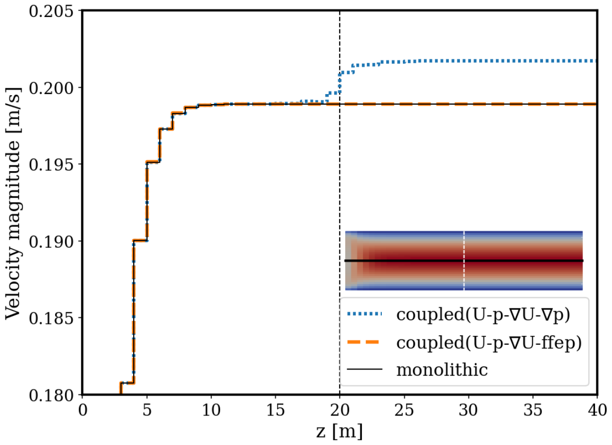


Figure 3: Velocity magnitude along the center of the pipe. Results are shown for coupling with all values and gradients (dotted blue) and after setting the *ffep* boundary condition at the coupling inlet (dashed orange). With this, the results are nearly identical to the monolithic solution.

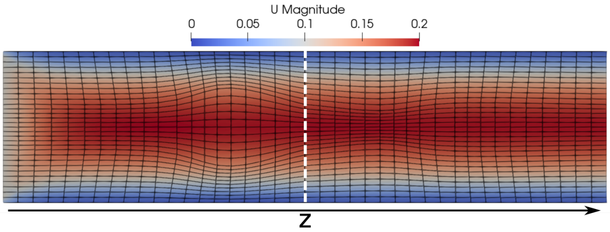


Figure 4: Partitioned pipe with a skewed mesh.

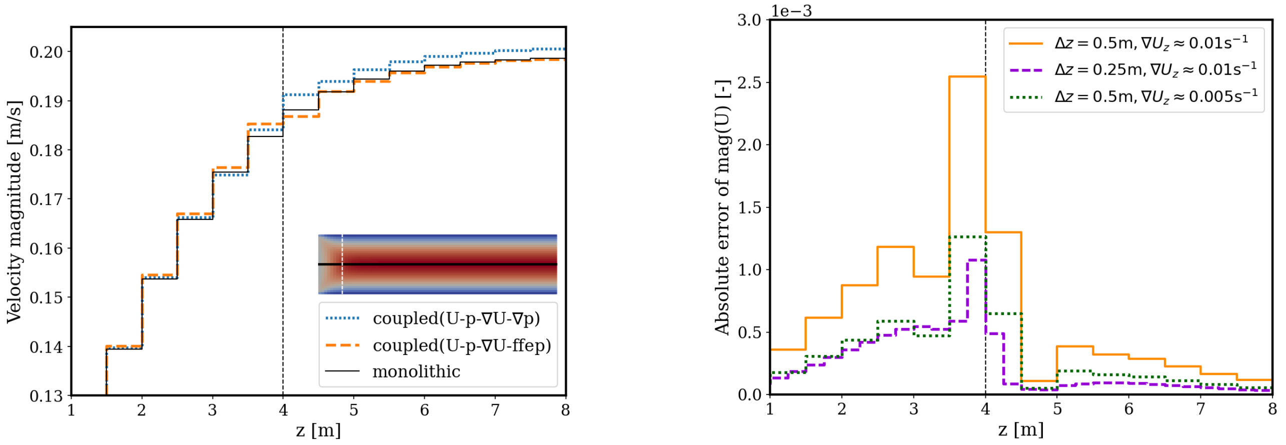
Figure 3 shows the velocity magnitude sampled along the center of the pipe in comparison to the complete Dirichlet-Neumann coupling setup from the previous work [2], and to the monolithic solution. It can be seen that the `fixedFluxExtrapolatedPressure` boundary condition eliminates the non-physical jump in velocity that was observed for coupling with a fixed pressure gradient. The relative errors compared to the monolithic solution are in the order of 10^{-6} and can be declared as purely numerical errors [3]. It is important to note that the `fixedFluxExtrapolatedPressure` boundary condition not only leads to quantitatively better results, but it also conserves the mass across the interface.

The mesh cells for the partitioned pipe are orthogonal with respect to the interface. Deforming the mesh as shown in Figure 4, we observe again a violation of mass conservation. With a cross-sectional area of 78.822m^2, the volumetric inflow rate is 7.822m^3/s. At the outlet of the right pipe, we measure an outflow rate of only 7.774m^3/s. However, after enabling the flux-correction option for velocity within the adapter, the outflow rate matches again the inflow rate. Whereas the boundary face values for velocity need to be extrapolated from the adjacent cell values, the values for the flux `phi` are directly evaluated on the faces and as such take the mass in- and outflow across the whole domain into account. However, this correction only works consistently when the meshes of the participants match at the interface.

By placing the interface at the center of the pipe, the coupling was done in a region with a fully developed flow profile. This implies that the velocity gradient is zero at steady-state. If we move the coupling interface closer to the inlet at $z = 4$m, the flow is still developing and a non-zero velocity gradient has to be expected. Figure 5a shows again the velocity magnitude in comparison to the monolithic solution as well as the full Dirichlet-Neumann coupling setup. While the `fixedFluxExtrapolatedPressure` boundary condition leads to a better overall coupling result, a noticeable error can still be observed. The error is largest in the cells adjacent to the coupling boundary, with the two neighboring velocity values being closer together than in the monolithic solution. This provides us with a heuristic formula for the largest expected error in velocity:

$$\epsilon(\mathbf{U}) < |\mathbf{U_L} - \mathbf{U_R}| \tag{9}$$

where $\mathbf{U_L}$ and $\mathbf{U_R}$ are the velocity values of the cells left and right of the interface from the monolithic simulation. This difference between two neighboring velocity values is depending on the magnitude of the gradient and the mesh width. As shown in Figure 5b, the error behaves indeed approximately proportional to these two factors.



(a) Velocity graph around the coupling interface at z=4m.

(b) Absolute errors of the velocity magnitude of coupled configurations vs their respective monolithic solution.

Figure 5: Velocity and error graphs for the partitioned pipe with the coupling interface at z=4m, where the flow profile is still in development.

6.2 Flow over a heated plate with partitioned flow

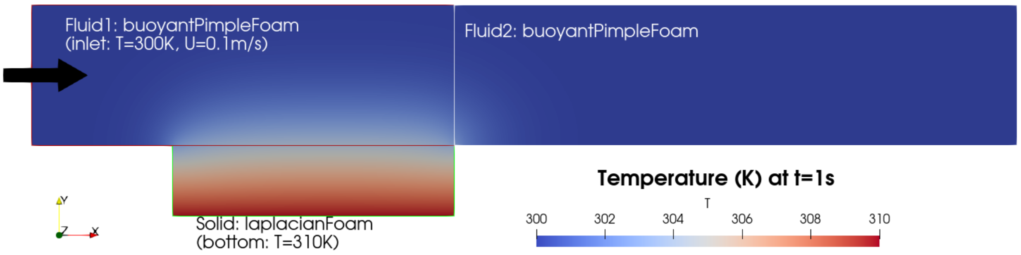


Figure 6: Setup and temperature profile for the flow over a heated plate scenario.

The *Flow over a heated plate* scenario is a preCICE tutorial for conjugate heat transfer[4]. The fluid enters a channel where it comes in contact with a plate that is heated from the bottom. The heat is transferred from the plate into the fluid where it is transported further by the stream. We extend this case by additionally partitioning the fluid domain and thereby validate the fluid-fluid module of our adapter for heat transfer with `buoyantPimpleFoam`. The fluid enters the domain with a fixed velocity of $U = 0.1\text{m/s}$ and a temperature of $T = 300\text{K}$. Figure 6 shows the complete setup of the scenario. The fluid-fluid coupling interface is located downstream of the heated plate. We use a basic Dirichlet-Neumann coupling for temperature, additionally to the pressure and velocity coupling described in the previous case. The case reaches the correct steady-state solution independent of the direction of the temperature coupling, as we can see from sampling the temperature at the interface, see Figure 7.

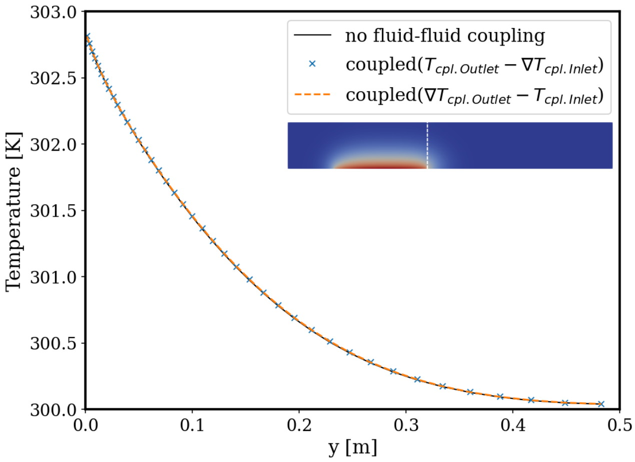


Figure 7: Temperature sampled across the fluid-fluid coupling interface at $t = 1\text{s}$ for the two coupling directions for temperature across the fluid-fluid interface.

6.3 Laminar flow over a backwards-facing step

Unidirectional Dirichlet-Neumann coupling for flow partitioning fails for more complex scenarios where the flow direction across the interface changes over time. In this next test case, we have

[4] Flow over heated plate tutorial: `https://precice.org/tutorials-flow-over-heated-plate.html`

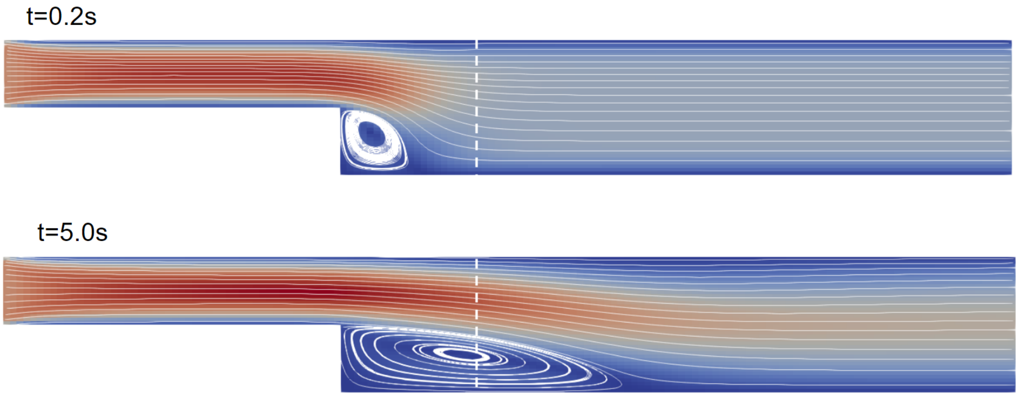


Figure 8: Velocity profile and streamlines for the coupled backwards-facing step scenario at $t = 0.2$s and at a converged time of $t = 5.0$s.

a two-dimensional laminar flow over a backwards-facing step. We use `pimpleFoam` for both participants and set the custom boundary conditions `coupledPressure` and `coupledVelocity` on both sides of the coupling interface. The inlet channel has a height of 0.01m and a length of 0.05m and opens up into a wider channel of double the height and length. The inlet velocity is fixed to 0.1m/s and the fluid viscosity is set to $\nu = 10^{-5}\text{m}^2/\text{s}$.

In the beginning of the simulation, fluid flows from the left into the right participant everywhere across the interface. This is shown for simulation time $t = 0.2$s in Figure 8. The recirculation pattern behind the step spreads to the right over time until it reaches the coupling interface at around $t = 0.5$s. As the vortex advances through the interface, the coupling boundary conditions change their behavior according to the flow direction for the bottom cells. Figure 8 shows the result at $t = 5.0$s which is identical to the monolithic solution except for the expected coupling error introduced in Section 6.1.

6.4 Breaking dam

As a last demonstration case, we present a two-dimensional breaking dam scenario, simulated with the volume-of-fluid multiphase solver `interFoam`. The case runs on a quadratic domain of length $l = 0.5$m with no-slip boundary conditions at the bottom and on both sides and an atmospheric boundary condition at the top. The simulation starts with a water column of height $h = 0.3$m and width $w = 0.15$m on the left side of the domain, which then falls to the ground as the simulation begins. Water is modelled with a viscosity $\nu_\text{W} = 1 \cdot 10^{-6}\text{m}^2/\text{s}$ and density $\rho_\text{W} = 1000\text{kg}/\text{m}^3$, air with $\nu_\text{A} = 1.48 \cdot 10^{-5}\text{m}^2/\text{s}$ and $\rho_\text{A} = 1\text{kg}/\text{m}^3$.

Initially, the water column is fully in the domain of the first participant, whereas the domain of the second participant is solely filled by air. As the water falls and spreads to the right, it crosses the interface at $x = 0.2$m. As it can be seen in Figure 9, the phase change travels smoothly across the coupling interface, where we use a basic Dirichlet-Neumann coupling for the phase variable α. Tracking the volume fraction of water in our domain, we realize that the coupling interface acts as a small sink for α. The initial volume fraction for water of 0.18 is reduced to 0.17996 at $t = 0.15$s and to 0.17988 at $t = 0.3$s. Figure 10 shows a plot of α across the interface, which confirms that the loss of water is experienced at the coupling interface.

The dam break scenario, similarly to many other multiphase scenarios, shows a vortex where

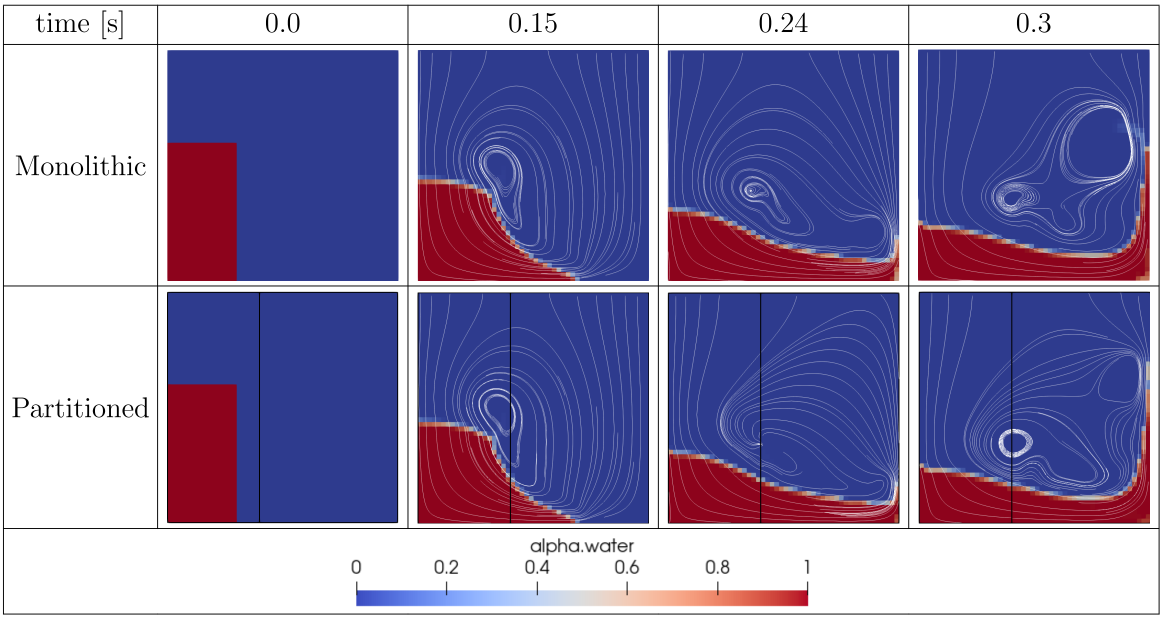


Figure 9: Streamlines and water phase fraction of the breaking dam scenario after several time steps. All domains are solved with `interFoam`.

air is flowing to the left into the space that was initially occupied by the water column. Due to the recirculating flow, we again used our custom inlet-outlet boundary conditions for pressure and velocity. However, it can be seen from the streamlines in Figure 9 that these boundary conditions fail at around $t = 0.24$s. The results are with a time step as low as $\delta t = 10^{-4}$s and an additional coupling of flux ϕ to stabilize the coupled boundary conditions. Yet, it appears that a mismatch of the boundary conditions occurs when the center of the vortex reaches the interface. Our interpretation is that, at this point, the velocity changes too much between the cells next to the interface, making a higher-order interpolation necessary at the interface.

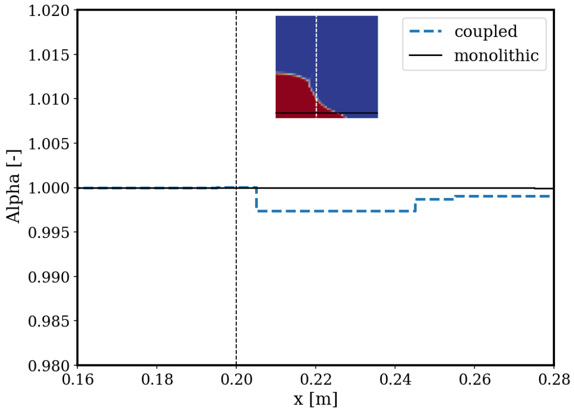


Figure 10: Values of α across the interface of the coupled breaking dam scenario at $t = 0.15$s.

7 CONCLUSION

Partitioned flow simulations with OpenFOAM via preCICE are possible for laminar incompressible flows. Numerical examples of partitioned pipe flows (with orthogonal, as well as non-orthogonal meshes across the interface) show results nearly identical to a monolithic OpenFOAM simulation when using a Dirichlet-Neumann coupling setup and an implicit coupling scheme. The OpenFOAM boundary condition `fixedFluxExtrapolatedPressure` allows setting non-zero pressure boundary gradients at the coupling interface while conserving mass across the interface, making it a better default recommendation than the previously recommended `fixedGradient` for pressure. Further numerical examples validate the adapter for coupling thermal, as well as two-directional flows with sufficient accuracy. A breaking dam scenario demonstrates the applicability of the approach to two-phase flows, with the results strongly affected by the intersection of small vortices with the coupling interface. The presented results are a drastic improvement upon previous work, and significantly expand the range of possible applications.

REFERENCES

[1] G. Chourdakis, K. Davis, B. Rodenberg, M. Schulte, F. Simonis, B. Uekermann et al. preCICE v2: A sustainable and user-friendly coupling library. *Open Research Europe*, 2:51, 2022. `https://doi.org/10.12688/openreseurope.14445.2`

[2] G. Chourdakis, B. Uekermann, G. van Zwieten, H. van Brummelen. Coupling OpenFOAM to different solvers, physics, models, and dimensions using preCICE. 14th OpenFOAM Workshop, 2019. `https://mediatum.ub.tum.de/node?id=1515271`

[3] M. Mühlhäußer. Partitioned flow simulations with preCICE and OpenFOAM. *Master's thesis*, Technical Univ. of Munich, 2022. `https://mediatum.ub.tum.de/node?id=1696254`

[4] G. Chourdakis, D. Schneider, B. Uekermann. OpenFOAM-preCICE: Coupling OpenFOAM with external solvers for multi-physics simulations. *OpenFOAM Journal*, 2023. `https://doi.org/10.51560/ofj.v3.88`

[5] G. F. Mintgen. Coupling of Shallow and Non-Shallow Flow Solvers – An Open Source Framework. *Dissertation*, Technical University of Munich, 2018. `https://mediatum.ub.tum.de/?id=1369622`

[6] M. Camps Santasmasas. Hybrid GPU/CPU Navier-Stokes lattice Boltzmann method for urban wind flow. *Dissertation*, University of Manchester, 2021. `https://research.manchester.ac.uk/en/studentTheses/hybrid-gpu-cpu-navier-stokes-lattice-boltzmann-method-for-urban-w`

[7] L. Grinberg and G. Em. Karniadakis. A scalable domain decomposition method for ultra-parallel arterial flow simulations. *Comm. in Comp. Physics*, 4(5):1151-1169, 2008. `http://global-sci.org/intro/article_detail/cicp/7832.html`

[8] M. A. Fernández, J.-F. Gerbeau, and S. Smaldone. Explicit coupling schemes for a fluid-fluid interaction problem arising in hemodynamics. *SIAM Journal on Scientific Computing*, 36(6):A2557 - A2583, 2014. `https://doi.org/10.1137/130948653`