OpenFOAM® Journal

Volume 3 [Full papers], Pages 1–25
ISSN: 2753-8168

OpenFOAM–preCICE: Coupling OpenFOAM with External Solvers for Multi-Physics Simulations

Gerasimos Chourdakis[1,*], David Schneider[2], and Benjamin Uekermann[2]

[1]Scientific Computing in Computer Science, School of Computation, Information and Technology
Technical University of Munich, Germany
Email address: gerasimos.chourdakis@tum.de

[2]Usability and Sustainability of Simulation Software, Institute for Parallel and Distributed Systems
University of Stuttgart, Germany

DOI: https://doi.org/10.51560/ofj.v3.88
Results with version(s): OpenFOAM® v2112
Repository: https://github.com/precice/openfoam-adapter

Abstract. Multi-physics simulations, such as conjugate heat transfer or fluid-structure interaction, are often constructed completely in OpenFOAM. However, they can also be formed by coupling OpenFOAM to third-party simulation software via a coupling tool. This approach indirectly adds to the capabilities of OpenFOAM those of other simulation tools (such as physical models or discretization methods more fitting for specific applications), and allows building complex multi-physics simulations by connecting specialized single-physics codes. We present the OpenFOAM-preCICE adapter, a function object that enables standard OpenFOAM solvers to use the open-source, massively parallel coupling library preCICE, without requiring any code modifications. We review alternative coupling approaches, analyze our design decisions, peek into key implementation details, validate the adapter, study the effect on runtime, and give an overview of the growing community of users and contributors.

1. Introduction

Multi-physics simulations have rapidly grown in importance during the past decade. Keyes et al. [1] give a good introduction and overview to the topic. The inherent complexity of these simulations make a flexible and maintainable software environment more and more important [2, 3]. The necessity to run efficiently on modern heterogeneous high performance computing systems further contributes to this fact.

There is a vast amount of approaches to implement multi-physics simulations with OpenFOAM (see section 2), which can be distinguished in internal and external approaches. Internal approaches implement multi-physics simulation completely within OpenFOAM (in a numerically monolithic or partitioned way [4]), but they inherently cannot go beyond the capabilities and restrictions of OpenFOAM itself. External approaches, on the other hand, try to couple OpenFOAM to other simulation codes (e.g., published codes with an established community in a specific field, or in-house codes of a research group), adding functionality. While such an approach sounds easy at first, it involves developing robust parallel algorithms for code communications, data mapping between non-matching meshes, acceleration and stabilization methods for strongly coupled problems, and more [5]. In this paper, we focus on such external (and by definition partitioned) approaches.

While individual codes have been coupled to OpenFOAM (see section 2), we are interested in code-independent coupling solutions. One such solution is the coupling library preCICE [6, 7]. preCICE is an open-source coupling library, which has proven scalability to tens of thousands of cores [8], comes with many ready-to-use adapters for other simulation codes, and has a fast-growing community of developers and users. There have already been previous preCICE adapter prototypes for specific OpenFOAM solvers for conjugate heat transfer (CHT) [9,10] and for fluid-structure interaction (FSI) [11,12]. They all have in common that they directly modify the OpenFOAM solver code, directly calling the preCICE application programming interface (API). This means that the user needs to become a developer, manually adapting

* Corresponding author

Received: 28 July 2022, Accepted: 9 February 2023, Published: 27 February 2023

each different solver they may need. With OpenFOAM already providing a wide range of ready-to-use solvers, a solution that lets the user directly couple these solvers is needed.

In this contribution, we present OpenFOAM-preCICE: a general, solver-agnostic preCICE adapter for OpenFOAM, implemented as an OpenFOAM function object calling preCICE. It supports CHT, FSI, and fluid-fluid coupling, but the adapter is extendable for other coupled problems as well, as demonstrated by the community. The initial software concept was proposed and implemented in the master's thesis of Gerasimos Chourdakis [13] and has been picked up and extended by many other research groups since. We give a summary of the main concepts, validation cases, and contributed extensions in this work, which serves as a reference for the community to build upon.

We start this paper by giving an overview of existing multi-physics approaches within OpenFOAM or between OpenFOAM and other solvers in section 2. We use this overview to also discuss different software engineering strategies for coupling in general. Afterwards, in section 3, we give a brief technical introduction to preCICE and review existing coupling approaches using preCICE and OpenFOAM. In section 4, we analyze the requirements of a general OpenFOAM adapter and justify our design decisions. In section 5, we characterize the adapter from the user's perspective, while in section 6, we lay out the technical implementation. In section 7, we validate the adapter through CHT and FSI benchmarks and study the effect of the adapter on the runtime. Finally, in section 8, we demonstrate the full potential of the general software concept by giving an overview of the community adoption and contributions. We conclude in section 9.

2. Review of existing coupling approaches with OpenFOAM

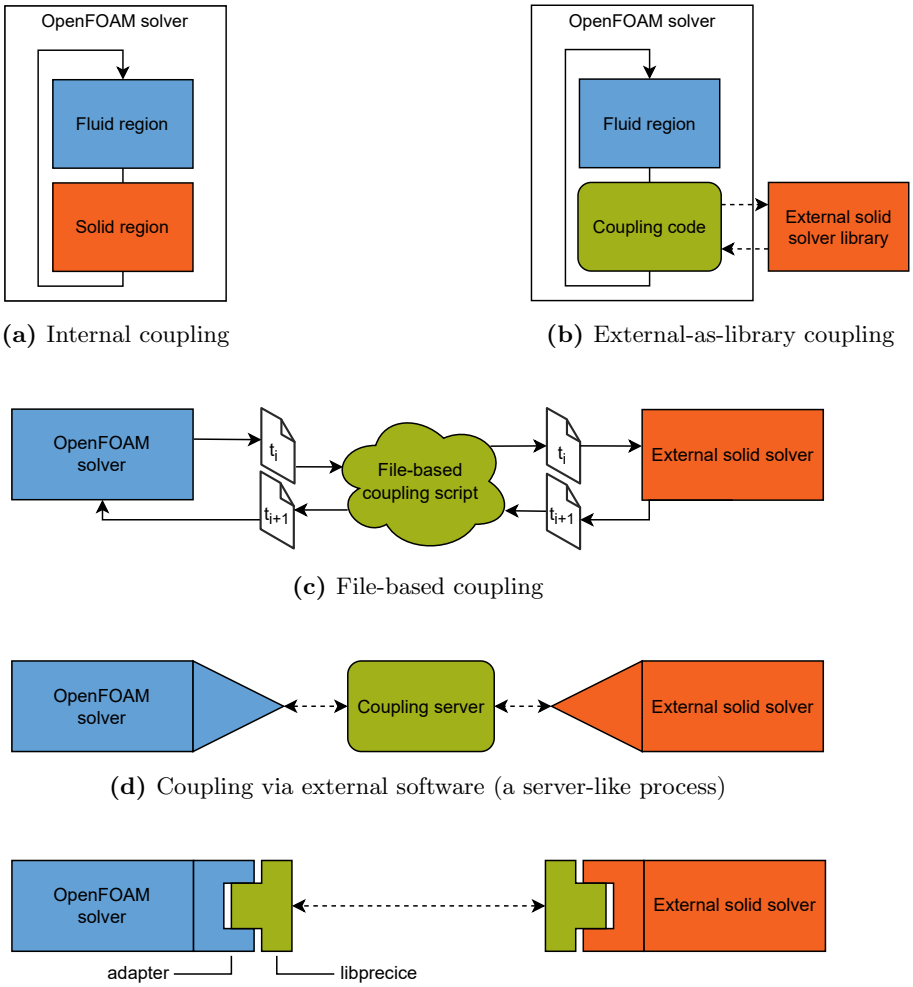
Different works have used OpenFOAM for solving multi-physics problems in a numerically partitioned way. We restrict the overview to the most wide-spread approaches of coupling domain regions and codes. We group the approaches into four categories: internal coupling, external-as-library coupling, file-based coupling, and coupling via external software. All approaches have their justifications, as they all offer advantages and disadvantages.

2.1. Internal coupling.

Various studies tackle multi-physics simulations exclusively within OpenFOAM (see Fig. 1a). A single mesh is read in and decomposed into several regions that are sharing an interface or overlapping over a volume. In each region, different equations are solved, while parallel communication goes through a single MPI communicator. The most prominent example for CHT is chtMultiRegionFoam [14] and the training code conjugateHeatFoam [15]. For FSI, prominent examples are the foam-extend packages fsiFoam [16] (studied e.g., by Luofeng et al. [17]) and solids4foam [18] (used, e.g., by Oliveira et al. [19]), as well as the training code simpleFsi [20]. The same approach is also followed by Wagner et al. [21] to extend OpenFOAM to FSI of 2D membranes. The main advantage of an internal coupling is simply that everything is integrated within OpenFOAM: An experienced OpenFOAM user does not need to learn any new tool. The respective codes are typically scalable on larger HPC systems, similarly to the single-region OpenFOAM solvers. Nevertheless, efficiency is limited, as the regions are typically solved one after another, leading to idling compute cores, a situation which can be prevented by parallel coupling schemes [22]. The clearest disadvantage, however, is the limited flexibility: All solvers need to be written in (the same) OpenFOAM framework. This approach leads to a monolithic (integrated / self-contained) software architecture, implementing the code for all equations in the same binary, while still implementing partitioned numerics.

2.2. External-as-library coupling.

To couple to external solvers, an external-as-library concept can be used (see Fig. 1b). OpenFOAM acts as the primary solver and calls an external (secondary) solver as a library. For FSI, such an approach is used, e.g., by Hewitt et al. [23, 24] and by Johan Lorentzon [25]. Beyond CHT and FSI, such approaches are also used for other multi-physics applications, such as for, e.g., coupling to 1D system codes for nuclear safety analysis [26] (via an additional layer). Compared to internal coupling, the main advantage of the external-as-library approach is simply that an external solver can be used. If the concept is properly implemented, this external solver can also be easily exchanged. The solver has to be rewritten as a library, however, which can lead to maintainability issues if the solver should also still be usable as a stand-alone tool. Another drawback is that such a coupling approach cannot easily be generalized for more than two coupled solvers, especially for asymmetric cases, where the external solver should be coupled yet again to another solver. Often, the external solver needs to use the same domain decomposition as OpenFOAM, limiting the efficiency of the approach if the external solver requires non-negligible compute effort.



(a) Internal coupling

(b) External-as-library coupling

(c) File-based coupling

(d) Coupling via external software (a server-like process)

(e) Peer-to-peer coupling approach with preCICE

Figure 1. Common coupling approaches to partitioned multi-physics simulations with OpenFOAM, applied on the example of FSI or CHT. The blue components denote the fluid region/solver, the orange components denote the solid region/solver, and the green components denote coupling machinery.

2.3. **File-based coupling.** Trying to couple external simulation packages without compatible software interfaces, one can rely on the exchange of input/output files (Fig. 1c). Typically, an external (in-house) script is used to process the result files, adjust configurations and restart the solvers. Such an approach is very intuitive and, thus, easy to implement and debug, but comes with efficiency, scalability, and maintainability issues. As the external solver does not need to be altered, even commercial closed-source simulation packages can be coupled. Note that this approach can also be used in combination with the rest of the approaches discussed in this section, to work around restrictive interfaces or licenses (see, e.g., the work by Huang et al. [27]).

2.4. **Coupling via external software.** Many researchers have applied one of the above approaches for their own specific setup, reinventing the wheel over many years for each application: how to exchange data between different solvers, how to map coupling data between different meshes, and how to deal with numerical instabilities for strongly-coupled problems. Furthermore, the effort of extending an existing solver by a coupling concept is often underestimated. Simple solutions quickly hit efficiency and scalability limitations for real applications. More complex solutions, on the other side, require higher maintenance effort. Coupling via an external software package is a good alternative for such cases. In some cases, the coupling software is a library, which gets called by both solvers. In other cases, the coupling software acts as a framework which then calls the solvers. Similarly to the external-as-library coupling concept above, this would mean to rewrite the solvers as libraries, introducing additional maintenance effort. Over the years, many coupling software packages have been developed: for an overview see, e.g., the dissertation of

Benjamin Uekermann [5]. Here, we restrict the review to those that feature an interface to OpenFOAM. Besides preCICE, the most wide-spread open-source alternative is probably OpenPALM [28]. Others are MUI [29], ifls [30, 31], or EMPIRE [32]. One proprietary example is MpCCI from Fraunhofer SCAI [33]. Further advantages of using an external coupling tool are the more efficient coupling algorithms, which are typically available, as well as the coupling of more than two participants (not true for all coupling software). The greatest disadvantage for users is typically the need to install, execute, and rely upon the sustainable future of a third software component, besides both coupled solvers. In our experience, however, this extra effort pays off in the long run. Concerning scalability, EMPIRE, ifls, and MpCCI use a further executable (a server-like process) to handle all communication, as shown in Fig. 1d. This typically limits the scalability of the approach, as the central process introduces communication and computation bottlenecks [5]. MUI, OpenPALM, and preCICE, on the other side, feature a pure peer-to-peer approach: The solvers communicate directly with each other without needing any central instance, allowing for scalable and efficient coupling (see Fig. 1e).

2.5. **Comparison: solids4foam and preCICE.** We observe that solids4foam and preCICE are both gaining interest in the OpenFOAM community at the moment, leading to the question: "what should I choose for my use case?". This is an overview of their main differences.

The multi-region FSI solver of solids4foam is a natural extension to OpenFOAM-based CFD simulations for an OpenFOAM user. solids4foam and preCICE are, however, addressing different needs and can work together. For example, one can couple the solid solver of solids4foam with the rest of the solvers that preCICE supports (see section 3). This list includes several OpenFOAM fluid solvers, which the user can couple to without needing to port them to solids4foam using OpenFOAM-preCICE[1].

An OpenFOAM user should consider solids4foam when they need an integrated OpenFOAM-based solution, while they should consider OpenFOAM-preCICE when they need to couple to a non-OpenFOAM-based code. In any case, both tools implement partitioned numerics (multiple regions), with some similarities in the coupling algorithms offered, but with preCICE being applicable to a wider range of problems. Finally, for users of third-party simulation codes, external coupling solutions are, by definition, the only way to extend their code with the capabilities of OpenFOAM.

3. Overview of the preCICE coupling library

preCICE[2] [6, 7] is a coupling library designed for minimally-invasive integration into existing codes. The library itself is not related or specific in any way to OpenFOAM and can be used by any simulation code, calling the high-level API of preCICE, which is available for several programming languages currently common in scientific computing (including C++, C, Fortran, Python, Julia, and Matlab). To minimize the additional software development for common use cases, the preCICE ecosystem currently includes official adapters for OpenFOAM, SU2, deal.II, FEniCS, Nutils, CalculiX, code_aster, and MBDyn, with new codes continuously added to list[3]. These typically manipulate the solver state, setting boundary conditions, volume terms, and time step size.

In the first part of this section, we describe the main concepts and terminology of preCICE that are needed to understand the rest of this paper, including terms that are used differently in the preCICE and the OpenFOAM literature. In the second part, we put this work into context with previous approaches to couple OpenFOAM with other solvers via preCICE.

3.1. **Fundamental concepts of preCICE.** preCICE allows for black-box coupling of an arbitrary number of *coupling participants*. It receives one cloud of points from each participant (*interface mesh*), maps the values between each pair of interface meshes, and iterates each *coupling time window* until convergence, while modifying the exchanged data with *acceleration methods*.

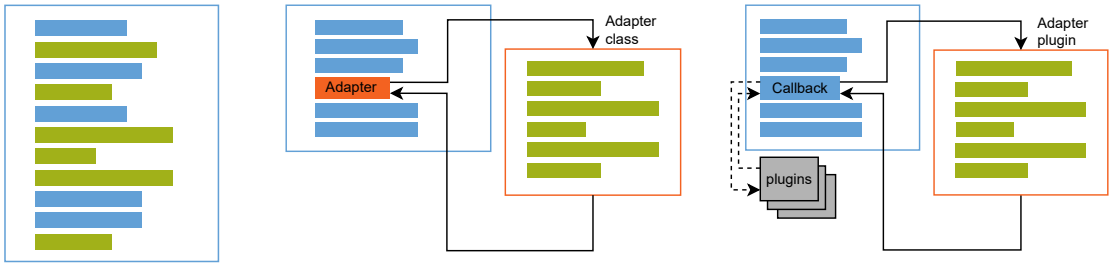
preCICE offers various coupling algorithms and acceleration methods, including algorithms for the simultaneous execution of all coupled solvers [34]. One prominent example is interface quasi-Newton algorithms [35]. For this iterative coupling procedure, we use the term *implicit coupling*, to contradict it from the single-iteration *explicit coupling*[4]. Both schemes are available in *parallel* (Jacobi-like) and *serial* (Gauss-Seidel-like) versions. During the coupling, each participant informs preCICE about the time step size it intends to simulate, which can be different for each participant and also different to the coupling

[1] See section 5 and the perpendicular flap tutorial: `https://precice.org/tutorials-perpendicular-flap.html`

[2] Source code on `https://github.com/precice/precice`, LGPLv3 license

[3] Overview of preCICE adapters: `https://precice.org/adapters-overview.html`

[4] In the OpenFOAM literature, the terms "subcycling" or "subiteration" may be used with the same meaning as "implicit coupling" here, while the terms "explicit" and "implicit" coupling may also be encountered with different meanings. preCICE follows the terminology used by Keyes et al. [1]. While the adapter can couple at every time step, OpenFOAM-internal approaches can also couple at every PIMPLE iteration, leading to numerically tighter coupling.



(a) Solver + preCICE calls **(b)** Solver calling adapter **(c)** Solver with callback support

Figure 2. Different software engineering approaches to implement a preCICE adapter: (A) directly modifying the solver to call preCICE, (B) modifying a small part of the code to call an Adapter class, (C) implementing the adapter as a plugin, if a solver provides suitable callbacks.

time window. We refer to this difference between solver time step and coupling time window sizes as *subcycling*. To participate in an implicit coupling scheme, a solver needs to be able to iterate over one or several time steps, returning its internal state to a *checkpoint* at the beginning of the coupling time window.

On the technical level, preCICE communicates in a fully-parallel, peer-to-peer fashion, using efficient communication channels, such as MPI ports or TCP sockets. Being developed primarily for high-perfomance computing, scalability of preCICE has been shown for more than 10 000 cores [8], with several improvements since. The communication method, meshes, mapping, coupling schemes, and several other coupling-related properties are configured at runtime using a single, XML-based *preCICE configuration file*, common to all participants. Anything specific to the coupling physics (mainly, boundary conditions) is usually configured by an *adapter configuration file*, specific to each adapter.

3.2. **Coupling OpenFOAM to other solvers via preCICE.** Writing a preCICE adapter for a single OpenFOAM solver is not an overly complicated task. We know of at least two variants for CHT [9, 10] and two variants for FSI [11, 12]. They all have in common that they directly modify the OpenFOAM solver code. This means that significant new development effort is needed to couple any new OpenFOAM solver.

In this paper, we present a general preCICE adapter for OpenFOAM[5], which works with arbitrary OpenFOAM solvers. This requires that the adapter does not modify the solver's source code and we use the callback functionality of function objects in OpenFOAM to avoid code changes, allowing the adapter to work even with precompiled solvers in binary form. Furthermore, the adapter needs to be independent not only of the specific solver, but also of the OpenFOAM version. We support recent and older versions from both OpenCFD as well as the OpenFOAM Foundation with version-specific adapter releases and development branches (including OpenFOAM v2212 and OpenFOAM 10, the latest releases at this time)[6]. We focus on CHT and FSI, but the adapter is extendable for any surface- or volume-coupled application. The design and implementation is described in detail in the master's thesis of Gerasimos Chourdakis [13] and is based upon previous work of Lucia Cheung Yau [9] for CHT. Later, it was extended with an FSI module, based on similar work by David Schneider [12] and in close collaboration with Derek Risseeuw [36]. More recently, the adapter added initial support for fluid-fluid coupling [37].

4. Requirement analysis and design decisions

To design a general OpenFOAM–preCICE adapter, we first have to carefully define what we mean by *general*. In this section, we look deeper into a few key requirements and design decisions that led us implement the adapter as an OpenFOAM function object. We then explain the concept and the most important implementation details of OpenFOAM function objects, including a few challenging limitations of the available interface.

4.1. **Requirement analysis.** Previous preCICE adapters for OpenFOAM solvers [9–12] either directly use the preCICE API within the OpenFOAM solver or offload the preCICE API calls to an adapter class and call the adapter interface in turn from the solver (Fig. 2). The first approach is intuitive and can be

[5]Source code on `https://github.com/precice/openfoam-adapter`, GPLv3 license

[6]Supporting older OpenFOAM versions is important for users, as they are often given a specific version pre-installed on a system, or they need to adhere to a specific version or framework over the time span of a research project.

implemented rather quickly, but hinders the maintainability and leads to duplicate effort to couple any other OpenFOAM solver. The second approach appears at first as a decent solution: to couple a new OpenFOAM solver, one would only need to add a few new adapter API calls in the solver, potentially with further modifications to support additional models. This approach, however, still requires the user to momentarily become a developer, using the adapter as a development framework instead of a ready-to-use tool. Partitioned multi-physics simulations are anyway complex due to the large number of software components involved, thus minimizing each barrier is crucial for the success of the user.

We require the following properties from a general OpenFOAM adapter:

(1) The adapter needs to be a separate software package with an independent building routine and only use already available code interaction features. The OpenFOAM solver code should not need any modifications or re-compilation.
(2) The adapter needs to be solver-agnostic. To allow coupling of arbitrary OpenFOAM solvers, the solver name should not be used in the adapter. In particular, the boundary conditions, where the adapter needs to decide which physical quantities are used as input and output of the coupling, should not depend on the implementation details of the solver, but only on the coupling physics.
(3) The adapter should support all preCICE features, such as implicit coupling (i.e., checkpointing and repeating several time steps) and nearest-projection data mapping (i.e., providing mesh connectivity).
(4) The adapter should be as independent of the OpenFOAM version as possible. This requirement has limitations, as OpenFOAM development progresses and the development paths of the various OpenFOAM variants diverge. To reduce code complexity, this requirement could be replaced by a comprehensive version support policy in the future. In any case, defining common standards across variants would be beneficial for the sustainable support of different variants and versions.

4.2. **Design decision: The adapter as a function object.** In order to avoid re-compilation of the solver's code (requirement #1), we need to isolate the adapter code into a separate software package. An approach could be to implement a new boundary condition, which would be selected at runtime. However, one quickly finds limitations in this approach: even if the data on the boundary can be set, the time step size cannot be manipulated and the solver's complete state cannot be saved and restored for checkpointing. This additional manipulation of the solver is possible via OpenFOAM function objects.

OpenFOAM function objects [14] are external tools called at predefined points during a solver's execution, and they are mainly used for post-processing, e.g., to probe fields on specific points or compute the forces acting on a patch. A wide palette of such post-processing tools are distributed with OpenFOAM and they have also been used for coupling, such as in the MpCCI adapter for OpenFOAM [38, Section 14.1.4] and in the simpleFsi project [20]. Our work has also influenced other researchers to implement an ifls adapter for OpenFOAM using function objects [31].

Function objects are shared libraries which implement the abstract class `functionObject`. At runtime, the solver collects a list of function objects and executes their respective methods when specific events are triggered. The main events of interest are a time advance (a call to `runTime.run()` in the outer loop) and a modification of the time step size (a call to `adjustDeltaT()`, usually within `setDeltaT.H`). This mechanism is depicted in Fig. 3 and gives us the possibility to inject calls to preCICE in the beginning of every time step (distinguishing between initialization, evolution, or finalization), as well as every time the time step size is updated. With careful grouping and positioning of the calls to preCICE, these points are enough for a complete adapter supporting all features of preCICE.

However, decoupling the adapter from the solver's main code comes with some implications. The necessary objects (e.g., time, state variables, boundaries) cannot be accessed directly, but they can be accessed through the registry of objects. A reference to it is provided through the mesh object, which is of type `Foam::polyMesh`, and objects can be accessed in the form `lookupObject<Type>(name)`. These objects are, however, not designed to be modified by function objects, even though this is technically possible and an important cornerstone of the adapter. In particular, OpenFOAM assumes strictly-increasing time and follows a different execution path for the first, last, and intermediate time steps. During the last time step, OpenFOAM already triggers the final execution of function objects (happening only once in a simulation), which requires special treatment by the adapter to be able to repeat this time step until convergence. Special treatment is also required in error handling and in collaboration with other function objects. We explain these aspects in section 6. Despite the aforementioned challenges, function objects remain an appropriate, full-featured interface for an adapter that satisfies the requirements described in this section.

Until now, we have mentioned design features that decouple our adapter from the solver's main code, although these are not enough to guarantee generality. It is correct that the same function object can

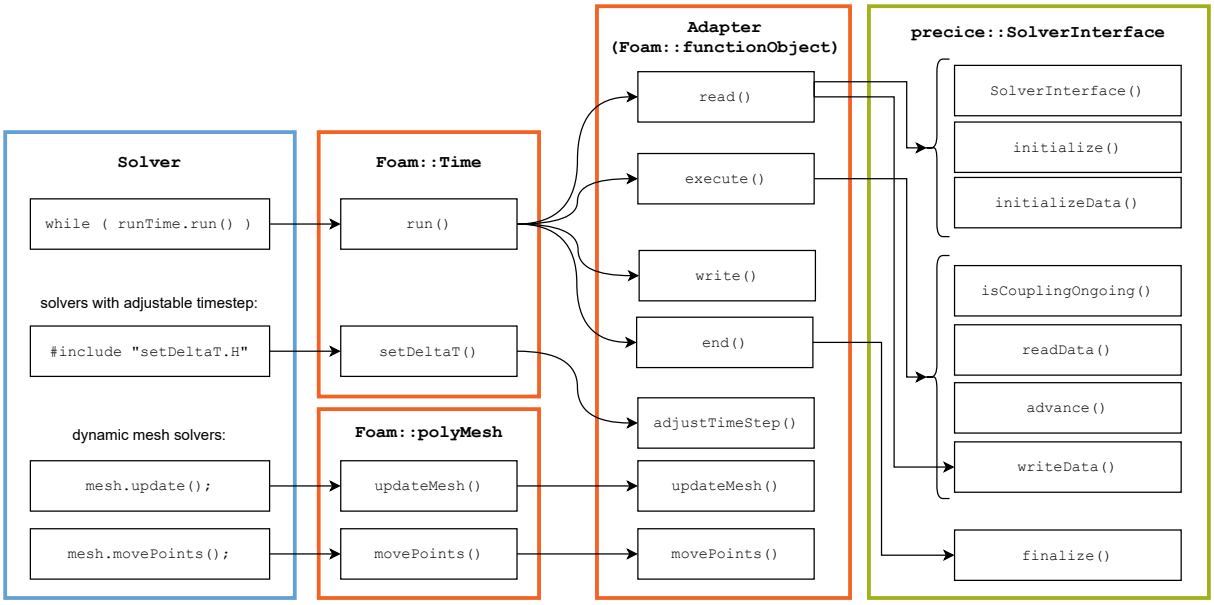


Figure 3. OpenFOAM function objects: call flow, based on the OpenFOAM v2206 code, including calls triggered by dynamic mesh solvers.

be loaded by any solver, but it is not correct that it will work with any solver: for example, looking for a field in the objects registry will lead to a runtime error in case the requested field does not exist. Different solvers may be based on different, independent OpenFOAM model classes[7], making it difficult or impossible to access the required objects in a uniform way. However, we can distinguish most of the solvers shipped with OpenFOAM in three groups (compressible flow, incompressible flow, and basic), based on the dimensions of the pressure field (m^2/s^2 or $kg/(m \cdot s^2)$), if available. This distinction allows us to access all required objects of each group in the same way.

5. User perspective

To use the OpenFOAM-preCICE adapter, one only needs to update configuration files and not write any additional code. The adapter can be loaded at runtime as a regular function object, specified in the `system/controlDict` configuration file. The association between exchanged data and mesh patches, as well as adapter-specific settings, are defined in the (new) `system/preciceDict` configuration file. All the coupling settings are defined in the preCICE configuration file (usually named `precice-config.xml`), which is shared among all coupling participants and not discussed here[8].

The adapter configuration file `system/preciceDict` is an OpenFOAM dictionary. In this file, the name of the preCICE participant is specified, as well as the path to the preCICE configuration file. Next, a list of preCICE interfaces follows, with each interface relating to a mesh defined in the preCICE configuration file, the OpenFOAM patches participating in the interface, and the names of data that the adapter has to write and read. The association between data names and fields is inferred by the data name, matching the prefix: for example, a coupling dataset called `TemperatureSolid` is treated as temperature, by default associated to the `T` field. A variety of fields are supported, which are currently grouped into *modules* for conjugate heat transfer, fluid-structure interaction, and fluid-fluid (FF) coupling:

- Module `CHT`: temperature, heat flux, heat transfer coefficient, sink temperature.
- Module `FSI`: force, stress, absolute displacement, displacement relative to the last coupling time window.
- Module `FF`: pressure, pressure gradient, velocity, velocity gradient.

Modules need to be specifically enabled and multiple modules can be enabled at the same time. This distinction into modules prevents field name conflicts and allows extensibility, as described in section 6.

The preCICE tutorials[9] demonstrate the adapter being used by the OpenFOAM solvers laplacianFoam, buoyantPimpleFoam, buoyantSimpleFoam (all for CHT), pimpleFoam (FSI, FF), and sonicLiquidFoam

[7]The class architecture of thermodynamical models was significantly simplified in OpenFOAM 8, making further simplifications possible for specific OpenFOAM versions.
[8]preCICE configuration: `https://precice.org/configuration-overview.html`
[9]preCICE tutorials: `https://precice.org/tutorials.html`

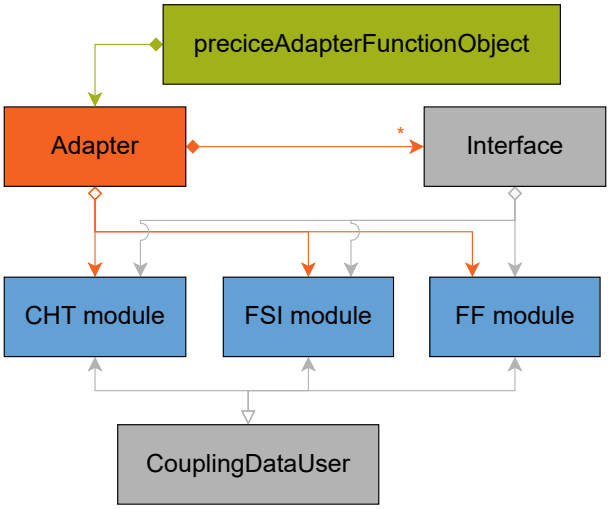


Figure 4. Structure of the adapter (simplified UML class diagram).

(FF), while the community has already used the adapter with several other solvers (see section 8). For details regarding the adapter configuration, see the OpenFOAM adapter documentation[10].

6. Implementation

In this section, we describe the architecture and the most important implementation details of the adapter specific to OpenFOAM. These include adjusting the time step size, checkpointing the internal solver state, error handling, as well as the coupling physics and extension possibilities.

6.1. **Architecture.** The general structure of the adapter is depicted in Fig. 4, where we present the main components in a simplified UML class diagram. The class `preciceAdapterFunctionObject` is only a wrapper to the actual adapter, implementing the `fvMeshFunctionObject` abstract class of OpenFOAM. It only calls the methods implemented in the `Adapter` class, separating the core of the adapter from the (version-specific) function object interface.

After reading the adapter configuration file, `Adapter` instantiates one or more `Interface` objects. Each interface object corresponds to a preCICE interface mesh: While a two-participant simulation normally needs one interface object per participant, the adapter can define more interfaces, enabling coupling to more than one other participants. Interface meshes can either be defined on face centers or face nodes of each patch, while the community (see section 8) has also extended the adapter to define meshes on cell centers. These interface meshes can be defined both for two-dimensional simulations (oriented in the (x, y) plane and using the cell centers), as well as for three-dimensional simulations. In addition to mesh points, the adapter can also extract mesh edges and construct mesh triangles, an essential component for nearest-projection data mapping (i.e., interpolation on the nearest mesh element of the other participant's mesh) [6].

The abstract class `CouplingDataUser` provides the skeleton for extracting the boundary values and setting the boundary conditions for each type (temperature, force, etc.). These types are encapsulated into *modules*, which mainly contain implementations of the `CouplingDataUser` class. The name *module* was chosen because this physics-related functionality is isolated from the rest of the code and new modules can easily be implemented with very few code changes, as shown in subsection 6.10. Further separation of the module code from the rest of the adapter is possible, but not currently planned. The adapter and the modular approach were originally designed for CHT, while the FSI and FF modules were added later.

6.2. **Steering.** The adapter needs to call the following methods of preCICE to steer the coupling:

- `double initialize()`, to start the communication and compute data mappings (returns the maximum allowed time step size),
- `void initializeData()`, to exchange (non-zero) initial data (optional),
- `double advance(double computedTimestepLength)` after every time step, to perform the coupling (expects the current time step size and returns the maximum allowed time step size for the next iteration), and

[10]Adapter configuration: `https://precice.org/adapter-openfoam-config.html`

- `void finalize()` at the end of the coupled simulation, to finalize communication channels.

The first two methods are called at the configuration stage of the adapter, via the function object's `read()` function. The next two methods are called in the function object's `execute()`. Even though `execute()` is evaluated in the beginning of each time step, the adapter treats this call as the end of the previous time step. Therefore, the boundaries already contain the computed solution and we can fill the buffers, call `advance()` and update the boundaries, before proceeding with the computations of the next time step. The end of the simulation is controlled by preCICE with `isCouplingOngoing()`, in which case the adapter calls `finalize()` to close the connections and tear down data structures.

6.3. **Adjusting the time step size.** After calling `advance()`, the adapter receives a new time step size from preCICE, which it applies by invoking the `setDeltaT()` on the reference to the solver's `runTime` object. The solver can use a smaller time step, if it wants, in which case it is *subcycling*. However, it cannot use a time step size that is larger than the one allowed by preCICE.

6.4. **Checkpointing.** During implicit coupling, we need to store the state of the solver in a *checkpoint* and later return the solver to this state. In order to not assume which are the essential, non-derived fields of each solver, we copy every field of `vol, surface, pointScalar, Vector, TensorField` and `volSymmTensorField` type that we can find in the registry of objects, using the `sortedNames<Type>()` function. Additionally, we store and reload the mesh fields "old-time" cell volumes (`fvMesh::V0`), the "old-old-time" cell volumes (`fvMesh::V00`), and the cell face motion fluxes (`fvMesh::phi`)[11]. Next to the checkpointed fields, the adapter also stores the time, which is later reloaded by calling the `setTime` method of the `runTime` object. The solver then re-executes the time step normally, including the time step size computation.

6.5. **Error handling.** OpenFOAM uses its own output and error streams for handling exceptions and we adopt them, as well. However, in some OpenFOAM versions, a potential error caused inside the `read()` method of a function object does not force a program exit, but it instead degrades to a warning and the execution continues[12]. As we perform several checks during the configuration phase and we want to throw errors, if needed, we track the errors during the configuration with a status variable, which we then use to throw an error during the first call of `execute()`. This and more challenges signify that we have reached the borders of what is possible with function objects at the moment.

6.6. **Collaboration with other function objects.** To collaborate with other function objects, and because the adapter controls the end of simulation, the adapter explicitly triggers the `end()` method of all function objects at the end of the simulation.

6.7. **Coupling physics: The CHT module.** The functionality specific for conjugate heat transfer simulations is isolated into the *CHT module*. This contains a set of `CouplingDataUser` classes (see Fig. 4), which can write and read temperature and heat flux (for Dirichlet-Neumann coupling), or sink temperature and heat transfer coefficient (for Robin-Robin coupling). These classes were first designed and implemented by Lucia Cheung Yau [9].

The module distinguishes between compressible, incompressible, and basic solvers, mainly affecting the way that the effective heat conductivity is accessed. For compressible solvers, the effective conductivity for each boundary cell is extracted from the compressible turbulence model class[13], using the method `kappaEff()`. For incompressible solvers, the effective conductivity is computed as:

$$k_{\text{eff}} = \alpha_{\text{eff}} \rho c_{\text{p}} = (\alpha + \alpha_{\text{t}})\, \rho c_{\text{p}} = \left(\frac{\nu}{Pr} + \frac{\nu_{\text{t}}}{Pr_{\text{t}}} \right) \rho c_{\text{p}} \ , \tag{1}$$

where ν is the kinematic viscosity of the fluid, α is the thermal diffusivity, $Pr = \nu/\alpha$ is the (dimensionless) Prandtl number, while ν_{t}, α_{t}, and Pr_{t} are the turbulent parts of the respective parameters. In such incompressible solvers, ν (total) and α_{t} are available through the incompressible turbulence model class, Pr (total) is already available, while ρ and c_{p} need to be provided as additional constant parameters in the adapter configuration file. For basic solvers (such as laplacianFoam), the conductivity is considered a constant scalar, which is read as an additional parameter from the adapter configuration file.

[11] Class `fvMesh` docs: `https://www.openfoam.com/documentation/guides/v2206/api/classFoam_1_1fvMesh.html`

[12] Related issue: `https://develop.openfoam.com/Development/openfoam/-/issues/1779`

[13] We refer here to OpenFOAM v2206 and OpenFOAM 7 or earlier. Since OpenFOAM 8, we can find `kappaEff` in the `thermophysicalTransportModel` class, a class that is used by both compressible and incompressible solvers.

6.8. **Coupling physics: The FSI module.** The *FSI module* shares a design similar to the CHT module. This module can read absolute and relative (to the previous coupling time window) displacement and write force and stress. Both incompressible and compressible solvers are supported, including turbulent flow models. These classes were first added by Derek Risseeuw [36].

Displacements are directly applied onto the `pointDisplacement` field boundary. Since the interface is moving over time, the velocity of the fluid next to it needs to be adjusted to follow the wall motion. The `movingWallVelocity` boundary condition[14] automatically sets the wall velocity for moving meshes and is used in all preCICE tutorials. The tutorials also use the displacementLaplacian mesh motion solver, but the adapter code is not specific to that. Special treatment is required for restarted simulations, in which case we need to move the interface points according to the previously accumulated displacements.

Forces are computed in two parts: pressure forces and viscous forces, which are computed as in the `forces` function object[15]. Both types of forces depend on the density and viscosity, while the pressure can be either kinematic or total, requiring again a separation between incompressible and compressible solvers.

6.9. **Coupling physics: The FF module.** The *FF module* provides velocity, pressure, and their gradients, for partitioned flow simulations. These fields were shown to be enough for reproducing basic multi-model flow simulations of a water hammer example [37], while the module allows users to easily add more fields, e.g., for turbulent flows, as described in subsection 6.10. This module is currently under development and further examples have been demonstrated by Markus Mhlhuer [39].

Pressure and velocity are directly read and written from and to the boundary of the respective field. The gradients are read to the `.gradient()` of the boundary and written from the `.snGrad()` of it. In the case of velocity, the gradient is inverted at the writing step, by convention. A tutorial with a partitioned pipe is available for the incompressible pimpleFoam and the compressible sonicLiquidFoam solvers.

6.10. **How to extend the adapter.** Several users have already extended the adapter for their own use cases, as we discuss in section 8. The code is widely documented with comments explaining the main steps and design decisions, as well as notes for code changes needed to add a new coupling data user or a complete module.

For example, to add a new field `MyField` in the FF module, one would need to modify only files in the `FF/` source directory. Two new files, `MyField.H` and `MyField.C`, would define the class `MyField` as a child of `CouplingDataUser`, implementing a constructor and the methods `write` and `read`. The configuration methods `addReaders` and `addWriters` of the FF module (in the file `FF.C`) would then need to be updated to add the new class as an option.

To add a complete new module `MyModule`, one would need to create a directory `MyModule/`, a header and implementation file defining a free-standing `MyModule` class[16], providing a constructor and the `configure`, `addWriters`, and `addReaders` methods. The configuration of the adapter, `configFileRead`, needs then to be extended with the name of the module as an available option. While these modules are not as simple as a "plug-in" mechanism, they still facilitate extending the adapter. In any case, the user does not need to modify any code interacting with preCICE and writing a new module should be possible for an OpenFOAM user with limited programming experience.

7. Validation

While the preCICE community has applied the adapter to a variety of complex projects, as discussed in section 8, conjugate heat transfer and fluid-structure interaction are the most common fields of application. Multiple preCICE tutorials are also demonstrating such scenarios. To know if the adapter (and a complete coupled setup) is producing correct results, we evaluate common benchmark scenarios[17]. We start by partitioning a heat equation and comparing the results to the single-domain OpenFOAM solution (subsection 7.1), before comparing a partitioned (buoyantPimpleFoam-laplacianFoam) and an OpenFOAM-only (chtMultiRegionFoam) conjugate heat transfer case (subsection 7.2). We then couple pimpleFoam and an external solid mechanics solver to simulate the 2D Turek-Hron FSI benchmarks [40]

[14] Source code of the `movingWallVelocity` boundary condition: `https://www.openfoam.com/documentation/guides/v2206/api/movingWallVelocityFvPatchVectorField_8C_source.html`

[15] OpenFOAM v2206 code for forces function object: `https://develop.openfoam.com/Development/openfoam/-/blob/OpenFOAM-v2206/src/functionObjects/forces/forces/forces.H`

[16] In Fig. 4, we show the modules as children of the `CouplingDataUser` class. This is a simplification, as in reality the modules only contain objects of types deriving from `CouplingDataUser`.

[17] While most of these benchmarks are simulation results from different codes, and since the main focus of this publication is the software (rather than the specific applications), we use the term "validation", to differentiate from the (formal) software verification.

(subsection 7.3) and a 3D FSI benchmark [41] (subsection 7.4). For all simulations presented in this section, we have used OpenFOAM v2112, preCICE v2.4.0, and the OpenFOAM adapter v1.1.0. A virtual machine image is available[18] for reproducibility, based on the preCICE Demo Virtual Machine[19].

Note that this section aims to validate only the adapter. The mapping and coupling algorithms of preCICE have already been validated in separate publications [5, 7, 42]. We compare the accuracy of the partitioned simulations against single-domain solutions (subsection 7.1), intrinsic OpenFOAM solvers (subsection 7.2), and against published results (subsection 7.3, subsection 7.4). A performance comparison of OpenFOAM-preCICE to other solutions presented in section 2 is out of the scope of this paper.

7.1. **Partitioned heat condution.** In order to validate the CHT module of the adapter, minimizing the influence from external components, we solve first a simple heat-conduction problem:

$$\frac{\partial T}{\partial t} - \nabla^2 T = f \qquad \text{for } x \in \Omega \tag{2a}$$

$$T = T_\mathrm{D} \quad \text{for } x \in \Gamma_\mathrm{D} \tag{2b}$$

on a domain Ω, with boundary Γ_D, the unknown T, the right-hand side f, and the boundary values T_D. We define $\Omega = [0, 2] \times [0, 1]$ and follow the example given in The FEniCS Tutorial I [43, section 3.1], which creates a problem with linear variation in time and quadratic variation in space:

$$T\left(x, y, t\right) = 1 + x^2 + \alpha y^2 + \beta t \tag{3}$$

Inserting (3) into (2) gives that the right-hand side f of (2) needs to be $f\left(x, y, t\right) = \beta - 2 - 2\alpha$. The boundary condition is $T = T_\mathrm{D} = 1 + x^2 + \alpha y^2 + \beta t$ and the initial condition is $T = T_0 = 1 + x^2 + \alpha y^2$. We select $\alpha = 3$ and $\beta = 1.3$, as used in the preCICE tutorials with other solvers[20].

To solve this problem in a partitioned fashion, we split the domain in two parts, $\Omega_1 = [0, 1] \times [0, 1]$ and $\Omega_2 = [1, 2] \times [0, 1]$, as shown in Fig. 5. The heat equation is solved in both subdomains, and the coupling is carried out through preCICE at the common coupling boundary $\Gamma_\mathrm{C} = \Omega_1 \cap \Omega_2$. For the left side of the domain, the coupling boundary Γ_C is a Dirichlet boundary (reading temperature), whereas for the right part of the domain, the coupling boundary is a Neumann boundary (reading heat flux). For the rest of the boundary, it still applies that $T = T_\mathrm{D}$.

Mathematically, the described exchange of boundary conditions corresponds to a Dirichlet-Neumann coupling, which is commonly used in coupled simulations. Alternatives such as Robin-Robin coupling, where each participant applies mixed boundary conditions, are supported by the adapter as well.

Since OpenFOAM does not include a heat-equation solver which can handle non-zero right-hand side values, we extended the laplacianFoam solver for this coupling scenario, which we use on both sides[21]. For setting the boundary values on Γ_D, we use groovyBC, part of swak4Foam[22]. The setup is available in the directory `partitioned-heat-conduction/` of the accompanying case files archive.

We compare how the error decreases for increasingly fine meshes of the partitioned and the OpenFOAM-only setups in Fig. 6, which shows identical results (same convergence behavior, average relative error of $4 \cdot 10^{-6}$, maximum error of $2 \cdot 10^{-5}$).

7.2. **Conjugate heat transfer.** After having validated the very basic partitioned heat conduction scenario in subsection 7.1, we can now validate a first multi-physics scenario. We simulate the flow over a heated plate, a case first studied in the context of preCICE by Lucia Cheung Yau [9] and originally inspired by the experimental benchmark case by Vynnycky et al. [44]. This benchmark comprises a 2D flow of a cold fluid above a thick solid plate, heated at its bottom, as depicted in Fig. 7.

In this case, we compare a partitioned setup (coupled with preCICE) to an OpenFOAM-only setup (internal coupling of two regions) for a transient and a steady-state simulation. For the partitioned setup, we couple buoyantPimpleFoam (fluid domain, both transient and steady-state simulations) with the (standard, unmodified) laplacianFoam. For the reference OpenFOAM-only setup, we use chtMultiRegionFoam. We have modified the original case [44] by coarsening the mesh and lowering the inlet velocity, in order to reach a steady-state in reasonable time, for both the partitioned and the OpenFOAM-only setup. The setup is available in the `flow-over-heated-plate/` directory of the accompanying case files archive.

[18] Vagrant box specific to this publication: `https://app.vagrantup.com/precice/boxes/openfoam-precice-paper-vm`

[19] preCICE Demo VM: `https://precice.org/installation-vm.html`

[20] Partitioned heat conduction tutorial: `https://precice.org/tutorials-partitioned-heat-conduction.html`

[21] Solver code: `https://github.com/precice/tutorials/tree/master/partitioned-heat-conduction/openfoam-solver` The code included in the case appendix contains additional changes for reporting error estimations.

[22] swak4Foam: `http://www.openfoamwiki.net/index.php/Contrib/swak4Foam`

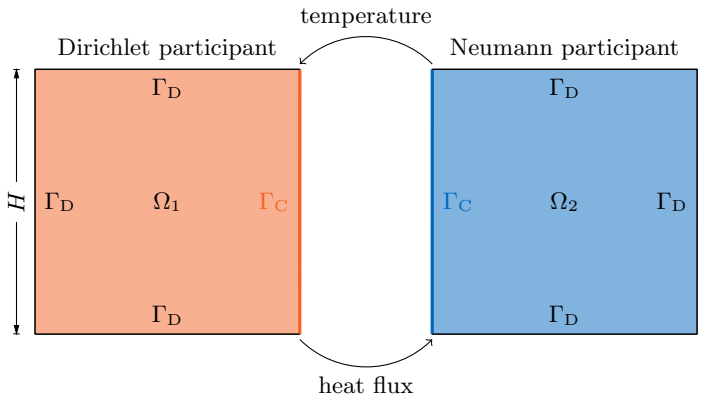


Figure 5. Partitioned heat conduction scenario: setup. Two square solid plates of side $H = 1\,\mathrm{m}$ are coupled on Γ_C via preCICE. On the boundaries Γ_D, the temperature is prescribed as a function of space and time (satisfying (3)). The Dirichlet participant is reading temperature values from preCICE, while the Neumann participant is reading heat flux.

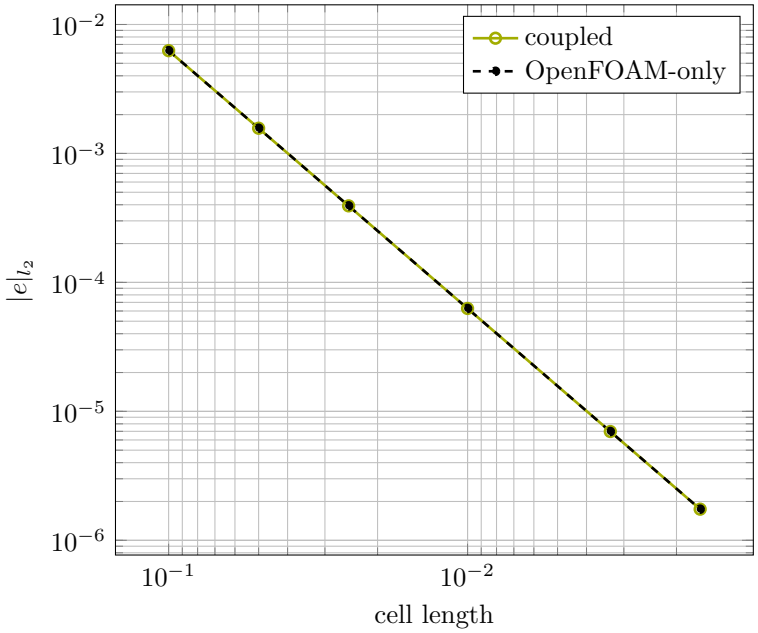
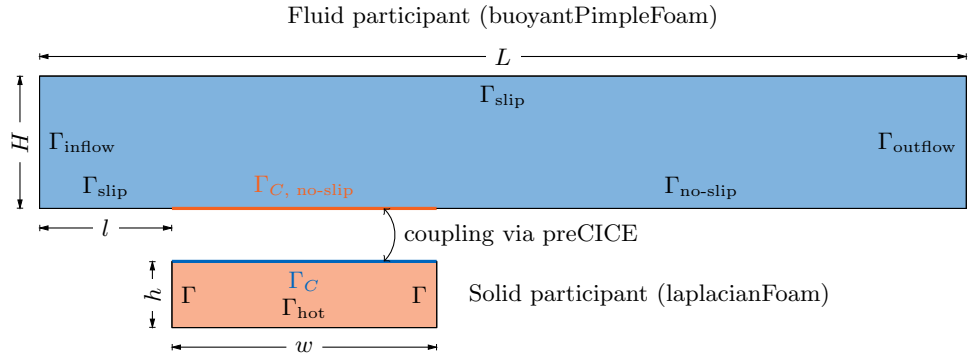


Figure 6. Partitioned heat conduction scenario: $|e|_{l_2} = \sqrt{\frac{1}{n}\sum_{i=0}^{n-1}\left(T\left(x,y,1\right)-T_i\right)^2}$ error over all vertices at $t = 1\,\mathrm{s}$ for the coupled problem as well as the OpenFOAM-only solution. The coupled scenario and the OpenFOAM-only solution employ an absolute convergence measure of 10^{-9}. The coupled scenario uses a nearest-neighbor mapping with matching interface nodes.

In the *transient simulation*, we first perform a serial explicit coupling with a coupling time window size of $\Delta t = 0.01\,\mathrm{s}$, set the number of outer correctors in chtMultiRegionFoam to $n_{\mathrm{corr}} = 1$ (i.e., explicit coupling between the two regions of chtMultiRegionFoam), and compare the results at $t = 1\,\mathrm{s}$. The difference of the dimensionless interface temperature, $\theta = (T - T_{\mathrm{inflow}})/(T_{\mathrm{hot}} - T_{\mathrm{inflow}})$, between the OpenFOAM-only (θ_{o}) and the coupled simulation (θ_{c}), over all interface nodes is $e_\theta = \sqrt{\frac{1}{n}\sum_{i}^{n-1}\left(\theta_{\mathrm{c}}^i - \theta_{\mathrm{o}}^i\right)} = 1.05{\cdot}10^{-2}$, with the result files parsed and handled manually in double precision[23].

[23]Using the `foamToVTK` tool and parsing the results with the VTK Python library, the precision of the results is not enough to compute all errors with a sufficient accuracy.



Dimensions		
L	3.50	m
H	0.50	m
l	0.50	m
h	0.25	m
w	1.00	m
z	0.40	m

Mesh (cells)	
Fluid	104×41
Solid	32×15

Fluid			
u_{inflow}	2	$\times 10^{-1}$	$\mathrm{m\,s^{-1}}$
T_{inflow}	3	$\times 10^{2}$	K
p_0	1.035	$\times 10^{5}$	Pa
g	9.81		$\mathrm{m\,s^{-2}}$
k_F	1	$\times 10^{2}$	$\mathrm{W\,m^{-1}\,K^{-1}}$
$c_{p,F}$	5	$\times 10^{3}$	$\mathrm{J\,kg^{-1}\,K^{-1}}$
μ	2	$\times 10^{-4}$	$\mathrm{kg\,m^{-1}\,s^{-1}}$
ρ_F	1		$\mathrm{kg\,m^{-3}}$
ν^*	2	$\times 10^{-4}$	$\mathrm{m^2\,s^{-1}}$
Pr^*	1	$\times 10^{-2}$	
M^*	24.10		$\mathrm{kg\,mol^{-1}}$

Solid			
T_{hot}	3.1	$\times 10^{2}$	K
k_S	1	$\times 10^{2}$	$\mathrm{W\,m^{-1}\,K^{-1}}$
$c_{p,F}$	1	$\times 10^{2}$	$\mathrm{J\,kg^{-1}\,K^{-1}}$
ρ_S	1		$\mathrm{kg\,m^{-3}}$
α_S^*	1		$\mathrm{m^2\,s^{-1}}$

Figure 7. Conjugate heat transfer scenario: case setup of a flow over a heated plate. Figure from the preCICE v2 reference paper [6], with the modified parameters in blue. The same parameters and mesh sizes are used for the corresponding chtMultiRegionFoam simulation.

The reliability of the results generated with chtMultiRegionFoam is expressed through the diffusion number. In the discussed transient simulation scenario, we determined a diffusion number of $Di_{\max} \approx 140$, which was too large to reliably compare the OpenFOAM-only and the coupled solution approach. Reducing the time step size of both sides, as well as the coupling time window size to $4 \cdot 10^{-4}$ ($Di_{\max} \approx 6$) reduces the error to $e_\theta = 2.42 \cdot 10^{-4}$. Switching from an explicit to an implicit coupling scheme with an absolute convergence measure of $1 \cdot 10^{-8}$ on the heat flux and setting the number of outer corrector loops to $n_{\text{corr}} = 20$ further reduces the error to $6.62 \cdot 10^{-5}$. The error and the value of θ along the interface are depicted in Fig. 8. However, the convergence measures for the OpenFOAM-only case and the coupled CHT simulations are still different (as the algorithms are different), so there is no guarantee that the two setups converge up to the same accuracy. In order to eliminate this uncertainty, we compare results for a steady-state simulation, where the time evolution is irrelevant.

In the *steady-state simulation*, we perform a serial explicit coupling with a coupling time window size of $\Delta t = 0.01\,\mathrm{s}$ and equal time step sizes for both solvers. For the OpenFOAM-only setup, we set the number of outer correctors in chtMultiRegionFoam to $n_{\text{corr}} = 1$ and compare the results at $t = 100\,\mathrm{s}$, after which we observe that the simulation has converged to a steady state. The steady-state computation results in an error of $e_\theta = 4.65 \cdot 10^{-8}$, therefore we conclude that our partitioned setup produces results very close to chtMultiRegionFoam for both steady-state ($e_\theta = 4.65 \cdot 10^{-8}$) and transient ($e_\theta = 6.62 \cdot 10^{-5}$) CHT scenarios, with the selected convergence measures of each algorithm.

7.3. **Turek & Hron fluid-structure interaction benchmarks.** To validate the FSI module of the adapter, we simulate the numerical benchmark cases FSI2 and FSI3 proposed by Stefan Turek and Jaroslav Hron [40]. We simulate the fluid domain with pimpleFoam and the solid domain with a coupled solid solver[24] from the ExaDG project [45]. The setup is available in the directory `turek-hron-fsi/` of the accompanying case files archive.

pimpleFoam applies three PIMPLE pressure corrections (`nCorrectors`), five SIMPLE corrections (`nOuterCcorrectors`), and three non-orthogonal corrector loop iterations (`nNonOrthogonalCorrectors`) in order to account for large mesh distortions. The mesh motion is calculated solving a Laplace equation. The mesh motion solver solves directly for the displacement field and a prescribed displacement serves as Dirichlet boundary condition in each coupling time step. An absolute tolerance of 10^{-6} is set for

[24] ExaDG-based solid solver: `https://github.com/davidscn/exadg/tree/fsi-setups`

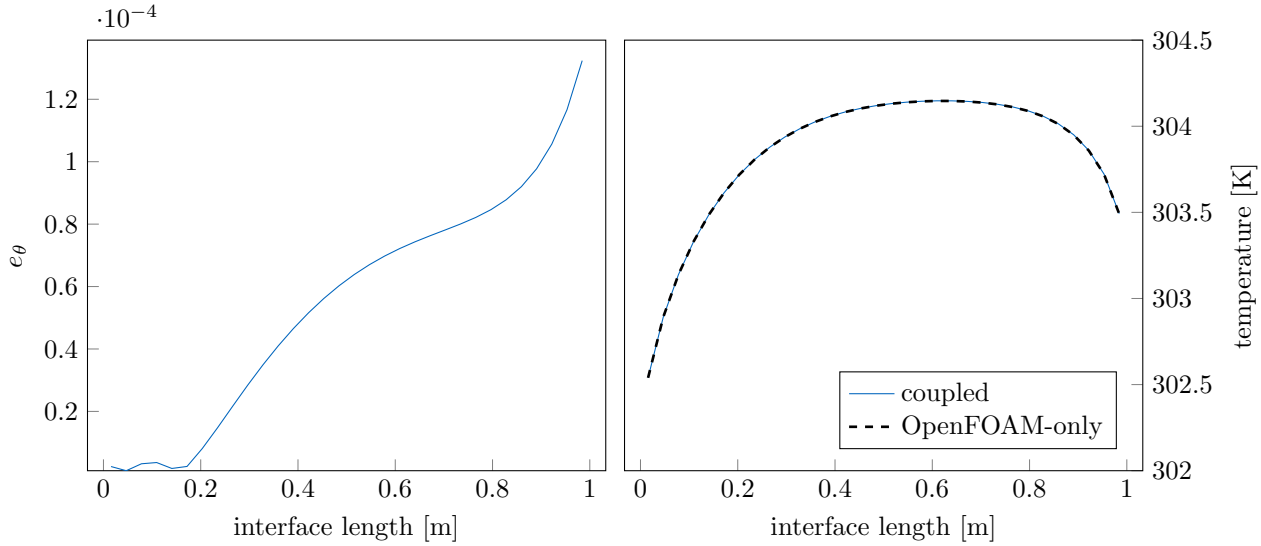


Figure 8. Conjugate heat transfer scenario: Error of the dimensionless temperature θ compared to the OpenFOAM-only results (left) and absolute temperature at the common coupling interface (right) for the transient scenario at $t = 1s$ using an implicit coupling scheme and a small time-step size of $\Delta t = 4 \cdot 10^{-4} s$.

`cellDisplacement` and the diffusion coefficient (`displacementLaplacianCoeffs`) is set to be inversely proportional to the distance to the flexible flap.

The solid participant is given by a solver from the ExaDG project. The material is elastic and compressible and modeled by a St. Venant-Kirchhoff material, as in the original paper [40]. We employ a Bossak-α method with a spectral radius of 0.8 for the discretization in time, implying a slight numerical damping, and continuous $\mathcal{Q}^2$ finite-elements for the discretization in space.

The two domains follow different discretizations (fluid: FVM, solid: FEM) and the meshes are non-conforming at the interface. We study three cases of different mesh refinement levels, as shown in Tab. 1 and use global thin plate spline radial basis functions for data mapping in preCICE [7]. We apply a parallel implicit coupling scheme with a relative convergence measure of $1 \cdot 10^{-4}$ for each coupling data vector. The convergence is accelerated using a least-square quasi-Newton scheme [35] for both data vectors using a residual-sum preconditioner [5] and QR2 filtering [46] with a threshold of $1.2 \cdot 10^{-3}$.

case	#cells fluid	#cells solid
1	20 969	16
2	38 489	64
3	45 903	256

Table 1. Turek-Hron FSI2 and FSI3 scenarios: Spatial resolution of the coupled simulations (coarse, medium, and fine meshes). The solid part has been discretized using quadrilateral $\mathcal{Q}^2$-elements, which leads to a total number of 5 018 degrees of freedom in the structural part for case 3.

The benchmark describes partial tests for each solver (CFD and CSM), as well as three FSI tests. In the following, we investigate the FSI2 and FSI3 setups as proposed and described in the original reference, while we omit the simpler FSI1 case. Both scenarios result in periodic solutions and the simulation setups differ only in the simulated solid material and inflow velocity. In order to stabilize the initial state of the test cases, we precompute two seconds of the flow field without coupling. Afterwards, we start the coupled simulation using the precomputed flow field on the fluid side and the initial configuration on the solid side. Each coupled simulation is carried out for 15 seconds of simulation time and we assume a converged state of the periodic data. The quantities of comparison are in particular the force exerted from the fluid onto the cylinder and the flap as well as the displacement of the flap tip center. The periodic data is characterized by its mean value, its amplitude, the corresponding frequency and a plot over the time period. The minimum and maximum values required for the mean value and the amplitude are computed by taking the minimum and maximum of the last two oscillations. In order to compute the

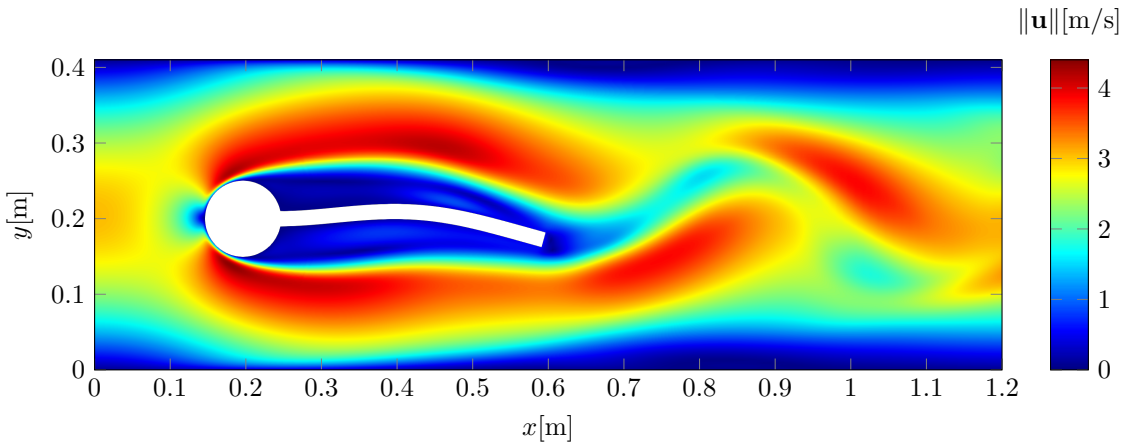


Figure 9. Turek-Hron FSI3 scenario: Velocity magnitude [m/s] of the refinement case 3 (with $\Delta t = 5 \cdot 10^{-4}$) at $t = 14\,\mathrm{s}$.

case	Δt	$\Delta x\ [\times 10^{-3}]$	$\Delta y\ [\times 10^{-3}]$	F_x (drag)	F_y (lift)
1	0.0010 s	-2.37 ± 2.26 [11.0]	1.55 ± 31.30 [5.6]	458.9 ± 21.76 [11.0]	1.84 ± 155.9 [5.6]
2	0.0010 s	-2.40 ± 2.29 [11.0]	1.52 ± 31.62 [5.6]	457.5 ± 23.46 [11.0]	2.26 ± 150.6 [5.6]
3	0.0010 s	-2.41 ± 2.30 [11.0]	1.51 ± 31.73 [5.6]	457.6 ± 23.75 [11.0]	2.36 ± 149.9 [5.6]
1	0.0005 s	-2.58 ± 2.46 [11.0]	1.53 ± 32.84 [5.4]	460.8 ± 23.13 [11.0]	1.72 ± 157.7 [5.4]
2	0.0005 s	-2.65 ± 2.52 [11.0]	1.50 ± 33.37 [5.4]	459.7 ± 25.27 [11.0]	2.24 ± 154.0 [5.4]
3	0.0005 s	-2.67 ± 2.54 [11.0]	1.49 ± 33.51 [5.4]	459.8 ± 25.57 [11.0]	2.22 ± 153.6 [5.4]
[40]	0.0005 s	-2.69 ± 2.53 [10.9]	1.48 ± 34.38 [5.3]	457.3 ± 22.66 [10.9]	2.22 ± 149.8 [5.3]

Table 2. Turek-Hron FSI3 scenario: Results for all refinement cases listed in Tab. 1. Δt refers to the time step size. Δx and Δy refer to the displacement of the tip of the flap in the x and y directions. F_x and F_y refer to the drag and lift forces acting on the complete immersed body (cylinder and flap). Displacements and forces are given in average $\pm$ amplitude [frequency] notation. Comparing the last two rows, the results of case 3 are almost identical to the reference, with the frequencies differing in the last significant digit, and the rest of the results also being directly comparable. Note that the reference results have been acquired using different codes, meshes, and numerical setups [40].

frequency, we apply a fast Fourier transform (FFT) on the periodic data for the last five seconds, time range for which we consider that the flow is fully developed. The choice of this time range can also lead to slight differences in the computed frequency: Depending on the considered time interval, the frequency of the lift as well as the y displacement varied from $5.33 - 5.66$ Hz.

A surface plot of the velocity field at a deformed state illustrating the setup of the FSI benchmarks is shown in Fig. 9. Results for the quantities of comparison for the FSI3 scenario are summarized in Tab. 2 and depicted in Fig. 10. The quantitative results in Tab. 2 indicate a convergence with increasing refinement in space and time and the plots in Fig. 10 are in agreement with the results reported in the original work.

Next, we analyze our results for the FSI2 scenario: the density ratio of $\rho_s/\rho_f = 10$ leads to lower frequency of oscillations and numerically more stable coupling conditions compared to the FSI3. However, the main challenge of these parameter settings is given by the significantly larger mesh deformations, which need to be handled properly by the mesh motion solver. The results for the FSI2 scenario are summarized in Tab. 3 and Fig. 11. Depending on the considered time interval used for the FFT, the frequency of the drag as well as the x displacement varied from $3.75 - 4$ Hz. Due to the large deformation of the flap, the flow solver crashed for refinement levels 2 and 3 (cf. Fig. 12) and only the coarsest refinement case 1 completed the entire 15 seconds used for our result analysis.

Such a mesh failure can be prevented using a more advanced mesh motion solver (here displacement-Laplacian). RBF-based approaches are known to perform better [47] and are implemented as extensions for OpenFOAM [48], but were not readily compatible with the adapter at the time of obtaining these

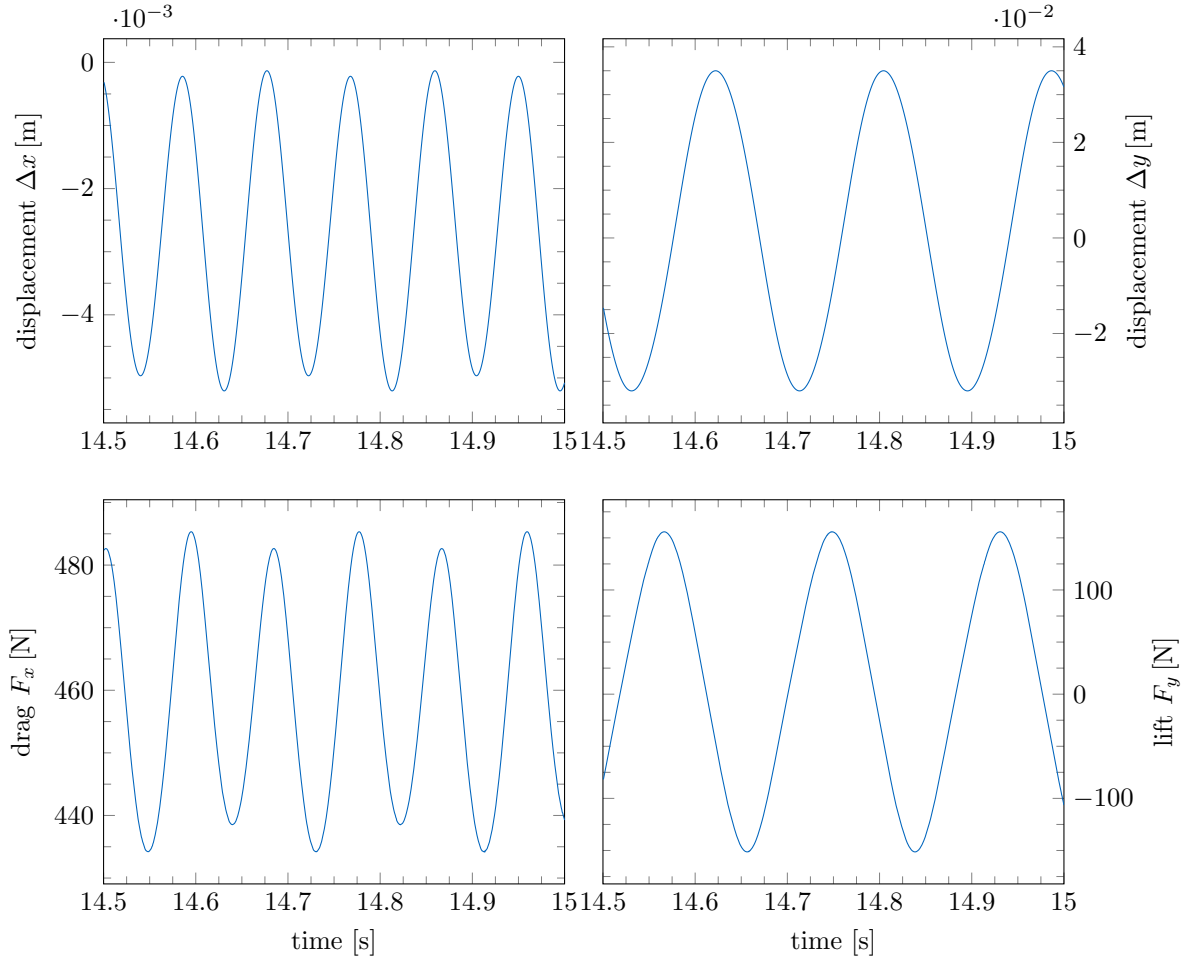


Figure 10. Turek-Hron FSI3 scenario: Plot of the periodic data for refinement case 3 and $\Delta t = 5 \cdot 10{-}4s$, which can be compared with the FSI3 results on page 259 of the original paper [40].

results. The (newer) adapter version 1.2.0 supports the updated implementation included in solids4foam, as a result of collaborative effort with the developer of the latter, and modifying the setup accordingly[25] does help the discussed case 2 to complete.

Even though we consider here only the coarsest refinement cases (while the original benchmark requires a mesh convergence study) with displacementLaplacian, the results are very close to the results presented by Turek & Hron [40] and the characteristic oscillations around the extremum of the lift force are resolved accurately by the flow solver.

7.4. **Fluid-structure interaction benchmark: pulsating flow over a flexible flap.** While the benchmark cases presented above are all two-dimensional, the adapter also supports three-dimensional simulations. We study here a 3D FSI case, as presented by Schott et al. [41]. This case comprises an elastic flap immersed in a channel flow, as depicted in Fig. 13 and Fig. 14. The setup is available in the directory `pulsating-flow-over-flexible-flap/` of the accompanying case files archive.

We simulate the fluid domain again with pimpleFoam, using the same numerical settings as described in subsection 7.3. As opposed to subsection 7.3, the solid material in this benchmark is a compressible non-linear neo-Hookean material. Therefore, we employ our own solid solver[26], based on the finite-element library deal.II [49]. The solver is based on the work of Davydov et al. [50] and is extended by an implicit Newmark scheme [51] with $\beta = 0.25$ and $\gamma = 0.5$ (no numerical damping).

[25]Using the RBFMeshMotionSolver requires a currently unreleased state of solids4foam, which is compatible with older OpenFOAM versions. It also only works with cell displacements, which the adapter supports by setting `locations faceCenters` for the respective interface at the `preciceDict`.

[26]solid solver: `https://github.com/davidscn/matrix-free-dealii-precice`

case	Δt	Δx $[\times 10^{-3}]$	Δy $[\times 10^{-3}]$	F_x (drag)	F_y (lift)
1	0.002 s	-13.78 ± 12.03 [3.8]	1.19 ± 77.9 [2.0]	209.00 ± 71.54 [3.8]	0.35 ± 229.6 [2.0]
2	0.002 s	–	–	–	–
3	0.002 s	–	–	–	–
1	0.001 s	-14.06 ± 12.17 [3.8]	1.22 ± 78.9 [2.0]	210.69 ± 73.41 [3.8]	0.31 ± 236.1 [2.0]
2	0.001 s	–	–	–	–
3	0.001 s	–	–	–	–
[40]	0.001 s	-14.58 ± 12.44 [3.8]	1.23 ± 80.6 [2.0]	208.83 ± 73.75 [3.8]	0.88 ± 234.2 [2.0]

Table 3. Turek-Hron FSI2 scenario: Results. Displacements and forces are given in average ± amplitude [frequency] notation. Refinement cases 2 and 3 failed due to mesh failure (cf. Fig. 12), but the coarsest case results are identical to the reference results in terms of frequencies, while the rest of the results are directly comparable. Note that the reference results have been acquired using different codes, meshes, and numerical setups [40].

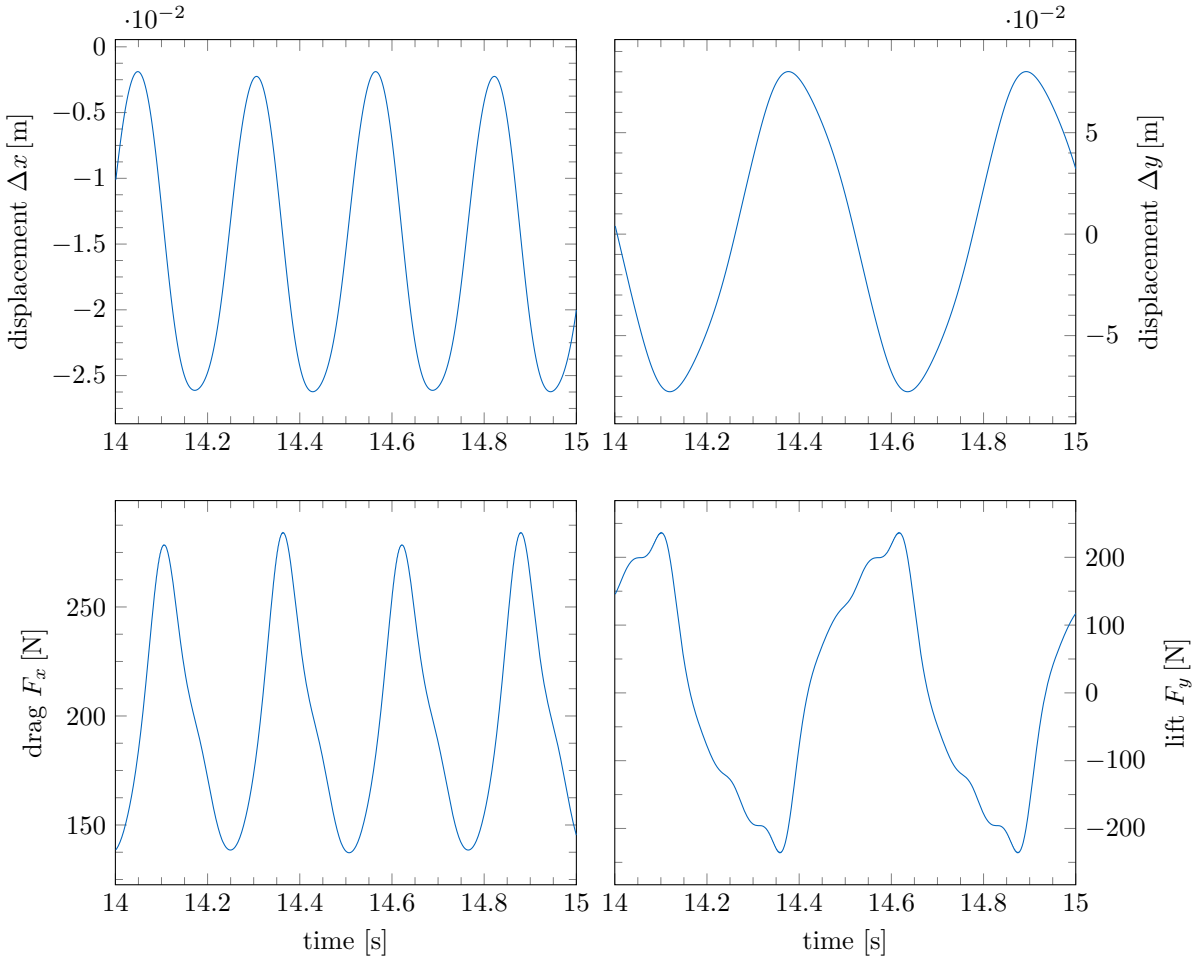


Figure 11. Turek-Hron FSI2 scenario: Plot of the periodic data for refinement case 1 and $\Delta t = 0.001s$, which can be compared with the FSI2 results on page 258 of the original paper [40].

We simulate the test case on three different mesh refinement levels, as described in Tab. 4. In order to resolve the interface numerics appropriately, we apply a mesh grading on the fluid mesh so that the mesh region around the flexible flap gets more refined. For the fine fluid mesh, the mesh grading towards the flap was lowered compared to the coarse and medium case, to allow large mesh deformations. We

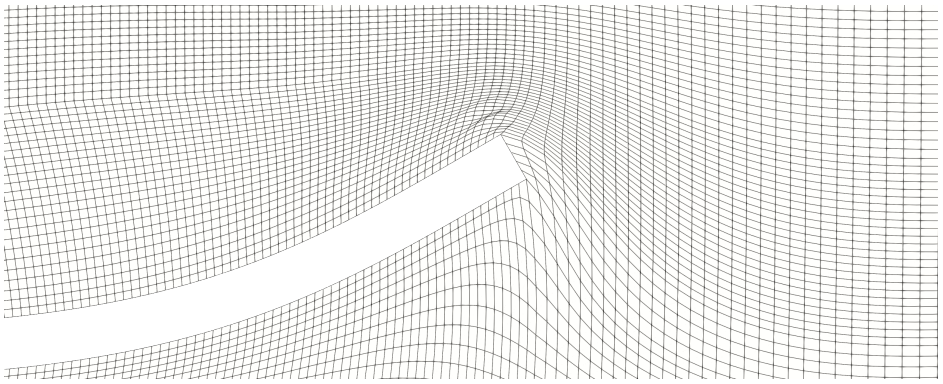


Figure 12. Turek-Hron FSI2 scenario: Deformed mesh for the refinement case 2 with displacement $\Delta y = 87 \times 10^{-3}\,\mathrm{m}$ at $t = 6.116\,\mathrm{s}$. Cells above the flap are distorted too much by the mesh motion solver, which leads to the flow simulation crashing. These results highlight the need for advanced mesh motion techniques in OpenFOAM.

observed that further refinement of the mesh around the flap causes numerical problems originating from the mesh motion.

To cope with the large number of unknowns for the different refinement cases, the fluid as well as the solid solver are executed in parallel. We configure again a parallel-implicit coupling scheme in preCICE and apply a relative convergence measure of 10^{-4} on the coupling data. As acceleration, we use again a quasi-Newton scheme combined with a QR1 filtering mechanism [46]. Since the interface meshes are considerably larger than in the two-dimensional case, we employ radial-basis-function data mapping with compactly supported basis functions, which results in sparse mapping matrices. Each radial basis function mapping is configured to cover $5-10$ vertices in each radial direction via the corresponding support radius of the basis function.

In Fig. 15, we compare the displacement at the tip of the flap in the center ($z = 0.0$) and the left corner ($z = -0.3$) for three different meshes to the *A2-fine* results by Schott et al. [41]. *A2-fine* refers in the original work to an unfitted and non-moving fluid mesh technique using the finest mesh resolution in order to perform the FSI simulation. The plots indicate a consistent convergence when refining the individual meshes. The overall results are in good agreement with the reference results: we only observe a slightly smaller damping compared to Schott et al. [41].

Since this is the most computationally demanding case of the ones presented, we timed the execution in detail and give an overview[27] in Tab. 5. We simulated the first 0.5 s of each case, coupling again every 0.01 s (i.e., 50 coupling time windows). We can interpret these results from different angles, but we focus here on the performance of the adapter implementation. The performance of the preCICE library and of the coupling methods involved has extensively been documented in the literature [5, 8, 46, 52].

Breaking down the OpenFOAM clockTime into three parts (total time spent in OpenFOAM, time spent exclusively in the adapter, time spent exclusively in preCICE), the adapter only adds minimal overhead to the execution (approximately 1-2% of the total time, depending on the size of the coupling interface), with reading checkpoints taking approximately half of that time. Calls to the preCICE `advance()` method take a significant part of the time in all cases, which can be attributed to the RBF mapping (most accurate and expensive of the available methods [6]). Work in progress on a partition-of-unity RBF mapping[28] has shown to drastically further improve the mapping performance (results not yet published). To simulate the initial 50 coupling time windows of the simulation, preCICE needed on average 2.6 coupling iterations for case 1, 2.7 for case 2, and 2.8 for case 3. As the simulation proceeds, the employed IQN method converges faster.

[27] The files `precice-Fluid-events-summary.log`, `precice-Fluid-iterations.log`, and `fluid.log` in each case directory of the accompanying data archive give more details.

[28] Partition-of-unity feature issue in the preCICE repository: `https://github.com/precice/precice/issues/1273`

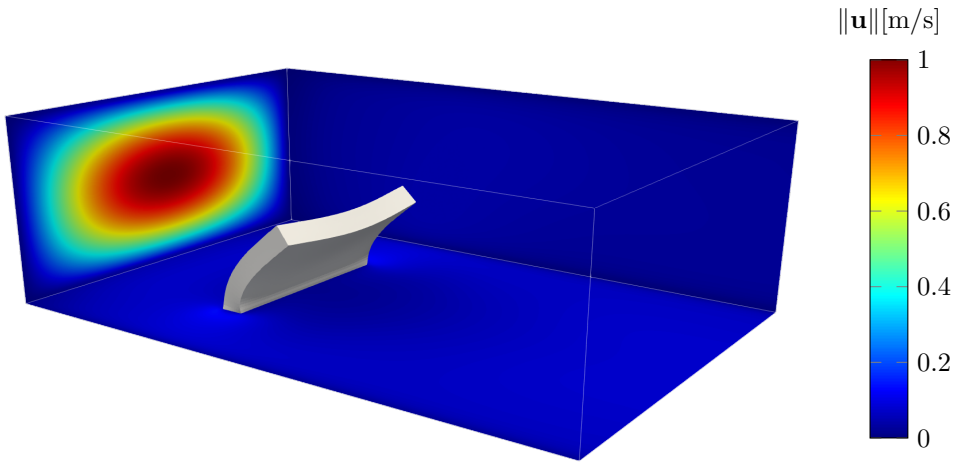


Figure 13. 3D FSI scenario: Simulation setup at $t = 2.5$. A flap is clamped at the bottom of a channel with pulsating parabolic inlet, as described by Schott et al. [41]. The inlet velocity is $\mathbf{u} = (u, 0, 0) \cdot g(t)$, where $u = u_{max}\,(2500/81)\,y\,(y - 0.6)\,(z + 0.6)\,(z - 0.6)$, $u_{max} = 2$, and the time dependency $g(t) = 0.5\,(1 - \cos(\pi t))$, $t \in [0, 10.0]$ with $g(t) = 0\ \forall\ t \in (10.0, 30.0]$. The solid domain is $\Omega_s = [0.5, 0.57] \times [0, 0.35] \times [-0.3, 0.3]$, immersed in the fluid $\Omega_f = [0, 1.8] \times [0, 0.6] \times [-0.6, 0.6]$.

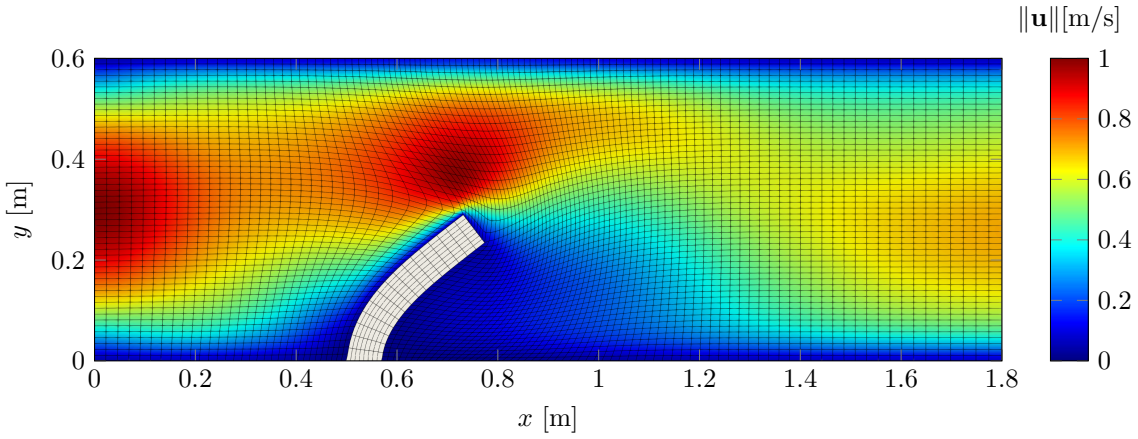


Figure 14. 3D FSI scenario: Cross-section of the computational domain at $z = 0$ and $t = 2.5s$.

case	#cells fluid	#cells solid
coarse	13 710	48
medium	105 960	384
fine	672 000	3 072

Table 4. 3D FSI scenario: Spatial resolution of the coupled simulations. The solid part was discretized using hexahedral $\mathcal{Q}^2$-elements, which leads to a total number of 82 467 degrees of freedom in the structural part for case 3.

8. Community extensions and applications

preCICE is in development already for more than ten years and is currently maintained by three research groups[29], with contributions from a rapidly growing community, and with a support program for sustainable funding[30]. OpenFOAM is currently the solver attracting the most interest in the preCICE

[29]See the About preCICE page: `https://precice.org/about.html`, as well as the preCICE v2 paper [6].
[30]See the support program page: `https://precice.org/community-support-precice.html`

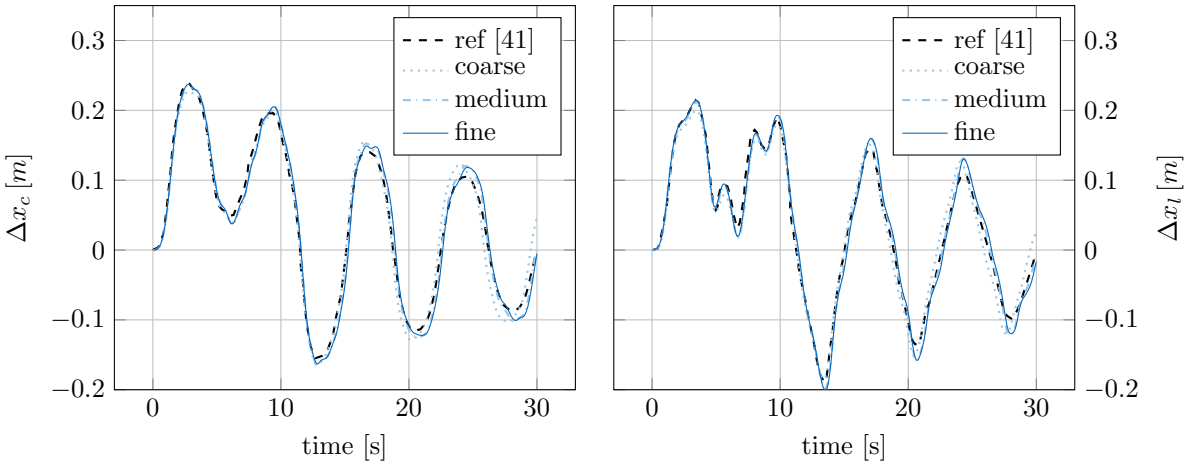


Figure 15. 3D FSI scenario: Displacement of the flexible flap at the center $x_c = (0.5, 0.35, 0.0)$ (left graph) and the left tip $x_l = (0.57, 0.35, -0.3)$ (right graph). Results from Schott et al. [41] refer to the *A2-fine* results presented in the paper. At 28 s, the case 3 results of Δx_l overshoot the reference results by 0.02 m. Note that the reference results have been acquired with different codes, meshes, and numerical setups.

Execution section	Case 1 [s]	%	Case 2 [s]	%	Case 3 [s]	%
OpenFOAM clockTime	62	100.0	115	100.0	250	100.0
reading preciceDict	0.0015	0.002	0.1678	0.146	0.0884	0.035
constructing preCICE	0.1003	0.162	0.4104	0.357	1.1382	0.455
setting up interfaces	0.0007	0.001	0.0005	0.000	0.0009	0.000
setting up checkpointing	0.0008	0.001	0.0007	0.000	0.0021	0.001
writing data	0.1215	0.196	0.4199	0.365	0.9519	0.381
reading data	0.0011	0.002	0.0017	0.002	0.0025	0.001
writing checkpoints	0.0061	0.009	0.0435	0.038	0.1267	0.051
reading checkpoints	0.3046	0.491	1.1181	0.972	3.2986	1.320
`precice.initialize()`	0.2403	0.388	0.9005	0.783	1.0108	0.404
`precice.advance()`	7.6515	12.34	13.107	11.40	52.562	21.03
`precice.finalize()`	0.3256	0.525	0.4710	0.410	0.9753	0.390

Table 5. 3D FSI scenario: runtime analysis for the first 0.5 s of the coarse (Case 1), medium (Case 2) and refined (Case 3) cases described in Tab. 4, executed on the SuperMUC-NG system hosted at LRZ (48 Intel Skylake Xeon Platinum 8174 cores per node). OpenFOAM is using 48 cores in cases 1&2, and 96 cores in case 3. The solid participant is using 48 cores in all cases. The time reported is the average among all ranks, while the "%" column compares to the total time reported by OpenFOAM. Time in the preCICE calls (last three rows) includes time interacting with and/or waiting for the solid participant, as well as data mapping (highest contribution) and IQN acceleration computations. All preCICE events are configured to be executed in synchronous mode, to make runtimes across ranks comparable. Detailed runtimes and iteration logs are available in the case data appendix.

community, as seen by the activity in the preCICE forum[31]. In this context, and while the OpenFOAM-preCICE adapter is being developed in the open (by more than two regular developers), it has been extended in different directions and and has been used in several projects of multiple research groups. This section contains a literature review of the known community extensions and a limited sample of community projects using the adapter, with the purpose of demonstrating the wider capabilities and potential of the code, as well as the impact to the OpenFOAM community. The examples listed here have been identified on Google Scholar based on citations to related previous work [7, 13, 53] (using the

[31]Topic tags in the preCICE forum: `https://precice.discourse.group/tags`

"cited by" feature, and filtering for references specifically using this adapter), as well as on pull requests to the adapter repository, and are categorized according to common use cases. As the history of the adapter is rather recent, some of these projects have only been documented in gray literature (student theses and conference proceedings indexed by Google Scholar, or web pages we have discovered via the preCICE community forum), but we include these here as starting points for future work.

Many of the community extensions have been implemented with only the help of the documentation and the occasional discussions in the preCICE forum, reaffirming the extensibility of the architecture. Since often contributions only come at the end of the respective research project, some of these contributions re-implement or build upon previous contributions. At the moment, 21 out of 110 pull requests on GitHub have been contributed by external collaborators. Of these, six have been merged, three remain open, and 12 have been closed.

8.1. **Extensions for fluid-structure interaction.** Derek Risseeuw contributed the first extensions for fluid-structure interaction, coupling the incompressible solver pimpleFoam [36]. Later extensions contributed by Julian Seuffert on the force computation allowed to also couple compressible solvers, applied on resin transfer molding [54]. Both of these early extensions are described in subsection 6.8.

8.2. **Extensions for multi-phase flow coupling.** Francisco Espinosa [55] extended the basic fluid-fluid coupling module (subsection 6.9) to two-phase flow, coupling interFoam with a shallow-water equation model. Two-phase flow coupling has also been demonstrated by Srivastava et al. [56], coupling SU2 (external aerodynamics) with a structure solver (motion) and interFoam (sloshing). We are currently working in extending and validating the support for flow coupling, including multi-phase flow.

8.3. **Extensions for volume coupling.** Marta Camps Santasmasas first extended the adapter for volume coupling, coupling OpenFOAM in an overlap region with a lattice Boltzmann solver for urban wind flow [57]. In parallel, Stefan Scheiblhofer et al. extended the adapter for thermo-mechanical coupling, exchanging temperature and heat flux also on internal field cells [58]. Different groups have built upon the code of Stefan Scheiblhofer, e.g., for coupling OpenFOAM with the particle solver XDEM [59] or for nuclear fusion reactor simulations [60], while we are working with the community to integrate volume coupling into the main branch of the adapter[32].

8.4. **Further community extensions.** Thomas Enzinger[33] heavily extended the adapter to further separate the coupling physics from the coupling logic, transferring a part of the code into dedicated boundary conditions (externalWallHeatFluxTemperature). This project only focused on conjugate heat transfer and coupled icoReactingMultiphaseInterFoam with laplacianFoam for simulating a multiphase heater.

DHCAE Tools GmbH integrated preCICE into the proprietary graphical interface CastNet[34], providing an interface to configure fluid-structure interaction simulations with OpenFOAM and CalculiX. In this context, the company has also performed independent validations of the adapter for fluid-structure interaction scenarios, including a scenario with two-phase flow.

8.5. **Further application examples.** Apart from the aforementioned extensions, researchers have applied the adapter in a variety of further projects. Caccia et al. [61] simulated the Turek-Hron FSI benchmarks with OpenFOAM and MBDyn, while Julian Schlieus and Louis Gagnon [62] also coupled OpenFOAM and MBDyn for simulating the motion of 2D rotating bodies in a fluid (using a chimera overset grid for the rotation). ojek et al. [63] applied the adapter in a fluid-structure-acoustics interaction scenario. Cars van Otterlo [64] coupled OpenFOAM with Nutils for simulating the turbulent heat transfer in an Axial-Flux Permanent Magnet machine. Max Firmbach [65] coupled OpenFOAM and DUNE for simulating slender wings for electric aircraft, while Yujia Wei and Tahsin Tezdogan coupled OpenFOAM and CalculiX for simulating the hydroelastic behavior of a container ship [66]. Munaf et al. extended and applied the adapter for simulating inductively-coupled plasma wind tunnels with the OpenFOAM-based PATO solver [67]. Finally, the OpenFOAM-based project blastFoam offers a fluid-structure interaction tutorial case for preCICE (coupling blastFoam and CalculiX)[35], while Zhang et al. [68] have also used the adapter for FSI simulations in the context of detonations (coupling OpenFOAM and deal.II).

[32]Central issue for volume coupling support: `https://github.com/precice/openfoam-adapter/issues/229`
[33]OpenFOAM Multiphase Heater Simulation: `https://blog.tefe.cc/post/2020/01-watercooker/`
[34]DHCAE Tools GmbH: preCICE `https://www.dhcae-tools.com/preCICE.html`
[35]blastFoam tutorial using preCICE: `https://github.com/synthetik-technologies/blastfoam/tree/master/tutorials/preCICE/flap`

9. Conclusions and outlook

The OpenFOAM-preCICE adapter enables extending OpenFOAM to partitioned multi-physics simulations via preCICE, a coupling library with a wide ecosystem of supported solvers, and which has been presented by separate publications. In contrast to other approaches, OpenFOAM-preCICE relies on existing (standard or in-house) OpenFOAM solvers, which do not need to be modified or recompiled, but rather coupled at runtime using a function object that encapsulates all calls to the preCICE library. The adapter has been validated for conjugate heat transfer (CHT) and fluid-structure interaction (FSI), while it has initial support for fluid-fluid coupling. For CHT, the adapter was validated using a partitioned heat conduction problem (compared to a single-domain OpenFOAM solution), as well as a flow-over-a-heated-plate scenario [44] (compared to chtMultiRegionFoam). For FSI, the adapter was validated using the Turek-Hron FSI2 and FSI3 benchmarks [40], as well as a 3D pulsating flow over a flexible flap case by Schott et al. [41]. Profiling the simulation of the latter case, the adapter code was also found to have minimal impact on the overall runtime of the 3D pulsating flow FSI simulation compared to the time spent in the solver (approximately 1-2%, depending on the size of the coupling interface), while the coupling converged on average in less than three iterations.

While the adapter provides a robust and flexible framework for coupled simulations, research and development is still active. Current efforts include integrating community contributions into the main development line, extending and validating the support for fluid-fluid coupling, working with the community to support more mesh motion solvers, and establishing a sustainable strategy on maintaining support for a wide range of OpenFOAM versions. A growing community has already used the adapter for several applications and extended it in different directions (e.g., two-phase flow and volume coupling), providing evidence for the versatility of the underlying architecture.

Acknowledgements

Funded by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2075 – 390740016, the European Union's Horizon 2020 research and innovation program under the Marie Sklodowska-Curie grant agreement No 754462, the DFG project preDOM, project number 391150578, and the German Federal Ministry for the Environment, Nature Conservation, Nuclear Safety and Consumer Protection project preCICE-ATHLET, project number 1501593. We acknowledge the support by the Stuttgart Center for Simulation Science (SimTech) and compute time on SuperMUC-NG at the Leibniz Supercomputing Centre of the Bavarian Academy of Sciences and Humanities.

Besides the authors of this paper and the extensions mentioned in Section 8, many more have contributed back to the preCICE OpenFOAM adapter since its first public release on GitHub in 2017. We especially want to thank (in alphabetical order) Philip Cardiff, Jeff Heylmun, Mark Olesen, and everybody else who has contributed to the GitHub repository in form of issues, code, and any kind of feedback.

Author Contributions: Conceptualisation, B.U.; methodology, G.C.; software, G.C. and D.S.; validation, G.C. and D.S.; data curation, G.C. and D.S.; writing—original draft preparation, G.C., D.S, and B.U.; writing—review and editing, G.C., D.S., and B.U.; visualisation, G.C. and D.S.; supervision, B.U.; funding acquisition, B.U. All authors have read and agreed to the published version of the manuscript.

References

[1] D. E. Keyes, L. C. McInnes, C. Woodward, W. Gropp, E. Myra, M. Pernice, J. Bell, J. Brown, A. Clo, J. Connors, E. Constantinescu, D. Estep, K. Evans, C. Farhat, A. Hakim, G. Hammond, G. Hansen, J. Hill, T. Isaac, X. Jiao, K. Jordan, D. Kaushik, E. Kaxiras, A. Koniges, K. Lee, A. Lott, Q. Lu, J. Magerlein, R. Maxwell, M. McCourt, M. Mehl, R. Pawlowski, A. P. Randles, D. Reynolds, B. Rivire, U. Rüde, T. Scheibe, J. Shadid, B. Sheehan, M. Shephard, A. Siegel, B. Smith, X. Tang, C. Wilson, and B. Wohlmuth, "Multiphysics simulations: Challenges and opportunities," *The International Journal of High Performance Computing Applications*, vol. 27, no. 1, pp. 4–83, 2013. [Online]. Available: https://doi.org/10.1177/1094342012468181

[2] J. Slotnick, A. Khodadoust, J. Alonso, and D. Darmofal, "CFD vision 2030 study: A path to revolutionary computational aerosciences," NASA Langley Research Center, Tech. Rep., 2014. [Online]. Available: https://ntrs.nasa.gov/citations/20140003093

[3] X. Liu, D. Furrer, J. Kosters, and J. Holmes, "Vision 2040: a roadmap for integrated, multiscale modeling and simulation of materials and systems," NASA Langley Research Center, Tech. Rep., 2018. [Online]. Available: https://ntrs.nasa.gov/citations/20180002010

[4] P. Cardiff, Z. Tuković, H. Jasak, and A. Ivankovic, "A block-coupled Finite Volume methodology for linear elasticity and unstructured meshes," *Computers & Structures*, vol. 175, pp. 100–122, Oct. 2016. [Online]. Available: https://doi.org/10.1016/j.compstruc.2016.07.004

[5] B. Uekermann, "Partitioned fluid-structure interaction on massively parallel systems," Dissertation, Department of Informatics, Technical University of Munich, Oct. 2016. [Online]. Available: https://mediatum.ub.tum.de/?id=1320661

[6] G. Chourdakis, K. Davis, B. Rodenberg, M. Schulte, F. Simonis, B. Uekermann, G. Abrams, H. Bungartz, L. Cheung Yau, I. Desai, K. Eder, R. Hertrich, F. Lindner, A. Rusch, D. Sashko, D. Schneider, A. Totounferoush, D. Volland, P. Vollmer, and O. Koseomur, "preCICE v2: A sustainable and user-friendly coupling library [version 1; peer review: 2 approved]," *Open Research Europe*, vol. 2, no. 51, 2022. [Online]. Available: https://doi.org/10.12688/openreseurope.14445.1

[7] H.-J. Bungartz, F. Lindner, B. Gatzhammer, M. Mehl, K. Scheufele, A. Shukaev, and B. Uekermann, "preCICE – a fully parallel library for multi-physics surface coupling," *Computers and Fluids*, vol. 141, pp. 250–258, 2016, Advances in Fluid-Structure Interaction. [Online]. Available: https://doi.org/10.1016/j.compfluid.2016.04.003

[8] H.-J. Bungartz, F. Lindner, M. Mehl, K. Scheufele, A. Shukaev, and B. Uekermann, "Partitioned fluid–structure–acoustics interaction on distributed data: Coupling via preCICE," in *Software for Exascale Computing - SPPEXA 2013-2015*. Cham: Springer International Publishing, 2016, pp. 239–266. [Online]. Available: https://doi.org/10.1007/978-3-319-40528-5_11

[9] L. Cheung Yau, "Conjugate heat transfer with the multiphysics coupling library preCICE," Master's thesis, Department of Informatics, Technical University of Munich, Dec. 2016. [Online]. Available: http://www5.in.tum.de/pub/Cheung2016_Thesis.pdf

[10] K. Rave, "Kopplung von OpenFOAM und deal.II Gleichungslösern mit preCICE zur Simulation multiphysikalischer Probleme," Master's thesis, University of Siegen, Jan. 2017.

[11] D. Blom, "FOAM–FSI: Fluid-structure interaction solvers for foam-extend," GitHub repository: https://github.com/davidsblom/FOAM-FSI, 2016, commit 5f43c09.

[12] D. Schneider, "Simulation von Fluid-Struktur-Interaktion mit der Kopplungsbibliothek preCICE," Bachelor's thesis, University of Siegen, 2018.

[13] G. Chourdakis, "A general OpenFOAM adapter for the coupling library preCICE," Master's thesis, Department of Informatics, Technical University of Munich, Oct. 2017. [Online]. Available: https://mediatum.ub.tum.de/1462269

[14] OpenCFD Ltd., "OpenFOAM User Guide," https://www.openfoam.com/documentation/user-guide/, last accessed: 25/07/2022.

[15] H. Marschall, "conjugateHeatFoam," Jun. 2020, training material for the 15th International OpenFOAM Workshop. [Online]. Available: https://bitbucket.org/hmarschall/conjugateheatfoam

[16] Ž. Tuković, A. Karač, P. Cardiff, H. Jasak, and A. Ivanković, "OpenFOAM Finite Volume Solver for Fluid-Solid Interaction," *Transactions of FAMENA*, vol. 42, no. 3, pp. 1–31, Oct. 2018. [Online]. Available: https://doi.org/10.21278/TOF.42301

[17] L. Huang and Y. Li, "Design of the submerged horizontal plate breakwater using a fully coupled hydroelastic approach," *Computer-Aided Civil and Infrastructure Engineering*, vol. 37, no. 7, pp. 915–932, 2022. [Online]. Available: https://doi.org/10.1111/mice.12784

[18] P. Cardiff, A. Karac, P. D. Jaeger, H. Jasak, J. Nagy, A. Ivankovic, and Z. Tukovic, "An open-source finite volume toolbox for solid mechanics and fluid-solid interaction simulations," 2018. [Online]. Available: https://arxiv.org/abs/1808.10736

[19] I. Oliveira, P. Cardiff, C. Baccin, and J. Gasche, "A numerical investigation of the mechanics of intracranial aneurysms walls: Assessing the influence of tissue hyperelastic laws and heterogeneous properties on the stress and stretch fields," *Journal of the Mechanical Behavior of Biomedical Materials*, vol. 136, p. 105498, 2022. [Online]. Available: https://doi.org/10.1016/j.jmbbm.2022.105498

[20] M. Kraposhin and I. Marchevsky, "simpleFsi," training material. [Online]. Available: https://wiki.openfoam.com/Simple_FSI_by_Matvey_Kraposhin

[21] S. Wagner, M. Mnsch, and A. Delgado, "An integrated OpenFOAM membrane fluid-structure interaction solver," *OpenFOAM Journal*, vol. 2, p. 4861, Mar. 2022. [Online]. Available: https://doi.org/10.51560/ofj.v2.45

[22] M. Mehl, B. Uekermann, H. Bijl, D. Blom, B. Gatzhammer, and A. van Zuijlen, "Parallel coupling numerics for partitioned fluid-structure interaction simulations," *Computers & Mathematics with Applications*, vol. 71, no. 4, pp. 869 – 891, 2016. [Online]. Available: https://doi.org/10.1016/j.camwa.2015.12.025

[23] S. Hewitt, L. Margetts, and A. Revell, "Parallel performance of an open source fluid structure interaction application," in *Proceedings of the 25th UKACM Conference on Computational Mechanics. Birmingham*, 2017. [Online]. Available: http://ukacm2017.ukacm.org/wp-content/uploads/2017/04/Compressed-Proceedings-UKACM2017.pdf

[24] S. Hewitt, L. Margetts, A. Revell, P. Pankaj, and F. Levrero-Florencio, "OpenFPCI: A parallel fluidstructure interaction framework," *Computer Physics Communications*, vol. 244, pp. 469–482, 2019. [Online]. Available: https://doi.org/10.1016/j.cpc.2019.05.016

[25] J. Lorentzon, "Fluid-structure interaction (FSI) case study of a cantilever using OpenFOAM and DEAL.II with application to VIV," 2009. [Online]. Available: https://lup.lub.lu.se/student-papers/record/1611353/file/1611440.pdf

[26] J. Herb, "Coupling OpenFOAM with thermo-hydraulic simulation code ATHLET," in *9th OpenFOAM Workshop*, Zagreb, 2014. [Online]. Available: http://openfoam-extend.sourceforge.net/OpenFOAM_Workshops/OFW9_2014_Zagreb/download.html

[27] Q. Huang, A. Abdelmoula, G. Chourdakis, J. Rauleder, and B. Uekermann, "CFD/CSD coupling for an isolated rotor using preCICE," in *14th World Congress on Computational Mechanics (WCCM)*. Paris, France (online): IACM, Feb 2021. [Online]. Available: https://www.scipedia.com/public/Huang_et_al_2021b

[28] F. Duchaine, S. Jaur, D. Poitou, E. Qumerais, G. Staffelbach, T. Morel, and L. Gicquel, "Analysis of high performance conjugate heat transfer with the OpenPALM coupler," *Computational Science & Discovery*, vol. 8, no. 1, p. 015003, 2015. [Online]. Available: https://doi.org/10.1088/1749-4699/8/1/015003

[29] W. Liu, S. M. Longshaw, A. Skillen, D. R. Emerson, C. Valente, and F. Gambioli, "A high-performance open-source solution for multiphase fluid-structure interaction," in *The 31st International Ocean and Polar Engineering Conference*. OnePetro, 2021. [Online]. Available: https://doi.org/10.17736/ijope.2022.jc844

[30] M. Haupt, R. Niesner, R. Unger, and P. Horst, "Coupling techniques for thermal and mechanical fluid-structure-interactions in aeronautics," *PAMM*, vol. 5, no. 1, pp. 19–22, 2005. [Online]. Available: https://doi.org/10.1002/pamm.200510006

[31] M. Müller, M. Haupt, and P. Horst, "Socket-based coupling of OpenFOAM and Abaqus to simulate vertical water entry of rigid and deformable structures," in *6th ESI OpenFOAM User Conference*, Hamburg, 2018. [Online]. Available: https://doi.org/10.13140/RG.2.2.33478.73281

[32] S. Sicklinger, T. Wang, C. Lerch, R. Wüchner, and K.-U. Bletzinger, "EMPIRE: A N-code coupling tool for multiphysic co-simulations with OpenFOAM," in *7th OpenFOAM Workshop*, Darmstadt, Jun 2012. [Online]. Available: https://www.researchgate.net/publication/319544926_EMPIRE_A_N-Code_Coupling_Tool_for_Multiphysic_Co-Simulations_with_OpenFOAM

[33] N. Wirth, P. Bayrasy, B. Landvogt, K. Wolf, F. Cecutti, and T. Lewandowski, *Analysis and Optimization of Flow Around Flexible Wings and Blades Using the Standard Co-simulation Interface MpCCI*. Cham: Springer International Publishing, 2017, pp. 283–321. [Online]. Available: https://doi.org/10.1007/978-3-319-50568-8_15

[34] F. Lindner, B. Uekermann, M. Mehl, and H.-J. Bungartz, "A plug-and-play coupling approach for parallel multi-field simulations," *Computational Mechanics*, vol. 55, pp. 1–11, 12 2014. [Online]. Available: https://doi.org/10.1007/s00466-014-1113-2

[35] J. Degroote, K.-J. Bathe, and J. Vierendeels, "Performance of a new partitioned procedure versus a monolithic procedure in fluid-structure interaction," *Computers & Structures*, vol. 87, no. 11, pp. 793 – 801, 2009, fifth MIT Conference on Computational Fluid and Solid Mechanics. [Online]. Available: https://doi.org/10.1016/j.compstruc.2008.11.013

[36] D. Risseeuw, "Fluid structure interaction modelling of flapping wings," Master's Thesis, Delft University of Technology, 2019. [Online]. Available: http://resolver.tudelft.nl/uuid:70beddde-e870-4c62-9a2f-8758b4e49123

[37] G. Chourdakis, B. Uekermann, G. van Zwieten, and H. van Brummelen, "Coupling OpenFOAM to different solvers, physics, models, and dimensions using preCICE," in *14th OpenFOAM Workshop*, Duisburg, Germany, 2019. [Online]. Available: https://mediatum.ub.tum.de/1515271

[38] Fraunhofer Institute for Algorithms and Scientific Computing SCAI, "MpCCI 4.5.0-1 documentation," https://www.mpcci.de/content/dam/scai/mpcci/documents/MpCCIdoc-4_5_0.pdf, Mar. 2017, part VI Codes Manual. Last accessed: 25/07/2022.

[39] M. Mhlhuer, "Partitioned flow simulations with preCICE and OpenFOAM," Master's Thesis, Technical University of Munich, Dec 2022. [Online]. Available: https://mediatum.ub.tum.de/node?id=1696254

[40] S. Turek and J. Hron, "Proposal for numerical benchmarking of fluid-structure interaction between an elastic object and laminar incompressible flow," in *Fluid-Structure Interaction*, H.-J. Bungartz and M. Schäfer, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 371–385. [Online]. Available: https://doi.org/10.1007/3-540-34596-5_15

[41] B. Schott, C. Ager, and W. A. Wall, "Monolithic cut finite element-based approaches for fluid-structure interaction," *International Journal for Numerical Methods in Engineering*, vol. 119, no. 8, pp. 757–796, 2019. [Online]. Available: https://doi.org/10.1002/nme.6072

[42] B. Gatzhammer, "Efficient and flexible partitioned simulation of fluid-structure interactions," Dissertation, Technical University of Munich, Munich, 2014. [Online]. Available: https://www5.in.tum.de/pub/Gatzhammer2014_preCICE.pdf

[43] H. P. Langtangen and A. Logg, *Solving PDEs in Python - The FEniCS tutorial I*. Cham: Springer International Publishing, 2016. [Online]. Available: https://doi.org/10.1007/978-3-319-52462-7

[44] M. Vynnycky, S. Kimura, K. Kanev, and I. Pop, "Forced convection heat transfer from a flat plate: the conjugate problem," *International Journal of Heat and Mass Transfer*, vol. 41, no. 1, pp. 45 – 59, 1998. [Online]. Available: https://doi.org/10.1016/S0017-9310(97)00113-0

[45] D. Arndt, N. Fehn, G. Kanschat, K. Kormann, M. Kronbichler, P. Munch, W. A. Wall, and J. Witte, "ExaDG: High-order discontinuous Galerkin for the exa-scale," in *Software for Exascale Computing - SPPEXA 2016-2019*, H.-J. Bungartz, S. Reiz, B. Uekermann, P. Neumann, and W. E. Nagel, Eds. Cham: Springer International Publishing, 2020, pp. 189–224. [Online]. Available: https://doi.org/10.1007/978-3-030-47956-5_8

[46] R. Haelterman, A. Bogaers, K. Scheufele, B. Uekermann, and M. Mehl, "Improving the performance of the partitioned QN-ILS procedure for fluidstructure interaction problems: Filtering," *Computers & Structures*, vol. 171, pp. 9–17, 2016. [Online]. Available: https://doi.org/10.1016/j.compstruc.2016.04.001

[47] A. Shamanskiy and B. Simeon, "Mesh moving techniques in fluid-structure interaction: robustness, accumulated distortion and computational efficiency," *Computational Mechanics*, vol. 67, no. 2, pp. 583–600, 2021. [Online]. Available: https://doi.org/10.1007/s00466-020-01950-x

[48] H. Jasak and Ž. Tuković, "Dynamic mesh handling in openfoam applied to fluid-structure interaction simulations," in *Proceedings of the V European Conference on Computational Fluid Dynamics ECCOMAS CFD 2010*, 2010. [Online]. Available: http://repozitorij.fsb.hr/id/eprint/4432

[49] D. Arndt, W. Bangerth, B. Blais, M. Fehling, R. Gassmöller, T. Heister, L. Heltai, U. Köcher, M. Kronbichler, M. Maier, P. Munch, J.-P. Pelteret, S. Proell, K. Simon, B. Turcksin, D. Wells, and J. Zhang, "The `deal.II` library, version 9.3," *Journal of Numerical Mathematics*, vol. 29, no. 3, pp. 171–186, 2021. [Online]. Available: https://doi.org/10.1515/jnma-2021-0081

[50] D. Davydov, J.-P. Pelteret, D. Arndt, M. Kronbichler, and P. Steinmann, "A matrix-free approach for finite-strain hyperelastic problems using geometric multigrid," *International Journal for Numerical Methods in Engineering*, vol. 121, no. 13, pp. 2874–2895, 2020. [Online]. Available: https://doi.org/10.1002/nme.6336

[51] P. Wriggers, *Nonlinear Finite Element Methods*. Springer Berlin Heidelberg, 2008, ch. 6, pp. 205–254. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-71001-1

[52] A. Totounferoush, F. Simonis, B. Uekermann, and M. Schulte, "Efficient and scalable initialization of partitioned coupled simulations with precice," *Algorithms*, vol. 14, no. 6, 2021. [Online]. Available: https://doi.org/1999-4893/14/6/166

[53] B. Uekermann, H.-J. Bungartz, L. Cheung Yau, G. Chourdakis, and A. Rusch, "Official preCICE adapters for standard open-source solvers," in *Proceedings of the 7th GACM Colloquium on Computational Mechanics for Young Scientists from Academia*, Oct. 2017.

[54] J. Seuffert, L. Kärger, G. Chourdakis, B. Uekermann, and F. Henning, "Fluid structure interaction during the resin transfer molding (RTM) manufacturing process for continuous fiber reinforced composites," in *ECCOMAS COUPLED*, vol. 2019, 2019. [Online]. Available: https://congress.cimne.com/coUPLED2019/admin/files/fileabstract/a17.pdf

[55] F. J. Espinosa Pelaez, "A flexible approach to 2D-3D coupling of a shallow-water equation solver to OpenFOAM," Master's thesis, Technical University of Munich, 2020. [Online]. Available: https://mediatum.ub.tum.de/1577072

[56] S. Srivastava, M. Damodaran, and B. C. Khoo, *A Computational Framework for Assessment of Fuel Sloshing Effects on Transonic Wing Flutter Characteristics*. [Online]. Available: https://doi.org/10.2514/6.2019-1527

[57] M. C. Santasmasas, "Hybrid GPU/CPU Navier-Stokes lattice Boltzmann method for urban wind flow," Ph.D. dissertation, The University of Manchester (United Kingdom), 2021. [Online]. Available: https://www.research.manchester.ac.uk/portal/files/194688535/FULL_TEXT.PDF

[58] S. Scheiblhofer, S. Jäger, and A. M. Horr, "Coupling fem and cfd solvers for continuous casting process simulation using preCICE," in *ECCOMAS COUPLED*. CIMNE, 2019, pp. 23–32. [Online]. Available: http://hdl.handle.net/2117/189920

[59] P. Adhav, X. Besseron, A. Rousset, and B. Peters, "AWJC nozzle simulation by 6-way coupling of DEM+CFD+FEM using preCICE coupling library," in *ECCOMAS COUPLED*, 2021. [Online]. Available: http://hdl.handle.net/10993/48725

[60] A. Sircar, J. Bae, E. Peterson, J. Solberg, and V. Badalasi, "FERMI: A multi-physics simulation environment for fusion reactor blanket," in *NURETH-19*, 2022. [Online]. Available: https://www.osti.gov/servlets/purl/1860756

[61] C. Caccia and P. Masarati, "Coupling multi-body and fluid dynamics analisys with preCICE and MBDyn," in *ECCOMAS COUPLED 2021*, 06 2021. [Online]. Available: https://doi.org/10.23967/coupled.2021.014

[62] J. Schließus and L. Gagnon, "Parallel fluid-structure simulation of cycloidal rotors," in *HPC ASIA 2022*, 2022. [Online]. Available: http://louisgagnon.com/articlesScientifiques/Schliessus2022_MMCP22.pdf

[63] P. Lojek, I. Czajka, A. Golas, and K. Suder-Debska, "Influence of the elastic cavity walls on cavity flow noise," *Vibrations in Physical Systems*, vol. 32, p. 2021, 11 2021. [Online]. Available: https://doi.org/10.21008/j.0860-6897.2021.1.09

[64] C. van Otterlo, "Numerical coupling between the finite volume method and the finite element method for modeling the turbulent heat transfer in an AFPM machine," Master's thesis, Eindhoven University of Technology, 2021. [Online]. Available: https://research.tue.nl/en/studentTheses/numerical-coupling-between-the-finite-volume-method-and-the-finit

[65] M. Firmbach, "Aeroelastic simulation of slender wings for electric aircraft: a partitioned approach with DUNE and preCICE," Master's thesis, Technical University of Munich, 2020. [Online]. Available: https://mediatum.ub.tum.de/1609293

[66] Y. Wei and T. Tezdogan, "A fluid-structure interaction model on the hydroelastic analysis of a container ship using preCICE," in *41st International Conference on Ocean, Offshore and Arctic Engineering*, 2022. [Online]. Available: https://strathprints.strath.ac.uk/81106/

[67] A. Munafò, R. Chiodi, S. Kumar, V. Le Maout, K. A. Stephani, F. Panerai, D. J. Bodony, and M. Panesi, "A multi-physics modeling framework for inductively coupled plasma wind tunnels," in *AIAA SCITECH 2022 Forum*, 2022, p. 1011.

[68] S. Zhang, X.-W. Guo, C. Li, Y. Liu, S. Fan, R. Zhao, and C. Yang, "A large scale parallel fluid-structure interaction computing platform for simulating structural responses to a detonation shock," *Software: Practice and Experience*, 2021. [Online]. Available: https://doi.org/10.1002/spe.3051