The 9th European Congress on Computational Methods in Applied Sciences and Engineering
ECCOMAS Congress 2024
3–7 June 2024, Lisboa, Portugal

DEVELOPMENT OF A COUPLING METHODOLOGY BETWEEN CFD CODES

Giacomo Barbi[1], Antonio Cervone[1], Federico Giangolini[1], Sandro Manservisi[1] and Lucia Sirotti[1]

[1] Laboratory of Montecuccolino, DIN, Alma Mater Studiorum University of Bologna, via dei Colli 16, 40136 Bologna (BO), Italy
email: a.cervone@unibo.it

Key words: Coupled Problem, Computational Fluid Dynamics, Turbulent Flow, Heat Transfer

Summary. This paper explores the integration of multiple Computational Fluid Dynamics (CFD) codes within a unified numerical platform to enable simulations of coupled multiphysics and multiscale problems. The objective is to develop a versatile and robust tool capable of simulating complex systems encompassing multiple regions with different, interconnected physical phenomena.

The coupling of these codes is achieved through the external library MEDCoupling. This library has been developed to provide algorithms for communications between different codes directly in memory (i.e., without using external files), ranging from a standard data file format, field management over computational domains, and interpolation schemes.

The platform is tested by coupling the in-house finite element code FEMuS with OpenFOAM, an open-source finite volume code. Both packages can simulate complex engineering problems such as fluid-structure interaction, heat transfer, and turbulent flow phenomena.

1 INTRODUCTION

Over the last few decades, several efforts have been made to model multiscale and multiphysics systems with innovative computational tools [1, 2, 3]. In computational fluid dynamics (CFD), achieving accuracy and efficiency poses significant challenges, especially when dealing with complex systems, spurring interest in employing multiscale and multiphysics numerical tools for conducting complex and realistic simulations. The comprehensive evaluation of systems necessitates modeling efforts across various scales and interactions of different components, necessitating the development of numerical tools capable of analyzing phenomena across multiple scales and physics.

The numerical and computational scientific community has developed numerous codes addressing varying engineering problems, spanning physics, chemistry, biology, and mathematics. Central to tackling these sophisticated issues is high-performance computing (HPC), which leverages additional computational power to address complex problems, and open-source software that enables the possibility to change the code and add desired functionalities. However, simulating highly intricate systems remains challenging due to the differing scales of phenomena. Existing codes generally cater to a specific problem type: as an example, OpenFOAM, TrioCFD, and code_Saturne specialize in fluid dynamics by solving the Navier-Stokes equations

in various forms, be it a finite volume or finite element discretization. Similarly, for thermomechanical problems, we can cite codes like Code-Aster or TFEL/MFront. Finally, some generic PDE solvers, such as libMesh, Deal-II, or FEniCS, are also available.

Addressing the simulation of highly complex problems requires the capability to model different physics from various domains. However, no single code can handle the complete complexity of any given physics phenomenon. Two strategies have emerged: developing a new numerical code encompassing all relevant physical phenomena (monolithic approach) or coupling existing, validated codes, enhancing simulation capabilities by leveraging their strengths. The latter, usually called code-coupling, proves beneficial for simulating tightly coupled physics phenomena by enabling the integration of multiple codes to exploit the features of different codes without the need to develop a new tool. This strategy requires efficient data exchange between codes, especially in the HPC framework. Therefore, reliance on external file transfers should be avoided.

This paper showcases a coupling strategy utilizing the open-source MED and MEDCoupling libraries to connect the FEMuS [4] code with OpenFOAM [5, 6] software. The paper highlights two multiphysics examples to illustrate the code-coupling methodology, whereby physical output variables from one code serve as input for another and vice versa. The paper first introduces the computational framework and the codes employed. It then elaborates on the coupling strategy and the numerical algorithm, concluding with a discussion of two numerical examples of code coupling between FEMuS and OpenFOAM — one involving the exchange of volumetric fields and the other coupling boundary conditions.

2 NUMERICAL ALGORITHM

In this section, we present the software that we leverage and the algorithms that we have developed to perform a generic coupled application.

2.1 Computational software

In this paper, we present a tool named CoCoA (Collaborative Coupling Applications) aimed at implementing a suitable code coupling strategy for existing open-source software in the CFD domain. The tool is available as open-source software at `https://github.com/capitalaslash/cocoa.git` [7]. This tool leverages the numeric platform developed at the University of Bologna [8] for multiphysics and multiscale applications, as well as other open-source software, principally in the CFD domain, such as OpenFOAM or the in-house library ProXPDE [9]. The platform models several physical PDE (Partial Differential Equations) using both FEM (finite element method) and FVM (finite volume method) for modeling fluids and solids [10].

The platform also leverages data entry for input/output using CAD/Mesh generators and visualization/postprocessing with tools from the Salome computing platform [11, 12], i.e., Paraview [13, 14]. In particular, the input/output data can be handled by the MEDCoupling library, which also requires the MEDFile and HDF5 libraries.

2.2 Coupling strategy

The CoCoA library implements a hub-and-spoke approach to code-coupling, in contrast to the point-to-point model. Figure 1 shows graphically the two approaches, with c_i being a code that we want to couple and the arrows representing a connection to another code. In the point-to-point framework, in order to connect a code to each other we need to develop a separate

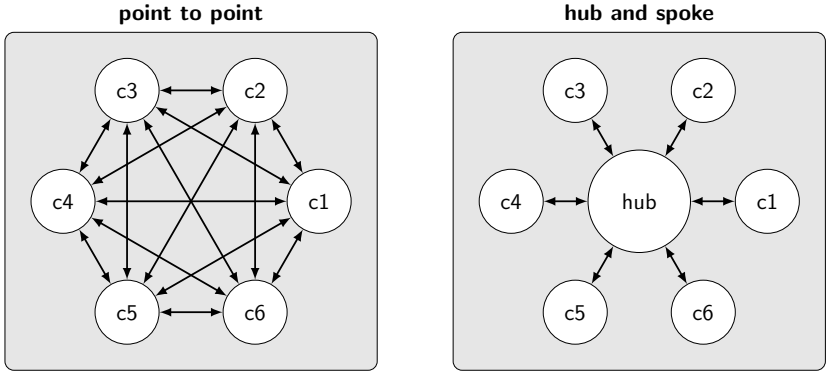

Figure 1: point-to-point (left) and hub-and-spoke (right) coupling approaches.

coupling algorithm for each pair of codes. On the other hand, with the hub-and-spoke, each code needs only to be capable of communicating with the central hub that manages the exchange with all the other codes. In order to do so, each interface between code and hub must rely on a suitable standard that can be universally adopted by the code involved. In particular, in the CFD domain, a common ground is generally represented by a computational domain (the mesh) and a number of fields that live on it, together with their characteristics. The MEDCoupling library from the Salome platform offers these basic tools with fast direct-in-memory exchange and the possibility to interpolate generic meshes with different discretizations, ready for HPC applications.

On top of it, the CoCoa library implements generic models for meshes, fields, coupling interfaces and computational problems that can pick and match from all the computational libraries configured to interface with it. An example of an application that solves separately the fluid problem with the Navier-Stokes equations and the energy problem with the heat equation is presented in Algorithm 1.

The code snippet is presented in Python, but the library is written in C++ and offers Python bindings to easily define the coupled application. The coupling library dispatches the connected libraries that perform the required calculations. The coupling interface defines a projection algorithm from the source mesh to the target mesh and manages the field exchange and synchronization.

3 NUMERICAL RESULTS

In our work, we illustrate the capabilities of the proposed algorithm through two numerical examples, each focusing on different aspects of data transfer between numerical codes: volume and boundary field transfer. Both these examples were specifically chosen to test the coupling methodology and to highlight its robustness and accuracy.

3.1 Buoyant driven cavity

This section presents the results of the code coupling procedure between volume fields, using literature reference data for validation [15, 16, 17, 18, 19]. The simulation involves a two-dimensional analysis of a Newtonian, incompressible fluid within a square cavity, where the

Algorithm 1 Code setup for a buoyant cavity.

```
import pycocoa

# define a Navier-Stokes problem using the OpenFOAM library
pNS = pycocoa.ProblemOpenFOAMNS()
pNS.setup(config_file="buoyant_ns.yaml")

# define a Heat equation problem using the ProXPDE library
pHeat = pycocoa.ProblemProXPDEHeat()
pHeat.setup(config_file="buoyant_heat.yaml")

# define a coupling manager for the Heat -> NS exchange
couplingHeatToNS = pycocoa.CouplingMED()
couplingHeatToNS.setup(problem_src=pHeat, problem_tgt=pNS)

# define another coupling manager for the NS -> Heat exchange
couplingNSToHeat = pycocoa.CouplingMED()
couplingNSToHeat.setup(problem_src=pNS, problem_tgt=pHeat)

# run the simulation until both problems have been completed
while pNS.run() or pHeat.run():

    # exchange the temperature field
    couplingHeatToNS.project("T")
    # compute the next time step size
    pNS.advance()
    # solve for the new time step
    pNS.solve()

    # exchange the velocity field
    couplingNSToHeat.project("vel")
    # compute the next time step size
    pHeat.advance()
    # solve for the new time step
    pHeat.solve()
```

Navier-Stokes and temperature equations are coupled through the buoyancy term. Thus, we have

$$\begin{aligned} &\nabla \cdot u = 0\,, \\ &\frac{\partial u}{\partial t} + u \cdot \nabla u = \frac{p}{\rho} + \nu \Delta u + g\beta(T - T_0)\,, \\ &\frac{\partial T}{\partial t} + u \cdot \nabla T = \alpha \Delta T, \end{aligned} \tag{1}$$

where ν is the kinematic viscosity, ρ the density, β the coefficient of thermal expansion, α the thermal diffusivity, and T_0 the reference temperature.

The simulation domain is reported in Figure 2, where the boundary conditions for the energy equation are shown. A temperature difference is imposed on two opposite edges, creating an hot and a cold wall, while homogeneous Neumann conditions are imposed on the other edges. Additionally, no-slip boundary conditions are applied to the velocity field along all boundary edges.

In order to validate the coupling algorithm, two different cases are presented: in the first case (c_1), FEMuS solves for the temperature field while OpenFOAM handles the velocity field; in the second case (c_2), this setup is reversed. The results from these coupled cases are then compared with the monolithic solutions obtained using FEMuS (F) and OpenFOAM (OF), as

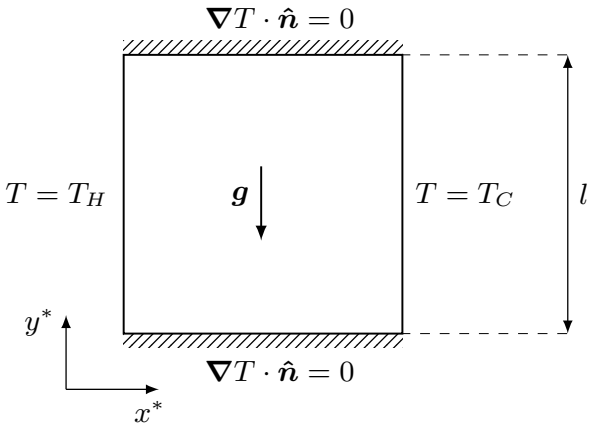

Figure 2: Geometry of the buoyant cavity problem with boundary conditions for the temperature field.

well as with reference data from the literature [15].

Following Algorithm 1, we briefly outline the procedure used for coupling applications that involve volume data transfer for cases c_1 and c_2. In the scenario c_1, Code 1 refers to FEMuS, and Code 2 refers to OpenFOAM, in c_2 the codes are reversed. The coupled applications start by initializing the volume MED meshes and setting up the volumetric temperature and velocity MED fields. Once the time loop starts, Code 1 solves the temperature equation described in (1). The temperature solution is then extracted from Code 1, and the corresponding MED field is updated. The temperature solution is interpolated using MED routines over the volumetric mesh of Code 2 to create the target MED field. The interpolated temperature is then used in Code 2 to compute the buoyancy term within the Navier-Stokes equation, and then Code 2 solves for the velocity field. The velocity solution can be extracted from Code 2, interpolated over the mesh of Code 1 and set into the velocity field of Code 1. In the next time iteration, Code 1 uses this updated velocity field in the advection term of the temperature equation.

3.2 Simulation results

The numerical tests have been performed for different values of the Rayleigh number, defined as $Ra = g\rho\beta L^3(T_H - T_C)/(\nu\alpha)$, where L is the reference length of the domain. The buoyant-driven cavity problem solution depends significantly on the value of the Ra number. Below a critical value Ra_c, the temperature gradient stays aligned with the direction from the cold wall to the hot wall, while when $Ra > Ra_c$ the gradient rotates by 90 degrees. We tested the coupling application using Ra numbers ranging from 10^3 to 10^6, but this section presents the results for the specific cases of $Ra = 10^3$ and $Ra = 10^5$.

In Figure 3, the non-dimensional velocity components and the non-dimensional temperature are shown for the four cases (F, OF, c_1, c_2), and the two considered Rayleigh numbers. The non-dimensional fields are defined by

$$x^* = \frac{x}{L}\,, \quad u^* = \frac{u\,L}{\alpha}\,, \quad \Theta = \frac{T - T_C}{\Delta T}\,, \tag{2}$$

where T_C is the fixed temperature on the cold wall. The numerical fields resulting from the computation have been compared with literature data from [15], marked with circles. In particular,

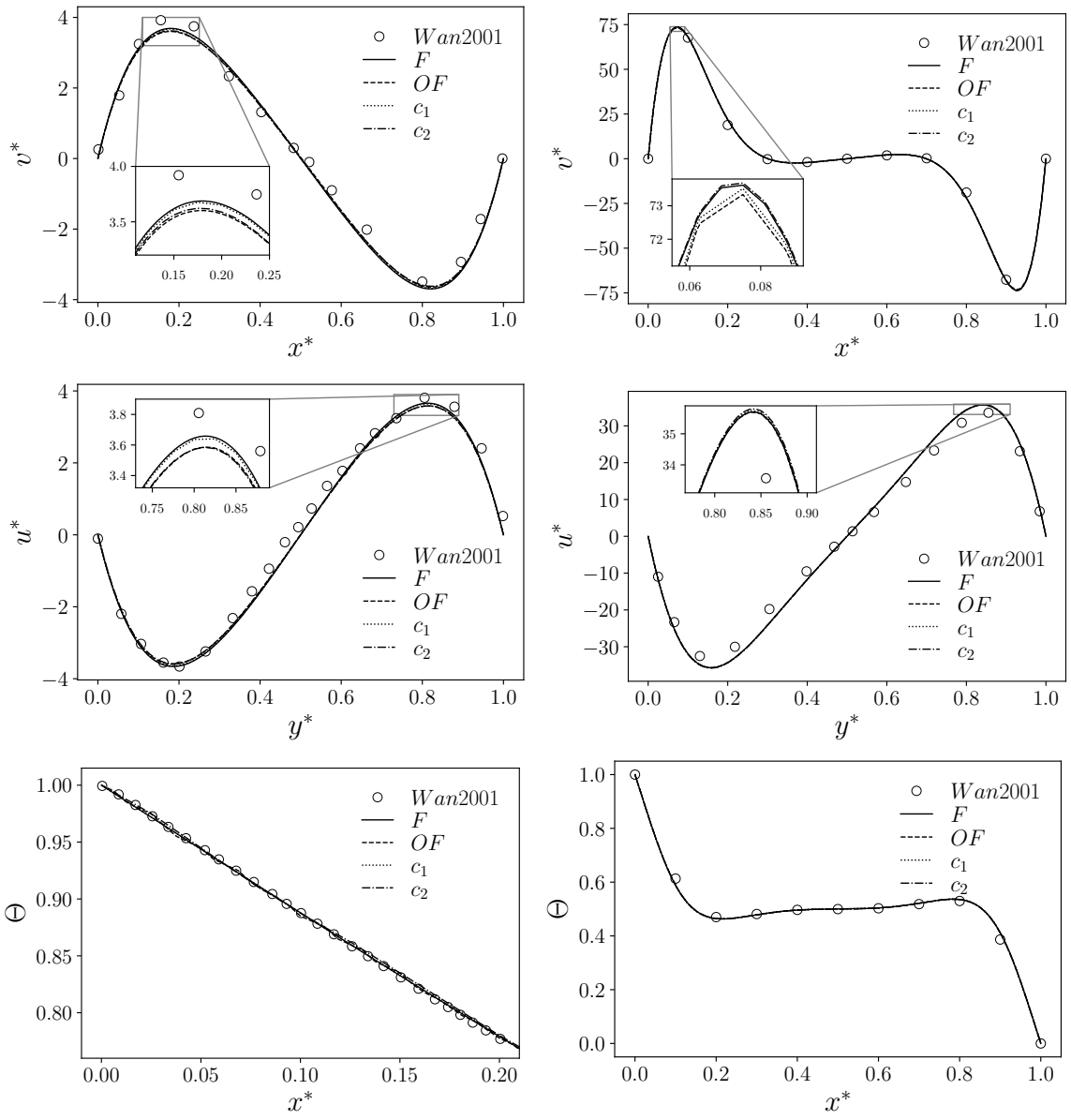

Figure 3: Non-dimensional components u^* (at $x^* = 0.5$, top) and v^* (at $y^* = 0.5$, middle) and non-dimensional temperature Θ (at $y^* = 0.5$, bottom) for the four types of simulations (F, OF, c_1 and c_2) with a comparison with literature data from [15] (circular markers). Case with $Ra = 10^3$ on the left and $Ra = 10^5$ on the right.

these plots illustrate the behavior of the variables at certain points within the domain: the u^* component is plotted along the line at $x^* = 0.5$, while the v^* component and the temperature Θ are represented along the line $y^* = 0.5$. We clarify that the Θ results for the case $Ra = 10^3$ are plotted within the restricted domain $x^* \in [0, 0.2]$, according to the availability of the literature data.

As observed in Figure 3, a good agreement with reference data in [15] is present for every case and every type of simulation, including both coupled algorithms. A zoomed-in view of the plots is included to provide a clearer comparison, focusing on the regions near the maximum and minimum velocity components. This closer view highlights minor differences between the four simulations and the literature results.

3.3 Conjugate heat transfer (CHT)

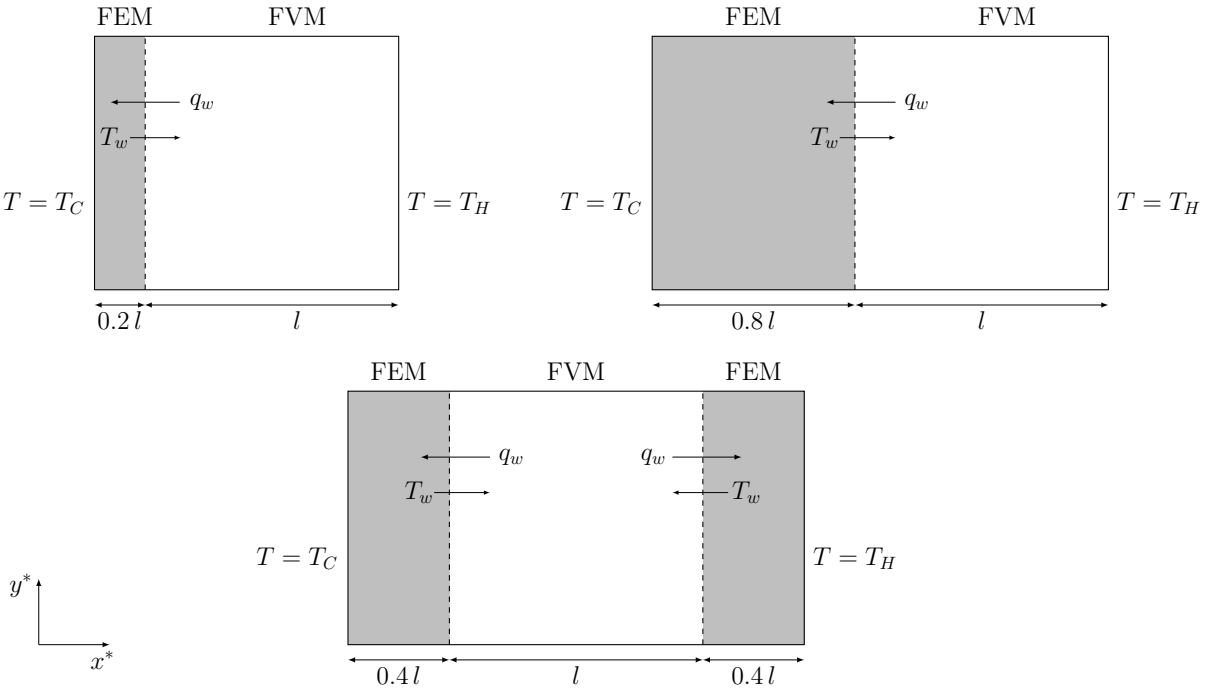

Figure 4: Geometrical configurations of the CHT problem: on the left the domain with the solid wall thickness equal to t_1, on the right equal to t_2 and on the bottom equal to t_3.

In this section, we introduce an application designed to test boundary data transfer between the two CFD codes, FEMuS and OpenFOAM. We focus on a Conjugate Heat Transfer (CHT) problem, which is highly relevant in various scientific and engineering fields, including solar heating systems [20], heat exchangers [21], and nuclear energy production [22].

The conjugate heat transfer phenomenon involves thermal interaction across an interface between two regions, a solid and a fluid. The solid domain is modeled as a two-dimensional

			t_1			
Ra	K_1	[23]	K_2	[23]	K_3	[23]
10^3	0.332	0.335	0.898	0.890	1.080	1.080
			t_2			
Ra	K_1	[23]	K_2	[23]	K_3	[23]
10^5	0.118	0.117	0.850	0.851	2.556	2.578
			t_3			
Ra	K_1	[23]	K_2	[23]	K_3	[23]
10^5	0.117	0.117	0.859	0.855	2.575	2.586

Table 1: Average Nusselt number with different conductivity ratios K, varying the Ra number and wall thickness t, compared to results from [23].

isotropic material with constant properties where only the temperature equation is solved

$$\frac{\partial T}{\partial t} = \alpha \Delta T \,. \tag{3}$$

Here, α is the thermal diffusivity, which is defined as $\alpha = k/(\rho c_p)$, where k is the thermal conductivity and c_p is the thermal capacity of the solid domain. Both momentum and temperature equations are solved within the fluid domain following the system of equations described in (1).

As illustrated in Figure 4, the wall heat flux and the wall temperature are exchanged at the interface between the solid and fluid regions. The fluid domain uses a non-homogeneous Dirichlet boundary condition based on the temperature from the solid, T_w, while the solid domain uses a non-homogeneous Neumann boundary condition determined by the heat flux from the fluid, q_w. In addition, both solid and fluid have adiabatic top and bottom walls. The other boundary conditions are specified in Figure 4.

Similar to the volume data transfer application, we describe the algorithm used for coupling applications involving boundary data transfer. In this scenario, the FEM code, FEMuS, solves the solid domain problem, while the fluid region is simulated by the FVM code, OpenFOAM. After initializing the boundary MED meshes and setting up the boundary MED fields, OpenFOAM solves the system of equations as described in (1). The wall heat flux at the interface, q_w, is computed and extracted from OpenFOAM, and the corresponding MED field is updated. This field is then interpolated over the FEMuS interface mesh using MED routines, creating the target MED field. The temperature equation (3) is solved by FEMuS using the interpolated flux as a non-uniform Neumann boundary condition over the interface. Once the temperature field is solved, the T_w can be extracted from FEMuS, interpolated over the boundary mesh of OpenFOAM, and set into the corresponding OpenFOAM boundary field as a non-uniform Dirichlet boundary condition.

3.4 Simulation results

The fluid region is modeled as a square cavity with dimensions $l \times l$, and the solid domain has been modeled considering different geometric configurations. The physical and geometric setup is based on [23] to validate the results with literature data. Here, the authors have studied different configurations, changing the Pr number, the Ra number, the conductivity ratio $K = k_s/k_f$ (s

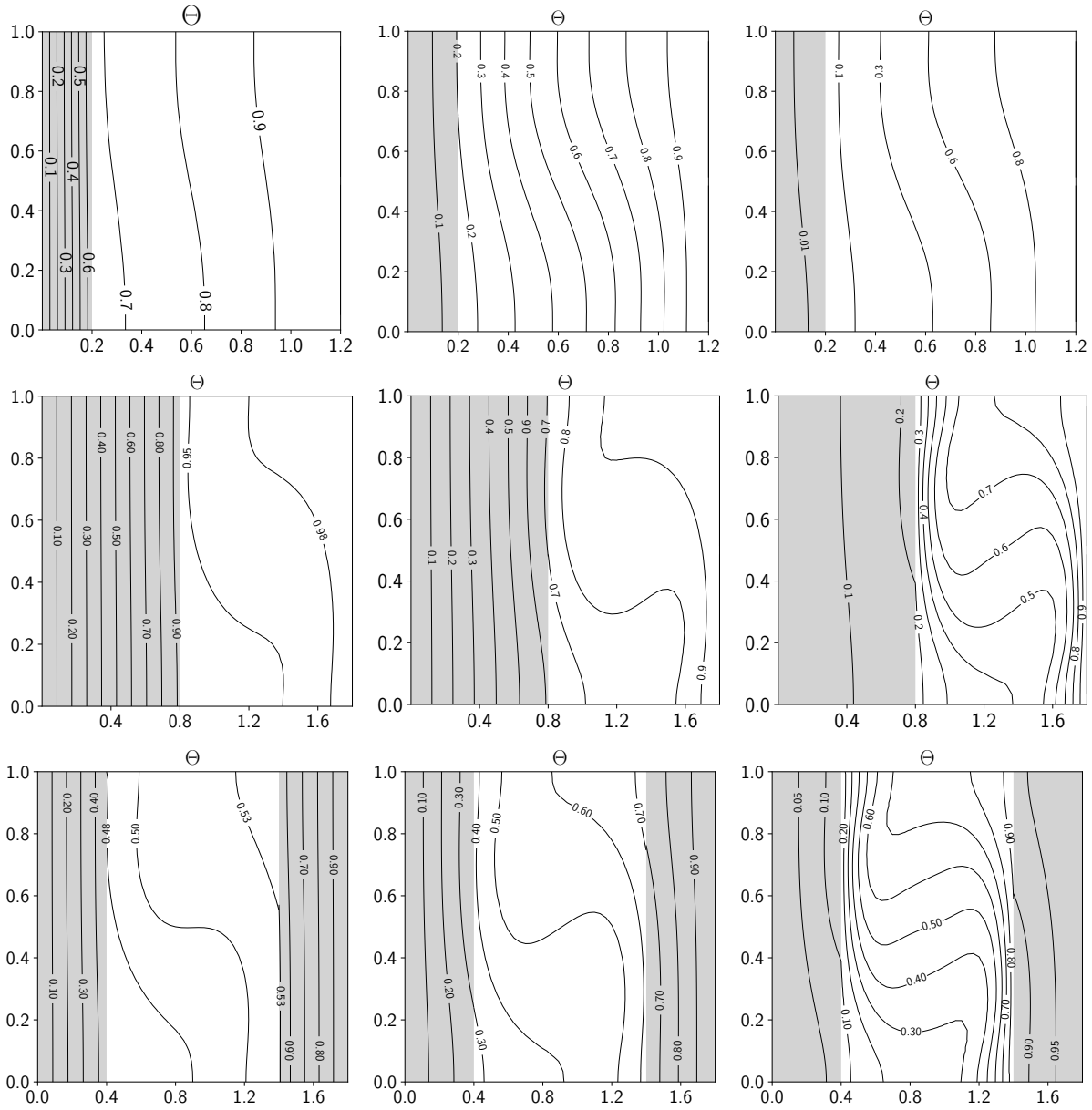

Figure 5: Contour of non-dimensional temperature Θ for $K = 0.1, 1, 10$ (from left to right). Simulations with solid thickness t_1 and $Ra = 10^3$ (top), t_2 and $Ra = 10^5$ (middle), t_3 and $Ra = 10^5$ (bottom).

and f refer to solid and fluid regions, respectively), the solid wall thickness t, and its geometrical position (hot side or cold side). In our work, the Pr number has been fixed to 0.015, while the Ra number has been considered equal to 10^3 and 10^5. We examined three different values of the thermal conductivity ratio, $K_1 = 0.1$, $K_2 = 1$ and $K_3 = 10$. In addition, the solid region is characterized by three different layouts: one with a thickness of $t_1 = 0.2l$, another with a thickness of $t_2 = 0.8l$, both located on the cold side, and a third one consisting of two solid regions on the two opposite sides of the cavity. In this case, each domain has a thickness of $0.4l$ resulting in $0.8l$ total length. We refer to this third case as having a thickness of $t_3 = 0.4l \times 2$. The physical domains of the simulations are shown in the schematic representation in Figure 4.

Figure 5 shows the contour plots of the non-dimensional temperature Θ for the simulated cases that can be compared with [23]. It is evident that as the conductivity ratio K decreases, the temperature contours become compressed towards the solid region. When $K < 1$, the temperature gradient is concentrated in the solid region, while for values of $K > 1$, the temperature gradient is primarily located in the fluid region. On the other hand, increasing the Rayleigh number causes the temperature contours to stretch. With a higher Rayleigh number, the temperature distribution moves from stratification to the formation of a recirculation cell, as expected. These considerations can be applied to all three geometric configurations with solid wall thicknesses t_1, t_2, and t_3.

Table 1 displays the average Nusselt number $\overline{Nu_l}$ on the interface boundary for the three simulations reported here. This table also includes a comparison with the corresponding values in [23]. The results demonstrate a good agreement with the literature data across all simulations. The comparison highlights the consistency of our simulations with previously published results, highlighting the reliability and accuracy of the numerical methods employed.

4 CONCLUSIONS

The paper presents a novel algorithm to couple existing state-of-the-art open-source codes to perform multiphysics and multiscale applications, with particular attention to being ready for High-Performance Computing. The coupling strategy is implemented in the CoCoA library, which leverages existing CFD codes such as OpenFOAM, FEMuS and ProXPDE for the physics modeling and the MEDCoupling library for information exchange between the codes.

The coupling algorithm has been demonstrated in two typical applications that demonstrate volume and boundary coupling, respectively. Both simulations show that the coupling methodology is reliable and accurate when compared to monolithic codes. Future developments will be targeted at extending the range of applications that can be simulated with this approach, such as fluid-structure interaction, turbulence modeling, etc.

REFERENCES

[1] Drikakis, D., Frank, M., and Tabor, G. 2019. Multiscale computational fluid dynamics. Energies, 12(17), 3272. http://dx.doi.org/10.3390/en12173272

[2] Groen, D., Zasada, S. J., and Coveney, P. V. 2013. Survey of multiscale and multiphysics applications and communities. Computing in Science & Engineering, 16(2), 34–43. https://doi.org/10.1109/MCSE.2013.47

[3] Cordero, M. E., at al. 2018. CFD Modelling of Coupled Multiphysics-Multiscale Engineering Cases. In Comput. Fluid Dyn.-Basic Instruments Appl. Sci. https://doi.org/10.5772/intechopen.70562

[4] Chierici, A., et al. 2021. FEMuS-platform: a numerical platform for multiscale and multi-physics code coupling. In 9th edition of the International Conference on Computational Methods for Coupled Problems in Science and Engineering (Coupled Problems 2021). https://doi.org/10.23967/coupled.2021.027

[5] Weller, H. G., Tabor, G., Jasak, H., Fureby, C.. 1998. A tensorial approach to computational continuum mechanics using object-oriented techniques. Computers in Physics, VOL. 12, NO. 6, NOV/DEC 1998. https://doi.org/10.1063/1.168744

[6] Greenshields, C., and Weller, H. 2022. Notes on Computational Fluid Dynamics: General Principles. CFD Direct Ltd, Reading, UK.

[7] `https://github.com/capitalaslash/cocoa.git`

[8] `https://github.com/FemusPlatform/NumericPlatform.git`

[9] `https://github.com/capitalaslash/proxpde.git`

[10] Barbi, G., Cervone, A., Giangolini, F., Manservisi, S., and Sirotti, L. 2024. Numerical Coupling between a FEM Code and the FVM Code OpenFOAM Using the MED Library. Appl. Sci., 14(9), 3744. https://doi.org/10.3390/app14093744

[11] `https://www.salome-platform.org/`

[12] Ribes, A., and Caremoli, C. 2007. Salome platform component model for numerical simulation. 31st annual international computer software and applications conference (COMPSAC 2007), 2, 553–564. https://doi.org/10.1109/COMPSAC.2007.185

[13] Ahrens, J., Geveci, B., Law, C. ParaView: An End-User Tool for Large Data Visualization, Visualization Handbook, Elsevier, 2005, ISBN-13: 9780123875822

[14] Ayachit, U., The ParaView Guide: A Parallel Visualization Application, Kitware, 2015, ISBN 9781930934306

[15] D. Wan, Patnaik, B., and G. Wei. 2001. A new benchmark quality solution for the buoyancy-driven cavity by discrete singular convolution. Numerical Heat Transfer: Part B: Fundamentals 40.3: 199-228. https://doi.org/10.1080/104077901752379620

[16] de Vahl Davis, G. 1983. Natural convection of air in a square cavity: a bench mark numerical solution. International Journal for numerical methods in fluids 3.3: 249-264. https://doi.org/10.1002/fld.1650030305

[17] Manzari, M. T. 1999. An explicit finite element algorithm for convection heat transfer problems. International Journal of Numerical Methods for Heat & Fluid Flow 9.8: 860-877. https://doi.org/10.1108/09615539910297932

[18] Massarotti, N., Nithiarasu, P., and Zienkiewicz, O. C. 1998. Characteristic-based-split (CBS) algorithm for incompressible flow problems with heat transfer. International Journal of Numerical Methods for Heat & Fluid Flow 8.8: 969-990. https://doi.org/10.1108/09615539810244067

[19] Mayne, D. A., Usmani, A. S., and Crapper, M. 2000. h-adaptive finite element solution of high Rayleigh number thermally driven cavity problem. International Journal of Numerical Methods for Heat & Fluid Flow 10.6: 598-615. https://doi.org/10.1108/09615530010347187

[20] Pangavhane, D. R., Sawhney, R. L. , and Sarsavadia, P. N. 2002. Design, development and performance testing of a new natural convection solar dryer.Energy 27.6: 579-590. https://doi.org/10.1016/S0360-5442(02)00005-1

[21] Fitzgerald, S. D., and Woods, A. W. 2007. Transient natural ventilation of a room with a distributed heat source. Journal of Fluid Mechanics vol. 591: 21-42. https://doi.org/10.1017/S0022112007007598

[22] Espinosa, F., Avila, R., Cervantes, J.G., and Solorio, F.J. 2004. Numerical simulation of simultaneous freezing–melting problems with natural convection. Nuclear engineering and design 232.2: 145-155.

[23] Basak, T., Anandalakshmi, R., and Singh, A. K. 2013. Heatline analysis on thermal management with conjugate natural convection in a square cavity. Chemical engineering Science 93 (2013): 67-90. https://doi.org/10.1016/j.ces.2013.01.033