VI International Conference on Particle-based Methods – Fundamentals and Applications
PARTICLES 2019
E. Oñate, M. Bischoff, D.R.J. Owen P. Wriggers, , & T. Zohdi (Eds)

OPENFOAM-INTERACTIVE (OFI): AN INNOVATIVE UNIVERSAL INTERFACE TO CONTROL SOLVERS IN OPENFOAM

ARPIT SINGHAL†*, RAPHAEL SCHUBERT†, ADHAM HASHIBON†

† Fraunhofer-Institut für Werkstoffmechanik IWM
Wöhlerstraße 11,
79108 Freiburg im Breisgau,
Germany
E-Mail: arpit.singhal@iwm.fraunhofer.de

Key words: OpenFoam-Interactive, Particle simulation, Interface Wrappers

Abstract. We present a new method for coupling fluid and particle systems that works by directly manipulating the flow field variables, mimicking the presence of solid particles rather than adding additional force terms in the governing equations as, e.g., in the traditional immersed boundary method (IBM). We demonstrate an implementation based on the open source OpenFOAM [2] package. The OpenFOAM-Interactive (OFI) presented here gives access to all internal field variables of the governing equations. This ease and facilitates complex computational and seamless data exchange and manipulation of the field variable. OFI contributes to reducing the time needed in creating the initial geometry and enables readily re-creating the geometry for the basic computational fluid dynamics (CFD) simulation steps. The presented methodology is verified for a reference simple problem (i) obstruction to flow with bluff bodies. The verified methodology is then applied as an example to demonstrate a realistic problem of heat transfer to a gas through particle bed [1]. The particle bed is created “on the fly” with OFI, this will facilitate studying fluid behavior over different particle configurations in particle beds (i.e. porosity variation) more efficiently in future.

1 INTRODUCTION

Computation fluid dynamics (CFD) is applied in various applications ranging from environmental studies to design the functionality of heat exchangers to aerodynamics applications and so on [1]–[3]. CFD works on the basic principle of dividing the entire system domain into several grids or cells and solving the governing equations of flow by numerically discretizing the partial differential equations (PDE) of the fluid phase in these small grid elements [4], [5]. The development in computational resources CFD simulations have gained quite a limelight, even simulations at resolved scales are possible today [6], [7]. Many commercial packages (like Fluent [8], Star CCM+ [9]) and open source package (like OpenFOAM [10], CFDEM Coupling [11]) exists in the current scenario.

Open Source Field Operation and Manipulation (OpenFOAM) [10], [12] is the most used open source suite of C++ libraries to CFD problems. It was first released in 1980, since then it has grown more mature to handle different complexities in physics (from macroscopic level to microscopic level). The best feature being the accessibility to the flexible framework to build your own applications as required. Different methods and algorithms are being added in every release with the help of a big community of users using OpenFOAM to back the continuous growth. These are being shared on online Git version controlled repositories (like GitHub [13], GitLab [14], etc.).

The flexibility in the OpenFOAM framework has led to many developments/improvements in solving CFD problems by developing new solvers pertaining to the specific industrial or academic question by improving the previous numerical schemes [15]–[18] or by adding features to solve addition problem domain[19]–[23].

The recent demands from industries to parametrize studies in order to optimize product and system geometries, requiring the intrinsic approach has led towards efficient automatization mechanisms. In a typical CFD case, mostly a small change in the geometry requires careful and intricate re-meshing. This limits the automation specifically and especially due to the lack of existing meshing tools in open source world.

There have been developments in the field of automating the whole procedure of running a case with OpenFOAM through python scripts which are used to serve as interface to pre and post process OpenFOAM (specifically, pyFoam and pyFlu)[24], [25]. These packages demand too many changes deep in the code to create a library interface for OpenFOAM. They offer excellent automation schemes for the cases which do not require re-creation and re-meshing of the geometry.

In this work we develop the access to OpenFOAM field variables i.e. volume scalar and volume vector fields using the client server architecture along with complete access to the OpenFOAM objects of mesh and runtime. Thus, this enables us to control and manipulate OpenFOAM operations such as Run, Edit, Print, Search, and Rerun with static meshing i.e. create objects, run and edit the cases rapidly.

In general, any OpenFOAM or any CFD associated problem consists of three stages (i) pre-processing (ii) solver and (iii) post-processing, the purpose of the work is to reduce the dependability or efforts towards pre-processing as much as possible with minimal change or duplicity required to the original OpenFOAM solvers. The aim is to introduce dynamic features to reduce the dependence on complex re-meshing and expensive dynamic meshing to work with the server client architecture in order to create, mesh and manipulate the domain variable easily. At this stage, this paper is the benchmark in this direction for our future works, where a more detailed numerical scheme behind the functionality of the OFI will be presented and applied to more physical problems.

The paper has been structured as follows, we first introduce the literature in Section 1, and we lay down the methodology implied to OpenFOAM-Interactive (OFI) in Section 2, we then add on to present the verification and results in Section 3 and in the end, we conclude with possible future outcomes from the work in Section 4.

2 METHODOLOGY

To give an idea of the overall methodology that we employ to render OpenFOAM-Interactive (OFI) to control OpenFOAM (v2.4.0 to v5.0.x tested in this paper, though we already have implementation supporting OF up to version 6.0) solvers through an interface wrapper is presented. The aim is to introduce minimal changes to the source code of OpenFOAM solver and make use of already well-established suite of the OpenFOAM framework. A schematic of representation of the workflow is shown in Fig. 1. This paper explains SimPhony OpenFoam Wrapper from Fig 1.

The main concept is shown schematically in Figure 3. OpenFOAM consists of a main loop over time or solver iterations. After each such iteration a condition is evaluated to determine whether convergence is achieved. OFI intercepts these calls and replaces them with additional machinery allowing changing the internal state of the case while it is running. Client-server architecture is designed and implemented to allow the user to send arbitrary commands to OpenFoam that are to be performed in these intercepts. These changes include: read and write cell value for scalar/vector volume fields, set up and fix a value at every time step, run for specific time, finding cells etc. While this may seem not very useful initially, an adept user can change the field values in order to exert or mimic various complex conditions e.g., simulating the presence of a rigid or flexible solid object within fluids, multiphase dynamics, chemical reactions, and arbitrary geometrical objects, to name but a few.

Key to the success of this method is the physical and mathematical soundness of the commands and variations induced by the user. The kind of changes and how to employ them is indeed not trivial and is the subject of a follow up publication. Here we demonstrate the method and some of its possibilities focusing on the software engineering aspects.

2.1 Approach and Implementation Detail

As shown in Fig. 1 any solver and any version of OpenFOAM package can be made to work with OFI. In this verification study, we consider as an example a case of incompressible Newtonian fluid in laminar flow regime for demonstrating the proposed method. The straightforward way to solve the problem of an incompressible Newtonian fluid in laminar regime with SimpleFoam solver given in Fig. 2 (a) includes, (i) creating a geometry, (ii), solution to the problem, followed by, (iii) visualizing the results. The complexity of each of these steps depends on the actual problem, and may range from being as trivial as shown in Fig. 2(b), or as complex as shown in Fig. 2(c).

If the problem is extended to behave as shown in Fig.2 (b), i.e. an obstacle to the flow has to be created (considering No-slip conditions); all the above-mentioned steps would have to be redone. This is due to the need to re-create and re-mesh of the geometry. The intention behind OFI is to reduce the efforts in doing so, i.e. to enable directly encoding the obstacle in the mesh without directly re-creating and re-meshing the whole geometry all over again. In fact, using OFI both cases in Fig. 2(a), Fig. 2(b) or Fig. 2(c) become equally easy.

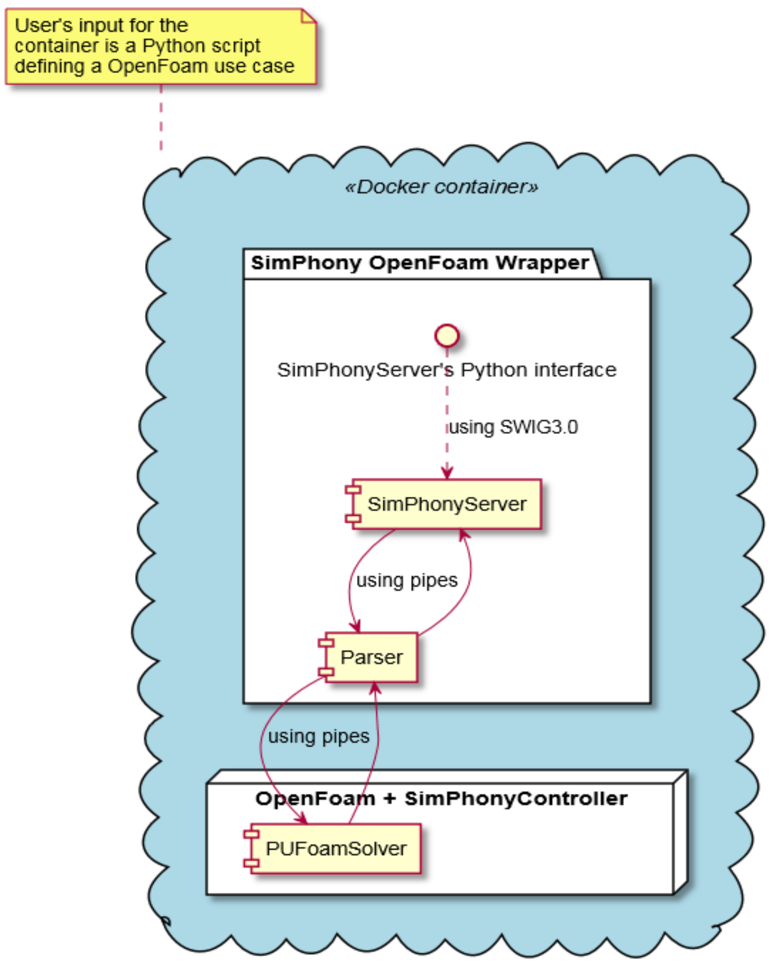
User's input for the container is a Python script defining a OpenFoam use case
«Docker container»
SimPhony OpenFoam Wrapper
SimPhonyServer's Python interface
using SWIG3.0
SimPhonyServer
using pipes
Parser
using pipes
OpenFoam + SimPhonyController
PUFoamSolver

Figure 1: A schematic representation of the OFI architecture. A think controller layer is embedded in the main loop of the OF solver which intercepts the execution of OF and alters the state seamlessly. The controller communicates through a pipe socket to the Parser which parses commands from the client. The parser in turn communicates through a socket with the client part where currently the user can make changes and issue commands. The Controller converts these commands into actions. There is python interface to this SimPhonyServer which makes the descrition of the use case in Python.

2.2 Treatment of obstacle in a flow with OFI

The obstruction to flow as shown in Fig. 2(b) is directly emulated by making the velocity of the cells in the region (i.e. the obstacle zone) fixed to zero in each iteration of the solver. In pictorial terms it is explained in Fig. 3, the case following the boundary conditions and initial conditions is prepared in OFI. In a typical SIMPLE algorithm, the previous iteration values in the cells provide as the initial values for the next iteration or current iteration. In OFI, the solver is forced to stop at each iteration and is reset (or re-initialized) with the constant value (in this case zero velocity) in cells (emulated as an obstacle). This makes the cells (in an emulated zone) behave as on obstruction zone.

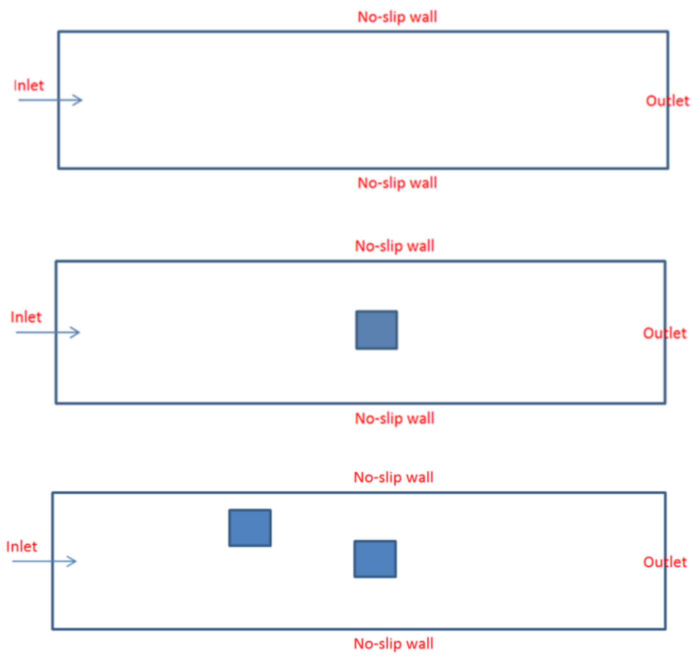

Figure 2: 2D schematic flow diagram of the problem solved in this work with OpenFOAM-Interactive (OFI) (bottom) and without OpenFOAM-Interactive, but with basic steps of any CFD setup (top). The square obstacle is created with OpenFOAM-Interactive and also with the basic step of re-creating the geometry and re-meshing for OpenFOAM-2.4.0.

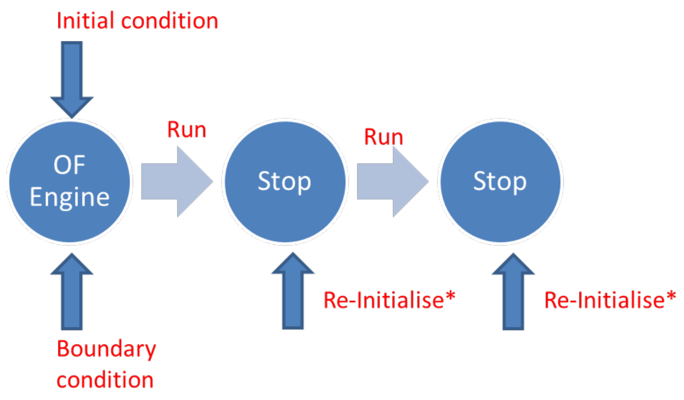

Figure 3: Pictorial representation of the re-Initializing steps for the cells in region* (obstruction) required to act as obstruction. OF-Engine represents the complete OpenFOAM-Interactive suite.

3   RESULTS

In this section, the proposed methodology of OpenFOAM Interactive (OFI) is applied to a case with SimpleFoam solver i.e. a 2D channel flow with two square obstacles disarranged to the flow. The fluid is considered to be of steady incompressible Newtonian in nature and within laminar regime (Reynolds number Re 100 and Re 50). The description of the geometry and flow parameters is listed in Table 1. The purpose of the comparison of the OFI SimpleFoam with OF-2.4.0 SimpleFoam solver is based on boundary resistance and velocity structure representation of the flow. Hence the sole criterion for comparison is velocity magnitude (in m/s).

This study is a one to one comparison between a case solved by OFI using the SimpleFoam solver and one solved entirely with the native OF SimpleFoam solver. That is, we compare the solution of an obstacle using the built in OpenFOAM methodology and compare it to a solution where in effect OpenFOAM is ran on a system without any obstacles internally (simple uniform mesh) but the obstacle is emulated by an external manipulation of cells directly using OFI. This is done to validate both the approach and the implementation. The goal is to demonstrate the capability of OFI solver to predict basic flow behavior with precision. Moreover, we consider only the hydrodynamics entrance region in this work to create the geometry; the fully developed region was not considered significant in this comparison.

The complexity used for comparison in this section is simulated with similar solver setting and geometry as previously explained in Table 1. Also, note that neither the numerical sensitivity analysis (like, grid independence) nor the solver sensitivity analyses are considered here, for a full validation there will be a follow up publication. The focus of this work is to demonstrate the capability of OFI solver to give corresponding results with similar flow and solver parameters as would be the case with the OpenFOAM-2.4.0 solver.

3.1 Results for Two square obstructions disoriented in arrangements

As an example, disoriented square obstacles to the flow were considered. This is quite a challenging problem in literature, and it is influenced by different numerical and solver accuracy parametrizations. This is because of the flow from the first obstacle creating a wake region with vortexes which hinders the inflow for the second obstacle and so on.

For similar numerical accuracy, solver parameters and Reynolds numbers (Re 100 and Re 50), the contour plots for velocity magnitude (in m/s; in Fig. 4) shows the ability of OFI to capture the flow behavior with much accuracy compared with OF-2.4.0. To have a thorough comparison of the flow profiles obtained from both approaches, 3 different regions (i.e. Lines L1, L2, L3) in the simulated geometry (all in different flow regimes) were analyzed as shown in Fig. 5. The results obtained for velocity magnitude fluctuations are within the acceptable error margin (within 5% difference margin).

3.2 Result of flow over a 2D representation of packed particle bed

In the previous section obstacles to flow is emulated to see the velocity fluctuation to the flow. A more complex case with simulation parameters from Table 2 and having a real application with multiple blocked cells portrayed as a packing of a packed bed is simulated in this section to extract the temperature profile. This happens without re-meshing for the particles and by recalibrating the values of temperature in the cells to behave as the particles of constant heat source. We emphasize that this is a first step towards a more detailed analysis of particle-fluid and solid-fluid interaction based on OpenFOAM-Interactive (OFI).

It is shown in Figure 6 that the 2D representation of the cells behaving as the constant heat source (573K) is heating the fluid (at 473K) till they reach an equilibrium temperature.

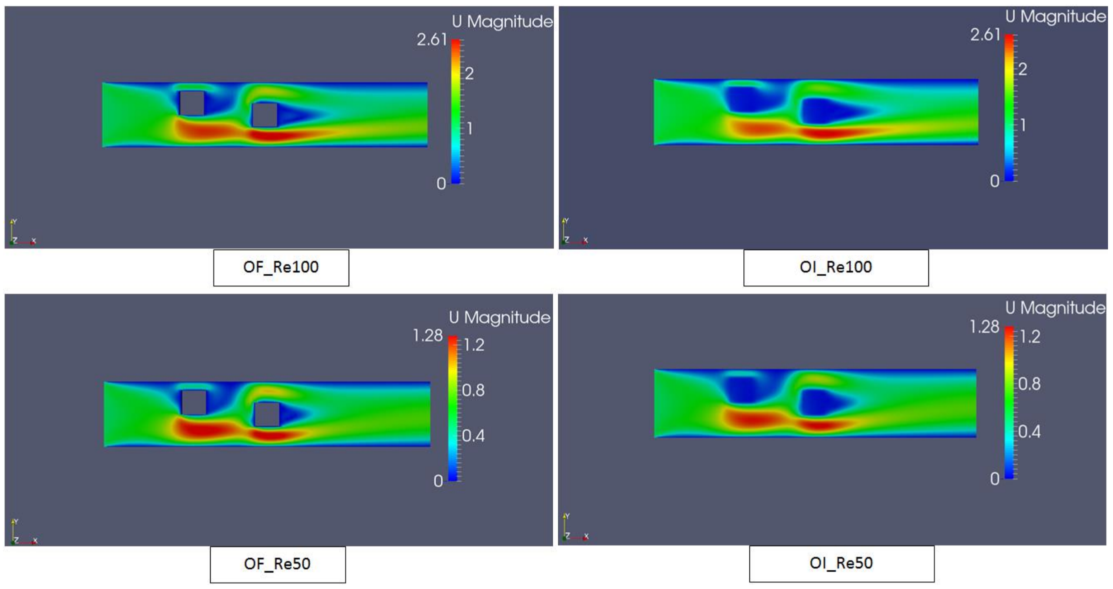

Figure 4: Velocity contour graphs for disoriented square obstruction to flow from OpenFOAM-Interactive (OFI) (right) and OpenFOAM-2.4.0 (OF) (left); SimpleFoam solver for a flow at Re100 and Re50 [NOTE: the scale is in (m/s); also, the dimensions of the square obstacles are 0.141m^2].

Table 1. Details of the parameters used in simulation. *The OFI is compatible with all versions of OpenFOAM

Parameter	Value
Length of the 2D channel (m)	5
Width of the 2D channel (m)	1
Mesh resolution	40x40
Number of mesh elements	1600
Reynolds number (Re)	100; 50
Solver used	SimpleFoam
OpenFOAM version*	5.0.x

Table 2. Details of the parameters used in simulation. *The OFI is compatible with all versions of OpenFOAM

Parameter	Value
Length of the 2D channel (m)	2
Width of the 2D channel (m)	5
Mesh resolution	40x100
Number of mesh elements	4000
Reynolds number (Re)	10
Solver used	byoyantBoussinesqSimpleFoam
OpenFOAM version*	5.0.x

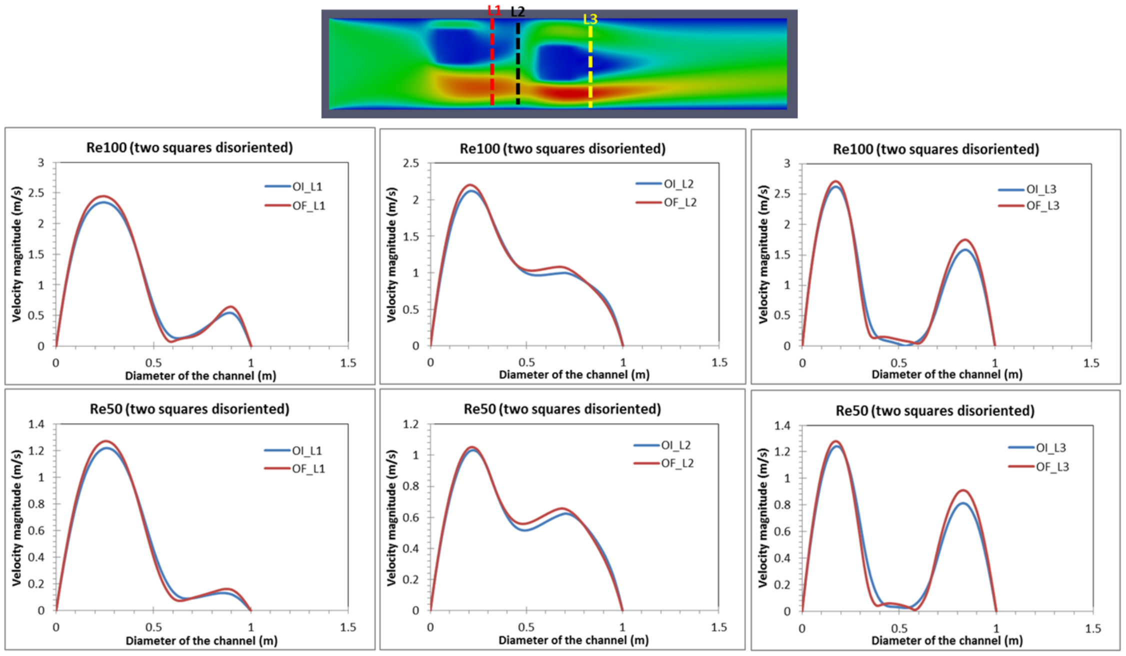

Figure 5: The comparison of velocity magnitude (m/s) profiles obtained for Re100, Re50 between OpenFOAM-Interactive (OFI) and OpenFOAM-2.4.0 (OF) for double disoriented square obstruction of similar sizes. The velocity data profiles are plotted over lines (L1, L2 and L3) from bottom wall of the channel to the top wall).

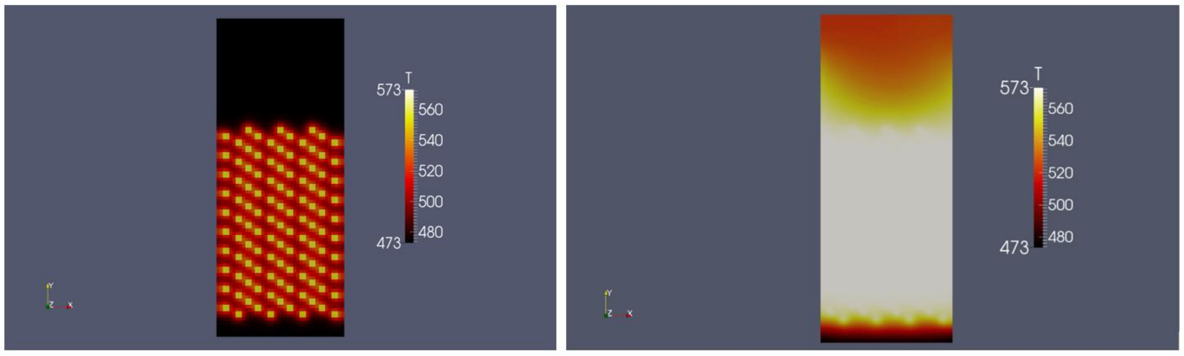

Figure 6: Temperature contour plots for 2D particle bed generated with OFI as cells of constant heat source. [left] is the profile at the first iteration showing the temperature in the cells manipulated to behave at 573K. [right] shows the evolution of the temperature profile in the geometry domain after the temperature has reached an equilibrium.

4 CONCLUSIONS

We present an innovative simple yet efficient means to control OpenFOAM with minimal changes to the code by manipulating directly the control flow. The proposed method allows enhanced automation and provides new means of performing complex CFD modelling, in particular it enables alteration of the numerical domain without re-meshing.

The concept introduced in this paper laid a path forward to utilizing server client architecture to completely control OpenFOAM and similar tools. This will form an integral part of an open simulation platform and is planned to be offered as a service and APP on the upcoming Materials Modelling MarketPlace (https://www.the-marketplace-project.eu/), which is a framework of integrating simulation frameworks in a semantic and interoperable environment facilitated by the common universal data structure (CUDS). The unique feature of the presented methodology i.e. OpenFOAM-Interactive (OFI) is the adaptability to the change in OpenFOAM versions. It is flexible to plug-in it to any OpenFOAM versions.

The presented package OFI has been applied to solve an interesting problem from literature i.e. bluff bodies to the flow. Our presented methodology helps scale the steps of re-creating and re-meshing the geometry when and if required according to the change in the simulated problem. The results obtained were within the error margin of 5% which demonstrates that our developed OFI suite works well.

Following that the methodology is applied to imitate a particle packing simulation. The validation and verification of the detailed heat transfer analysis resulting from the change in porosities and validation from the literature pertaining to more real-time engineering problems with packed beds will be a part of the future studies.

REFERENCES

[1] B. Sanderse, S. P. Pijl, and B. Koren, "Review of computational fluid dynamics for wind turbine wake aerodynamics," *Wind Energy*, (2011), **14**, 7, 799–819.

[2] M. M. Aslam Bhutta, N. Hayat, M. H. Bashir, A. R. Khan, K. N. Ahmad, and S. Khan, "CFD applications in various heat exchangers design: A review," *Appl. Therm. Eng.*, (2012), **32**, 1–12.

[3] H. Wang, H. Wang, F. Gao, P. Zhou, and Z. (John) Zhai, "Literature review on pressure–velocity decoupling algorithms applied to built-environment CFD simulation," *Build. Environ.*, (2018), **143**, 671–678.

[4] Y. Lim and S. B. Jørgensen, "Partial Differential Equations (PDE) and Computational Fluid Dynamics (CFD)," (2006), **Chapter 1.**, 2006, 35-106 BT-Computer Aided Process & Products Eng.

[5] P. F. Galpin, J. P. Van Doormaal, and G. D. Raithby, "Solution of the incompressible mass and momentum equations by application of a coupled equation line solver," *Int. J. Numer. Methods Fluids*, (2018), **5**, 7, 615–625.

[6] A. Singhal, S. Cloete, S. Radl, R. Quinta-Ferreira, and S. Amini, "Heat transfer to a gas from densely packed beds of monodisperse spherical particles," *Chem. Eng. J.*, (2017), **314**, 27–37.

[7] A. Singhal, S. Cloete, S. Radl, R. Quinta-Ferreira, and S. Amini, "Heat transfer to a gas from densely packed beds of cylindrical particles," *Chem. Eng. Sci.*, (2017), **172**, 1–12.

[8] "ANSYS® Academic Research Mechanical, Release 18.1, Help System, Coupled Field Analysis Guide, ANSYS, Inc." .

[9] "STAR CCM+ Users Manual. http://www.cd-adapco.com/products/star-ccm/documentation." .

[10] H. G. Weller, G. Tabor, H. Jasak, and C. Fureby, "A tensorial approach to computational continuum mechanics using object-oriented techniques," *Comput. Phys.*, (1998), **12**, 6, 620–631.

[11] C. Goniva, C. Kloss, N. G. Deen, J. A. M. Kuipers, and S. Pirker, “Influence of rolling friction on single spout fluidized bed simulation,” *Particuology*, (2012), **10**, 5, 582–591.

[12] “OpenFOAM.” .

[13] Http://en.wikipedia.org/wiki/GitHub, “Github.” [Online]. Available: http://github.com/.

[14] https://en.wikipedia.org/wiki/GitLab, “GitLab.” [Online]. Available: https://about.gitlab.com/.

[15] G.-H. Kim and S. Park, “Development of a numerical simulation tool for efficient and robust prediction of ship resistance,” *Int. J. Nav. Archit. Ocean Eng.*, (2017), **9**, 5, 537–551.

[16] M. V. Kraposhin, E. V. Smirnova, T. G. Elizarova, and M. A. Istomina, “Development of a new OpenFOAM solver using regularized gas dynamic equations,” *Comput. Fluids*, (2018), **166**, 163–175.

[17] C. Goniva, C. Kloss, A. Hager, and S. Pirker, *An open source CFD-DEM perspective*, (2010). 2010.

[18] C. Fernandes, F. Habla, O. Carneiro, O. Hinrichsen, and J. M. Nobrega, *Development of a new 3D OpenFOAM® solver to model the cooling stage in profile extrusion*, (2016). 2016.

[19] S. M. Longshaw *et al.*, “mdFoam+: Advanced molecular dynamics in OpenFOAM,” *Comput. Phys. Commun.*, (2018), **224**, 1–21.

[20] C. White *et al.*, *DsmcFoam+: An OpenFOAM based direct simulation Monte Carlo solver*, (2017). 2017.

[21] H. Jasak and M. Beaudoin, *OpenFOAM turbo tools: From general purpose CFD to turbomachinery simulations*, (2011), **1**. 2011.

[22] P. Horgue, C. Soulaine, J. Franc, R. Guibert, and G. Debenest, “An open-source toolbox for multiphase flow in porous media,” *Comput. Phys. Commun.*, (2015), **187**, 217–226.

[23] L. Orgogozo *et al.*, “An open source massively parallel solver for Richards equation: Mechanistic modelling of water fluxes at the watershed scale,” *Comput. Phys. Commun.*, (2014), **185**, 12, 3358–3371.

[24] B. F. W. Gschaider and A. Arbor, “Automatic case setup with pyFoamPrepareCase,” (2015)June, 1–185.

[25] B. F. W. Gschaider, “Automatization with pyFoam - How Python helps us to avoid contact with OpenFOAM,” *5th OpenFOAM Work.*, (2010)June, 1–80.