ARCHER2-eCSE01-28 - Multi-Resolution Coupling for Exascale Engineering - Technical Report

Marta Camps Santasmasas,[1] Chrysovalantis Tsinginos,[2] Ian Hinder,[3] Yang Zhou,[1] Alessandro de Rosis,[1] Jianping Meng,[2] Charles Moulinec,[2] and Alistair Revell[1]

[1)] *Department of Mechanical, Aerospace and Civil Engineering, The University of Manchester, Manchester M13 9PL, UK.*

[2)] *Scientific Computing Department, Science and Technology Facilities Council, Daresbury Laboratory, Keckwick Lane, Daresbury, Warrington WA4 4AD, UK.*

[3)] *Research IT, The University of Manchester, Manchester M13 9PL, UK.*

We present a framework that couples the Code_Saturne unstructured-mesh finite volume Navier-Stokes code to the LUMA Lattice-Boltzmann code. We show here the results of validation tests against solutions in the literature and scaling tests on ARCHER2. The coupled code has been run on up to 512 nodes, where it shows good weak scaling. Good strong scaling is obtained from 2 to 256 nodes. LUMA is also coupled with the discrete element package of LAMMPS, a molecular dynamics solver. The developed coupling framework was validated against benchmark cases and strong scaling tests were performed.

I. INTRODUCTION

In recent years, multi-scale numerical simulation has drawn attention in computational fluid dynamics (CFD) since multi-scale physical phenomena are widespread in science and engineering[1]. Generally, different scale problems, in space and time, are described by different physical laws. Based on the continuum medium hypothesis, traditional numerical methods, *e.g.* finite volume methods (FVMs), are best suited to macroscopic-scale simulations, while the effect of micro scales is ignored or replaced by empirical equations. To go beyond this approximation, micro/mesoscopic methods have been developed over the past few decades, *i.g.* molecular dynamics (MD) and lattice Boltzmann methods (LBM). However, it is often impractical to simulate the whole domain across the scales, using these pure micro/mesoscopic methods due to limitations of computational resource. Therefore, coupling different scale methods provide the possibility for the investigation of multi-scale problems.

Up to now, a lot of effort has been devoted to coupling simulations[2,3], including FVM-LBM[4,5] and LBM-MD[6,7]. Mesoscopic LBM bridges the macroscopic and microscopic simulation scales and is able to handle arbitrarily geometric complexity via the 'bounceback' approach. Meanwhile, the macroscopic FVM is widely used in engineering applications with the advantages of numerical stability and computational efficiency. Consequently, in this project, we developed two coupling models: one coupling between (i) a finite volume method code, Code_Saturne[8] (version 7.0.2), and (ii) LUMA[9] (version 1.7.12), a code implementing the lattice Boltzmann method[10], to perform 3-dimensional simulations of fluid flows. In the second part of this project, the discrete element package of the molecular dynamics code, LAMMPS[11] (version 23 November 2013) was coupled with LUMA (version 1.7.12) for developing an efficient platform for the direct modelling of fluid-particle systems. The discrete element method represents a granular material as a collection of rigid bodies[12] that may interact with other particles through a contact model and/or with the surrounding fluid. In the rest of the report, Sec. II and Sec. III focuses on the coupling model establishment, validation and parallel performance for LUMA coupled to Code_Saturne and LAMMPS, respectively. Finally, a brief conclusion and future developments are presented in Sec. IV.

II. COUPLING BETWEEN CODE_SATURNE AND LUMA

A two-way coupling model is established between the finite volume and lattice Boltzmann methods implemented by the Code_Saturne and LUMA codes, respectively, whilst the communication between the two relies on the Parallel Location and Exchange (PLE) library, also distributed within Code_Saturne release. Here, we first outline the coupling process at each time step. Next, we describe a verification of the accuracy and reliability of the proposed approach. Finally, we discuss the performance and scaling of the coupling model on ARCHER2.

A. Coupling implementation

Code_Saturne will be the code driving the coupling process, which consists of three main parts: configuration, point-to-point mapping based on the PLE library and data transfer process between Code_Saturne and LUMA.

1) Configuration for Code_Saturne and LUMA:

- In Code_Saturne, the file *setup.xml* contains standard Code_Saturne parameters and the problem specification. The coupling configuration is stored in the source file *cs_user_coupling.c*, and the function `cs_luma_coupling_define` defines the coupling parameters from the point of view of Code_Saturne[13]. The combination of *definitions.h*,

setup.xml and *cs_user_coupling.c* defines a coupling case.

- From LUMA side, all settings are in the file *definitions.h*, where `L_ACTIVE_PLE` activates the coupling functionality. `L_PLE_INTERFACES` defines one-way or two-way coupling, and the fields to couple are set by `pleRead` and `pleWrite`. The coupling interface regions are defined by the coordinates (`plePosX`, `plePosY`, `plePosZ`) and the corresponding size (`pleSizeX`, `pleSizeY`, `pleSizeZ`) in each direction. The `eCoupling` boundary condition is used on LUMA boundaries to indicate that they should receive data from Code_Saturne.

The PLE coupling library is designed to simplify the coupling of distributed parallel computational codes. It provides support for synchronising parallel codes at predefined points, enabling parallel mapping of points to meshes, and transfer of variables using the created mapping[14]. Code_Saturne and LUMA pass the coupling configuration to the PLE library to set up the mapping and data transfer process.

2) Point-to-point mapping between Code_Saturne and LUMA:

For each coupling interface, a PLE locator is created through `ple_locator_create`. To configure the PLE locator, the point-to-point mapping is established based on the created IDs and coordinates array using the PLE function (`ple_locator_set_mesh`), and both Code_Saturne and LUMA call this function. Specifically, Code_Saturne and LUMA exchange the coupling points and calculate the distance between the distant and the local coupling points. If the distance of two points is within the default tolerance, they are added to the mapping. In addition, PLE provides a function (`ple_locator_get_n_exterior`) to check if all defined coupling points have been located; if they have not been, the code exits with an error.

LUMA defines a class (`PLEAdapter`) which contains variables, structures and functions to couple with Code_Saturne. During initialisation, LUMA reads the coupling information, *i.e.* coupling point coordinates and field types. Next, the mapping of points to LUMA cells is established via the function (`pointInMesh`). If the coupling points are found in LUMA cells on the current MPI rank, LUMA obtains the points' IDs and global coordinates and stores them. Meanwhile, Code_Saturne executes similar operations after calling the function (`cs_luma_coupling_define`).

To clarify the mapping process, we now give an example. The domains used by the two codes overlap, and the boundary of the domain of one code is used as a coupling interface to receive data from the other code. In Fig. 1, we show the part of the mesh near the coupling region, with LUMA on the left in blue, and Code_Saturne on the right in red. On the left coupling interface (green box), LUMA extracts data and sends it to Code_Saturne while data are sent back to LUMA from Code_Saturne on the right coupling interface. LUMA stores data at cell centres, while Code_Saturne

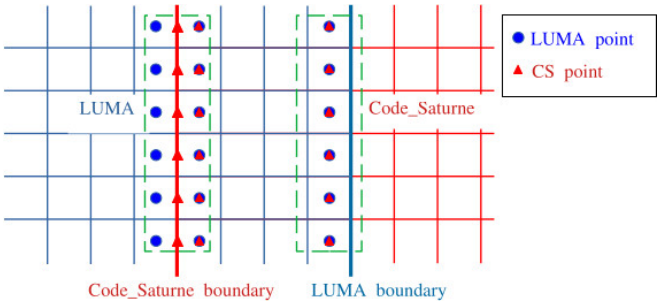

Figure 1. Coupling mesh on LUMA and Code_Saturne side

stores both cell and face centered data, as shown in Fig. 1. As a result, it is possible to align the interfaces with the mesh such that the two meshes are completely coincident on the right coupling interface, while the face centre is located halfway between two cells of LUMA on the left interface.

3) Data exchange between Code_Saturne and LUMA:

Following the initialisation and mapping, Code_Saturne and LUMA exchange data with each other using the functions `receiveData()` and `sendData()`. Using the PLE locator, the local code obtains the total number of coupling points and their IDs which correspond to the distant points `ple_locator_get_n_dist_points` and `ple_locator_get_dist_locations`. During the data exchange process, Code_Saturne sends data to LUMA through the PLE function (`ple_locator_exchange_point_var`), and vice versa. LUMA performs unit conversion between the physical units used by Code_Saturne and the LBM units used by LUMA.

In the aforementioned example, in the left coupling domain as shown in Fig.1, the Code_Saturne faces lie halfway between two LUMA cells, so LUMA employs linear interpolation to approximate data at the face positions before sending them to Code_Saturne.

Code_Saturne simulates directly the physical fields, such as velocity and pressure, whereas LUMA simulates population distribution functions. During the exchange of data between the two codes, the physical fields for Code_Saturne can be uniquely determined from the LUMA population distributions, whereas there is not a unique way to define the population distributions for LUMA from the physical fields in Code_Saturne. Some schemes have been proposed, such as making use of the equilibrium distribution[15] and the use of a reconstruction operator[16,17]. Here, we use the equilibrium distribution.

B. Validation

To validate the coupling implementation, we set up a 2-dimensional lid-driven cavity problem on a unit square domain. Since Code_Saturne and LUMA are both 3-dimensional codes, the 2-dimensional problem is solved by the use of a 3-dimensional mesh with periodic boundary conditions in the third direction. For technical reasons, the minimum size of the domain in that direction is set to 5 cells.

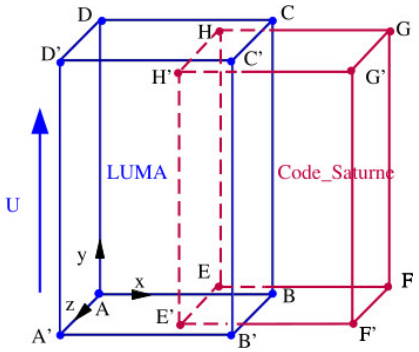

Figure 2. 3D coupling model

Figure 2 shows the 3-dimensional geometry model with LUMA in blue and Code_Saturne in red. While Code_Saturne is an unstructured mesh code, we here adopt a structured mesh with the same structure as the LUMA mesh for simplicity.

We choose a grid spacing of dx $= 1/200\,\mathrm{m}$ to mesh the domain of size $1 \times 1 \times 5\,\mathrm{dx\,m}^3$. The LUMA part covers $0 \le x \le 0.6\,\mathrm{m}$, while the Code_Saturne part covers $0.4\,\mathrm{m} \le x \le 1\,\mathrm{m}$, leading to an overlap in the horizontal (x) direction of size 0.2 m. The Reynolds number of the flow is chosen as 400, and a driving velocity boundary condition on the LUMA boundary at $x = 0$ is $u_y = 1\,\mathrm{m/s}$. It should be noted that LUMA has two extra layers of cells in the vertical (y) direction due to the wall boundary condition implementation there.

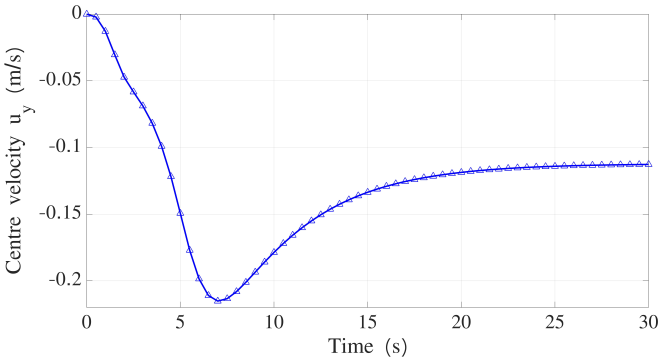

Figure 3. Velocity u_y in the centre of the domain as a function of time

Figure 3 shows the velocity u_y in the centre of the domain as a function of time. The velocity converges after about 25 s, and it only increases by 7.5×10^{-6} during the last 5 s. This accuracy is reasonable given the low computational cost during the verification stage.

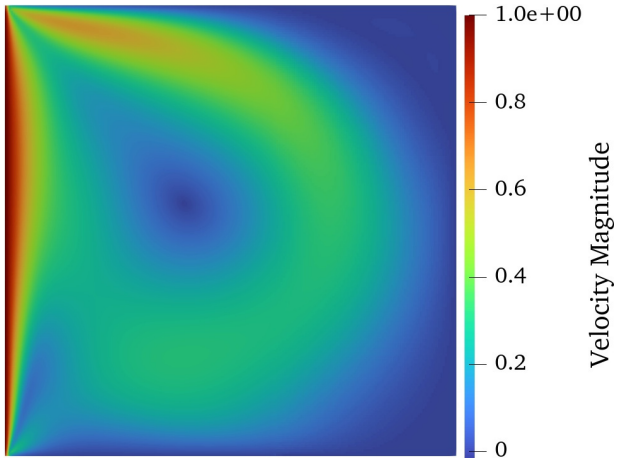

Figure 4. Velocity field for coupled lid-driven cavity simulation at Re = 400

The velocity field at $t = 30\,\mathrm{s}$ is illustrated in Fig.4, in which the fluid smoothly flows rightward from LUMA to Code_Saturne in the top half of the domain, and leftward from Code_Saturne to LUMA in the bottom half of the domain.

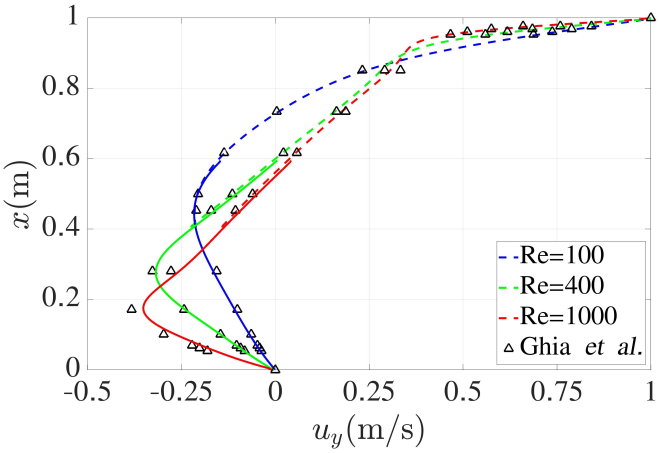

Figure 5. Quantitative analysis of coupling results for various Re numbers

We have run several cases with different Reynolds numbers (100, 400 and 1000) to test the robustness of the coupling model. Figure 5 quantitatively shows the velocity field at different Reynolds numbers. The results obtained from our coupled simulation match well with the reference data[18], though there is some deviation for $Re = 1000$. This is because a resolution of 200 cells is insufficient to capture the fine flow features arising at $Re = 1000$[19] with the LBM scheme.

The bounceback boundary condition is an interesting and versatile feature of LBM, which readily enables any

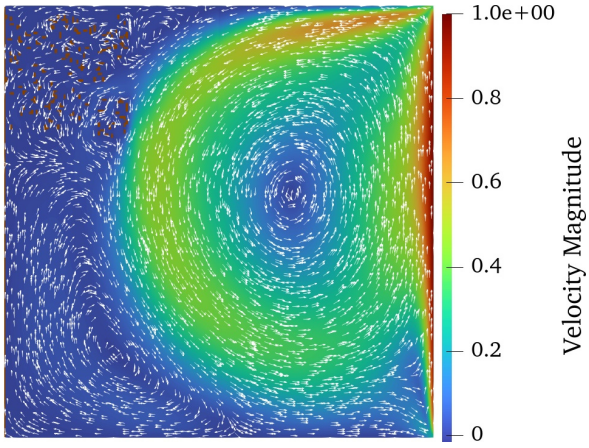

Figure 6. Flow field of lid-driven cavity with media of porosity 0.9

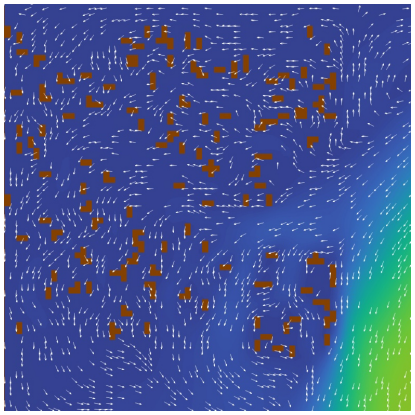

Figure 7. Enlarged view of top-left of Fig.6 for media of porosity 0.9

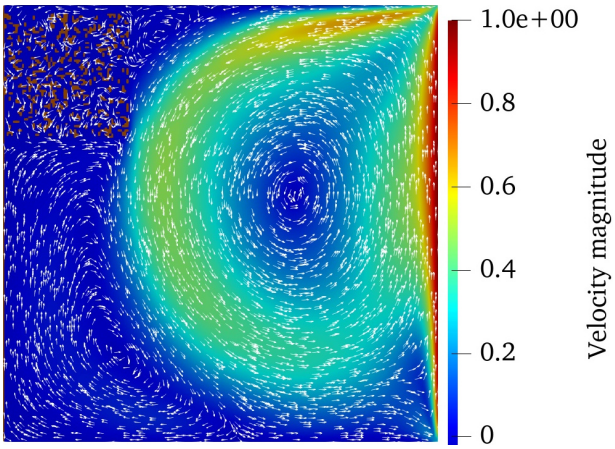

Figure 8. Flow field of lid-driven cavity with media of porosity 0.8

lattice point to be treated as a solid boundary. This feature is particularly useful for complex geometry like porous media. We have simulated two cases where porous media is placed on the top-left of the LUMA domain,

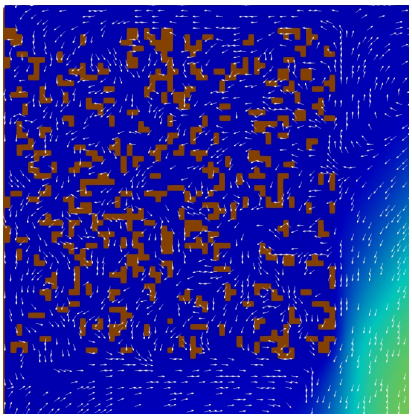

Figure 9. Enlarged view of top-left of Fig.8 for media of porosity 0.8

coupled with Code_Saturne on the right side. The porous media domain is $0.3 \times 0.3 \times 0.025\text{m}^3$, and its porosity is 0.9 and 0.8 in the two cases. The converged velocity fields from the two cases are shown in Figs.6 and 8, and Figs.7 and 9 shows the local flow field in porous media. it can be observed that the complicated flows inside of the porous structure is well simulated.

C. Performance

In order to check that good scaling is obtained on ARCHER2, we have run both weak and strong scaling tests for the 2-dimensional lid-driven cavity case with Code_Saturne and LUMA coupled, based on the aforementioned validation case. This is a 2-dimensional test, but we have artificially extended the domain in the third direction with periodic boundary conditions, because both codes are 3-dimensional codes.

The case is similar as described in the validation section, except that the overlap is of size $0.1\,m$ rather than $0.2\,m$. We define the resolution $N = 1/\text{dx}$ as the number of cells per unit length.

In all cases, we run with 128 processes (MPI tasks) on each 128-core node. We take 256 processes (2 nodes) as the base case, as in the coupled case, each code runs on a separate node. We use a pure MPI configuration, rather than hybrid MPI/OpenMP.

To assess parallel scaling, we run the coupled code on different numbers n of nodes and measure the speed of the simulation. We define the speed of the simulation as the number of cell updates per unit time, i.e. speed $:= N_{\text{cells}} \times N_{\text{steps}}/T$ where T is the time taken to simulate N_{steps} timesteps on a mesh with N_{cells} cells. Ideally, this quantity should scale with the number of processes, independent of the problem size, resolution, or timestep.

We run for 40 fixed timesteps, with dt $= 0.005(100/N)^2$ in both codes, since in the current coupling implementation, the timesteps taken by the two

codes must be the same. The speed measured is the average of the last 10 timesteps (any finalisation time is excluded from the measurement). All output, checkpointing and logging have been disabled during these 10 timesteps.

In the weak scaling test, we run cases with $n = 2, 4, 8, ..., 512$ nodes. We adjust the resolution N of each run such that there are approximately the same number ($\sim$ $500K$) of cells on each node for each value of n. This test demonstrates how well the code scales to larger problem sizes.

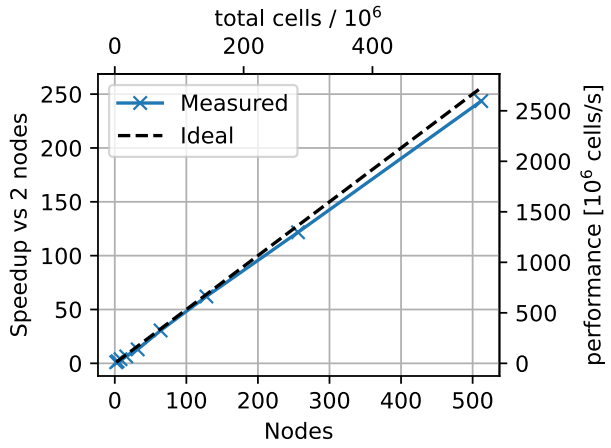

Figure 10. Weak scaling of the coupled Code_Saturne and LUMA codes on ARCHER2

Figure 10 shows that the performance scales slightly less than linearly (the dashed line) with n up to 512 nodes.

For the strong scaling test, we run cases with $n = 2, 4, 8, ..., 512$ nodes, but in this case, we keep the resolution N of each run fixed, so that the same problem with $\sim$ 70M cells is being solved on different numbers of nodes. This test demonstrates how much speedup can be obtained for the same problem by running on more nodes.

Figure 11 shows that the code scales reasonably up to 256 nodes, but that for higher numbers of nodes, the performance drops off. This is likely due to the high ratio of communication to computation.

III. COUPLING BETWEEN LUMA AND LAMMPS

We have also developed a coupling framework for the direct modelling of fluid-particle systems, in order to prepare for future work related to a 3-way coupling CFD-LBM-DEM. In the developed coupling framework, LUMA is coupled with the popular molecular dynamics solver, LAMMPS[20] which is used to update the response of the granular phase. Then, an interface was developed to handle the intersolver communication and model the

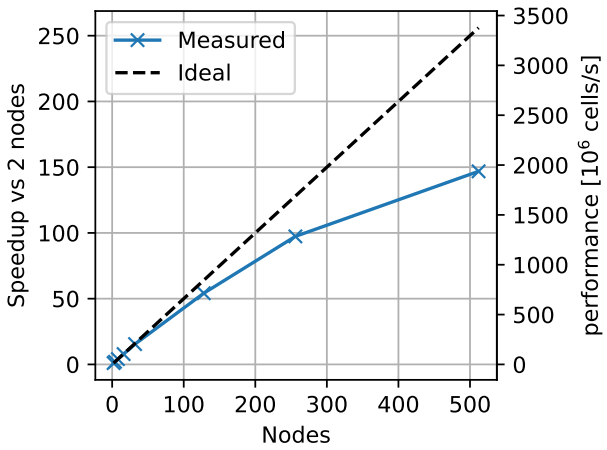

Figure 11. Strong scaling of the coupled Code_Saturne and LUMA codes on ARCHER2

interactions between fluid and particles in the LBM code.

A. Coupling framework

The developed coupling interface relies on the multiscale universal interface (MUI) library[21] to exchange data between the two solvers, while the immersed boundary method (IBM)[22] is used to model the fluid-particle interactions. The IBM was already implemented in LUMA but was improved to support much larger marker deformations as required by the coupling framework.

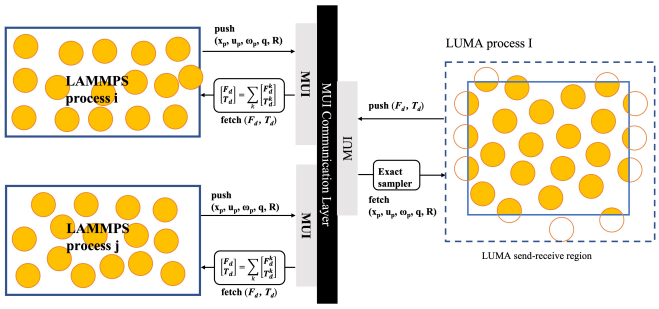

Figure 12. Overview of the exchange of data between LUMA and LAMMPS in the developed coupling framework.

To facilitate the multiscale modelling of fluid-particle systems, particle data must be exchanged between the two solvers (see Fig. 12). For updating the motion of each particle, the hydrodynamic force and torque exerted by the fluid on the particle must be included in the particle equations of motion. In the developed coupling framework, the hydrodynamic forces and torques are computed in LUMA and sent to LAMMPS. As LAMMPS particles may be distributed to more than one LUMA processes, a reconstruction process has been developed to update

the hydrodynamic force and torque exerted to each particle. For updating the fluid flow, the particles must be projected to the LBM grid and then a fluid-particle interaction model is used to resolve the interactions between particles and fluid. The default data that are required from numerical schemes that model the fluid-particle interactions within the lattice Boltzmann method are the positions and velocities of the centre of mass of the particles and the parametres that define the particle surface. For modeling the fluid-particle interactions with the IBM, the particle orientations must also be included in the list of particle data that are sent to LUMA. The particle orientations are used to update the marker position with respect to the fixed LBM grid. In the developed coupling framework, quaternions are used instead of Euler angles to update the particle orientation as the former are singularity-free parametres.

The data exchange between the solvers relies on the definition of send/receive regions as intersolver communications to take place only between processes with overlapping regions. In LUMA, the send/receive region assigned to each process is predefined as the part of the computational grid assigned to the given process. This region is expanded by the maximum particle diametre to ensure the correct particle projection in the LBM grid. In LAMMPS, the send/receive region is defined as the part of the simulation domain assigned to each process at the current step. This allows the coupling to be used with evolving boundaries and dynamic domain decomposition in LAMMPS.

For extending the LUMA IBM to large deformations, we followed the principle that a given particle is owned by multiple LUMA processes while the markers of each particle are owned by the LUMA process on which they are projected. A build-update policy was introduced to compute the kernel functions of the markers owned by the given process. By exploiting the Lagrangian nature of the IBM, a new support and marker communication list is built if a single marker enters the computational cell of a different grid node. Marker communication lists are built only for markers of particles that intersect the boundary marker zone which has a thickness of $2\Delta x$.

B. Validation cases

To validate the developed coupling framework, we consider the sedimentation of (I) a single particle and (II) two particles in a 2-dimensional box.

Case I: Sedimentation of a single particle

The sedimentation of a circular disk of diameter $D = 0.025\,\mathrm{m}$ under the action of gravity $g = 9.806\,\mathrm{m/s^2}$ in a box of $0.02\,\mathrm{m} \times 0.06\,\mathrm{m}$ is used to validate the developed coupling framework. The disk has a density of $\rho_s = 1250\,\mathrm{kg/m^3}$ and is initially located at $(0.01\,\mathrm{m}, 0.04\,\mathrm{m})$. The fluid density, ρ, and the kinematic viscosity are $1000\,\mathrm{kg/m^3}$ and $10^{-5}\,\mathrm{m^2/s}$, respectively. The bounce-back scheme is used to enforce no-slip condition at the walls of the container. Particle-wall interactions are modelled with the linear contact model and the friction coefficient is set to 0.5. A grid with a particle distribution ratio, $D/\Delta x$, of 40 was used.

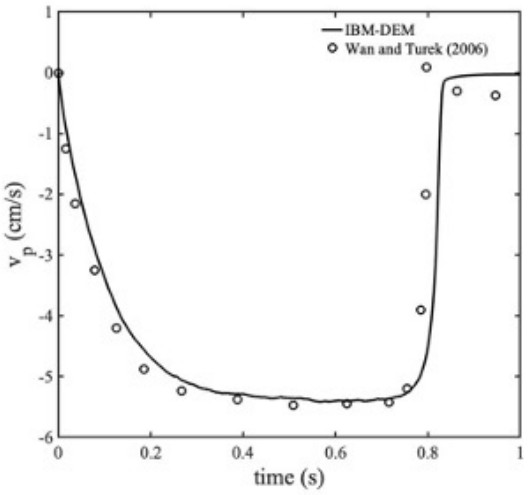

Figure 13. Time histories of vertical velocities for sedimentation of a single particle

The evolution of the vertical particle velocity is presented in Fig. 13. The predictions of the developed coupling framework agree quite well with the finite-element simulations of Wan and Turek[23]. As the particle approaches the base of the container, the two methods diverge as different interaction models were used to model the particle-wall interactions in the two studies.

Case II: Sedimentation of two particles

In the second case, we consider the sedimentation of two circular particles in a two dimensional box as shown in Fig. 14. Two cylinders of diameter $D = 0.002\,\mathrm{m}$ and density $\rho_s = 1010\,\mathrm{kg/m^3}$ are positioned in a box 0.02 m wide and 0.08 m high and fall under the action of gravity $g = 9.8\,\mathrm{m/s^2}$. The particles are initially at rest at the location $(0.00999\,\mathrm{m}, 0.072\,\mathrm{m})$ and $(0.01\,\mathrm{m}, 0.068\,\mathrm{m})$, respectively. The density and fluid viscosity are $1000\,\mathrm{kg/m^3}$ and $10^{-4}\,\mathrm{kg/(m \cdot s)}$, respectively. A uniform grid of $D/\Delta x = 40$ is used to mesh the fluid domain. The boundary conditions and contact model are the same as for Case I. The stiffness of the normal and tangential spring are set to $280000\,\mathrm{N/m}$ while the friction coefficient is set to 0.4.

The time histories of the vertical velocities are presented in Fig. 15. For comparison purposes, the results

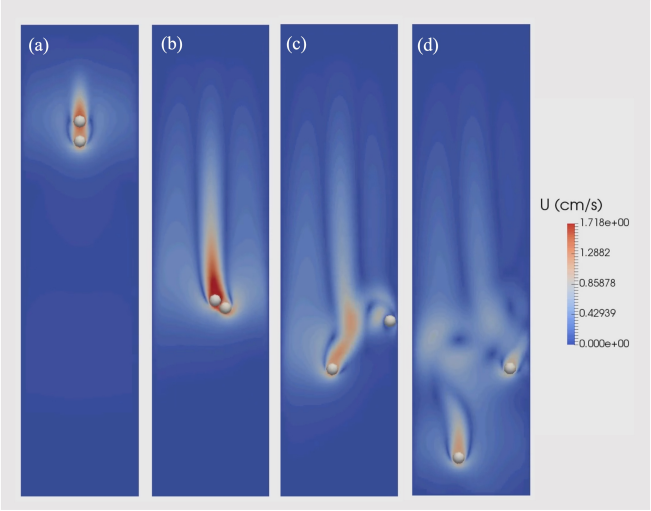

Figure 14. Fluid velocities and particle positions for two particles settling in a two dimensional channel at (a) $t = 0.97\,\mathrm{s}$, (b) $t = 2.86\,\mathrm{s}$, (c) $t = 3.8s$ and (d) $t = 5.2s$.

of Feng and Michaelides[24] are also included in the figure. The predictions of the developed coupling framework agree quite well with the results of Feng and Michaelides. Small differences in the results are attributed to the different interaction models that are used to model the particle-particle interactions in the two studies.

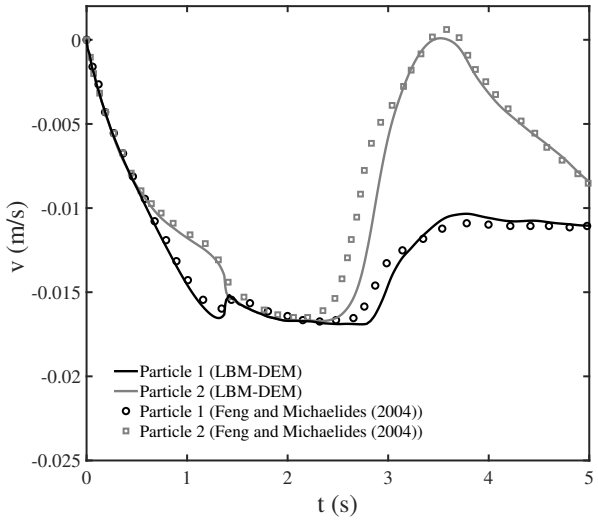

Figure 15. Time histories of vertical velocities for sedimentation of two cylindrical particles

C. Parallel performance

The performance of the developed coupling framework was assessed by conducting strong scaling tests. A preliminary analysis indicated that at least 95% of the computational time of the LAMMPS solver was spent on the coupling interface. Hence, the critical part of the developed coupling framework is the performance of LUMA in conjunction with the employed fluid-particle interaction scheme and the intersolver communications. We also considered the parallel performance of LUMA and of the DEM-LBM initialisation stage of the coupled framework, as they constitute upper bounds on the performance of the framework. In the LBM-DEM, initialization stage, the particles are considered to be fixed in space.

For the strong scaling test, we considered the migration of 100,000 cylindrical particles of diameter D in a 2-dimensional channel with a length of $4,572D$ and a height of $93.02D$. The fluid density and kinematic viscosity are equal to ρ_0 and $0.001c_sD$, respectively. A Poiseuille velocity profile with a maximum velocity of $u_m = 0.001c_s$ was prescribed at the inlet, where c_s denotes the lattice sound speed. A uniform grid of $196,608 \times 4,000$ grid nodes was used to mesh the simulation domain. The density of each particle was 2.6ρ. Particle-particle and particle-wall interactions were modelled with a linear contact model with a normal and tangential stiffness of $0.1E$, where E is the modulus of elasticity, and a friction coefficient of 0.4.

The parallel performance of the developed coupling framework is analysed based on the speed-up ratio defined as the ratio of the computational time taken by 1 node per solver to perform 500 steps to the time taken by p nodes per solver to perform the same computational task. Unfortunately, issues related to the fabric of ARCHER2 prevented us from running the developed coupling framework on more than 64 ranks per solver. Thus, we present only the results of the scaling for the pure LUMA and the LBM-DEM initialisation stage in Fig. 16. Both LUMA and the initialisation stage of the coupling framework scale well up to 4096 MPI tasks per solver, with the scaling of the LBM-DEM initialisation found to be slightly worse than that of LUMA. The drop in performance is attributed to minor load imbalances. From our previous work on volumetric coupling between LBM and DEM codes, we are expecting that the developed coupling framework will scale well up to 8 nodes per solver.

IV. CONCLUSIONS AND FUTURE WORK

To prepare for simulations of multi-scale problems, this project has developed coupling 2 frameworks between the macroscopic finite volume method and the mesoscopic lattice Boltzmann method first, and the mesoscopic lattice Boltzmann method and the microscopic molecular dynamics method, which are implemented using the Code_Saturne, LUMA and LAMMPS solvers, respectively. For each coupling framework, we have presented the coupling methodology, model verification and parallel performance analysis. In both cases, the coupling

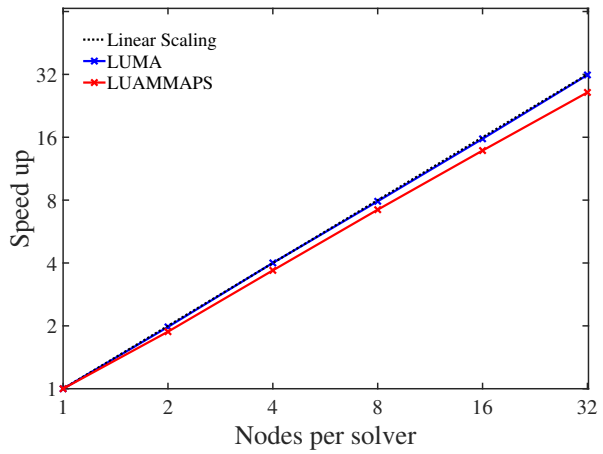

Figure 16. Parallel performance of the LBM-DEM coupling framework; strong scaling test

results are consistent with previous studies and expectations for parallel calculations. This work lays a solid foundation for later multi-scale simulations for engineering applications.

Based on the present work, some optimisations will be performed in the near future: 1) a different time step coupling scheme will be studied to fully take advantage of the implicit method in Code_Saturne; 2) complicated coupling interfaces will be tested, such as having multiple LUMA domains entirely inside the Code_Saturne domain; and 3) for application to real-world engineering problems, the scaling tests will be repeated on larger numbers of nodes. The coupling of LUMA with LAMMPS must be extended to three-dimensional non-spherical particles for having an efficient platform for modeling industrial scale fluid-particle systems.

REFERENCES

[1]E. Weinan and B. Engquist. Multiscale modeling and computation. *Notices of the AMS*, 50(9):1062–1070, 2003.

[2]Z.X. Tong, Y.L. He, and W.Q. Tao. A review of current progress in multiscale simulations for fluid flow and heat transfer problems: The frameworks, coupling techniques and future perspectives. *Int. J. of Heat and Mass Trans.*, 137:1263–1289, 2019.

[3]Y.L. He and W.Q. Tao. Multiscale simulations of heat transfer and fluid flow problems. *J. of Heat Trans.*, 134(3), 2012.

[4]H.B. Luan, H. Xu, L. Chen, D.L. Sun, and W.Q. Tao. Numerical illustrations of the coupling between the lattice boltzmann method and finite-type macro-numerical methods. *Num. Heat Trans., Part B: Fundamentals*, 57(2):147–171, 2010.

[5]M.R. Salimi, M. Taeibi Rahni, and F. Jam. Pore-scale simulation of fluid flow passing over a porously covered square cylinder located at the middle of a channel, using a hybrid mrt-lbm–fvm approach. *Theo. and Comput Fluid Dyn.*, 29(3):171–191, 2015.

[6]A. Dupuis, E.M. Kotsalis, and P. Koumoutsakos. Coupling lattice boltzmann and molecular dynamics models for dense fluids. *Phys. Rev. E.*, 75(4):046704, 2007.

[7]H. Wittenberg and P. Neumann. Transient two-way molecular-continuum coupling with openfoam and mamico: A sensitivity study. *Computation*, 9(12):128, 2021.

[8]Y. Fournier, J. Bonelle, C. Moulinec, Z. Shang, A.G. Sunderland, and J.C. Uribe. Optimizing Code_Saturne computations on Petascale systems. *Comput. & Fluids*, 45(1):103–108, 2011. 22nd International Conference on Parallel Computational Fluid Dynamics (ParCFD 2010).

[9]A.R.G. Harwood, J. O'Connor, J. Sanchez Muñoz, M. Camps Santasmasas, and A.J. Revell. LUMA: A many-core, fluid–structure interaction solver based on the Lattice-Boltzmann method. *SoftwareX*, 7:88–94, 2018.

[10]S. Chen and G.D. Doolen. Lattice boltzmann method for fluid flows. *Ann. Rev. of Fluid Mech.*, 30(1):329–364, 1998.

[11]A.P. Thompson, H.M. Aktulga, R. Berger, D.S. Bolintineanu, W.M. Brown, P.S. Crozier, P.J. in 't Veld, A. Kohlmeyer, S.G. Moore, T.D. Nguyen, R. Shan, M.J. Stevens, J. Tranchida, C. Trott, and S.J. Plimpton. LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales. *Comp. Phys. Comm.*, 271:108171, 2022.

[12]P.A. Cundall and O.D.L. Strack. A discrete numerical model for granular assemblies. *Geotechnique*, 29(1):47–65, 1979.

[13]Y. Fournier. Massively parallel location and exchange tools for unstructured meshes. *Int. J. of Comput. Fluid Dyn.*, 34(7-8):549–568, 2020.

[14]Parallel location and exchange (ple) documentation. https://www.code-saturne.org/documentation/ple-2.0/html/index.html Accessed Jun 2021.

[15]M. Camps Santasmasas, X. Zhang, B. Parslew, G.F. Lane-Serff, J. Millar, and A.J. Revell. Comparison of lattice boltzmann and navier-stokes for zonal turbulence simulation of urban wind flows. *Fluids*, 7(6):181, 2022.

[16]L. Chen, Y.L. He, Q. Kang, and W.Q. Tao. Coupled numerical approach combining finite volume and lattice boltzmann methods for multi-scale multi-physicochemical processes. *J. of Comput. Phys.*, 255:83–105, 2013.

[17]L. Chen, Y.L. Feng, C.X. Song, L. Chen, Y.L. He, and W.Q. Tao. Multi-scale modeling of proton exchange membrane fuel cell by coupling finite volume method and lattice boltzmann method. *Int. J. of Heat and Mass Trans.*, 63:268–283, 2013.

[18]U. Ghia, K.N. Ghia, and C.T. Shin. High-re solutions for incompressible flow using the navier-stokes equations and a multigrid method. *J. of Comput. Phys.*, 48(3):387–411, 1982.

[19]A. De Rosis. Non-orthogonal central moments relaxing to a discrete equilibrium: A d2q9 lattice boltzmann model. *E.P.L.*, 116(4):44003, 2017.

[20]S. Plimpton. Fast parallel algorithms for short-range molecular dynamics. *J. Comput. Phys.*, 117(1):1–19, 1995.

[21]Y.H. Tang, S. Kudo, X. Bian, Z. Li, and G.E. Karniadakis. Multiscale universal interface: a concurrent framework for coupling heterogeneous solvers. *J. Comput. Phys.*, 297:13–31, 2015.

[22]S.K. Kang Y.A. and Hassan. A comparative study of direct-forcing immersed boundary-lattice boltzmann methods for stationary complex boundaries. *Int. J. Num. Meth. Fluids*, 66(9):1132–1158, 2011.

[23]D. Wan and S. Turek. Direct numerical simulation of particulate flow via multigrid fem techniques and the fictitious boundary method. *Int. J. Numer. Meth. Fluids*, 51(5):531–566, 2006.

[24]Z.G. Feng and E.E. Michaelides. The immersed boundary-lattice Boltzmann method for solving fluid–particles interaction problems. *J. Comput. Phys.*, 195(2):602–628, 2004.

Acknowledgement:
This work was funded under the embedded CSE programme of the ARCHER2 UK National Super-computing Service (https://www.archer2.ac.uk/ecse/).