eCSE06-01: Hybrid Atomistic-Continuum Simulations of Boiling Across Scales

M. Magnini[1], G. Gennari[1], E. Smith[2], G. Pringle[3]

[1] University of Nottingham, NG7 2RD Nottingham, UK
[2] Brunel University London, UB8 3PH Uxbridge, UK
[3] EPCC, University of Edinburgh, EH9 3FD Edinburgh, UK

Abstract. Many fluidic processes in science and engineering are underpinned by the two-way interactions between physics happening at the micro-scale and the resulting behaviour observed at the macroscale. One such example is the flow of a fluid over a rough surface, where the bulk fluid motion is set by a large scale force such as a pressure gradient, while the microscale surface topography determines the local shear stress and the resulting drag. Another example is boiling on typical metallic surfaces, where the boiling dynamics depend on near-surface molecular interactions that trigger nucleation, but also on external forces such as buoyancy and drag/lift forces which determine the thickness of thermal boundary layers and the bubble detachment and departure from the heated surface. These are examples of multiscale problems and are very difficult to study at a fundamental level due to the separation of scales. In this eCSE project, we have coupled two popular opensource simulation software, LAMMPS for Molecular Dynamics and OpenFOAM for continuum-scale (CS) computational fluid dynamics (CFD) simulations. This coupling uses a domain decomposition framework where MD resolves the near-wall region where molecular interactions are important and CFD resolves the bulk flow. To showcase the capabilities of the coupled framework, we have studied two traditional benchmark cases, a Couette flow and a pool-boiling case. The results achieved show that the coupling of MD and CFD is successful from both algorithmic and physics perspective, and thus the modules released on ARCHER2 can be utilised by users to develop their own multiscale atomistic-continuum frameworks.

Keywords: multiscale, simulation, boiling, atomistic, eCSE, ARCHER2.

1. The MD-CFD coupling principle

Molecular Dynamics solves the Newton's law of motion for a system of N particles, formulated as follows,

$$m_i \ddot{\boldsymbol{r}}_{\boldsymbol{i}} = \boldsymbol{F}_{\boldsymbol{i}} \qquad \text{for all i in N},$$

where m_i is the mass of the i-th molecule, $\ddot{\boldsymbol{r}}_{\boldsymbol{i}}$ is its acceleration vector and $\boldsymbol{F}_{\boldsymbol{i}}$ the overall force acting on it due to the surrounding molecules. The force is calculated from the pairwise electrostatics interactions which approximate quantum mechanics. Perhaps the most widely studied molecular fluid (and the one used in this work) is based on the Lennard-Jones potential,

$$\boldsymbol{F}_{\boldsymbol{i}} = \sum_{i \neq j}^{N} \nabla \Phi_{i,j}\,, \qquad \Phi_{i,j} = 4\epsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}\right],$$

where ϵ_{ij} and σ_{ij} are the energy and the distance parameters between two generic atoms *i* and *j*, respectively, at distance r_{ij}. By solving the equation of motion over time for each of the N atoms, the position and velocity is known at each time instant.

Meanwhile, CFD solves the equations of motion based on the continuum assumption. The general transport equation solved in CFD is the following advection-diffusion equation:

$$\frac{\partial(\rho\varphi)}{\partial t} + \nabla \cdot (\rho \boldsymbol{U} \varphi) = \nabla \cdot (\Gamma \nabla \varphi) + S_{\varphi},$$

where φ is the scalar being transported, e.g. momentum or enthalpy, ρ is the fluid density, t is time, $\boldsymbol{U}$ is the fluid speed, Γ the diffusion coefficient for φ, and S_{φ} a generic source term which implement the creation or destruction of φ. MD and CFD can be spatially coupled by having them solving their own regions in space, see for example Fig. 1, which is a setup referred to as *domain decomposition* coupling. In this kind of coupling, MD typically resolves the near-wall region where the molecular interactions between the fluid and the solid become important, for example to determine the shear stress at the wall or during the nucleation of bubbles in a superheated liquid upon boiling. The CFD solver resolves the bulk flow, where molecular and continuum behaviour can be considered identical. However, for the coupling to take place, the two descriptions must communicate at the respective domain boundaries. The CFD fields must be converted in MD

constraints to be applied to the molecules in the MD solver, for example as a constraint force which changes the average speed of molecules to match the CFD values. On the other hand, the MD data must be averaged in both time and space to be imposed on the CFD as boundary conditions for the solution of density, momentum and energy transport equations. Since both MD and CFD solutions evolve in time, this exchange of information must occur at least every CFD timestep. While different strategies exist for temporal coupling, the present work implemented a straightforward *intermittent coupling*, where one CFD time-step is run every *n* MD time-steps, and the coupled fields updated at any exchange of information.
In the present project, OpenFOAM was adopted as CFD solver due to its popularity and to the fact that it is available as a module in ARCHER2. The ESI version 2106 was chosen as it was recent enough to be useful, but not too close to a development version. LAMMPS was adopted for the MD, as it is also available in ARCHER2. By choosing only software available in ARCHER2, the user does not need to compile external software, but can simply load the relevant modules. The coupling of the two software is handled by the CPL library [1] developed by Dr Smith and publicly available on GitHub. This has also been deployed in ARCHER2 and available as a module as explained in the next section.

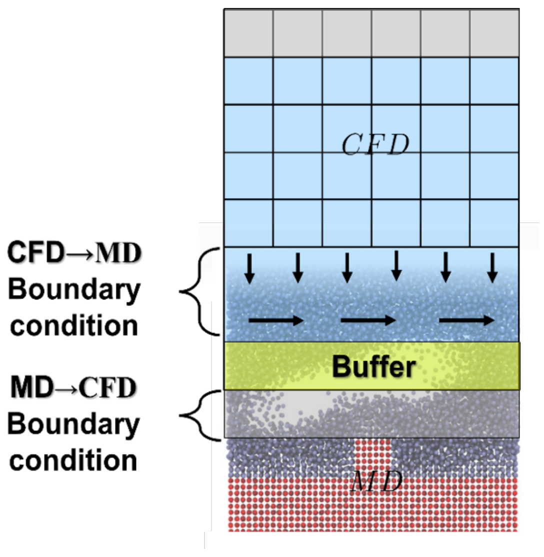

Figure 1. Schematic of MD-CFD domain decomposition coupling.

2. The coupling framework in ARCHER2

The CPL Library and all associated application codes are accessible to ARCHER2 users, along with instructions [2] on how to employ them via the module system. There is also a guide on how to install them by hand. CPL library itself is just a topology mapping and information exchange library [6], focused on the setup of Figure 1. It provides commands to initialise both codes to exchange information (CPL_init), map topologies (CPL_ setup_md/cfd) and send/receive information (CPL_send/recv). The specific code for OpenFOAM [7] and LAMMPS [8] to work with CPL library are then available in APPS, which handle packing information to send and unpacking in both codes, applying constraints to LAMMPS and boundary conditions to OpenFOAM. In addition, solvers using these are provided as examples in OpenFOAM and example input scripts to switch on the CPL fix in LAMMPS.

2.1 Availability of Software

CPL library can be built from source, obtained as a Docker container and, as a result of this eCSE project, the associated applications can be employed via centrally-installed third-party modules, by running the following four commands on ARCHER2:

```
module load other-software
module load cpl-openfoam
source $FOAM_CPL_APP/SOURCEME.sh
module load cpl-lammps
```

The first command enables access to third-party modules on ARCHER2, where these modules are supported by non-ARCHER2 staff. In this case, the associated modules will be maintained by Edward Smith and Gavin Pringle. Indeed, the detailed instructions on how these modules were created and maintained are included within the hpc-uk github project, specifically [3].

The second and third commands set the user's environment variables such that both the CPL library and OpenFOAM (v2106), employed via the openfoam/com/v2106 module, can be employed by the user. Users can now employ the CPL's compiler wrapper commands for C (`cplc++`) and Fortran (`cplf90`) and the package for Python (`from cplpy import CPL`), enabling any two applications to be coupled together.

The fourth and final command above enables the users to employ the CPL-enabled version of LAMMPS, which has been built also using GCC v10.3, along with the modules cray-fftw and cray-python, and employed the most-recent stable release of LAMMPS at the time of writing (Version

23 June 2022 - Update 3). The continuous integration testing of CPL library tracks the latest stable LAMMPS so future versions should be supported.

Once these four commands are run, the user can then access the CPL library tutorials, to investigate how to write their own dummy applications, in C, Fortran or Python, and couple them with the CPL library. Users can employ the provided source code and executables or create their own codes and executables using the CPL library compiler wrappers. Further, there are many other tutorials available which are designed to show the user how to couple a dummy MD code to OpenFOAM and a dummy CFD code to LAMMPS and, finally, how to couple OpenFOAM and LAMMPS, which example simulations throughout. Again, detailed instructions on how to use the tutorials, along with the aforementioned installation instructions, can be found via [2].

2.2 Software Sustainability

CPL library aims to be sustainable, aiming to be easily maintainable, covered by both unit and integration tests and employing some form of continuously integration (CI) for test automation and deployment. A suite of unit tests already existed to check various topology setups, and for granular coupling examples, these have been extended in this work to include more fundamental tests, including CPLTestFoam – which checks the linking and exchange through sends/recv of dummy information and CPLTestSocketFoam which checks the boundaries are set correctly by this dummy information, both coupled to a minimal script. As part of this project, the CI framework has been changed from Travis CI to Github to provide better integration. The possibility of CI on ARCHER2 was also explored for module deployment, but requires GitLab runners.

Typically, coupled MPI applications have to be rewritten to employ a single MPI_Comm_world global communicator, where each application's previous global communicator is then created by splitting the new global communicator. This manner of running coupled applications is known as the ***shared*** communicator mode, and requires changes to both codes to replace MPI_Comm_world with part of it using MPI_comm_split. This is a problem for sustainability as patched versions of both coupled codes must be created. In LAMMPS, the top level lammps.cpp is changed on the fly, using a Python script, tested by CI which tracks the latest stable branch. The OpenFOAM's communication library, Pstream, contains all MPI commands and so has to be replaced with CPL's version of Pstream, called CPLPstream, during build. A major component of the work undertaken in this eCSE was to update the version of OpenFOAM from 3.0.2 to incorporate a refactored Pstream implemented by ESI in v2106. The supported version of OpenFOAM has been designed for v2106 but a softlink system is used so Pstream can be swapped to another version of OpenFOAM. The older version of Pstream is present in both the OpenFOAM Foundation (openfoam.org) branch and the Foam extend versions, so most versions of OpenFOAM should now work with minimal changes. To enable the required changes to build a chosen version of OpenFOAM with CPL library, we provide detailed instructions on how to do this on ARCHER2 [2].

An alternative to ***shared*** communication is to use MPI_Comm_connect and MPI_Intercomm_merge, so both applications retain their original MPI_Comm_World communicators and create a new intercom by opening a port. This is termed the ***distinct*** communicator model and avoids any changes to either code, provided MPI_intercomms is supported (since the latest patch by Cray, towards the end of the project, this now works on ARCHER2). Users can choose to run the CPL library in *either* mode on ARCHER2.

3. Results

3.1 Couette flow

Unsteady Couette flow is a traditional benchmark case for numerical solvers of fluid flow equations, because it is one of the few cases for which we have an analytical solution. In our setup, see Fig. 2, the top wall moves to the right with constant speed U_w, the bottom wall is stationary, the right and left boundaries are periodic. In the coupled example, MD solves the near-wall region near the bottom wall and CFD solves the remainder of the domain, including the top wall. The MD simulation is run using LAMMPS. The CFD simulation is run using the self-developed OpenFOAM solver CPLicoFoam, which is a modified version of OpenFOAM's built-in icoFoam and solves the continuity and momentum equation for an incompressible flow:

$$\nabla \cdot \boldsymbol{U} = 0,$$

$$\frac{\partial \boldsymbol{U}}{\partial t} + \nabla \cdot (\boldsymbol{U}\boldsymbol{U}) - \nabla \cdot (\nu \nabla \boldsymbol{U}) = -\nabla p,$$

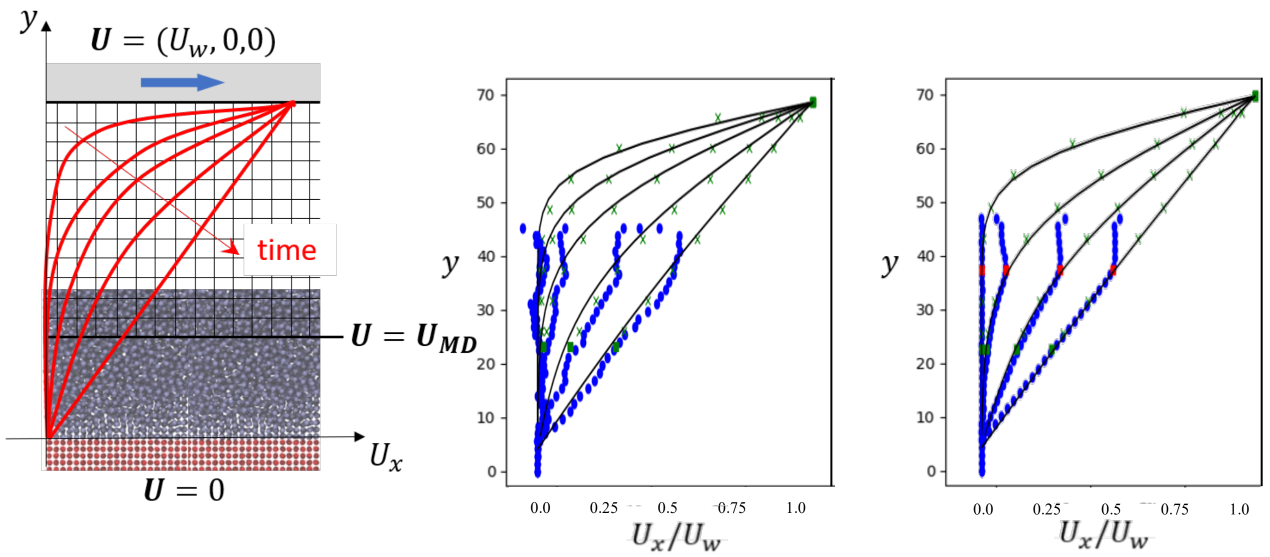

Figure 2. a) Schematic of coupled Couette flow, b) Couette flow results for example case [9] and c) scaled case on 1024 cores (~10 million atoms) with OpenFOAM on 4 cores. Velocity profile for CFD (green crosses, square halos), MD (blue circles, constrained cell in red) and analytical solution (black line).

where $\boldsymbol{U}$ is the velocity, p the pressure (rescaled by the liquid density) and ν the kinematic viscosity of the fluid. In the OpenFOAM setup, the bottom boundary of the domain is named *CPLReceiveMD* and will receive velocity data from the coupled MD software, sent using CPL_send from LAMMPS. At time instant $t = 0$, a zero velocity is set to the whole domain. Over time, the velocity is expected to converge towards a steady-state situation as depicted in Fig. 2, where the velocity profile exhibits a linear variation along the vertical direction from a zero value at $y = 0$ to U_w on the top wall. The results of the coupled LAMMPS-OpenFOAM simulation are displayed in Fig. 2b) and c), where it can be seen that the CFD solution is effective in driving the flow in the MD domain as time elapses. The coupled solution matches the analytical ones quite well at all time instants considered. The green squares in Fig. 2b) ($y \approx 23$) indicate the CFD solution at the coupled boundary, which coincides with the local MD velocity value imposed to the CFD domain. In the overlap region, $23 \leq y \leq 47$, both MD and CFD solution exist and they compare relatively well, thus indicating that the coupling is successful. Both the MD and CFD regions are of size 45.3 in the y direction, with 8 cells in the CFD of which 4 overlap the MD, so the total domain is 68.0 in y. The full set-up for the test case in Fig 2b) can be found at [9]. This example uses 2 processors in both domains to demonstrate the minimum coupled example which uses multiple processes, but has noticeable noise in the MD results due to its small size. To demonstrate the improved performance for larger runs, the results shown in Fig 2c) are for an example scaled up to 1024 cores on ARCHER2, simulated with 32 cells in x and z where the domain size is scaled up to 537.5 reduced units in both CFD and MD (just over 12 million molecules in the MD). Scaling on CPL library has been presented previously so this work simply tested large MD jobs ran as expected.

3.2 Nucleate boiling

When a fluid is heated up above its boiling point, vapour bubbles begin nucleating from the hot surface, as experienced in our everyday life when boiling water in a tea kettle. Nucleation occurs due to molecular-scale interactions within the fluid in the proximity of a wall, and thus cannot be captured by CFD models based on continuum-scale transport equations. MD is capable of capturing nucleation from first principles, but is limited to nanoscale size. Coupling MD and CFD for the multiscale simulation of nucleation and boiling is a challenging example to showcase the capability of the present framework. At present, the coupling has been completed as a one-way example from MD to CFD, with the MD that evolves agnostic of CFD and the CFD that receives MD fields of density, velocity and temperature at a defined coupled boundary, with these fields varying as the MD simulation evolves over time. The setup of the coupled MD-CFD simulation of nucleate boiling is shown in Fig. 3. The MD simulation, run using FlowMol [4], models the solid wall with a square cavity and a region of fluid above it. The fluid is Argon at an initial system pressure of about 1.2 MPa, to which it corresponds a saturation temperature of about 120 K. The wall is heated up at about 130 K. The domain is a box of x width 160 nm and a small depth in z to make it essentially a 2D simulation, with periodic boundaries in x and z. A CFD mesh is positioned on top of the MD domain (although the CFD domain could be larger), and the bottom boundary of the CFD mesh identifies the coupled boundary, where CFD fields of density, velocity and temperature will be fed by MD data. In the CFD model, the top boundary is an open boundary and the side boundaries are cyclic. The CFD simulation is run using the OpenFOAM solver CPLinterFoamHardtPhaseChange developed by Dr Magnini, which is based on the work of Municchi et al. [5]. The solver employs a Volume Of Fluid (VOF) method to track the interface between immiscible liquid and vapour phases and solves the volume fraction, continuity, momentum and energy equations formulated as follows:

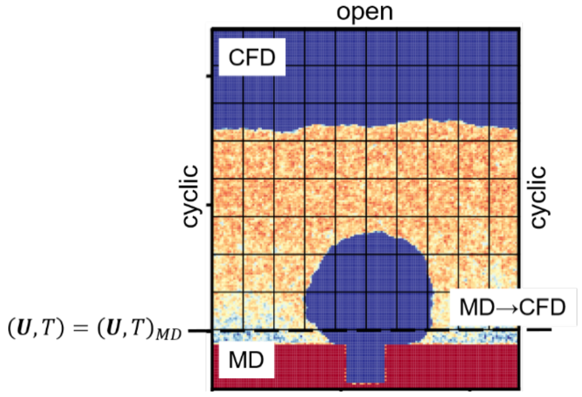

Figure 3. Setup of coupled MD-CFD simulation of nucleate boiling.

$$\frac{\partial \alpha}{\partial t} + \nabla \cdot (\alpha \boldsymbol{U}) = \frac{\dot{\rho}}{\rho}\alpha,$$

$$\nabla \cdot \boldsymbol{U} = \frac{\dot{\rho}}{\rho},$$

$$\frac{\partial (\rho \boldsymbol{U})}{\partial t} + \nabla \cdot (\rho \boldsymbol{U}\boldsymbol{U}) = -\nabla p + \nabla \cdot \boldsymbol{\tau} + \boldsymbol{F}_{\boldsymbol{\sigma}},$$

$$\frac{\partial (\rho c_p T)}{\partial t} + \nabla \cdot (\rho c_p \boldsymbol{U} T) = \nabla \cdot (\lambda \nabla T) - \dot{\rho} h_{lv},$$

where α is the volume fraction of liquid, i.e. the volume of the cell occupied by liquid divided by the volume of the cell, $\dot{\rho}$ is the liquid-to-vapour mass transfer due to evaporation, ρ is the mixture fluid density, $\boldsymbol{\tau}$ is the viscous shear, $\boldsymbol{F}_{\boldsymbol{\sigma}}$ is the surface tension force, c_p the constant pressure specific heat, T the fluid temperature, λ the thermal conductivity and h_{lv} the latent heat. All fluid mixture properties are evaluated as weighted averages of liquid- and vapour-specific properties, $\rho = \rho_l \alpha + \rho_v (1-\alpha)$. The mass transfer due to evaporation is evaluated as:

$$\dot{\rho} = \frac{h_i}{h_{lv}}(T_i - T_{sat})|\nabla \alpha|, \qquad h_i = \frac{2\gamma}{2-\gamma}\left(\frac{\bar{M}}{2\pi \bar{R}}\right)^{1/2} \frac{\rho_v h_{lv}^2}{{T_{sat}}^{3/2}},$$

where γ is the evaporation coefficient, $\bar{M}$ is the molecular mass of the fluid and $\bar{R}$ the universal gas constant. At the coupled boundary, the MD velocity and temperature is imposed to the CFD. The liquid volume fraction at the boundary, necessary for the bubble to enter the CFD domain, is calculated from the MD density field as follows:

$$\rho_{MD} \to \alpha = \begin{cases} 1, & if \rho_{MD} > 0.5(\rho_{l,CFD} + \rho_{v,CFD}) \\ 0, & if \rho_{MD} \leq 0.5(\rho_{l,CFD} + \rho_{v,CFD}) \end{cases}$$

where ρ_{MD} is the MD density and $\rho_{l,CFD}$, $\rho_{v,CFD}$ are CFD liquid and vapour specific densities. Figure 4 shows the results of the coupled simulations. The left panel displays CFD and MD fields of density side by side, depicted at the end of the simulation. It can be seen that the qualitative agreement on the bubble size and shape between MD and CFD is excellent, except for the layer of vapour placed on the top of the liquid in the MD simulation to prevent pressure build-up, which is not included in the CFD model because it is not necessary, due to the open boundary. The right panel plots the mass of vapour in both CFD and MD, evaluated at $y > 0$ in the region where the CFD exists. Again, the agreement is excellent, showing that the coupled MD-CFD framework can achieve the same results as the MD alone in the region above the wall. The full set-up for this test case can be found at [9].

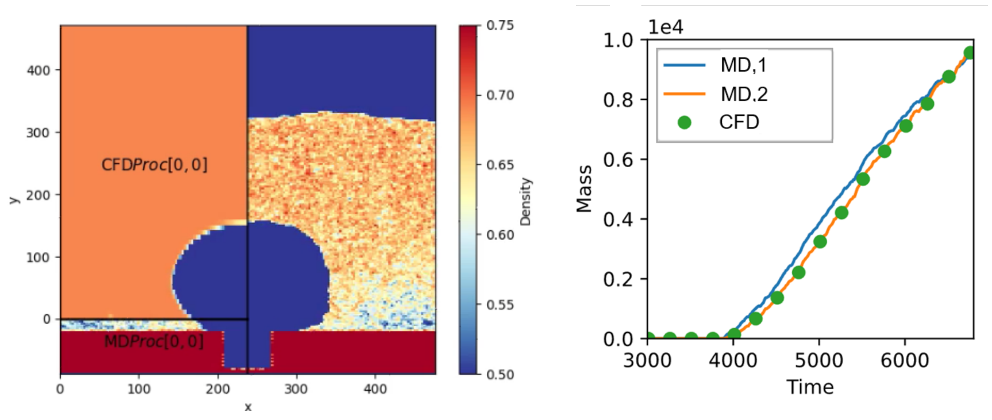

Figure 4. Coupled simulation of pool boiling of Argon. [left] The left-half displays the CFD density field, the right-half side the MD density field. The unit length is 0.34 nm; a MD density of 0.7 corresponds to 1160 kg/m^3. [right] Total mass of vapour in the CFD and MD ($y > 0$) domains.

4 Conclusions

In this project, we have coupled two popular opensource Molecular Dynamics (LAMMPS) and Computational Fluid Dynamics (OpenFOAM) solvers to perform hybrid atomistic-continuum simulations of single-phase and two-phase boiling flows, with a domain decomposition strategy. The coupled framework extends MD capabilities to larger domains and CFD capabilities towards incorporating molecular-level interactions. The coupled framework is available as a module in ARCHER2 [2], whereas the whole library with test cases and tutorials is publicly available on Github [6].

Acknowledgment

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

References

[1] E. R. Smith, D.J. Trevelyan, E. Ramos-Fernandez, A. Sufian, C. O'Sullivan, D. Dini, CPL library - A minimal framework for coupled particle and continuum simulation, Comput. Phys. Commun. 250 (2020) 107068.
[2] https://www.cpl-library.org/docs/Running_on_ARCHER2.pdf
[3] https://github.com/hpc-uk/build-instructions/tree/main/apps/CPL-OpenFOAM-LAMMPS
[4] E. R. Smith; A molecular dynamics simulation of the turbulent Couette minimal flow unit. Physics of Fluids 1 November 2015; 27 (11): 115105. https://doi.org/10.1063/1.4935213
[5] F. Municchi, I. El Mellas, O. K. Matar, M. Magnini, Conjugate heat transfer effects on flow boiling in microchannels, Int. J. Heat Mass Transf. 195 (2022) 123166.
[6] https://github.com/Crompulence/cpl-library/
[7] https://github.com/Crompulence/CPL_APP_OPENFOAM/
[8] https://github.com/Crompulence/CPL_APP_LAMMPS-DEV
[9] https://github.com/Crompulence/cpl-library/tree/master/examples/LAMMPS_OPENFOAM