Nucleate Boiling: a coupled MD-CFD case

M. Magnini[1,*], E. R. Smith[2], G. J. Pringle[3], G. Gennari[4]

[1]*Dept. of Mechanical Engineering, University of Nottingham, Nottingham.* E-mail: mirco.magnini@nottingham.ac.uk

[2]*Mechanical and Aerospace Engineering, Brunel University London: edward.smith@brunel.ac.uk*

[3]*EPCC, University of Edinburgh: g.pringle@epcc.ed.ac.uk*

[4]*Dept. of Mechanical Engineering, University of Nottingham, Nottingham*: *gabriele.gennari@nottingham.ac.uk*

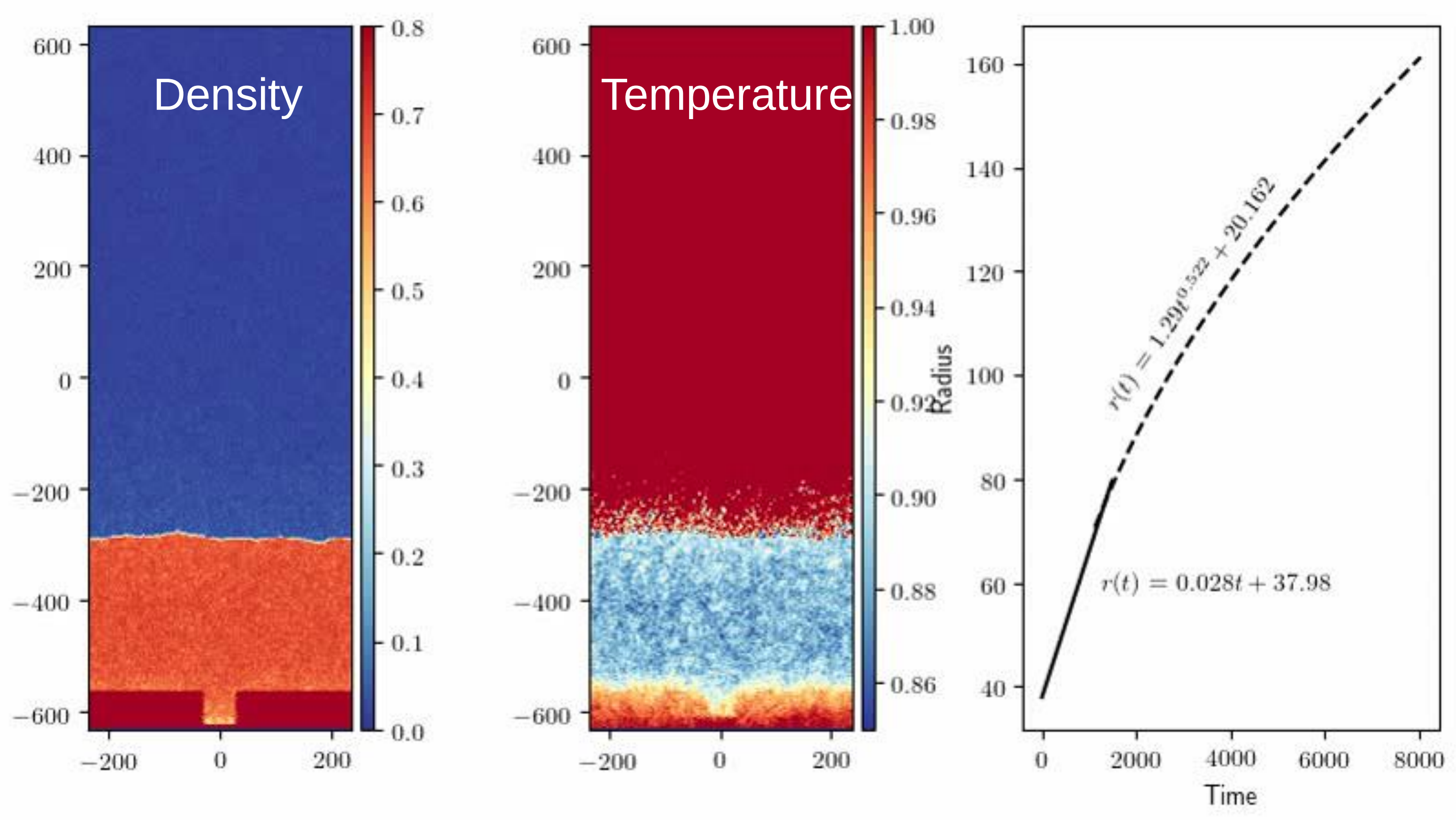
eCSE06-01: "Hybrid Atomistic-Continuum Simulations of Boiling Across Scales"

Outline

- Problem description
- Case setup
- Overview of the configuration files
- Demo
- Results

Problem description

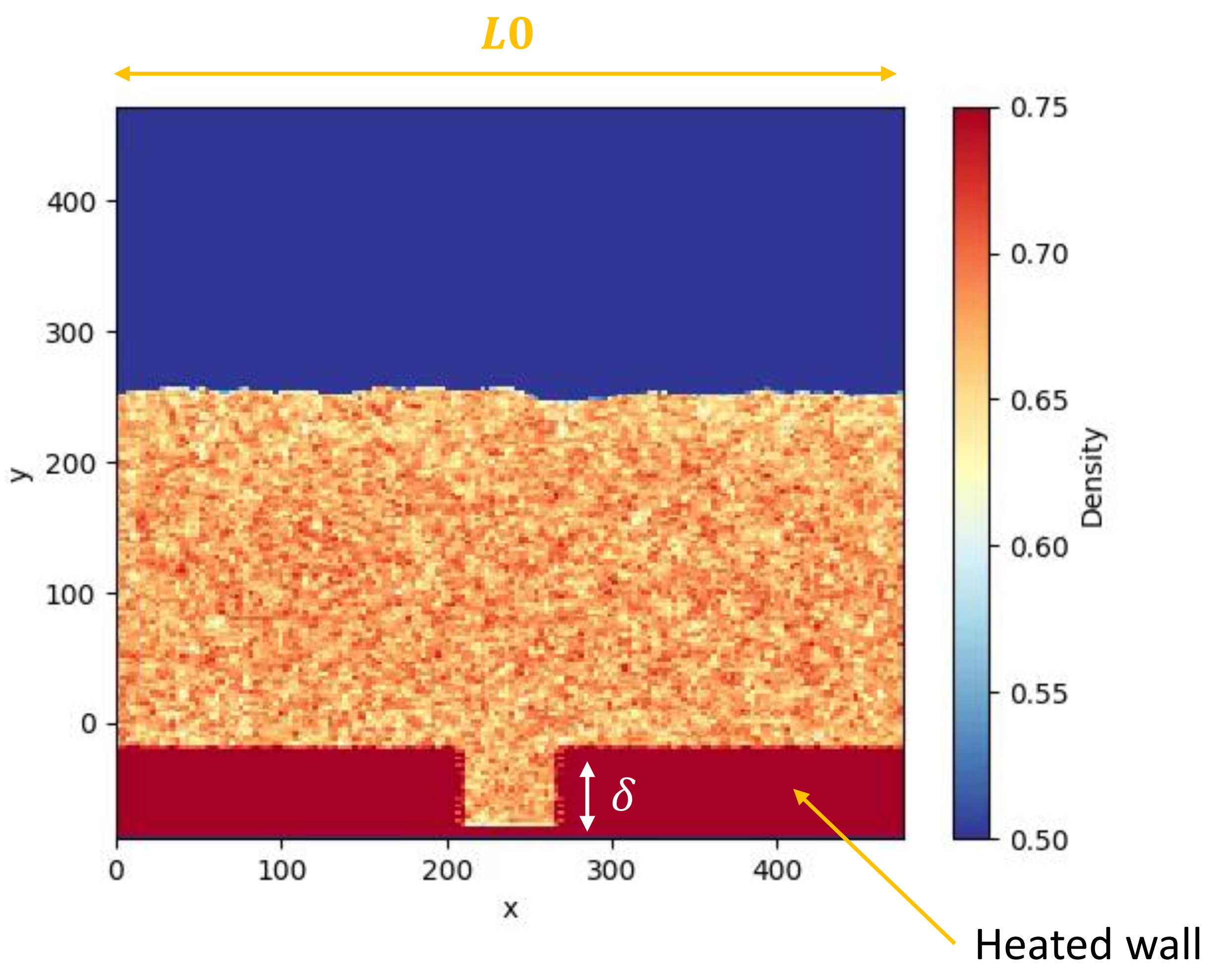
- Nucleate bubble in a heated cavity
- Data generated using flowmol (github.com/edwardsmith999/flowmol)



- Conversion factors
  - Length - $1 = 0.34$ nm
  - Temp. - $1 = 125$ K
  - Density - $0.7 = 1160$ kg m^{-3}

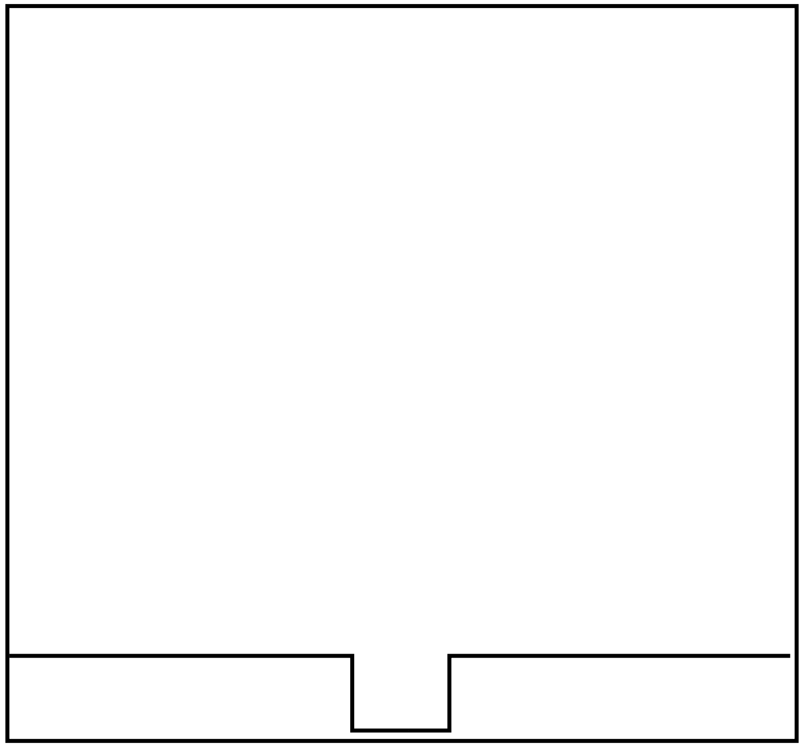
Problem description

- Nucleate bubble in a heated cavity
- Properties and dimensions:
  - Fluid: Argon
  - Domain size, $L0 = 1.6 \times 10^{-7}$ m
  - Cavity size, $\delta = 2.1 \times 10^{-8}$ m



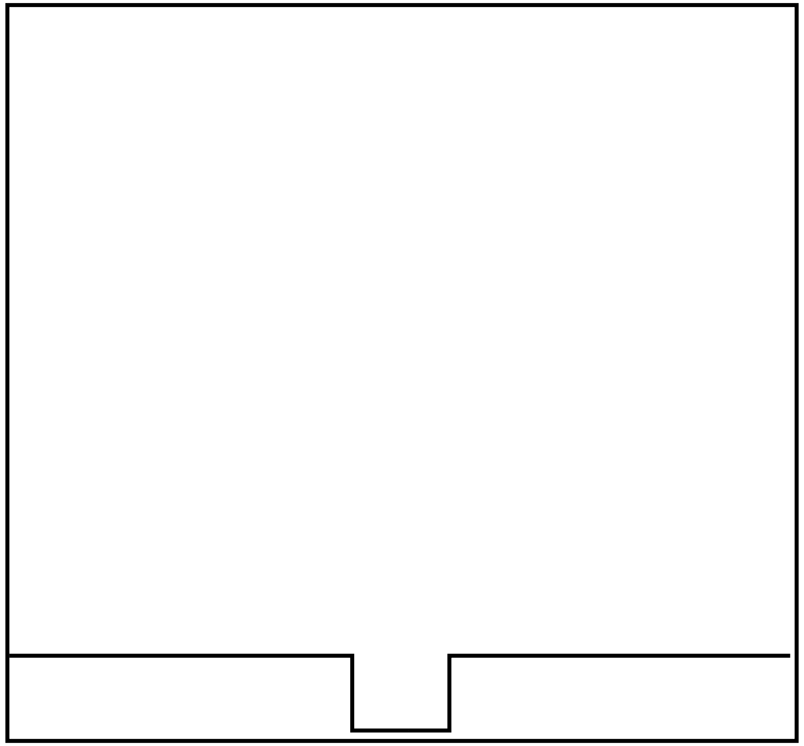
Case setup

- One-way coupled simulation



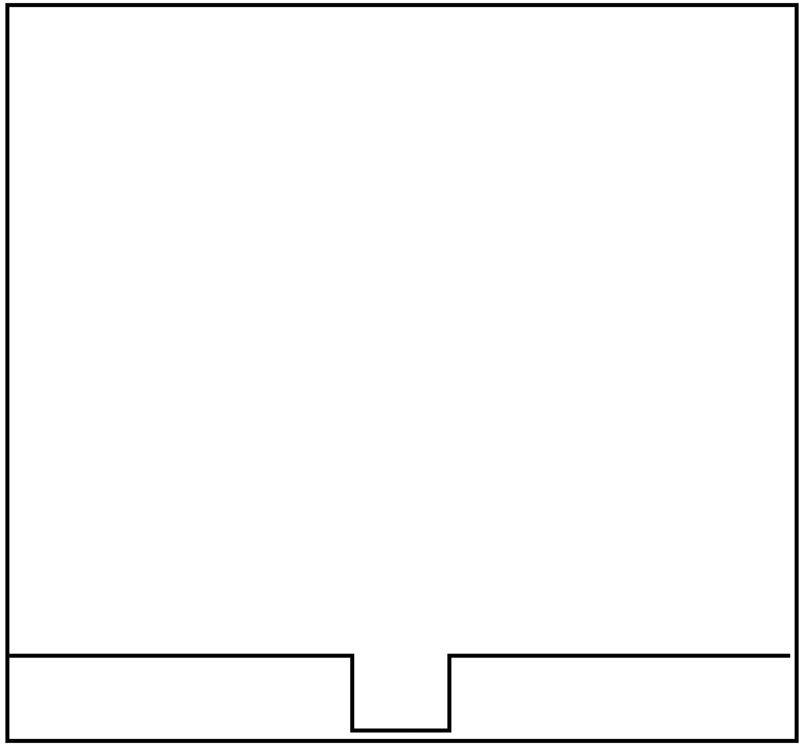
Case setup

- One-way coupled simulation
- MD runs first and data is stored in several snapshots



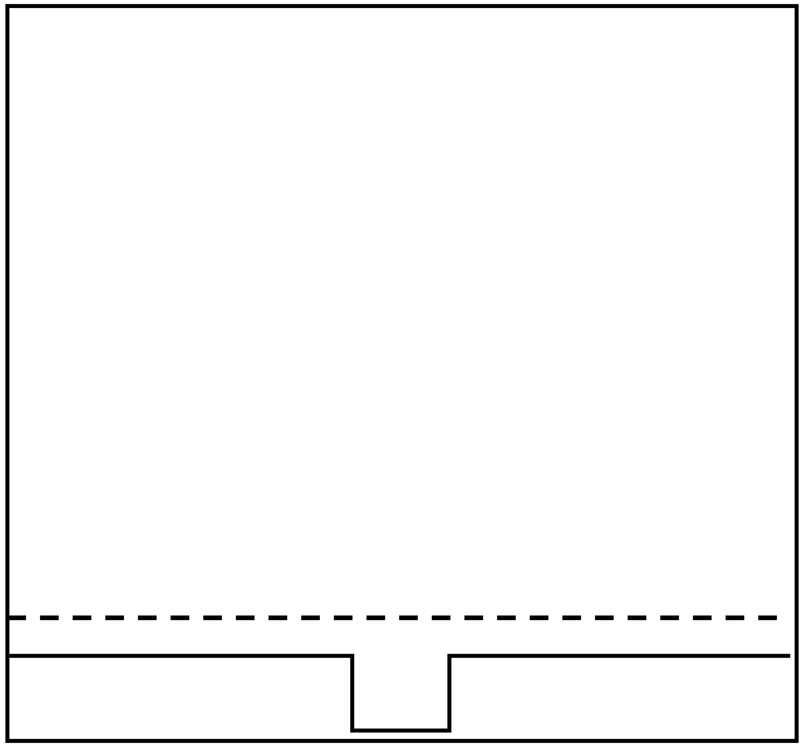
Case setup

- One-way coupled simulation
- MD runs first and data is stored in several snapshots
- CFD is then coupled with MD through a time-dependent boundary condition



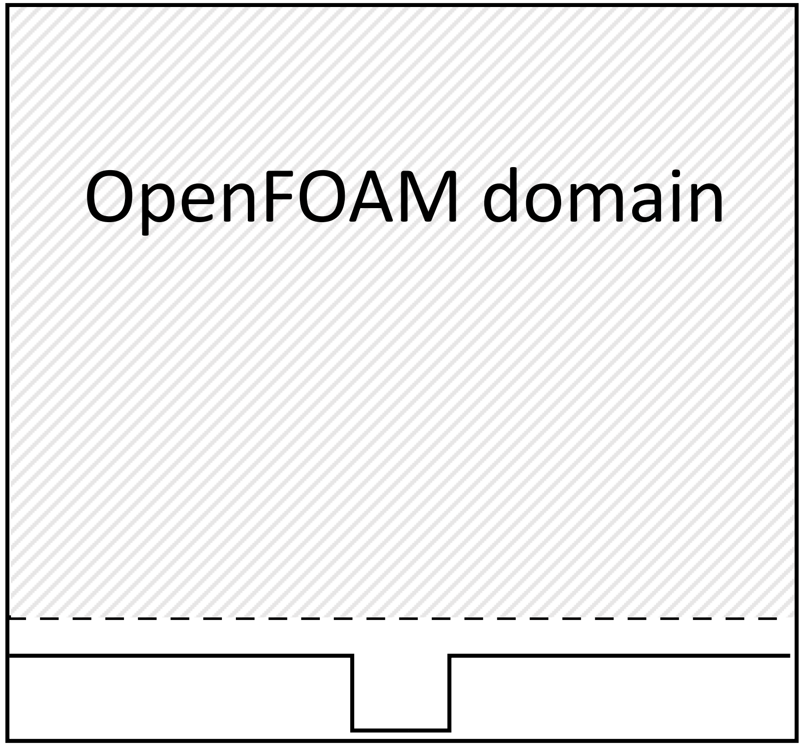
Case setup

- One-way coupled simulation
- MD runs first and data is stored in several snapshots
- CFD is then coupled with MD through a time-dependent boundary condition



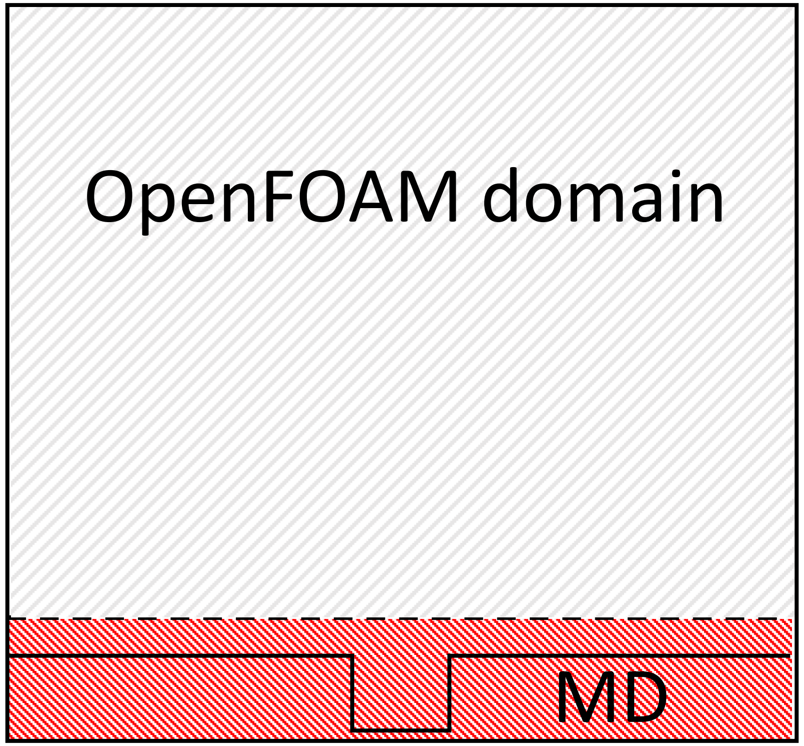
Case setup

- One-way coupled simulation
- MD runs first and data is stored in several snapshots
- CFD is then coupled with MD through a time-dependent boundary condition



Case setup

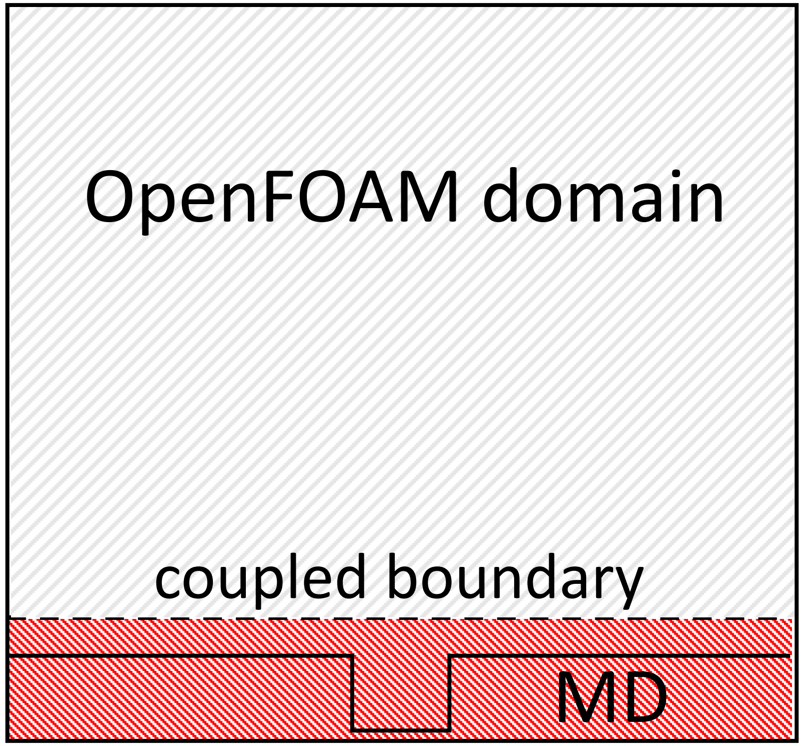
- One-way coupled simulation
- MD runs first and data is stored in several snapshots
- CFD is then coupled with MD through a time-dependent boundary condition



Case setup

- One-way coupled simulation
- MD runs first and data is stored in several snapshots
- CFD is then coupled with MD through a time-dependent boundary condition

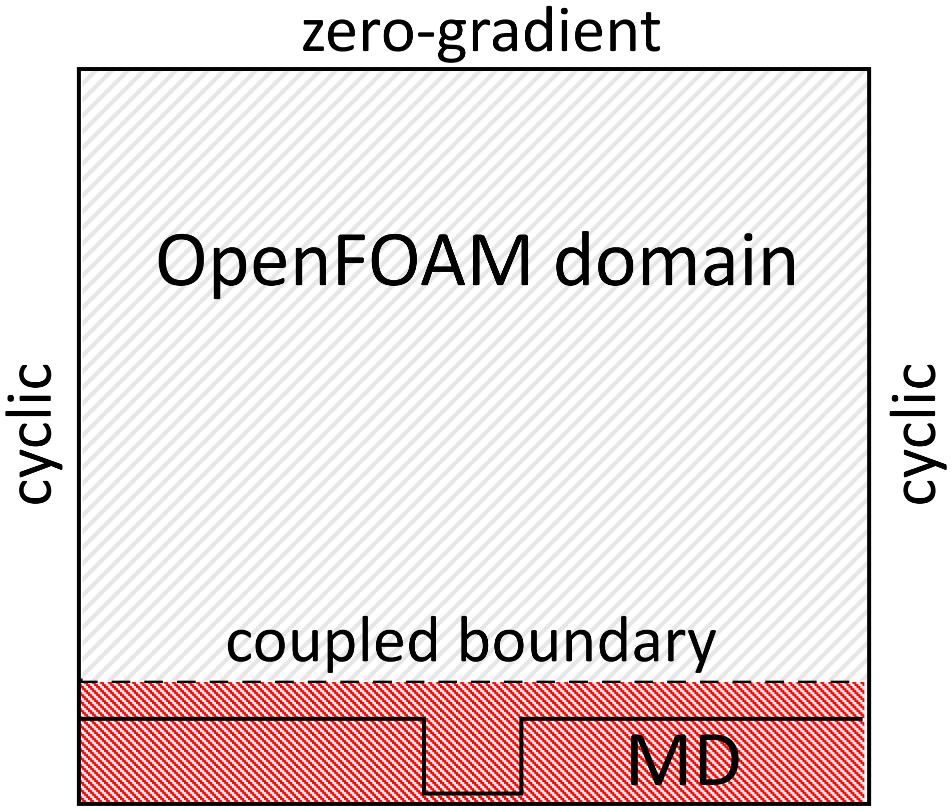
- Coupled boundary:
  - Data is read from the MD snapshots every 10 timesteps
  - Fields: momentum (u, v, w), temperature (T), liquid volume fraction (α_L)
  - The volume fraction can be obtained in several ways, e.g.:
    - $\alpha_L = \frac{\rho - \rho_V}{\rho_L - \rho_V}$ (VOF equation)
    - $\begin{cases} \alpha_L = 1, if\ \rho > 0.5(\rho_L + \rho_V) \\ \alpha_L = 0, if\ \rho \leq 0.5(\rho_L + \rho_V) \end{cases}$



Case setup

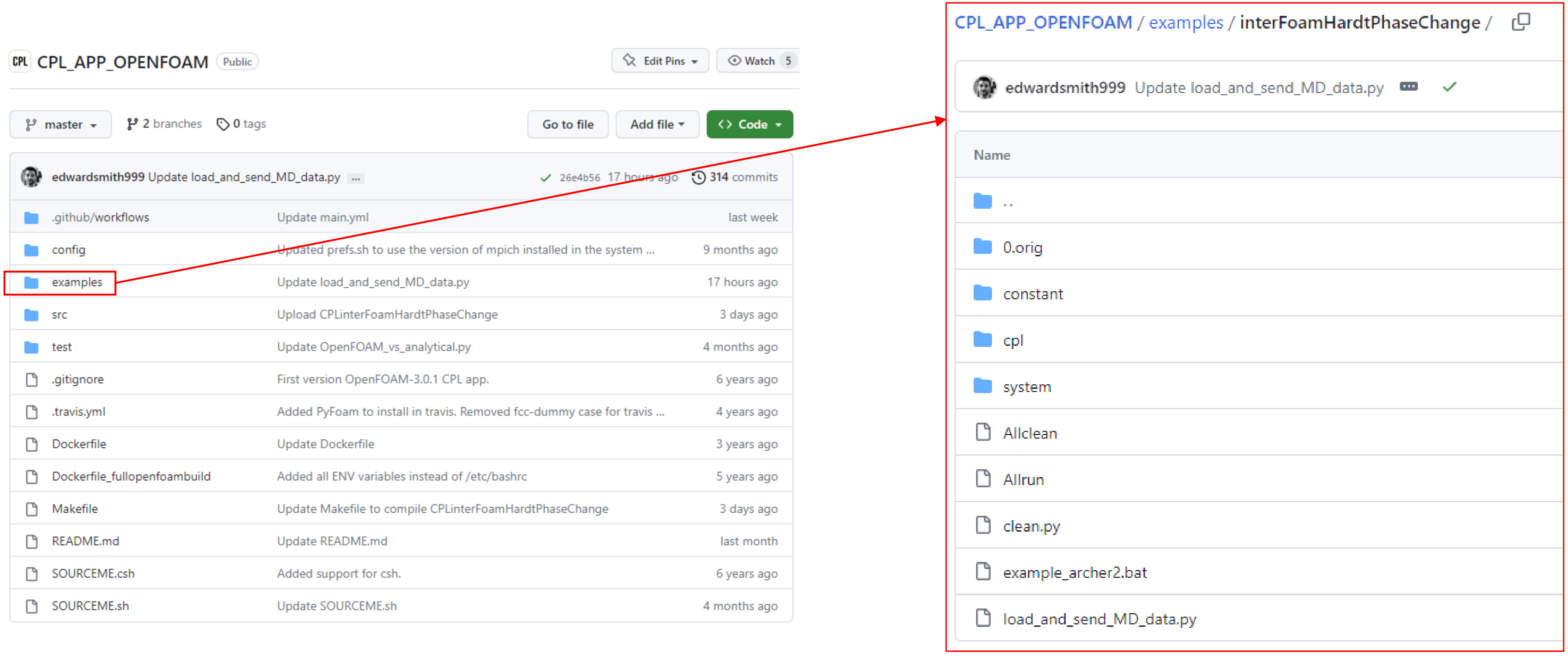
- One-way coupled simulation
- MD runs first and data is stored in several snapshots
- CFD is then coupled with MD through a time-dependent boundary condition

- Coupled boundary:
  - Data is read from the MD snapshots every 10 timesteps
  - Fields: momentum (u, v, w), temperature (T), liquid volume fraction (α_L)
  - The volume fraction can be obtained in several ways, e.g.:
    - $\alpha_L = \frac{\rho - \rho_V}{\rho_L - \rho_V}$ (VOF equation)
    - $\begin{cases} \alpha_L = 1, if\ \rho > 0.5(\rho_L + \rho_V) \\ \alpha_L = 0, if\ \rho \leq 0.5(\rho_L + \rho_V) \end{cases}$



Overview of the configuration files (1/8)

- Example availabe on Github (https://github.com/Crompulence/CPL_APP_OPENFOAM.git)



Overview of the configuration files (2/8)

- pyDataView is used to load MD data (https://github.com/edwardsmith999/pyDataView.git)

pyDataView Public · Watch 4

master · 2 branches · 0 tags · Go to file · Add file · Code

Ed Merge remote-tracking branch 'origin/master' · 88fc2cb 2 weeks ago · 171 commits

.github/workflows	Update main.yml	4 months ago
matlab	Update plot_matlab.m	2 years ago
misclib	Added shebang which avoids ascii error, hopefully allows python2 and ...	2 years ago
postproclib	Merge remote-tracking branch 'origin/master'	2 weeks ago
.gitignore	Initial commit	8 years ago
LICENSE	Initial commit	8 years ago
README.md	Update README.md	4 months ago
logo.icns	Add files via upload	4 months ago
pyDataView.py	Removed print statements and changed default directory for pydatavie...	5 years ago
pyDataView.spec	Update pyDataView.spec	4 months ago
pyDataView_apple1.spec	Create pyDataView_apple1.spec	4 months ago
pyDataView_apple2.spec	Rename pyDataView_apple.spec to pyDataView_apple2.spec	4 months ago
pyDataView_screenshot.png	added screenshot	8 years ago
pyDataview_final.pdf	Add files via upload	6 years ago
requirements.txt	Update requirements.txt	4 months ago
run_vmd.py	Added run vmd code to reformat temporary files and open vmd	3 years ago
setup.py	Chanegd setup name to pyDataView no viewer	4 months ago

README.md

pyDataView

"A graphical user interface for people who hate graphical user interfaces"

If you know how to read your data by writing Python code, PyDataView simply provides a way to quickly explore the data with a slider/file viewer. Once you see something you like, click Save Script to generate a starting point for further analysis.

Introduction

A Data Viewer GUI written in python, wxpython and matplotlib.

This is a lightweight interface for quick insight into scientific data. Data can be explored as a matplotlib lineplot or contour using sliders to traverse the range of existing records and bins. For more detailed analysis, a figure can be saved, the data output as a csv file or a minimal python script generated. The emphasis of this project is the provision of a simple reader to get data into python, a minimal GUI to get quick insight into that data and generation of python scripts for more detailed analysis.

Quickstart

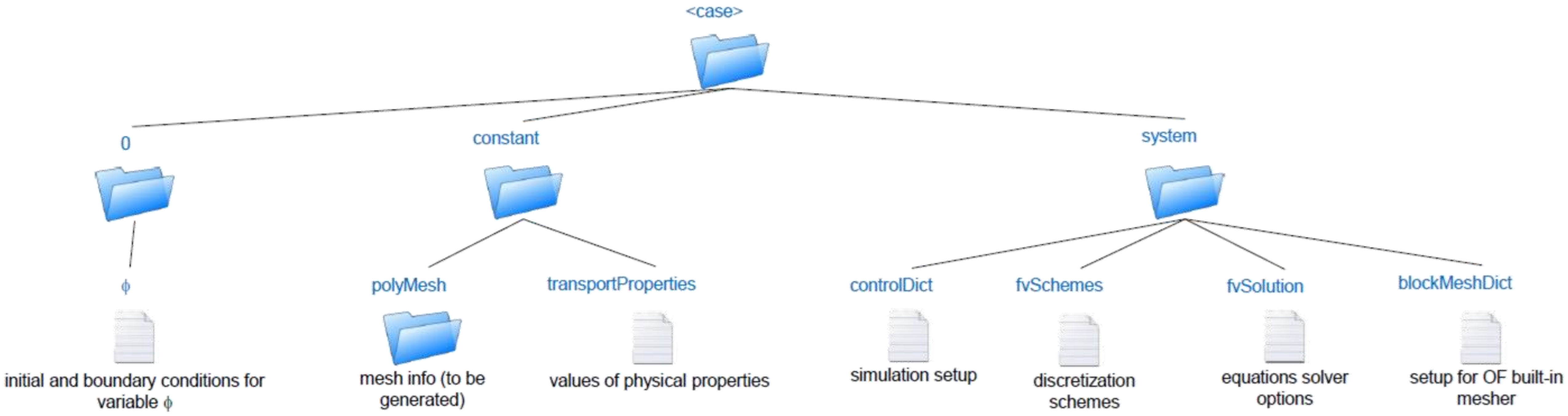
Clone the repository

On linux, you need Python3, numpy, scipy, wxpython, matplotlib and vispy installed. You can then clone the repository and run pyDataView from the command line,

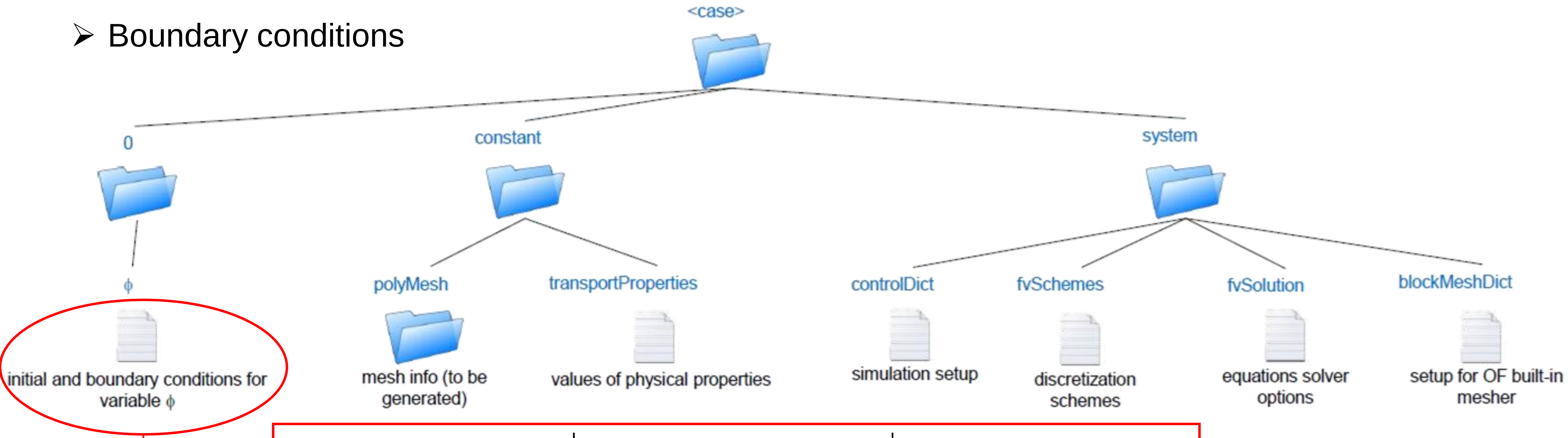
```
python3 pyDataView.py -d ./path/to/dir
```

Overview of the configuration files (3/8)

- Typical folder tree structure of any OpenFOAM simulation



- Typical folder tree structure of any OpenFOAM simulation
- Boundary conditions



In 0.orig/U:

```
boundaryField
{
    CPLReceiveMD
    {
        type        fixedValue;
        value       $internalField;
    }
```

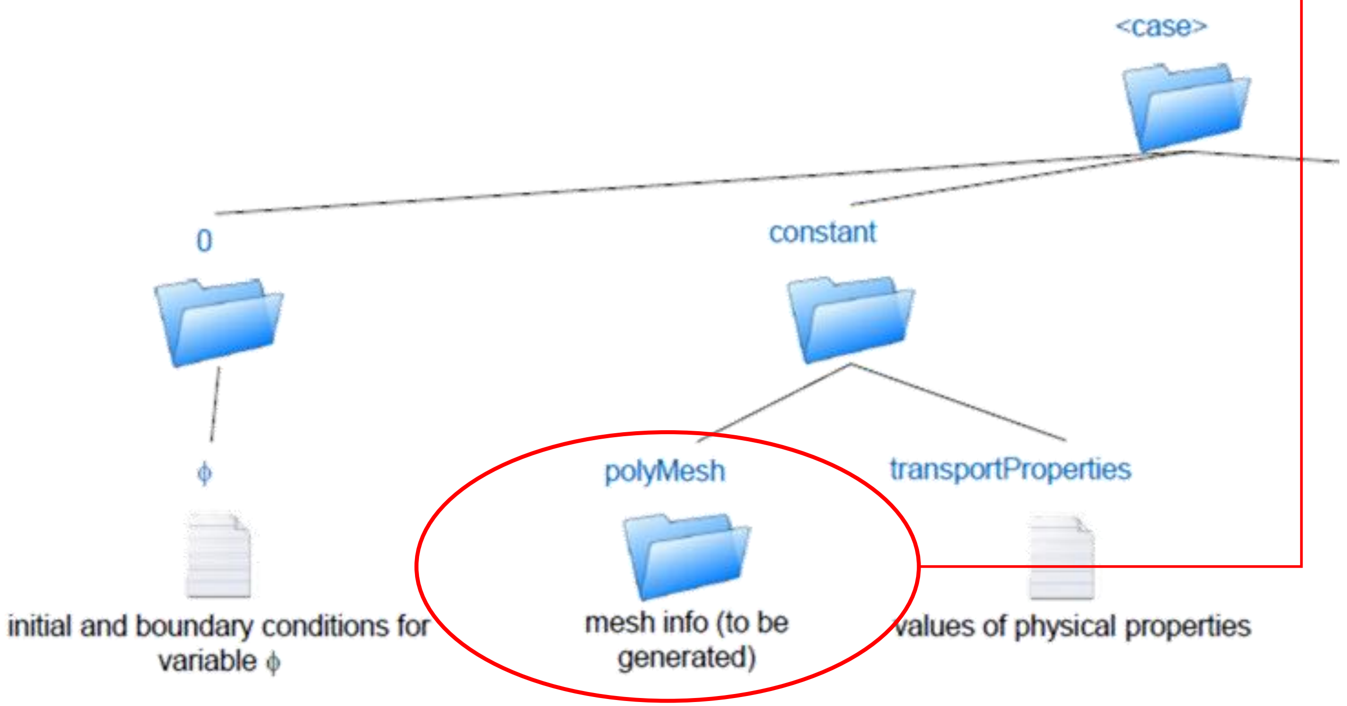
In 0.orig/T:

```
boundaryField
{
    CPLReceiveMD
    {
        type        fixedValue;
        value       $internalField;
    }
```

In 0.orig/alpha.liquid:

```
boundaryField
{
    CPLReceiveMD
    {
        type        fixedValue;
        value       $internalField;
    }
```

- Mesh

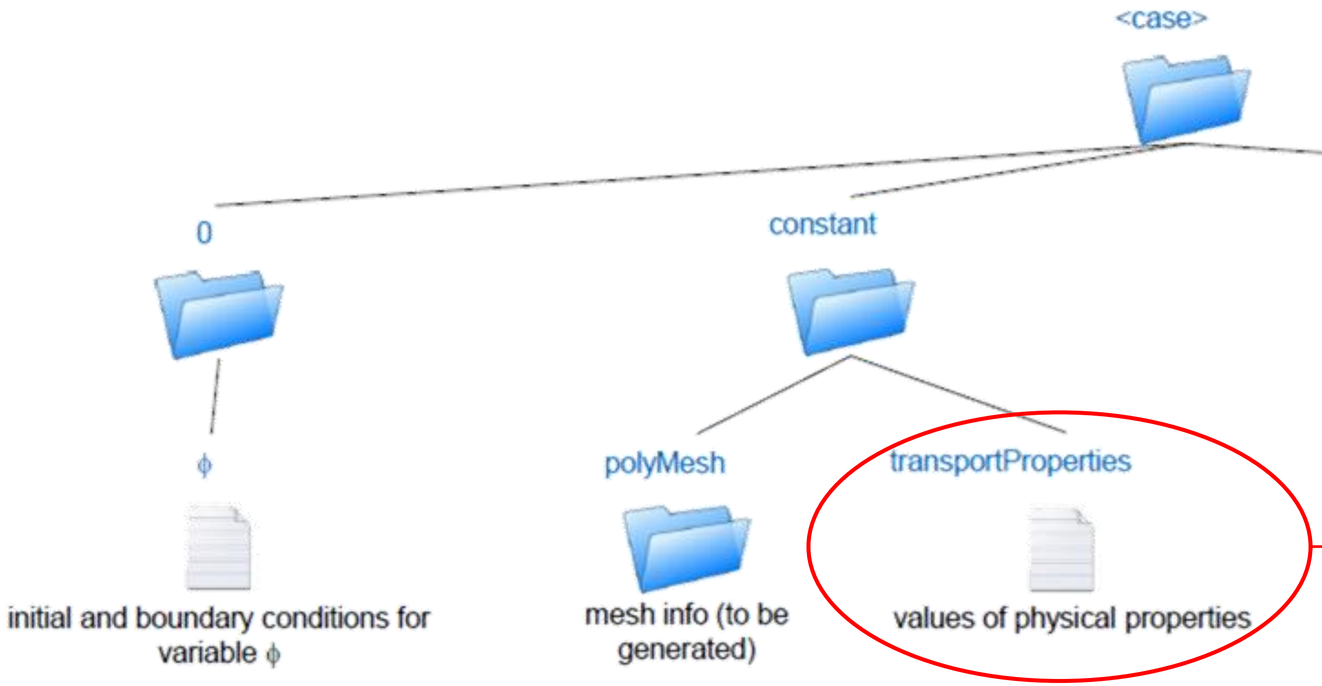


In constant/polyMesh/blockMeshDict

```
scale 476.22031559046;

vertices
(
    (0 0 0)
    (1.0 0 0)
    (1.0 1.0 0)
    (0 1.0 0)
    (0 0 0.02)
    (1.0 0 0.02)
    (1.0 1.0 0.02)
    (0 1.0 0.02)
);

…

boundary
(
…
    CPLReceiveMD
    {
        type patch;
        faces
       (
          (1 5 4 0)
      );
    }
```

Overview of the configuration files (5/8)

- Properties



In constant/transportProperties

```
liquid
{
    transportModel  Newtonian;
    nu              nu [ 0 2 -1 0 0 0 0 ]           1.76;
    rho             rho [ 1 -3 0 0 0 0 0 ]          0.69;
    Cp              Cp [ 0 2 -2 -1 0 0 0]           6.74;
    kappa           kappa [1 1 -3 -1 0 0 0]         4.64;
    Hf              Hf [0 2 -2 0 0 0 0]             6.54;
}

vapour
{
        …
}

PhaseChangeProperties
{
    model           hardtSimple;
    Tsat            Tsat [0 0 0 1 0 0 0]            0.92;
    R               R [0 2 -2 -1 0 0 0]             1.053;
    sigmaEvap       sigmaEvap [0 0 0 0 0 0 0 ] 1.0;
    sigmaCond       sigmaCond [0 0 0 0 0 0 0 ] 1.0;
    DmDot           DmDot [0 2 0 0 0 0 0]           500.0;
}
```

Phase-change model*

$$\dot{m} = \frac{2\gamma}{2-\gamma}\left(\frac{M}{2\pi R_g}\right)^{1/2}\frac{\rho_v H_f}{T_{sat}^{3/2}}(T - T_{sat});\, R = \frac{R_g}{M}$$

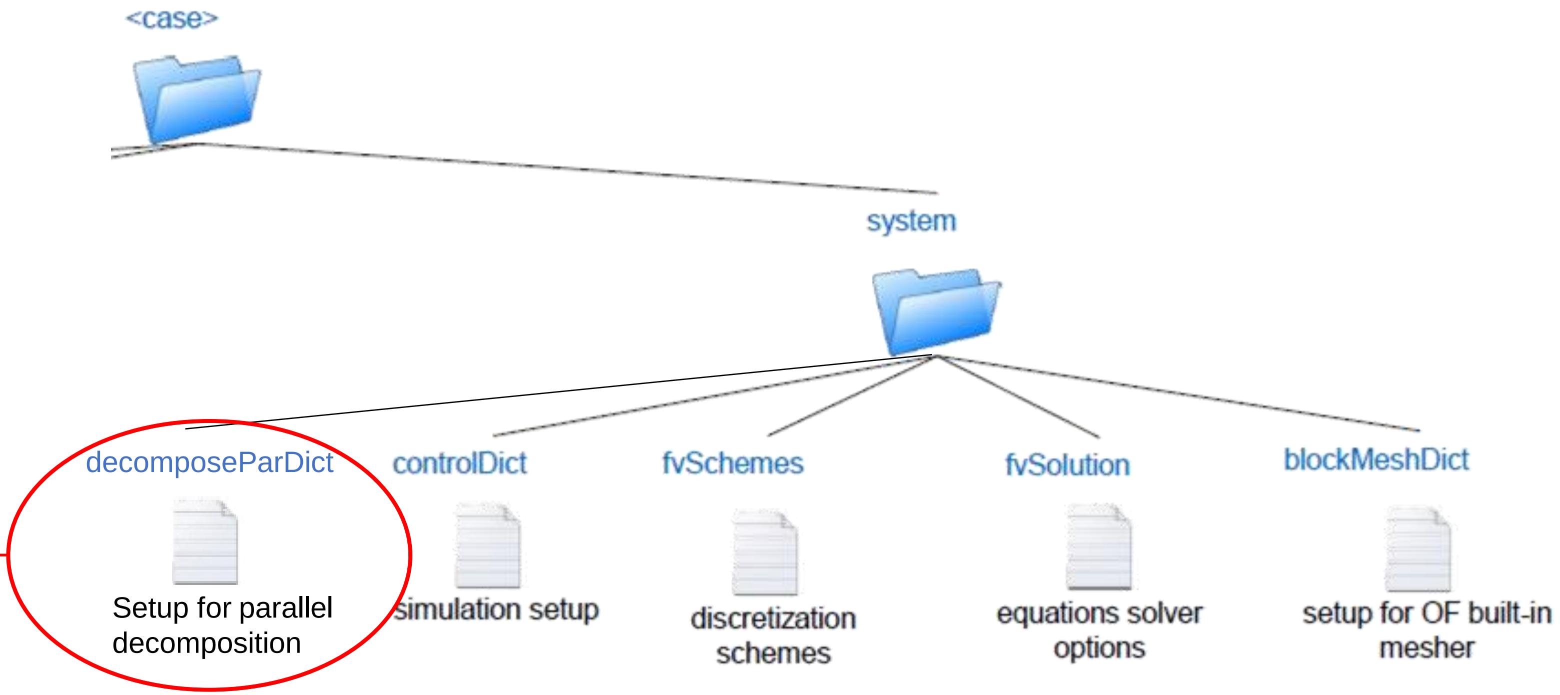
* Ferrari, A., Magnini, M. and Thome, J.R. (2018). Numerical analysis of slug flow boiling in square microchannels. *International Journal of Heat and Mass Transfer*, 123, pp.928–944.

Overview of the configuration files (6/8)

- Domain parallel decomposition

In system/decomposeParDict

```
numberOfSubdomains 2;

method          simple;

simpleCoeffs
{
    n               ( 2 1 1 );
    delta            0.001;
```



Overview of the configuration files (7/8)

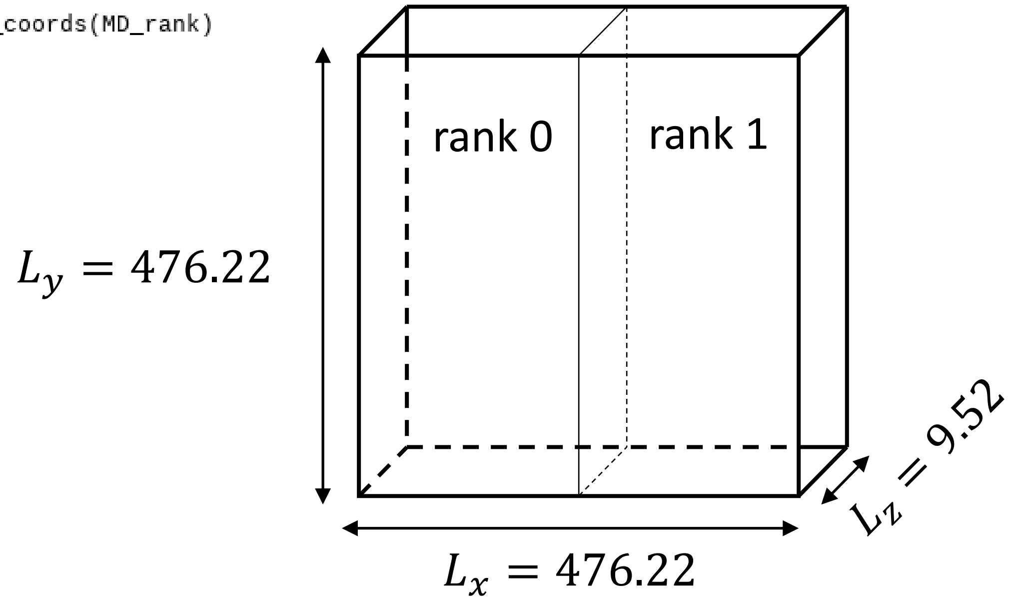
- Python script to load and send MD data to OpenFOAM

(load_and_send_MD_data.py)

- xyzL = Domain extension
- MD_COMM.Create_cart([*nx*, *ny*, *nz*])

nx, MPI subdomains along x
ny, MPI subdomains along y
nz, MPI subdomains along z

```
#Start coupled run
comm = MPI.COMM_WORLD
CPL = CPL()

MD_COMM = CPL.init(CPL.MD_REALM)
MD_Cart_COMM = MD_COMM.Create_cart([2, 1, 1])
CPL.setup_md(MD_Cart_COMM,
             xyzL=[476.22031559046, 476.22031559046, 9.5244063118092],
             xyz_orig=[0.0, 0.0, 0.0])


MD_rank = MD_COMM.Get_rank()
MD_coords = MD_Cart_COMM.Get_coords(MD_rank)
```



Overview of the configuration files (8/8)

- Python script to load and send MD data to OpenFOAM

(load_and_send_MD_data.py)

```python
for timestep in range(1,ntimestep):

    if MD_rank == 0:
        print("CFD timestep = ", timestep, "MD record loaded=", mdrec, flush=True)

    if timestep%timestepratio == 0:
        mdrec = mdrec + 1

    recv_array, ierr = CPL.recv(recv_array)

    #Get density bottom boundary condition
    MDrho = rhoObj.read(startrec=mdrec, endrec=mdrec)
    MDrhoBC = np.mean(MDrho[:,yloc,:,:,:],(2,3))

    MDu = uObj.read(startrec=mdrec, endrec=mdrec)
    MDuBC = np.mean(MDu[:,yloc,:,:,:],(2))

    Npercell = MDrhoBC[portion[0]:portion[1]+1,portion[4]:portion[5]+1]*dV
    send_array[3,:,0,:] = Npercell

    uvw = MDuBC[portion[0]:portion[1]+1,portion[4]:portion[5]+1,:]
    send_array[0,:,0,:] = uvw[:,:,0]*Npercell
    send_array[1,:,0,:] = uvw[:,:,1]*Npercell
    send_array[2,:,0,:] = uvw[:,:,2]*Npercell


    MDT = TObj.read(startrec=mdrec, endrec=mdrec)
    MDTBC = np.mean(MDT[:,yloc,:,:,:],(2))
    T = MDTBC[portion[0]:portion[1]+1,portion[4]:portion[5]+1,0]
    send_array[4,:,0,:] = T
    CPL.send(send_array)
```

- MDrho, MDu, MDT are density, velocity and temperature, respectively, from the MD simulation.
- Boundary conditions are evaluated at the horizontal line $y = yloc$.
- Data is sent to OpenFOAM through *send_array*

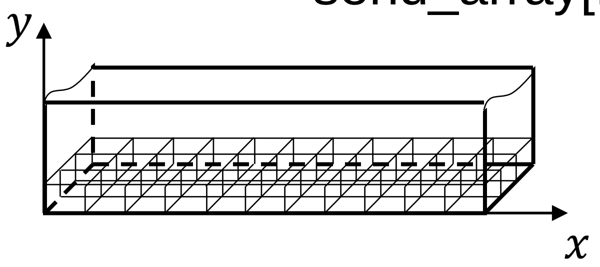
$(0,1,2) = (u,v,w)$

$3 = \rho$

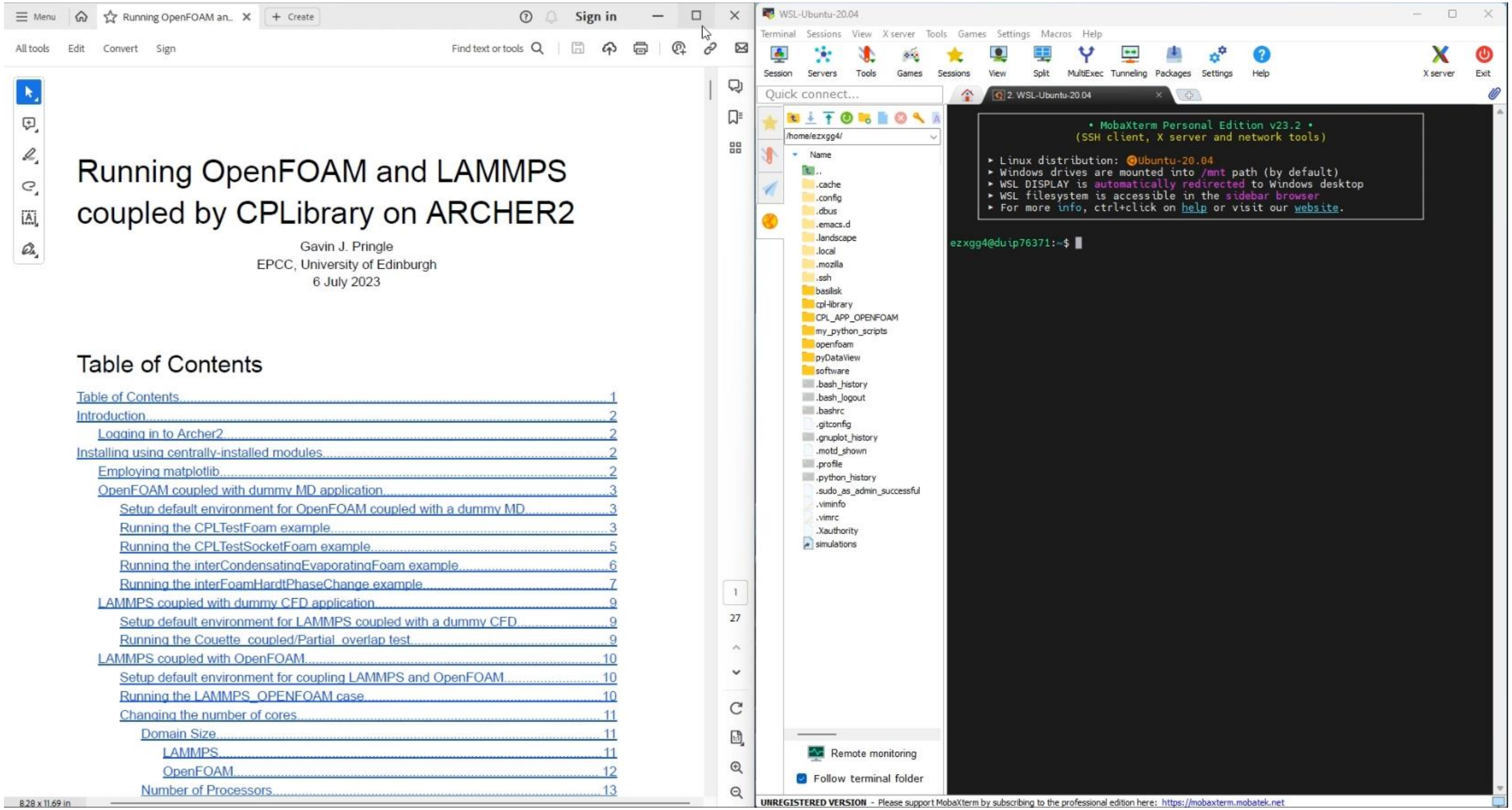
$4 = T$

send_array[field, :, 0, :]

x y z

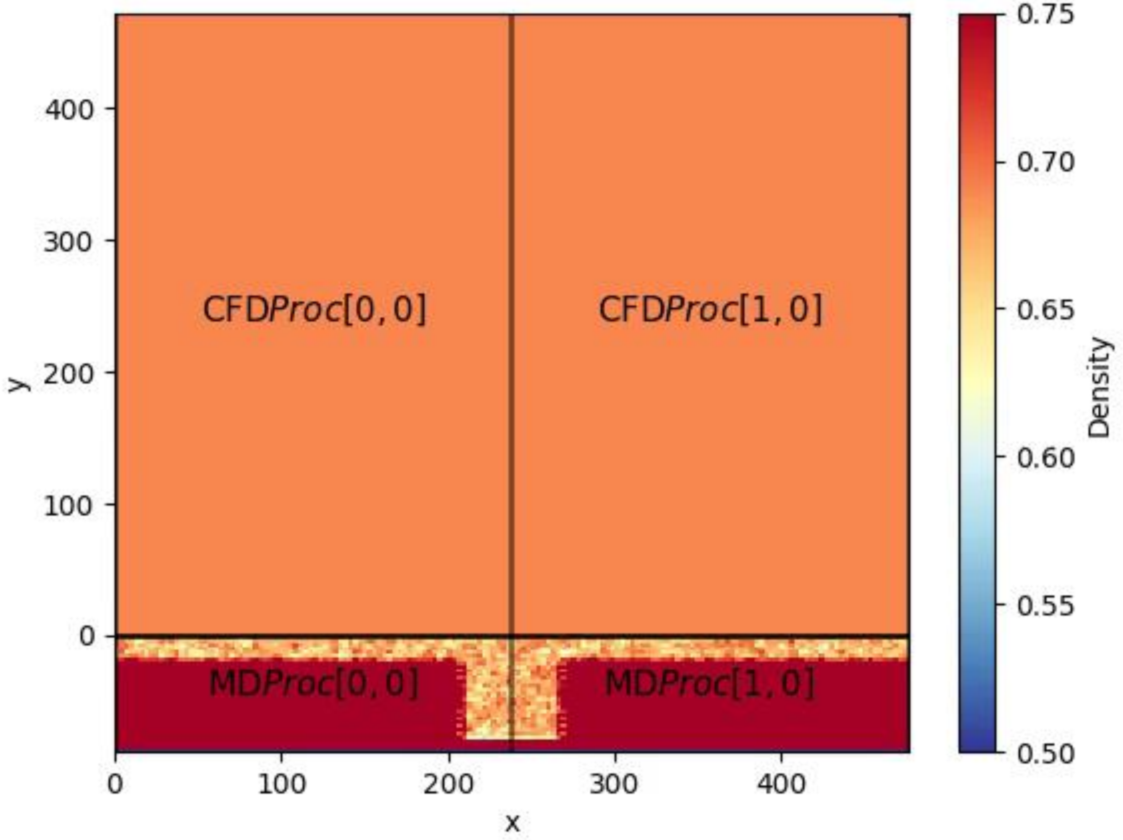


Demo


Running OpenFOAM and LAMMPS coupled by CPLibrary on ARCHER2
Gavin J. Pringle
EPCC, University of Edinburgh
6 July 2023
Table of Contents
Table of Contents 1
Introduction 2
Logging in to Archer2 2
Installing using centrally-installed modules 2
Employing matplotlib 2
OpenFOAM coupled with dummy MD application 3
Setup default environment for OpenFOAM coupled with a dummy MD 3
Running the CPLTestFoam example 3
Running the CPLTestSocketFoam example 5
Running the interCondensatingEvaporatingFoam example 6
Running the interFoamHardtPhaseChange example 7
LAMMPS coupled with dummy CFD application 9
Setup default environment for LAMMPS coupled with a dummy CFD 9
Running the Couette_coupled/Partial_overlap test 9
LAMMPS coupled with OpenFOAM 10
Setup default environment for coupling LAMMPS and OpenFOAM 10
Running the LAMMPS_OPENFOAM case 10
Changing the number of cores 11
Domain Size 11
LAMMPS 11
OpenFOAM 12
Number of Processors 13
WSL-Ubuntu-20.04
• MobaXterm Personal Edition v23.2 •
(SSH client, X server and network tools)
► Linux distribution: Ubuntu-20.04
► Windows drives are mounted into /mnt path (by default)
► WSL DISPLAY is automatically redirected to Windows desktop
► WSL filesystem is accessible in the sidebar browser
► For more info, ctrl+click on help or visit our website.
ezxgg4@duip76371:~$
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

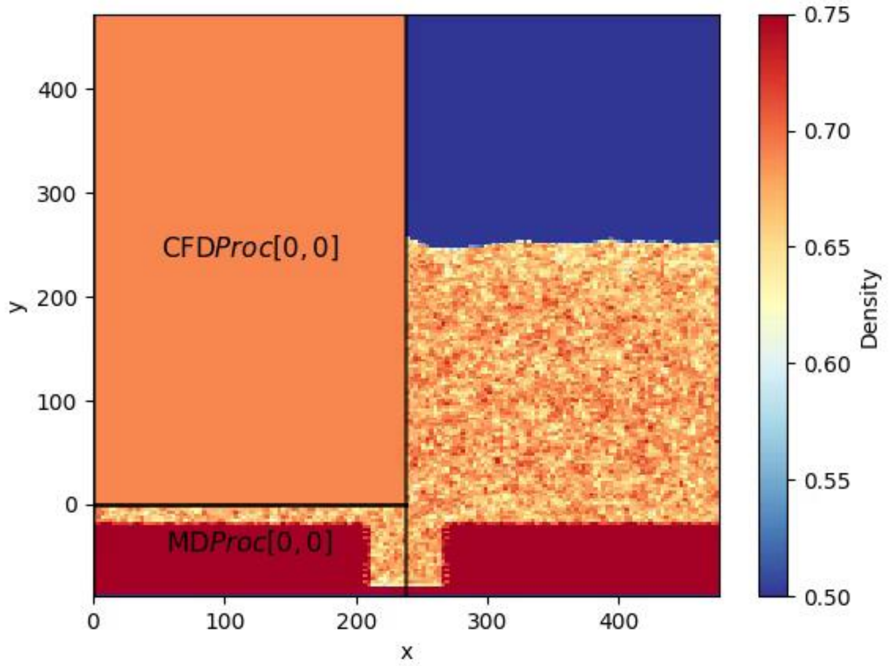
Results (1/2)

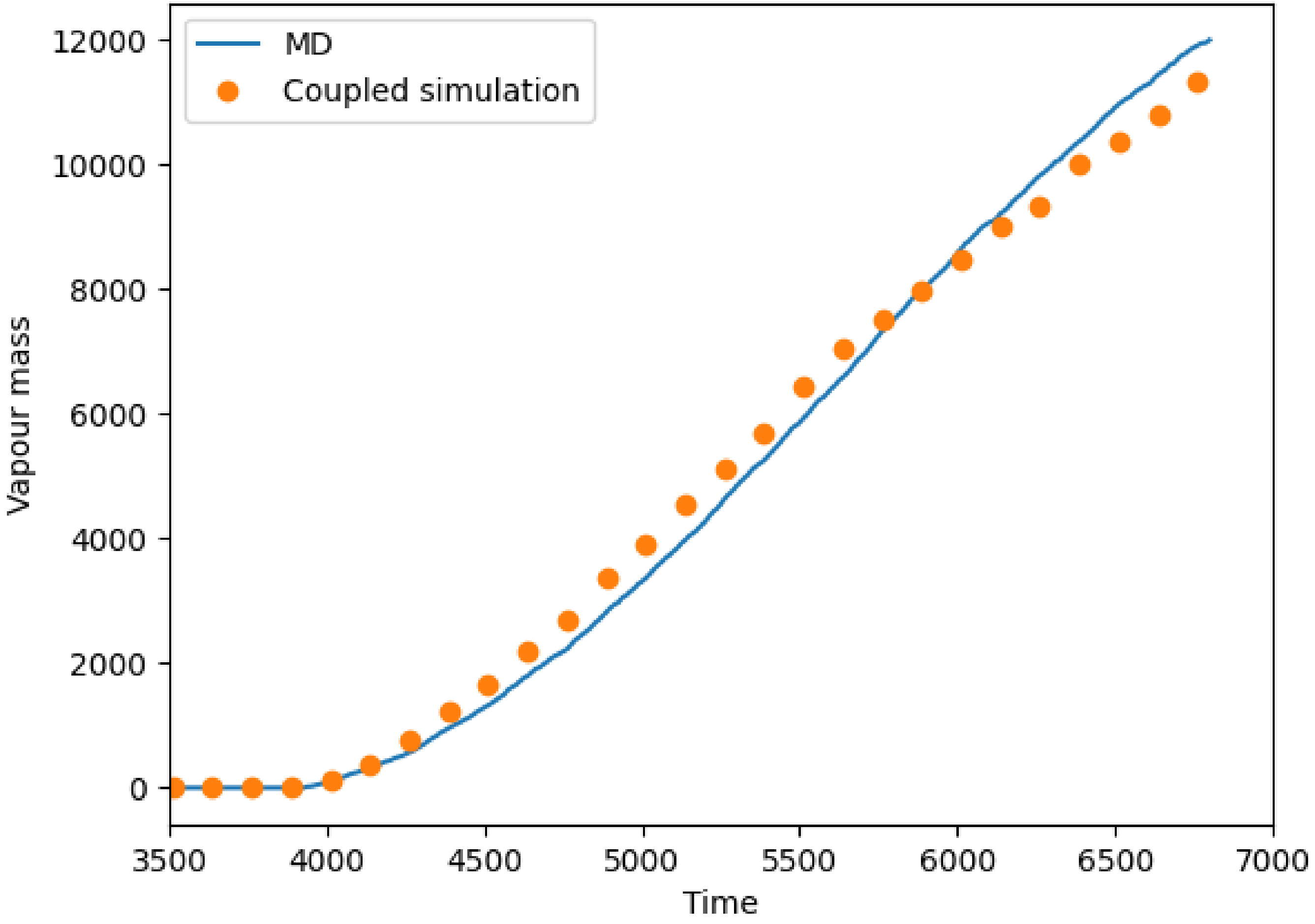
➢ Coupled simulation: contours of density



Results (2/2)

➢ Coupled simulation Vs MD





Where can I find the data to reproduce the example?

- The MD data used in this tutorial can be copied from the shared folder:

  /work/ecseaf01/shared/MDBoilingData

Where can I find the data to reproduce the example?

- The MD data used in this tutorial can be copied from the shared folder: /work/ecseaf01/shared/MDBoilingData
- If you don't have an account on Archer2, you can still run this case using the minimal example (minimal_MD.py), where a uniform jet of vapor is injected at the coupled boundary instead of the actual MD data.

```
recv_array, send_array = CPL.get_arrays(recv_size=3, send_size=5)
uwall = 0.; vwall = 0.5
olap_limits = np.zeros(6); portion = np.zeros(6)
olap_limits = CPL.get_olap_limits()
portion = CPL.my_proc_portion(olap_limits)
dV = CPL.get("dx")*CPL.get("dy")*CPL.get("dz")
for time in range(501):
    recv_array, ierr = CPL.recv(recv_array)
    for i in range(send_array.shape[0]):
        for k in range(send_array.shape[2]):
            ig = i + portion[0]
            print(time, i, k, ig)
            if (ig > 60 and ig < 90):
                rho = 0.02;
                send_array[3,i,0,k] = rho*dV
                send_array[0,i,0,k] = uwall*send_array[3,i,0,k]
                send_array[1,i,0,k] = vwall*send_array[3,i,0,k]
            else:
                rho = 0.7;
                send_array[3,i,0,k] = rho*dV
                send_array[0,i,0,k] = uwall*send_array[3,i,0,k]
                send_array[1,i,0,k] = 0.0*send_array[3,i,0,k]
            send_array[4,i,0,k] = 0.95
    CPL.send(send_array)
```

Velocity components of the vapor jet

Uniform vertical flow of vapor with fixed velocity

Liquid region at the bottom boundary