Science and Technology Facilities Council

Guide to LBE exercises

Michael Seaton
UKRI STFC Daresbury Laboratory
michael.seaton@stfc.ac.uk

Mesoscale exercises

Put together practical exercises on DPD and LBE

- Available via main CCP5 Summer School exercises page:

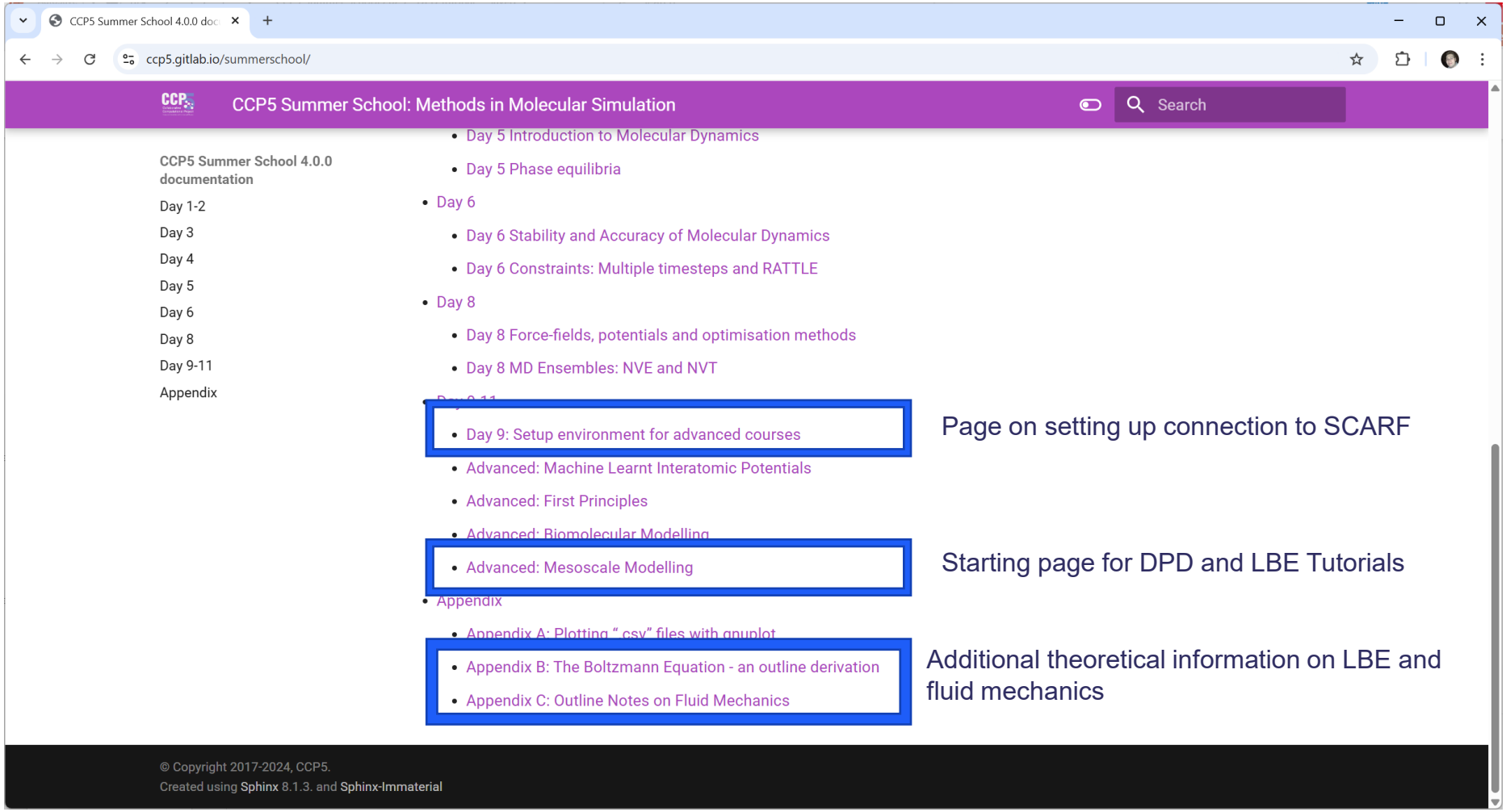
  summer.ccp5.ac.uk

  - Points to Jupyter notebooks for each exercise, available to download using Google Colab to run on your laptops
  - Requirements to use notebooks: Fortran/C/C++ compilers, GNU Make, Python and modules, VMD, ParaView

- DPD Exercises available as Day 9
- LBE Exercises available as Day 10 and Day 11

- Additional background information on exercises, theory and brief C programming guide (needed for LBE exercises)
- Download link to these slides on the LBE exercises and similar ones for the DPD exercises

Mesoscale exercises



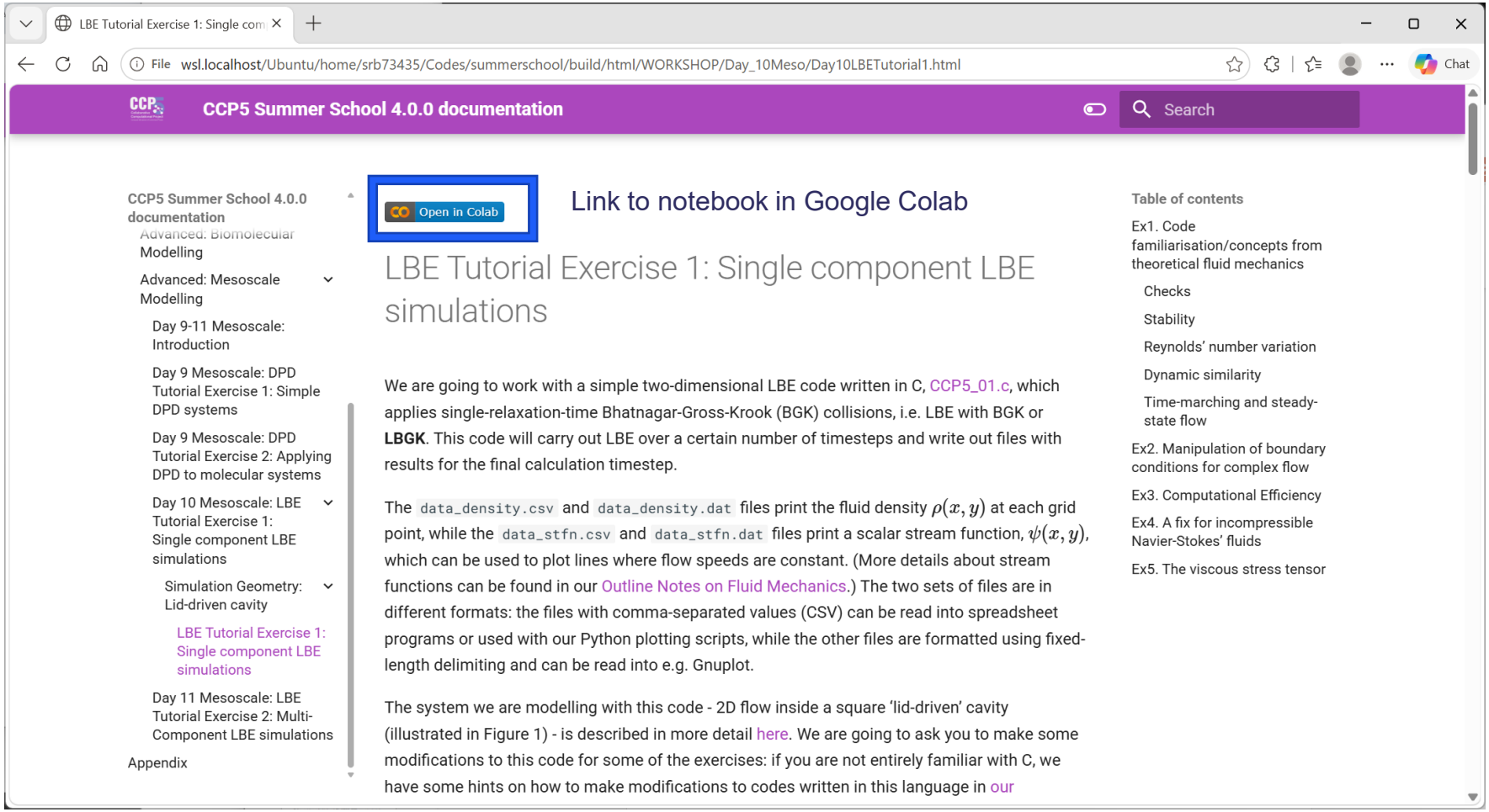
Page on setting up connection to SCARF

Starting page for DPD and LBE Tutorials

Additional theoretical information on LBE and fluid mechanics

Science and Technology Facilities Council

LBE Jupyter notebooks



LBE practical exercises

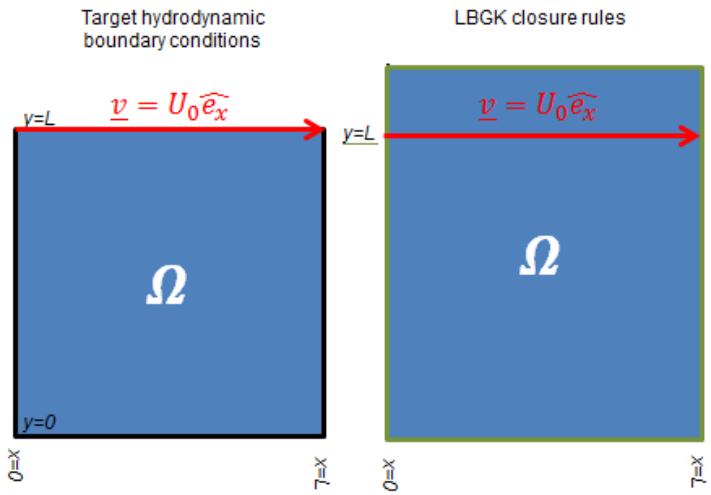
Designed to show LBE's capabilities as a mesoscale modelling method

- Two tutorials (notebooks covering multiple exercises):
  1. Single-component LBE simulations (lid-driven cavity)
     - Code familiarisation/concepts from theoretical fluid mechanics
     - Manipulation of boundary conditions for complex flow
     - Computational efficiency
     - Fix for incompressible Navier-Stokes' fluids
     - Viscous stress tensor
  2. Multi-component LBE simulations (suspended drop)
     - Code familiarisation, Laplace law validation, fluid component segregation, curvature calculations
     - Code transformation, equilibration dynamics, algorithm change
- All but last notebook use custom codes written in C, last notebook uses DL_MESO_LBE

Tutorial Exercise 1: Single component LBE simulations

Ex1. Code familiarisation/concepts from theoretical fluid mechanics

- Notebook: Day10LBETutorial1.ipynb
- Introduces a simple LBE code in C: `CCP5_01.c`
  - Includes a command-line option for required number of timesteps
  - Models a 2D 'lid-driven cavity' system
  - On-site bounce-back for side and bottom walls
  - Equilibrium forcing for top moving wall at speed U_0
  - Additional lattice points above top wall to prevent interference between top and bottom boundaries – have no effect on system itself and thus discarded



Science and Technology Facilities Council

Tutorial Exercise 1: Single component LBE simulations

Ex1. Code familiarisation/concepts from theoretical fluid mechanics

- Need to compile before running code using (GNU) C compiler:

```
gcc -o LBE CCP5_01.c
```

- Run code with command-line option for number of timesteps (`N`):

```
./LBE N
```

  - Defaults to 4000 timesteps if `N` not specified
  - Python script available to launch code and display progress bar in Jupyter notebook

- Top of code has `#define` statements and a variable declaration setting simulation properties (size of lattice, initial fluid density, speed of top wall U_0, relaxation frequency ω)
  - **Changing any of these requires re-compilation to take effect!**

Tutorial Exercise 1: Single component LBE simulations

Ex1. Code familiarisation/concepts from theoretical fluid mechanics

- Code generates 'snapshot' files with fluid densities and constant-velocity stream functions ψ at all grid points (apart from discarded region above wall)
  - Python scripts available to read files and plot data in notebook
  - Also have MATLAB/Octave and Gnuplot plotting scripts

- Sub-exercises:
  - Calculational stability when varying Reynolds' number:

$$Re = \frac{U_0 L_y}{\nu}$$

  - Dynamic similarities in solutions for same Re but different system sizes, relaxation frequencies and/or wall speeds
  - How simulation gets to a steady state (number of required timesteps)

Tutorial Exercise 1: Single component LBE simulations

Ex2. Manipulation of boundary conditions for complex flow

- Default boundary condition for stationary walls is on-grid bounce-back
- `CCP5_01.c` also has code to use mid-grid bounce-back instead
  - Places no-slip boundary mid-way between lattice points
  - Code for this currently commented out – can uncomment this section and comment out the on-grid bounce-back lines

- Can also add a submerged jet into the flow by using equilibrium forcing
  - Replace distribution functions with values calculated using **`f_equ`** function
    - Function inputs: lattice link **`i`**, velocity (**`ux`**, **`uy`**), **`density`**
  - Either set values in distribution function array **`f`** directly after propagation, or fix values (**`ux`**, **`uy`**, **`rho`**) for given point in **`Calc_obs`** subroutine

Tutorial Exercise 1: Single component LBE simulations

Ex3. Computational efficiency

- Most time-consuming parts of `CCP5_01.c` are:
  - Calculation of observables (`Calc_obs`)
  - Collisions (`Collide`)
- `Collide` subroutine calls `f_equ` function multiple times per grid point
  - Could in-line calculations of $f_i^{eq}(\rho, \mathbf{u})$ directly into this subroutine instead of repeatedly calling this function for each lattice link
  - Alternatively, could change `f_equ` to calculate all local equilibrium distribution functions at a grid point in one go
- `Calc_obs` subroutine could also be optimised
  - HINT 1: Try assigning the distribution function `f[x][y][i]` to an intermediate variable before calculating `rho`, `ux` and `uy`
  - HINT 2: Multiplication is cheaper than division, so try calculating the reciprocal of `rho` for grid point and multiplying this into `ux` and `uy`

Tutorial Exercise 1: Single component LBE simulations

Ex4. Incompressible Navier-Stokes' fluids

- Change local equilibrium distribution function (mildly compressible form):

$$f_i^{eq}(\rho, \mathbf{u}) = t_i \rho \left[1 + 3(\mathbf{c}_i \cdot \mathbf{u}) + \frac{9}{2}(\mathbf{c}_i \cdot \mathbf{u})^2 - \frac{3}{2}u^2 \right]$$

  to form for exactly incompressible liquids:

$$f_i^{eq}(\rho, \mathbf{u}) = t_i \rho + t_i \rho_0 \left[3(\mathbf{c}_i \cdot \mathbf{u}) + \frac{9}{2}(\mathbf{c}_i \cdot \mathbf{u})^2 - \frac{3}{2}u^2 \right]$$

  - Density of fluid now constant, ρ_0
  - Uses ρ as variable density (analogue to pressure)
  - Velocity now calculated as $\mathbf{u} = \frac{\sum_i f_i \mathbf{c_i}}{\rho_0}$
- Either change `f_equ` function in copy of `CCP5_01.c` or use previously modified version (`EILBGK.c`)

Tutorial Exercise 1: Single component LBE simulations

Ex5. Viscous stress tensor

- Can be calculated at a lattice point using:

$$\sigma'_{\alpha\beta} = \frac{\rho c_s^2}{\omega} \sum_i \left(f_i^{eq} - f_i\right) c_{i\alpha} c_{i\beta}$$

  - No need to find velocity gradients: big advantage of LBE over CFD
  - Can be used to impose non-Newtonian and/or turbulent flows by finding shear rates and using these to set viscosities/relaxation times for each lattice point

- In lid-driven cavity system, bulk flow approximates potential flow (negligible viscous stress)

- Either modify `Calc_obs` subroutine in copy of `CCP5_01.c` to calculate and write file with $\sigma'_{\alpha\beta}$ values or use previously modified version (`sigma.c`)

Multi-component LBE algorithm

Lishchuk's multi-component (chromodynamic) algorithm

- Two fluids: background fluid = blue, drop fluid = red
- Each fluid at each lattice point has its own distribution function (B_i, R_i) and an 'achromatic' distribution function can be defined, $f_i = B_i + R_i$
  - Fluid densities at lattice points available:
    - $\rho_B = \sum_i B_i$
    - $\rho_R = \sum_i R_i$
    - $\rho = \rho_B + \rho_R = \sum_i f_i$

- We define a 'phase index':

$$\rho^N = \frac{\rho_R - \rho_B}{\rho_R + \rho_B}$$

  - Values of ρ^N range from +1 (entirely red) to -1 (entirely blue)
  - Gradients of ρ^N in interfacial regions help us determine interfacial normals

$$\hat{\mathbf{n}} = \frac{\nabla \rho^N}{|\nabla \rho^N|}$$

Gradient calculations using lattice 'stencils', e.g.

$$\nabla \rho^N(\mathbf{x}) = \frac{1}{c_s^2} \sum_i t_i \rho^N(\mathbf{x} + \mathbf{c}_i)\mathbf{c}_i$$

Multi-component LBE algorithm

Lishchuk's multi-component (chromodynamic) algorithm

- Interfacial curvature available from spatial gradient of interfacial normal

$$K = n_x n_y \left(\frac{\partial n_x}{\partial y} + \frac{\partial n_y}{\partial x} \right) - n_x^2 \frac{\partial n_y}{\partial y} - n_y^2 \frac{\partial n_x}{\partial x}$$

- Force acting between red and blue fluids at interface:

$$\mathbf{F} = \frac{1}{2} \alpha K \nabla \rho^N$$

Interfacial tension α

  - Force can be applied to fluids during collision step: several forcing schemes available, Guo forcing scheme recommended

- For continuity of tangential stresses and fluid velocity, need to carry out collisions and apply interfacial force to achromatic distribution functions (giving f_i'), then segregate back out:

$$R_i' = \frac{\rho_R}{\rho} f_i' + \beta t_i \frac{\rho_R \rho_B}{\rho} \mathbf{c}_i \cdot \hat{\mathbf{n}}$$

$$B_i' = \frac{\rho_B}{\rho} f_i' - \beta t_i \frac{\rho_R \rho_B}{\rho} \mathbf{c}_i \cdot \hat{\mathbf{n}}$$

Segregation parameter β controls (non-zero) interfacial thickness, prevents 'pinning' of drop to lattice points

Tutorial Exercise 2: Multi-component LBE simulations

Ex1. Code familiarisation/Laplace law pressure step validation

- Notebook: Day11LBETutorial2Ex1-3.ipynb
- Another simple LBE code in C: `CCP5_02.c`
  - Models a static drop surrounded by background fluid in a periodic box using Lishchuk algorithm
  - Need to compile before running code using (GNU) C compiler, linking in some mathematics libraries:

```
gcc -o LBE CCP5_02.c -lm
```

  - Run code with command-line option for number of timesteps (`N`):

```
./LBE N
```

  - Defaults to 10000 timesteps if `N` not specified
  - Properties (e.g. drop radius, α, β) hard-coded: re-compile for changes

Tutorial Exercise 2: Multi-component LBE simulations

Ex1. Code familiarisation/Laplace law pressure step validation

- Python scripts available in notebook to launch code, display progress bar and plot results written to files (properties at all lattice points):
  - Overall density ρ
  - Phase index/field ρ^N
  - Interfacial curvature K
  - Velocity modulus $|\mathbf{u}|$ - will not be zero due to 'interfacial microcurrents'

- Further script available to find drop radius and find densities inside/outside drop – can confirm Laplace law holds and interfacial tension α obtained:

$$p_R - p_B = c_s^2(\rho_R - \rho_B) = \frac{\alpha}{R}$$

Tutorial Exercise 2: Multi-component LBE simulations

Ex2. Fluid component segregation (re-colouring)

- Segregation parameter β given as **`BETA_LKR`** in **`#define`** line at top of **`CCP5_02.c`** code
- Try reducing this value and contrast/compare results:
  - Values of ρ^N around boundary should vary as a hyperbolic tangent profile
  - Interfacial width should be inversely proportional to β
  - Microcurrent magnitudes should be inversely proportional to interfacial width
- What happens if β is increased?

Tutorial Exercise 2: Multi-component LBE simulations

Ex3. Curvature calculations

- Calculations of K currently fairly complicated, but could be simplified to divergence of interfacial normal, i.e.:

$$K = \frac{\partial n_x}{\partial x} + \frac{\partial n_y}{\partial y}$$

- Edit `calc_obs_2` in a copy of `CCP5_02.c` to use this definition
  - What happens to the microcurrents compared with the original code?
  - Can a stable solution still be obtained?

DL_MESO

General purpose mesoscopic modelling software package

- Includes Lattice Boltzmann (LBE) and DPD codes (serial and parallel with MPI and/or OpenMP), Java GUI (optional)

- Created for CCP5 at UKRI STFC Daresbury Laboratory in 2004
  - Now developed for UKCOMES (EPSRC High-End Computing consortium)

- Free for academic use (licence on annual subscription for commercial use)

- Described in Molecular Simulation articles:
  - Seaton *et al.*, *Mol Sim* **39** (10), 796–821 (2013)
  - Seaton, *Mol Sim* **47** (2–3), 228–247 (2021)

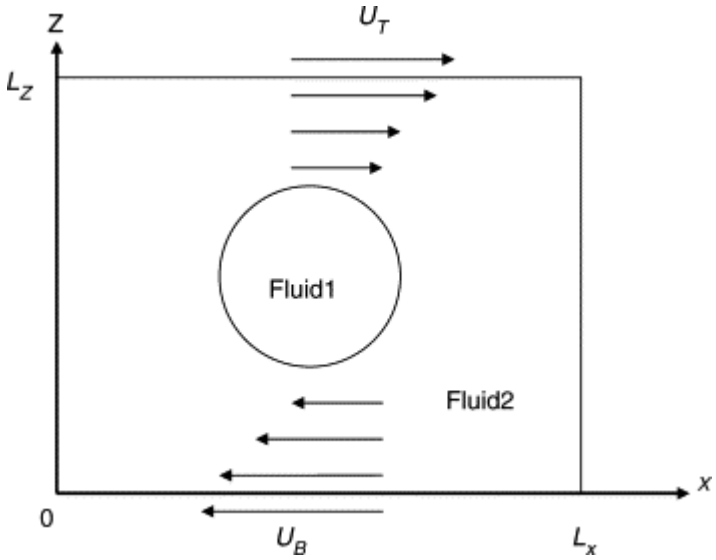
- Copy of beta version (2.8) included with course materials, available for you to take home with you
  - Small additions included for exercises (makefiles in working directory, additional Python scripts)

Tutorial Exercise 2: Multi-component LBE simulations

Ex4. Equilibration dynamics with DL_MESO_LBE

- Notebook: Day11LBETutorial2Ex4-5.ipynb
- Switching over to LBE code in DL_MESO
  - Includes Lishchuk multi-component algorithm used in `CCP5_02.c` plus some extensions (will use one of these later)
- Makefile available to compile DL_MESO_LBE with OpenMP multithreading
- Python script available to launch in notebook
- Modelling drop subjected to linear shear flow
  - Using fixed velocity boundary conditions at top and bottom of box
  - Shear only applied after drop given time to equilibrate

Different options for implementing fixed velocity boundary conditions: default in DL_MESO_LBE based on bounce back of non-equilibrium distribution functions (Zou-He)



Tutorial Exercise 2: Multi-component LBE simulations

Ex4. Equilibration dynamics with DL_MESO_LBE

- Input files for DL_MESO_LBE:
  - Simulation controls in **`lbin.sys`** (right)
  - Boundary conditions in **`lbin.spa`**
  - Initial conditions in **`lbin.init`**
- Relevant keywords in **`lbin.sys`** file:
  - **`collision_type`** (also forcing type)
  - **`interaction_type`**
  - **`total_step`**
  - **`equilibration_step`**
  - **`save_span`**
  - **`speed_top_0`** (u_x at top boundary)
  - **`speed_bot_0`** (u_x at bottom boundary)
  - **`interaction_0_1`** (α)
  - **`segregation`** (β)

```
discrete_speed              9
number_of_fluid             2
number_of_solute            0
temperature_scalar          0
phase_field                 0
incompressible_fluids       0
collision_type              BGKGuo
interaction_type            Lishchuk
output_format               VTK
output_type                 ANSI
grid_number_x               150
grid_number_y               50
grid_number_z               1
total_step                  105000
equilibration_step          5000
save_span                   500
speed_top_0                 0.0025
speed_top_1                 0.0
speed_top_2                 0.0
speed_bot_0                 -0.0025
speed_bot_1                 0.0
speed_bot_2                 0.0
relaxation_fluid_0          1.0
relaxation_fluid_1          1.0
interaction_0_0             0.0
interaction_0_1             0.0034
interaction_1_1             0.0
segregation                 1.4
```

Tutorial Exercise 2: Multi-component LBE simulations

Ex4. Equilibration dynamics with DL_MESO_LBE

- Segregation works slightly differently in DL_MESO_LBE cf. `CCP5_02.c`, e.g.

$$R_i' = \frac{\rho_R}{\rho} f_i' + \beta t_i \frac{\rho_R \rho_B}{\rho^2} \mathbf{c}_i \cdot \hat{\mathbf{n}}$$

- Apart from changing value of β, will give identical results to previous code

- Run calculation with supplied input files
  - DL_MESO_LBE generates structured grid VTK files as outputs with densities and velocities at lattice points
  - Output files can be read using ParaView (all at once) or by Python scripts supplied with notebook (individually)
    - Filters available in ParaView to calculate additional properties (e.g. phase indices ρ^N) from those supplied, plot streamlines etc.

Tutorial Exercise 2: Multi-component LBE simulations

Ex4. Equilibration dynamics with DL_MESO_LBE

- Reynolds' and capillary numbers available for drop in linear shear flow:

$$Re = \frac{R^2 \dot{\gamma}}{\nu}$$

$$Ca = \frac{\rho \nu U}{\alpha} = \frac{R \rho \nu \dot{\gamma}}{\alpha}$$

$$\dot{\gamma} = \frac{u_x^{\text{top}} - u_x^{\text{bottom}}}{L_y} = \frac{\partial u_x}{\partial y}$$

- Vary wall velocities (shear rate) and interfacial tension
  - Should be a capillary number (or set of capillary numbers?) that cause the drop to break up due to the flow

- First output file (`lbout000000.vts`) gives equilibrated state before shear is included
  - Can examine microcurrent magnitudes with varying Ca

Tutorial Exercise 2: Multi-component LBE simulations

Ex5. Changing Lishchuk algorithm to 'Spencer tensor'

- DL_MESO_LBE has variety of multi-component algorithms based on Lishchuk's original
- One form ('Spencer tensor') replaces calculation of interfacial forces with direct forcing terms applied during collision stage:

$$F_i = \frac{t_i \beta \alpha \omega \rho_B \rho_R}{c_s^4 \rho^3 \Delta t} (\hat{\mathbf{n}}\hat{\mathbf{n}} - \mathbf{I}) : (\mathbf{c}_i \mathbf{c}_i - c_s^2 \mathbf{I})$$

- No longer need to calculate curvature (from gradients of interfacial normals)
  - Fewer core-to-core MPI communications when running on multiple processors
- Still need interfacial normals and thus phase index gradient calculations
  - Local phase index gradients calculations possible distribution functions directly – not being used here but also available in DL_MESO_LBE, makes Lishchuk algorithm fully localised to each lattice point

Tutorial Exercise 2: Multi-component LBE simulations

Ex5. Changing Lishchuk algorithm to 'Spencer tensor'

- Start by modifying `lbin.sys` from previous calculation to use Spencer tensor variant – change these lines:

```
collision_type          BGKGuo
interaction_type        Lishchuk
```

to

```
collision_type          BGK
interaction_type        LishchukSpencerTensor
```

- Check microcurrents for a given interfacial tension and shear rate
  - Spencer tensor variant likely to give larger spurious microcurrents, but not by much

Tutorial Exercise 2: Multi-component LBE simulations

Ex5. Changing Lishchuk algorithm to 'Spencer tensor'

- Another set of input files for a three-component system with linear shear: background fluid and two immiscible drops

- Initially placed drops too close together so they overlap
  - Segregation should separate them out during equilibration
  - Higher interfacial tension between fluids in drops to prevent them getting too close together

- Try this simulation with both original Lishchuk method and Spencer tensor variant
  - What has happened to the drops during equilibration?
  - What happens afterwards when shear is applied?
  - Why do the two (otherwise similar) approaches give different results?

Before you finish …

Registering for DL_MESO

- If you want to use DL_MESO for your own research (either DPD or LBE), please register via the website

www.ccp5.ac.uk/DL_MESO

DL_Software Digital Guide

- If you want more information about DL_MESO, the mesoscale modelling methods it includes and/or other DL_Software packages, we have an online guide available at

https://dl-sdg.github.io/