1 General information

This software is a script compilation for FreeFem++ to solve interface cell problems and arrive with coefficients in boundary condition (BC) between porous medium and free fluid. This module gives the possibility to compute the coefficients for BC using three-dimensional (3D) structures for porous case, defined and meshed by GMSH software. These coefficients can be used in a 2D cavity solver, provided in module "2D_MSE_bc_velocity". If you find this software useful, please cite the corresponding publication <DOI:10.1017/jfm.2016.838> (open access).

The contents of this module are listed below.

```
src/3D_MSE_bc_Ksolver.edp       - script to solve for K tensor
src/3D_MSE_bc_Lsolver.edp       - script to solve for L tensor
src/3D_MSE_bc_parameters.in     - example parameter input file
src/3D_MSE_bc_gen_geom.py       - script for geometry definition in GMSH *.geo format
doc/3D_MSE_bc_velocity.pdf      - this document
```

In this document we explain the key elements of the software, parameters of the simulation, and implemented variational formulation.

2 Instructions for obtaining coefficients

In order for the current FreeFem++ scripts to work, the geometry of pore structure and the interface has to be prescribed as a mesh. The current software provides a way to accomplish this task using GMSH *.geo file, which is generated using a Python script. Provided routines can define a fully connected, isotropic network of spheres, connected using cylinders. Parameters for geometry definition of the interface cell, located in file "3D_MSE_bc_parameters.in", are:

1. coordinate of the top boundary of the cell ($ztop$);
2. number of periodic structures to include below the interface ($Nbot$);
3. solid volume fraction ($theta$);
4. ratio between radii of connecting cylinders and hub spheres (cR/hR);
5. mesh spacing ($dsMin$, $dsMax$).

The interface is always located at $y = 0$, i.e., at the tip of the solid structure. After defining the required parameters, one has to execute the Python script by typing "./3D_MSE_bc_gen_geom.py", which produces four files

```
3D_MSE_bc_parameters.txt       - complemented parameter file for FreeFem++ solver
3D_MSE_bc_gen_geomB.geo        - geometry and mesh definition for bottom part of interface cell
3D_MSE_bc_gen_geomT.geo        - geometry and mesh definition for top part of interface cell
3D_MSE_bc_gen_geomL.geo        - geometry and mesh definition for Lagrange multipliers
3D_MSE_bc_gen_geomI.geo        - geometry and mesh definition for interior cell
```

The text file “3D_MSE_bc_parameters.txt” contains additional parameter, the bottom coordinate of the interface cell, which is computed by the Python script based on the number of periodic structures contained in the interface cell. The mesh can be generated using GMSH from command line

```
gmsh -3 3D_MSE_bc_gen_geomX.geo
```

where X can be B, T, L or I. Alternatively, the GUI of GMSH can be used to load the corresponding *.geo file. In such case, one can see the geometry directly, and the mesh can be generated using Mesh module and pressing on "3D".

After all three mesh files have been generated, the FreeFem++ solvers for K_{ij} and L_{ijk} tensors can be used. The usage is essentially the same as for $2D$ versions, i.e., one can execute in the command line

```
FreeFem++-nw 3D_MSE_bc_Ksolver.edp
```

for finding the K_{ij} field. Similarly, L_{ijk} solver can be executed. The current version use direct solver to solve the resulting linear matrix, which can be found in code snapshot below.

```
// Set the solver, solve the system
set(CoupSYS, solver=sparsesolver);
real[int] SOL = CoupSYS^-1*RHS;
```

This is the quickest way to solve the outline problem, if sufficient amount of memory is available. The simulation with test parameters take roughly $50\ GB$ of RAM. If such memory amount is not available, the iterative solver should be employed, possible choice is GMRES solver. To choose it, the code must be modified to read

```
// Set the solver, solve the system
set(CoupSYS, solver=GMRES, eps=1.0e-10);
real[int] SOL = CoupSYS^-1*RHS;
```

Solution of the problem will take much longer compared to the direct solver, however, it would now be feasible on computers having only fraction of the original RAM requirements. If the iteration residual statistics are required, the verbosity variable has to be modified accordingly. For example, adding

```
// Output more information to screen
verbosity = 5;
```

to the beginning of script file, will output enough details to the screen to see progression of GMRES solver. For the weak formulation of the problem and some more comments, reader can check the documentation in module “2D_MSE_bc_velocity” of solvers “2D_MSE_bc_Ksolver.edp” and “2D_MSE_bc_Lsolver.edp”.