Frequency-Domain Lattice Boltzmann (FreqD-LBM) Simulation Software

Quick Reference Guide

1. Scope.

The software implements frequency-domain lattice-Boltzmann simulation method described in the article *The Frequency-Domain Lattice Boltzmann Method (FreqD-LBM): A Versatile Tool to Predict the QCM Response Induced by Structured Samples* by Diethelm Johannsmann, Paul Häusner, Arne Langhoff, Christian Leppin, Ilya Reviakine, and Viktor Vanoppen. The paper currently is under review. It is referred to in this manual as the "FLBM paper". The details of the method and its applications are explained in the article. Here, the focus is on how to use the software in practice.

The software is written in Python.
FreqD-LBM is an open-source software under MIT license (en.wikipedia.org/wiki/MIT_License)

2. Requirements

The software was tested in Python 3.9 and 3.11.5 on personal computers running Windows 10 and Windows 11. It requires pandas, *numpy*, *matplotlib*, and *tooltip* libraries that do not normally come with the standard python distribution and have to be installed separately (on a computer with an internet connection, use *pip install <library name>*).

3. Software Organization

The software consists of three parts (Fig. 1): the core libraries (collected under \Lib) that can be accessed via a GUI (FLBM.py, fblmn.py, fblmpar.py) or via a wrapper (Main_FreqDLBM.py). **The GUI** helps the user define simulation parameters (SPs). **In the case of the wrapper** (Main_FreqDLBM), the user directly edits the code defining the SPs. SPs are then passed to the simulation core (SingleSim). In the case of the GUI, SPs are passed to the simulation via a file that is named <yyy-mm-dd-hh_mm_ss.flbmsym.npy> stored in the default simulation subfolder (flbm) or in the folder of the user's choice. The user has the option to run the simulation in a serial or in a parallel mode. The file with the simulation parameters is saved and can be used to re-run the exact same simulation if desired. This is accomplished by using the command

"python fblmn.py flbm\yyy-mm-dd-hh_mm_ss.flbmsym.npy" (in serial mode)

or

"python fblmpar.py flbm\yyy-mm-dd-hh_mm_ss.flbmsym.npy" (in parallel mode).

(assuming the default location). The wrapper directly passes SPs to SingleSim and runs in a serial mode only. Both the GUI and the wrapper have provisions for displaying simulation progress plots (see below). In the GUI, there is also an option to display simulation results.
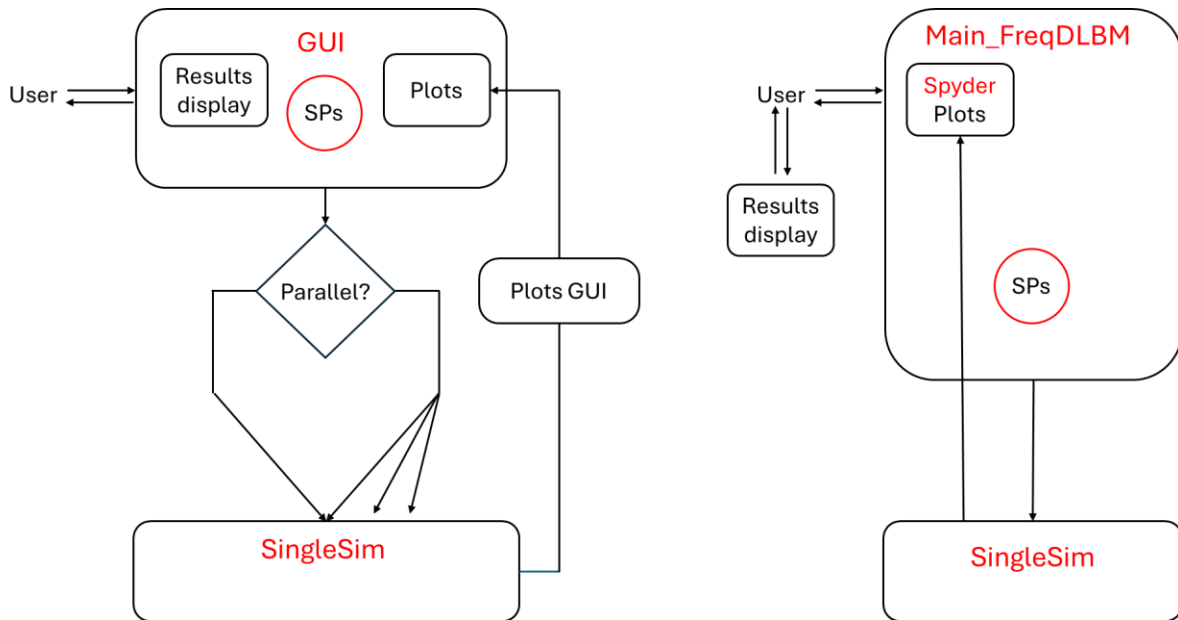

Figure 1: Software Organization

4.   Running the software

The software can be run from the command line by typing "python FLBM.py" in the folder containing the FLBM.py file.

The software can also be run from within Spyder. For the GUI version, load FLBM.py into Spyder, and configure external shell execution (in the Spyder main menu, press "Run" and "Configuration per file"; then select "Run file with custom configuration" and "execute in an external system console".

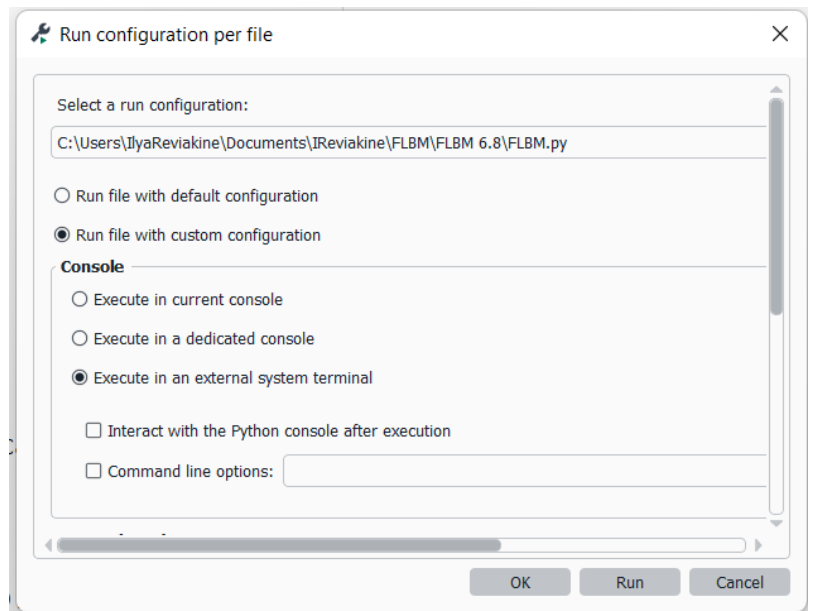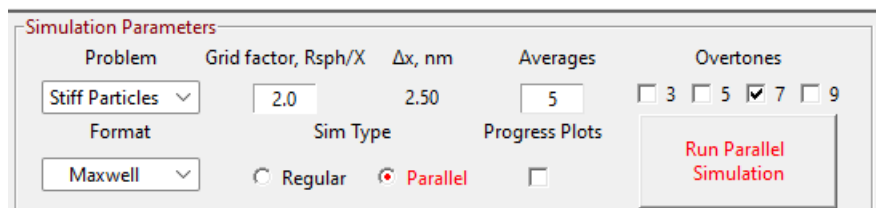The wrapper, Main_FreqDLBM.py, can be run directly from Spyder, but the GUI cannot.

Figure 2: Configuring Spyder for running FLBM in an external shell.

5. Defining simulation parameters.

5.1 GUI

Basic simulation parameters are grouped into frames: the parameters that apply to the entire simulation (**Error! Reference source not found.**), bulk and resonator parameters (Figure 6), and sphere(s) parameters. Furthermore, there is a set of advanced parameters that can be accessed through the menu "*Advanced SimPars*".

Figure 3: *Simulation Parameters frame*

Problem Type: Currently implemented are the simulations of "soft spheres" modelled as regions of increased shear stiffness, and "stiff spheres" modelled as oscillating boundaries. The implementation of roughness, film resonance, etc., are left for the future.

Format: Specifies how the viscoelasticity of the sphere/surface contact is defined. Three options are available: $|\eta|$, $\tan(\delta)$ or J', J'' with their corresponding exponents, β', β'', and Maxwell, with $|\eta|, \tau$. Viscoelastic parameters have their usual meaning (see [1]). In the case of the format specified by Maxwell formalism, $|\eta|$ is used to calculate shear elastic modulus at infinite frequency, while the relaxation time τ separates frequency ranges with liquid-like solid-like behaviour (Fig. 4).
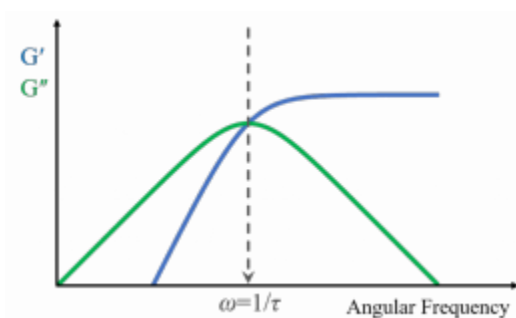

Figure 4: *Maxwell viscoelasticity format*

Grid Factor: simulation grid resolution in fractions of the sphere radius, Rsph. The recommended value is 5, but a value of 3 is sufficient for approximate simulations. Grid resolution in nm is displayed in the adjacent field.

Averages: the number of times the simulation is repeated with different particle positions in the simulation box. The recommended number is 5.

Overtones to be included in the simulation. At least one overtone is required.

Simulation type (SimType): *Regular* or *Parallel*. For Parallel simulations, the choice to display progress plots is also offered. This is useful for diagnostics but this option is currently slow. It is recommended that parallel simulations are run without progress plots. Progress plot files are saved in the *tmpplot* subfolder

---

[1] Johannsmann, D. Viscoelastic analysis of organic thin films on quartz resonators. *Macromol. Chem. Phys.* **200**, 501–516 (1999); Johannsmann, D. *The Quartz Crystal Microbalance in Soft Matter Research*. (Springer Berlin Heidelberg, New York, NY, 2014).

and can be accessed at a later time. In the future, a separate display option for browsing those plots will be implemented.

Run simulation button will start the simulation in a new console. A file containing simulation parameters is created, that is named <yyy-mm-dd-hh_mm_ss.flbmsym.npy>. It is stored in the default simulation subfolder (flbm) or in the folder of the user's choice. It is passed to flbmsym.py or flbmsympar.py depending on whether the simulation is run in the serial mode or in the parallel mode. The progress of simulation is monitored in the *Simulation Progress* frame that opens at the bottom of the GUI window (Figure 5), and the plots frame to the right. The plots frame will display the ring-in process plots, sphere motional parameter plots, and, finally, the velocity and density profiles, such as those shown in Figure 3.4 and Figure 3.5 of the FLBM paper. Simulation parameters of the current step are displayed above each plot.

In the case of the simulations run in parallel, the number of simultaneously running simulations depends on the number of processor cores. Only one of them can be selected for plotting at any given time,
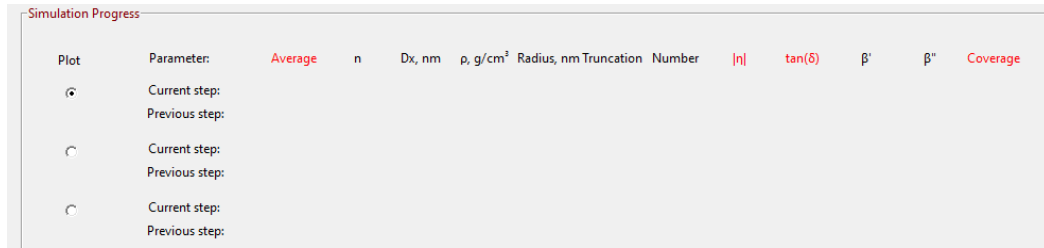

Plot	Parameter:	Average	n	Dx, nm	ρ, g/cm³	Radius, nm	Truncation	Number	|η|	tan(δ)	β'	β"	Coverage
⦿	Current step:												
	Previous step:												
○	Current step:												
	Previous step:												
○	Current step:												
	Previous step:												

Figure 5: Simulation Progress frame.

selected using a radio button in the *Simulation Progress* frame (Figure 5).

The file with the simulation parameters can be used to re-run the exact same simulation if desired. This is accomplished by using the command

"python fblmn.py flbm\ yyy-mm-dd-hh_mm_ss.flbmsym.npy" (in serial mode)

or

"python fblmpar.py flbm\ yyy-mm-dd-hh_mm_ss.flbmsym.npy" (in parallel mode).

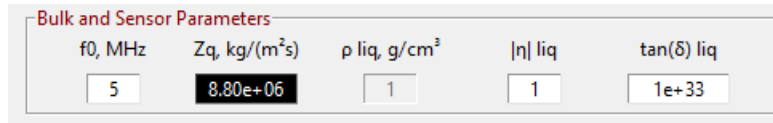| Bulk and Sensor Parameters				
f0, MHz	Zq, kg/(m²s)	ρ liq, g/cm³	|η| liq	tan(δ) liq
5	8.80e+06	1	1	1e+33

Figure 6: Bulk and sensor parameters.

Bulk and sensor parameters (Figure 6) include the fundamental frequency f_0, acoustic impedance of the resonator material Z_q and the viscoelastic properties of bulk liquid, specified as $|\eta|$, $\tan(\delta)$. By default, the fundamental frequency is set to 5 MHz, the acoustic impedance—to that of quartz, and bulk liquid parameters to those of water. Bulk liquid density is 1 g/cm² and cannot be modified.

Sphere(s) parameters (Figure 7).

The parameters defining the sphere(s) include density, radius, degree of truncation, the number of spheres per unit cell, viscoelastic properties specified according to the format selected in the *Simulation*
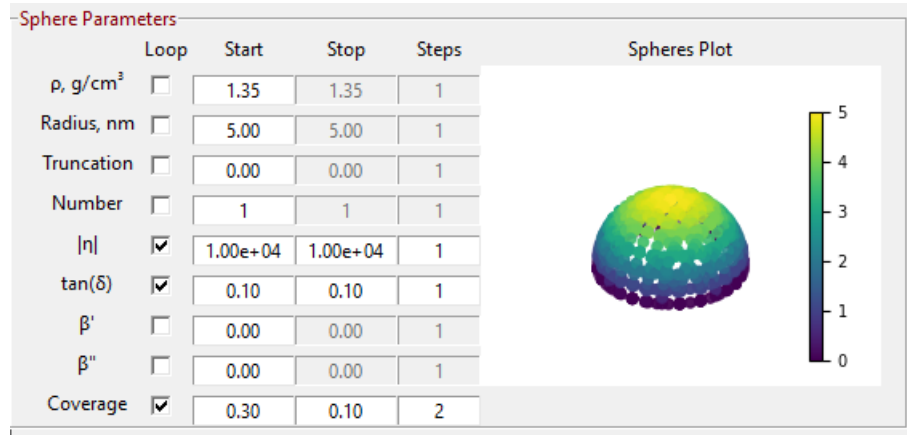
Figure 7: Sphere parameters.

Parameters frame, and coverage. Furthermore, here the user can select the parameters which are looped, the starting and the ending values of the parameters in a loop, and the number of steps on a linear scale. To the right, there is a plot of the simulation cell with the vertical scale in nm. The appearance of the plot will depend on the grid resolution, the radius, truncation, and number of spheres. The time to update the plot will depend on the grid resolution.

Most of these parameters are self-explanatory. Special attention is drawn to the truncation, which is specified on a scale of $0 - 1$, where 0 corresponds to half-spheres (50% truncation) and 1 corresponds to full spheres and 0% truncation), or

%truncation = (1 – truncation)/2*100, and the height, h = radius*(1 + truncation).

The height corresponds to 2x sphere radius when truncation is 1 and % truncation is zero, and to 1x sphere radius when truncation is 0 and % truncation is 50%.

For the viscoelastic parameters, the limits on the exponents are specified under the *Limits* menu. Limits do not apply to the Maxwell format, and the *Limits* menu is disabled when Maxwell format is selected.

In the case of multiple spheres, sphere positioning may fail at high coverages (~ 0.45).

Menu

The *File* menu contains functions relating to saving and opening simulation parameters, making the current set of simulation parameters default, and re-setting the simulation parameters to the values that are hard-coded in the IO library. The hard-coded simulation parameters are "safe" in the sense that they will not lead to a simulation hanging or crashing. *Limits* offers the possibility on adjusting the limits of the viscoelastic exponents, while *Advanced SimPars* (Figure 8) allow the user to modify the parameters that control ring-in convergence, smoothing, and selecting the collision operator Λ.[2] Hovering the mouse over the parameters will display suggested values.

---

[2] Krüger, T. *et al. The Lattice Boltzmann Method: Principles and Practice*. (Springer International Publishing, Cham, 2017). doi:10.1007/978-3-319-44649-3, page 429.

Figure 8: Advanced Simulation Parameters

DispalySimResults will open a window where results of simulations can be displayed.

Help displays About information and this document.

5.2   Wrapper

```python
SPs['ProblemType']  = 'StiffParticles' #'SoftParticles', 'StiffParticles',\
SPs['Dx_nm']      = 3   # important

#parameters which often are looped, "s" at the end of a name indicates an array
RSph_nms           = np.array([5])
ySphbyRs           = np.array([0.0])
CovTargets         = np.array([0.3,0.1])
ns                 = np.array([7])
rhoSphs            = np.array([1])
etaabscenSphmPass  = np.array([1e4]) # 1e4 corresponds to 1.9 GPa
tandelcenSphs      = np.array([0.1])

Jp_FacSphs     = np.array([10])
Jpp_FacSphs    = np.array([10])

Pars1 = etaabscenSphmPass;   SPs['Par1str'] = 'etaabscenSphmPas';
Pars2 = tandelcenSphs;       SPs['Par2str'] = 'tandelcenSph';
Pars3 = CovTargets;          SPs['Par3str'] = 'CovTarget'

SPs['folder'] = 'test'
SPs['fname0'] = 'test'
SPs['Do_from_GUI'] = False
```

Figure 9: Editing simulation parameters in the wrapper

When using the wrapper, Main_FreqDLBM.py, the simulation parameters (SPs) are edited directly in the python code, as shown in Figure 9. For example, problem type is selected by assigning a value "StiffParticles" or "SoftParticles" to the parameter SPs['ProblemType']. Sphere radius, which is set to 5 nm in Figure 9, can be changed by changing "5" to any other desired value greater than zero, and so on.

Parameters to be iterated over in the simulation are created by including multiple values in the square brackets, like it is done with CovTargets in Figure 9: CovTargets = np.array([0.3,0.1]): this simulation will run at two coverages, 0.1 and 0.3. One advantage of using the wrapper is that logarithmic spacing can be used for parameters to be iterated over. For example, $|\eta|$ can be set to 1e4, 1e3, 1e2, etc.: etaabscenSphmPass = np.array([1e4, 1e3, 1e2]). This feature is not yet implemented in the GUI.

It is important to match the iteration parameters to their names by setting Pars1, Pars2, and Pars3 variables, and the corresponding SPs, to the appropriate parameter names. In Figure 9, parameters to iterate over are coverage, $|\eta|$, and tan (δ), while the choice of viscoelastic parameters is specified by setting the value of SPs['VEPars_Choice'] (not shown in Figure 9, it appears in line 57 of the wrapper). To

iterate over a different set of parameters, one needs to include multiple values for the parameter in question as well as change the values of one of Pars*i* and SPs[Par*i*str], where *i* is 1, 2, or 3, to correspond with it. Do not assign multiple values to parameters that are not iterated over.

The grid resolution is set by adjusting the value of SPs['Dx_nm']. In Figure 9, it is set to 3 nm. It is recommended that it is set to $1/5^{th}$ of the radius of the sphere.

It is important to keep SPs['Do_from_GUI'] = False.

SPs['folder'] = 'test' and SPs['fname0'] = 'test' set the subfolder and the file name prefix for placing simulation results.

The values of SPs in the wrapper have no effect on the functioning of the GUI or on the simulation run from the GUI.

6.   Simulation Results

Simulation results are collected in three files. In the case of the simulations run from the GUI, they are named FreqD-LBM-Output_yyyy-mm-dd-hh_mm_ss.txt, .cfg, and _Aux.txt, respectively. The date and time match those of the .npy file containing simulation parameters. In the case of the simulation run from the wrapper, the prefix of the file names is specified by the SPs['fname0'] parameter, followed by _yyyy-mm-dd-hh_mm_ss, followed by .txt, .cfg, and _Aux.txt.

Simulation results can be viewed using *DisplaySimResults* menu option in the GUI that becomes available upon completion of the simulation. This option will work for simulations run from the GUI as well as from the wrapper. Further analysis of the results can be performed using external software.

Simulation results can also be viewed using Main_FreqDLBM_Display_Results.py utility run from Spyder by setting the parameter fname to the path and file name of the simulation results file.

7.   Limitations, acceleration, and other considerations.

Access to several problem types mentioned in the FLBM paper that can be addressed with these simulations have not yet been implemented in the GUI. In some instances, they can be accessed through the wrapper, however, the code has only been tested StiffParticles and SoftParticles.

Acceleration is mostly done with numba. A command of the form @jit(nopython = True) switches numba on. The respective routine is translated to C-code when it is called first. Those routines must not make use of dictionaries (like SPs). There are quite a few instructions from numpy, which numba does not tolerate. Numba is a bit particular about data types. When numba throws error messages concerning data types, debugging is cumbersome.

Most importantly: Those routines do know of global parameters, but these are not updated, when the routine is called repeatedly (and there is no error message). If the routine is part of a loop, the parameter, which is looped over, must be part of the arguments of the call. If not, the routine keeps using the value, which it found when it was called for the first time.

We believe that FreqD-LBM can be faster, when the np.roll command is used in the streaming step. We did not succeed in getting that to work reliably. If someone can do that …he or she will serve the community.