Link-wise (directional) boundary conditions for LBM

Francesco Marson, Department of Computer Science, University of Geneva

Palabos Summer School 2021

Updated version with improved CMakelists.txt and bug fix from Julius Weinmiller (Ulm University) [see at the end of this document].

To run the solution, go to CMakelists.txt and change the following line

```
add_executable(${EXECUTABLE_NAME} "./src/${EXECUTABLE_NAME}.cpp")
```

to

```
add_executable(${EXECUTABLE_NAME} "./src/solutions/${EXECUTABLE_NAME}_solution.cpp")
```

Be a Palabos developer!

The main goal of this exercise is to have fun and to implement a new boundary condition (the HW BB).

As beneficial collateral effects, you may familiarize yourself with the following topics

- Compile an example with CMake;
- Use Palabos vti writer;
- Use paraview to analyze the results;
- Use the voxelizer;
- Generate TriangleSets;
- Read external .stl files;
- Use sponge zones;
- Use off-lattice boundary conditions implemented in Palabos;
- Compute the drag coefficient.

What it does not cover:

- Moving objects
- Refilling algorithms

Tasks list

1. Install CMake, ParaView, and update your `c++` compiler
2. Test the compilation with CMake
3. Compile and run the exercise after choosing the simulation parameters;
4. Compute the drag coefficient and write it on a file using pcout;
5. Warm-up
   - Try the available schemes implemented in palabos with different flow parameters;
   - Compute the drag for each method and save them on different files;
   - Turn on and off a sponge zone to dump pressure waves;
   - Using Bouzidi or Guo, add a second sphere behind the first one;
   - Transform the two spheres into ellipsoids.
   - (optional) Use the TRT dynamics for low Reynolds and Smagorinsky for the turbulent case.
6. Have fun:
   - Change the planar Poiseuille configuration to periodic+neumann (on the sides) flow configuration;
   - replace the ellipsoids with the “cbc_example_2021.stl” file;

- translate and rotate the cbc_example_2021.stl

7. Implement the half-way bounce-back starting from the implementation of the BFL scheme

Guide

1. Install CMake, ParaView, and update your c++ compiler

To compile this example you need CMake installed in your system. To check the installation type on your terminal: `cmake --version`. If it returns an error, you need to proceed with the installation. Try with the following command:

- Archlinux based (Manjaro): `sudo pacman -S cmake`
- Debian based (Ubuntu): `sudo apt install cmake`
- redhat based (OpenSuSE, Fedora, Centos): `yum install cmake`

for other systems, please refer to the official installation instructions.

Update the c++ compiler

To compile this exercise, you need an updated compiler compatible with the c++17 standard.

- if you are using an Arch-based distribution, you are updated by definition.
- if you are using ubuntu, check this link https://linuxize.com/post/how-to-install-gcc-compiler-on-ubuntu-18-04/ and read the section "Installing Multiple GCC Versions".

[optional, takes time, do it later] Install meshlab

Meshlab is an open-source program to manipulate meshes. It is particularly useful to scale and rotate `.stl` files before importing them into Palabos. The easiest way is to install it from snap: `snap install meshlab`. Otherwise, you can google for your distribution-dependent instructions.

2. Test the compilation

If Palabos is on the directory of the exercises, the provided CMakeLists.txt should already work. In all the other cases, you can check the following lines of the CMakeLists.txt to modify the palabos location:

```
# # # ADD PALABOS PATH:
# # CASE 1: PALABOS_ROOT is on your path
# file(TO_CMAKE_PATH $ENV{PALABOS_ROOT} PALABOS_ROOT)

# # CASE 2: specify an ABSOLUTE DIRECTORY PATH
#set(PALABOS_ROOT /home/mars/MEGA2/projects/palabos)

# # CASE 3: specify a relative path to the build folder (eg if you are in the showCases folder)
set(PALABOS_ROOT ${CMAKE_CURRENT_SOURCE_DIR}/../../..)
```

Open a terminal in the current directory and type the following commands to compile the code with CMake:

```
cd build
cmake .. && make -j 2
cd ..
```

in the case of a Linux-based system. CMake runs in automatic in release mode, if you want to change the build mode to debug, you can use the CMake with the option `-DCMAKE_BUILD_TYPE=Debug`.

3. Compile and run the exercise after choosing the simulation parameters

Check now the file `cbc_example_3d.cpp`. First, choose your numerical resolution in:

```
IncomprFlowParam<Real> parameters =
    lu.initLrefluNodim(000000/*resolution*/, &dimless, 00000 /*u_lb*/,
                       false /*tau, optional*/)
```

```
        .printParameters()
        .getIncomprFlowParam();
```

then, compile it again, and run the simulation in parallel using mpirun (if installed in your system)

```
mpirun -np 2 cbc_example_3d
```

in case of problems with mpi, try to run the serial version

```
./cbc_example_3d
```

4. Drag coefficient

Now it is time to compute the drag coefficient, I can give you three hints:

1. Check the available methods of `OffLatticeBoundaryCondition3D<Real, Descriptor, Array<Real, 3>>`
2. Use the definition of Drag coefficient;
3. Use `pcout` to write output files in parallel simulations.

5. Warm-up

Change boundary condition

At point 7 of the function `setupLattice3d()` you will find that a function is called to set up the off-lattice boundary condition. Try to figure out how to modify it to use a different boundary condition.

Use sponge zones

At point 5. of `setupLattice3d()` you will find a call to a function to set up a sponge zone. Check inside this helper function to understand how it works. Then, try to set up sponge zones and to see their effects on the simulation. For instance, compare the time evolution of the drag coefficient with and without its implementation.

Add a second sphere and transform them into ellipsoids

Using either Guo (GZS) or Bouzidi (BFL), add a second sphere behind the first one. Then, try to transform them into ellipsoids. Hint: check file `shapes.h`.

6. Have fun

Use an external .stl mesh file

Inside this folder, you will find a file called `cbc_example_2021.stl`. I downloaded it from https://www.thingiverse.com/, then I have scaled, rotated, and simplified it using `meshlab`. Try to read it using the available method in `shapes.h`, then try to run simulations with different parameters.

[optional, it takes time, do it later] Use your own .stl file

Look for a stl file on https://www.thingiverse.com/. Then using meshlab:

1. center on the origin of the axes using the translate filter
2. rotate it using the rotation filter
3. reduce the amount triangles using a simplifying filter.

From planar Poiseuille to periodic flow + neumann external boundary conditions:

Go to the following function:

```
template <typename Real, template <typename U> class Descriptor>
auto inject_on_lattice_bc(MultiBlockLattice3D<Real, Descriptor>* target_lattice,
                          IncomprFlowParam<Real> const& parameters)
```

you will see that initial and boundary conditions are imposed with the help of "functors":

- `PoiseuilleVelocity<Real>(parameters);`

- `PoiseuilleDensity<Real,Descriptor>(parameters);`
- `PoiseuilleVelocityAndDensity<Real, Descriptor>(parameters).`

Look for equivalent function objects that allow imposing a constant velocity and density profile.

Use periodic boundaries on the z-normal boundary faces of the domain, Neumann boundary conditions on the y-normal, and use a uniform pressure and velocity initial condition.

7. Implement HW

Try to implement the HW boundary condition starting from the BFL (Bouzidi) implementation. To begin with do:

```
cd src
cp path/to/palabos/src/offLattice/bouzidiOffLatticeModel3D.h HWLatticeModel3D.h
cp path/to/palabos/src/offLattice/bouzidiOffLatticeModel3D.hh HWLatticeModel3D.hh
```

As firs step, rename all the "bouzidiOffLattice" naming to the respective "HWLattice". For example, from:

```
#ifndef BOUZIDI_OFF_LATTICE_MODEL_3D_H
#define BOUZIDI_OFF_LATTICE_MODEL_3D_H
```

to

```
#ifndef HW_LATTICE_MODEL_3D_H
#define HW_LATTICE_MODEL_3D_H
```

Also, search for all the occurrences of `BouzidiOffLatticeModel3D` in the two files and replace them with `HWLatticeModel3D`. Do the analogous thing also in the `.hh` file. Then comment out the following `#include` directives in the `.hh` file:

```
#include "offLattice/bouzidiOffLatticeModel3D.h"
#include "latticeBoltzmann/geometricOperationTemplates.h"
#include "latticeBoltzmann/externalFieldAccess.h"
#include <algorithm>
#include <vector>
#include <cmath>
```

and replace them with

```
#include "palabos3D.h"
#include "palabos3D.hh"  // explicit inclusion because it contains templates
```

Finally, navigate to `setup.h` and include the following preprocessor directives:

```
#include "HWLatticeModel3D.h"
#include "HWLatticeModel3D.hh"
```

Now you are ready to implement the HW. Open `HWLatticeModel3D.hh` and navigate to

```
template<typename T, template<typename U> class Descriptor>
void HWLatticeModel3D<T,Descriptor>::cellCompletion ()
```

and try to modify the algorithm to use a simple half-way bounce-back. Finally, test it!

Bug fix:

The following lines have been removed (they are not needed in this example, they slow the computations, but they should not change the results):

```
lattice.executeInternalProcessors();  // i.e. boundary conditions in
                                      // this case
lattice.incrementTime();
```

Questions? Read the FAQ