ภาคผนวก ก

โปรแกรมแบบจำลองแลตทิซ-โบทสแมนน์ ของ Palabos

แบบจำลองแลตทิซ-โบทสแมนน์ของ Palabos ที่นำมาใช้ในการทดสอบ เป็นตัวอย่างหนึ่งของโปรแกรมซึ่งทำการจำลองลักษณะการเคลื่อนที่พลศาสตร์ของไหล โดยสามารถกำหนดให้ทำการคำนวณค่าพลังงานจล, ค่าความหนาแน่น และค่าความเร็ว ในแต่ละรอบของการจำลองได้ตามต้องการ โดยให้เก็บผลลัพธ์ของการคำนวณเป็นไฟล์ข้อมูล และสามารถทำการแปลงข้อมูลให้เห็นเป็นรูปภาพได้ ซึ่งโปรแกรมตัวอย่างมีรายละเอียดดังนี้

```
1 /* This file is part of the Palabos library.
2  * Copyright (C) 2009 Jonas Latt
3  * E-mail contact: jonas@lbmethod.org
4  * The most recent release of Palabos can be downloaded at
5  * <http://www.lbmethod.org/palabos/>
6  *
7  * The library Palabos is free software: you can redistribute it
     and/or
8  * modify it under the terms of the GNU General Public License as
9  * published by the Free Software Foundation, either version 3 of
     the
10 * License, or (at your option) any later version.
11 *
12 * The library is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public
     License
18 * along with this program. If not, see
     <http://www.gnu.org/licenses/>.
19 */
20
21 #include "palabos2D.h"
22 #ifndef PLB_PRECOMPILED // Unless precompiled version is used,
23 #include "palabos2D.hh" // include full template code
24 #endif
25 #include "poiseuille.h"
26 #include "poiseuille.hh"
27
28 #include <vector>
29 #include <cmath>
30 #include <iostream>
31 #include <fstream>
32 #include <iomanip>
33
34 using namespace plb;
35 using namespace std;
```

```
36
37 typedef double T;
38 #define DESCRIPTOR descriptors::D2Q9Descriptor
39
40 int main(int argc, char* argv[]) {
41    plbInit(&argc, &argv);
42    global::directories().setOutputDir("./tmp/");
43
44    IncomprFlowParam<T> parameters (
45          (T) 1e-2,   // uMax
46          (T) 10.,    // Re
47          63,         // N
48          1.,         // lx
49          1.          // ly
50    );
51
52    plint nx = parameters.getNx();
53    plint ny = parameters.getNy();
54
55    writeLogFile(parameters, "Poiseuille flow");
56
57    MultiBlockLattice2D<T, DESCRIPTOR> lattice (
58          nx, ny,
59          new BGKdynamics<T,DESCRIPTOR>(parameters.getOmega()) );
60    OnLatticeBoundaryCondition2D<T,DESCRIPTOR>*
61            boundaryCondition =
              createLocalBoundaryCondition2D<T,DESCRIPTOR>();
62    createPoiseuilleBoundaries(lattice, parameters,
        *boundaryCondition);
63    lattice.initialize();
64
65    // The following command opens a text-file, in which the
velocity-profile
66    // in the middle of the channel will be written and several
successive
67    // time steps. Note the use of plb_ofstream instead of the
standard C++
68    // ofstream, which is required to guarantee a consistent
behavior in MPI-
69    // parallel programs.
70
71    plb_ofstream successiveProfiles("./tmp/KineticEnergy.txt");
72
73    // Main loop over time steps.
74    for (plint iT=0; iT<100; ++iT) { //10
75          if (iT%10==0) {
76                Box2D profileSection(1, nx-2, 1, ny-2);
77                pcout << "At iteration step " << iT
78                      << ", the density along the channel is " <<
endl;
79                pcout << setprecision(7)
80                      << *computeDensity(lattice, profileSection)
81                      << endl << endl;
82
83                successiveProfiles
84                      << setprecision(7)
85
```

```
86                      <<
computeMax(*computeKineticEnergy(lattice,profileSection
)) << " "
87                      <<
computeMin(*computeKineticEnergy(lattice,profileSection
)) << " "
88                      <<
*computeKineticEnergy(lattice,profileSection)
89                      //<< *computeDensity(lattice,profileSection)
90                      //<< *computeVelocity(lattice,
profileSection)
91
92                      << endl;
93
94                ImageWriter<T> imageWriter("air");
95                imageWriter.writeScaledGif(createFileName("u", iT,
4),
96                *computeKineticEnergy(lattice,profileSection));
97                //*computeDensity(lattice,profileSection));
98                //*computeVelocityNorm(lattice,profileSection));
99          }
100
101         // Lattice Boltzmann iteration step.
102         lattice.collideAndStream();
103   }
104
105   delete boundaryCondition;
106 }
107
```