Unit Verification and Validation Plan for Lattice Boltzmann Solver

Peter Michalski

October 22, 2019

1 Revision History

Date	Version	Notes
October 28, 2019	1.0	Initial Document

Contents

List of Tables

[Do not include if not relevant —SS]

List of Figures

[Do not include if not relevant —SS]

2 Symbols, Abbreviations and Acronyms

symbol	description
T	Test

[symbols, abbreviations or acronyms – you can reference the SRS, MG or MIS tables if needed —SS]

This document covers the unit verification and validation of Lattice Boltzmann Solver . Functional and non-functional requirements, as found in Section 3.1 and retrieved from the Commonality Analysis (Michalski [1]), will be tested. The document will outline the general information of the system in Section 3, which will be followed by a testing plan in Section 4. Finally, individual tests will be described in Section 5.

3 General Information

3.1 Purpose

The purpose of this plan is to verify that select functional requirements and non-functional requirements (NFR), as found in the Commonality Analysis (CA) titled Lattice Boltzmann Solvers (Michalski [1]), have been met in select system units.

The functional requirements are:

R1: The system shall read a set of input fluid parameters.

R2: The system shall allow the user to select from a set of model and velocity direction parameters.

R3: The system shall verify that the inputs fall within the allowable parameters of variation.

R4: The system shall instantiate required data types and structures for the selected model.

R5: The system shall import the relevant coefficient weights for the selected model.

R6: The system shall calculate and store the predicted fluid parameters, iterating through streaming and collision processes over the desired model time.

R7: The system shall output the results of the calculations in a manner consistent with the decisions made regarding variabilities.

The NFRs are:

1. Correctness: The allowable error of the output results, per module, will be less than the average error taken from the results of a sample of OTS solutions.

2. Maintainability: The system shall be documented with a Commonality Analysis (CA), Verification and Validation (VnV) plan, MG (Module Guide), MIS (Module Interface Specification), and User Guide.

3. Performance: The system shall be able to run MIN_INSTANCES at once.

4. Portability: The system shall be able to run on macOS, Windows, and Linux operating systems.

5. Reliability: The mean time between failures (MTBF) will be longer than the average MTBF of a sample of the OTS solutions.

6. Reusability: Individual modules of the system can be removed and reused in other systems.

7. Robustness: The system will not crash when a user provides invalid input.

8. Scalability: The system must be able to support additional computational models.

9. Understandability: New users must easily understand which models are available for their use.

10. Usability: Users will find the system easy to use.

3.2 Scope

The modules that will be tested include Poiseuille flow modeled in a D2Q9 lattice, and Von Karman Vortex Street modeled in a D2Q9 lattice. These are being tested as they can be verified against the pseudo oracle pyLBM. This unit VnV plan will test units within these two modules.

4 Plan

4.1 Verification and Validation Team

This VnV plan will be conducted by Peter Michalski.

4.2 Automated Testing and Verification Tools

Robot Framework for Python will be used for automated testing. The automated testing will be limited to checking if the right parameter from the input file have been assigned to the parameter variables in a module.

4.3 Non-Testing Based Verification

A code walkthrough of each module and unit will be conducted by the developer using the rubber duck method.

5 Unit Test Description

This test plan has been built with reference to the MIS document (REFERENCE) . The test cases have been selected by splitting each module into major components: input assignment, calculations, and output. Each of the these components will have one unit test.

5.1 Tests for Functional Requirements

MUST ADD TESTS FOR R2 R4 R5

5.1.1 Von Karman Vortex Street

[Include a blurb here to explain why the subsections below cover the module. References to the MIS would be good. You will want tests from a black box perspective and from a white box perspective. Explain to the reader how the tests were selected. —SS]

1. input-id1

   Type: Dynamic Automatic White Box

   Initial State: Input file is located in Input directory of LBM solver.

   Input: A comma delimited file with the following input parameters and values:
   Reynolds Number: 500
   Final time of simulation: 75
   Density: 1.0
   Bulk Viscosity: 1.e-3

   Output: The following values will be held in the module variables.
   Reynolds Number: 500
   Final time of simulation: 75
   Density: 1.0
   Bulk Viscosity: 1.e-3

Test Case Derivation: If the unit is set up correctly, then the appropriate variables for the module will be read from the file and assigned to the module variables of the same name.

How test will be performed:

(a) The input file will be placed into the Input folder and will contain the input variable names and values.

(b) The unit code will be placed into the Robot Framework Edit panel.

(c) The code will be run using the Robot Framework Run panel.

(d) The variable values will be output to the screen of the Run panel.

2. calculation-id2

Type: Dynamic Manual Black Box

Initial State: Module variables are instantiated with the values listed below:

Input:
Reynolds Number: 500
Final time of simulation: 75
Density: 1.0
Bulk Viscosity: 1.e-3

Output: The output will be a vorticity vector printed to the screen. This will be compared to the vorticity vector values from our pseudo oracle pyLBM

Test Case Derivation: The test is passed if the output results match the pseudo oracle results, within an acceptable percentage of error (2 percent - see assumption XX)

How test will be performed:

(a) The unit code is modified in PyCharm to have the above variables assigned the above values.

(b) The unit is run.

(c) Upon completion of the run, the output values of the vorticity vector will be compared to the vorticity vector values from Pylbm generated using the same input parameters.

3. output-id3

   Type: Dynamic Manual Black Box

   Initial State: Vorticity vector variable is instantiated with the vorticity vector values found in appendix XXX.

   Input: Vorticity vector values found in appendix XXX.

   Output: A file containing the image of the vorticity vector.

   Test Case Derivation: The test is passed if the unit successfully converts the input vorticity vector values into output as a jpeg image on a file. The correctness of the file will be judged visually by the reviewer, and compared to an image from the pseudo oracle using the same parameters.

   How test will be performed:

   (a) The vorticity vector variable will be initialized to the values found in Appendix XXX using pyCharm

   (b) The code will be run.

   (c) The output folder will be checked for a jpeg image.

4. author-walkthrough-id4

   Type: add test for walkthrough by developer as ms document is developed. issues shall be created in github. this is as per section 4.4 of system vnv

   Initial State:

   Input:

   Output:

   Test Case Derivation:

   How test will be performed:

5. peers-walkthrough-id5

Type: add test for walkthrough by developer after ms document is developed. issues shall be created in github. this is as per section 4.4 of system vnv

Initial State:

Input:

Output:

Test Case Derivation:

How test will be performed:

5.1.2 Poiseuille Flow

[Include a blurb here to explain why the subsections below cover the module. References to the MIS would be good. You will want tests from a black box perspective and from a white box perspective. Explain to the reader how the tests were selected. —SS]

1. input-id1

   Type: Dynamic Automatic White Box

   Initial State: Input file is located in Input directory of LBM solver.

   Input: A comma delimited file with the following input parameters and values:
   Final time of simulation: 50
   Density: 1.0
   Bulk Viscosity: 1.e-2
   Shear Viscosity: 1.e-2

   Output: The following values will be held in the module variables. parameters and values:
   Final time of simulation: 50
   Density: 1.0
   Bulk Viscosity: 1.e-2
   Shear Viscosity: 1.e-2

Test Case Derivation: If the unit is set up correctly, then the appropriate variables for the module will be read from the file and assigned to the module variables of the same name.

How test will be performed:

(a) The input file will be placed into the Input folder and will contain the input variable names and values.

(b) The unit code will be placed into the Robot Framework Edit panel.

(c) The code will be run using the Robot Framework Run panel.

(d) The variable values will be output to the screen of the Run panel.

2. calculation-id2

Type: Dynamic Manual Black Box

Initial State: Module variables are instantiated with the values listed below:

Input:
Final time of simulation: 75
Density: 1.0
Bulk Viscosity: 1.e-2
Shear Viscosity: 1.e-2

Output: The output will be a pressure gradient value printed to the screen. This will be compared to the pressure vector value from our pseudo oracle pyLBM

Test Case Derivation: The test is passed if the output results match the pseudo oracle results, within an acceptable percentage of error (2 percent - see assumption XX)

How test will be performed:

(a) The unit code is modified in PyCharm to have the above variables assigned the above values.

(b) The unit is run.

(c) Upon completion of the run, the output value of the pressure gradient will be compared to the pressure gradient value from Pylbm generated using the same input parameters.

3. output-id3

   Type: Dynamic Manual Black Box

   Initial State: Pressure gradient variable is instantiated with the randomly chosen value.

   Input: Pressure gradient variable is set to 20.

   Output: A file containing the value of the pressure gradient.

   Test Case Derivation: The test is passed if the unit writes the pressure gradient value to the output file.

   How test will be performed:

   (a) The pressure gradient variable will be initialized to 20 in PyCharm.

   (b) The code will be run.

   (c) The output folder will be checked for a file containing the pressure gradient value of 20.

4. author-walkthrough-id4

   Type: add test for walkthrough by developer as ms document is developed. issues shall be created in github. this is as per section 4.4 of system vnv

   Initial State:

   Input:

   Output:

   Test Case Derivation:

   How test will be performed:

5. peers-walkthrough-id5

   Type: add test for walkthrough by developer after ms document is developed. issues shall be created in github. this is as per section 4.4 of system vnv

Initial State:

Input:

Output:

Test Case Derivation:

How test will be performed:

5.2 Tests for Nonfunctional Requirements

5.2.1 Reusability

1. input-id1

   Type: Automatic Dynamic

   Initial State: An input file will be in the Input directory and will contain the values listed below.

   Input/Condition:
   Reynolds Number: 500
   Final time of simulation: 75
   Density: 1.0
   Bulk Viscosity: 1.e-3
   Shear Viscosity: 1.e-2

   Output/Result: The following values will be held in the Von Karman Vortex Street variables:
   Reynolds Number: 500
   Final time of simulation: 75
   Density: 1.0
   Bulk Viscosity: 1.e-3
   The following values will be held in the Poiseuille Flow variables:
   Final time of simulation: 75
   Density: 1.0
   Bulk Viscosity: 1.e-3
   Shear Viscosity: 1.e-2

   If the Von Karman Vortex Street and Poiseuille Flow modules have their variables correctly assigned to the above values then the test is passed, as we will know that the code for reading the input file is reusable between modules.

   How test will be performed:

   (a) An input file with the above variables and values will be placed in the input directory.

(b) The input units of the Von Karman Vortex Street and Poiseuille Flow modules will be run in Robot Framework.

(c) If the variable values in each of the modules match the input values in the file then the test is passed.

2. output-id2

Type: Manual Dynamic

Initial State: The vorticity vector of the Von Karman Vortex output unit is initialized.

Input: The vorticity vector values found in appendix XXX.

Output: A jpeg image. This will ensure that even randomized input, representing an external module that may use this code, will create some sort of an image file.

How test will be performed:

(a) The unit code is initialized with the vorticity vector values found in appendix XXX.

(b) The unit is run.

(c) Upon completion of the run, the output directory is checked for a jpeg image.

...

5.3 Traceability Between Test Cases and Modules

[Provide evidence that all of the modules have been considered. —SS]

References

[1] Peter Michalski. Lattice boltzmann solvers - ca. URL https://github.com/peter-michalski/LatticeBoltzmannSolvers/blob/master/docs/SRS/CA.pdf.

6 Appendix

[This is where you can place additional information, as appropriate —SS]

6.1 Symbolic Parameters

[The definition of the test cases may call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance. —SS]