Unit Verification and Validation Plan for Lattice Boltzmann Solver

Peter Michalski

December 12, 2019

1 Revision History

Date	Version	Notes
Dec. 11, 2019	1.0	Initial Document

Contents

List of Tables

2 Symbols, Abbreviations and Acronyms

Please see Section 2.2 and Section 2.3 of the Commonality Analysis (Michalski [1]).

This document covers the unit Verification and Validation (VnV) of Lattice Boltzmann Solver. Functional and non-functional requirements (NFRs), as found in Section 3.1 and retrieved from the Commonality Analysis (CA) (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 NFRs, as found in Section 7.1 and Section 7.2 of the CA titled Lattice Boltzmann Solvers (Michalski [1]), have been met in select system units.

3.2 Scope

The units that will be tested include those required for the modules of the Von Karman Vortex Street problem modeled in a D2Q9 lattice, as this will encompass the first stage of implementation.

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.

4.3 Non-Testing Based Verification

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

5 Unit Test Description

This test plan has been built with reference to the MIS document (Michalski [2]). Test cases have been selected to verify functionality of separate modules.

5.1 Tests for Functional Requirements

5.1.1 Module M2SystemControl

This module calls and combines all other modules. It is covered in entirety in Section 5.1.2 of the System VnV Plan (Michalski [3]), which conducts a black box test of the module. The tests below will verify subsections of the module, which are introduced in Section 6 of the MIS (Michalski [2]): that the problem and boundary data are correctly set up, and that the simulation is performed.

1. problem-boundary-data-id1

   Type: Dynamic Automatic Test

   Initial State: An input.txt file is in the Input directory, and the program is not running.

   Input:A file with inputs marked in the manner outlined in the User Guide (Michalski [4]).
   The input values for this test will be:

   Library 1
   Problem 1
   Dimensions 2
   VelocityDirections 9
   Size 1
   ReynoldsNumber 500
   Density 1
   BulkViscosity 0.001
   ShearViscosity 1
   Time 75
   Viscosity 1
   AccelerationRate 1

SpeedSound 1
Velocity 1
Force 1
Mass 1
MacroscopicVelocity 1
RelaxationRate 1

Output: It should be the same as that found in problem-boundary-data-id1.txt of the UnitVnVPlan directory.

Test Case Derivation: This module calls the M8Problem module to set up the problem and boundary data that will be input into the M5LBMControl module. In order for this to succeed it first passes the system input data into M8Problem module. This test verifies that what is returned by M8Problem module is correctly formatted, and that the problem parameters and boundary data variables of this module correctly store the data.

How test will be performed:

(a) Outside of the system, the input parameter values will be written to a text file titled input.txt, as outlined in Section 4.2.1 of the User Guide.

(b) The file will be placed into the Input directory, under the home directory of the project.

(c) M2SystemControl module shall be modified to print the problem parameters and boundary data variable contents.

(d) Lattice Boltzmann Solver shall be run.

(e) The output values that are printed shall be compared with problem-boundary-data-id1.txt. This can be done by hashing and comparing the contents. We can use hashing for this test since the problem module should return exactly the same contents as the pseudo-oracle. There is no math involved in setting up these parameters, only the structure of the problem data is created.

2. simulation-id2

Type: Dynamic Automatic Test

Initial State: An input.txt file is in the Input directory, and the program is not running.

Input:A file with inputs marked in the manner outlined in the User Guide (Michalski [4]).
The input values for this test will be:

Library 1
Problem 1
Dimensions 2
VelocityDirections 9
Size 1
ReynoldsNumber 500
Density 1
BulkViscosity 0.001
ShearViscosity 1
Time 75
Viscosity 1
AccelerationRate 1
SpeedSound 1
Velocity 1
Force 1
Mass 1
MacroscopicVelocity 1
RelaxationRate 1

Output: It should be the same as that found in simulation-id2.txt of the UnitVnVPlan directory.

Test Case Derivation: This module calls the M5LBMControl module to perform LBM calculations. In the initial development of ProgName M5LBMControl will not be implemented as a separate module, but will instead call a function of the external pyLBM library. This test verifies that the solution returned from the pyLBM function is correct.

How test will be performed:

(a) Outside of the system, the input parameter values will be written to a text file titled input.txt, as outlined in Section 4.2.1 of the User Guide.

(b) The file will be placed into the Input directory, under the home directory of the project.

(c) M2SystemControl module shall be modified to print the pyLBM return contents.

(d) Lattice Boltzmann Solver shall be run.

(e) The output values that are printed shall be compared with simulation-id2.txt. This can be done by hashing and comparing the contents. We can further compare sections of the output with the pseudo-oracle using a relative error calculation, as found in Section 6.1.

5.1.2 Module M3InputReading

This module is partially covered in the Input Reading Section of Section 5.1.1 of the System VnV Plan (Michalski [3]). That document tests the ability of the system to read the input file, as well as checks that a correctly formatted input file is read correctly. The test found below will check if the module correctly handles not being able to find an input file.

1. input-reading-id3

   Type: Dynamic Automatic Test

   Initial State: An input.txt file is NOT in the Input directory, and the program is not running.

   Input: There is no input for this test.

   Output: An error message of "Input file not found." will be printed to log_file.log.

   Test Case Derivation: The system will need to handle a condition where the input file is not found in the appropriate directory. A descriptive error message needs to notify the user of what condition caused the program to fail.

   How test will be performed:

   (a) An input.txt file will not be in the Input directory.

   (b) M3InputReading module will be selected to run.

   (c) The user will check log_file.log for the correct error message.

5.1.3 Module M4InputChecking

This module is partially covered in the Input Bounds Section of Section 5.1.1 of the System VnV Plan (Michalski [3]). That document tests the ability of the system to handle minimum and maximum input bounds of several input variables. The tests found below will check and handle situations where the key input variables are known or unknown to the system, and where a specific library problem does or does not have all required inputs.

1. known-input-variables-id4

   Type: Dynamic Automatic Test

   Initial State: An input.txt file is in the Input directory, and the program is not running.

   Input:A file with inputs marked in the manner outlined in the User Guide (Michalski [4]).
   The input values for this test will be:

   Library 1
   Problem 1
   Dimensions 2
   VelocityDirections 9
   Size 1
   ReynoldsNumber 500
   Density 1
   BulkViscosity 0.001
   ShearViscosity 1
   Time 75
   Viscosity 1
   AccelerationRate 1
   SpeedSound 1
   Velocity 1
   Force 1
   Mass 1
   MacroscopicVelocity 1
   RelaxationRate 1

Output: A message of "All parameters known." printed to the screen.

Test Case Derivation: The first section of this module will check if the user provided input keys of the input.txt file are known to the system. If the keys are known, then the test will have the system print a confirmation message. If any keys are unknown then there will be an error message printed to the log_file.log file, and the system will exit.

How test will be performed:

(a) Outside of the system, the input parameter values will be written to a text file titled input.txt, as outlined in Section 4.2.1 of the User Guide.

(b) The file will be placed into the Input directory, under the home directory of the project.

(c) M4InputChecking module shall be modified to print "All parameters known." if all keys are known.

(d) Lattice Boltzmann Solver shall be run.

(e) The output shall be observed for the above message.

2. unknown-input-variables-id5

Type: Dynamic Automatic Test

Initial State: An input.txt file is in the Input directory, and the program is not running.

Input:A file with inputs marked in the manner outlined in the User Guide (Michalski [4]).
The input values for this test will be:

Library 1
Problem 1
Dimensions 2
FakeField 9
Size 1
ReynoldsNumber 500
Density 1
BulkViscosity 0.001

ShearViscosity 1
Time 75
Viscosity 1
AccelerationRate 1
SpeedSound 1
Velocity 1
Force 1
Mass 1
MacroscopicVelocity 1
RelaxationRate 1

Output: A message of "The parameter FakeField is not known to the system. Please see the User Guide." printed to log_file.log.

Test Case Derivation: The first section of this module will check if the user provided input keys of the input.txt file are known to the system. If the keys are known, then the test will have the system print a confirmation message. If any keys are unknown then there will be an error message printed to the log_file.log file, and the system will exit.

How test will be performed:

(a) Outside of the system, the input parameter values will be written to a text file titled input.txt, as outlined in Section 4.2.1 of the User Guide.

(b) The file will be placed into the Input directory, under the home directory of the project.

(c) Lattice Boltzmann Solver shall be run.

(d) The log file will be checked for the appropriate error message.

3. missing-input-requirements-id6

Type: Dynamic Automatic Test

Initial State: An input.txt file is in the Input directory, and the program is not running.

Input:A file with inputs marked in the manner outlined in the User Guide (Michalski [4]).

The input values for this test will be:

Library 1
Problem 1
Dimensions 2
Size 1
ReynoldsNumber 500
Density 1
BulkViscosity 0.001
ShearViscosity 1
Time 75
Viscosity 1
AccelerationRate 1
SpeedSound 1
Velocity 1
Force 1
Mass 1
MacroscopicVelocity 1
RelaxationRate 1

Output: A message of "The input.txt file is missing (or has incorrect) required parameters for the designated problem. Please see the User Guide." printed log_file.log.

Test Case Derivation: The final section of this module will check if the selected Library and Problem have all addition required inputs present in the input.txt file. If any are missing there will be an error message printed to the log_file.log file, and the system will exit. This test will verify that the system will recognize if some required inputs are missing and print an appropriate error message.

How test will be performed:

(a) Outside of the system, the input parameter values will be written to a text file titled input.txt, as outlined in Section 4.2.1 of the User Guide.

(b) The file will be placed into the Input directory, under the home directory of the project.

(c) Lattice Boltzmann Solver shall be run.

(d) The log file shall be read.

4. present-input-requirements-id7

Type: Dynamic Automatic Test

Initial State: An input.txt file is in the Input directory, and the program is not running.

Input:A file with inputs marked in the manner outlined in the User Guide (Michalski [4]).
The input values for this test will be:

Library 1
Problem 1
Dimensions 2
VelocityDirections 9
Size 1
ReynoldsNumber 500
Density 1
BulkViscosity 0.001
ShearViscosity 1
Time 75
Viscosity 1
AccelerationRate 1
SpeedSound 1
Velocity 1
Force 1
Mass 1
MacroscopicVelocity 1
RelaxationRate 1

Output: A message of "Required input variables present" printed to the screen.

Test Case Derivation: The final section of this module will check if the selected Library and Problem have all addition required inputs present in the input.txt file. If any are missing there will be an error message

printed to the log_file.log file, and the system will exit. This test will verify that if all required inputs are present, the system will know to proceed.

How test will be performed:

(a) Outside of the system, the input parameter values will be written to a text file titled input.txt, as outlined in Section 4.2.1 of the User Guide.

(b) The file will be placed into the Input directory, under the home directory of the project.

(c) M4InputChecking module shall be modified to print "Required input variables present" if all required inputs are present.

(d) Lattice Boltzmann Solver shall be run.

(e) The output shall be observed for the verifying message.

5.1.4 Module M5LBMControl

This section will not be implemented in the first iteration of system development. This is because of two reasons: 1) there is an available external library for the solution of the single problem that the first iteration will provide; and 2) since we are only implementing one library in the first iteration, we will not need a control module for multiple libraries. The functionalities covered by this module will be tested in entirety in Section 5.1.2 of the System VnV Plan (Michalski [3]), which conducts a black box test which includes the submodules of this module.

5.1.5 Module M6Streaming

This section will not be implemented in the first iteration of system development for reasons stated in 5.1.4.

5.1.6 Module M7Collision

This section will not be implemented in the first iteration of system development for reasons stated in 5.1.4.

5.1.7 Module M8Problem

This module sets up the structure of the LBM input parameters, as outlined in Section 12 of the MIS (Michalski [2]. Since we are only implementing one library and problem in the first iteration of system development we will only test this module for that problem, comparing the output to a pseudo-oracle implementation of pyLBM.

1. problem-output-id8

   Type: Dynamic Automatic Test

   Initial State: An input.txt file is in the Input directory, and the program is not running.

   Input:A file with inputs marked in the manner outlined in the User Guide (Michalski [4]).
   The input values for this test will be:

   Library 1
   Problem 1
   Dimensions 2
   VelocityDirections 9
   Size 1
   ReynoldsNumber 700
   Density 1
   BulkViscosity 0.01
   ShearViscosity 1
   Time 80
   Viscosity 1
   AccelerationRate 1
   SpeedSound 1
   Velocity 1
   Force 1
   Mass 1
   MacroscopicVelocity 1
   RelaxationRate 1

Output: It should be the same as that found in problem-output-id8.txt of the UnitVnVPlan directory.

Test Case Derivation: This module sets up the problem and boundary data that will be input into the M5LBMControl module. This test verifies that what is returned by the module is correctly formatted by comparing its output to a pseudo-oracle.

How test will be performed:

(a) Outside of the system, the input parameter values will be written to a text file titled input.txt, as outlined in Section 4.2.1 of the User Guide.

(b) The file will be placed into the Input directory, under the home directory of the project.

(c) M8Problem module shall be modified to print the return variable.

(d) Lattice Boltzmann Solver shall be run.

(e) The output values that are printed shall be compared with problem-output-id8.txt. This can be done by hashing and comparing the contents. We can further compare sections of the output with the pseudo-oracle using a relative error calculation, as found in Section 6.1.

5.1.8 Module M9Lattice

This module sets up the structure of the lattice model, as outlined in Section 13 of the MIS (Michalski [2]. The single library that will be implemented in the first iteration of system development will not require the functionality of this module, as it will be part of the library in M5LBMControl module; however, preliminary development of this module will be started, which will be tested here.

1. lattice-return-id9

   Type: Dynamic Automatic Test

   Initial State: The initial state is that of the program not running.

   Input: No input is necessary. Instead there will be a slight modification as outlined below.

Output: "[0.4444444444444444, 0.1111111111111111, 0.1111111111111111, 0.1111111111111111, 0.1111111111111111, 0.02777777777777776, 0.02777777777777776, 0.02777777777777776, 0.02777777777777776]" printed to the screen.

Test Case Derivation: This module returns the velocity weights for an LBM problem with specific standard dimensions and velocity directions. This test will return the velocity weights for a problem that has 2 dimensions and 9 velocity directions.

How test will be performed:

(a) M9Lattice module shall be modified to print the return variable for a problem with 2 dimensions and 9 velocity directions.

(b) M9Lattice module shall be modified to set the dimensions to 2, and the velocity directions to 9 outside of the module function. The module will further be modified to call itself to run.

(c) The module shall be run.

(d) The output values that are printed shall be compared with the expected output values.

5.1.9 Module M10Boundary

This module sets up the structure of the model boundary, as outlined in Section 14 of the MIS (Michalski [2]. The first iteration of system development will include a single fixed boundary. The entirety of the module will be tested here, returning the single fixed boundary.

1. boundary-return-id10

   Type: Dynamic Automatic Test

   Initial State: An input.txt file is in the Input directory, and the program is not running.

   Input:A file with inputs marked in the manner outlined in the User Guide (Michalski [4]).
   The input values for this test will be:

Library 1
Problem 1
Dimensions 2
VelocityDirections 9
Size 1
ReynoldsNumber 700
Density 1
BulkViscosity 0.01
ShearViscosity 1
Time 80
Viscosity 1
AccelerationRate 1
SpeedSound 1
Velocity 1
Force 1
Mass 1
MacroscopicVelocity 1
RelaxationRate 1

Output: "{'x_min': 0.0, 'x_max': 3.0, 'y_min': 0.0, 'y_max': 1.0}" printed to the screen.

Test Case Derivation: This module returns the boundary dimensions of SIZE 1, which represents a fixed dimension boundary for a problem with 2 dimensions and 9 velocity directions.

How test will be performed:

(a) Outside of the system, the input parameter values will be written to a text file titled input.txt, as outlined in Section 4.2.1 of the User Guide.

(b) The file will be placed into the Input directory, under the home directory of the project.

(c) M10Boundary module shall be modified to print the return variable.

(d) Lattice Boltzmann Solver shall be run.

(e) The output values that are printed shall be compared with the expected output values.

5.1.10 Module M11ImageRendering

This module will rely on an external library and will not be tested. It will be assumed that the external library performs its functions correctly.

5.1.11 Module M12DataStructure

This module will not be tested in the first iteration of system development as it will only set a few variables and a simple dictionary.

5.1.12 Module M13InputTypes

This module will not yet be tested as it will currently only set a variable.

5.2 Tests for Nonfunctional Requirements

Several tests for NFRs have been covered in Section 5.2 of the System VnV Plan (Michalski [3]). The tests below will add to and further enhance NFR testing. The tests below will cover performance and scalability. Robustness has been tested in Section 5.1.2 and Section 5.1.3.

5.2.1 Module M2SystemControl

1. performance-id11

   Type: Dynamic Automatic Test

   Initial State: An input.txt file is in the Input directory, and the program is not running.

   Input:A file with inputs marked in the manner outlined in the User Guide (Michalski [4]).
   The input values for this test will be: Please see Table 3 in Section 6.2.

   Test Case Derivation: This case is a comparison of the running time of the simulation section of this module representing the M5LBMControl module. The ReynoldsNumber parameter will be incremented in 5 iterations of the test, and the output time will be compared. This will allow us to see how changes in such parameters affect computational time of our program.

   How test will be performed:

   (a) M2SystemControl will be modified to time the running of the simulation section of the module representing the M5LBMControl module. The time will be output to the screen.

   (b) Outside of the system, the input parameter values will be written to a text file titled input.txt, as outlined in Section 4.2.1 of the User Guide.

   (c) The file will be placed into the Input directory, under the home directory of the project.

   (d) Lattice Boltzmann Solver shall be run.

   (e) The output time shall be noted.

(f) Steps (b) to (e) will be repeated for all 5 iterations.

(g) The timing of each iteration will be compared and graphed.

5.2.2 Module M9Lattice

1. scalability-id12

   Type: Manual

   Initial State: The initial state is that of the program not running.

   Input: No input is necessary. Instead there will be a slight modification as outlined below.

   Output: "[0.6666666666666667, 0.1666666666666667, 0.1666666666666667]" printed to the screen.

   Test Case Derivation: This module returns the velocity weights for a second set of standard LBM dimensions and velocity directions. This test will return the velocity weights for a problem that has 1 dimension and 3 velocity directions.

   How test will be performed:

   (a) M9Lattice module shall be modified to print the return variable for a problem with 1 dimensions and 3 velocity directions.

   (b) M9Lattice module shall be modified to set the dimensions to 1, and the velocity directions to 3 outside of the module function. The module will further be modified to call itself to run.

   (c) M9Lattice shall be modified to return the above output values when a problem with 1 dimension and 3 velocity directions is called.

   (d) The module shall be run.

   (e) The output values that are printed shall be compared with the expected output values.

5.3 Traceability Between Test Cases and Modules

	R1	R2	R3	R4	R5	R6	R7
id1	✓	✓	✓	✓	✓		
id2	✓	✓	✓	✓	✓	✓	
id3	✓						
id4	✓	✓	✓				
id5	✓	✓	✓				
id6	✓	✓	✓				
id7	✓		✓				
id8	✓	✓	✓	✓	✓		
id9		✓		✓	✓		
id10	✓	✓	✓	✓			
id11	✓	✓	✓	✓	✓	✓	
id12		✓		✓	✓		

Table 1: Traceability Matrix Showing the Connections Between Test Cases and Functional Requirements

Cases / NFR	1	2	3	4	5	6	7	8	9	10
id1										
id2	✓									
id3										
id4										
id5										
id6										
id7										
id8	✓									
id9										
id10										
id11			✓							
id12								✓		

Table 2: Traceability Matrix Showing the Connections Between Test Cases and NFRs

References

[1] Peter Michalski. Lattice Boltzmann Solvers - CA, . URL https://github.com/peter-michalski/LatticeBoltzmannSolvers/blob/master/docs/SRS/CA.pdf.

[2] Peter Michalski. Module Interface Specification for Lattice Boltzmann Solvers, . URL https://github.com/peter-michalski/LatticeBoltzmannSolvers/blob/master/docs/Design/MIS/MIS.pdf.

[3] Peter Michalski. System Verification and Validation Plan for Lattice Boltzmann Solver, . URL https://github.com/peter-michalski/LatticeBoltzmannSolvers/blob/master/docs/VnVPlan/SystemVnVPlan/SystemVnVPlan.pdf.

[4] Peter Michalski. Lattice Boltzmann Solvers - User Guide, . URL https://github.com/peter-michalski/LatticeBoltzmannSolvers/tree/master/docs/UserGuide.

6 Appendix

6.1 Mathematical Equation for Error

For the purpose of testing, the equation for error is:

$$\% \text{ relative error} = \frac{|ProgName\ output\ -\ pseudo{-}oracle\ output|}{pseudo{-}oracle\ output}$$

6.2 M2SystemCOntrol Performance Test Inputs

Input	Iteration1	Iteration2	Iteration3	Iteration4	Iteration5
Library	1	1	1	1	1
Problem	1	1	1	1	1
Dimensions	2	2	2	2	2
VelocityDirections	9	9	9	9	9
Size	1	1	1	1	1
ReynoldsNumber	100	200	400	800	1200
Density	1	1	1	1	1
BulkViscosity	0.01	0.01	0.01	0.01	0.01
ShearViscosity	1	1	1	1	1
Time	80	80	80	80	80
Viscosity	1	1	1	1	1
AccelerationRate	1	1	1	1	1
SpeedSound	1	1	1	1	1
Velocity	1	1	1	1	1
Force	1	1	1	1	1
Mass	1	1	1	1	1
MacroscopicVelocity	1	1	1	1	1
RelaxationRate	1	1	1	1	1

Table 3: M2SystemControl Performance Inputs