System Verification and Validation Plan for Lattice Boltzmann Solver

Peter Michalski

October 20, 2019

1 Revision History

Date	Version	Notes
October 28, 2019	1.0	Initial Document

Contents

1 Revision History **i**

2 Symbols, Abbreviations and Acronyms **iv**

3 General Information **1**
- 3.1 Summary . . . 1
- 3.2 Objectives . . . 1
- 3.3 Relevant Documentation . . . 3

4 Plan **4**
- 4.1 Verification and Validation Team . . . 4
- 4.2 CA Verification Plan . . . 4
- 4.3 Design Verification Plan . . . 4
- 4.4 Implementation Verification Plan . . . 4
- 4.5 Software Validation Plan . . . 5

5 System Test Description **6**
- 5.1 Tests for Functional Requirements . . . 6
  - 5.1.1 Input . . . 6
  - 5.1.2 Von Karman Vortex Street . . . 9
  - 5.1.3 Poiseuille Flow . . . 18
- 5.2 Tests for Nonfunctional Requirements . . . 26
  - 5.2.1 Correctness . . . 26
  - 5.2.2 Maintainability . . . 26
  - 5.2.3 Performance . . . 27
  - 5.2.4 Portability . . . 28
  - 5.2.5 Robustness . . . 29
  - 5.2.6 Usability . . . 29
- 5.3 Traceability Between Test Cases and Requirements . . . 31

6 Appendix **34**
- 6.1 Symbolic Parameters . . . 34
- 6.2 Usability Survey Questions . . . 34
- 6.3 Input Data . . . 35

List of Tables

1 Traceability Matrix Showing the Connections Between Test Cases and Functional Requirements 31
2 Traceability Matrix Showing the Connections Between Test Cases and NFRs 32
3 Input Data Bounds 35

List of Figures

1 Input out of Bounds 8

2 Symbols, Abbreviations and Acronyms

symbol	description
1D	One Dimensional
2D	Two Dimensional
C	Pseudo-Oracle Control Value
CA	Commonality Analysis
D	All Input Values
I	Individual Parameter Input Values
IDE	Integrated Development Environment
LBM	Lattice Boltzmann Method
LBS	Lattice Boltzmann Solvers
MG	Module Guide
MIS	Module Interface Specification
MTBF	Mean Time Between Failures
NFR	Non-Functional Requirement
O	Program Output
OTS	Off The Shelf
P	Program
R	Functional Requirement
SRS	Software Requirement Specification
VnV	Verification and Validation
Re	Reynolds Number
p	Density
t	Time
η_b	Bulk Viscosity
η_s	Shear Viscosity

This document covers the system verification and validation of Lattice Boltzmann Solver. Functional and non-functional requirements, as found in Section 3.2 and retrieved from the Commonality Analysis (Michalski [2]), 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 Summary

The software being tested is Lattice Boltzmann Solver. This software reads inputs from a file, and calculates outputs using Lattice Boltzmann Methods (LBM).The software outputs the results to the screen, to a file, and/or to memory.

Lattice Boltzmann Solver can model many fluid dynamics problems, including Poiseuille Flow and Von Karman Vortex Street. Both of these problems will be used in our functional test cases, as the results of Lattice Boltzmann Solvercan be compared to the results of the pseudo-oracle pyLBM (pyl [1]).

3.2 Objectives

The intended objective of this plan is to verify that functional requirements and non-functional requirements (NFR), as found in the Commonality Analysis (CA) titled Lattice Boltzmann Solvers (Michalski [2]), have been met.

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 CA, VnV, MG, and MIS.

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.

3.3 Relevant Documentation

The Commonality Analysis of Lattice Boltzmann Solvercan be found in (Michalski [2]).

4 Plan

4.1 Verification and Validation Team

This VnV plan will be conducted by Peter Michalski.

4.2 CA Verification Plan

The CA of Lattice Boltzmann Solver shall be verified in the following ways:

1. Feedback: Reviewers from the class shall provide feedback on GitHub.
2. Static Analysis - Walkthrough: The document shall be manually reviewed by the writer using the SRS checklist upon its initial creation, as found in the CAS741 GitLab repository (Smith [5]).
3. Static Analysis - Review: The document shall be manually reviewed by the writer using the SRS checklist after VnV completion, as found in the CAS741 GitLab repository (Smith [5]).
4. Static Analysis - Final: The document shall be manually reviewed by the writer using the SRS checklist after MG and MIS development, as found in the CAS741 repository (Smith [5]).

4.3 Design Verification Plan

The design shall be verified by ensuring that functional requirements and NFRs are tested, as listed in Section 3.2.

The system functional requirements shall be tested first, as outlined in 5.1. This will be followed by reviewing the NFRs as outlined in 5.2.

4.4 Implementation Verification Plan

The implementation shall be verified in the following ways:

1. Code walkthrough - author: Module unit code shall be inspected for functional errors by the author as the MS document is developed. Defects shall be noted in GitHub using issue creation. This plan is described in Section 5.1 of the document Unit Verification and Validation Plan for Lattice Boltzmann Solver (Michalski [3]).

2. Code walkthrough - peers: Module unit code shall be inspected for functional errors by class peers after the MS document is developed. Defects shall be noted in GitHub using issue creation. This plan is described in Section 5.1 of the document Unit Verification and Validation Plan for Lattice Boltzmann Solver (Michalski [3]).

3. Dynamic unit tests: Dynamic unit test will be carried out, as listed in Section 5.1 of the document Unit Verification and Validation Plan for Lattice Boltzmann Solver (Michalski [3]).

4. System tests: System tests will be carried out, as listed in Section 5.

4.5 Software Validation Plan

The system shall be validated by ensuring that all functional requirements, as listed in Section 3.2, are met. The pseudo-oracle data that the output can be tested against can be found in Section **??**.

5 System Test Description

5.1 Tests for Functional Requirements

The subsections below are designed to cover several of the functional requirements of the system. Subsection 5.1.1 will cover R1 and R3. Subsections 5.1.2 and 5.1.3 will cover R6 and R7, and will test out the input vales at their allowable edges. The remaining functional requirements may be tested using the Unit Vnv Plan.

5.1.1 Input

Input Reading

1. input-reading-id1

   Control: Program (P) reading of input values (D) from a file.

   Initial State: No input file in Input directory.

   Input: A comma delimited file, with inputs marked in the manner outlined in the User Guide (Michalski [4]).
   The inputs values for this test will be:
   Re: 500
   Final time of simulation: 75
   Density: 1.0
   Bulk Viscosity: 1.e-3

   Output: All inputs printed to the screen as output (O). An error message of "cannot read file" should be shown if the system could not read the file.
   Outputs printed to the screen:
   Re: 500
   t: 75
   p: 1.0
   η_b: 1.e-3

Test Case Derivation: P(D) is correct if P:D -> O and O = D

The result required for the test to pass is the successful printing of all input values to the screen. This test satisfies R1.

How test will be performed:

(a) Outside of the system, the input parameter values will be written to a comma delimited text file, as outlined in the User Guide.
(b) The file will be placed into the Input directory, under the home directory of the project.
(c) The module TestInput found under the home directory will be run using pyCharm.
(d) If successful, The system will output the input parameters to the screen. If unsuccessful, the system will output an error message to the screen.

Input Bounds

1. input-bounds-id2

Control: Program (P) testing of input values (D) read from a file.

Initial State: No input file in Input directory.

Input: A comma delimited file, with inputs marked in the manner outlined in the User Guide (Michalski [4]).
The inputs values for this test will be:
Re: 500000
t: 0
p: 15
η_b: 100
η_s: 30000

Output: A descriptive failure message printed to the screen as seen below. In the below figure, X will contain all parameter names that

are out of bounds.

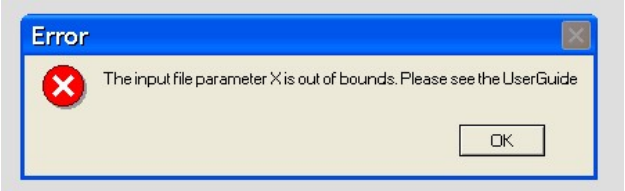

Figure 1: Input out of Bounds

Test Case Derivation: I (individual inputs) are an element of D P(D) is incorrect if P:D and at least one I is out of bounds as per Table 3: Input Data Bounds. This test satisfies R3.

How test will be performed:

(a) Outside of the system, the input values will be written to a comma delimited text file, as outlined in the User Guide (Michalski [4]).

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

(c) The module TestInput found under the home directory will be run using a python IDE.

(d) If the test is successful, The system will output a descriptive error message, as seen above, to the screen.

5.1.2 Von Karman Vortex Street

Dynamic Testing - Manual:

These dynamic tests will compare output with output from the pseudo-oracle myLBM (pyl [1]). The pseudo-oracle will have the same inputs as Lattice Boltzmann Solver for each test. The allowable error is AVERAGE_ERROR, found in Section 6.1.

1. tutorial-test-id3

   Control: Program (P) module execution using inputs (D), and expecting output (O) to match a control value (C).

   Initial State: No input file in Input directory.

   Input: A comma delimited file, with inputs marked in the manner outlined in the User Guide (Michalski [4]).
   The inputs values for this test will be:
   Re: 500
   t: 75
   p: 1.0
   η_b: 1.e-3

   Output: Vorticity vector values printed to the screen.

   Test Case Derivation: This case is a comparison with the pseudo-oracle pyLBM. The output vector values of this test for pyLBM can be found in the file id3output.txt located in the OracleOutput folder. The result required for the test to pass is an average error rate between cells of the output vector (O) and (C) of less than or equal to AVERAGE_ERROR. This test satisfies R6 and R7.

   How test will be performed:

   (a) The Von Karman Vortex Street module will be adjusted to print the vorticity vector values to the screen.

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

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

(d) The module for Von Karman Vortex Street will be selected to run.

(e) Upon completion of the module, the output values of the vorticity vector will be compared to the vorticity vector values from pyLBM - comparison will be done per cell. Comparisons can be done manually using Excel, or through a script.

2. laminar-test-id4

Control: Program (P) module execution using inputs (D), and expecting output (O) to match a control value (C).

Initial State: No input file in Input directory.

A comma delimited file, with inputs marked in the manner outlined in the User Guide (Michalski [4]).
The inputs values for this test will be:
Re: 0.0001
t: 75
p: 1.0
η_b: 1.e-3

Output: Vorticity vector values printed to the screen.

Test Case Derivation: This case is a comparison with the pseudo-oracle pyLBM. The output vector values of this test for pyLBM can be found in the file id4output.txt located in the OracleOutput folder. The result required for the test to pass is an average error rate between cells of the output vector (O) and (C) of less than or equal to AVERAGE_ERROR. This test satisfies R6 and R7. It also covers a very small Reynolds number representing laminar flow.

How test will be performed:

(a) The Von Karman Vortex Street module will be adjusted to print the vorticity vector values to the screen.

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

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

(d) The module for Von Karman Vortex Street will be selected to run.

(e) Upon completion of the module, the output values of the vorticity vector will be compared to the vorticity vector values from pyLBM - comparison will be done per cell. Comparisons can be done manually using Excel, or through a script.

3. turbulent-test-id5

Control: Program (P) module execution using inputs (D), and expecting output (O) to match a control value (C).

Initial State: No input file in Input directory.

A comma delimited file, with inputs marked in the manner outlined in the User Guide (Michalski [4]).
The inputs values for this test will be:
Re: 50000
t: 75
p: 1.0
η_b: 1.e-3

Output: Vorticity vector values printed to the screen.

Test Case Derivation: This case is a comparison with the pseudo-oracle pyLBM. The output vector values of this test for pyLBM can be found in the file id3output.txt located in the OracleOutput folder. The result

required for the test to pass is an average error rate between cells of the output vector (O) and (C) of less than or equal to AVERAGE_ERROR. This test satisfies R6 and R7. It also covers a very large Reynolds number representing very turbulent flow.

How test will be performed:

(a) The Von Karman Vortex Street module will be adjusted to print the vorticity vector values to the screen.

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

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

(d) The module for Von Karman Vortex Street will be selected to run.

(e) Upon completion of the module, the output values of the vorticity vector will be compared to the vorticity vector values from pyLBM - comparison will be done per cell. Comparisons can be done manually using Excel, or through a script.

4. low-density-test-id6

Control: Program (P) module execution using inputs (D), and expecting output (O) to match a control value (C).

Initial State: No input file in Input directory.

A comma delimited file, with inputs marked in the manner outlined in the User Guide (Michalski [4]).
The inputs values for this test will be:
Re: 500
t: 75
p: 7.08e-2
η_b: 1.e-3

Output: Vorticity vector values printed to the screen.

Test Case Derivation: This case is a comparison with the pseudo-oracle pyLBM. The output vector values of this test for pyLBM can be found in the file id3output.txt located in the OracleOutput folder. The result required for the test to pass is an average error rate between cells of the output vector (O) and (C) of less than or equal to AVERAGE_ERROR. This test satisfies R6 and R7. It also covers the low bound for density using a density of liquid hydrogen.

How test will be performed:

(a) The Von Karman Vortex Street module will be adjusted to print the vorticity vector values to the screen.

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

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

(d) The module for Von Karman Vortex Street will be selected to run.

(e) Upon completion of the module, the output values of the vorticity vector will be compared to the vorticity vector values from pyLBM - comparison will be done per cell. Comparisons can be done manually using Excel, or through a script.

5. high-density-test-id7

Control: Program (P) module execution using inputs (D), and expecting output (O) to match a control value (C).

Initial State: No input file in Input directory.

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

Re: 500
t: 75
p: 13.6
η_b: 1.e-3

Output: Vorticity vector values printed to the screen.

Test Case Derivation: This case is a comparison with the pseudo-oracle pyLBM. The output vector values of this test for pyLBM can be found in the file id3output.txt located in the OracleOutput folder. The result required for the test to pass is an average error rate between cells of the output vector (O) and (C) of less than or equal to AVERAGE_ERROR. This test satisfies R6 and R7. It also covers the high bound for density using a density close to that of mercury.

How test will be performed:

(a) The Von Karman Vortex Street module will be adjusted to print the vorticity vector values to the screen.

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

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

(d) The module for Von Karman Vortex Street will be selected to run.

(e) Upon completion of the module, the output values of the vorticity vector will be compared to the vorticity vector values from pyLBM - comparison will be done per cell. Comparisons can be done manually using Excel, or through a script.

6. low-bulk-viscosity-test-id8

Control: Program (P) module execution using inputs (D), and expecting output (O) to match a control value (C).

Initial State: No input file in Input directory.

A comma delimited file, with inputs marked in the manner outlined in the User Guide (Michalski [4]).
The inputs values for this test will be:
Re: 500
t: 75
p: 1.0
η_b: 1.e-4

Output: Vorticity vector values printed to the screen.

Test Case Derivation: This case is a comparison with the pseudo-oracle pyLBM. The output vector values of this test for pyLBM can be found in the file id3output.txt located in the OracleOutput folder. The result required for the test to pass is an average error rate between cells of the output vector (O) and (C) of less than or equal to AVERAGE_ERROR. This test satisfies R6 and R7. It also covers the low bound for bulk viscosity.

How test will be performed:

(a) The Von Karman Vortex Street module will be adjusted to print the vorticity vector values to the screen.

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

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

(d) The module for Von Karman Vortex Street will be selected to run.

(e) Upon completion of the module, the output values of the vorticity vector will be compared to the vorticity vector values from pyLBM - comparison will be done per cell. Comparisons can be done manually using Excel, or through a script.

7. high-bulk-viscosity-test-id9

   Control: Program (P) module execution using inputs (D), and expecting output (O) to match a control value (C).

   Initial State: No input file in Input directory.

   A comma delimited file, with inputs marked in the manner outlined in the User Guide (Michalski [4]).
   The inputs values for this test will be:
   Re: 500
   t: 75
   p: 1.0
   η_b: 1.e-2

   Output: Vorticity vector values printed to the screen.

   Test Case Derivation: This case is a comparison with the pseudo-oracle pyLBM. The output vector values of this test for pyLBM can be found in the file id3output.txt located in the OracleOutput folder. The result required for the test to pass is an average error rate between cells of the output vector (O) and (C) of less than or equal to AVERAGE_ERROR. This test satisfies R6 and R7. It also covers the high bound for bulk viscosity.

   How test will be performed:

   (a) The Von Karman Vortex Street module will be adjusted to print the vorticity vector values to the screen.
   (b) Outside of the system, the input parameter values will be written to a comma delimited text file, as outlined in the User Guide.
   (c) The file will be placed into the Input directory, under the home directory of the project.
   (d) The module for Von Karman Vortex Street will be selected to run.

(e) Upon completion of the module, the output values of the vorticity vector will be compared to the vorticity vector values from pyLBM - comparison will be done per cell. Comparisons can be done manually using Excel, or through a script.

5.1.3 Poiseuille Flow

Dynamic Testing - Manual:

These dynamic tests will compare output with output from the pseudo oracle myLBM (pyl [1]). The input oracle will have the same inputs as Lattice Boltzmann Solver for each test. The allowable error is AVERAGE_ERROR, found in Section 6.1.

1. tutorial-test-id10

   Control: Program (P) module execution using inputs (D), and expecting output (O) to match a control value (C).

   Initial State: No input file in Input directory.

   A comma delimited file, with inputs marked in the manner outlined in the User Guide (Michalski [4]).
   The inputs values for this test will be:
   t: 50
   p: 1
   η_b: 1.e-2
   η_s: 1.e-2

   Output: Pressure gradient value printed to the screen.

   Test Case Derivation: This case is a comparison with the pseudo-oracle pyLBM, which has a module for Pouiseuille Flow. The output pressure gradient value of this test is to be compared to the output value from the pseudo-oracle, -7.074e-03. The result required for the test to pass is an average error rate between cells of the output vector (O) and (C) of less than or equal to AVERAGE_ERROR.
   This test satisfies R6 and R7.

   How test will be performed:

(a) Outside of the system, the input parameter values will be written to a comma delimited text file, as outlined in the User Guide.

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

(c) The module for Poiseuille Flow will be selected to run.

(d) Upon completion of the module, the pressure gradient output value will be compared to the above output value from the pseudo-oracle.

2. low-density-test-id11

Control: Program (P) module execution using inputs (D), and expecting output (O) to match a control value (C).

Initial State: No input file in Input directory.

A comma delimited file, with inputs marked in the manner outlined in the User Guide (Michalski [4]).
The inputs values for this test will be:
t: 50
p: 0.0708
η_b: 1.e-2
η_s: 1.e-2

Output: Pressure gradient value printed to the screen.

Test Case Derivation: This case is a comparison with the pseudo-oracle pyLBM, which has a module for Pouiseuille Flow. The output pressure gradient value of this test is to be compared to the output value from the pseudo-oracle, -4.103e-03. The result required for the test to pass is an average error rate between cells of the output vector (O) and (C) of less than or equal to AVERAGE_ERROR.
This test satisfies R6 and R7.

How test will be performed:

(a) Outside of the system, the input parameter values will be written to a comma delimited text file, as outlined in the User Guide.

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

(c) The module for Poiseuille Flow will be selected to run.

(d) Upon completion of the module, the pressure gradient output value will be compared to the above output value from the pseudo-oracle.

3. high-density-test-id12

Control: Program (P) module execution using inputs (D), and expecting output (O) to match a control value (C).

Initial State: No input file in Input directory.

A comma delimited file, with inputs marked in the manner outlined in the User Guide (Michalski [4]).
The inputs values for this test will be:
t: 50
p: 13.6
η_b: 1.e-2
η_s: 1.e-2

Output: Pressure gradient value printed to the screen.

Test Case Derivation: This case is a comparison with the pseudo-oracle pyLBM, which has a module for Pouiseuille Flow. The output pressure gradient value of this test is to be compared to the output value from the pseudo-oracle, -1.522e-01. The result required for the test to pass is an average error rate between cells of the output vector (O) and (C) of less than or equal to AVERAGE_ERROR.
This test satisfies R6 and R7.

How test will be performed:

(a) Outside of the system, the input parameter values will be written to a comma delimited text file, as outlined in the User Guide.

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

(c) The module for Poiseuille Flow will be selected to run.

(d) Upon completion of the module, the pressure gradient output value will be compared to the above output value from the pseudo-oracle.

4. low-bulk-viscosity-test-id13

   Control: Program (P) module execution using inputs (D), and expecting output (O) to match a control value (C).

   Initial State: No input file in Input directory.

   A comma delimited file, with inputs marked in the manner outlined in the User Guide (Michalski [4]).
   The inputs values for this test will be:
   t: 50
   p: 1
   η_b: 1.e-4
   η_s: 1.e-2

   Output: Pressure gradient value printed to the screen.

   Test Case Derivation: This case is a comparison with the pseudo-oracle pyLBM, which has a module for Pouiseuille Flow. The output pressure gradient value of this test is to be compared to the output value from the pseudo-oracle, -1.462e-01. The result required for the test to pass is an average error rate between cells of the output vector (O) and (C) of less than or equal to AVERAGE_ERROR.
   This test satisfies R6 and R7.

   How test will be performed:

(a) Outside of the system, the input parameter values will be written to a comma delimited text file, as outlined in the User Guide.

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

(c) The module for Poiseuille Flow will be selected to run.

(d) Upon completion of the module, the pressure gradient output value will be compared to the above output value from the pseudo-oracle.

5. high-bulk-viscosity-test-id14

Control: Program (P) module execution using inputs (D), and expecting output (O) to match a control value (C).

Initial State: No input file in Input directory.

A comma delimited file, with inputs marked in the manner outlined in the User Guide (Michalski [4]).
The inputs values for this test will be:
t: 50
p: 1
η_b: 20000
η_s: 1.e-2

Output: Pressure gradient value printed to the screen.

Test Case Derivation: This case is a comparison with the pseudo-oracle pyLBM, which has a module for Pouiseuille Flow. The output pressure gradient value of this test is to be compared to the output value from the pseudo-oracle, -3.262e-03. The result required for the test to pass is an average error rate between cells of the output vector (O) and (C) of less than or equal to AVERAGE_ERROR.
This test satisfies R6 and R7.

How test will be performed:

(a) Outside of the system, the input parameter values will be written to a comma delimited text file, as outlined in the User Guide.

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

(c) The module for Poiseuille Flow will be selected to run.

(d) Upon completion of the module, the pressure gradient output value will be compared to the above output value from the pseudo-oracle.

6. low-shear-viscosity-test-id15

Control: Program (P) module execution using inputs (D), and expecting output (O) to match a control value (C).

Initial State: No input file in Input directory.

A comma delimited file, with inputs marked in the manner outlined in the User Guide (Michalski [4]).
The inputs values for this test will be:
t: 50
p: 1
η_b: 1.e-2
η_s: 1.e-3

Output: Pressure gradient value printed to the screen.

Test Case Derivation: This case is a comparison with the pseudo-oracle pyLBM, which has a module for Pouiseuille Flow. The output pressure gradient value of this test is to be compared to the output value from the pseudo-oracle, -1.604e-03. The result required for the test to pass is an average error rate between cells of the output vector (O) and (C) of less than or equal to AVERAGE_ERROR.
This test satisfies R6 and R7.

How test will be performed:

(a) Outside of the system, the input parameter values will be written to a comma delimited text file, as outlined in the User Guide.

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

(c) The module for Poiseuille Flow will be selected to run.

(d) Upon completion of the module, the pressure gradient output value will be compared to the above output value from the pseudo-oracle.

7. high-shear-viscosity-test-id16

Control: Program (P) module execution using inputs (D), and expecting output (O) to match a control value (C).

Initial State: No input file in Input directory.

A comma delimited file, with inputs marked in the manner outlined in the User Guide (Michalski [4]).
The inputs values for this test will be:
t: 50
p: 1
η_b: 1.e-2
η_s: 20000

Output: Pressure gradient value printed to the screen.

Test Case Derivation: This case is a comparison with the pseudo-oracle pyLBM, which has a module for Pouiseuille Flow. The output pressure gradient value of this test is to be compared to the output value from the pseudo-oracle, -6.868e-01. The result required for the test to pass is an average error rate between cells of the output vector (O) and (C) of less than or equal to AVERAGE_ERROR.
This test satisfies R6 and R7.

How test will be performed:

(a) Outside of the system, the input parameter values will be written to a comma delimited text file, as outlined in the User Guide.

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

(c) The module for Poiseuille Flow will be selected to run.

(d) Upon completion of the module, the pressure gradient output value will be compared to the above output value from the pseudo-oracle.

5.2 Tests for Nonfunctional Requirements

5.2.1 Correctness

System correctness is tested via the functional tests of Section 5.1 using an allowable AVERAGE_ERROR as stated in A8 of the CA (Michalski [2]).

5.2.2 Maintainability

Document Walkthrough

1. maintainability-id17

   Type: Maintainability Walkthrough

   Initial State: Maintainability of the repository has not been tested.

   Input/Condition: A production version of Lattice Boltzmann Solver has been released.

   Output/Result: A graded report describing the maintainability of the repository.

   How test will be performed:

   (a) A reviewer with a background in technical writing is selected for the review process.

   (b) The reviewer shall check the repository for the following documentation: SRS/CA, VnV Plan, MG, MIS, User Guide.

   (c) The reviewer shall mark 1 point for each of the above documents.

   (d) The review shall read through each of the above reports and provide a grade between 1 and 5 for clarity of the writing. A score of 1 represents a document that is hard to understand, and a score of 5 represents a document that is easy to understand. The user shall divide the sum of the scores for all of the reports by 5.

(e) The review shall read through each of the above reports and provide a grade between 1 and 5 for traceability within the document. A score of 1 represents no links within the report, and a score of 5 represents many links between sections of the report. The user shall then divide the sum of the scores for all of the reports by 5.

(f) The reviewer will generate a brief report with the above three values, from steps (c) to (e).

5.2.3 Performance

Multiple Instances

1. performance-id18

   Type: Concurrent Performance

   Initial State: Input file is in the appropriate directory. The file contains the required parameter values for 2 different modules - Von Karman Vortex Street and Poiseuille Flow.

   Input/Condition: A comma delimited file with the following input parameters and values:
   Re: 500
   t: 200
   p: 1.0
   η_b: 1.e-3
   η_s: 1.e-2

   Output/Result: The time taken to run two modules sequentially as well as the time taken to run the two modules concurrently. If the time taken to run the modules concurrently is less than the time taken to run the modules sequentially then the system has successfully passed this performance test.

   How test will be performed:

(a) The user shall initiate the Poiseuille Flow module in one tab of PyCharm IDE, and Van Harman Vortex Street in another tab.

(b) The user shall begin a timer and run the Poiseuille Flow module, waiting until the pressure gradient is printed to the screen to stop the timer. The user shall repeat this process five times and then average the amount of time taken to run the module.

(c) The Von Karman Vortex Street module shall be modified to print the vorticity vector to the screen.

(d) The user shall begin a timer and run the Von Karman Vortex Street module, waiting until the vorticity vector is printed to the screen to stop the timer. The user shall repeat this process five times and then average the amount of time taken to run the module.

(e) The user shall begin a timer and run the Von Karman Vortex Street module and Poiseullie Flow module simultaneously, waiting until the vorticity vector and pressure gradient are printed to the screen to stop the timer. The user shall repeat this process five times and then average the amount of time taken to run the module.

(f) The user shall compare the timed values to ensure that the time taken to run both modules simultaneously is less than the sum of the times taken to run the modules independently.

5.2.4 Portability

Operating System Portability

1. portability-id19

   Type: Portability Check

   Initial State: Have PyCharm IDE installed on three separate operating systems (can use virtual machines for this): macOS, Windows, Linux.

Input/Condition: A comma delimited file with the following input parameters and values:
Re: 500
t: 75
p: 1.0
η_b: 1.e-3

Output/Result: This case is a check of the ability of the LBM solution to run on multiple operating systems. The result required for the test to pass is a vorticity vector printed to the screen of each instance which matches, within the allowable AVERAGE_ERROR, the vorticity vector value found in the file id19output.txt located in the OracleOutput folder.

How test will be performed:

(a) Fore each operating system, the input parameter values will be written to a comma delimited text file, as outlined in the User Guide.

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

(c) The Von Karman Vortex Street module shall be modified to print the vorticity vector to the screen.

(d) The module for Von Karman Vortex Street will be selected to run

(e) Upon completion of the module, the output values of the vorticity vector will be compared to the vorticity vector values of the pseudo-oracle.

5.2.5 Robustness

System robustness is tested through the Input Bounds check in Section 5.1.

5.2.6 Usability

Usability Survey

1. usability-id20

   Type: Usability Survey

   Initial State: The LBM solver repository, the below input parameters, and an instruction to run the Von Karman Vortex Street module are provided to five users via GitHub.

   Input/Condition: A usability survey with the questions listed in Section 6.2 and a comma delimited file with the following input parameters and values:
   Re: 500
   t: 75
   p: 1.0
   η_b: 1.e-3

   Output/Result: Survey results.

   Test Case Derivation: This case is a check of the usability of the LBM solver. Respondents will be asked to rank their experience of running a module.

   How test will be performed:

   (a) Each participant will be instructed to run a Von Karman Vortex Street simulation using the provided input parameters.
   (b) The participants will attempt to run the simulation.
   (c) Upon completion of the attempt, the participants will be asked to answer the survey questions.
   (d) The questionnaire results will be tallied and averaged.

5.3 Traceability Between Test Cases and Requirements

	R1	R2	R3	R4	R5	R6	R7
id1	✓						
id2			✓				
id3						✓	✓
id4						✓	✓
id5						✓	✓
id6						✓	✓
id7						✓	✓
id8						✓	✓
id9						✓	✓
id10						✓	✓
id11						✓	✓
id12						✓	✓
id13						✓	✓
id14						✓	✓
id15						✓	✓
id16						✓	✓
id16						✓	✓

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	✓									
id13	✓									
id14	✓									
id15	✓									
id16	✓									
id17		✓								
id18			✓							
id19	✓			✓						
id20										✓

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

References

[1] pylbm. URL https://github.com/pylbm/pylbm.

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

[3] Peter Michalski. Lattice boltzmann solvers - unit vnv, . URL https://github.com/peter-michalski/LatticeBoltzmannSolvers/blob/master/docs/VnVPlan/UnitVnVPlan/UnitVnVPlan.pdf.

[4] Peter Michalski. Lattice boltzmann solvers - user guide, . URL https://github.com/peter-michalski/LatticeBoltzmannSolvers/tree/master/docs/UserGuide.

[5] Smith. Srs checklist. URL https://gitlab.cas.mcmaster.ca/smiths/cas741/blob/master/BlankProjectTemplate/docs/SRS/SRS-Checklist.pdf.

6 Appendix

6.1 Symbolic Parameters

AVERAGE_ERROR: 3%

6.2 Usability Survey Questions

Please answer the following five questions using the rubric:

1 - strongly disagree
2 - somewhat disagree
3 - neither agree nor disagree
4 - somewhat agree
5 - strongly agree

1. The formatting of the input file was easy to understand.
2. The location to place the input file was easy to find.
3. Navigating to the correct module was straightforward.
4. The User Guide of this product explained the modules well.
5. I would recommend this product.

6.3 Input Data

variability	**value**
Re	0.0001 - 50000
p	0.0708 - 13.6
η_b	0.0001 - 0.01
η_s	0.001 - 20000
t	$\mathbb{N} > 0$

Table 3: Input Data Bounds