System Verification and Validation Plan for Lattice Boltzmann Solver

Peter Michalski

November 18, 2019

1 Revision History

Date	Version	Notes
Oct. 28, 2019	1.0	Initial Document

Contents

List of Tables

List of Figures

2 Symbols, Abbreviations and Acronyms

2.1 Abbreviations and Acronyms

symbol	description
1D	One Dimensional
2D	Two Dimensional
A	Assumption
C	Pseudo-Oracle Control Value
CA	Commonality Analysis
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
NAN	Not A Number
NFR	Non-Functional Requirement
O	Program Output
OTS	Off The Shelf
P	Program
R	Functional Requirement
SRS	Software Requirement Specification
V	Set of All Inputs
VnV	Verification and Validation

2.2 Symbols

symbol	description
D	Signifies the Dimension Component of Lattice Model
D_l	Length of the Domain
D_w	Width of the Domain
$\mathrm{e_{fac}}$	Velocity of Scheme
$\mathrm{e_{max}}$	Max Velocity in the Middle of Channel
$\mathrm{e_{sch}}$	Velocity Slowdown Factor
E_r	Relative Error
η_b	Bulk Viscosity
η_s	Shear Viscosity
$\mathbb{N}$	natural numbers
Q	Signifies Number of Velocity Directions of Lattice Model
R	Radius of Cylinder
Re	Reynolds Number
ρ	Density
S	Spatial Step - Number of Spatial Subsections
t	Time
X_{max}	Highest X-Axis Interval of Boundary
X_{min}	Lowest X-Axis Interval of Boundary
Y_{max}	Highest Y-Axis Interval of Boundary
Y_{min}	Lowest Y-Axis Interval of Boundary

This document covers the system Verification and Validation (VnV) of Lattice Boltzmann Solver. Functional and non-functional requirements, as found in Section 3.2 and retrieved from the Commonality Analysis (CA) (Michalski [3]), 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

[There should generally be text between section headings. In this case a "roadmap" of the contents of this section would be helpful. —SS]

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 Solver can 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 CA titled Lattice Boltzmann Solvers (Michalski [3]), have been met. [Copying and pasting your requirements here from your CA is a maintenance problem. It will be difficult to keep the two documents in sync. You can just reference the document where the requirements are, without repeating them. You can even use cross-referencing to reference labels in another document. That is why the original Blank Project Template has make files, so that running make will update the references between documents. —SS]
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. [This requirement has to be more specific. What are the velocity and direction parameters? I feel like you summarized them in your CA. —SS]

R3: The system shall verify that the inputs fall within the allowable parameters of variation. [Where is this documented? There should be a pointer to this information. Again, I think this is in your CA, which is another argument for not reproducing the requirements here. —SS]

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. [NFRs should not mention modules. They should be about the software as a whole. You might want to specify correctness as the comparison between the OTS solution and your solution running the test cases defined in this document. —SS]

2. Maintainability: The system shall be documented with a CA, VnV plan, MG (Module Guide), MIS (Module Interface Specification), and User Guide. [This is how you intend to achieve maintainability, not how maintainability is defined. —SS]

3. Performance: The system shall be able to run modules faster than a sample OTS implementation.

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 shall behave reasonably in circumstances that were not anticipated in the requirements specification [2].

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

9. Understandability: New users must easily understand which LBM models are available in Lattice Boltzmann Solver.

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

3.3 Relevant Documentation

The CA of Lattice Boltzmann Solver can be found in (Michalski [3]). [Although they aren't written yet, you should fake this as if the design documentation is written. The documentation is also relevant for this problem. —SS]

4 Plan

4.1 Verification and Validation Team

This VnV plan will be conducted by Peter Michalski and several classmates. [Be specific and name the relevant classmates here. —SS]

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. [Are you going to be more specific? Is this task directed feedback, or just "read the document" and tell me what you think? Who specifically is going to review the document? —SS]

2. Initial Review: The document shall be manually reviewed by the author using the SRS checklist upon its initial creation, as found in the CAS741 GitLab repository (Smith [6]). [Good! —SS]

3. Second Review: The document shall be manually reviewed by the author using the SRS checklist after VnV completion, as found in the CAS741 GitLab repository (Smith [6]). [Good! —SS]

4. Final Review: The document shall be manually reviewed by the author using the SRS checklist after MG and MIS development, as found in the CAS741 repository (Smith [6]). [Great! —SS]

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 immediately after MIS document version 1.0 is developed. A rubber duck debugging method will be followed. Any defects shall be immediately fixed. This plan is described in Section 5.1 of the document Unit Verification and Validation Plan for Lattice Boltzmann Solver (Michalski [4]).

2. Code Walkthrough - Peer: Module unit code shall be inspected for functional errors by Ao Dong after MIS document version 1.0 is submitted on GitHub. 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 [4]).

3. Dynamic Unit Tests - Author: 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 [4]).

4. System Tests: System tests will be carried out as listed in Section 5. The author shall conduct the tests of the functional requirements. Tests of the NFRs shall be conducted by those listed in the outline of each test.

4.5 Software Validation Plan

Lattice Boltzmann Solver shall be validated by ensuring that all functional requirements, as listed in Section 3.2, are met. The file names containing the pseudo-oracle output data that the system output is tested against can be found in the specific test sections below.

[You do not actually have a validation step. You should explain in this section that validation is related to comparing the outputs of models to experimental values. Your software doesn't care whether the models agree with reality, only that they the solutions are correct mathematically. —SS]

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 R1, R2, R3, R4, R5, R6, and R7, and will test out the input vales at their allowable edges.

5.1.1 Input

Input Reading

1. input-reading-id1A

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

   Initial State: No input file in Input directory. [If the initial state is no input file, then wouldn't this test fail to generate output? I would think you would get an exception if the file does not exist? —SS]

   Input: A comma delimited text file, with inputs marked in the manner outlined in the User Guide (Michalski [5]).

   The input values for this test will be:

   Re: 500
   t: 75
   ρ: 1.0
   η_b: 1.e-3
   $X_{min} = 0.$
   $X_{max} = 3.$
   $Y_{min} = 0.$
   $Y_{max} = 1.$
   $R = 0.05$
   $\mathrm{e_{sch}} = 1.$
   $\mathrm{e_{fac}} = 20$
   $\mathrm{e_{max}} = 0.1$
   $S = 64$
   $\mathrm{D} = 2$
   $\mathrm{Q} = 9$
   $D_l = 2$

$D_w = 1$

Output: All inputs printed to the screen as output (O).
Outputs printed to the screen:
Re: 500
t: 75
ρ: 1.0
η_b: 1.e-3
$X_{min} = 0.$
$X_{max} = 3.$
$Y_{min} = 0.$
$Y_{max} = 1.$
$R = 0.05$
$\mathrm{e_{sch}} = 1.$
$\mathrm{e_{fac}} = 20$
$\mathrm{e_{max}} = 0.1$
$S = 64$
D = 2
Q = 9
$D_l = 2$
$D_w = 1$

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

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 titled input.txt, as outlined in the User Guide.

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

(c) Lattice Boltzmann Solver will be run.

(d) If successful, The system will output the input parameters to the screen.

2. input-reading-id1B

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

   Initial State: No input file in Input directory.

   Input: A comma delimited text file with the following input: ccQseHfFspBLAYZjCGZtJYMAdeAc

   Output: An error message of "cannot read file" should be shown.

   Test Case Derivation: P(V) is correct if P:V -> Error Message

   The result required for the test to pass is the successful printing of the error message. This test satisfies R1.

   How test will be performed:

   (a) Outside of the system, the input parameter value will be written to a comma delimited text file titled input.txt, as outlined in the User Guide.
   (b) The file will be placed into the Input directory, under the home directory of the project.
   (c) Lattice Boltzmann Solver will be run.
   (d) If successful, the system will output an error message to the screen.

Input Bounds

1. input-bounds-id2A

   Control: Program (P) testing of high input values (V) 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 [5]).
The input values for this test will be:
Re: 500000
t: 0
ρ: 15
η_b: 100
η_s: 30000
$X_{min} = 0.$
$X_{max} = 3.$
$Y_{min} = 0.$
$Y_{max} = 1.$
$R = 0.05$
$\mathrm{e_{sch}} = 1.$
$\mathrm{e_{fac}} = 20$
$\mathrm{e_{max}} = 0.1$
$S = 64$
$\mathrm{D} = 2$
$\mathrm{Q} = 9$
$D_l = 2$
$D_w = 1$

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. In this test that will include Re, t, ρ, η_b, and η_s.

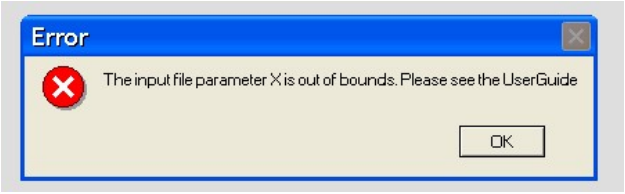

Figure 1: Input out of Bounds

Test Case Derivation: I (individual inputs) are an element of V. P(V)

is incorrect if P:V 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 titled input.txt, as outlined in the User Guide (Michalski [5]).

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

(c) Lattice Boltzmann Solver will be run.

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

2. input-bounds-id2B

Control: Program (P) testing of low input values (V) 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 [5]).
The input values for this test will be:
Re: -500000
t: 0
ρ: -15
η_b: -100
η_s: -30000
$X_{min} = 0.$
$X_{max} = 3.$
$Y_{min} = 0.$
$Y_{max} = 1.$
$R = 0.05$
$\mathrm{e_{sch}} = 1.$
$\mathrm{e_{fac}} = 20$

$e_{max} = 0.1$
$S = 64$
D = 2
Q = 9
$D_l = 2$
$D_w = 1$

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. In this test that will include Re, t, ρ, η_b, and η_s.

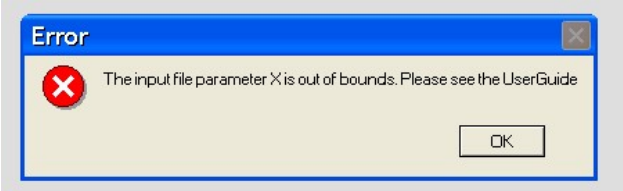

Figure 2: Input out of Bounds

Test Case Derivation: I (individual inputs) are an element of V. P(V) is incorrect if P:V 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 titled input.txt, as outlined in the User Guide (Michalski [5]).

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

(c) Lattice Boltzmann Solver will be run.

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

[The previous few tests are a bit repetitive. I like the idea of testing that the input checks are working, but you could likely summarize the different

tests cases in a table. Give the outline as above for one test case, but then substitute in a reference to the table and explain how the table provides x number of additional tests. You can still give each of the tests an identifying label, but the label would be in the table. —SS]

5.1.2 Von Karman Vortex Street

Dynamic Testing - Manual:

These dynamic tests will compare Lattice Boltzmann Solver 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. [Rather than an allowed error, we might be better off simply summarizing the calculated error. Even if there is an allowed error, a report of the actual error values is more useful information than simply PASS/FAIL. —SS]

1. tutorial-test-id3

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

   Initial State: No input file in Input directory. [Again, this seems like an odd initial state. —SS]

   Input: A comma delimited file, with inputs marked in the manner outlined in the User Guide (Michalski [5]).
   The input values for this test will be:
   Re: 500
   t: 75
   ρ: 1.0
   η_b: 1.e-3
   $X_{min} = 0.$
   $X_{max} = 3.$
   $Y_{min} = 0.$
   $Y_{max} = 1.$
   $R = 0.05$
   $\mathrm{e_{sch}} = 1.$
   $\mathrm{e_{fac}} = 20$
   $S = 64$
   $\mathrm{D} = 2$
   $\mathrm{Q} = 9$

[It is great that you will be using a pseudo-oracle. However, it would be nice to explain what the test inputs mean. How would a scientist explain the test you are running? What is the physics of the test? I haven't checked, but I would guess that your pyl reference would explain the test cases more than just listing numbers? —SS]

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 (C) 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. [All of your tests that compare to average error should instead be written to output the error values. I would also like to see your specific relative error calculation formulae for each test case where you mention error. —SS]

How test will be performed:

(a) The Von Karman Vortex Street module shall be modified by the author to print the vorticity vector as output.

(b) Outside of the system, the input parameter values will be written to a comma delimited text file titled input.txt, 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.

[This shouldn't really be a manual test. Everything described above can be done algorithmically. You can automate this test. You will need to go further to define the mathematical equation for error, but you would still need to do this for the manual version. As part of your development process, I could certainly see you doing the test manually

the first time, to help understand how the pieces fit together. Once you understand though, the target should be automation. This same comment about changing your manual tests to automated tests applies for your other proposed tests as well. —SS]

2. Reynolds-Rel-Error-test-id4

   Control: Program (P) module execution using inputs (V) with varying Re parameter (I), and observing variance of the relative error (E_r) of the output (O) when compared to a control value (C). [If all of the tests are the same except for the value of Re, then you could again use a table to summarize the different test cases. —SS]

   Initial State: No input file in Input directory.

   Input (first iteration of test): A comma delimited file, with inputs marked in the manner outlined in the User Guide (Michalski [5]).
   The input values for this test will be:
   Re: 500
   t: 75
   ρ: 1.0
   η_b: 1.e-3
   $X_{min} = 0.$
   $X_{max} = 3.$
   $Y_{min} = 0.$
   $Y_{max} = 1.$
   $R = 0.05$
   $\mathrm{e_{sch}} = 1.$
   $\mathrm{e_{fac}} = 20$
   $S = 64$
   $\mathrm{D} = 2$
   $\mathrm{Q} = 9$

   Input (second iteration of test): A comma delimited file, with inputs marked in the manner outlined in the User Guide (Michalski [5]).
   The input values for this test will be:
   Re: 1500
   t: 75

ρ: 1.0
η_b: 1.e-3
$X_{min} = 0.$
$X_{max} = 3.$
$Y_{min} = 0.$
$Y_{max} = 1.$
$R = 0.05$
$\mathrm{e_{sch}} = 1.$
$\mathrm{e_{fac}} = 20$
$S = 64$
D = 2
Q = 9

Input (third iteration of test): A comma delimited file, with inputs marked in the manner outlined in the User Guide (Michalski [5]).
The input values for this test will be:
Re: 2500
t: 75
ρ: 1.0
η_b: 1.e-3
$X_{min} = 0.$
$X_{max} = 3.$
$Y_{min} = 0.$
$Y_{max} = 1.$
$R = 0.05$
$\mathrm{e_{sch}} = 1.$
$\mathrm{e_{fac}} = 20$
$S = 64$
D = 2
Q = 9

Output: In each iteration we will have vorticity vector values printed to the screen.

Test Case Derivation: This case is a comparison with the pseudo-oracle pyLBM. The C values of this test for pyLBM can be found in the files id4_1_output.txt, id4_2_output.txt and id4_3_output.txt, located in the

OracleOutput folder. The files hold the C values for iterations 1, 2, and 3 respectively. Each iteration will calculate the relative error of O to its respective C counterpart. The E_r values of the iterations will be compared with the intent to discover how changes in Re affect correctness. This may be helpful in improving correctness of the implementation.

How test will be performed:

(a) The Von Karman Vortex Street module shall be modified by the author to print the vorticity vector as output.

(b) Outside of the system, the input parameter values will be written to a comma delimited text file titled input.txt, 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. A relative error value will be calculated.

(f) Steps (a) - (e) will be repeated for each test iteration.

(g) The E_r of each iteration will be compared.

3. laminar-test-id5

Control: Program (P) module execution using inputs (V), 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 [5]).
The input values for this test will be:
Re: 0.0001

t: 75
ρ: 1.0
η_b: 1.e-3
$X_{min} = 0.$
$X_{max} = 3.$
$Y_{min} = 0.$
$Y_{max} = 1.$
$R = 0.05$
$\mathrm{e_{sch}} = 1.$
$\mathrm{e_{fac}} = 20$
$S = 64$
D = 2
Q = 9

Output: Vorticity vector values printed to the screen.

Test Case Derivation: This case is a comparison with the pseudo-oracle pyLBM. The C values of this test for pyLBM can be found in the file id5output.txt located in the OracleOutput folder. The result required for the test to pass is an average error rate between cells of O and C of less than or equal to AVERAGE_ERROR.

This test covers a very small Reynolds number representing laminar flow.

How test will be performed:

(a) The Von Karman Vortex Street module shall be modified by the author to print the vorticity vector as output.

(b) Outside of the system, the input parameter values will be written to a comma delimited text file titled input.txt, 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. turbulent-test-id6

Control: Program (P) module execution using inputs (V), 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 [5]).
The input values for this test will be:
Re: 50000
t: 75
ρ: 1.0
η_b: 1.e-3
$X_{min} = 0.$
$X_{max} = 3.$
$Y_{min} = 0.$
$Y_{max} = 1.$
$R = 0.05$
$\mathrm{e_{sch}} = 1.$
$\mathrm{e_{fac}} = 20$
$S = 64$
D = 2
Q = 9

Output: Vorticity vector values printed to the screen.

Test Case Derivation: This case is a comparison with the pseudo-oracle pyLBM. The C values of this test for pyLBM can be found in the file id6output.txt located in the OracleOutput folder. The result required for the test to pass is an average error rate between O and C of less than or equal to AVERAGE_ERROR.

This test covers a very large Reynolds number representing very turbulent flow.

How test will be performed:

(a) The Von Karman Vortex Street module shall be modified by the author to print the vorticity vector as output.

(b) Outside of the system, the input parameter values will be written to a comma delimited text file titled input.txt, 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. low-density-test-id7

Control: Program (P) module execution using inputs (V), 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 [5]).
The input values for this test will be:
Re: 500
t: 75
ρ: 7.08e-2
η_b: 1.e-3
$X_{min} = 0.$
$X_{max} = 3.$
$Y_{min} = 0.$

$Y_{max} = 1.$
$R = 0.05$
$\mathrm{e_{sch}} = 1.$
$\mathrm{e_{fac}} = 20$
$S = 64$
D = 2
Q = 9

Output: Vorticity vector values printed to the screen.

Test Case Derivation: This case is a comparison with the pseudo-oracle pyLBM. The C values of this test for pyLBM can be found in the file id7output.txt located in the OracleOutput folder. The result required for the test to pass is an average error rate between O and C of less than or equal to AVERAGE_ERROR.

This test covers the low bound for density using a density of liquid hydrogen.

How test will be performed:

(a) The Von Karman Vortex Street module shall be modified by the author to print the vorticity vector as output.

(b) Outside of the system, the input parameter values will be written to a comma delimited text file titled input.txt, 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. high-density-test-id8

Control: Program (P) module execution using inputs (V), 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 [5]).
The input values for this test will be:
Re: 500
t: 75
ρ: 13.6
η_b: 1.e-3
$X_{min} = 0.$
$X_{max} = 3.$
$Y_{min} = 0.$
$Y_{max} = 1.$
$R = 0.05$
$\mathrm{e_{sch}} = 1.$
$\mathrm{e_{fac}} = 20$
$S = 64$
D = 2
Q = 9

Output: Vorticity vector values printed to the screen.

Test Case Derivation: This case is a comparison with the pseudo-oracle pyLBM. The C values of this test for pyLBM can be found in the file id8output.txt located in the OracleOutput folder. The result required for the test to pass is an average error rate between O and C of less than or equal to AVERAGE_ERROR.

This test 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 shall be modified by the author to print the vorticity vector as output.

(b) Outside of the system, the input parameter values will be written to a comma delimited text file titled input.txt, 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. low-bulk-viscosity-test-id9

Control: Program (P) module execution using inputs (V), 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 [5]).
The input values for this test will be:

Re: 500
t: 75
ρ: 1.0
η_b: 1.e-4
$X_{min} = 0.$
$X_{max} = 3.$
$Y_{min} = 0.$
$Y_{max} = 1.$
$R = 0.05$
$\mathrm{e_{sch}} = 1.$
$\mathrm{e_{fac}} = 20$

$S = 64$
D = 2
Q = 9

Output: Vorticity vector values printed to the screen.

Test Case Derivation: This case is a comparison with the pseudo-oracle pyLBM. The C values of this test for pyLBM can be found in the file id9output.txt located in the OracleOutput folder. The result required for the test to pass is an average error rate between O and C of less than or equal to AVERAGE_ERROR.
This test covers the low bound for bulk viscosity.

How test will be performed:

(a) The Von Karman Vortex Street module shall be modified by the author to print the vorticity vector as output.

(b) Outside of the system, the input parameter values will be written to a comma delimited text file titled input.txt, 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.

8. high-bulk-viscosity-test-id10

Control: Program (P) module execution using inputs (V), 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 [5]).
The input values for this test will be:
Re: 500
t: 75
ρ: 1.0
η_b: 1.e-2
$X_{min} = 0.$
$X_{max} = 3.$
$Y_{min} = 0.$
$Y_{max} = 1.$
$R = 0.05$
$\mathrm{e_{sch}} = 1.$
$\mathrm{e_{fac}} = 20$
$S = 64$
D = 2
Q = 9

Output: Vorticity vector values printed to the screen.

Test Case Derivation: This case is a comparison with the pseudo-oracle pyLBM. The C values of this test for pyLBM can be found in the file id10output.txt located in the OracleOutput folder. The result required for the test to pass is an average error rate between O and C of less than or equal to AVERAGE_ERROR.

This test covers the high bound for bulk viscosity.

How test will be performed:

(a) The Von Karman Vortex Street module shall be modified by the author to print the vorticity vector as output.

(b) Outside of the system, the input parameter values will be written to a comma delimited text file titled input.txt, 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 Lattice Boltzmann Solver 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-id11

   Control: Program (P) module execution using inputs (V), 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 [5]).
   The input values for this test will be:
   t: 50
   ρ: 1
   η_b: 1.e-2
   η_s: 1.e-2
   $S = 16$
   $D_l = 2$
   $D_w = 1$
   $\mathrm{e_{sch}} = 1.$
   $\mathrm{e_{max}} = 0.1$
   $X_{min} = 0.$

   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 Poiseuille Flow. The output pressure gradient value of this test is to be compared to the C value from the pseudo-oracle, -7.074e-03. The result required for the test to pass is

an average error rate between O and C of less than or equal to AVERAGE_ERROR.

How test will be performed:

(a) Outside of the system, the input parameter values will be written to a comma delimited text file titled input.txt, 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-id12

Control: Program (P) module execution using inputs (V), 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 [5]).
The input values for this test will be:

t: 50
ρ: 0.0708
η_b: 1.e-2
η_s: 1.e-2
$S = 16$
$D_l = 2$
$D_w = 1$
$\mathrm{e_{sch}} = 1.$
$\mathrm{e_{max}} = 0.1$

$X_{min} = 0$.

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 Poiseuille Flow. The output pressure gradient value of this test is to be compared to the C value from the pseudo-oracle, -4.103e-03. The result required for the test to pass is an average error rate between O and C of less than or equal to AVERAGE_ERROR.

How test will be performed:

(a) Outside of the system, the input parameter values will be written to a comma delimited text file titled input.txt, 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-id13

Control: Program (P) module execution using inputs (V), 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 [5]).
The input values for this test will be:
t: 50
ρ: 13.6

η_b: 1.e-2
η_s: 1.e-2
$S = 16$
$D_l = 2$
$D_w = 1$
$\mathrm{e_{sch}} = 1.$
$\mathrm{e_{max}} = 0.1$
$X_{min} = 0.$

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 Poiseuille Flow. The output pressure gradient value of this test is to be compared to the C value from the pseudo-oracle, -1.522e-01. The result required for the test to pass is an average error rate between O and C of less than or equal to AVERAGE_ERROR.

How test will be performed:

(a) Outside of the system, the input parameter values will be written to a comma delimited text file titled input.txt, 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-id14

Control: Program (P) module execution using inputs (V), 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 [5]).
The input values for this test will be:
t: 50
ρ: 1
η_b: 1.e-4
η_s: 1.e-2
$S = 16$
$D_l = 2$
$D_w = 1$
$\mathrm{e_{sch}} = 1.$
$\mathrm{e_{max}} = 0.1$
$X_{min} = 0.$

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 Poiseuille Flow. The output pressure gradient value of this test is to be compared to the C output from the pseudo-oracle, -1.462e-01. The result required for the test to pass is an average error rate between O and C of less than or equal to AVERAGE_ERROR.

How test will be performed:

(a) Outside of the system, the input parameter values will be written to a comma delimited text file titled input.txt, 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-id15

   Control: Program (P) module execution using inputs (V), 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 [5]).
   The input values for this test will be:
   t: 50
   ρ: 1
   η_b: 20000
   η_s: 1.e-2
   $S = 16$
   $D_l = 2$
   $D_w = 1$
   $\mathrm{e_{sch}} = 1.$
   $\mathrm{e_{max}} = 0.1$
   $X_{min} = 0.$

   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 Poiseuille Flow. The output pressure gradient value of this test is to be compared to the C value from the pseudo-oracle, -3.262e-03. The result required for the test to pass is an average error rate between O and C of less than or equal to AVERAGE_ERROR.

   How test will be performed:

   (a) Outside of the system, the input parameter values will be written to a comma delimited text file titled input.txt, 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-id16

Control: Program (P) module execution using inputs (V), 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 [5]).
The input values for this test will be:
t: 50
ρ: 1
η_b: 1.e-2
η_s: 1.e-3
$S = 16$
$D_l = 2$
$D_w = 1$
$\mathrm{e_{sch}} = 1.$
$\mathrm{e_{max}} = 0.1$
$X_{min} = 0.$

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 Poiseuille Flow. The output pressure gradient value of this test is to be compared to the C value from the pseudo-oracle, -1.604e-03. The result required for the test to pass is an average error rate between O and C of less than or equal to AVER-

AGE_ERROR.

How test will be performed:

(a) Outside of the system, the input parameter values will be written to a comma delimited text file titled input.txt, 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-id17

Control: Program (P) module execution using inputs (V), 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 [5]).
The input values for this test will be:
t: 50
ρ: 1
η_b: 1.e-2
η_s: 20000
$S = 16$
$D_l = 2$
$D_w = 1$
$\mathrm{e_{sch}} = 1.$
$\mathrm{e_{max}} = 0.1$
$X_{min} = 0.$

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 Poiseuille Flow. The output pressure gradient value of this test is to be compared to the C value from the pseudo-oracle, -6.868e-01. The result required for the test to pass is an average error rate between O and C of less than or equal to AVERAGE_ERROR.

How test will be performed:

(a) Outside of the system, the input parameter values will be written to a comma delimited text file titled input.txt, 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 Section 6.1.

5.2.2 Maintainability

Document Walkthrough

1. maintainability-test-id18

   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.

   Test Case Derivation: This test will determine if the system documentation allows for easy maintainability by checking how much documentation is available, as well as reviewing its clarity and traceability. A score of higher than 3 in each of the tested categories will indicate that the system documentation is maintainable.

   How test will be performed:

   (a) Ao Dong shall check the repository for the following documentation: SRS/CA, VnV Plan, MG, MIS, User Guide.

   (b) Ao Dong shall mark 1 point for each of the above documents.

   (c) Ao Dong shall read through each of the above documents 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.

(d) Ao Dong shall read through each of the above documents 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.

(e) Ao Dong will generate a brief report with the above three values, from steps (c) to (e).

[This maintainability measure is quite subjective, but I still like what you are trying to do here. The absolute value that Ao comes up with will not be particularly meaningful, but the process may uncover some areas of concern, or build confidence by the lack of areas of concern. I think we will need better measures for your MEng project, but this okay as a starting point. —SS]

5.2.3 Performance

Running Time

1. performance-test-id19

   Type: Running Time

   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
   ρ: 1.0
   η_b: 1.e-3
   $X_{min} = 0.$
   $X_{max} = 3.$

$Y_{min} = 0.$
$Y_{max} = 1.$
$R = 0.05$
$\mathrm{e_{sch}} = 1.$
$\mathrm{e_{fac}} = 20$
$S = 64$
D = 2
Q = 9

Output/Result: A timed comparison of running each of the two system modules, Von Karman Vortex Street and Poiseuille Flow, against the psuedo-oracle pyLBM using pyCharm IDE. The percentage difference in running time between Lattice Boltzmann Solver and the pseudo-oracle will noted and analyzed.

How test will be performed:

(a) Outside of the system, the input parameter values will be written, by the author, to a comma delimited text file titled input.txt, 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 author shall first start a timer and then select the Poiseuille Flow module to run.

(d) Upon completion of the module the author shall note the module running time.

(e) The author shall repeat steps (c) to (d) for the Von Karman Vortex Street module.

(f) The author shall then run each of the problems, Poiseuille Flow and Von Karman Vortex Street, using the pseudo-oracle pyLBM using pyCharm IDE, making sure to time the running of each problem.

(g) The author shall compare the running time of each of the two system modules, Von Karman Vortex Street and Poiseuille Flow, against the pseudo-oracle.

[This tests sounds interesting. I like that you did not set a target for performance. You are just going to report the percentage difference. The timers are in the code, right? It reads as if they are going to be timed with an external stop-watch, but I'm assuming starting the timer is something that is done in the testing code. This test should also be automated. —SS]

5.2.4 Portability

Operating System Portability

1. portability-test-id20

   Type: Portability Check

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

   Input/Condition: A comma delimited file with the following input parameters and values:
   Re: 500
   t: 75
   ρ: 1.0
   η_b: 1.e-3
   $X_{min} = 0.$
   $X_{max} = 3.$
   $Y_{min} = 0.$
   $Y_{max} = 1.$
   $R = 0.05$
   $\mathrm{e_{sch}} = 1.$
   $\mathrm{e_{fac}} = 20$
   $S = 64$
   $\mathrm{D} = 2$
   $\mathrm{Q} = 9$

   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 id20output.txt located in the OracleOutput folder.

How test will be performed:

(a) The input parameter values will be written, by the author, to a comma delimited text file titled input.txt, as outlined in the User Guide.

(b) The file will be placed into the Input directory, under the home directory of the project, in each of the operating system instances.

(c) The Von Karman Vortex Street module shall be modified by the author to print the vorticity vector as output.

(d) The author shall run the Von Karman Vortex Street module.

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

[For portability I suggest running all of your automated tests on all three platforms, and reporting the results. This is part of why we want automation, and why we report the calculated relative error, rather than just PASS/FAIL. —SS]

5.2.5 Robustness

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

5.2.6 Usability

Usability Survey

1. usability-test-id21

   Type: Usability Survey

Initial State: The Lattice Boltzmann Solver repository, the below input parameters, and an instruction to run the Von Karman Vortex Street module are provided to three anonymous 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
ρ: 1.0
η_b: 1.e-3
$X_{min} = 0.$
$X_{max} = 3.$
$Y_{min} = 0.$
$Y_{max} = 1.$
$R = 0.05$
$\mathrm{e_{sch}} = 1.$
$\mathrm{e_{fac}} = 20$
$S = 64$
D = 2
Q = 9

Output/Result: Survey results.

Test Case Derivation: This case is a check of the usability of Lattice Boltzmann Solver. Respondents will be asked to rank their experience of running a module. A final average grade of 3 will indicate that the users found the system to have average usability. The higher the score, the better the perception of usability.

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. [Where is the survey? You should point to your appendix. —SS]

(d) The questionnaire results will be tallied and averaged.

5.3 Traceability Between Test Cases and Requirements

	R1	R2	R3	R4	R5	R6	R7
id1A	✓						
id1B	✓						
id2A			✓				
id2B			✓				
id3	✓	✓	✓	✓	✓	✓	✓
id4	✓	✓	✓	✓	✓	✓	✓
id5	✓	✓	✓	✓	✓	✓	✓
id6	✓	✓	✓	✓	✓	✓	✓
id7	✓	✓	✓	✓	✓	✓	✓
id8	✓	✓	✓	✓	✓	✓	✓
id9	✓	✓	✓	✓	✓	✓	✓
id10	✓	✓	✓	✓	✓	✓	✓
id11	✓	✓	✓	✓	✓	✓	✓
id12	✓	✓	✓	✓	✓	✓	✓
id13	✓	✓	✓	✓	✓	✓	✓
id14	✓	✓	✓	✓	✓	✓	✓
id15	✓	✓	✓	✓	✓	✓	✓
id16	✓	✓	✓	✓	✓	✓	✓
id17	✓	✓	✓	✓	✓	✓	✓
id18							
id19	✓	✓	✓	✓	✓	✓	✓
id20	✓	✓	✓	✓	✓	✓	✓
id21	✓	✓	✓	✓	✓	✓	✓

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
id1A										
id1B										
id2A							✓			
id2B							✓			
id3	✓									
id4										
id5	✓									
id6	✓									
id7	✓									
id8	✓									
id9	✓									
id10	✓									
id11	✓									
id12	✓									
id13	✓									
id14	✓									
id15	✓									
id16	✓									
id17	✓									
id18		✓								
id19			✓							
id20	✓			✓						
id21										✓

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

References

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

[2] Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of software engineering*. Prentice Hall PTR, 1991.

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

[4] Peter Michalski. Lattice Boltzmann Solvers - Unit VnV, . URL https://github.com/peter-michalski/LatticeBoltzmannSolvers/blob/master/docs/VnVPlan/UnitVnVPlan/UnitVnVPlan.pdf.

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

[6] 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%
This error allowance will be used for both scalar and vector outputs, adjusted accordingly.

[I like that you used a symbolic constant, although as I mentioned above, I think the goal is to describe the error, rather than *a priori* specify the error bounds. —SS]

6.2 Usability Survey Questions

Using the following rubric please rate the five statements found below:
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.

[Good start on the survey questions. —SS]

6.3 Input Data

variability	value
Re	0.0001 - 50000
ρ	0.0708 - 13.6
η_b	0.0001 - 0.01
η_s	0.001 - 20000
t	$\mathbb{N}$
X_{min}	0.
X_{max}	3.
Y_{min}	0.
Y_{max}	1.
R	0.05
$\mathrm{e_{sch}}$	1.
$\mathrm{e_{fac}}$	20
$\mathrm{e_{max}}$	0.1
S	64 or 16 (module specific)
D	2
Q	9
D_l	2
D_w	1

Table 3: Input Data Bounds

The upper and lower bounds of Re, ρ, η_b and η_s are derived from the allowable input range of the pseudo-oracle pyLBM. Input values outside of these bounds are known to sometimes return NAN module results. t can be any $\mathbb{N}$. Fixed value variables of tests in this testing report are: X_{min}, X_{max}, Y_{min}, Y_{max}, R, D_l, D_w, $\mathrm{e_{sch}}$, $\mathrm{e_{fac}}$, $\mathrm{e_{max}}$, S (module specific), D, and Q.