Test Report: Lattice Boltzmann Solver

Peter Michalski

December 15, 2019

1 Revision History

Date	Version	Notes
Dec. 15	1.0	Initial Document

2 Symbols, Abbreviations and Acronyms

Please see Section 2.2 and Section 2.3 of the Commonality Analysis (Michalski (a)).

Contents

List of Tables

List of Figures

This document reports the results of the tests found in the Unit VnV Plan (Michalski (c)).

3 Functional Requirements Evaluation

Functional requirements are evaluated in unit tests of id1 to id10. The results of these tests can be found in Section 6. Traceability of the tests of this document to functional requirements is found in Table 1 of Section 9.

4 Nonfunctional Requirements Evaluation

NFRs are evaluated in unit tests of id11 to id12. The results of these tests can be found directly below. Traceability of the tests of this document to NFRs is found in Table 2 of Section 9.

4.1 Performance

Unit test id11 (performance-id11) found in Section 5.2.1 of the Unit VnV Plan (Michalski (c) gathers information about the performance of the simulation section of M2SystemControl as the Reynolds Number input is increased.

As we can see from Figure 1 below, the computational time of the program is not affected as the Reynolds Number parameter is increased within the acceptable range.

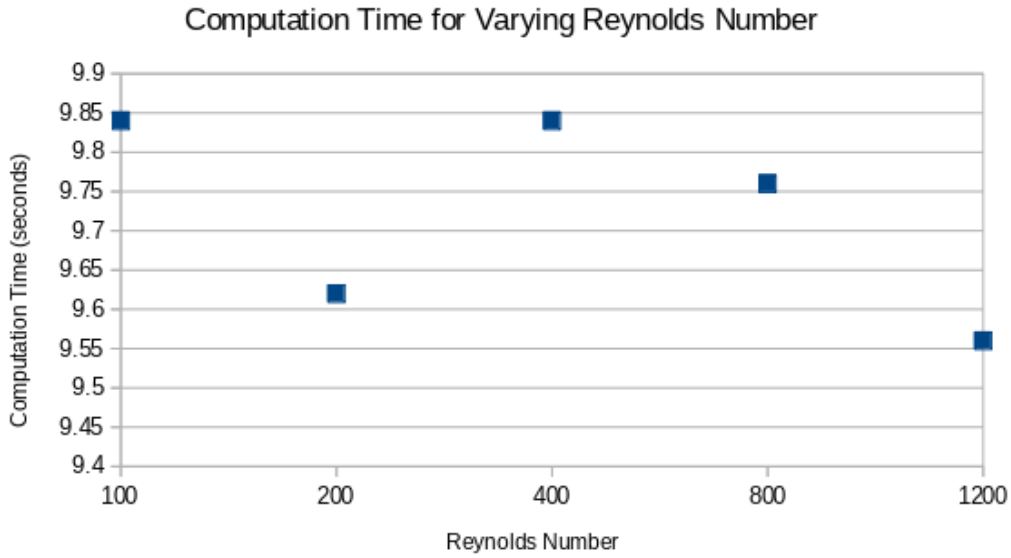

Figure 1: Computation Time for Varying Reynolds Number

4.2 Scalability

Unit test id12 (scalability-id12) found in Section 5.2.2 of the Unit VnV Plan (Michalski (c) verifies if the M9Lattice module can be modified to return additional velocity weights of differing LBM parameters.

The system returned an output of: "[0.6666666666666667, 0.1666666666666667, 0.1666666666666667]". These values are the velocity weights for a problem that has 1 dimension and 3 velocity directions, verifying that M9Lattice is scalable for LBM problems of varying dimensions and velocity directions.

5 Comparison to Existing Implementation

The first stage of implementation will not incorporate the M9Lattice module since the library that will be used at this stage will incorporate the functions of that module in the M5LBMControl module. Thus, the test in Section 4.2 is not relative to the current stage of the system, but will be relevant in future development.

6 Unit Testing

6.1 M2SystemControl

6.1.1 problem-boundary-data-id1

Unit test id1 found in Section 5.1.1 of the Unit VnV Plan (Michalski (c) verifies if the M8Problem module and M10Boundary module return correct and correctly formatted parameters.

The md5sum hash value of the test file 'problem-boundary-data-id1.txt' is 0b332c9b4fc94adbb9c3f2e9a078fdb7. The hash value of the test output result is 5e518335d4054b401b83e699446a31c7.

Since the test resulted in a negative match a closer inspection of the two files was conducted and it was found that the returned data structure contains what appears to be a hashed value of either time or a location in the system that is used purely for identification purposes. The final output image of the system appears to be identical, however further testing may be required to conclusively confirm the hypothesis regarding the assumed identification hash found in the files.

6.1.2 simulation-id2

Unit test id2 found in Section 5.1.1 of the Unit VnV Plan (Michalski (c) verifies if the solution returned in the section of code representing M5LBMControl module is correct.

The md5sum hash value of the test file 'simulation-id2.txt' is 3e4fe44bbc9278d6536366a57ec8f467. The hash value of the test output result is 3e4fe44bbc9278d6536366a57ec8f467.

This confirms that the output of the section of code representing M5LBMControl module is correct. It also confirms the assumption in test id1 above that the differing hash value found within the test file 'problem-boundary-data-id1.txt' and the output of test id1 is not used in the actual simulation, as this test used the above section of code prior to calculating the LBM solution the output of which turned out to be identical between the control output and the test output.

6.2 M3InputReading

6.2.1 input-reading-id3

Unit test id3 found in Section 5.1.2 of the Unit VnV Plan (Michalski (c) verifies if the module correctly handles not being able to find an input file.

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

This output confirms that the module handles missing parameters correctly.

6.3 M4InputChecking

6.3.1 known-input-variables-id4

Unit test id4 found in Section 5.1.3 of the Unit VnV Plan (Michalski (c) verifies if the system correctly identifies a situation where the user provided input keys of the input.txt file are known to the system.

Output of test: An message of "All parameters known.".

This output confirms that the module correctly identifies a situation where an input file has known keys.

6.3.2 unknown-input-variables-id5

Unit test id5 found in Section 5.1.3 of the Unit VnV Plan (Michalski (c) verifies if the system correctly identifies a situation where the user provided input keys of the input.txt file are unknown to the system.

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

This output confirms that the module correctly identifies a situation where an input file has unknown keys.

6.3.3 missing-input-requirements-id6

Unit test id6 found in Section 5.1.3 of the Unit VnV Plan (Michalski (c) verifies if the system correctly identifies a situation where the selected Library and Problem do not have all required inputs present in the input.txt file.

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

This output confirms that the module correctly identifies a situation where an input file does not have all of the required inputs for a library problem.

6.3.4 present-input-requirements-id7

Unit test id7 found in Section 5.1.3 of the Unit VnV Plan (Michalski (c) verifies if the system correctly identifies a situation where the selected Library and Problem do have all required inputs present in the input.txt file.

Output of test: A message of "Required input variables present.".

This output confirms that the module correctly identifies a situation where an input file does have all of the required inputs for a library problem.

6.4 M8Problem

6.4.1 problem-output-id8

Unit test id8 found in Section 5.1.7 of the Unit VnV Plan (Michalski (c) verifies if the generated output of the module is correctly formatted.

The md5sum hash value of the test file 'problem-output-id8.txt' is 28984d413764f684243ba3632da3ef3d. The hash value of the test output result is e9946e61dfb8ecf717fa082c109edcc8.

Since the test resulted in a negative match a closer inspection of the two files was conducted and it was found, just as in test id1 of Section 6.1.1, that the returned data structure contains what appears to be a hashed value of either time or a location in the system that is used purely for identification purposes.

The final output image of the system again appears to be identical. As found in test id2 of Section 6.1.2, this one alphanumeric value is not relevant to the final correctness of the system.

6.5 M9Lattice

6.5.1 lattice-return-id9

Unit test id9 found in Section 5.1.8 of the Unit VnV Plan (Michalski (c) verifies if the module returns the correct velocity weights for an LBM problem.

Output of test: "[0.4444444444444444, 0.1111111111111111, 0.1111111111111111, 0.1111111111111111, 0.1111111111111111, 0.02777777777777776, 0.02777777777777776, 0.02777777777777776, 0.02777777777777776]"

This output confirms that the module returns correct velocity weights for a specified problem.

6.6 M10Boundary

6.6.1 boundary-return-id10

Unit test id10 found in Section 5.1.9 of the Unit VnV Plan (Michalski (c) verifies if the module returns desired boundary dimensions.

Output of test: "{x_min: 0.0, x_max: 3.0, y_min: 0.0, y_max: 1.0}"

This output confirms that the module returns the desired boundary dimensions.

7 Changes Due to Testing

No changes are necessary to the first stage of implementation due to these test results.

8 Automated Testing

The Unit VnV Plan (Michalski (c)) specifies which unit tests were to be automated. Time constraints have resulted in manual testing of the tests reported in this document.

9 Trace to Requirements

A complete description of functional requirements is found in Section 7.1 of the CA (Michalski (a)).

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

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

A complete description of NFRs is found in Section 7.2 of the CA (Michalski (a)).

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

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

10 Trace to Modules

A complete description of modules is found in the MG (Michalski (b)).

Cases / Modules	1	2	3	4	5	6	7	8	9	10	11	12	13
id1		✓											
id2		✓											
id3			✓										
id4				✓									
id5				✓									
id6				✓									
id7				✓									
id8								✓					
id9									✓				
id10										✓			
id11		✓											
id12									✓				

Table 3: Traceability Matrix Showing the Connections Between Test Cases and Modules

11 Code Coverage Metrics

Module coverage is guaranteed for those modules that are implemented and not outsourced to external libraries. Module coverage is outline in Table 3 of Section 10.

References

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

Peter Michalski. Module Guide for Lattice Boltzmann Solvers , b. URL https://github.com/peter-michalski/LatticeBoltzmannSolvers/blob/master/docs/Design/MG/MG.pdf.

Peter Michalski. Unit Verification and Validation Plan for Lattice Boltzmann Solver, c. URL https://github.com/peter-michalski/LatticeBoltzmannSolvers/blob/master/docs/VnVPlan/UnitVnVPlan/UnitVnVPlan.pdf.