Module Interface Specification for Lattice Boltzmann Solvers

Peter Michalski

December 14, 2019

1 Revision History

Date	Version	Notes
Nov. 25, 2019	1.0	Initial Document
Dec. 13, 2019	2.0	Fix Initial Issues

2 Symbols, Abbreviations and Acronyms

See CA Documentation for Lattice Boltzmann Solvers (Michalski, a).

Contents

3 Introduction

The following document details the Module Interface Specifications for Lattice Boltzmann Solvers, which provides a library of services based on Lattice Boltzmann Methods (LBM). LBM are a family of fluid dynamics algorithms for simulating single-phase and multiphase fluid flows, often incorporating additional physical complexities (Chen and Doolen, 1998).

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found here (Michalski, b).

4 Notation

The structure of the MIS for modules comes from Hoffman and Strooper (1999), with the addition that template modules have been adapted from Ghezzi et al. (2003). The mathematical notation comes from Chapter 3 of Hoffman and Strooper (1999). For instance, the symbol := is used for a multiple assignment statement and conditional rules follow the form $(c_1 \Rightarrow r_1 | c_2 \Rightarrow r_2 | ... | c_n \Rightarrow r_n)$.

The following table summarizes the primitive data types used by Lattice Boltzmann Solvers.

Data Type	**Notation**	**Description**
Boolean	Boolean	true or false
natural number	$\mathbb{N}$	a number without a fractional component in $[1, \infty)$
real	$\mathbb{R}$	any number in $(-\infty, \infty)$
string	string	single or multiple symbols or digits

The specification of Lattice Boltzmann Solvers uses some derived data types: sequences, strings, and tuples. Sequences are lists filled with elements of the same data type. Strings are sequences of characters. Tuples contain a list of values, potentially of different types. In addition, Lattice Boltzmann Solvers uses functions, which are defined by the data types of their inputs and outputs. Local functions are described by giving their type signature followed by their specification.

5 Module Decomposition

The following table is taken directly from the Module Guide document for this project.

Level 1	Level 2
Hardware-Hiding Module	
Behaviour-Hiding Module	M2: System Control Module M3: Input Reading Module M4: Input Checking Module M5: LBM Control Module M6: Streaming Module M7: Collision Module M8: Problem Module M9: Lattice Module M10: Boundary Module
Software Decision Module	M11: Image Rendering Module M12: Data Structure Module M13: Input Types Module

Table 1: Module Hierarchy

6 MIS of M2: System Control Module

The secret of this module is the algorithm to control Lattice Boltzmann Solvers.

6.1 Module

SystemControl

6.2 Uses

- Hardware Hiding
- Input Reading (Section 7)
- LBM Control (Section 9)
- Problem Parameter (Section 12)
- Boundary (Section 14)
- Image Rendering (Section 15)
- Data Structure (Section 16)

6.3 Syntax

6.3.1 Exported Constants

N/A

6.3.2 Exported Access Programs

N/A

6.4 Semantics

6.4.1 State Variables

inputData: DataStructure
problemParameters: A library dependent structure of LBM input parameters ($\mathbb{R}$).
boundaryData: A dimension and problem dependent dictionary of boundary dimensions ($\mathbb{R}$).
solution: A library dependent structure of LBM solution parameters ($\mathbb{R}$).
imageData: array[$\mathbb{R}$]

6.4.2 Environment Variables

N/A

6.4.3 Assumptions

A correctly formatted input.txt file is in the Input directory.

6.4.4 Access Routine Semantics

N/A

6.4.5 Local Functions

Name	In	Out	Exceptions
mainFunction	-	-	-

6.4.6 Local Function Semantics

mainFunction():

- transition: *out* := N/A
- exception: N/A

The function calls other modules of Lattice Boltzmann Solvers to solve the given problem. The function calls InputReading.inputArray(), followed by ProblemParameter.formatInput(), Boundary.getBoundary(), and LBMControl.performLBM(). Finally the module calls ImageRendering.imageFunc() and sends the output of that function to the hardware.

```
Procedure mainFunction():
    inputData = InputReading.inputArray()
    if inputData[Library] == 1 AND inputData[Problem] == 1:
        problemParameters = ProblemParameter.formatInput(inputData)
        boundaryData = Boundary.getBoundary(inputData)
    solution = LBMControl.performLBM(problemParameters)
    imageData = imageImageRendering.imageFunc(solution, boundaryData)
    Hardware.print(imageData)
```

7 MIS of M3: Input Reading Module

The secret of this module is the algorithm that gathers the input data.

7.1 Module

InputReading

7.2 Uses

- Hardware Hiding
- Input Checking (Section 8)
- Data Structure (Section 16)

7.3 Syntax

7.3.1 Exported Constants

N/A

7.3.2 Exported Access Programs

Name	In	Out	Exceptions
inputArray	-	inputData: DataStructure	NotFound, ErrorRead

7.4 Semantics

7.4.1 State Variables

inputData: DataStructure
readingError: Boolean

7.4.2 Environment Variables

input: lines of a string followed by a $\mathbb{R}$

7.4.3 Assumptions

N/A

7.4.4 Access Routine Semantics

inputArray():

- transition: inputData := +input

  Each line of the input file, consisting of a string key followed by a $\mathbb{R}$ value, will be read and placed into inputData.

- exception: if)input.txt $\notin$ ./Input/) $\Rightarrow$ NotFound and if (inputData $\Leftarrow$?) $\Rightarrow$ readingError = False $\Rightarrow$ ErrorRead

7.4.5 Local Functions

N/A

8 MIS of M4: Input Checking Module

This secret of this module is the algorithm that checks if input values fall within allowable parameters.

8.1 Module

InputChecking

8.2 Uses

- Data Structure (Section 16)

8.3 Syntax

8.3.1 Exported Constants

N/A

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
verifyInputs	inputData: DataStructure	Boolean	OutBounds, UnknwnParm, MssngPrb

8.4 Semantics

8.4.1 State Variables

The set of acceptableRanges is:
$LIBRARY$: $\mathbb{N}$: $\{1\}$ - Libraries are associated by numbers in the program. The numbers are available for reference in the User Guide.
$PROBLEM$: $\mathbb{N}$: $\{1\}$ - Problems are associated by numbers in the program. The numbers are available for reference in the User Guide.
$DIMENSIONS$: $\mathbb{N}$: $\{2\}$
VEL_DIRS: $\mathbb{N}$: $\{9\}$
$REYNOLDS\_MIN$: $\mathbb{R}$: $\{0.001\}$
$REYNOLDS\_MAX$: $\mathbb{R}$: $\{5000\}$
$DENSITY\_MIN$: $\mathbb{R}$: $\{0.0708\}$
$DENSITY\_MIN$: $\mathbb{R}$: $\{13.6\}$
$BULK\_VIS\_MIN$: $\mathbb{R}$: $\{0.0001\}$
$BULK\_VIS\_MIN$: $\mathbb{R}$: $\{20000\}$
$SHEAR\_VIS\_MIN$: $\mathbb{R}$: $\{0.001\}$

SHEAR_VIS_MIN: $\mathbb{R}$: {20000}
TIME_MIN: $\mathbb{N}$: {1}

8.4.2 Environment Variables

N/A

8.4.3 Assumptions

N/A

8.4.4 Access Routine Semantics

verifyInputs(inputData):

- output: (inputData[value] ! $>$ acceptableRanges) $\wedge$ (inputData[value] ! $<$ acceptableRanges) $\Rightarrow$ return True
- exception: if ((inputData[value] $>$ acceptableRanges) $\vee$ (inputData[value] $<$ acceptableRanges) $\Rightarrow$) OutBounds and if (inputData[key] $\notin$ inputTypes) $\Rightarrow$ UnknwnParm

The function will iterate through each inputValue InputTypes key and check if the key is known to the program, as well as check if associated values of known keys fall within an acceptable range of the state variables.

8.4.5 Local Functions

N/A

9 MIS of M5: LBM Control Module

The secret of this module is the algorithm which controls the LBM library.

9.1 Module

LBMControl

9.2 Uses

- Streaming (Section 10)
- Collision (Section 11)

9.3 Syntax

9.3.1 Exported Constants

9.3.2 Exported Access Programs

Name	In	Out	Exceptions
performLBM	inputData: DataStructure, problemData: $\mathbb{R}$	imageData: array[$\mathbb{R}$]	-

9.4 Semantics

9.4.1 State Variables

imageData: array[$\mathbb{R}$]

9.4.2 Environment Variables

N/A

9.4.3 Assumptions

N/A

9.4.4 Access Routine Semantics

performLBM(inputData, problemData):

- transition: *out* := imageData

  This function will generate imageData output for the desired problem using the provided parameters of inputData and problemData. It will perform the major aspects of the LBM modeling including moving fluid particles to neighbouring nodes using the Streaming Module, and handling collisions between particles using the Collision Module, iterating over time. The variable imageData will hold the vorticity vector values. The first stage of implementation will use a library for this function.

- exception: N/A

9.4.5 Local Functions

N/A

10 MIS of M6: Streaming Module

The secret of this module is the algorithm to calculate the streaming of particles.

10.1 Module

Streaming

10.2 Uses

N/A

10.3 Syntax

10.3.1 Exported Constants

N/A

10.3.2 Exported Access Programs

Name	In	Out	Exceptions
streamingFunc	Velocityi: $\mathbb{R}$, Time: $\mathbb{N}$, PositionVector: $\mathbb{R}$, Force: $\mathbb{R}$, Mass: $\mathbb{R}$	$\mathbb{R}$	NAN

10.4 Semantics

10.4.1 State Variables

N/A

10.4.2 Environment Variables

N/A

10.4.3 Assumptions

N/A

10.4.4 Access Routine Semantics

streamingFunc(Velocityi, Time, PositionVector, Force, Mass):

- transition: $out := f_i(\mathbf{x} + \mathrm{e}_i dt, t + dt) - f_i(\mathbf{x}, t)$
- exception: $(f_i(\mathbf{x} + \mathrm{e}_i dt, t + dt) - f_i(\mathbf{x}, t) >$ maximum size of primitive data type$) \Rightarrow$ NAN

The function calculates the streaming step value for each velocity direction (i).

The mapping/connections of the parameters and equation variables can be found in Section 2.2 of the CA and Section 2.2 of the System VnV Plan.

10.4.5 Local Functions

Name	In	Out	Exceptions
pdfFunc $(f(\mathbf{x}, \mathrm{e}, t))$	Velocityi: $\mathbb{R}$, Time: $\mathbb{N}$, PositionVector: $\mathbb{R}$, Force: $\mathbb{R}$, Mass: $\mathbb{R}$	$\mathbb{R}$	NAN

10.4.6 Local Function Semantics

pdfFunc(Velocityi, Time, PositionVector, Force, Mass):

- transition: $out := f(\mathbf{x}, \mathrm{e}, t) = f(\mathbf{x} + \mathrm{e}dt, \mathrm{e} + \frac{F}{kg}dt, t + dt)$
- exception: $(f(\mathbf{x} + \mathrm{e}dt, \mathrm{e} + \frac{F}{kg}dt, t + dt)) >$ maximum size of primitive data type$) \Rightarrow$ NAN

The function finds the probability that a particle is at position $\mathbf{x}$ and has velocity e at time t.

11 MIS of M7: Collision Module

The secret of this module is the algorithm to calculate the collision of particles.

11.1 Module

Collision

11.2 Uses

N/A

11.3 Syntax

11.3.1 Exported Constants

N/A

11.3.2 Exported Access Programs

Name	In	Out	Exceptions
collisionFunc	RelaxationRate : $\mathbb{R}$, Weighti: $\mathbb{R}$, Density: $\mathbb{R}$, UnitVector: $\mathbb{R}$, MacroscopicVelocity: $\mathbb{R}$, SpeedSound: $\mathbb{R}$, VelocityDirection: $\mathbb{N}$	$\mathbb{R}$	NAN

11.4 Semantics

11.4.1 State Variables

N/A

11.4.2 Environment Variables

maxVariableSize = The maximum allowable value held in the variable.
minVariableSize = The minimum allowable value held in the variable.

11.4.3 Assumptions

N/A

11.4.4 Access Routine Semantics

collisionFunc(RelaxationRate, Weighti, Density, UnitVector, MacroscopicVelocity, SpeedSound, VelocityDirection):

- transition: $out := \frac{1}{\tau}(f_i^{eq} - f_i)$
- exception: $(\frac{1}{\tau}(f_i^{eq} - f_i) > \text{maxVariableSize}) \vee (\frac{1}{\tau}(f_i^{eq} - f_i) < \text{minVariableSize}) \Rightarrow \text{NAN}$

The function calculates the collision step value for each velocity direction (i).

The mapping/connections of the parameters and equation variables can be found in Section 2.2 of the CA and Section 2.2 of the System VnV Plan.

11.4.5 Local Functions

Name	In	Out	Exceptions
edfFunc (f_i^{eq})	Weighti: $\mathbb{R}$, Density: $\mathbb{R}$, UnitVector: $\mathbb{R}$, MacroscopicVelocity: $\mathbb{R}$, SpeedSound: $\mathbb{R}$, VelocityDirection: $\mathbb{N}$	$\mathbb{R}$	NAN
relFunc (f_i)	Weighti: $\mathbb{R}$, Density: $\mathbb{R}$, UnitVector: $\mathbb{R}$, MacroscopicVelocity: $\mathbb{R}$, SpeedSound: $\mathbb{R}$, VelocityDirection: $\mathbb{N}$	$\mathbb{R}$	NAN

11.4.6 Local Function Semantics

edfFunc(Weighti, Density, UnitVector, MacroscopicVelocity, SpeedSound, VelocityDirection):

- transition: $out := f_i^{eq} = w_i p(1 + 3 \cdot \frac{\mathbf{e_i} \cdot \mathbf{u}}{c_s^2} + \frac{9}{2} \cdot \frac{(\mathbf{e_i} \cdot \mathbf{u})^2}{c_s^4} - \frac{3}{2} \cdot \frac{\mathbf{u} \cdot \mathbf{u}}{c_s^4})$
- exception: $(w_i p(1 + 3 \cdot \frac{\mathbf{e_i} \cdot \mathbf{u}}{c_s^2} + \frac{9}{2} \cdot \frac{(\mathbf{e_i} \cdot \mathbf{u})^2}{c_s^4} - \frac{3}{2} \cdot \frac{\mathbf{u} \cdot \mathbf{u}}{c_s^4}) > \text{maxVariableSize}) \vee (w_i p(1 + 3 \cdot \frac{\mathbf{e_i} \cdot \mathbf{u}}{c_s^2} + \frac{9}{2} \cdot \frac{(\mathbf{e_i} \cdot \mathbf{u})^2}{c_s^4} - \frac{3}{2} \cdot \frac{\mathbf{u} \cdot \mathbf{u}}{c_s^4}) < \text{minVariableSize}) \Rightarrow \text{NAN}$

The function calculates the equilibrium distribution for each velocity direction (i).

relFunc(Weighti, Density, UnitVector, MacroscopicVelocity, SpeedSound, VelocityDirection):

- transition: $out := f_i = f_i - \frac{1}{\tau}(f_i^{eq} - f_i)$
- exception: $(f_i - \frac{1}{\tau}(f_i^{eq} - f_i) > \text{maxVariableSize}) \vee (f_i - \frac{1}{\tau}(f_i^{eq} - f_i) < \text{minVariableSize})$ $\Rightarrow$ NAN

The function calculates the relaxation update for each velocity direction (i).

12 MIS of M8: Problem Module

The secret of this module is the structure of the LBM input parameters.

12.1 Module

ProblemParameter

12.2 Uses

- Lattice (Section 13)
- Data Structure (Section 16)

12.3 Syntax

12.3.1 Exported Constants

N/A

12.3.2 Exported Access Programs

Name	In	Out	Exceptions
formatInput	inputData: DataStructure	problemData: $\mathbb{R}$	-

12.4 Semantics

12.4.1 State Variables

N/A

12.4.2 Environment Variables

N/A

12.4.3 Assumptions

N/A

12.4.4 Access Routine Semantics

formatInput(inputData):

- transition: $out :=$ problemData $= \mathbb{N} \in$ inputData, $\mathbb{R} \Leftarrow$ Boundary.getBoundary()

- exception: N/A

The function will set up the structure for the LBM input parameters based on the library that the user has requested. The function will use the Lattice Module functions and input dictionary to create a dictionary that will be input into the LBM Control Module.

12.4.5 Local Functions

N/A

13 MIS of M9: Lattice Module

The secret of this module is the structure of the lattice model.

13.1 Module

Lattice

13.2 Uses

- Data Structure (Section 16)

13.3 Syntax

13.3.1 Exported Constants

parameterValues := (9, 4/9, 1/9, 1/9, 1/9, 1/9, 1/36, 1/36, 1/36, 1/36)

The above holds the coefficient weights for velocity directions of a Q9 model.

13.3.2 Exported Access Programs

Name	In	Out	Exceptions
getWeights	inputData: DataStructure	$\mathbb{R}$	NoLattice

13.4 Semantics

13.4.1 State Variables

N/A

13.4.2 Environment Variables

N/A

13.4.3 Assumptions

N/A

13.4.4 Access Routine Semantics

getWeights(inputData):

- transition: *out* := if (inputData[VelocityDirections] == 9) $\wedge$ (inputData[Dimensions] == 2) $\Rightarrow$ return parameterValues

- exception: inputData[VelocityDirections] $\notin$ parameterValues $\Rightarrow$ NoLattice

The function sets coefficient weight data for the input lattice model, and returns an error if the weight data is not available for the input lattice model.

13.4.5 Local Functions

N/A

14 MIS of M10: Boundary Module

The secret of this module is the structure of the model boundary.

14.1 Module

Boundary

14.2 Uses

- Data Structure (Section 16)

14.3 Syntax

14.3.1 Exported Constants

boundary_data = {“x_min”: 0., “x_max”: 3., “y_min”: 0., “y_max”: 1.}

14.3.2 Exported Access Programs

Name	**In**	**Out**	**Exceptions**
getBoundary	inputData: DataStructure	$\mathbb{R}$	-

14.4 Semantics

14.4.1 State Variables

N/A

14.4.2 Environment Variables

N/A

14.4.3 Assumptions

N/A

14.4.4 Access Routine Semantics

getBoundary(inputData):

- transition: *out* := if (inputData[Library] == 1) $\wedge$ (inputData[Problem] == 1) $\wedge$ (inputData[Size] == 1) $\Rightarrow$ returns boundary_data
- exception: N/A

The function returns the boundary from the DataStructure library.

14.4.5 Local Functions

N/A

15 MIS of M11: Image Rendering Module

The algorithm to convert the LBM output into an image.

15.1 Module

ImageRendering

15.2 Uses

N/A

15.3 Syntax

15.3.1 Exported Constants

N/A

15.3.2 Exported Access Programs

Name	In	Out	Exceptions
imageFunc	imageData: array[$\mathbb{R}$]		IncorrectFormat

15.4 Semantics

15.4.1 State Variables

N/A

15.4.2 Environment Variables

imageOut: PNG

15.4.3 Assumptions

N/A

15.4.4 Access Routine Semantics

imageFunc(imageData[]):

- transition: out := imageOut: PNG
- exception: (imageData[] $\neq \mathbb{R} \Rightarrow$ IncorrectFormat)

The function converts the information from the LBM algorithm output (for example the vorticity vector values) into an image format using an external image library.

15.4.5 Local Functions

N/A

16 MIS of M12: Data Structure Template Module

The format of a non-primitive data structure for input variables.

16.1 Template Module

DataStructure

16.2 Uses

- Input Types (Section 17)

16.3 Syntax

16.3.1 Exported Types

N/A

16.3.2 Exported Access Programs

Routine Name	In	Out	Exceptions
init	-	-	-
add	InputTypes, $\mathbb{R}$	-	IncorrectFormat
getElm	InputTypes	$\mathbb{R}$	DoesNotExist

16.4 Semantics

16.4.1 State Variables

s: set of tuple of (key: InputType, value: $\mathbb{R}$)

16.4.2 State Invariant

N/A

16.4.3 Assumptions

N/A

16.4.4 Access Routine Semantics

init():

- transition: $s := \{\}$
- exception: N/A

add(InputTypes, $\mathbb{R}$):

- transition: $s := s \cup \{\langle$ InputTypes, $\mathbb{R} \rangle\}$
- exception: ($\langle$ InputTypes, value $\rangle$ | value is not a $\mathbb{R} \Rightarrow$ IncorrectFormat)

remove(InputTypes):

- transition: $s := s$ - $\{\langle$ InputTypes, $\mathbb{R} \rangle\}$ where $\langle$ InputTypes, $\mathbb{R} \rangle \in s$
- exception: ($\langle$ InputTypes, $\mathbb{R} \rangle \notin s \Rightarrow$ DoesNotExist)

getElm(InputTypes):

- transition: $out := \mathbb{R}$ if $\langle$ InputTypes, $\mathbb{R} \rangle \in s$
- exception: ($\langle$ InputTypes, $\mathbb{R} \rangle \notin s \Rightarrow$ DoesNotExist)

16.4.5 Local Functions

N/A

17 MIS of M13: Input Types Module

The input types for the non-primitive data structures.

17.1 Module

InputTypes

17.2 Uses

N/A

17.3 Syntax

17.3.1 Exported Constants

InputTypes (string) = {Library, Problem, Dimensions, VelocityDirections, ReynoldsNumber, Density, BulkViscosity, ShearViscosity, Time, Density, Viscosity, AccelerationRate, SpeedSound, Velocity, Force, Mass, CrossSectionalArea, MacroscopicVelocity, RelaxationRate}

17.3.2 Exported Access Programs

N/A

17.4 Semantics

17.4.1 State Variables

N/A

17.4.2 State Invariant

N/A

17.4.3 Assumptions

N/A

17.4.4 Access Routine Semantics

N/A

17.4.5 Local Functions

N/A

References

Shiyi Chen and Gary D Doolen. Lattice Boltzmann method for fluid flows. *Annual review of fluid mechanics*, 30(1):329–364, 1998.

Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2003.

Daniel Hoffman and Paul Strooper. Software Design, Automated Testing, and Maintenance–A Practical Approach. 1999.

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

Peter Michalski. Lattice Boltzmann Solvers, b. URL https://github.com/peter-michalski/LatticeBoltzmannSolvers.

18 Appendix

Table 2: Possible Exceptions

Message ID	Error Message
DoesNotExist	Error: The input type does not exist.
ErrorRead	Error: Could not read the input file.
IncorrectFormat	Error: The format is not a real number.
MssngPrb	Error: The input.txt file is missing (or has incorrect) required fields for the designated problem. Please see the User Guide.
NAN	Error: The calculated result is not a number.
NoLattice	Error: The chosen velocity directions do not have a known lattice structure.
NotFound	Error: Input file not found.
OutBounds	Error: The input file parameter X is out of bounds. Please see the User Guide.
UnknwnParm	Error: The parameter X is not known to the system. Please see the User Guide.