Improving the Accuracy of the Lattice Boltzmann Method with a Minimal Computational Cost Trade-off Through an Iterative Averaging Algorithm

Arsha Niksa
ar.niksa@outlook.com

August 22, 2021

1 Objectives

Primary Objective: This project primarily aims at creating more physically-reliable simulations which utilise the Lattice Boltzmann method.
Secondary Objective: This project subsidiarily aims at minimizing the computational efficiency issues stemming from successfully achieving the primary objective.

2 Research Questions

In this project, answers to the following questions will be sought:

1. Is the implementation of an Iterative Averaging Algorithm technically feasible?
2. What is the time complexity of the method used in this project and how does it compare to other Lattice Boltzmann simulations which are readily available (such as the simulations provided by Ref.1)?
3. How does the accuracy of the results yielded by using the method proposed in this project compare to other Lattice Boltzmann simulations which are readily available?

3 Research Methodology

The Lattice Boltzmann method (or LBM in short) acts as a median between macroscopic and microscopic world, which is largely owed to its implementation of Statistical Mechanics [1]. This allows the users of LBM to avoid the disadvantages which are often associated with Molecular Dynamics (MD) simulations. These disadvantages include, but are not limited to, requiring large storage space and having an unfathomable computational cost, which make the use of MD impractical for large-scale simulations [2].

While this apparent advantage gives LBM a relative edge with respect to other methods, LBM does share its own set of limitations. To better understand this, we have

to identify two defining components of LBM: 1) nodes and 2) timesteps. If we are to tweak these components to achieve better and more accurate results, we will have to inevitably face a trade-off of accuracy and computational cost. As an example, if we increase the length of each timestep in an LBM-based simulation, which can decrease the computational cost of the simulation, we will have to face a loss in the accuracy of the simulation. However, if this trade-off is accompanied by an additional averaging process, the implementation of which will be investigated in this project, this trade-off might be minimized.

To achieve this, an Iterative Averaging Algorithm (or IAA in short) will be used. To expand on this, let us assume that we have a D2Q9 model for simulating flow inside a 2D rectangular channel with arbitrary dimensions. In this model, we have N nodes, each of which are defined by 9 velocity distribution functions, namely f_i where $i \in \cup_{i=0}^{8} i$. Each of the nodes are further defined by local particle velocities, which are denoted as e_i [1,3].

At first, the LBM simulation will be run for a short amount of time, t_{test}, with a timestep of δ_{test}. After storing the results of the first run, which we will call the test run, we will run the simulation again for t_{IAA} seconds, where $t_{IAA} = t_{test}$, with a timestep of δ_{IAA}, where $\delta_{IAA} > \delta_{test}$. With the increased length of each timestep, it is expected to decrease the computational cost of the simulation in the second run and thus, face a shorter runtime. However, as explained before, a loss in the accuracy of the simulation results will be inevitably faced. In other words, execution speed will have to be traded with the accuracy of the results. To avoid this trade-off, IAA will be implemented. Let us assume that the second run of the simulation is active, being at time $t_{example}$, where $t_{example} \in [0, t_{IAA}]$. For an exemplary node, f_i is determined per each of the local directions. If we go a timestep, namely t_{IAA}, further in time, the values of f_i will be updated. At this stage, we will take the value-wise average of the functions f_i at $t = t_{example}$ and f_i at $t = t_{example} + \delta_{IAA}$ and we will call the resulting function a_i at $t = t_{example} + \delta_{IAA}$. Now that a_i is defined, we will compare it to an f_i in the test run which is nearest in time to a_i. Then, we will take the difference of the functions f_i (which is for the first run and is different from the function of the same name which is produced in the second run) and a_i and we will call it $d_i = a_i - f_i$. Having defined the difference function, d_i, we will move onto defining another function, w_i, which is the weight function per each input value of d_i. Note that at first, w_i will be a constant function, the value of which will be arbitrarily defined by the user.

With both d_i and w_i defined, we will take their product, $w_i d_i$, and add it to f_i at $t = t_{example} + \delta_{IAA}$ in the second run. If f_i in the second run is consistent with another f_i which we will take from the nearest time in the test run, then the weight function for the ith direction is correct. If not, the correct w_i will be recalculated and updated. This process is iteratively repeated until a satisfactory w_i is yielded.

If we acquire the desired w_i, then we can run the simulation for a longer time with a longer timestep which will be faster with a minimal loss of accuracy. Put differently, the computational cost will decrease due to having longer timesteps (and thus, decreasing the number of steps), while w_i will closely replicate what would have happened to f_i if an additional timestep was to be added.

This method has been recapitulated in Fig.1.

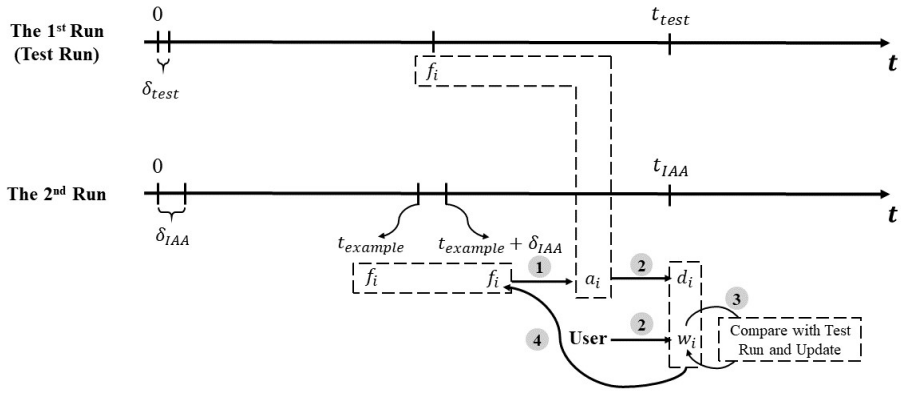

Figure 1: A visual demonstration of the proposed methodology is provided.

A possible setback of this method would be the fact that the computational cost associated with calculating w_i will not necessarily be the same for all configurations and might even exceed the computational cost of ordinary LBM-based simulations. This problem can be solved by decreasing the testing time, t_{test}, and thus, t_{IAA} to decrease the computational cost of calculating w_i. This can lead to a loss of accuracy. Whether this loss is greater than the inaccuracy yielded by simply increasing the timestep length is a question to be investigated.

References

[1] A. A. Mohamad, Lattice Boltzmann Method: Fundamentals and Engineering Applications with Computer Codes, 2nd Ed [Internet], Springer (2019). Available from: https://www.springer.com/gp/book/9781447160991. [Cited on Aug 3 2021]
[2] V. Esfahanian, Fundamentals of Computational Fluid Dynamics, Department of Mechanical Engineering, University of Tehran (2015) [Cited on Aug 3 2021]
[3] S. Chen, G. D. Doolen, Lattice Boltzmann Method for Fluid Flows [Internet], Annu. Rev. Fluid Mech (1998). Available from: https://www.annualreviews.org/doi/abs/10.1146/annurev.fluid.30.1.329?journalCode=fluid. [Cited on Aug 3 2021]