gLBM: A GPU enabled Lattice Boltzmann Method Library

Aaron Bray[1], **Rachel B. Clipp**[*1], **M. Umar Qureshi**[1], **Sorin Mitran**[2], and **Andinet Enquobahrie**[1]

1 Kitware, Inc., Carrboro, NC 27510 **2** Department of Mathematics, University of North Carolina, Chapel Hill, NC 27599-3250

DOI: 10.21105/joss.02555

Software

- Review
- Repository
- Archive

Editor: Eloisa Bentivegna

Reviewers:

- @ianhinder
- @gouarin

Submitted: 24 July 2020
Published: 14 February 2022

Summary

Lattice Boltzmann Methods (LBM) are a class of computational fluid dynamics (CFD) algorithms for simulation. Unlike traditional formulations that simulate fluid dynamics on a macroscopic level with a mesh, the LBM characterizes the problem on a mesoscopic level applied to a grid discretization. LBM solves the fluid density problem with collide and stream (relaxation) processes. This approach has several advantages, including its adaptability to numerous fluid domains (i.e., vapours, liquid droplets), complex boundaries, irregular interior geometries, and parallelization.
Traditional CFD methods are limited in the ability to parallelize the algorithm; however, the LBM algorithm discretization can be easily parallelized both for CPUs and GPUs. This enables fast fluid solutions for complex fluid domains. There are limitations associated with the LBM, including high Mach number applications. However, active research is addressing these limitations.

Statement of need

The `gLBM` library is the three-dimensional Lattice Boltzmann algorithm implemented using GPUs to accelerate the fluid solver. The library is implemented in C++ and Cuda and is validated using a robust suite of custom verification and validation tools for sustainable community-based use and development. `gLBM` leverages an easy to use API that is well-documented to import geometries for analysis using formats supported by ITK (McCormick et al., 2014), the open source Insight Toolkit, and configuration files that define the fluid parameters, grid discretization properties, and simulation parameters. An Apache 2.0 license was selected to support the widest distribution and use of the `gLBM` library. This stands in contrast to the copyleft licenses of other available libraries ("Lattice Boltzmann (LBM) Simulation Package for GPUs (CUDA, OpenCL)," 2020; "OpenLB – Open Source Lattice Boltzmann Code," 2020; "Palabos," 2020), which limit usability for commercial projects seeking to protect intellectual property.

The LBM algorithm builds on established work by Krüger et al. (2017), He & Luo (1997), Latt et al. (2008), Succi (2001), and Ubertini et al. (2010) to solve for the pressure and flow in the fluid domain. `gLBM` expands on this work to integrate solutions for wall shear stress and temperature. Initial work using this library has been published, applied to analysis of the upper airways (Rachel B. Clipp et al., 2018; Rachel B. Clipp et al., 2019; Quammen et al., 2016). The combination of GPU implementation, platform independence, permissive

[*] corresponding author

licensing, and integrated verification and validation make this library unique when compared to other available libraries of the LBM, some of which include a GPU implementation ("Advanced Simulation Library," 2015; "GPU-LBM-Simulation," 2014; "Lattice Boltzmann (LBM) Simulation Package for GPUs (CUDA, OpenCL)," 2020; "Lbm-Gpu," 2014; "OpenLB – Open Source Lattice Boltzmann Code," 2020; "Palabos," 2020). When investigating these libraries, we found that some had no license stipulations, which defaults to GitLab's most stringent license, or had a license ideal for academics and researchers but placed limitations on commercial use. Therefore, we chose the permissible Apache 2.0 license to ensure academic and commercial freedom and address our needs. Our library is limited to NVIDIA graphics cards due to the CUDA implementation, but does provide cross-platform functionality. Our D3Q19 implementation of the LBM algorithm has been tested in standard geometries, such as channel flow and biomedical applications, such as the nasal airways, and can be used for fluid flow in closed domains at this time. The use of a verification and validation suite provides a means to optimize and update algorithms and easily ensure the integrity of the solution is maintained. This library is ideal for students, researchers, and industry users looking to expand their use of the LBM and will be supported and maintained by Kitware, Inc., leaders in open source software development.

gLBM in Action

The typical LBM algorithm relies on a lattice connectivity rather than a mesh configuration. The lattice connectivity then results in a probability distribution function (PDF) for velocity (Equation 1).

$$f_i(x,t) = f(x, c_i, t) \tag{1}$$

The "stream and collide" algorithm can be described by the time step shift and relaxation that occur for the PDF. The PDF (f_i), for component i is shifted during one time step to the new position, $x + c_i t$; this is the "streaming" step. At each node, the continuum fluid density ρ and velocity u are evaluated as the moments of the PDF. The PDFs for each node in the lattice are then relaxed towards the thermodynamic equilibrium values (Equation 2). This represents the molecular collision or the "collision" step.

$$f_{i,eq} = \omega_i \rho (1 + \frac{\mu c_i}{c_s^2} + \frac{(\mu c_i)^2}{c_s^4} - \frac{\mu^2}{2c_s^2}) \tag{2}$$

where u is the continuum velocity (first moment of the particle velocities), c_s is the speed of sound, and ω_i are lattice-specific constants.

We applied a constant pressure boundary condition at the inlet and outlet and enforced a zero velocity boundary condition at the walls for all simulations.

Numerical Verification

We verified the LBM implemented in `gLBM` by comparing the solution in cylindrical and rectangular channels against the analytic solutions. The cylindrical channel of length 0.1 meters and diameter of 0.01 meters was executed for a grid spacing (Δx) of 0.0004, 0.0005, 0.0006, and 0.0007 meters. A rectangular channel with dimensions of 0.08 meters in the x-direction, 0.02 meters in y-direction, and 0.01 meters in the z-direction was analyzed with the same grid spacings. The solutions were executed for 10,000 iterations under an imposed constant pressure gradient. The results for numerical and grid convergence were similar for both geometries, with the results showing an iteration convergence occurring at less than

3,000 iterations and grid convergence evident. The analytical solutions are implemented in the gLBM library for future verification and validation efforts. The comparison between the analytical and the computed solutions for the rectangular channel are shown in **Figure 1**. For both geometries, we observed an entrance effect and a fast convergence to a fully developed parabolic profile with less than 1% error. This is clearer when investigating the results slice by slice through both channels, as shown in **Figure 2**.

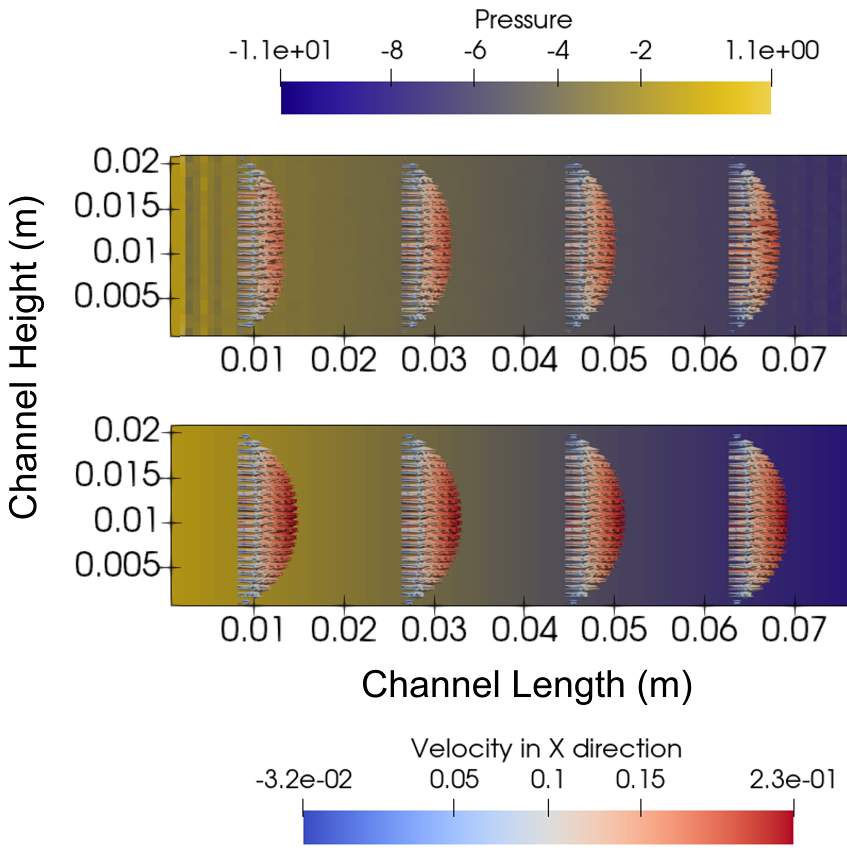


Figure 1: The analytical solution (bottom) is shown with the LBM computed solution (top) in ParaView for the axial velocity profile that varies along the channel width.

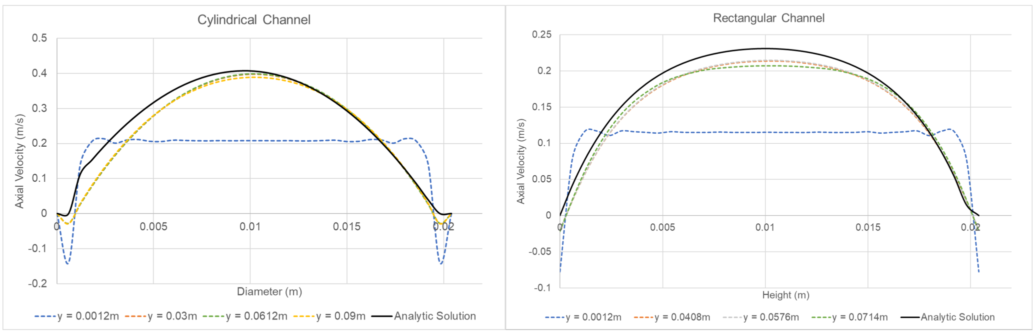


Figure 2: The solid black line depicts the exact solution (Poiseuille velocity profile) and the dashed lines represent the LBM solution at locations along the length of the cylinder.

Automated Verification and Validation

To maintain its validity, we developed a verification and validation (V&V) suite to continually verify any algorithm changes and automatically execute multiple geometries. `gLBM` includes a verification execution library that is designed to simulate a list of geometries found in a configuration file. The analytical solutions for the cylindrical and rectangular channels are included in the V&V library. The dimensions, grid discretization, and fluid and simulation parameters can be set in the configuration file, which applies to both the LBM and analytical results. This allows for reproducible, easy simulation of the analytic cases with auotmatic error reporting and plotting, which provides a method to continuously verify the numerical solution with the analytic solution during algorithm development and comparison. We automatically calculate the error at designated slices (axial locations) along the geometry and list these errors in a table for easy evaluation. We also automatically plot the velocity profile in each dimension, overlaying the analytic solution, the baseline (the previously validated or best case results), and the computed solution for easy visual inspection, as shown in **Figure 3**. For more complex geometries, where the analytical solution is unavailable, only the baseline and current results are plotted for evaluation. As changes to the `gLBM` libary are made, it is easy to compare results to ensure they are moving closer to the analytic solution. We also plot the overall error at each iteration of the solution to evaluate the convergence time for each solution.

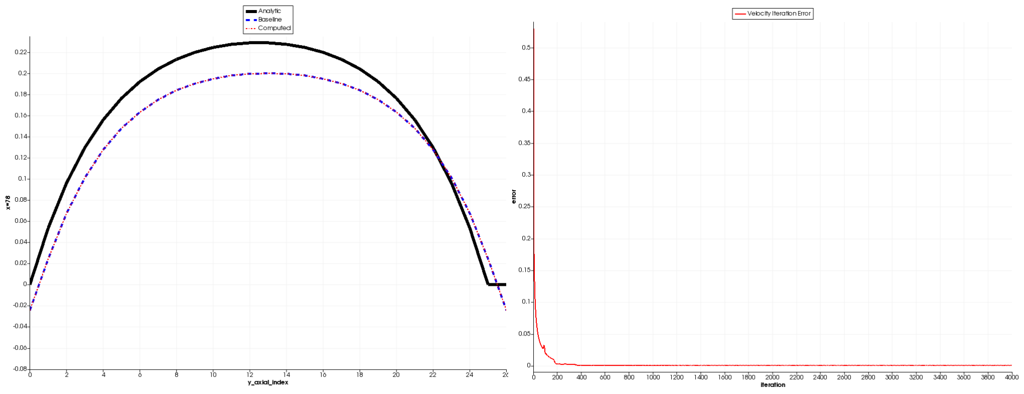
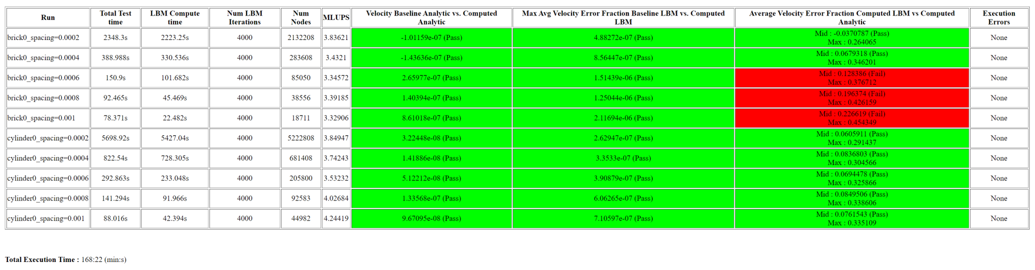


Figure 3: Example results from the verification and validation suite.

A summary table is also generated for developers to quickly assess the overal performance and verification data for each V&V run performed. Computational performance of each run is performed by calculating the Million Lattice Updates Per Second (MLUPS) on the provided geometry.



Run	Total Test time	LBM Compute time	Num LBM Iterations	Num Nodes	MLUPS	Velocity Baseline Analytic vs. Computed Analytic	Max Avg Velocity Error Fraction Baseline LBM vs. Computed LBM	Average Velocity Error Fraction Computed LBM vs Computed Analytic	Execution Errors
brick0_spacing=0.0002	2348.3s	2223.25s	4000	2132208	3.83621	-1.01159e-07 (Pass)	4.88272e-07 (Pass)	Mid : -0.0370787 (Pass) Max : 0.264065	None
brick0_spacing=0.0004	388.988s	330.536s	4000	283608	3.4321	-1.43636e-07 (Pass)	8.56447e-07 (Pass)	Mid : 0.0679318 (Pass) Max : 0.346201	None
brick0_spacing=0.0006	150.9s	101.682s	4000	85050	3.34572	2.65977e-07 (Pass)	1.51439e-06 (Pass)	Mid : 0.128386 (Fail) Max : 0.376712	None
brick0_spacing=0.0008	92.465s	45.469s	4000	38556	3.39185	1.40394e-07 (Pass)	1.25044e-06 (Pass)	Mid : 0.196374 (Fail) Max : 0.426159	None
brick0_spacing=0.001	78.371s	22.482s	4000	18711	3.32906	8.61018e-07 (Pass)	2.11694e-06 (Pass)	Mid : 0.226619 (Fail) Max : 0.454349	None
cylinder0_spacing=0.0002	5698.92s	5427.04s	4000	5222808	3.84947	3.22448e-08 (Pass)	2.62947e-07 (Pass)	Mid : 0.0605911 (Pass) Max : 0.291437	None
cylinder0_spacing=0.0004	822.54s	728.305s	4000	681408	3.74243	1.41886e-08 (Pass)	3.3533e-07 (Pass)	Mid : 0.0836803 (Pass) Max : 0.304566	None
cylinder0_spacing=0.0006	292.863s	233.048s	4000	205800	3.53232	5.12212e-08 (Pass)	3.90879e-07 (Pass)	Mid : 0.0694478 (Pass) Max : 0.325866	None
cylinder0_spacing=0.0008	141.294s	91.966s	4000	92583	4.02684	1.33568e-07 (Pass)	6.06265e-07 (Pass)	Mid : 0.0849506 (Pass) Max : 0.338606	None
cylinder0_spacing=0.001	88.016s	42.394s	4000	44982	4.24419	9.67095e-08 (Pass)	7.10597e-07 (Pass)	Mid : 0.0761543 (Pass) Max : 0.335109	None

Total Execution Time : 168:22 (min:s)

Figure 4: Example summary results from the verification and validation suite.

Future Directions

Our team is working towards a fast fluid solution for the upper airways to enable clinically relevant analysis of patient-specific surgical analysis. Our initial studies have successfully executed the gLBM library for this domain with mixed results. The library is able to produce reasonable results across the geometry studied; however, the local results within the geometry need further work and the execution speed is not optimal. An example of this geometry is included in the open source repository. Though initial results show promise, more work is required to improve the accuracy of the simulation. We also plan to address the high mach number limitations to perform aerospace simulations in hours, rather than the currently required days of analysis. Our future work will expand on the initial results shown in **Figure 5** and advancements will be committed to the gLBM repository. We also plan to optimize the CUDA implementation for faster performance. A final domain we are exploring is in open boundary solutions, such as airfoils, which requires a difference boundary and boundary condition implementation.

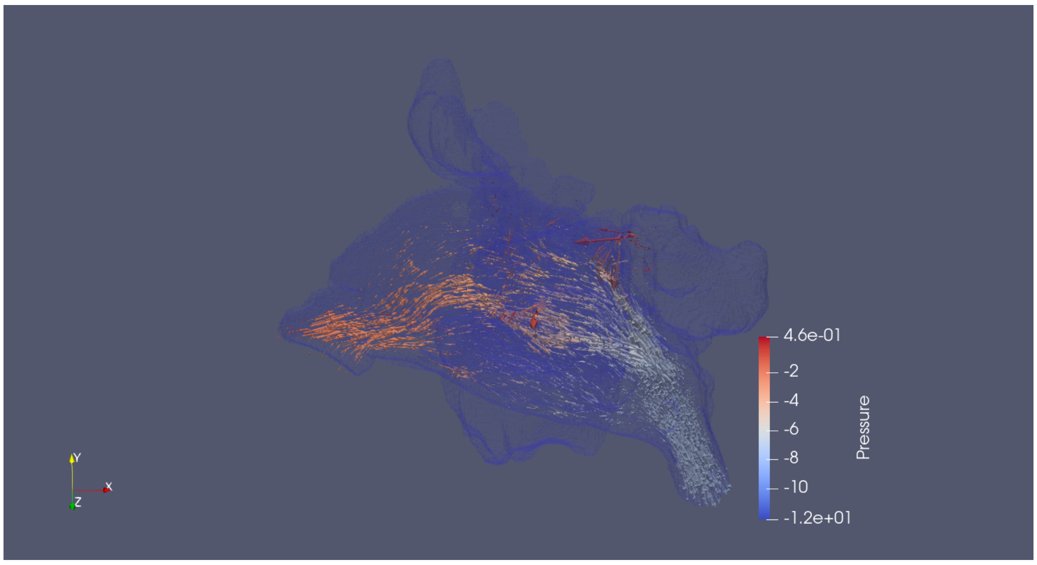


Figure 5: Initial results for the LBM simulation in the upper airways.

Acknowledgements

This study was supported by the National Institutes of Health Grant Number R44EB023121.

References

Advanced simulation library. (2015). In *GitHub repository*. GitHub. http://asl.org.il/

Clipp, Rachel B., Vicory, J., Horvath, S., Mitran, S., Kimbell, J. S., Rhee, J. S., & Enquobahrie, A. (2018). An interactive, patient-specific virtual surgical planning system for upper airway obstruction treatments. *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 5802–5805. https://doi.org/10.1109/EMBC.2018.8513672

Clipp, Rachel B., Vicory, J., Mitran, S., S., K. J., Rhee, J. S., & Enquobahrie, A. P. (2019). Towards a real-time solution for virtual surgical planning of upper airway procedures, including computational fluid dynamics. *Computer Methods in Biomechanics and Biomedical Engineering*.

GPU-LBM-simulation. (2014). In *GitHub repository*. GitHub. https://github.com/mrpgraae/GPU-LBM-Simulation

He, X., & Luo, L.-S. (1997). Lattice Boltzmann model for the incompressible Navier–Stokes equation. *Journal of Statistical Physics*, *88*(3-4), 927–944. https://doi.org/10.1103/PhysRevE.56.6811

Krüger, T., Kusumaatmaja, H., Kuzmin, A., Shardt, O., Silva, G., & Viggen, E. M. (2017). *The lattice Boltzmann method*. Springer. https://doi.org/10.1007/978-3-319-44649-3

Latt, J., Chopard, B., Malaspinas, O., Deville, M., & Michler, A. (2008). Straight velocity boundaries in the lattice Boltzmann method. *Physical Review E*, *77*(5), 056703. https://doi.org/10.1103/PhysRevE.77.056703

Lattice boltzmann (LBM) simulation package for GPUs (CUDA, OpenCL). (2020). In *GitHub repository*. GitHub. https://github.com/sailfish-team/sailfish

Lbm-gpu. (2014). In *GitHub repository*. GitHub. https://github.com/nyxcalamity/lbm-gpu

McCormick, M. M., Liu, X., Ibanez, L., Jomier, J., & Marion, C. (2014). ITK: Enabling reproducible research and open science. *Frontiers in Neuroinformatics*, *8*, 13. https://doi.org/10.3389/fninf.2014.00013

OpenLB – open source lattice boltzmann code. (2020). In *Website*. OpenLB. https://www.openlb.net/download/

Palabos: Parallel lattice boltzmann solver. (2020). In *GitHub repository*. GitHub. https://gitlab.com/unigespc/palabos

Quammen, C. W., Taylor 2nd, R. M., Krajcevski, P., Mitran, S., Enquobahrie, A., Superfine, R., Davis, B., Davis, S., & Zdanski, C. (2016). The virtual pediatric airways workbench. *Studies in Health Technology and Informatics*, *220*, 295. https://doi.org/10.3233/978-1-61499-625-5-295

Succi, S. (2001). *The lattice Boltzmann equation: For fluid dynamics and beyond*. Oxford University Press. https://doi.org/10.1063/1.1537916

Ubertini, S., Asinari, P., & Succi, S. (2010). Three ways to lattice Boltzmann: A unified time-marching picture. *Physical Review E*, *81*(1), 016311. https://doi.org/10.1103/PhysRevE.81.016311