SECTION XIX. SYSTEM ANALYSIS, MODELING AND OPTIMIZATION

DOI 10.36074/logos-10.12.2021.v2.10

FLOW MODELLING AROUND AN OSCILLATING OBJECT IN A CHANNEL BY THE LATTICE BOLTZMANN METHOD ON GPU DEVICES

Khliestov Illarion
Student of the Faculty of Information Technologies
Pryazovskyi State Technical University

ORCID ID: 0000-0003-3368-8203

Ostapenko Artem
Phd (Technical Sciences),
Senior Teacher of the Department of High and Applied Mathematics
Pryazovskyi State Technical University

UKRAINE

Abstract. *This paper addresses the idea of evaluation fluid flow behaviour in case of an oscillating object in a channel. Simulation performed with the Lattice Boltzmann method using the origin program. Fluid flows over a circular cylinder with R = 0.00625 at different Reynolds numbers were modeled to verify the computation algorithm. Implementation was designed to work on CPU and GPU devices based on the Tensorflow framework.*

1. Introduction. Research of the motion of a viscous fluid near bodies of various shapes is one of the most important problems in nonlinear fluid mechanics. Interest in this problem is caused by the formation of zones of closed reverse flows , the instability of these zones, starting from a certain critical Reynolds number, their separation from the body and descent into the wake region.

Despite that fluid behaviour can be described via Navier-Stokes equation, this approach is not the easiest one in terms of implementation. Thus the Lattice Boltzman method became quite widespread. Although there are a lot of implementations of this method, no solution available for simple transfer between CPU and GPU devices was made.

2. Related work. Many studies explored and assessed the Lattice Boltzmann Method. A detailed analysis was presented by Shuling Hou et al. [1], Shiyi Chen, and Gary D. Doolen [2] to highlight the capabilities of the lattice Boltzmann approach. Error and convergence rate studies were conducted. For various maximum velocities, compressibility effects are quantified, and parameter ranges for stable simulations are discovered. P. A. Skordos looked at the current boundary conditions and offered a new one[3]. A comprehensive assessment of existing LBM implementations was carried out by KeijoMattila et al[4].

More emphasis was placed on method implementation and optimization in a more recent study. On a hybrid CPU–GPU accelerated system, Yu Ye et al[5] investigated parallel computation of the Entropic Lattice Boltzmann technique.

Christian Obrecht et al 2010[6], Marco-FelipeKing et al[7], and J. Habich et al[8] work on GPU-based LBM implementations using diverse techniques. M. Januszewski et al 2013[9], C. Obrecht et al 2011[10] focuses on multi-GPU solutions.

Hurlbut S. E et all[11], Sheng Bao[12] evaluated behaviour of oscillating objects numerically. Nature experiments for similar conditions were performed by Koopmann G. H[13].

3. Lattice Boltzmann method description and implementation. Generally motion of the viscous fluid can be fully described by Navier–Stokes equations partial differential equations. It mathematically expresses conservation of momentum and conservation of mass for Newtonian fluids. But despite its strength and power Navier-Stokes equations hard to implement and paralize on modern devices due to its continuum nature. Thus other approaches more suitable for discrete space were developed.

One of such algorithms is the lattice Boltzman method. The main idea behind it is to represent fluid on a mesoscopic level with a few assumptions and approximations. It models the fluid consisting of fictive particles, and such particles perform consecutive propagation and collision processes over a discrete lattice. Those particles describe fluid though discrete-velocity distribution function $f_i(x,t)$. Function represents the density of particles with velocity $c_i = (c_{ix}, c_{iy}, c_{iz})$ at position x and time t. The mass density ρ and momentum density ρu at (x,t) can be found through weighted sums known as momentum of f_i[14]:

$$\rho(x,t) = \sum_i f_i(x,t) \quad (1)$$

$$\rho u(x,t) = \sum_i c_i f_i(x,t) \quad (2)$$

Each lattice in this case can be represented as a bunch of predefined vectors with corresponding velocity weights. For 2d case D2Q9 lattice description is a regular choice (fig. 1):

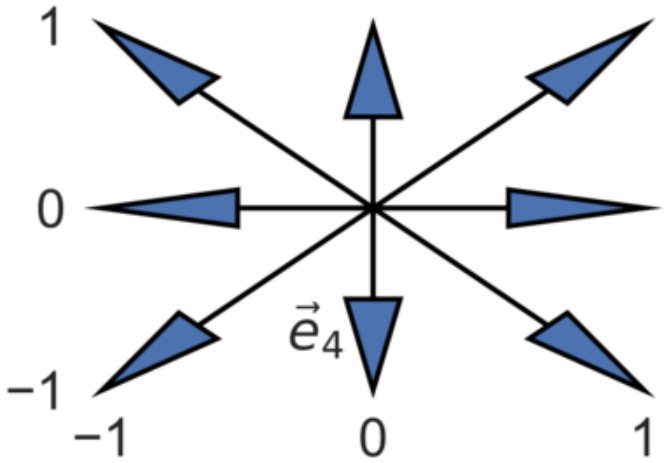


Fig. 1. **D2Q9 lattice**

By discretizing the Boltzmann equation in time and physical and velocity spaces we find the lattice Boltzmann equation:

$$f_i(x + c_i\Delta t, t + \Delta t) = f_i(x,t) + \Omega_i(x,t) \quad (3)$$

where Ω_i is the collision operator. While there are many different collision operators available, the simplest one that can be used for Navier-Stokes simulations is the Bhatnagar-Gross-Krook (BGK) operator:

$$\Omega_i(f) = -(f_i - f_i^{eq}) / \tau * \Delta t \quad (4)$$

It relaxes the populations towards an equilibrium f_i^{eq} at a rate determined by the relaxation time τ. Equilibrium is given by

$$\mathrm{f_i^{eq}(x,t) = w_i\rho(1 + \frac{u\cdot c_i}{{c_s}^2} + \frac{(u\cdot c_i)^2}{2{c_s}^4} - \frac{u\cdot u}{2{c_s}^2})} \quad (5)$$

with the weights w_i specific to the chosen velocity set. The equilibrium f_i^{eq} depends on the local quantities density ρ and fluid velocity u only. These are calculated from the local values of f_i , with the fluid velocity found as $u(x,t) = \rho u(x,t) / \rho(x,t)$.

During implementation the fully discretized Boltzmann equation with the BGK collision operator can be separated in two isolated steps: collision and streaming. Boundary conditions are handled additionally.

The first part is collision (or relaxation)

$$\mathrm{f_i^*(x,t) = f_i(x,t) - \frac{\Delta t}{\tau}(f_i(x,t) - f_i^{eq}(x,t))} \quad (6)$$

where f_i^* represents the distribution function after collisions and f_i^{eq} is found from f_i.

The second part is streaming

$$\mathrm{f_i(x + c_i\Delta t, t + \Delta t) = f_i^*(x,t)} \quad (7)$$

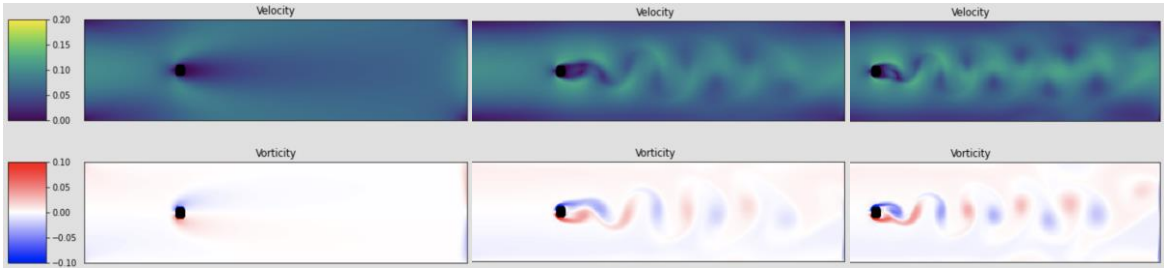
when we move computed lattice variables to the neighbor lattices.

Boundary conditions are handled before or after streaming with help of lattice entries reassignment with reverse values.

Because the collision is simply an algebraic local operation and the streaming is a data movement between arrays the whole algorithm computation can be implemented on devices with high parallelism. For that, different frameworks can be used: OpenMP, CUDA, etc. For our implementations we've stuck to TensorFlow. Generally TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications. But what is more important for our project is that this framework allows us to define an execution graph in Python, and later on compile it to the C/C++ executable with native CUDA support or to JavaScript with web browser compatibility. Thus we can reuse the same code base with various optimizations under the hood for different devices and use cases.

4. Experiments. To verify our implementation a few experiments with already known results were performed. We've tested flow around a fixed round cylinder with radius R = 0.0625 and the same object oscillating with amplitude equal to its radius and frequencies 0.05Hz, 0.1Hz and 0.2Hz. In addition other amplitudes were considered and CPU\GPU performance analysis was held.

4.1 Fluid flow around a circular cylinder. First, we simulated the flow of a viscous fluid around a stationary circular cylinder in a flat channel of size 4x1. Results with Reynolds numbers 10, 60 and 100 are displayed below (fig. 2).



a) Re=10 b)Re=60 c)Re=100

Fig 2. **Velocity and vorticity for various Reynolds numbers**

As shown in figure 2 the flow at Re = 10 is symmetric and there is a Karman vortex street at Reynolds numbers starting from Re = 60, the vortex frequency increases with the Reynolds number increasing. Such results correspond to other full-scale and numerical experiments [15, 16].

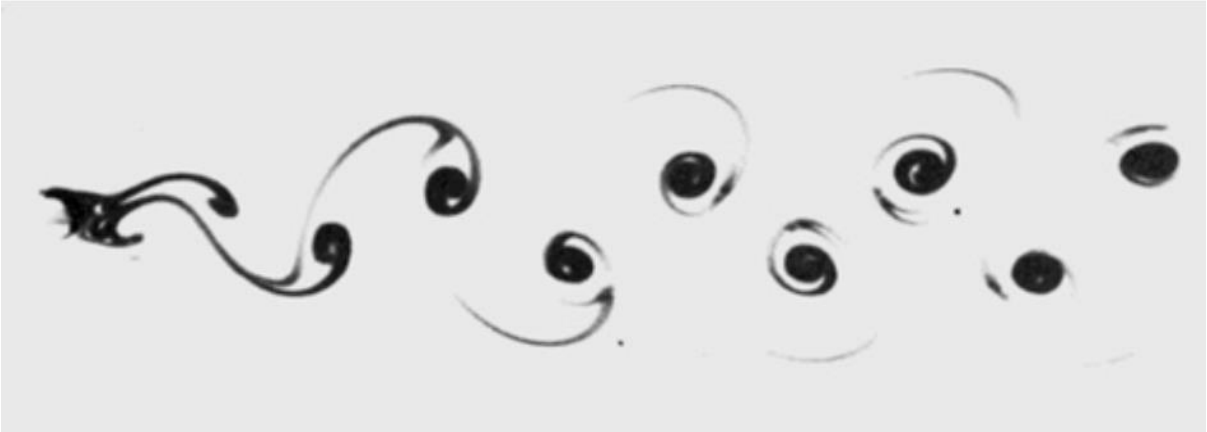
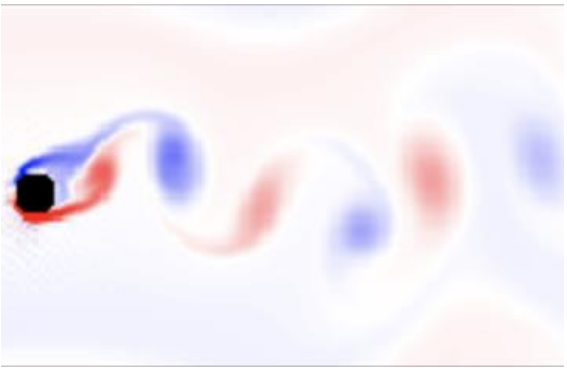
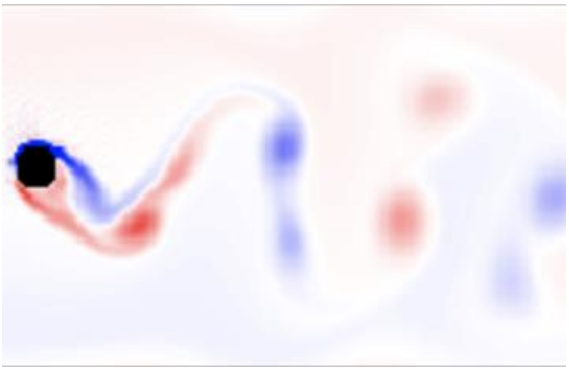


Fig. 3. **The Karman vortex street behind the circular cylinder at Re=105. Physical experiment[16]**

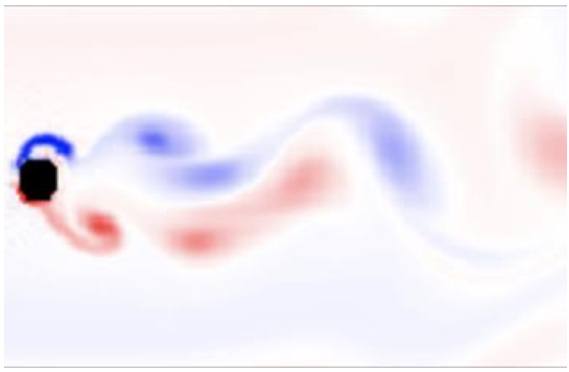
4.2 Flow around an oscillating circular cylinder. For this experiment we consider the same object oscillating by Y axis with periods 0.05Hz, 0.1Hz and 0.2Hz and amplitude equal to object radius 0.0625 (fig. 4). Reynolds number was fixed at 100.



f = 0.05Hz



f = 0.1Hz



f = 0.2Hz

Fig. 4. **Vorticity for round cylinder oscillating with various frequencies**

The simulation results showed that the vortex zones when flowing around a rigidly fixed cylinder have the same symmetric shape. Oscillations of the cylinder violate this structure: symmetry is broken when vortices form around the cylinder, and the diameter of the vortex zones increases. In the near part of the vortex wake, several vortex bunches are formed on both sides of the cylinder, but on a smaller scale compared to a stationary cylinder. Moreover, the scale of these formations decreases with an increase in the vibration frequency, and later these formations merge into structures of a larger scale. Similar results were received by Sheng Bao et al[12].

To extend our results experiments with larger amplitudes were considered as well (fig. 5, 6).

Fig. 5. **Vorticity by cylinder oscillating with A=0.1 and f=0.1Hz(left) 0.2Hz(right)**

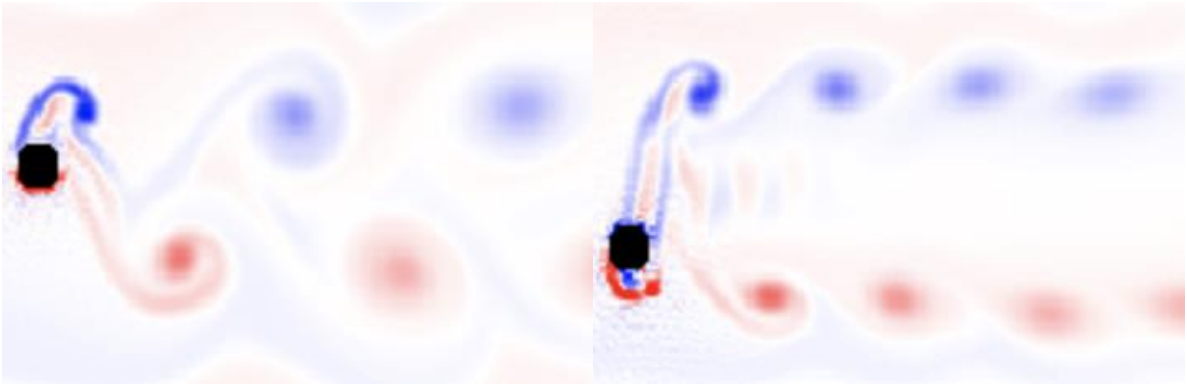


Fig. 6. **Vorticity by cylinder oscillating with A=0.2 and f=0.1Hz(left) 0.2Hz(right)**

4.3 Performance analysis. For performance analysis we run the experiment with the fixed object for a different domain scale. Evaluation was made on a server with Intel(R) Xeon(R) CPU E5-1630 v4 @ 3.70GHz CPU and GeForce GTX 1060 6GB GPU device.

Table 1

Evaluation of algorithm performance. Figures in seconds(less is better)

Device\Scale	50	100	150	200	250
CPU	31	133	339	938	1536
GPU	28	89	215	445	724

[author's development]

It was observed that while GPU vs CPU difference for small scales is subtle, with larger scales it becomes more crucial.

Conclusion. In this work it was shown that the lattice Boltzman method is well suited for simulation of different conditions. Implementation was verified based on well known experiments with fixed objects and different Reynolds numbers. Additional research was carried on for oscillating objects with various frequencies and amplitudes.

Moreover it was shown the algorithm can benefit a lot from the parallel GPU utilization with help of modern frameworks. Double speedup was achieved only by moving exactly the same code from CPU to GPU devices.

References:

[1] Shuling Hou, Qisu Zou, Shiyi Chen, Gary Doolen, Allen C. Cogley. (1995). Simulation of Cavity Flow by the Lattice Boltzmann Method. *Journal of Computational Physics*, (118/2), 329-347. https://doi.org/10.1006/jcph.1995.1103

[2] Chen, Shiyi and Doolen, Gary D. (1998). Lattice Boltzmann method for fluid flows. *Annual Review of Fluid Mechanics*, (30), 329-364. https://doi.org/10.1146/annurev.fluid.30.1.329

[3] Skordos, P. (1993). Initial and boundary conditions for the lattice Boltzmann method. *Phys. Rev. E*, (48), 4823–4842. 10.1103/PhysRevE.48.4823

[4] Keijo Mattila, Jari Hyväluoma, Jussi Timonen, & Tuomo Rossi (2008). Comparison of implementations of the lattice-Boltzmann method. *Computers & Mathematics with Applications*, (55/7), 1514-1524. https://doi.org/10.1016/j.camwa.2007.08.001

[5] Ye, Y., Li, K., Wang, Y., & deng, T. (2015). Parallel computation of Entropic Lattice Boltzmann method on hybrid CPU–GPU accelerated system. *Computers & Fluids*, (110). 10.1016/j.compfluid.2014.06.002

[6] Christian Obrecht, Frédéric Kuznik, Bernard Tourancheau, & Jean-Jacques Roux (2011). A new approach to the lattice Boltzmann method for graphics processing units. *Computers & Mathematics with Applications*, (61/12), 3628-3638.

[7] Marco-Felipe King, Amirul Khan, Nicolas Delbosc, Hannah L. Gough, Christos Halios, Janet F. Barlow, & Catherine J. Noakes (2017). Modelling urban airflow and natural ventilation using a GPU-based lattice-Boltzmann method. *Building and Environment*, (125), 273-284. https://doi.org/10.1016/j.buildenv.2017.08.048

[8] Johannes Habich and Christian Feichtinger and Harald Köstler and Georg Hager and Gerhard Wellein (2011). Performance engineering for the Lattice Boltzmann method on GPGPUs: Architectural requirements and performance results. *CoRR, abs*(1112.0850).

[9] Januszewski, M., & Kostur, M. (2014). Sailfish: A flexible multi-GPU implementation of the lattice Boltzmann method. *Computer Physics Communications*, (185/9), 2350–2368. 10.1016/j.cpc.2014.04.018

[10] Christian Obrecht, Frédéric Kuznik, Bernard Tourancheau, & Jean-Jacques Roux (2013). Multi-GPU implementation of the lattice Boltzmann method. *Computers & Mathematics with Applications*, (65/2), 252-261. https://doi.org/10.1016/j.camwa.2011.02.020

[11] Hurlbut, S., Spaulding, M., & White, F. (1982). Numerical Solution for Laminar Two Dimensional Flow About a Cylinder Oscillating in a Uniform Stream. *Journal of Fluids Engineering*, (104/2), 214-220. https://doi.org/10.1115/1.3241810

[12] Bao, S., Chen, S., Liu, Z., Li, J., Wang, H., & Zheng, C. (2012). Simulation of the flow around an upstream transversely oscillating cylinder and a stationary cylinder in tandem. *Physics of Fluids*, (24/2), 023603. https://doi.org/10.1063/1.3683565

[13] Koopmann, G. (1967). The vortex wakes of vibrating cylinders at low Reynolds numbers. *Journal of Fluid Mechanics*, (28/3), 501-512. doi:10.1017/S0022112067002253

[14] Krüger, T., Kusumaatmaja, H., Kuzmin, A., Shardt, O., Silva, G., & Viggen, E. (2016). *The Lattice Boltzmann Method - Principles and Practice*. Geneva: Springer International.

[15] Paul K. Chang (1973). *Separation of flow*. Oxford: Pergamon Press.

[16] He, X., Luo, LS. (1997). Lattice Boltzmann Model for the Incompressible Navier–Stokes Equation. *Journal of Statistical Physics*, (88), 927–944. https://doi.org/10.1023/B:JOSS.0000015179.12689.e4