

A Simplified GPU Implementation of the Hybrid Lattice Boltzmann Model for Three-Dimensional High Rayleigh Number Flows

Alexander Nee*

*Research and Educational Center of I.N. Butakov
National Research Tomsk Polytechnic University, Tomsk, Russia
nee_alexander@mail.ru*

Ali J. Chamkha

*Faculty of Engineering
Kuwait College of Science and Technology
Doha District, Kuwait*

Received 6 December 2022

Revised 23 March 2023

Accepted 25 March 2023

Published 29 April 2023

This paper provides an analysis of the numerical performance of a hybrid computational fluid dynamics (CFD) solver for 3D natural convection. We propose to use the lattice Boltzmann equations with the two-relaxation time approximation for the fluid flow, whereas thermodynamics is described by the macroscopic energy equation with the finite difference solution. An in-house parallel graphics processing unit (GPU) code is written in MATLAB. The execution time of every single step of the algorithm is studied. It is found that the explicit finite difference scheme is not as stable as the implicit one for high Rayleigh numbers. The most time-consuming steps are energy and collide, while stream, boundary conditions, and macroscopic parameters recovery are executed in no time, despite the grid size under consideration. GPU code is more than 30 times faster than a typical low-end central processing unit-based code. The proposed hybrid model can be used for real-time simulation of physical systems under laminar flow behavior and on mid-range segment GPUs.

Keywords: GPU computing; LBM; natural convection; two-relaxation time; hybrid solver; FDM.

1. Introduction

Rayleigh–Benard convection or the so-called buoyancy-driven flows are extensively studied over the past two decades. Various phenomena, such as double diffusion, porosity and magnetism, affecting the natural convection along with benchmark solutions are examined in order to better understand heat transfer and fluid flow

*Corresponding author.

patterns. When performing three-dimensional (3D) simulations under turbulent flow behavior, the problem associated with the code execution time is encountered. In other words, modeling of turbulent buoyancy-driven flows is accompanied by massive computations since the number of governing equations or mesh points is high. To overcome this problem, parallel implementation on central processing units (CPUs) or graphics processing units (GPUs) is used. However, the numerical procedure for parallel algorithms is a very difficult task. Hence, multithread CFD computing is still a great challenge for many researchers. The relevant works in the field of parallel computations of 3D thermal flows are discussed below.

One of the earliest attempts to perform parallel CPU implementation of 3D natural convection was made in 1991. Schafer [1991] introduced finite difference algorithms suitable for cluster working in Single Instruction Multiple Data mode. Numerical tests were conducted on a grid of $32 \times 32 \times 16$ points with the Rayleigh number of 4000. Rathish Kumar and Kumar [2004] developed a finite element-based parallel solver for two-dimensional (2D) natural convection in trapezoidal cavities filled with a porous medium. The eight-noded Anupam cluster with the Anulib message passing libraries was used. Xu and Chen [2020] proposed a parallel cell-centered finite volume solver for 2D cavities with unstructured grids. Heat transfer and fluid flow were computed by an alternative CFD tool called the thermal lattice Boltzmann method (LBM). To perform the partition of the grid, the MPI-based parallel library was used. Simulations were conducted on the Tianhe-2A supercomputer with a grid size of up to 5 billion.

Wen *et al.* [2021] developed a 3D CFD solver based on a discrete unified gas-kinetic scheme (DUGKS) to study transition from laminar to turbulent flow inside the boundary layer. To perform parallel implementation, domain decomposition and MPI were utilized. To conduct direct numerical simulation of Rayleigh–Benard convection, Yilmaz [2021] created an in-house 3D parallel code. The algorithm builds on a finite volume implicit, nondissipative, discrete kinetic energy conserving schemes with PETSc parallel library.

In order to study the rising of a 3D spherical catalyst in a cavity due to free convection, Wachs [2011] developed the PeliGRIFF code within the framework of the parallel open source platform PELICANS. The author stressed that the code demonstrated impressive, but not optimal, scalability. 2D and 3D natural convective melting with parallel code is discussed in Sadaka *et al.* [2020]. Sadaka *et al.* [2020] presented a free finite-element software called FreeFem++. This software supports scalable Schwarz domain decomposition methods and a GMRES Krylov method with a Restricted Additive Schwarz (RAS) preconditioner.

The above studies deal with parallel implementation on CPU. On the other hand, GPU provides higher performance at the same cost. It should be stressed that GPU acceleration is predominantly used for the lattice Boltzmann method since this CFD technique is well suited for massively parallel computing. Obrecht *et al.* [2012, 2013] built the multiple relaxation time (MRT) lattice Boltzmann scheme coupled

with the finite difference solution of energy equation within the TheLMA project. The authors utilized the compute unified device architecture (CUDA) to perform parallel computing of 3D natural convection in a closed cube on GPU. The variation of $10^4 \leq Ra \leq 10^9$ was examined, and the grid size of up $3.2 \cdot 10^8$ nodes was used.

Ren and Chan performed a series of GPU computing of various physical phenomena such as melting [Ren and Chan, 2016a], double diffusion [Ren and Chan, 2016b] and natural convection inside the cavity with heat-conducting obstacles [Ren and Chan, 2016c]. The authors formulated 2D lattice Boltzmann model with a passive scalar approach to recover thermal fields. An in-house CUDA code was developed in FORTRAN. The Rayleigh range of $10^3 \leq Ra \leq 10^7$ corresponding to the laminar flow behavior was analyzed.

Heat transfer and fluid flow in a porous medium during melting are studied in Luo *et al.* [2020] and Noyola-García and Rodriguez-Romo [2021]. Luo *et al.* [2020] implemented an enthalpy-based 2D MRT LBM parallel GPU code for paraffin melting in FORTRAN, whereas Noyola-García and Rodriguez-Romo [2021] used CUDA technology in Python to analyze gallium melting in a non-Darcy porous medium. The parallel CUDA GPU lattice Boltzmann scheme combined with a total variation diminishing technique is proposed by Wang *et al.* [2019] to study 2D natural convection of nanofluids.

Parallel implementation of Navier–Stokes equations on GPU is discussed in Fu *et al.* [2012, 2013] and Jacobsen and Senocak [2013]. In order to solve the governing equations for compressible flow, Fu *et al.* [2012, 2013] applied the Roe scheme and preconditioning method. The grid of $250 \times 160 \times 10$ nodes and a range of $1.58 \cdot 10^6 \leq Ra \leq 1.95 \cdot 10^7$ were used to study 3D natural convection between horizontal parallel plates with bottom heating. On the other hand, Jacobsen and Senocak [2013] proposed the multi-level parallel schemes for incompressible flows. Computations were performed on GPU clusters within the MPI-CUDA and hybrid MPI-OpenMP-CUDA standards. Both forced and natural convection benchmark problems were considered.

To summarize the above, it can be concluded that parallel algorithms are both implemented for 2D and 3D CFD problems. MPI or CUDA is adopted for CPU and GPU computations. It has been found that the buoyancy-driven flows with $Ra \geq 10^{10}$ are discussed in a few works. Along with that, it needs to be stressed that numerical solution of governing equations is itself a difficult task. Moreover, enormous efforts must be undertaken in terms of coding when implementing parallel computing in CUDA or MPI.

This study aims to introduce a CFD solver for transient 3D low and high Rayleigh number flows with a simple and robust parallel implementation. The governing equations are formulated in terms of the mesoscopic lattice Boltzmann equation and macroscopic energy equation. As far as the authors know, the MRT approximation is widely used in lattice Boltzmann schemes. However, it was previously found that the two relaxation time (TRT) model is as stable as the MRT. On

the contrary, the MRT model is computationally heavy and more complex than the TRT. Hence, on the one hand, numerical implementation of the TRT is far easier than the MRT. Moreover, computational performance of the TRT is higher than MRT. On the other hand, the TRT model has the same numerical stability as the MRT. Thus, the novelty of the work lies in implementing the TRT approximation of collision operator instead of the widespread single and MRT models.

2. Model Description

A benchmark problem of incompressible low-Mach number buoyancy-driven flow inside a closed parallelepiped is considered. Two opposite vertical walls are kept at a constant temperature, whereas the other boundaries are heat-insulated. We assume that the fluid is Newtonian and transparent for thermal radiation. Along with that, viscous energy dissipation and surface radiation are negligibly small. The Boussinesq approximation and temperature-independent physical properties are implemented in the current model.

Fluid flow can be mathematically described in terms of the macroscopic Navier-Stokes equations. The governing equations are as follows:

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} + w \frac{\partial u}{\partial z} = -\frac{1}{\rho} \frac{\partial p}{\partial x} + \nu \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \right), \quad (1)$$

$$\frac{\partial v}{\partial t} + u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} + w \frac{\partial v}{\partial z} = -\frac{1}{\rho} \frac{\partial p}{\partial y} + \nu \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial z^2} \right), \quad (2)$$

$$\begin{aligned} \frac{\partial w}{\partial t} + u \frac{\partial w}{\partial x} + v \frac{\partial w}{\partial y} + w \frac{\partial w}{\partial z} \\ = -\frac{1}{\rho} \frac{\partial p}{\partial z} + \nu \left(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} + \frac{\partial^2 w}{\partial z^2} \right) + g \cdot \beta \cdot (T - T_0), \end{aligned} \quad (3)$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0, \quad (4)$$

where u , v , w are velocity components; x , y , z are Cartesian coordinates, p is pressure, ρ is density; ν is kinematic viscosity; g is gravity; β is thermal expansion; T is temperature.

On the contrary, the lattice Boltzmann equation can be used to describe flow behavior too. In order to avoid numerical instabilities with high Rayleigh numbers, a simple but efficient TRT approximation of the collision operator is implemented. A distribution function is split into two parts known as symmetric and antisymmetric under the TRT formulation. In this case, the symmetrical relaxation time is related to viscosity, whereas asymmetrical relaxation time is related to energy flux. The governing equation in discretized form is as follows [Ginzburg, 2008]:

$$f_k(\vec{x} + \vec{c}_k \cdot \Delta t, t + \Delta t)$$

$$\begin{aligned}
&= f_k(\vec{x}, t) - \frac{\Delta t}{\tau_{\text{sym}}} \cdot (f_k^{\text{sym}} - f_k^{\text{seq}}) \\
&\quad - \frac{\Delta t}{\tau_{\text{asym}}} \cdot (f_k^{\text{asym}} - f_k^{\text{aeq}}) + \Delta t \cdot c_k \cdot F_k.
\end{aligned} \tag{5}$$

The following relations are used to compute symmetrical (τ_{sym}) and asymmetrical (τ_{asym}) relaxation times, symmetrical (f_k^{sym}) and asymmetrical (f_k^{asym}) distribution functions, symmetrical (f_k^{seq}) and asymmetrical (f_k^{aeq}) equilibrium distribution functions, force (F_k), density and velocity:

$$\tau_{\text{sym}} = \frac{\nu}{c_s^2 \cdot \Delta t} + \frac{1}{2}, \quad \tau_{\text{asym}} = \frac{0.5 \cdot \tau_{\text{sym}}}{\tau_{\text{sym}} - 0.5}, \tag{6}$$

$$f_k^{\text{sym}} = \frac{f_k + f_{-k}}{2}, \quad f_k^{\text{seq}} = \frac{f_k^{\text{eq}} + f_{-k}^{\text{eq}}}{2}, \quad f_k^{\text{asym}} = \frac{f_k - f_{-k}}{2}, \quad f_k^{\text{aeq}} = \frac{f_k^{\text{eq}} - f_{-k}^{\text{eq}}}{2}, \tag{7}$$

$$f_k^{\text{eq}} = w_k \cdot \rho \cdot \left[1 + \frac{c_k \cdot \vec{u}}{c_s^2} + \frac{(c_k \cdot \vec{u})^2}{2 \cdot c_s^4} - \frac{\vec{u}^2}{2 \cdot c_s^2} \right], \tag{8}$$

$$F_k = w_k \cdot \frac{F}{c_s^2}, \quad F = \rho \cdot g \cdot \beta \cdot (T - T_0), \tag{9}$$

$$\rho = \sum_{k=1}^{19} f_k, \quad \vec{u} = \frac{1}{\rho} \cdot \sum_{k=1}^{19} c_k \cdot f_k, \tag{10}$$

where w_k is weights for D3Q19 scheme; c_k is particle speeds; c_s is lattice speed of sound.

The subscript “ $-k$ ” represents the counter direction of the subscript “ k ” (Fig. 1) in the above equations. For example, if $k = 2$, the counter direction is $-k = 3$.

In order to obtain a better performance in terms of numerical stability, mesoscopic lattice Boltzmann equation is coupled with macroscopic energy equation. Lallemand and Luo [2003] stated that this hybrid coupling [Bettaibi *et al.*, 2021; Mhamdi *et al.*, 2022] yields higher numerical stability. The time-dependent energy equation under the assumptions made above is as follows:

$$\frac{\partial T}{\partial t} + u \frac{\partial T}{\partial x} + v \frac{\partial T}{\partial y} + w \frac{\partial T}{\partial z} = a \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2} \right), \tag{11}$$

where a is thermal diffusivity.

2.1. Numerical procedure

The governing equations are consecutively solved. To run the code execution, first of all, initial data, such as temperature, distribution function, velocity and density, symmetrical and asymmetrical relaxation times, should be specified. The time or

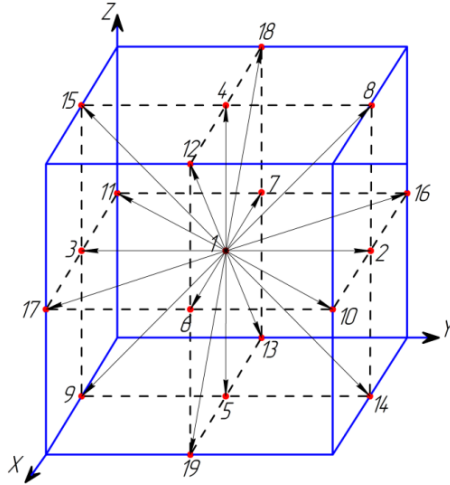


Fig. 1. Particle directions in D3Q19 scheme.

iteration loop begins with the energy equation solution. When considering parallel algorithms, the governing equations should be discretized by explicit schemes. Hence, the temporal, convective and diffusive terms in Eq. (7) are approximated by explicit Euler scheme, upwind scheme and central differences, respectively. After the solution of energy equation, collision process takes place in accordance with Eq. (1). Along with that, the symmetrical, asymmetrical and equilibrium distribution functions are calculated by means of the relations (3)–(5). After the collision step, the streaming step is according to the particle direction scheme (Fig. 1). Hereafter, a simple bounce-back condition is implemented to find the distribution function at the solid walls. Finally, macroscopic density and velocity are reconstructed from the mesoscopic distribution function in accordance with Eq. (6).

2.2. Parallel implementation

A source code is written in MATLAB. In order to perform high-performance CPU or GPU computations, the parallel computing toolbox (PCT) must be installed. The great benefit of MATLAB PCT is its simplicity in implementation. You need almost no code modification and special preparations for GPU computing as in the case of CUDA or C languages. Preliminarily, parallel GPU computations can be performed with only one command “gpuArray”. This command creates an array stored on a GPU and allows one to do direct calculations on GPUs. To accelerate code execution, the “arrayfun” function can be implemented for the GPU array. This function is applicable to the vectorized code without indices. It should be stressed that CUDA kernels can be executed in MATLAB too. However, this procedure requires two additional scripts in CUDA and C.

In this work, the GTX 1060 ARMOR with 6 GB of GDDR5 RAM is used for numerical experiments. It needs to be noted that the source code is not fully optimized since the task is to examine GPU performance with minimum efforts in terms of coding. But still, since the collision step is computationally heavy, this part of the code is fully vectorized and the “arrayfun” function is applied to this step. The maximum grid resolution is limited by the amount of RAM. In our case, the highest resolution is 17,779,581 grid points. All computations were performed on a single PC.

3. Numerical Results and Analysis

To make the analysis more convenient, the results of the study were nondimensionalized with the following relations:

$$X = \frac{x}{L}; \quad Y = \frac{y}{L}; \quad Z = \frac{z}{L}; \quad \tau = \frac{t}{t_0}; \quad U = \frac{u}{V_{nc}}; \quad W = \frac{w}{V_{nc}}; \quad \Theta = \frac{T - T_c}{T_h - T_c};$$

$$V_{nc} = \sqrt{g \cdot \beta \cdot (T_h - T_c) \cdot L}; \quad t_0 = \frac{L}{V_{nc}}; \quad Ra = \frac{g \cdot \beta \cdot (T_h - T_c) \cdot L^3}{\nu \cdot a}; \quad Pr = \frac{\nu}{a};$$

$$Nu = \int_0^1 \int_0^1 \left. \frac{\partial \Theta}{\partial Y} \right|_{Y=0} dX \cdot dZ.$$

When performing the analysis, attention was focused on the robustness of the model and computational performance. An air-filled closed cube is considered. Both laminar and turbulent flow behaviors are examined in a range of $10^4 \leq Ra \leq 10^9$ corresponds to a range of domain characteristic sizes of $0.017 \leq L \leq 0.81$ m. The thermal and flow fields are not presented in the main text since the heat transfer and the fluid dynamics patterns in differentially heated cavities have already been well-studied. However, the temperature of iso-surfaces and the velocity magnitudes (Fig. B.1) are given in Appendix B for validation purposes of other works.

3.1. Validation

Table 1 presents the mean Nusselt numbers at the hot vertical wall under laminar and turbulent flow behavior.

When analyzing the data presented in Table 1, it is found that the proposed hybrid TRTLB-FD model reproduces well the values of the Nusselt numbers in a studied range of the Rayleigh number. It needs to be mentioned that no grid refinement technique is implemented in the proposed hybrid model. Nevertheless, good accuracy in terms of the integral analysis is demonstrated. It should be stressed that although the number of grid points might be decreased when performing a mesh refinement procedure, this technique introduces additional complexity in numerical implementation. Additionally, numerical viscosity might also be increased since the nonuniform grid spacing requires an interpolation procedure in case of LBM.

Table 1. Variation of mean Nusselt numbers.

Ra	Fusegi <i>et al.</i> [1991]		Trias <i>et al.</i> [2010]		Wang <i>et al.</i> [2017]		This study	
	Grid	Nu	Grid	Nu	Grid	Nu	Grid	Nu
10^4	$62 \times 62 \times 62$	2.189*	—	—	$50 \times 50 \times 50$	2.248	$51 \times 51 \times 51$	2.08
10^5	$62 \times 62 \times 62$	4.19*	—	—	$50 \times 50 \times 50$	4.6	$51 \times 51 \times 51$	4.28
10^6	$62 \times 62 \times 62$	8.02*	—	—	$50 \times 50 \times 50$	8.78	$101 \times 101 \times 101$	8.63
10^7	—	15.353*	—	—	$100 \times 100 \times 100$	16.415	$101 \times 101 \times 101$	16.26
10^8	$62 \times 62 \times 62$	29.39*	—	—	$200 \times 200 \times 200$	29.917	$151 \times 151 \times 151$	29.57
10^9	—	56.258*	—	54.334*	$200 \times 200 \times 200$	52.3	$151 \times 151 \times 151$	49.92

* Note that Nuselt numbers were computed using Nu-Ra correlations.

During the validation, some important patterns are revealed in terms of the finite difference solution of the energy equation. It needs to be stressed before the main discussion that, previously, it was demonstrated that the 2D lattice Boltzmann model coupled with the implicit scheme for the energy equation was capable to simulate very high Rayleigh number flows up to 10^{11} . So, it was found that the 3D model was not as numerically stable as the 2D model with $Ra > 10^9$. The first thought was to extend the number of particle directions. Hence, the D3Q27 scheme was implemented. However, the 3D hybrid model was still numerically unstable with $Ra = 10^{10}$, whereas the 2D model with the D2Q9 scheme worked fine with this value of buoyancy force. The next step was to analyze how the convective terms approximation affected the numerical stability. The forward time and central space discretization, the upwind scheme and the monotonic Samarskii approximation were implemented. These ways of discretization led to unstable behavior with a low grid resolution with $Ra = 10^9$ and with a maximum grid resolution with $Ra = 10^{10}$. As a matter of interest, when the parallelizable explicit scheme was swapped to the nonparallelizable three-layer implicit scheme, the model became stable even with the relatively coarse grid of 151^3 points with $Ra = 10^9$. The mean Nusselt number was around 52 in the case of the implicit scheme. This issue needs to be further studied. Perhaps, the energy-conservative formulation of the hybrid model will improve the numerical stability. We will find out this in our future work.

3.2. Performance

In this section, we evaluate the numerical performance in terms of the execution time of every single step of the code. Conventionally, flops are given to estimate numerical performance. However, these data might be hard to understand for a common reader. Hence, it is more convenient to operate with time. Table 2 presents execution times in seconds when varying the Rayleigh number. The mesh size corresponds to the grid-independent solution (see Table 1). The convergence criterion was set to 10^{-5} . The execution times for energy, collide, stream, boundary conditions (BC) and macroscopic parameters recovery (MPR) are given for 1 iteration. The code was executed several times. Deviation in steps and total time was around 10% for both GPU and CPU computing.

Table 2. Execution times in seconds for every solution step within the hybrid lattice Boltzmann method.

Ra	Energy (s)	Collide (s)	Stream (s)	BC (s)	MPR (s)	Iterations	Total time (s)	
							With arrayfun	Without arrayfun
10^4	0.03	0.16	0.045	0.06	0.02	1956	35.75	58.27
10^5	0.06	0.2	0.045	0.06	0.02	2653	56.77	83.82
10^6	0.09	0.19	0.05	0.06	0.02	4267	111.03	141.79
10^7	0.09	0.19	0.05	0.06	0.02	7860	204.75	262.1
10^8	0.24	0.26	0.04	0.06	0.02	17,568	880	1424.12
10^9	0.25	0.3	0.04	0.06	0.02	100,000	5735.11	9557.34

As seen from Table 2, the computation speed of the stream, BC and MPR steps is almost insensible to the grid size variation. These steps are instantly executed. On the other hand, the most time-consuming steps are energy and collide as could be predicted. It is clearly seen that the difference in execution time between 3,442,951 and 4,090,601 mesh points is insignificant. It was found that “arrayfun” function gave speedup even with a low grid resolution of 51^3 . However, the implementation of this function is more reasonable with a high number of iterations since the difference in total time, for example, in the case of $Ra = 10^7$ is only around one minute. On the contrary, code execution without “arrayfun” takes more than 1 h longer when $Ra = 10^9$. It should be stressed that if energy and collide steps would be optimized, the real-time simulation could be performed. Probably, manual domain decomposition by means of distributed arrays will improve computational performance. But of course, this procedure will make the code more complicated. To make it clear, why we should apply GPU for computation instead of CPU, execution times are listed in Table 3 when using the CPUs.

First, an important remark should be done that CPU code is absolutely identical to GPU code. In other words, our code can be executed both on CPU and GPU almost without any changes. In the case of CPU computing, GPU arrays creation should be missed. MATLAB automatically creates arrays for CPU. According to Table 3, the Intel Core i3-8100 demonstrates the best numerical performance, whereas the lowest is observed in the case of Intel Celeron G3930. These results are predictable and directly concerned with the CPU frequency. Along with that, a number of physical cores had no effect on execution time. Speedup is almost proportional to the growth in frequency. When comparing the GPU and CPU times, the speedup is increased with the grid resolution. GTX 1060 is around 27 times faster than Intel Core i3-8100 when $Ra = 10^7$ while the cost of this hardware is close.

Table 3. Variation of CPU times with the Rayleigh number.

Ra	Intel Celeron G3930, 2.9 GHz	Intel Core i5-4440, 3.3 GHz	Intel Core i3-8100, 3.6 GHz
10^4	179.46	125.83	92.97
10^5	785.47	698.61	607.11
10^6	3610.03	3263.87	2950.35
10^7	6666.83	5710.27	5543.18

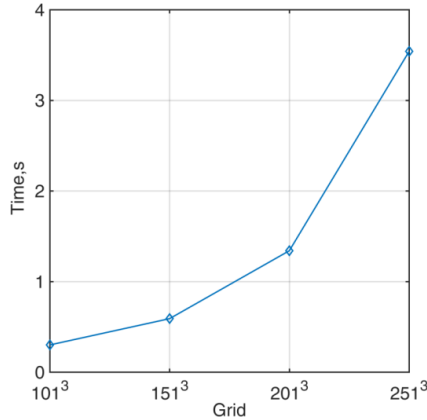


Fig. 2. Time variation with the grid size.

A more detailed examination of the grid size effect on execution time is shown in Fig. 2. The data are presented for 10 iterations when the Rayleigh number is equal to 10^9 .

As seen from Fig. 2, 10 iterations are executed in less than 1 s when the grid resolution is lower than 151^3 . Hence, the real-time simulation is possible with this mesh size. The hybrid solver proposed in this study can accurately predict laminar fluid flow and heat transfer patterns with the grid size of 151^3 . Thus, the thermal and flow fields in engineering applications such as microelectronics cooling can be observed in real-time. To extend the capability of the proposed model to study turbulent flow behavior in real-time, the Smagorinsky sub-grid scale model can be implemented or a more high performance GPU is used. It is found that when the mesh resolution is increased by around 16 times, the execution time is raised by around 12 times.

4. Summary

In this work, a hybrid GPU solver based on the nontypical TRT lattice Boltzmann scheme coupled with an explicit finite difference solution of the energy equation is introduced. The great benefit of the CFD approach proposed in this study is a simple GPU implementation. Moreover, the TRT model is more numerically stable than the single relaxation time approximation, whereas the numerical procedure is not as complicated as in the case of the MRT scheme.

It needs to be noted that the code can be executed on the CPU too without almost any changes. An impressive speedup in terms of the execution time was demonstrated when comparing GPU and CPU computations. In particular, the computations on GPU NVIDIA GTX 1060 are around 27 times faster than on CPU Intel Core i3-8100 when $Ra = 10^7$.

Despite the lower grid resolution with high Rayleigh numbers, the hybrid model reproduces similar results of integral analysis even without the grid refinement technique. The mean Nusselt numbers computed by the proposed hybrid model were

well matched with the data generated by the discrete unified gas kinetic scheme in a range of $10^4 \leq Ra \leq 10^8$. Moreover, the relative error is lower than 5% when $Ra = 10^9$.

It was found that the running time for 10 iterations is less than one real second when the grid resolution is lower than 151^3 . Hence, the real-time simulation of laminar flow behavior is possible with this mesh size. For example, the thermal and flow fields in engineering applications such as microelectronics cooling can be observed in real-time.

Further discussions will cover 3D buoyancy-driven flow with double diffusion when taking into account a radiatively participating medium.

Appendix A. GPU Commands for MATLAB

```

%% GPU array creation with single precision
T = (single(zeros(Nx,Ny,Nz,'gpuArray')));
%% GPU array creation with double precision
T = zeros(Nx,Ny,Nz,'gpuArray');
%% Return array from GPU to CPU
T = gather(T);
%% "arrayfun" command is applicable to functions where vectorized code without indices is stored
[f] = arrayfun(@collide, f, U, V, W, rho, T);

```

Appendix B. Thermal and Velocity Magnitude Fields

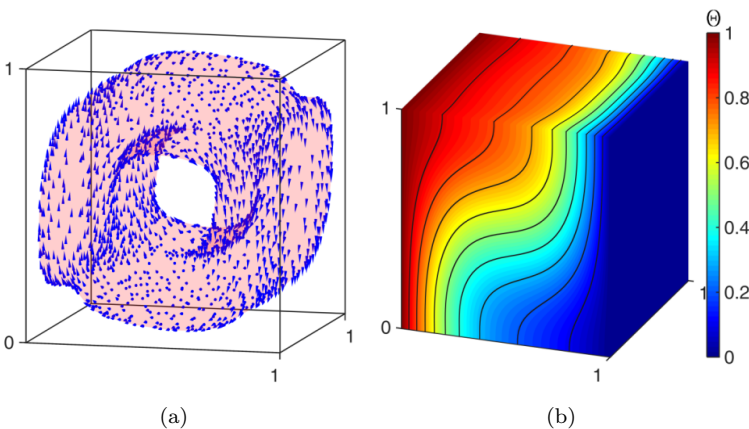


Fig. B.1. Steady-state velocity magnitude (a, c, e, g, i, k) and temperature fields (b, d, f, h, j, l) when $Pr = 0.7$: (a, b) $Ra = 10^4$; (c, d) $Ra = 10^5$; (e, f) $Ra = 10^6$; (g, h) $Ra = 10^7$; (i, j) $Ra = 10^8$; (k, l) $Ra = 10^9$.

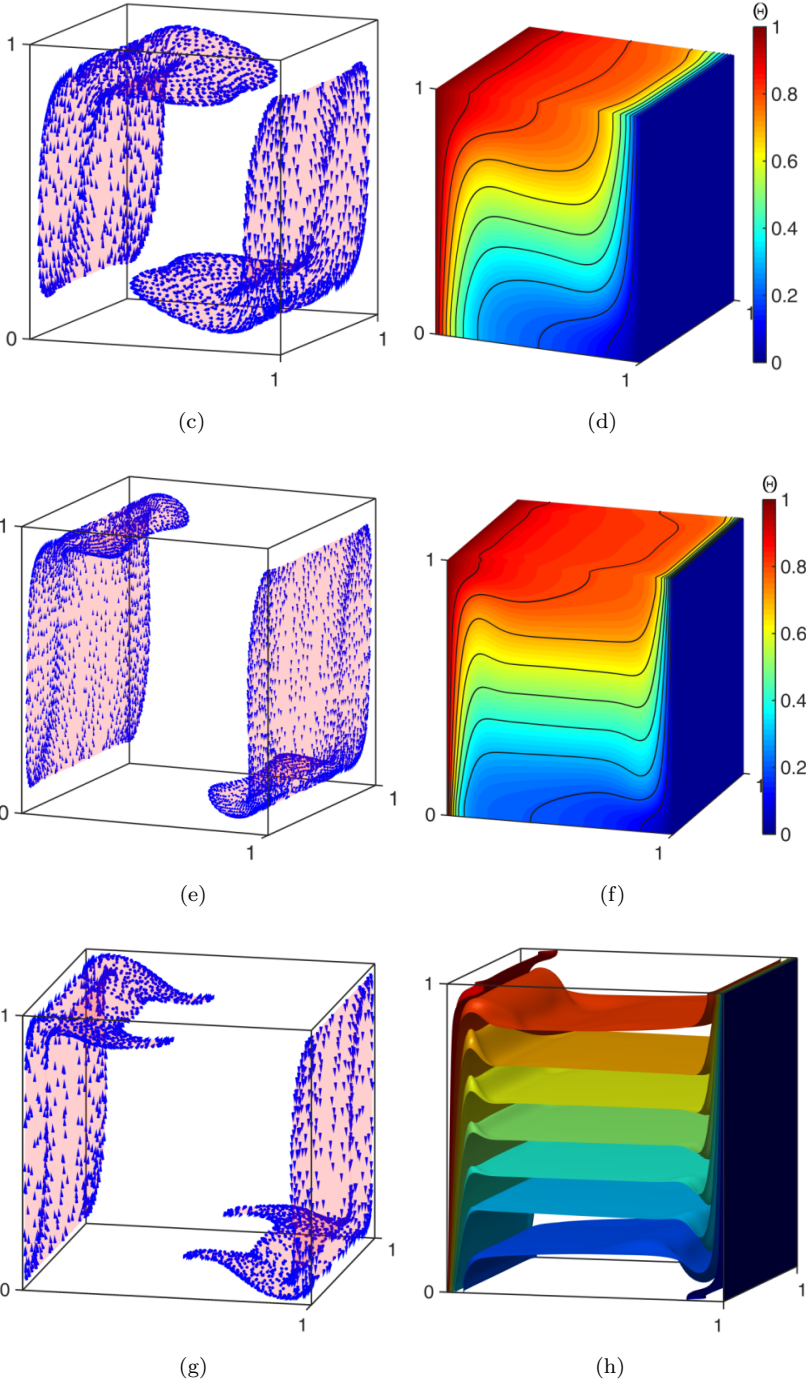


Fig. B.1. (Continued)

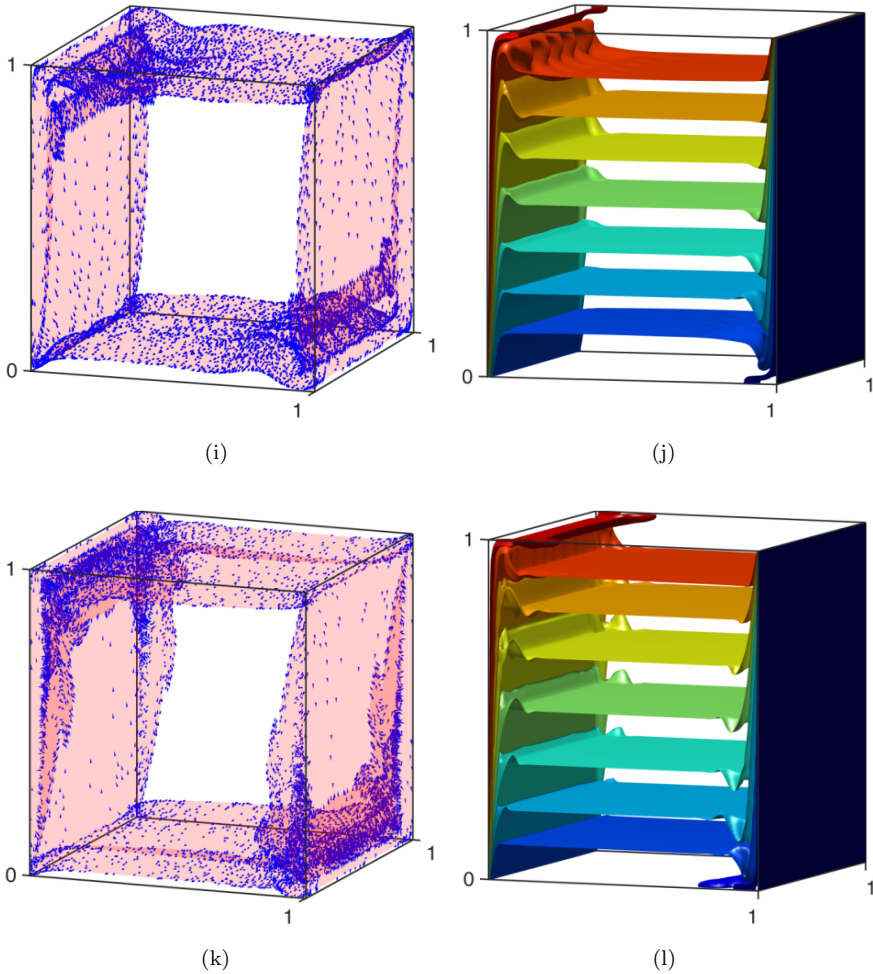


Fig. B.1. (Continued)

References

- Bettaibi, S., Kuznik, F., Sediki, E. and Succi, S. [2021] “Numerical study of thermal diffusion and diffusion thermo effects in a differentially heated and salted driven cavity using MRT-lattice Boltzmann finite difference model,” *International Journal of Applied Mechanics* **13**, 2150049.
- Fu, W.-S., Wang, W.-H. and Huang, S.-H. [2012] “An investigation of natural convection of three dimensional horizontal parallel plates from a steady to an unsteady situation by a CUDA computation platform,” *International Journal of Heat and Mass Transfer* **55**, 4638–4650.
- Fu, W.-S., Wang, W.-H. and Huang, S.-H. [2013] “An investigation of effects of heights and a length on natural convection of three dimensional parallel plates with a heated bottom surface by a CUDA computation platform,” *International Journal of Heat and Mass Transfer* **67**, 118–130.

- Fusegi, T., Hyun, J. M. and Kuwahara, K. [1991] “Three-dimensional simulations of natural convection in a sidewall-heated cube,” *International Journal for Numerical Methods in Fluids* **13**, 857–867.
- Ginzburg, I., Verhaeghe, F. and d’Humières, D. [2008] “Two relaxation-time lattice Boltzmann scheme: About parameterization, velocity, pressure and mixed boundary conditions,” *Communications in Computational Physics* **3**, 427–478.
- Jacobsen, D. A. and Senocak, I. [2013] “Multi-level parallelism for incompressible flow computations on GPU clusters,” *Parallel Computing* **39**, 1–20.
- Lallemand, P. and Lou, L.-S. [2003] “Hybrid finite-difference thermal lattice Boltzmann equation,” *International Journal of Modern Physics B* **17**, 41–47.
- Luo, Z., Xu, H., Lou, Q., Feng, L. and Ni, J. [2020] “GPU-accelerated lattice Boltzmann simulation of heat transfer characteristics of porous brick roof filled with phase change materials,” *International Communications in Heat and Mass Transfer* **119**, 104911.
- Mhamdi, B., Bettaibi, S., Jellouli, O. and Chafra, M. [2022] “MRT-lattice Boltzmann hybrid model for the double diffusive mixed convection with thermodiffusion effect,” *Natural Computing* **21**, 393–405.
- Noyola-García, B. S. and Rodríguez-Romo, S. [2021] “Simulations of Ga melting based on multiple-relaxation time lattice Boltzmann method performed with CUDA in Python,” *Mathematics and Computers in Simulation* **181**, 170–191.
- Obrecht, C., Kuznik, F., Tourancheau, B. and Roux, J.-J. [2012] “The TheLMA project: A thermal lattice Boltzmann solver for the GPU,” *Computers & Fluids* **54**, 118–126.
- Obrecht, C., Kuznik, F., Tourancheau, B. and Roux, J.-J. [2013] “Multi-GPU implementation of a hybrid thermal lattice Boltzmann solver using the TheLMA framework,” *Computers & Fluids* **80**, 269–275.
- Rathish Kumar, B. V. and Kumar, B. [2004] “Parallel computation of natural convection in trapezoidal porous enclosures,” *Mathematics and Computers in Simulation* **65**, 221–229.
- Ren, Q. and Chan, C. L. [2016a] “GPU accelerated numerical study of PCM melting process in an enclosure with internal fins using lattice Boltzmann method,” *International Journal of Heat and Mass Transfer* **100**, 522–535.
- Ren, Q. and Chan, C. L. [2016b] “Numerical study of double-diffusive convection in a vertical cavity with Soret and Dufour effects by lattice Boltzmann method on GPU,” *International Journal of Heat and Mass Transfer* **93**, 538–553.
- Ren, Q. and Chan, C. L. [2016c] “Natural convection with an array of solid obstacles in an enclosure by lattice Boltzmann method on a CUDA computation platform,” *International Journal of Heat and Mass Transfer* **93**, 273–285.
- Sadaka, G., Rakotondrandisa, A., Tournier, P.-H., Luddens, F., Lothodé, C. and Danaila, I. [2020] “Parallel finite-element codes for the simulation of two-dimensional and three-dimensional solid–liquid phase-change systems with natural convection,” *Computer Physics Communications* **257**, 107492.
- Schafer, M. [1991] “Parallel algorithms for the numerical simulation of three-dimensional natural convection,” *Applied Numerical Mathematics* **7**, 347–365.
- Trias, F. X., Gorobets, A., Soria, M. and Oliva, A. [2010] “Direct numerical simulation of a differentially heated cavity of aspect ratio 4 with Rayleigh numbers up to 10^{11} —Part I: Numerical methods and time-averaged flow,” *International Journal of Heat and Mass Transfer* **53**, 665–673.
- Wachs, A. [2011] “Rising of 3D catalyst particles in a natural convection dominated flow by a parallel DNS method,” *Computers & Chemical Engineering* **35**, 2169–2185.

- Wang, P., Zhang, Y. and Guo, Z. [2017] “Numerical study of three-dimensional natural convection in a cubical cavity at high Rayleigh numbers,” *International Journal of Heat and Mass Transfer* **113**, 217–228.
- Wang, L., Yang, X., Huang, C., Chai, Z. and Shi, B. [2019] “Hybrid lattice Boltzmann-TVD simulation of natural convection of nanofluids in a partially heated square cavity using Buongiorno’s model,” *Applied Thermal Engineering* **146**, 318–327.
- Wen, X., Wang, L.-P., Guo, Z. and Zhakebayev, D. B. [2021] “Laminar to turbulent flow transition inside the boundary layer adjacent to isothermal wall of natural convection flow in a cubical cavity,” *International Journal of Heat and Mass Transfer* **167**, 120822.
- Xu, L. and Chen, R. [2020] “Scalable parallel finite volume lattice Boltzmann method for thermal incompressible flows on unstructured grids,” *International Journal of Heat and Mass Transfer* **160**, 120156.
- Yilmaz, I. [2021] “Parallel direct numerical simulation and analysis of turbulent Rayleigh–Bénard convection at moderate Rayleigh numbers using an efficient algorithm,” *Computers & Fluids* **213**, 104754.