Appl. Math. Mech. -Engl. Ed.
DOI 10.1007/s10483-017-2194-9

Applied Mathematics and Mechanics (English Edition)

Multi-relaxation-time lattice Boltzmann simulations of lid driven flows using graphics processing unit*

Chenggong LI[1,†], J. P. Y. MAA[2]

1. National Engineering Laboratory for Methanol to Olefins, Dalian National Laboratory for
Clean Energy, Dalian Institute of Chemical Physics, Chinese Academy
of Sciences, Dalian 116023, Liaoning Province, China;

2. College of William and Mary, Virginia Institute of Marine Science,
Virgina 23062, U. S. A.

Abstract Large eddy simulation (LES) using the Smagorinsky eddy viscosity model is added to the two-dimensional nine velocity components (D2Q9) lattice Boltzmann equation (LBE) with multi-relaxation-time (MRT) to simulate incompressible turbulent cavity flows with the Reynolds numbers up to 1×10^7. To improve the computation efficiency of LBM on the numerical simulations of turbulent flows, the massively parallel computing power from a graphic processing unit (GPU) with a computing unified device architecture (CUDA) is introduced into the MRT-LBE-LES model. The model performs well, compared with the results from others, with an increase of 76 times in computation efficiency. It appears that the higher the Reynolds numbers is, the smaller the Smagorinsky constant should be, if the lattice number is fixed. Also, for a selected high Reynolds number and a selected proper Smagorinsky constant, there is a minimum requirement for the lattice number so that the Smagorinsky eddy viscosity will not be excessively large.

Key words large eddy simulation (LES), multi-relaxation-time (MRT), lattice Boltzmann equation (LBE), two-dimensional nine velocity components (D2Q9), Smagorinsky model, graphic processing unit (GPU), computing unified device architecture (CUDA)

Chinese Library Classification O357
2010 Mathematics Subject Classification 76F65

1 Introduction

The lattice Boltzmann method (LBM)[1–3] has been proved to be an effective computational tool for simulating and modeling many complex fluid problems such as multiphase flows[4–5], porous media[6], and turbulent flows[7–12]. In contrast to the conventional numerical solution of macroscopic equation, i.e., the Navier-Stokes equation (NSE), the LBM represents the macroscopic averaged properties by the evolution of the statistical distribution of microscopic particles in term of the discrete kinetic theory. The main advantages of using LBM include easy implementation of boundary conditions, short codes, and natural parallelism. Thus, the LBM has gained more and more attention recently[13].

Although the simplest LBM that used a single relaxation time (SRT) parameter, based on the Bhatnagar Gross Krook (BGK) approximation, has successfully simulated various hydrodynamics problems,

* Received Oct. 10, 2016 / Revised Dec. 19, 2016
† Corresponding author, E-mail: chenggongli@dicp.ac.cn

it has some weaknesses, especially, the low numerical instability which leads to an unstable pressure oscillation throughout the entire computational domain. Therefore, the multiple relaxation time (MRT) lattice Boltzmann equation (LBE) has been proposed for improving the numerical stability[14–17].

Generally, the LBM is used as a direct numerical simulation (DNS) method for low Reynolds number ($Re < 10\ 000$) flows because the required computer memory is limited[18]. With the MRT-LBE, cavity flows with high Reynolds numbers (up to 1×10^6) have been simulated[17]. With a further higher Reynolds number, even the MRT-LBE method will become unstable, or the computation cost will become too high. An improvement is to add the large eddy simulation (LES) technique to the MRT-LBE model[7] so that the small scale turbulence will be mimicked using a suitable model, e.g., the Smagorinsky eddy viscosity model, but the large scale eddy can be directly simulated.

Because of the significant improvement of numerical stability of MRT-LBE and the use of LES to model small scale eddies but directly simulate large scale eddies, these two approaches have been combined together to simulate turbulent flows. Many good results show that the above approach (hereafter called the MRT-LBE-LES model) is able to provide more accurate predictions of turbulent flows[8–12,19].

However, the computation efficiency of an MRT-LBE-LES model for turbulent flows is still too low to solve any practical flow problems unless a parallel computation can be used to accelerate the simulation pace. Even though the parallel programs of LBM based on the message passing interface (MPI) standard which requires an expensive computer clusters with hundred cores are available[12], the initial investment is high, or the access is limited. Recently, the graphic process unit (GPU) has been developed into an inexpensive and novel massively parallel computing device with tremendous computational power and high memory bandwidth[20]. In particular, after NVIDIA introduced a general purpose parallel computing platform and programming model called CUDA in November 2006, the parallel programming on GPU becomes relatively easy for software developers familiar with a standard programming languages such as C[21]. Thus, a GPU with CUDA has been installed in a personal computer to accelerate many applications, such as computational finance, molecular dynamics, seismic processing, and computational fluid dynamics[22].

Since an LBM updates a node only with the nearest neighbor nodes, this method can take advantage of the massively parallel capability provided by GPU and CUDA. For example, the MRT-LBE models[23–24] have been accelerated by using a NVIDIA 8800 Ultra graphics card with CUDA for the simulation of the square array cylinders and the moving sphere in a pipe, respectively; the two-dimensional (2D) lattice Bhatnagar Gross Krook (LBGK) codes'[25] computation efficiency has been improved by using a NVIDIA GTX 280 CUDA-enabled graphics card. In this study, we develop a GPU-based MRT-LBE-LES model for simulating turbulent flows. In order to evaluate the computation efficiency of GPU-based MRT-LBE-LES program and the ability of MRT-LBE-LES model to simulate turbulent flows, 2D lid driven cavity flows for various Reynolds numbers, up to 10^7, are chosen as test cases. Application of this GPU-based MRT-LBE-LES model for a three-dimensional (3D) weak turbulent flow (with $Re = 1\times10^4$) is also available[19].

This study is organized as follows. In Section 2, the LES with the Smagorinsky eddy viscosity model is extended to the MRT-LBE. In Section 3, the implementation details of the MRT-LBE-LES model with CUDA-GPU are introduced. In Section 4, the simulation results of 2D lid driven cavity flows with the help of CUDA-GPU parallel computing are presented, and conclusions are summarized in Section 5.

2 MRT-LBE with turbulence modeling

2.1 MRT-LBE

The MRT-LBE describes the fluid flow by using two processes: collision and stream processes of particle distribution function (PDF), $\boldsymbol{f}$, on each lattice node, $\boldsymbol{r}_i$, in which the collision step happens on the moment space M, and the stream step is executed in the velocity space V. Although the details are available in Ref. [15], a summary is repeated here to make this a completed study.

The evolution equation of MRT lattice Boltzmann model that can also be called the generalized LBE or the moment method, for Q velocity directions on the D-dimensional lattice, can be written as

$$\boldsymbol{f}(\boldsymbol{r}_i + \boldsymbol{e}_\alpha \delta t, t + \delta t) = \boldsymbol{f}(\boldsymbol{r}_i, t) - M^{-1}\widehat{S}(\boldsymbol{m}(\boldsymbol{r}_i, t) - \boldsymbol{m}^{(\mathrm{eq})}(\boldsymbol{r}_i, t)), \qquad (1)$$

where the bold symbols denote the Q-dimensional column vectors, i.e., $\boldsymbol{f}$=$(f_0,\, f_1,\, \cdots,\, f_{Q-1})^{\mathrm{T}}$, $\boldsymbol{m}$=$(m_0,\, m_1,\, \cdots,\, m_{Q-1})^{\mathrm{T}}$, and the superscript T stands for the transposal operator. In Eq. (1), the left-hand side describes the particle stream from the lattice node $\boldsymbol{r}_i$ to the nearest location $\boldsymbol{r}_i + \boldsymbol{e}_\alpha \delta t$ at the particle velocities $\boldsymbol{e}_\alpha$ along the α-direction with the time interval δt in the velocity space V. The right-hand side of Eq.(1) represents the particle collision step, i.e., the relaxation process, executed in the moment space M, in which the moment vector $\boldsymbol{m}(\boldsymbol{r}_i,\, t)$ returns to the local equilibrium value $\boldsymbol{m}^{(\mathrm{eq})}(\boldsymbol{r}_i,\, t)$ at a rate determined by the diagonal relaxation matrix S, and then, the transformation matrix M (with size $Q \times Q$) linearly transforms the PDF vector $\boldsymbol{f}$ to its moment vector $\boldsymbol{m}$, that is, $\boldsymbol{m}$= $M \cdot \boldsymbol{r}$ and $\boldsymbol{r}$= $M^{-1} \cdot \boldsymbol{m}$.

Here, the MRT-LBE model with nine (i.e., $\alpha = 0, 1, \cdots, 8$) discrete particle velocities in the 2D lattice (D2Q9 MRT-LBE model in Fig. 1) is selected, and these directions are defined as $\boldsymbol{e}_{\alpha=0} = (0,\, 0)$, $\boldsymbol{e}_{\alpha=1-4} = c(\cos((\alpha-1)\pi/2),\, \sin((\alpha-1)\pi/2))$, and $\boldsymbol{e}_{\alpha=5-8} = \sqrt{2}c(\cos((\alpha-1)\pi/2+\pi/4),\, \sin((\alpha-1)\pi/2+\pi/4))$, where c=$\delta x/\delta t$ stands for the magnitude of the particle velocity, and δx refers to the dimensionless lattice length. For simplicity, δx and δt are set equal to 1, that is, c= $\delta x = \delta t = 1$. In particular, the corresponding transformation matrix M is defined as

$$M = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -4 & -1 & -1 & -1 & -1 & 2 & 2 & 2 & 2 \\ 4 & -2 & -2 & -2 & -2 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & -1 & 0 & 1 & -1 & -1 & 1 \\ 0 & -2 & 0 & 2 & 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 1 & 0 & -1 & 1 & 1 & -1 & -1 \\ 0 & 0 & -2 & 0 & 2 & 1 & 1 & -1 & -1 \\ 0 & 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 \end{bmatrix}. \tag{2}$$

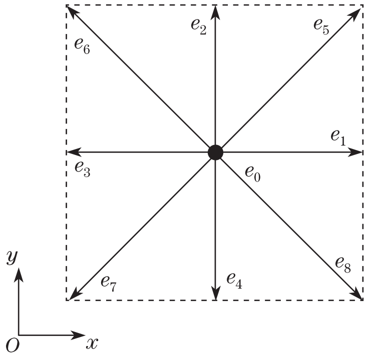


Fig. 1 Sketch of D2Q9 MRT lattice Boltzmann model

For the D2Q9 MRT-LBE model, the nine components $\{m_\alpha(\boldsymbol{r}_i,\, t) | \alpha = 0,\, 1,\, \cdots,\, 8\}$ of the moment vector $\boldsymbol{m} = (\rho,\, e,\, \varepsilon,\, j_x,\, q_x,\, j_y,\, q_y,\, p_{xx},\, p_{xy})^{\mathrm{T}}$ are set as the below order: m_0=ρ is the mass density, m_1 =e stands for the kinetic energy, m_2= ε is related to the kinetic energy square, $m_3 = j_x$ and $m_5 = j_y$ are the two components of the momentum, $m_4 = q_x$ and $m_6 = q_y$ stand for the two components of the energy flux, and $m_7 = p_{xx}$ and $m_8 = p_{xy}$ are the stress tensors. The nine components $\{m_\alpha^{(\mathrm{eq})}(\boldsymbol{r}_i, t) |$ α= $0, 1, \cdots, 8\}$ of the equilibrium moment vectors $\boldsymbol{m}^{(\mathrm{eq})}$=$(\rho,\, e^{(\mathrm{eq})},\, \varepsilon^{(\mathrm{eq})},\, j_x,\, q_x^{(\mathrm{eq})},\, j_y,\, q_y^{(\mathrm{eq})},\, p_{xx}^{(\mathrm{eq})},\, p_{xy}^{(\mathrm{eq})})^{\mathrm{T}}$ consist of two parts, i.e., the conserved moment that includes mass density $m_0^{(\mathrm{eq})}$ =ρ and two components of the momentum $m_3^{(\mathrm{eq})} = j_x$, $m_5^{(\mathrm{eq})} = j_y$, and the non-conserved moments that are shown as

$$e^{(\mathrm{eq})} = -2\rho + 3(j_x^2 + j_y^2), \quad \varepsilon^{(\mathrm{eq})} = \rho - 3(j_x^2 + j_y^2), \tag{3a}$$

$$q_x^{(\mathrm{eq})} = -j_x, \quad q_y^{(\mathrm{eq})} = -j_y, \tag{3b}$$

$$p_{xx}^{(\mathrm{eq})} = (j_x^2 - j_y^2), \quad p_{xy}^{(\mathrm{eq})} = j_x j_y. \tag{3c}$$

Thus, the equilibrium PDF $f_\alpha^{(\mathrm{eq})}(\boldsymbol{r}_i,t)$ which corresponds to the equilibrium moments $\boldsymbol{m}^{(\mathrm{eq})}$ is defined as

$$f_\alpha^{(\mathrm{eq})}=\omega_\alpha\rho\Big(1+\frac{\boldsymbol{e}_\alpha\cdot\boldsymbol{u}}{c_\mathrm{s}^2}+\frac{(\boldsymbol{e}_\alpha\cdot\boldsymbol{u})^2}{2c_\mathrm{s}^4}-\frac{\boldsymbol{u}^2}{2c_\mathrm{s}^2}\Big), \tag{4}$$

where ω_α is the weight parameters, $\omega_0=4/9$, $\omega_{1-4}=1/9$, $\omega_{5-8}=1/36$, and $c_\mathrm{s}=c/\sqrt{3}$ stands for the lattice sound speed. The macroscopic velocity $\boldsymbol{u}$ and the density ρ are functions of $f_\alpha(\boldsymbol{r}_i,\ t)$, i.e., $\rho=\sum\limits_{\alpha=0}^{8}f_\alpha$ and $\rho\boldsymbol{u}=\sum\limits_{\alpha=0}^{8}\boldsymbol{e}_\alpha f_\alpha$.

In Eq.(1), the diagonal relaxation matrix S is given by

$$\widehat{S}=\mathrm{diag}(s_0,s_1,s_2,s_3,s_4,s_5,s_6,s_7,s_8), \tag{5}$$

where s_i $(i=0,\ 1,\ \cdots,\ 8)$ stands for the relaxation rate corresponding to nine velocity directions. Since the incompressible NSE could be deduced from the MRT-LBE by using the multi-scale technique Chapman-Enskog expansion, the viscosity υ, which is obtained from this expansion under the condition $s_7=s_8=1/\tau$ where τ is the dimensionless relaxation time corresponding to the viscosity, is shown as

$$\upsilon=c_\mathrm{s}^2\delta t(\tau-0.5). \tag{6}$$

2.2 LES with D2Q9 MRT lattice Boltzmann model

In the LES, any physical variable of the government equations, e.g., f, is decomposed into a resolved component $\langle f\rangle$ and an unresolved component f' by using a filter function so that the filtered equations for the large eddies (resolved component) could be directly computed, and the effects of the smaller eddies (unresolved component) are replaced by using a selected model to represent it, e.g., the Smagorinsky eddy viscosity, υ_t, model[7,26]. With the use of LES in the D2Q9 MRT-LBE, the total viscosity υ is expressed as the sum of the molecular viscosity υ_0 and the eddy viscosity υ_t, i.e., $\upsilon=\upsilon_0+\upsilon_\mathrm{t}$[7]. Thus, the corresponding total relaxation time τ can also be separated into two components: $\tau=\tau_0+\tau_\mathrm{t}=3(\upsilon_0+\upsilon_\mathrm{t})+0.5$, where $\tau_0=3\upsilon_0+0.5$, and $\tau_\mathrm{t}=3\upsilon_\mathrm{t}$ are the dimensionless relaxation time. In the Smagorinsky model[26–27], the eddy viscosity υ_t is defined as

$$\upsilon_\mathrm{t}=(C_\mathrm{sm}\Delta)^2S^*,\quad S^*=\sqrt{2S_{ij}S_{ij}}, \tag{7}$$

where C_sm is the Smagorinsky constant, Δ stands for the filtered size, which is set as the lattice length, i.e., $\Delta=\delta x=1$ because of the use of uniform lattice, S^* is the magnitude of the shear strain rate tensor S_{ij}, and $S_{ij}=\Big(\dfrac{\partial u_i}{\partial x_j}+\dfrac{\partial u_j}{\partial x_i}\Big)\Big/2$ can be computed directly from the second-order moments of PDF, Q_{ij}. It has been proved[10] that

$$Q_{ij}=\sum e_{\alpha i}e_{\alpha j}(f_\alpha-f_\alpha^{(\mathrm{eq})})=-\frac{2\delta t\rho c_\mathrm{s}^2S_{ij}}{s_{xx}},\quad i,j\in(x,y), \tag{8}$$

where s_{xx} is the relaxation rate for the second-order moments, p_{xx} and p_{xy}, that is, $s_{xx}=s_7=s_8=1/\tau$. Thus, the eddy viscosity can be estimated as

$$\upsilon_\mathrm{t}=(C_\mathrm{sm}\Delta)^2S^*=\frac{s_{xx}}{2c_\mathrm{s}^2\rho\delta t}(C_\mathrm{sm}\Delta)^2Q^*,\quad Q^*=\sqrt{2Q_{ij}Q_{ij}}, \tag{9}$$

where the details on how to calculate Q^* were given in Ref. [10].

By combining $\upsilon=\upsilon_0+\upsilon_\mathrm{t}$, $\tau=\tau_0+\tau_\mathrm{t}$, and Eq. (9), the eddy viscosity can be obtained as follows:

$$\upsilon_\mathrm{t}=(-\tau_0+\sqrt{(\tau_0)^2+6(\rho\delta tc_\mathrm{s}^2)^{-1}(C_\mathrm{sm}\Delta)^2Q^*})/6. \tag{10}$$

2.3 Boundary condition

For 2D cavity flows, the no-slip boundary condition, i.e., the half-way bounce back boundary condition[12], is applied on all the boundaries. Because of the use of this boundary condition, the physical boundary is specified between the fluid nodes and the inner fictional ghost nodes. Thus, the collision step is only applied on the fluid nodes. Because of this setup, the unknown PDFs of the fluid nodes that are immediately inside of the boundary can be estimated as follows[12]:

$$f_{\bar{\alpha}} = f_\alpha + 2\omega_\alpha \rho_0 \frac{e_{\bar{\alpha}} U_{\rm lid}}{c_{\rm s}^2}, \tag{11}$$

where $f_{\bar{\alpha}}$ is the PDF of $\boldsymbol{e}_{\bar{\alpha}}=-\boldsymbol{e}_\alpha$, and $U_{\rm lid}$ stands for the velocity on the boundary, e.g., the lid speed or zero for a stationary boundary.

3 Parallel algorithm (GPU-based MRT-LBE-LES model)

3.1 Introduction of CUDA

CUDA$^{\rm TM}$ is a parallel computing platform and programming model that enables the dramatic increase in computing performance by using the power of a GPU. After the release of the CUDA, the parallel computational technique based on GPU has been successfully used in various fields[21–22]. The CUDA is designed to support various programming languages or application programming interfaces, e.g., FORTRAN, C, MATLAB, and OPENCL. Thus, the researchers familiar with the standard programming languages can easily code the CUDA programs for solving complex computational problems. The software environment of CUDA consists of the CUDA Toolkit and the CUDA Software Development Kit (SDK). The available operation systems include Windows XP, Vista, 7, Linux, Unix, and Mac[21–22].

As shown in Fig. 2, a CUDA program consists of a host part that runs on the central processing unit (CPU) side in serial and a device part that runs on the GPU side in parallel. The device part code, called kernels, can be invoked by the serial code on the CPU side. The primary tasks of serial codes include (i) initial clarifications, (ii) allocation and de-allocation memory spaces, and (iii) data transmissions between the host memory and the device memory. When the data are transferred to device memory, many threads (up to thousands) are launched by the kernel functions to exploit the data parallelism. In order for an easy programming, these threads with independent one-dimensional (1D), 2D, or 3D thread index are divided into many equally-shaped thread blocks according to the data sizes, and these thread blocks are organized into 1D or 2D grid of thread blocks, as shown in Fig. 2. Because of GPU hardware limitations, the number of threads in one thread block and the number of thread blocks in a grid should be properly chosen.

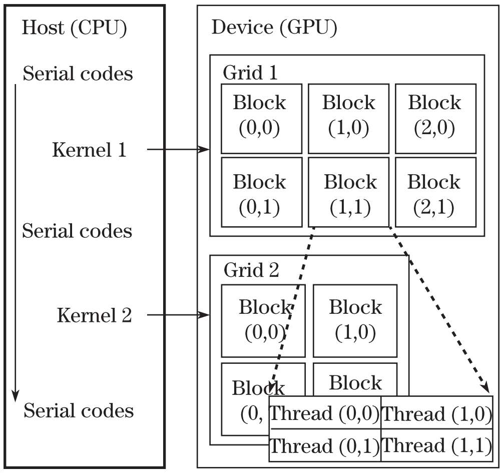
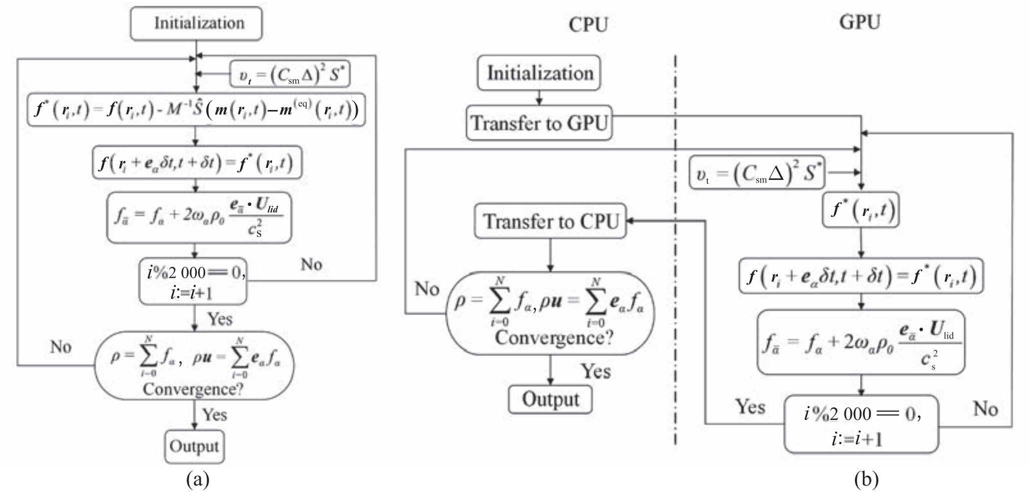


Fig. 2 Heterogeneous CUDA programming model

3.2 Implementation details

The solving of an MRT-LBE model consists of three major calculation parts: (i) initialization, (ii) collision, stream, and boundary processing, and (iii) convergence check, see the flowchart in Fig. 3(a). The main task of the first part is to initialize all the variables, e.g., the PDFs and the corresponding moments according to the velocity and Reynolds number. Since this is a one-time process and only spends little time, it is unworthy of the concurrent execution. The second part (i.e., collision, stream, and boundary functions) is repeatedly calculated many times for updating all the PDFs. This part takes about 98% of the total computing time in the CPU-only code, and therefore, this part is rewritten as three kernels executed on the GPU device in parallel. The third part is to check convergence. For laminar flows, the convergence is to check whether the results reach the steady state. For turbulent flows, it is used for determining whether the lattice size or the Smagorinsky constant is enough. The reason is that if the lattice size or the Smagorinsky constant is not right, the program will blow up because the relaxation time is close to 0.5. The number of executions of this part is much smaller than that for the second part, and does not have much advantage to be parallelized. Thus, it is run in serial in CPU, see the flowchart in Fig. 3(b).


Initialization
$\upsilon_t=(C_{\rm sm}\Delta)^2 S^*$
$f^*(r_i,t)=f(r_i,t)-M^{-1}\hat{S}(m(r_i,t)-m^{(\rm eq)}(r_i,t))$
$f(r_i+e_\alpha\delta t,t+\delta t)=f^*(r_i,t)$
$f_{\bar{\alpha}}=f_\alpha+2\omega_\alpha\rho_0\frac{e_{\bar{\alpha}}\cdot U_{\rm lid}}{c_s^2}$
$i\%2\,000==0,$
$i{:=}i+1$
No
Yes
No
$\rho=\sum_{i=0}^{N}f_\alpha,\ \rho u=\sum_{i=0}^{N}e_\alpha f_\alpha$
Convergence?
Yes
Output
CPU
GPU
Initialization
Transfer to GPU
$\upsilon_t=(C_{\rm sm}\Delta)^2 S^*$
Transfer to CPU
$f^*(r_i,t)$
$f(r_i+e_\alpha\delta t,t+\delta t)=f^*(r_i,t)$
No
$\rho=\sum_{i=0}^{N}f_\alpha,\rho u=\sum_{i=0}^{N}e_\alpha f_\alpha$
Convergence?
$f_{\bar{\alpha}}=f_\alpha+2\omega_\alpha\rho_0\frac{e_{\bar{\alpha}}\cdot U_{\rm lid}}{c_s^2}$
Yes
Output
Yes
$i\%2\,000==0,$
$i{:=}i+1$
No

(a) (b)

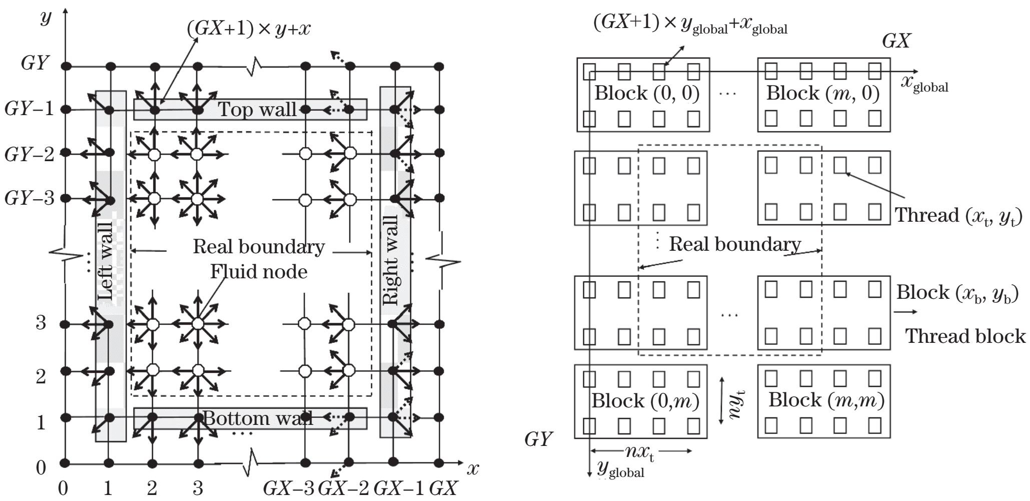
Fig. 3 Flow charts of sequential CPU-only program of MRT-LBE (a) and parallel GPU-based (b)

A 2D lid driven cavity flow deals the flow field within a square enclosure that is driven by a selected uniform velocity on the top lid from left to right. All the other three walls are fixed. Cavity flows are typically selected to verify a numerical scheme, and thus, it is used here again to demonstrate the efficiency of GPU parallel computing. The cavity enclosure can be fitted by using a uniform lattice with lattice units GX and GY in the x- and y-directions, respectively. In other words, there are $(GX+1)$ and $(GY+1)$ nodes in the x- and y-directions, respectively. Although $GX=GY$ in this special case, different symbols are used to map the lattice to the thread. In order to have efficient parallel computation, two ghost layers of nodes (solid circles in Fig. 4(a)) are added around the real boundary. The inner ghost layer is to incorporate the half-way bounce back boundary condition into the LBM, and the outer layer is used for improving the efficiency of using parallel computing in the GPU kernel with the minimum 'if statement'. More details will be given later.

In the CPU-only program, the PDFs on the D2Q9 MRT-LBE model are often set as a 3D array, such as $f[x][y][k]$, where x and y stand for a node coordinate in the Cartesian system, i.e., $0 \leqslant x \leqslant GX$ and $0 \leqslant y \leqslant GY$, and k indicates the nine different directions of particles moving. For the GPU-based program, the 3D array is rewritten as nine 1D arrays on the host memory, i.e., $f_0[P]$, $f_1[P]$, $\cdots$, $f_8[P]$, where $P=(GX+1)\times y+x$ represents the node index because of the change of a 2D array into a linear addressed memory stored in a 1D array by the row-major convention. Thus, the transfer from the 2D node array to the 1D array starts from the bottom row and moves upward, as shown in Fig. 4(a). For example, Node (3, 3) will change to Node $((GX+1)\times 3+3)$. For avoiding the read-write conflict on the GPU side, two sets of these nine linear 1D arrays, i.e., $f_{0\_\rm old}[P]$, $f_{1\_\rm old}[P]$, $\cdots$, $f_{8\_\rm old}[P]$ and $f_{0\_\rm new}[P]$, $f_{1\_\rm new}[P]$, $\cdots$, $f_{8\_\rm new}[P]$, are allocated on device memory, one set for reading data, and another for writing data.

After the transfer of data from the host to the device memory, one thread which has the general index $T=((GX+1)\times y_{\rm global}+x_{\rm global})$ is mapped to a lattice node $(GX+1)\times y+x$, and these threads are equally divided into many thread blocks in one grid, as shown in Fig. 4(b), where $x_{\rm global}$ and $y_{\rm global}$

are the global coordinates of one thread, $x_{\rm t}$ and $y_{\rm t}$ stand for the local coordinates of one thread in one thread block, $x_{\rm b}$ and $y_{\rm b}$ are the coordinates of one block in the grid, and $x_{\rm t}$ and $y_{\rm t}$ stand for the numbers of threads in the x- and y-directions in one block, respectively. Thus, the global coordinates of a thread, $x_{\rm global}$ and $y_{\rm global}$, are defined as $x_{\rm global}$=$x_{\rm b} \times x_{\rm t}$+ $x_{\rm t}$ and $y_{\rm global}$=$y_{\rm b}\times$ $y_{\rm t}$ +$y_{\rm t}$, and each thread block has $x_{\rm t} \times y_{\rm t}$ threads which is chosen by the node assignment.



(a) Uniform physical lattice, left part (i.e., columns 1, 2, 3) explaining stream kernel, and right part (i.e., column GX–2 and GX–1) showing boundary kernel

(b) Threads and thread blocks

Fig. 4 Uniform physical lattice and corresponding threads and thread blocks for lid driven flows simulated using D2Q9 MRT-LBE-LES model

In our GPU-based program, three kernels are repeatedly invoked for particles: collision, stream, and boundary condition processing, respectively.

Case 1 Collision kernel

This kernel is responsible for the particle collision. Since the particle collisions happen only on the fluid nodes, the computation is only carried out on fluid nodes. As a consequence, the execution of collision kernel is done simultaneously in GPU instead of using a 'for loop' in the CPU-only program. Herein, since each thread is mapped to a fluid node, only one 'if statement' is used in the collision kernel for choosing the threads that correspond to the fluid nodes. The eddy viscosity is computed by Eq. (10) before the calculations of the post-collision moments of PDFs in the above process.

Case 2 Stream kernel

This kernel is responsible for particle streaming. The particles on all fluid nodes will propagate to the adjacent nodes after the particle collisions on each fluid node, and the PDFs on these fluid nodes that are next to the boundary nodes are still unknown, e.g., the fluid nodes on the x = 2 column, as shown in the left part of Fig. 4(a), where the solid arrows stand for the post-stream PDFs in the arrow directions. For example, only three fluid nodes, i.e., (2, GY–3), (3, GY–3), and (3, GY–2), will give particles with the north, northwest, and west directions to the corner fluid node (2, GY–2), respectively. Thus, on the (2, GY–2) node, there are three solid arrows in the north, northwest, and west directions, respectively. No arrows in the other directions on this node show that the PDFs with the other directions are unknown. It is the same to the hatched box on the left side of Fig. 4(a), e.g., the boundary node (1, GY–1) only receives particles from the fluid node (2, GY–2), and so one solid arrow is on it. The stream process can also be viewed as one set of nine procedures, i.e., the PDFs with the same direction on the all fluid nodes are given to the neighbor nodes, totally eight directions (the PDFs with the $\boldsymbol{e}_0$-direction are at rest). In this kernel function, the nine procedures can be simultaneously performed by the corresponding threads on GPU because only the fluid nodes give particles to the adjacent nodes. In addition, one thread mapped with the selected fluid node has the thread index $T = (GX\ +1) \times (y_{\mathrm{global}}) + x_{\mathrm{global}}$. Therefore, the thread indexes of the nearest eight neighbor nodes can be searched by the equation N=$(GX$+1$) \times (y_{\mathrm{global}} + \delta t \times \boldsymbol{e}_\alpha) + (x_{\mathrm{global}} + \delta t \times \boldsymbol{e}_\alpha)$, where $\alpha = 1$, 2, $\cdots$, 8. Note that $\delta t = 1$ and the two components of discrete particle velocity $\boldsymbol{e}_\alpha$ are either 0 or 1. Thus, δt and $\boldsymbol{e}_\alpha$ are only used to find the thread indexes of neighbor nodes.

Case 3 Boundary kernel

The kernel is responsible for boundary conditions, i.e., after the stream kernel, all the unknown PDFs on the fluid nodes are updated using Eq. (11). In the CPU-only program, this process can be easily performed even without 'if statement' by returning the PDFs with $\boldsymbol{e}_{\bar{\alpha}}$, which has been updated on the boundary in the stream step, back to the unknown PDFs with $\boldsymbol{e}_\alpha$ on the fluid nodes. Unfortunately, in the GPU-based program, the implementation of the above procedure needs too many ‘if statements’ for choosing the specified threads that correspond to the lattice nodes. This leads to a significant loss of the parallel efficiency. In order to avoid this, another outer ghost layer is introduced (see Fig. 4(a)). The approach proposed in this study is that the PDFs which have the direction of the dotted and solid arrows on the hatched box of the right wall (the GX–1 column) give back to the adjacent left fluid nodes for updating all the unknown PDFs according to Eq. (11)(see the right part of Fig. 4(a)). For example, on the node (GX–1, 2), the solid arrow with the southeast direction is for returning to the solid arrow with the northwest direction on the node (GX–2, 3), the solid arrow with the east direction is for returning to the solid arrow with the west direction on node (GX–2, 2), and the dotted arrow with the northeast direction is for returning to the dotted arrow with the southwest direction on node (GX–2, 1), where the solid and dotted arrows on the right wall are the PDFs that have and have not been updated in the stream step, respectively. In this process, even though the PDFs with the directions of the dotted arrows are assigned to the opposite PDFs of the neighbor nodes (the dotted line arrows on the GX–2 column), it has no effect on the results due to the fact that no changes happen on the PDFs of the fluid nodes, and the macroscopic variables are computed by the PDFs of the fluid nodes only. This has the advantage that the half-way bounce back boundary condition could be easily rewritten as the GPU kernel, and the parallel efficiency of GPU program can be improved significantly by using a minimum ‘if statement’. The additional lattice nodes only request a small extra amount of memory and a negligible small computing time for data copying.

With all that implemented, the massive parallel computing can be used more smoothly, and the efficiency of the MRT-LBE-LES program is excellent for numerical simulations of the 2D lid driven flow. With the capability of using the same double precision in the GPU and the CPU programs, they have the same accuracy.

4 Results and discussion

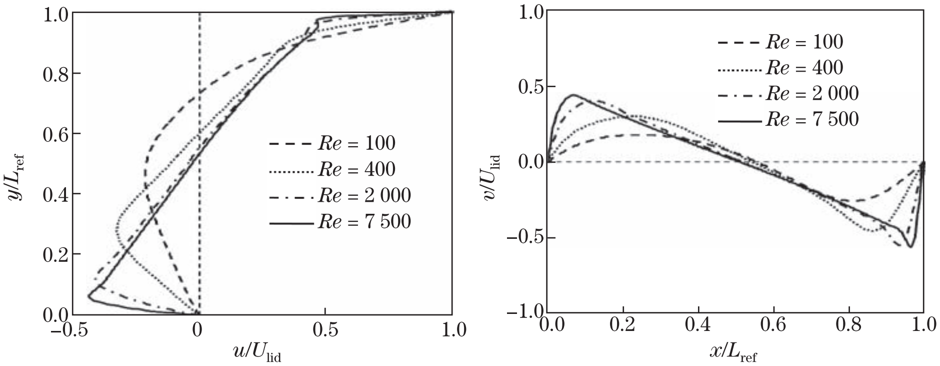
4.1 2D lid driven laminar cavity flow

Here, the numerical experiments on 2D cavity flows with Re =100, 400, 2 000, 7 500, 8 400 (i.e., the laminar flow) are performed for testing the GPU-based program of the D2Q9 MRT-LBE-LES model with C_{sm}=0. The dimensionless uniform top driven velocity in the lattice Boltzmann system, U_{lid}, is set to be equal to 0.1, and the Reynolds number is defined as $Re = U_{\mathrm{lid}} L_{\mathrm{ref}}/\upsilon$, where υ is the dimensionless kinetic viscosity of the fluid, and L_{ref}=GX–3=GY–3 is the dimensionless characteristic length used in the lattice Boltzmann system, where the extra 3 is from the half-way bounce back boundary condition and the extra layer for a better parallel efficiency to process the half-way bounce back boundary condition in the GPU kernel. Therefore, the total lattice units in each line are GX.

A uniform lattice unit of 259 (i.e., GX=GY=259, L_{ref}=GX–3=256) is selected in the x- and y-directions, respectively. The nine components in the diagonal collision matrix are given by $s_0 = s_3 = s_5 = 0$, $s_4 = s_6$= 1.2, $s_1 = s_4$ – 0.1, $s_2 = s_1$ – 0.1, and s_7= s_8= $1/\tau$. The steady situation of a laminar flow can be claimed if the difference of normalized kinetic energy, δ, between two consecutive checking times is less than a small value, e.g., 10^{-12},

$$\delta = \sqrt{\frac{1}{N}\sum_{i=1}^{N}((u_i^{n+1} - u_i^n)^2 + (v_i^{n+1} - v_i^n)^2)/((u_i^{n+1})^2 + (v_i^{n+1})^2)}. \tag{12}$$

For laminar flows, the simulation results (except for $Re = 8\ 400$ because no data can compare with it) at the steady situation (i.e., velocity profiles at the center of cavity, see Fig. 5) agree well with the reference[18,28–30]. The velocity profiles change from a smooth curve at low *Re* to a nearly straight line with sharply bends near the boundaries at high *Re*. The locations of the primary and secondary vortices for different Reynolds numbers also agree well with the references[18,28–30] (see Table 1).



(a) Horizontal velocity u profiles through geometric center

(b) Vertical velocity v profiles through geometric center

Fig. 5 D2Q9 MRT-LBE-LES model simulated velocity profiles for laminar cavity flows through geometric center

Table 1 Comparisons of calculated coordinates for primary vortex (x_{c}, y_{c}) and two secondary vortices, left: (x_{l}, y_{l}) and right: (x_{r}, y_{r})

Re		x_{c}	y_{c}	x_{l}	y_{l}	x_{r}	y_{r}
100	Ghia et al.[28]	0.617 2	0.734 4	0.0313	0.039 1	0.945 3	0.062 5
	Hou et al.[18]	0.619 6	0.737 3	0.039 2	0.035 3	0.945 1	0.062 7
	Present	0.617 2	0.738 3	0.039 1	0.035 2	0.944 9	0.062 5
400	Ghia et al.[28]	0.554 7	0.605 5	0.050 8	0.046 9	0.890 6	0.125 0
	Hou et al.[18]	0.560 8	0.607 8	0.054 9	0.051 0	0.890 2	0.125 5
	Present	0.558 6	0.609 4	0.054 7	0.050 8	0.886 7	0.125 0
2 000	Hou et al.[18]	0.525 5	0.549 0	0.090 2	0.105 9	0.847 1	0.098 0
	Du et al.[30]	0.523 4	0.546 9	0.085 9	0.097 7	0.847 7	0.097 7
	Present	0.523 4	0.550 8	0.085 9	0.105 5	0.843 8	0.097 7
7 500	Ghia et al.[28]	0.511 7	0.532 2	0.064 5	0.150 4	0.781 3	0.062 5
	Hou et al.[18]	0.517 6	0.533 3	0.070 6	0.152 9	0.792 2	0.066 7
	Du et al.[30]	0.515 6	0.535 2	0.066 4	0.152 3	0.796 9	0.066 4
	Erturk[31]	0.513 7	0.532 2	0.064 5	0.152 3	0.791 0	0.065 4
	Present	0.515 6	0.535 2	0.066 4	0.152 3	0.793 0	0.066 4

In this study, the maximum Reynolds number of 2D lid driven cavity flows which can be computed with the steady solution is 8 400. To obtain the steady solution for the cavity flow with a higher Reynolds number (i.e., $Re > 8\ 400$), the number of lattice size is increased to $3\ 075^2$ for allowing the cavity flow to have steady solutions[31]. However, we observe that the solutions are not convergent but oscillating with time. This suggests that the 2D lid driven cavity flow field is unstable when *Re* is larger than 8 400, and the time-averaged results should be used to judge whether the solution converges to a steady state.

4.2 2D lid driven cavity turbulent flow

As $Re > 8\ 400$, the computed instantaneous velocity of the lid driven flow will oscillate despite the selection of large lattice units. This indicates the start of changing flow pattern from laminar to turbulent flows. For a fixed number of lattice unit, e.g., 259^2 lattice units, the D2Q9 MRT-LBE-LES model with $C_{\rm sm}= 0$ will continue to produce results, although unstable, until $Re > 29\ 796$. At that time, the fluctuation is too large, and the program crashes because the relaxation time τ is too close to 0.5. This suggests that keeping increasing the lattice units can make the relaxation time sufficiently larger than 0.5, and thus, it is possible to use the MRT-LBE model for turbulent flows. However, the higher the Reynolds number (i.e., the larger the lattice number), the more the computational cost, and even with the help from a GPU, the computation cost will be too high to do the task on a personal computer for large (more than $3\ 300^2$) lattice sizes. That is the reason to implement the LES for simulating high Reynolds number flows with a reasonably selected coarse lattice.

To evaluate the efficiency and capability of the GPU-based MRT-LBE-LES program in simulating turbulent flows, the 2D lid driven flows with five high Reynolds numbers, i.e., 5×10^4, 1×10^5, 5×10^5, 1×10^6, and 1×10^7 are checked. The dimensionless driven velocity at the lid $U_{\rm lid}$ is still 0.1. Since the instantaneous velocity field is fluctuating with time, it is impossible to find a stable flow field unless the time-averaged flow is used. The selection of duration for taking average may depend on the following factors: the number of lattice units, the Smagorinsky constant, the initial running time before taking time average, and the duration of average. Here, we find that the initial running time of $500T$ (where $T = L_{\rm ref}/U_{\rm lid}$) and averaged time of $150T$ are sufficient to warrant a stable time-averaged 2D lid driven flow. A longer initial running time and longer average time make little difference in the time-averaged velocity profile. Thus, the initial running time and duration for average are set as $500T$ and $150T$, respectively, for all cases. This is one of the major differences compared with the early studies[7,17].

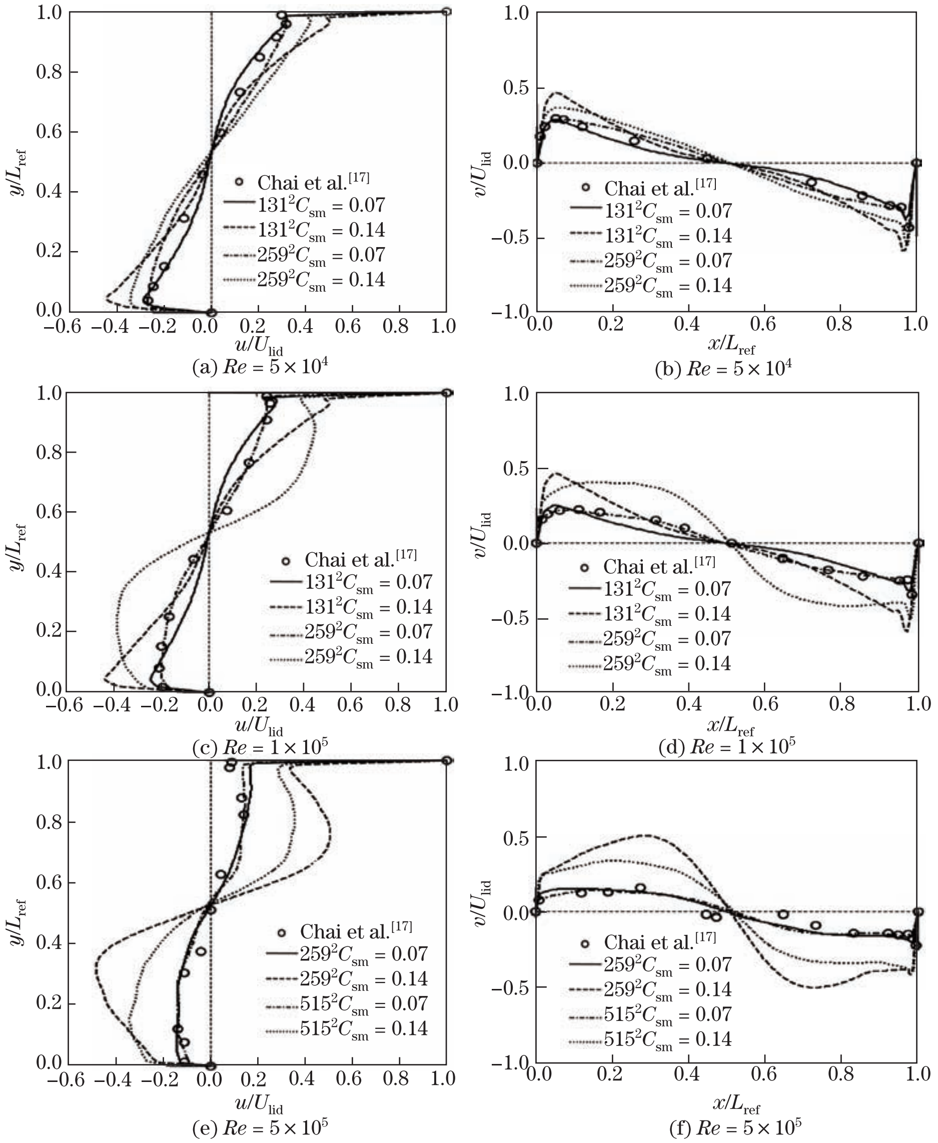
In addition, possible values for the Smagorinsky constant, from 0.05 to 0.3, are tested to address its effects on the time-averaged results. Different lattice units are also checked to examine its effect. For checking the computation efficiency in different thread numbers, the lattice nodes are arranged as a multiple of 16, i.e., 128^2, 256^2, 512^2, $\cdots$, $3\ 072^2$ lattice nodes. The selection of the finest lattice units, i.e., $3\ 072^2$ is limited by our graphic card memory.

The results suggest that the Smagorinsky constant $C_{\rm sm}$ has great effect on the time-averaged velocity, if the lattice units are the same for the fixed Re (see Fig. 6). For instance, the results with $C_{\rm sm} = 0.07$ are close to those given by Chai et al.[17], but smaller than the results with $C_{\rm sm}$=0.14 on the 127^2, 255^2, and 511^2 lattice units. It may be because that as $C_{\rm sm}$ increases, the eddy viscosity also increases, and thus, the momentum can be further transferred into the cavity. It appears that for the same $C_{\rm sm}$, there is a minimum requirement for the lattice units. For example, with $C_{\rm sm}$=0.07 and $Re = 5\times10^4$ and 1×10^5, the time-averaged velocity profiles show that the selected minimum lattice units, i.e., 127^2, is acceptable, but it may be too coarse to provide comparable results for higher Reynolds numbers. For $Re = 5\times10^5$ and 1×10^6, the results on the 511^2 lattice units are better to match with the results of Chai et al.[17] when compared with that using 255^2 lattice units, and the lattice units of 127^2 is not acceptable at all.

After the Smagorinsky constant and the number of lattice are determined, the model computed streamlines are constructed (see Fig. 7). For Re=5×10^4 and Re=1×10^5, it can be seen that a primary vortex dominates the entire flow field with several secondary vortices appearing in the three corners, and the shapes and locations of these secondary vortices will change vastly with time. Although only one primary vortex dominates the entire flow field, the center for the primary core is relatively stable for $Re = 5\times10^4$, but the center starts to rotate for $Re = 1\times10^5$, and these secondary vortices fluctuate more. When Re increases to 5×10^5, it appears that a secondary vortex may be released from the boundary and moves into the area mainly occupied by the primary vortex. When Re=1×10^6, two primary vortices rotate around the cavity center, and more and more secondary vortices are produced in corners. Generally, as Re increases, the flow field becomes more turbulent and more oscillatory.

For further evaluating the eddy viscosity term in the D2Q9 MRT-LBE-LES model, the simulations of 2D cavity flows at $Re = 1\times10^7$ are performed with 255^2, 511^2, and $1\,023^2$ lattice units. Because there are no other numerical or/and experimental data at this high Reynolds number, no comparison of velocity profiles can be made. The numerical results show that the time-averaged velocity is also greatly affected by the Smagorinsky constant $C_{\rm sm}$ for the same lattice size at the fixed Re (see Fig. 8). It is assumed that for $Re = 1\times10^7$, $C_{\rm sm}$ should be slightly reduced, and thus $C_{\rm sm}$=0.065 is selected for the three different lattice sizes. It is also observed that the instantaneous streamline contour plots are very different after the $500T$ initial running.

As the lattice units increase but no change on $C_{\rm sm}$, the primary vortex is split and gradually replaced with many secondary vortices (see Figs. 9 and 10(a)). This may be due to the use of the same C_{sm} for different lattice numbers. Since $C_{\mathrm{sm}}\Delta$ is corresponding to the mixing length[26], the maximum length is limited to Δ (for the traditional mixing length hypothesis, the length could be much larger, e.g., half of the water depth). For this reason, the Smagorinsky eddy viscosity model only addresses small scale, local eddy viscosity, and cannot describe the variation of eddy viscosity globally. For a fixed Reynolds number, if the lattice number is small, the physical lattice length Δ represents must be large. For this reason, $C_{\mathrm{sm}}\Delta$ should increase as the lattice number decreases. If the local turbulent eddy viscosity becomes excessively large, it will affect the velocity and produce unrealistic velocity profiles. As a consequence, the selection of lattice size will affect the simulation results for a fixed C_{sm}. This also explains why the time-averaged velocity increases with C_{sm} when the lattice size remains unchanged (see Fig. 6).



(a) $Re = 5\times10^4$

(b) $Re = 5\times10^4$

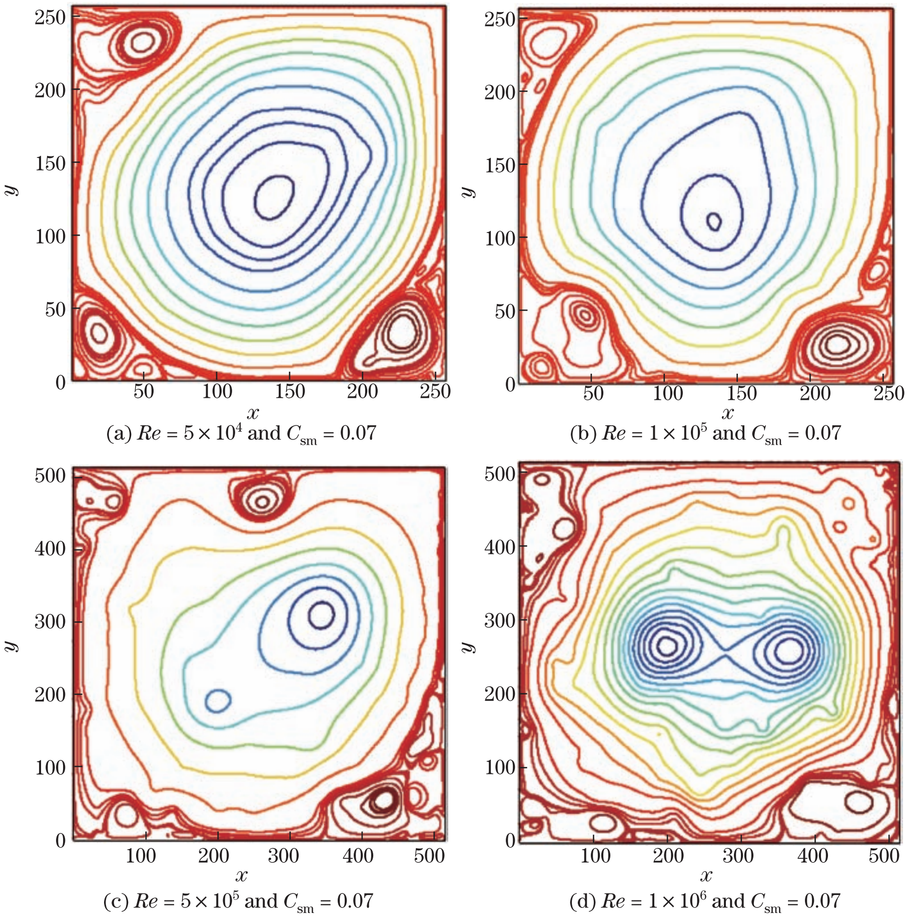
(c) $Re = 1\times10^5$

(d) $Re = 1\times10^5$

(e) $Re = 5\times10^5$

(f) $Re = 5\times10^5$

Fig. 6 Comparisons of GPU model simulated time-averaged horizontal ((a), (c), (e), (g)) and vertical ((b), (d), (f), (h)) velocity profiles at center of turbulent cavity flows with $Re = 5\times10^4$, 1×10^5, 5×10^5, and 1×10^6, where circles are simulation results given by Chai et al.[7] using MRT-LBE model without LES



(a) $Re = 5 \times 10^4$ and $C_{\text{sm}} = 0.07$

(b) $Re = 1 \times 10^5$ and $C_{\text{sm}} = 0.07$

(c) $Re = 5 \times 10^5$ and $C_{\text{sm}} = 0.07$

(d) $Re = 1 \times 10^6$ and $C_{\text{sm}} = 0.07$

Fig. 7 Model simulated instantaneous streamlines at selected Reynolds numbers after initial running time $500T$

4.3 Performances of MRT-LBE-LES with GPU

In order to accelerate the 2D MRT-LBE-LES program, a personal computer (AMD Phenom II 1100T CPU) with 16 GB memory and one CUDA-enable graphic card (GTX 580) is built for this purpose. The GTX 580 has 512 CUDA cores and 192.4 GB/s memory bandwidth and can deliver over 0.2 TeraFLOPs in double precision operations. The software environment for the computer consists of Windows XP professional 64bit, CUDA Toolkit version 4.1, and GPU driver 286.19[21].

When taking time-average to show the mean velocity field, as the typical approach for turbulent flows, the time-averaged flow pattern is expected (see Fig. 10(b)). In contrast with the instantaneous streamlines plot, these small vortices, except those at the three corners, are smoothed out by taking time-average over a sufficiently long period, e.g., $150T$. Moreover, the total kinetic energy k defined as $k = \langle (u')^2 + (v')^2 \rangle /2$, for this high Re (1×10^7) is much stronger than that for the cavity flow with Re=1×10^5 (see Fig. 11), where the notation $\langle\ \rangle$ stands for the time averaged over $150T$, and u' and v' are the velocity fluctuations in the x- and y-directions, respectively.

The computational efficiency of lattice Boltzmann simulations is often measured according to how many million lattice nodes could be updated per second (MLUPS)[23–25]. Since the calculation of the eddy viscosity has negligible effect on the MLUPS, the 2D lid driven flow at $Re = 7\ 500$ is taken as the example (see Table 2). In general, the calculation efficiency of GPU-based program increases with the number of threads per block, but Table 2 indicates that the efficiency is about the same when more than 32 threads per block are used. The best performance is achieved with 128 threads per block for different grid sizes. The comparison of MLUPS between GPU-based (128 threads) and CPU-only program on the 512^2 lattice shows a ratio of 338/4.317, about 78 times improvements.

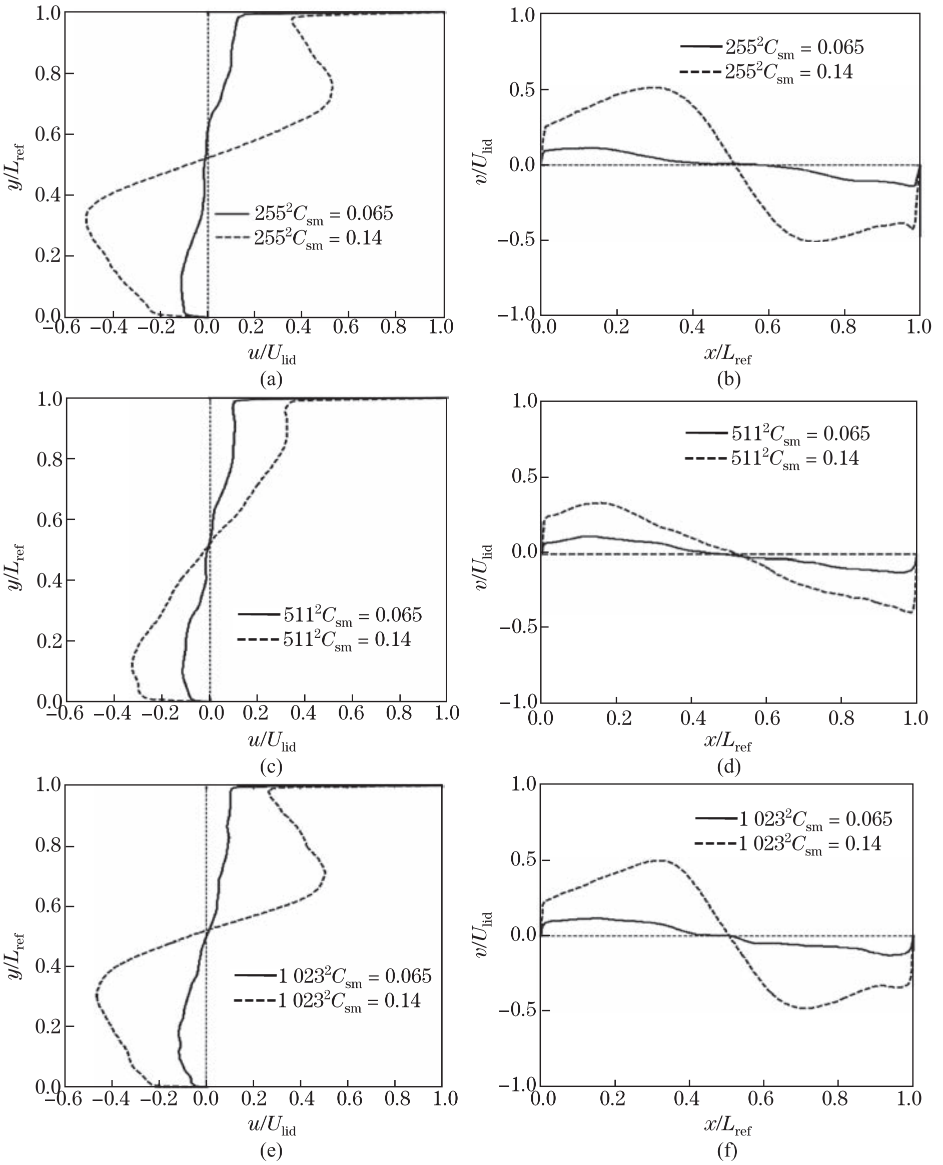
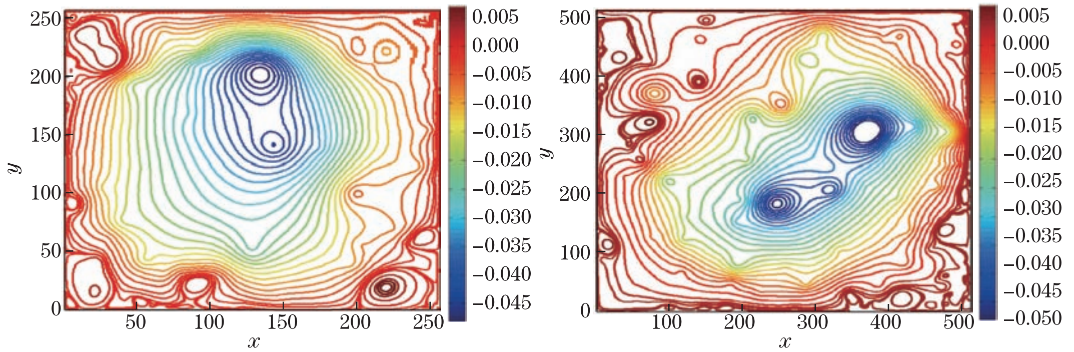


Fig. 8 Comparisons of GPU model simulated time-averaged horizontal ((a), (c), (e)) and vertical ((b), (d), (f)) velocity profiles at center of turbulent cavity flows with $Re=1\times10^7$ for two Smagorinsky constants, $C_{\text{sm}}=0.065$ and 0.14, in which lattice numbers are different among subplots

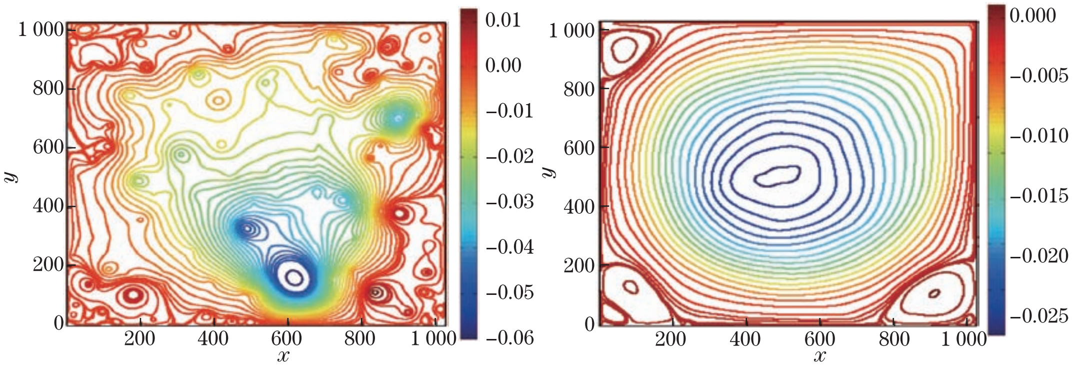
Table 2 Comparisons of performance in MLUPS of CPU-only and GPU-based programs on simulation of 2D cavity flow at $Re = 7\,500$ and averaged for 10 000 iterations using double precision floating point

Domain size (lattice node)	CPU	GPU (thread per block)					
		16	32	64	128	256	512
256^2	5.791	160	293	295	310	310	303
512^2	4.317	160	306	326	338	321	298
$1\,024^2$	4.503	172	332	337	337	329	297
$2\,048^2$	4.660	169	320	329	335	332	312
$3\,072^2$	5.802	171	316	319	326	322	312



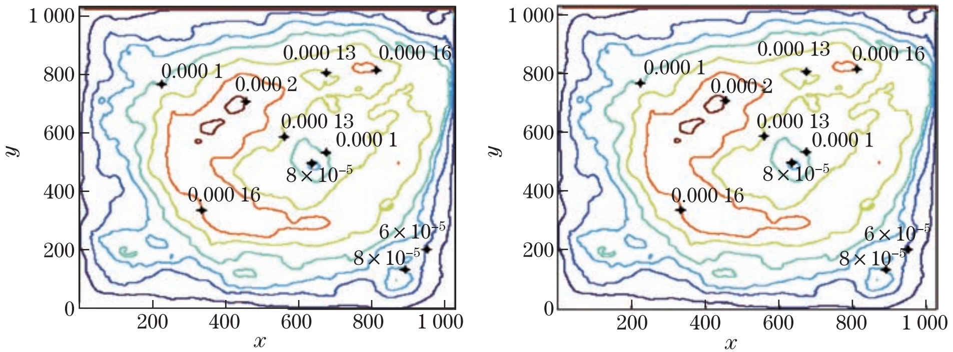
(a) Lattice number is 255^2 (b) Lattice number is 511^2

Fig. 9 Instantaneous streamlines after initial run time, $500T$, for cavity flow with $Re=1\times10^7$ and Smagorinsky constant $C_{\rm sm}=0.065$



(a) Instantaneous snap shot at $500T$ (b) Time averaged (over $150T$ snap shots) streamlines

Fig. 10 Comparison of streamlines for $Re=1\times10^7$ with lattice units$=1\ 023^2$ and Smagorinsky constant $C_{\rm sm}=0.065$ after $500T$



(a) Lattice units are $1\ 023^2$ (b) Lattice units are 259^2

Fig. 11 Comparisons of turbulent kinetic energy contours for 2D cavity flows with $Re = 1 \times 10^7$ and $C_{\rm sm} = 0.065$

5 Conclusions

In this study, the implementation of D2Q9 MRT-LBE-LES model based on the CUDA technology is presented, and the numerical codes are validated by the 2D lid driven flows over a large range of Reynolds numbers.

For the 2D lid driven flows at low Reynolds numbers, i.e., laminar flows, the results agree well with the numerical results from the others. The maximum Reynolds number of 2D cavity flows which can be simulated for the steady solution by using the MRT-LBE-LES model is $Re = 8\ 400$. When the flows are changed from laminar flows to turbulent flows, i.e., $Re = 5\times10^4$, 1×10^5, 5×10^5, 1×10^6, and 1×10^7, the time-averaged variables should be used because the instantaneous variables are always fluctuating. The modeling results show that MRT-LBE-LES model can well simulate the 2D lid driven flows at high Reynolds numbers, up to 1×10^7.

As the change of $C_{\rm sm}$ and lattice number may alter the local eddy viscosity, the entire flow field will be affected. As the increase of $C_{\rm sm}$ or the decrease of physical lattice size roughly means the increase of local eddy viscosity (if the shear strain rate tensor remains the same), excess momentum transfer from the top lid to the water below increases the mean velocity, and thus, larger and probably unrealistically large velocity may result. This implies that there is a minimum requirement on the lattice number, which depends on the Reynolds number, for successful simulations of turbulent flows. More studies are necessary to clearly address this issue.

Since the double precision floating point is used in the CPU-only and GPU-based program, there is the same computational accuracy. An increase of 76 times in computation speed when compared with CPU-only codes shows that the GPU massively parallel computation method is excellent. This advantage could be further enhanced if the limitation caused by the GPU hardware, e.g., 0.2 TeraFLOPs in double precision FLOPs and 1.5GB memory for the GTX 580 graphic card, is relaxed. Simulations with finer resolutions (e.g., $4\ 096^2$) lid driven flows or 3D turbulent flows with $Re > 1\times10^4$ will become possible. Other alternatives, e.g., use of multiple GPUs, or a better GPU, e.g., Tesla C2050, or a new GPU-architecture, Kepler, which delivers 3 to 4 times better double precision FLOPs performance, when compared with that for a GTX 580, is possible.

Since using a CUDA-enabled GPU and a regular desktop PC is inexpensive, smaller, and less power-demanding than a cluster with multi-CPU, the LBM with CUDA-GPU technique can be more frequently applied for simulating fluid dynamic phenomena.

Acknowledgements The authors would like to thank Prof. Lishi LUO for providing the initial MRT-LBE CPU code. This work is supported by College of William and Mary, Virginia Institute of Marine Science for the study environment.

References

[1] McNamara, G. R. and Zanetti, G. Use of the Boltzmann equation to simulate lattice-gas automata. *Physical Review Letters*, **61**, 2332–2335 (1988)

[2] Chen, H. D., Chen, S. Y., and Matthaeus, W. H. Recovery of the Navier-Stokes equations using a lattice-gas Boltzmann method. *Physical Review A*, **45**, 5339–5342 (1992)

[3] Qian, Y. H., d'Humières, D., and Lallemand, P. Lattice BGK models for Navier-Stokes equation. *Europhysics Letters*, **17**, 479–484 (1992)

[4] Shan, X. W. and Chen, H. D. Lattice Boltzmann model for simulating flows with multiple phases and components. *Physical Review E*, **47**, 1815–1820 (1993)

[5] Dou, Z., Zhou, Z. F., Huang, Y., and Wu, W. Numerical study of non-aqueous phase liquid transport in a single filled fracture by lattice Boltzmann method. *Journal of Hydrodynamics*, **24**, 130–137 (2012)

[6] Guo, Z. L. and Zhao, T. S. Lattice Boltzmann model for incompressible flows though porous media. *Physical Review E*, **66**, 036304 (2002)

[7] Hou, S. L., Sterling, J., Chen, S. Y., and Doolen, G. D. A lattice Boltzmann subgrid model for high Reynolds number flows. *Fields Institute Communications*, **6**, 151–166 (1996)

[8] Yu, H. D., Girimaji, S. S., and Luo, L. S. DNS and LES of decaying isotropic turbulence with and without frame rotation using lattice Boltzmann method. *Journal of Computational Physics*, **209**, 599–616 (2005)

[9] Fernandino, M., Beronov, K., and Ytrehus, T. Large eddy simulation of turbulent open duct flow using a lattice Boltzmann approach. *Mathematics and Computers in Simulation*, **79**, 1520–1526 (2009)

[10] Krafczyk, M., Tölke, J., and Luo, L. S. Large-eddy simulations with a multiple-relaxation-time LBE model. *International Journal of Modern Physics B*, **17**, 33–39 (2003)

[11] Yu, H. D., Luo, L. S., and Girimaji, S. S. LES of turbulent square jet flow using an MRT lattice Boltzmann model. *Computers and Fluids*, **35**, 957–965 (2006)

[12] Premnath, K. N. and Pattison, M. J. Generalized lattice Boltzmann equation with forcing term for computation of wall-bounded turbulent flows. *Physical Review E*, **79**, 026703 (2009)

[13] Sukop, M. C. and Thorne, D. T. *Lattice Boltzmann Modeling: an Introduction for Geoscientists and Engineers*, Springer, Berlin (2006)

[14] d'Humières, D. Generalized lattice Boltzmann equations. *Rarefied Gas Dynamics: Theory and Simulations* (eds. Shizgal, B. D. and Weaver, D. P.), *Progress in Astronautics and Aeronautics*, American Instiute of Aeronautics and Astronauics, Reston (1992)

[15] Lallemand, P. and Luo, L. S. Theory of the lattice Boltzmann method: dispersion, dissipation, isotropy, Galilean invariance, and stability. *Physical Review E*, **61**, 6546–6562 (2000)

[16] d'Humières, D., Ginzburg, I., Krafczyk, M., Lallemand, P., and Luo, L. S. Multiple-relaxation-time lattice Boltzmann models in three dimensions. *Philosophical Transactions of the Royal Society A*, **360**, 437–451 (2002)

[17] Chai, Z. H., Shi, B. C., and Zheng, L. Simulating high Reynolds number flow in two-dimensional lid-driven cavity by multi-relaxation-time lattice Boltzmann method. *Chinese Physics*, **15**, 1855–1863 (2006)

[18] Hou, S. L., Chen, S. Y., Doolen, G. D., and Cogley, A. C. Simulation of cavity flow by the lattice Boltzmann method. *Journal of Computational Physics*, **118**, 329–347 (1995)

[19] Li, C. G., Maa, J. P. Y., and Kang, H. G. Solving generalized lattice Boltzmann model for 3-D cavity flows using CUDA-GPU. *Science China Physics, Mechanics and Astronomy*, **55**, 1894–1904 (2012)

[20] Li, W., Wei, X. M., and Kaufman, A. Implementing lattice Boltzmann computation on graphics hardware. *Visual Computer*, **19**, 444–456 (2003)

[21] NVIDIA Corporation. *NVIDIA CUDA* 4.1 *Programming Guide* (2011)

[22] Kirk, D. B. and Hwu, W. W. *Programming Massively Parallel Processors: a Hands-on Approach*, Morgan Kaufmann, Burlington (2010)

[23] Tölke, J. Implementation of lattice Boltzmann kernel using the compute unified device architecture developed by NVIDIA. *Computing and Visualization in Science*, **13**, 29–39 (2009)

[24] Tölke, J. and Krafczyk, M. TeraFLOP computing on a desktop PC with GPUs for 3D CFD. *International Journal of Computational Fluid Dynamics*, **22**, 443–456 (2008)

[25] Kuznik, F., Obrecht, C., Rusaouen, G., and Roux, J. LBM based flow simulation using GPU computing processor. *Computers and Mathematics with Applications*, **59**, 2380–2392 (2010)

[26] Pope, S. *Turbulent Flows*, Cambridge University Press, New York (2000)

[27] Smagorinsky, J. General circulation experiments with the primitive equations: I, the basic equations. *Monthly Weather Review*, **91**, 99–164 (1963)

[28] Ghia, U., Ghia, K. N., and Shin, C. T. High-*Re* solutions for incompressible flow using the Navier-Stokes equations and a multigrid method. *Journal of Computational Physics*, **48**, 387–411 (1982)

[29] Erturk, E., Corke, T. C., and Gökçöl, C. Numerical solutions of 2-D steady incompressible driven cavity flow at high Reynolds numbers. *International Journal for Numerical Methods in Fluids*, **48**, 747–774 (2005)

[30] Du, R., Shi, B. C., and Chen, X. W. Multi-relaxation-time lattice Boltzmann model for incompressible flow. *Physical Letters A*, **359**, 564–572 (2006)

[31] Erturk, E. Discussions on driven cavity flow. *International Journal for Numerical Methods in Fluids*, **60**, 275–294 (2009)