Implementation of an incompressible lattice Boltzmann model on GPU to simulate Poiseuille flow

Tanmay Agrawal*
Department of Mechanical Engineering
National Institute of Technology Hamirpur, India
tanmayagrawal7@gmail.com
Tel: +919882109547

Chao-An Lin
Department of Power Mechanical Engineering
National Tsing Hua University, Hsinchu, Taiwan
calin@pme.nthu.edu.tw
Tel: +886-3-5742602

Abstract— Incompressible two dimensional lattice Boltzmann method (LBM) is adopted to simulate flow inside a pipe commonly called Poiseuille flow. Instead of D2Q9 lattice Boltzmann model, we used an incompressible two-dimensional model, D2Q9i to simulate the flow conditions. At the inlet, velocity boundary condition, while at outlet, pressure boundary condition and at the surfaces, no slip boundary condition is used. The computing platform used in the present simulation is NVIDIA GTX 560Ti which is capable to deliver nearly 1263 Giga Floating-point Operations Per Second (GFLOPS). Results are reported for laminar flows and are in excellent agreement with those available in literature.

Keywords- lattice Boltzmann method, fluid flow simulation, poiseuille flow.

INTRODUCTION
In recent years, the lattice Boltzmann method as a relatively new numerical scheme has achieved considerable success in simulating hydrodynamic problems [1-5]. LBM originated from the lattice gas automata (LGA), a discrete particle kinetics utilizing a discrete lattice and discrete time. Since there are some difficulties resulting from the LGA to LBM, various approximations such as linearization of the collision operator and the Bhatnagar-Gross-Krook (BGK) approximation have been developed. From a computational viewpoint, due to the major advantages that it is explicit, easy to implement and natural to parallelize, the lattice BGK model has become the most popular lattice Boltzmann model.

Different from the traditional computational fluid dynamic methods which solve the nonlinear Navier-Stokes equations directly, the LBM simulates fluid flows from the particle’s viewpoint. By applying the macroscopic constraints, such as low expansion velocity, the Navier-Stokes equation can be recovered from the lattice Boltzmann equation. Instead of solving time consuming Poisson equation for pressure, LBM uses an extremely simple formula, \( P=\rho C_s^2 \) [6].

As an explicit numerical scheme with intensive local computation, the LBM algorithm is very suitable for parallelization. This can be achieved using the Graphical Processing Unit (GPU) through the Compute Unified Device Architecture (CUDA). Graphic processing unit has been successfully used for lattice Boltzmann computations [7, 8, 9]. Computational power of GPUs has far exceeded that of PC-based CPUs. Researchers have focused to increase the parallel performance by reducing the data transaction between host and device. Some efforts adopt different streaming strategy to increase the parallel performance [7, 8]. Another way to increase the performance of a GPU is to utilize the shared memory [10]. A detailed description of CUDA memory can be found in [11].

This paper is arranged as follows. Section II presents the mathematical fundamentals of D2Q9i lattice model...
Boltzmann method along with the boundary conditions to be used for the simulation. A brief introduction about the graphic processing unit is given in Section III. Programming language used to write the code is introduced in Section IV. Section V presents the results obtained using this incompressible LBM model.

MATHEMATICAL FORMULATION AND BOUNDARY CONDITIONS

D2Q9i lattice Boltzmann equations adopting a uniform lattice with BGK collision model can be expressed as,

\[ f_i(\vec{x} + \vec{e}_i \Delta t, t + \Delta t) = f_i(\vec{x}, t) - \frac{1}{\tau} [f_i(\vec{x}, t) - f_i^{eq}(\vec{x}, t)] \]  

(1)

where \( f_j \) is the particle density function along the particle velocity direction \( e_j \), \( \tau \) is the dimensionless relaxation time that controls the rate approaching equilibrium [12].

Equilibrium distribution function is given by:

\[ f_i^{eq} = w_i \left[ \rho + 3(e_i, u) + \frac{9}{2} (e_i, u)^2 - \frac{3}{2} (u, u) \right] \]  

(2)

Based on the density distribution function, the macroscopic variables are defined as:

\[ \sum_{i=0}^{9} f_i = \rho \]  

(3)

\[ \sum_{i=0}^{9} f_i e_i = u \]  

(4)

To implement lattice Boltzmann simulations, two essential steps are required, which are defined as follows:

**Collision step:**

\[ f_i(\vec{x}, t) = f_i(\vec{x}, t) - \frac{1}{\tau} [f_i(\vec{x}, t) - f_i^{eq}(\vec{x}, t)] \]

**Streaming step:**

\[ f_i(\vec{x} + \vec{e}_i \Delta t, t + \Delta t) = f_i(\vec{x}, t) \]

In any lattice Boltzmann simulation, boundary treatment is considered to be an extremely critical step. On boundaries, some distribution functions become unknown due to its upstream direction is out of computational domain. Modified bounce-back boundary conditions [13] at the wall and pressure/velocity boundary conditions are applied at the inlet and outlet [14]. While on the corners, the unknown distribution functions are obtained by equating the non-equilibrium part of the LBE.

GRAPHIC PROCESSING UNIT

A graphics processing unit (GPU), also occasionally called visual processing unit (VPU), is a specialized electronic circuit designed to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display. GPUs are used in embedded systems, mobile phones, personal computers, workstations, and game consoles. Modern GPUs are very efficient at manipulating computer graphics, and their highly parallel structure makes them more effective than general-purpose CPUs for algorithms where processing of large blocks of data is done in parallel. In a personal computer, a GPU can be present on a video card. The term GPU was popularized by NVIDIA in 1999, who marketed the GeForce 256 as "the world's first 'GPU', or Graphics Processing Unit, a single-chip processor with integrated transform, lighting, triangle setup/clipping, and rendering engines that are capable of processing a minimum of 10 million polygons per second".

It is becoming increasingly common to use a general purpose graphics processing unit as a modified form of stream processor. GPGPU can be used for many types of embarrassingly parallel tasks including ray tracing, computational fluid dynamics and weather modelling. They are generally suited to high-throughput type computations that exhibit data-parallelism to exploit the wide vector width SIMD architecture of the GPU.

COMPUTE UNIFIED DEVICE ARCHITECTURE

CUDA (aka Compute Unified Device Architecture) is a parallel computing platform and programming model created by NVIDIA and implemented by the graphics processing units (GPUs) that they produce. CUDA gives developers access to the virtual instruction set and memory of the parallel computational elements in CUDA GPUs. Using CUDA, the latest NVIDIA GPUs become accessible for computation like CPUs.
Before the execution of kernel functions, each relevant data are declared and initialized. For the Particle Distribution Functions (PDFs) and macroscopic quantities such as density and velocity, these are allocated on both host and device memory. After initialization, values of above quantities will be passed from host memory into device memory for kernel computations. Once the maximum computational step is reached, values stored in device memory are called back to update the information in the host memory. In order to increase the parallel performance, data transfer between host and device memory will only take place in the first beginning and the end of whole simulation to reduce the overall computational time.

Kernel functions executed in our program contain four major parts:

a. Collision step: In this step, processors will execute collision operator which is explained in Section II. The collision operator requires only local computation and each cell can execute this step independently without cells interaction. The results after the collision operator are stored in the post PDF array for the next operation.

b. Streaming step: In the streaming function, each PDF will propagate to the next cell by following the direction of its lattice velocity. Because of the propagation, communication between cells will occur. For cells belong to different thread blocks, it is unavoidable to access data through global memory, and the parallel performance will then be affected owing to the large latency of global memory.

c. Boundary step: This function is responsible for the boundary condition implementation. Boundary condition implementation involves only local computation just the same as the collision step.

d. Macro-computing step: In this step, macroscopic quantities like density and velocity in each cell are calculated using the updated PDFs. Values of these quantities will then be used for next time step computation. When the maximum time step is reached, these macroscopic quantities will be transferred back to host memory and whole simulation loop is closed.

RESULTS

A code written in CUDA was developed for the simulation of Poiseuille flow considering the boundary conditions mentioned above in the text. To validate our simulation results quantitatively, we ran a test case and compared our velocity profile with the analytical solution available in the literature. For this test case, Reynolds number was set to 10 and domain size was 128 by 64. Comparison is shown in Fig. 1.

![Fig. 1 Validation of LBM simulation](image)

After successfully validating our code, we obtained the following results for a two dimensional Poiseuille flow:

a. **Velocity Distribution in the channel:**

![Fig. 2. Horizontal velocity contour in the channel](image)
Fig. 2. shows the distribution of horizontal component of velocity. It is expected that the center line should be the line of maximum velocity and the result is up to the expectations. As stated earlier, using an incompressible model eliminates the compressibility error present inherently in D2Q9 LBM model. This fact is clear from Fig. 3 as there is no change in velocity profile as the flow progresses.

Fig. 3 Velocity profiles at different sections

b. **Pressure Distribution:**

It is known that as the fluid progresses in the channel, its pressure slightly decreases because of dissipative effects such as viscosity and shear stresses. Fig. 4. shows the decrease in pressure along the length of the channel.

Fig. 4. Pressure contour in the channel

c. **Performance Improvement:**

In general, we use global device memory which is an off-chip memory and thus its access is not too fast. To improve the performance of GPU, we utilize shared memory which is an on-chip memory and accessing it is very fast. However, its size is very less compared to the global device memory. Performance is measured in MLUPS (Million Lattice sites Update Per Second), which is defined as:

\[
MLUPS = \frac{\text{grid size} \times \text{max.iteration}}{\text{time} \times 10^6}
\]

Table 1 presents the performance of NVIDIA GTX 560 Ti GPU with using global and shared memories independently. Also, it shows how block size effects the GPU performance and thereby reaching an approximate conclusion of optimum block size of 128. Compared to a CPU, computations on a GPU using only global memory can be approximately 20 times faster depending upon the grid size [9]. Using shared memory further speed up the computations.

<table>
<thead>
<tr>
<th>Block size</th>
<th>D2Q9i 1024×128 Global memory</th>
<th>D2Q9i 1024×128 Shared memory</th>
<th>Improvement in GPU Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>96.09</td>
<td>183.06</td>
<td>1.92x</td>
</tr>
<tr>
<td>32</td>
<td>182.3</td>
<td>334.36</td>
<td>1.89x</td>
</tr>
<tr>
<td>64</td>
<td>296</td>
<td>564.96</td>
<td>1.92x</td>
</tr>
<tr>
<td>128</td>
<td>336.94</td>
<td>682.66</td>
<td>2.1x</td>
</tr>
<tr>
<td>256</td>
<td>329.32</td>
<td>681.59</td>
<td>2.09x</td>
</tr>
<tr>
<td>512</td>
<td>287.43</td>
<td>562.5</td>
<td>2.02x</td>
</tr>
</tbody>
</table>

Table 1: Performance of GPU using global and shared memory

**CONCLUSIONS**

This paper presented simulation of Poiseuille flow using an incompressible two dimensional lattice Boltzmann method. Nine velocity D2Q9 incompressible model with modified bounce back and pressure/velocity boundary condition was used to predict the flow properties. The computing platform for the present work was NVIDIA GTX 560 Ti graphic processor which can be programmed using Compute Unified Device Architecture (CUDA). We also presented the performance analysis of a GPU if its shared memory is utilized for
computations and thereby its calculations can be made faster as compared to the case when it uses only global memory.

ACKNOWLEDGEMENTS

Authors gratefully acknowledge the computational facilities provided by the Taiwan National Center for High-Performance Computing.

REFERENCES