Imperial College London

Department of Earth Science and Engineering

MSc in Applied Computational Science and Engineering

Independent Research Project
Final Report

Optimising Graphics Processor Unit Accelerated Lattice Boltzmann Methods for High Reynolds Number Flows

by

Jakob Torben

jakob.torben17@imperial.ac.uk
GitHub login: acse-jrt3817

Supervisors:

Prof. Matthew Piggott
Dr. Steven Dargaville

August 27, 2021

Abstract

The lattice Boltzmann Method (LBM) has proven to be an effective tool in fluid simulations due to its simple implementation and parallel capabilities. This study aims to evaluate the LBM for simulating high Reynolds number flows, using three different approaches: Single-Relaxation-Time (SRT), Multiple-Relaxation-Time (MRT) and a Smagorinsky model based Large-Eddy Simulation (MRT-LES). The numerical computation was accelerated using a Graphics Processing Unit (GPU), via NVIDIA's Compute Unified Device Architecture (CUDA). Data layout and memory access optimisations resulted in speedup factors up to 79, compared to the CPU implementation, and achieved 96 % of the effective memory broadband for the MRT model. The implementation was validated using a 2D lid-driven cavity flow. The results show good agreement with literature up to Reynolds numbers of 20 000 before the boundary conditions affected the accuracy. The study demonstrated the MRT models' superior stability and found LES to increase the accuracy and stability at high Reynolds numbers.

Keywords: Lattice Boltzmann Method (LBM), lid-driven cavity flow, Single-Relaxation Time (SRT), Multiple-Relaxation Time (MRT), Large-Eddy Simulation (LES), Graphics Processing Unit (GPU).

Acknowledgment

I would like to offer my special thanks and gratitude to my supervisors: Prof. Matthew Piggott, for taking on my project and providing his expert domain knowledge on computational fluid dynamics and turbulence modelling, and Dr. Steven Dargaville for sharing his expertise and advice in lattice Boltzmann methods. Furthermore, I would like to offer my gratitude towards my family and friends for their continuous support and for providing an excellent environment for me to complete my project.

1 Introduction

Developing accurate and computational efficient methods to predict turbulent flows remains a significant challenge in the field of Computational Fluid Dynamics (CFD). Since most fluid flows naturally are in the turbulent regime, it is of great interest to increase our understanding and ability to model turbulent flows. Better turbulence models would benefit several essential fields, ranging from industrial applications such as the aerodynamics of wings and vehicles to blood flows in arteries and weather prediction. Despite the success of modern methods in numerical modelling of turbulent flows (Argyropoulos & Markatos 2015), the high computational costs remain a barrier to increasing the simulations' accuracy further.

The Lattice Boltzmann Method (LBM) is popular in CFD due to its ability to model complex flows and computational efficiency. One of the initial models in LBM provided a simple implementation by using a Single-Relaxation Time (SRT) and the Bhatnagar, Gross and Kroos approximation (Bhatnagar et al. 1954, Chen et al. 1992), known as the lattice BGK (LBGK) model. The LBGK model was found to be an efficient simulation method capable of simulating several different complex flow problems. However, it is well understood that for flows at high Reynolds numbers, the LBGK model suffers from numerical instabilities. To circumvent this deficiency, modern implementations tend to use the Multiple-Relaxation Time (MRT) model due to its superior numerical stability to the LBGK model (Coreixas et al. 2019). Several studies have confirmed the improved stability and efficiency in simulating a variety of flow problems (Lallemand & Luo 2000, Zhen-Hua et al. 2006).

The LBM applied to small Reynolds number flows can use a Direct Numerical Simulation (DNS) approach by using a lattice size capable of resolving all the eddies across all scales. However, as the Reynolds number increases, memory requirements and computation time, limit the lattice size. To overcome this challenge, turbulence models can reduce the computational cost by not directly simulating the smallest scales. In the LBM framework, Large-Eddy Simulation (LES) is a popular method as it requires only a minor modification and comes at a relatively small computational cost (Hou et al. 1995). Several studies have confirmed MRT-LES as an appropriate method for turbulence modelling, such as the surface-mounted cube benchmark (Krafczyk et al. 2003) and turbulent jet (Yu et al. 2006).

In the LBM, each node only depends on its nearest neighbours, making it an excellent candidate for parallelisation. The LBM community quickly recognised this capability and led to it being implemented successfully on several parallel computing platforms (Li et al. 2003, Bemaschi et al. 2010). Recently, Graphics Processing Units (GPU) have received increased attention in high-performance computing due to their floating-point operation rate and high memory bandwidth. In 2007, NVIDIA launched its Compute Unified Device Architecture (CUDA), which made general-purpose GPU computing more accessible and attracted interest from the scientific community. GPU's large number of computing units and the local dependence of the LBM makes the two an excellent match. The pioneering work by Tölke (2008) demonstrated an effective LBM CUDA implementation. Later, several studies effectively limited the memory requirements to further increase the performance of the GPU implementation (Bailey et al. 2009, Obrecht et al. 2011, Delbosc et al. 2014).

In this study, LBM models using SRT, MRT and a coupled MRT-LES are implemented and evaluated for their capability to simulate flows at high Reynolds numbers. The code will be GPU accelerated using the CUDA language, with several optimisations from literature. The goal is to develop a highly efficient LBM implementation capable of simulating high Reynolds number flows. The study is organised as follows: First, a brief introduction to the LBM and governing equations before describing the implementation details and applied optimisations. Finally, the implementation is tested for Reynolds numbers up to 20 000 using the standard lid-driven cavity flow problem.

2 Lattice Boltzmann Method

2.1 Lattice Boltzmann Method

The LBM is a mesoscopic model that connects the microscopic collision of random moving particles to macroscopic fluid flows through solving the Boltzmann kinetic equation, a transport equation of particle distribution functions. These distribution functions represent the population of particles with a velocity ξ, positioned at r at time t. A solution, therefore, requires a discretisation in velocity space and space-time. By confining the particles to a lattice with spacing δr, introducing the time step δt, and limiting the velocities to a discrete velocity set e_α, the lattice Boltzmann equation is obtained:

$$|f_\alpha(\boldsymbol{r}_i + \boldsymbol{e}_\alpha \delta r, t + \delta t)\rangle - |f_\alpha(\boldsymbol{r}_i, t)\rangle = |\Omega_\alpha(\boldsymbol{r}_i, t)\rangle, \tag{1}$$

where f_α is the particle probability distribution function along direction α, Ω_α is the collision operator that represents the change of f_α from collisions and $\boldsymbol{r}_i$ is the lattice node position. The terms on the left hand side corresponds to the physical process of particles streaming to neighbouring sites, while the right-hand side represents the change in populations from collisions. A particle is thus only allowed to stream to a given set of directions from each node. In the D2Q9 velocity model considered here, this amounts to 9 different directions, including staying at rest, as illustrated in Figure 1.

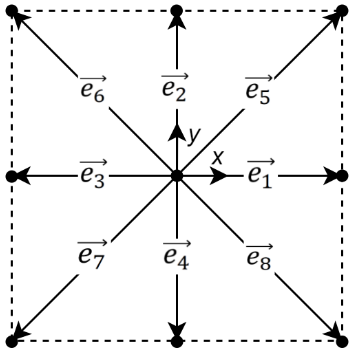


Figure 1: D2Q9 velocity model

2.2 Single-Relaxation-Time

An important simplification to the LBM was the linearisation of the collision operator, resulting in the LBGK model (Higuera & Jiménez 1989, Chen et al. 1992). By assuming small deviations and that the single-particle distributions relax towards the local equilibrium using a single-relaxation time τ the updated equation reads:

$$|f_\alpha\left(\boldsymbol{r}_i + \boldsymbol{e}_\alpha \delta r, t + \delta t\right)\rangle = |f_\alpha(\boldsymbol{r}_i, t)\rangle - \frac{\delta t}{\tau}|f_\alpha(\boldsymbol{r}_i, t)\rangle - |f_\alpha^{eq}(\boldsymbol{r}_i, t)\rangle). \tag{2}$$

τ and the local equilibrium distribution f_α^{eq} can be tuned such that in the low-Mach number limit, the incompressible Navier-Stokes (N-S) equations can be recovered up to second-order accuracy in space and time (Qian et al. 1992):

$$\tau = 3\nu + \frac{1}{2} \tag{3}$$

$$f_\alpha^{\mathrm{eq}} = w_\alpha \rho[1 + \frac{\boldsymbol{e}_\alpha \cdot \boldsymbol{u}}{c_s^2} + \frac{(\boldsymbol{e}_\alpha \cdot \boldsymbol{u})^2}{c_s^4} - \frac{\boldsymbol{u} \cdot \boldsymbol{u}}{2c_s^2}]. \tag{4}$$

where ν is the lattice kinematic viscosity. In the D2Q9 velocity model, the lattice speed of sound, c_s, is defined as $c_s^2 = \delta r^2/(3\delta t^2)$ and the weighting coefficients are given by $w_0 = 4/9, w_{1,2,3,4} = 1/9$, and $w_{5,6,7,8} = 1/36$. A benefit of the LBM, is that the macroscopic values for density and momentum can be directly obtained from the distributions:

$$\rho = \sum f_\alpha, \qquad \rho \boldsymbol{u} = \sum \boldsymbol{e}_\alpha f_\alpha \tag{5}$$

The LBGK model's simplicity and efficiency have made it into the most commonly used LBM. However, from Eq. (3), $\tau > 1/2$ is a critical stability condition for ensuring a non-negative kinematic viscosity. As $\tau \to 1/2$, numerical instabilities arise that result in non-physical negative distribution functions and effectively limits the flow to low Reynolds number flows.

2.3 Multiple-Relaxation-Time

Developed around the same time as the LBGK model, the MRT model introduced several improvements (D'Humieres et al. 1992), and has been found to have superior stability to its simpler counterpart (Lallemand & Luo 2000). The key difference of MRT is that collisions are performed in moment space, which is related to the velocity distributions through a linear transformation: $|\boldsymbol{m}\rangle = \boldsymbol{M}|f\rangle$. Performing the collisions thus requires the velocity distributions $|f_\alpha\rangle$ to be transformed into moment space, where the moments are collided towards their equilibrium, before being transformed back into their velocity distributions: $|\boldsymbol{f}\rangle = \boldsymbol{M}^{-1}|\boldsymbol{m}\rangle$. Then the updated lattice Boltzmann equation is given by

$$|f(\boldsymbol{r}_i + \boldsymbol{e}_\alpha \delta t, t + \delta t)\rangle - |f(\boldsymbol{r}_i, t)\rangle = -\boldsymbol{M}^{-1}\boldsymbol{S}[|m(\boldsymbol{r}_i, t)\rangle - |m^{(\mathrm{eq})}(\boldsymbol{r}_i, t)\rangle] \tag{6}$$

where $\boldsymbol{S}$ is the diagonal relaxation matrix. The transformation matrix $\boldsymbol{M}$, can be constructed such that the moments $|m\rangle$ resemble hydrodynamic properties, such as density, momentum and fluxes. This approach provides a convenient mean to incorporate the physics into the Lattice Boltzmann equation. Since every moment has its individual relaxation parameter, it can be tuned to achieve desired properties and stability. In this study, $\boldsymbol{M}$ is constructed through the Gram-Schmidt procedure and is given by (Lallemand & Luo 2000):

$$\boldsymbol{M} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -4 & -1 & -1 & -1 & -1 & 2 & 2 & 2 & 2 \\ 4 & -2 & -2 & -2 & -2 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & -1 & 0 & 1 & -1 & -1 & 1 \\ 0 & -2 & 0 & 2 & 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 1 & 0 & -1 & 1 & 1 & -1 & -1 \\ 0 & 0 & -2 & 0 & 2 & 1 & 1 & -1 & -1 \\ 0 & 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 \end{bmatrix}. \tag{7}$$

The row vectors of M are mutually orthogonal, ensuring that all the components of $|m\rangle$ correspond to unique hydrodynamic moments. The corresponding moment vector is defined as

$$|m\rangle = (\rho, e, \varepsilon, j_x, q_x, j_y, q_y, p_{xx}, p_{xy})^{\mathrm{T}}. \tag{8}$$

The components of $|m\rangle$ have an explicit physical significance: $m_0 = \rho$ is the density, $m_1 = e$ kinetic energy, $m_2 = \epsilon$ (kinetic energy)^2, $m_3 = j_x$ and $m_4 = j_y$ is the momentum components, $m_4 = q_x$ and $m_6 = q_y$ is the energy flux components, and $m_7 = p_{xx}$ and $m_8 = p_{xy}$ correspond to the diagonal and off-diagonal stress tensors. As there is no analytical method to calculate the relaxation parameters in matrix S, a trial and error approach must be used to obtain the most stable and accurate configuration. In this study, the highly stable relaxation parameters from D'Humières et al. (2002) are used:

$$\boldsymbol{S} = \mathrm{diag}(0, 1.4, 1.4, 0, 1.2, 0, 1.2, 1/\tau, 1/\tau), \tag{9}$$

The only conserved moments in the system is the density ρ and momenta j_x and j_y. These are not affected under collision, such that their equilibrium value is simply $m_0^{eq} = \rho$, $m_3^{eq} = j_x$ and $m_5^{eq} = j_y$. For the nonconserved moments, their equilibrium values can be carefully crafted to optimise the properties of the model (Lallemand & Luo 2000):

$$\begin{array}{ll} m_1^{eq} = e = -2\rho + 3\left(j_x^2 + j_y^2\right), & m_2^{eq} = \epsilon = \rho - 3\left(j_x^2 + j_y^2\right) \\ m_4^{eq} = q_x = -j_x, \quad m_6^{eq} = q_y = -j_y & \\ m_7^{eq} = p_{xx} = \left(j_x^2 - j_y^2\right), & m_8^{eq} = p_{xy} = j_x \cdot j_y \end{array} . \tag{10}$$

2.4 Large-Eddy Simulation in LBM

A defining feature for turbulent flows is the small-scale fluctuations from the shear stress generated in high Reynolds number flows. Since these fast fluctuations at the smallest scale are not accounted for in LBM, numerical instabilities occur at high Reynolds numbers (i.e. as $\tau \to 1/2$). A popular method for including the unresolved small-scale fluid motions is LES, where the large-scale resolved component is directly computed, while a subgrid-scale model represents the unresolved component. This method introduces a positive turbulent eddy viscosity, ν_t, such that the total viscosity is given by

$$\nu_{total} = \nu_0 + \nu_t \tag{11}$$

following this, the corresponding relaxation time can be defined as $\tau = \tau_0 + \tau_\mathrm{t} = 3\left(v_0 + v_\mathrm{t}\right) + 1/2$. The most common subgrid-scale model was introduced by Smagorinsky (1963), which uses the strain rate $\mathrm{S}_{ij} = \frac{1}{2}\left(\partial_i u_j + \partial_j u_i\right)$, to define the turbulent viscosity:

$$\nu_t = (C_S \Delta)^2 \bar{S}, \qquad \bar{S} = \sqrt{\sum S_{ij} \cdot S_{ij}}, \tag{12}$$

where $\bar{S}$ represents the characteristic grid-filtered strain rate, C_S is the Smagorinsky constant and $\Delta = \Delta x$ is the filter width, which together sets the Smagorinsky lengthscale. The Smagorinsky constant is an adjustable parameter that must be set according to what flow is studied. Setting C_s too low means that the small-scale high-frequency fluctuations are not damped adequately, whereas setting it too high may cause excessive damping of large-scale fluctuations. In the present study, C_S was set to 0.1, which achieved an optimal balance between accuracy and numerical stability and is often the standard value in CFD software.

An important advantage of the LBM over conventional N-S approaches is that the strain rate tensor S_{ij} can be computed locally from the second-order moments of the local distribution functions (Krafczyk et al. 2003):

$$\mathrm{S}_{ij} = \frac{3 s_{xx}}{2\rho} \mathrm{Q}_{ij} \tag{13}$$

where s_{xx} represents the second-order moments relaxation rate. Krafczyk et al. (2003) presented a method for finding the components of the tensor Q_{ij}, using the second-order moments. Here a similar procedure is used, but applied to the D2Q9 velocity model as outlined in Appendix A. Combining Eq. (12) and Eq. (13), and approximating the density to $\rho = 1$, an expression for the turbulent viscosity is obtained:

$$\nu_t = (C_S\Delta)^2\bar{S} = (C_S\Delta)^2\frac{3s_{xx}}{2\rho}\bar{Q} \qquad \bar{Q} = \sqrt{2Q_{ij}\cdot Q_{ij}}. \tag{14}$$

Finally, by combining $\nu_{total} = \nu_0 + \nu_t$, $\tau_{total} = \tau_0 + \tau_t$, Eq. (14) and requiring that ν_t depends on the strain rate at the current time step, the expression for the turbulent part of the relaxation time can be obtained:

$$\tau_t = 3\nu_t = \frac{1}{2}\left(\sqrt{\tau_0^2 + 18C_s^2\Delta_x^2\bar{Q}} - \tau_0\right), \tag{15}$$

where τ_0 is calculated from the characteristic length, flow velocity and Reynolds number: $\tau_0 = 3\nu + 1/2 = 3UL/Re + 1/2$.

2.5 Lid-Driven Cavity Flow and Boundary Conditions

The numerical implementation is here applied to the popular lid-driven cavity flow problem. The simple boundary condition treatments of this problem mean that the accuracy, stability and parallel computational performance of the LBM methods can be the main focus. Lid-driven cavity is a popular benchmark in CFD that allows studying both laminar and turbulent flows. Its popularity means that accurate numerical solutions from literature can be used to validate the results. The geometry of the test case is shown to the left in Figure 2, where no-slip boundary conditions are applied to the fixed bottom and sidewalls, and Dirichlet boundary conditions at the lid with a velocity U_{lid} to the right. In the LBM framework, no-slip boundary conditions can be implemented using the standard bounce-back rule, which reflects the velocity distributions leaving the boundary node $\boldsymbol{r_b}$, with a velocity $\boldsymbol{e}_{\bar{\alpha}} = -\boldsymbol{e}_\alpha$

$$f_{\bar{\alpha}}(\boldsymbol{r_b}, t+\delta t) = f_\alpha^*(\boldsymbol{r_b}, t). \tag{16}$$

Bounce-back at the bottom wall is illustrated in the right bottom of Figure 2. In this study, the walls are located at the boundary nodes, making it a first-order accurate scheme. Despite bounce-back's simplicity, it persists as the most popular boundary condition treatment in the LBM community. Much due to its inexpensive and straightforward implementation and being a stable numerical scheme even as $\tau \rightarrow 1/2$ (Ginzburg & D'Humières 2003), crucial for the high Reynolds number flows studied here. A drawback is that when using the SRT model, the location of the boundary depends on the viscosity and thus the hydrodynamic solution will differ with different viscosities, even though the Reynolds number is kept constant. However, this is not the case when using the MRT model (Krüger et al. 2017). For the moving lid, second-order accurate methods exist such as (Zou & He 1997) which uses bounce-back of the non-equilibrium distributions and methods that replaces only some of the distributions at the boundaries. However, these two approaches can lead to instabilities at even moderate Reynolds numbers (Latt et al. 2008). Consequently, this study uses the bounce-back method with a moving wall, thus making a trade-off for numerical stability over accuracy. The moving wall has a similar form as the bounce-back formula, but with a minor alteration as the particles gain or lose momentum (Krüger et al. 2017):

$$f_{\bar{\alpha}}(\boldsymbol{r_b}, t+\delta t) = f_\alpha^*(\boldsymbol{r_b}, \boldsymbol{t}) - 2w_\alpha\rho_w\frac{\boldsymbol{e_\alpha u_w}}{c_s^2} \tag{17}$$

where w denotes the properties defined at the wall location. Bounce-back boundary conditions at the moving lid is illustrated in the top right of Figure 2.

The model is initialised with a uniform density, zero y-velocity, and only a non-zero x-velocity at the lid, set to U_{lid}. Next, the velocity distributions are initialised to their equilibrium values, using the equilibrium distribution from Eq. (4). The corner nodes of the lid are singular nodes and must be treated explicitly. Here they are treated as part of the vertical wall, using standard bounce-back.

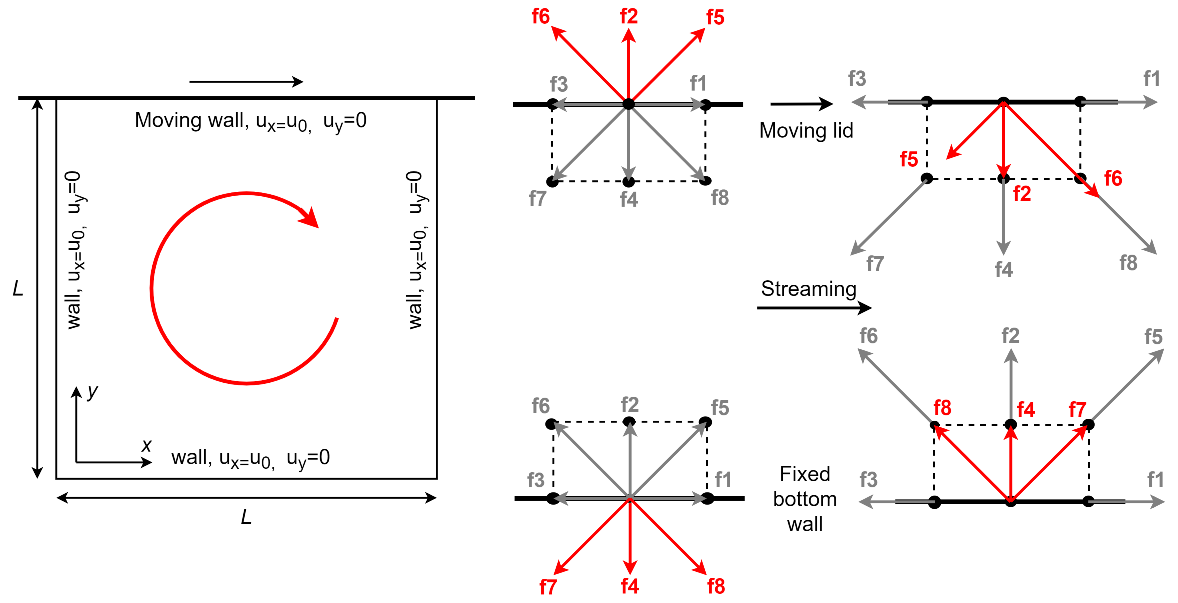

Moving wall, $u_{x=}u_{0,}$ u_y=0
wall, $u_{x=}u_{0,}$ u_y=0
wall, $u_{x=}u_{0,}$ u_y=0
wall, $u_{x=}u_{0,}$ u_y=0
L
L
y
x
f6
f2
f5
f3
f1
f7
f4
f8
Moving lid
f3
f1
f5
f2
f6
f7
f4
f8
Streaming
f6
f2
f5
f8
f4
f7
f3
f1
Fixed bottom wall
f6
f2
f5
f3
f1
f7
f4
f8

Figure 2: Left: Geometry of the lid-driven cavity flow studied here. Right: Illustration of the reflected distributions (in red) of the bounce-back boundary conditions, here shown for the fixed bottom wall and the moving lid.

3 Implementation and Software Description

3.1 Implementing LBM Using CUDA

One of the main motivations behind using the LBM is the local nature that offers abundant parallelism and an excellent opportunity for GPU acceleration. The most widely used framework for general-purpose GPU computing is using NVIDIA hardware and its CUDA language. CUDA uses the single instruction multiple data (SIMD) execution model, where the host (CPU) controls the program and launches parallel kernels on the device (GPU). The host runs in serial and is responsible for initialisation, allocating and transferring memory, and launching functions called kernels on the device. On the device, a large number of threads execute a kernel independently in parallel. The threads are organised in a three-level hierarchy, consisting of blocks of threads arranged in a grid.

Figure 3 gives a schematic algorithm of the GPU implementation of the three different LBM models in this study. First, the CPU allocates memory, sets up the thread organisation before the distributions are then initialised on the GPU. In the main loop, the distributions are streamed into temporary variables. The boundary nodes are explicitly set following the description in Section 2.5. In the SRT model, the collision is directly performed in velocity space using Eq. (2). Whereas for the MRT-LES model, the distributions are transformed into moment space using the matrix $\boldsymbol{M}$. If LES is applied, the effective relaxation time is updated from Eq. (15). If not, it is set to τ_0. Next, the MRT-LES collision is performed in moment space using Eq. (6), before being transformed back into velocity space using matrix $\boldsymbol{M}^{-1}$. At user-defined intervals, the macroscopic variables are stored in an array which is then transferred to the CPU and saved to file.

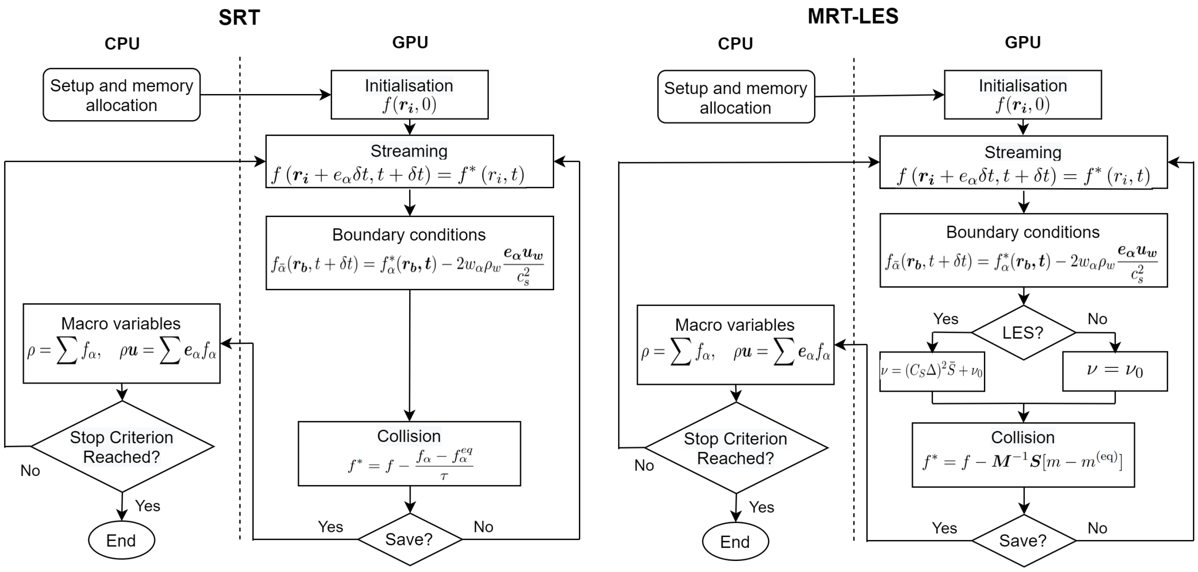

SRT
CPU
GPU
Setup and memory allocation
Initialisation $f(\boldsymbol{r_i},0)$
Streaming $f\left(\boldsymbol{r_i}+e_\alpha\delta t,t+\delta t\right)=f^*\left(r_i,t\right)$
Boundary conditions $f_{\bar\alpha}(\boldsymbol{r_b},t+\delta t)=f^*_\alpha(\boldsymbol{r_b},\boldsymbol{t})-2w_\alpha\rho_w\frac{\boldsymbol{e_\alpha u_w}}{c_s^2}$
Macro variables $\rho=\sum f_\alpha,\quad \rho\boldsymbol{u}=\sum \boldsymbol{e}_\alpha f_\alpha$
Stop Criterion Reached?
No
Yes
End
Collision $f^*=f-\frac{f_\alpha-f_\alpha^{eq}}{\tau}$
Yes
Save?
No
MRT-LES
CPU
GPU
Setup and memory allocation
Initialisation $f(\boldsymbol{r_i},0)$
Streaming $f\left(\boldsymbol{r_i}+e_\alpha\delta t,t+\delta t\right)=f^*\left(r_i,t\right)$
Boundary conditions $f_{\bar\alpha}(\boldsymbol{r_b},t+\delta t)=f^*_\alpha(\boldsymbol{r_b},\boldsymbol{t})-2w_\alpha\rho_w\frac{\boldsymbol{e_\alpha u_w}}{c_s^2}$
Yes
LES?
No
$\nu=(C_S\Delta)^2\bar{S}+\nu_0$
$\nu=\nu_0$
Macro variables $\rho=\sum f_\alpha,\quad \rho\boldsymbol{u}=\sum \boldsymbol{e}_\alpha f_\alpha$
Stop Criterion Reached?
No
Yes
End
Collision $f^*=f-\boldsymbol{M}^{-1}\boldsymbol{S}[m-m^{(\mathrm{eq})}]$
Yes
Save?
No

Figure 3: Schematic algorithm of this study's implementation of the three models.

Before a kernel can be launched, the grid that organises the threads must be defined. In this study, one-dimensional blocks along the x-direction were used. In this configuration, nodes along the x-direction are updated in consecutive order, which, as will be discussed later, has an important impact on the memory access pattern. The resulting dimensions of the grid is thus N_y rows and N_x/block_size columns. Figure 4a illustrates the described thread organisation. For hardware purposes, the block size must be set to a multiple of 32, which is the number of threads executed simultaneously in current devices. The present study sets it to 32, as larger block sizes did not increase performance, as there is weak interaction between blocks for LBM. Note that this thread organisation requires the x-domain size to also be a multiple of 32.

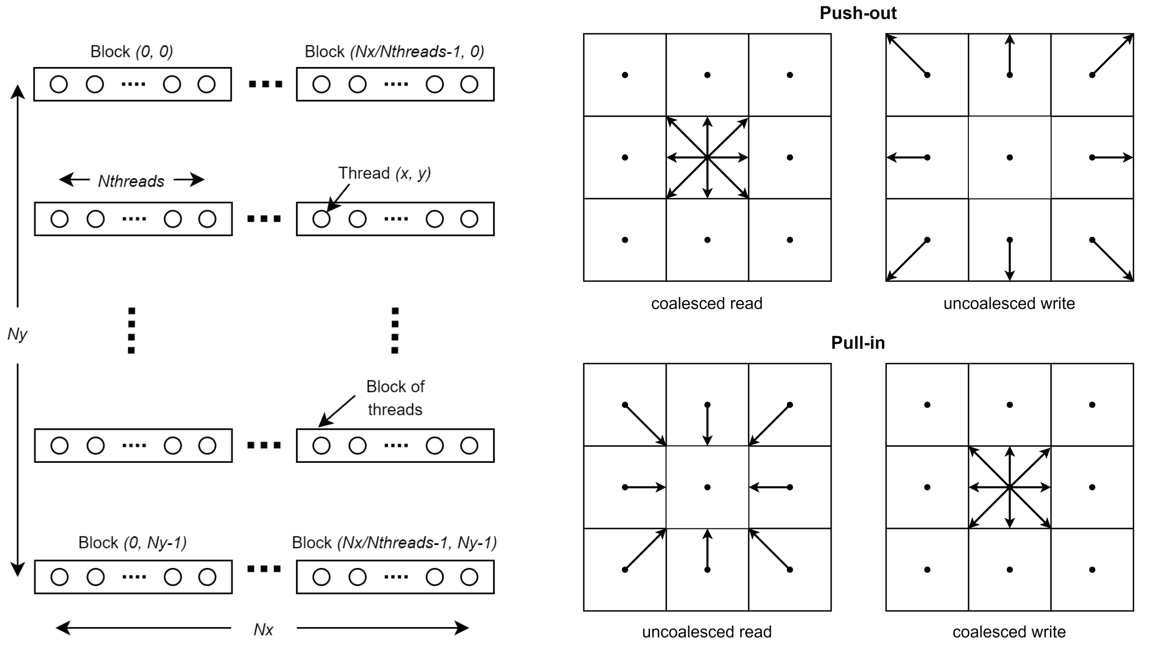
Listing 1 shows how the thread organisation and indexing can be implemented in code. The grid dimensions are set up in the main function before launching a kernel on each thread. The thread's 2D position on the grid and the corresponding array index are determined in the kernel, using indexing described in the subsequent section.

```

__global__ void kernel(int Nx, int Ny, int a, float* f_arr)
{
    // compute 2D position of given thread
    int x = blockDim.x*blockIdx.x + threadIdx.x;
    int y = blockIdx.y;
    // compute 1D index of 3D array
    int index = Nx*Ny*a + Nx*y + x
    // access index (x, y, a) in array
    f_arr[index] = ...
}

int main()
{
    // set up dimensions of grid
    dim3  grid(Nx/block_size, Ny, 1);
    // define number of threads in a block
    dim3  threads(block_size, 1, 1);
    // launch kernel on separate threads
    kernel<<< grid, threads >>>(Nx, Ny, a, f_arr);
}
```

Listing 1: Example of indexing and thread organisation. Here block_size is set to 32.



(a) Diagram illustrating the thread organisation on the GPU.

(b) Diagram illustrating the two different methods to read and write to neighbour nodes. Here the pull-in method was found to be faster.

Figure 4: Implementation details

3.2 GPU Optimisation

It is well established that accessing the distributions in memory usually takes longer than the required computations in the LBM algorithm, making the memory bandwidth the limiting factor (Tölke 2008). Therefore, efficient memory access and minimising global memory are crucial to optimise performance. An efficient method to reduce the global memory access is to combine the streaming and collision step into one kernel, such that only one memory read at the streaming step and one memory write at the collision step are needed. This method was utilised in the present study by storing the distributions from the streaming step into temporary local variables, applying boundary conditions and finally writing the temporary variable into global memory through the collision step. The calculation of the equilibrium distributions and collisions are performed in a combined operation, such that the equilibrium distributions do not need to be stored. Furthermore, as transferring data between the device to host has a high latency on GPUs, the macroscopic properties are only written to global memory and transferred to the host when needed.

The data layout used to store the distributions is vital for achieving coalesced memory access. Coalesced memory access is achieved when consecutive threads access consecutive locations in memory, such that all the memory accesses can, at the hardware level, be combined into a single request. One common data layout is the Array of Structures (AoS), where the distributions at each node are structured together. This method is often used in CPU implementations, where the single core is accessing consecutive nodes and allowing the distributions to be loaded into the low-latency cache. However, as GPUs uses the SIMD execution model, the parallel threads are accessing strided memory locations, resulting in uncoalesced memory access. Using Structure of Arrays (SoA) instead, where all the corresponding distributions for the different nodes are grouped together, consecutive threads access consecutive memory patterns. The access pattern for the two different layouts are illustrated in Figure 5. SoA can be achieved by storing the distributions with the index $(N_x \times N_y \times a + N_x \times y + x)$. In the present study, the distributions are stored in single precision to maximise performance, as double-precision hardware is still limited on GPUs, except for recent high-end models. Changing from

AoS to SoA resulted in a speedup factor of 2.1 for the present implementation.

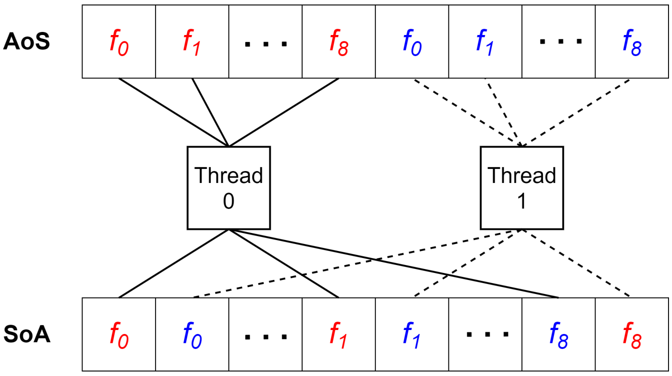


Figure 5: Access patterns for Array of Structures (AoS) and Structure of Arrays (SoA), using the Single Instruction Multiple Data (SIMD) execution model.

Achieving perfect coalesced memory access is not possible since the streaming step requires access to neighbouring nodes. Therefore, it is of essence to select the optimal streaming scheme. In LBM, there are mainly two methods to perform the streaming step (Wellein et al. 2006), the push-out method that reads the distributions at the current node and streams the distributions outwards, resulting in a coalesced read and uncoalesced write. Whereas the pull-in method reads the incoming distributions from the neighbouring nodes and updates the distributions at the current node, resulting in an uncoalesced read and coalesced write. The two different methods are illustrated in Figure 4b. Since uncoalesced read-access is quicker than uncoalesced write-access, using the pull-in scheme is preferred. Changing to the pull-in method increased the performance by 4 % in the present implementation. The GPU used here has the Pascal architecture, which caches global memory access. Thus the performance gain is limited, but a more significant increase in performance is expected for older architectures.

3.3 Code Metadata

The project was built using the open-source, cross-platform build system CMake, making it compatible with most platforms (Cedilnik et al. 2021). It was developed and tested primarily on a Linux Ubuntu 20.04 installation.

The serial CPU code uses C++ 11 language standard, whereas the parallel GPU code uses NVIDIA's CUDA language, which requires a CUDA-enabled GPU and a working installation of the CUDA toolkit (NVIDIA et al. 2021). If no CUDA enabled GPU are available, a CPU only target can be built. The software was developed and tested on a GPU device with compute capability 6.1 and the CUDA toolkit 11.4 environment. The compute capability is by default set to 6.1 but can be changed by the user. A CMake version of 3.18, or newer, is required to utilise the CUDA integration of CMake.

At compile time, the collision model and if using LES must be specified, which is by default set to the MRT-LES model. The input parameters of the simulation, such as grid size, iterations and Reynolds number, are specified in input files, and no recompilation is required for changing the parameters. A complete description of the compiler options and input files can be found in the README on the GitHub repository, linked at the top of the paper.

The file format vtk was used for postprocessing, which can be read by postprocessing software such as the open-source application Paraview (Ayachit et al. 2021). To limit the dependencies in the project, a simple file writer was implemented, which outputs the velocity field in the *vtk* format. It was not optimised as file I/O time was negligible. For larger file I/O, the write time can easily be reduced by using third-party software such as *The Visualization Toolkit (VTK)* (Schroeder et al. 2006).

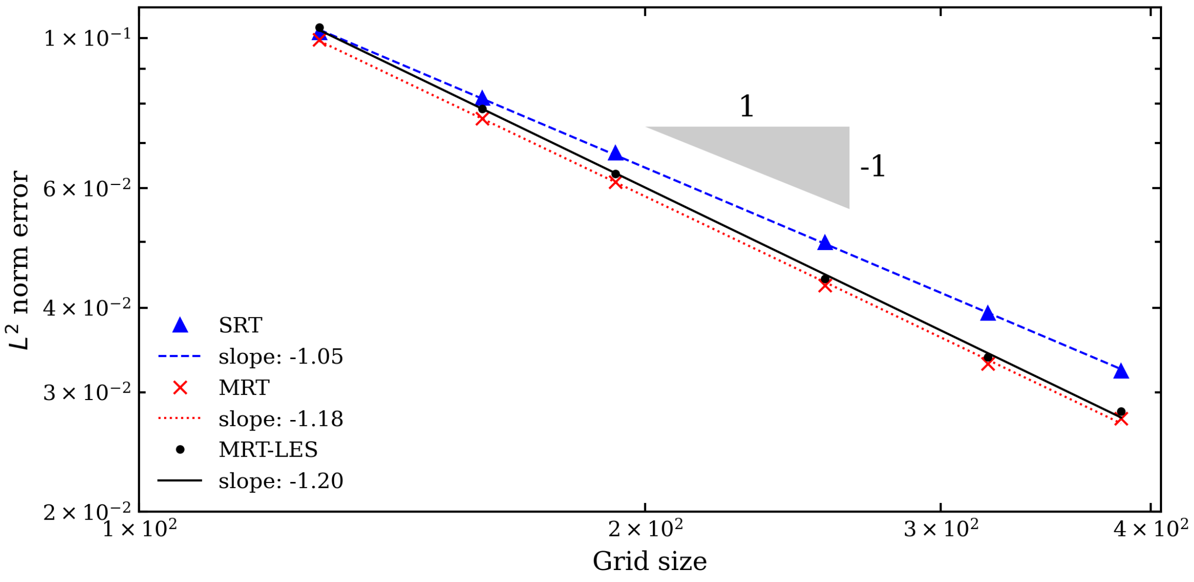


Figure 6: Logarithmic error plot, comparing the SRT and MRT models to data from Erturk et al. (2005) for Re 1000.

4 Numerical results and Discussion

4.1 Validation

The implementation was validated against numerical solutions for incompressible steady lid-driven cavity flow from Erturk et al. (2005). The numerical solutions are generated from iterative solutions to the N-S equations using a 601×601 grid. An interesting point about these results is that steady solutions are obtained for Reynolds numbers up to 21 000, whereas other studies found unsteady solutions for Re $\geq$ 10 000 (Zhen-Hua et al. 2006, Chen 2009).

As pointed out earlier, the LBM can recover N-S equations to second-order spatial accuracy. However, since the standard bounce-back boundary conditions have been used, spatial accuracy closer to first-order is expected. To verify this, the numerical results are compared to Erturk et al. (2005) at Re 1000, using the sum of the L^2 norm error along the vertical and horizontal centerline. Since the error is calculated from a numerical solution and not an analytical solution, the grid size was limited to coarse grids, below the grid size used to generate the numerical solutions. The errors for different grid sizes are plotted in Figure 6, which demonstrates close to first-order accuracy for all three models, with the MRT based models achieving a higher order of accuracy at 1.2 compared to 1.05 for the SRT model. The MRT-LES model has similar errors to the MRT model, as the effect of the LES model is limited for the laminar flow at Re 1000.

4.2 Lid-Driven Cavity Simulations

To evaluate the accuracy of the models, lid-driven cavity flows at from Re 1000 to 20 000 were considered. The range was set to evaluate the accuracy at both steady laminar flows and unsteady flows and allow comparisons to the data from Erturk et al. (2005). A square grid of size 512×512 was used, that sets the reference length in lattice units to $L_{ref} = 511$. To ensure stability, the lattice Mach number was kept constant at 0.1, such that the reference velocity in lattice units is given by $U_{lid} = c_s Ma = 0.0577$. The Reynolds number is defined as $Re = U_{lid} L_{ref}/\nu$, and controls the characteristic properties of the simulation. To obtain the specified Reynolds number, the kinematic viscosity ν, and thus the relaxation time τ, is altered. With a grid size of 512, the SRT model was only stable for Re $\leq$ 5000, due to the expected instabilities as $\tau \to 1/2$.

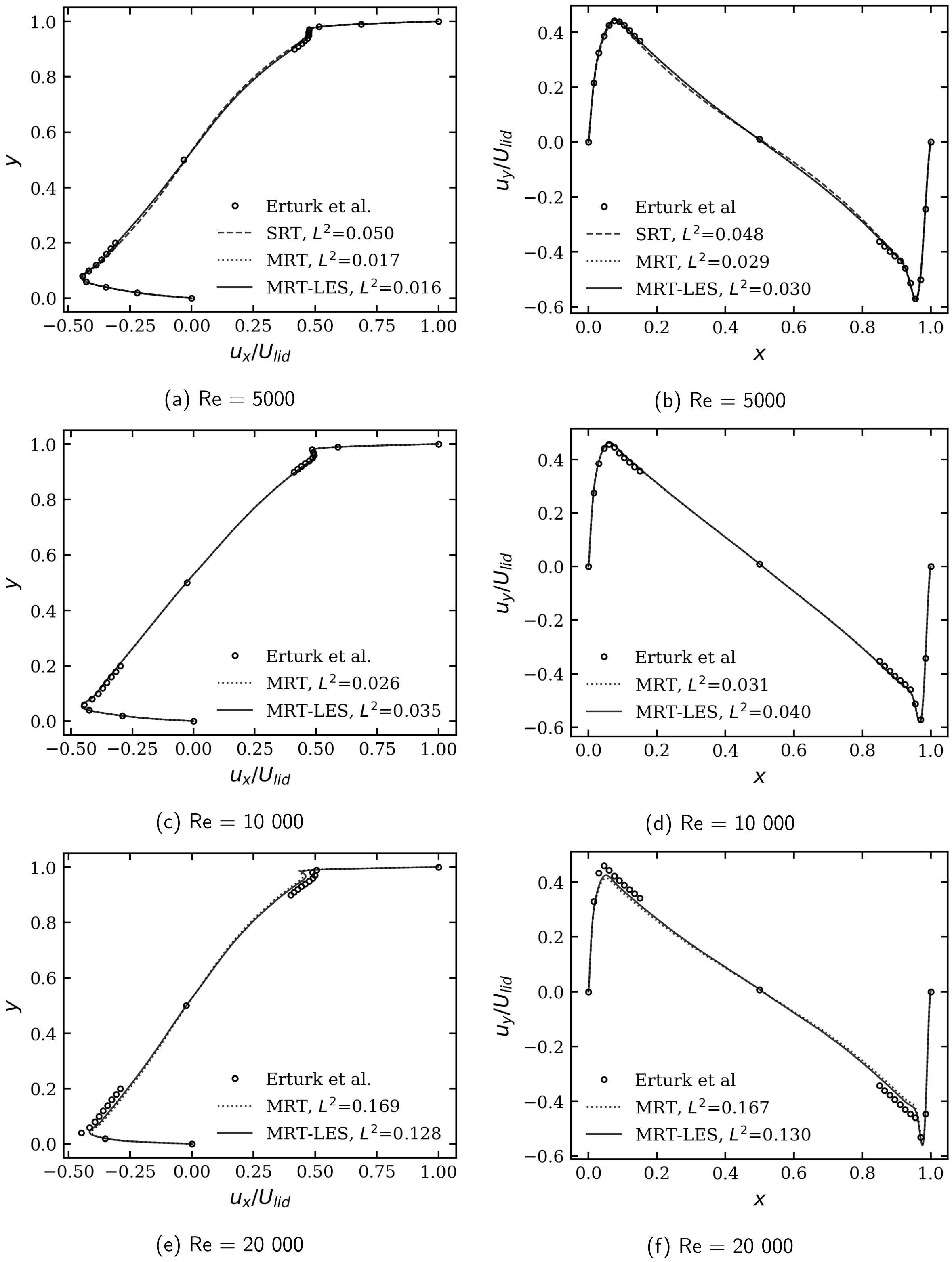
For Re 1000 and 5000, 1M timesteps were used to ensure that the flow converged to a steady solution. At Re $\geq$ 10 000, the flow develops unsteady periodic behaviour. A potential source of the periodic behaviour is numerical oscillations that can occur at the corner of the cavity. If this is the case, increasing the grid size should reduce these oscillations. However, numerical experiments showed that the flow at Re 20 000 was unsteady using a 1024×1024 grid as well, indicating a changing flow from laminar to turbulent. To allow a study of the time-dependent unsteady flows at Re $\geq$ 10 000, the flow was averaged over a time range that converges to a steady solution. In the present study, a time-average starting after timestep number 2M and using the output at every 3000th timestep until a total of 5M timestep was used.

The accuracy of the models is evaluated by comparing velocity profiles along the centerlines, to the data by Erturk et al. (2005). Figure 7 shows the velocity profiles for Re 5000, 10 000 and 20 000 and quotes the L^2 norm error from the literature data. For Re 5000, the numerical solution for all three models agrees well with literature. The two MRT based models have equal accuracy for the laminar flow, while the SRT model's errors are 3 and 1.6 times larger for u_x and u_y, respectively. At Re 10 000, the MRT and MRT-LES models agree reasonably well with literature. The MRT-LES model predicts slightly higher speeds than the MRT model and literature, giving a 30 % higher error. Finally, for the Re 20 000 velocity profiles, both models can capture the velocity profile from literature but consistently predicts lower speeds than literature. Again, the MRT-LES model predicts higher speeds than the MRT model, which in this case yields a 23 % lower error.

For the range of Reynolds numbers studied here, the flow develops from a steady laminar flow into an unsteady and partially turbulent flow. To study the flow characteristic of the different regimes, streamlines were calculated for the four cases, using the time-averaged result of the MRT-LES model described above. The results are shown in Figure 8. At Re $\leq$ 5000, the flow is steady and laminar. In addition to the primary vortex at the centre, a secondary vortex is present in both bottom corners. At Re $\geq$ 5000, a new vortex appears in the top left corner. At Re $\geq$ 10 000, a second small vortex emerges in the right bottom corner. Finally, at Re 20 000, a second vortex forms at the left bottom corner and a second tertiary vortex in the top left corner. All these characteristic observations agree with the numerical solution obtained by Erturk et al. (2005). Figure 8d indicates that the turbulent behaviour emerges from the bottom right corner, which is likely the source of the unsteady behaviour. It is interesting to note that some regions remain laminar while being turbulent in other regions.

A more quantitative assessment of the vortex positions can be conducted by comparing the centre position of the vertices to literature. For clarity, Table 1 reports the L^2 norm of the distance to the vortex centre. The vortex positions can be found in Appendix B. Table 1 demonstrates that all three models agree well with reference for Re $\leq$ 5000. Moreover, all models considered predict the position of the centre vortex with great accuracy at all Reynolds numbers. The left and right vortex positions is reasonably well predicted for the unsteady flows at Re $\geq$ 10 000, except for being slightly shifted in the positive y-direction. This shift might indicate an excessive numerical diffusion at the corner nodes. From Figure 7e, the magnitude of the x-velocity is underestimated along the bottom wall and top wall after leaving the system of vertices at the corners, which again supports an excessive numerical diffusion. Overall, the MRT-LES model was found most accurate and could recreate all the characteristic behaviour from literature.

The excess numerical diffusion observed at high Reynolds numbers are likely due to the first-order accurate boundary condition used in the present study. Although the standard bounce-back offers a simple implementation and stable numerical results, a higher-order method is likely needed to study flows at higher Reynolds numbers. Several second-order accurate methods for determining the unknown distributions exist, but as discussed in Section 2.5, can introduce instabilities at high Reynolds numbers. More advanced boundary treatments for high Reynolds numbers exist in literature, such as using non-equilibrium extrapolation (Zhen-Hua et al. 2006) and using a regularised scheme to solve a system of equations (Hegele et al. 2018), but these were not tested in the present study.



(a) Re = 5000

(b) Re = 5000

(c) Re = 10 000

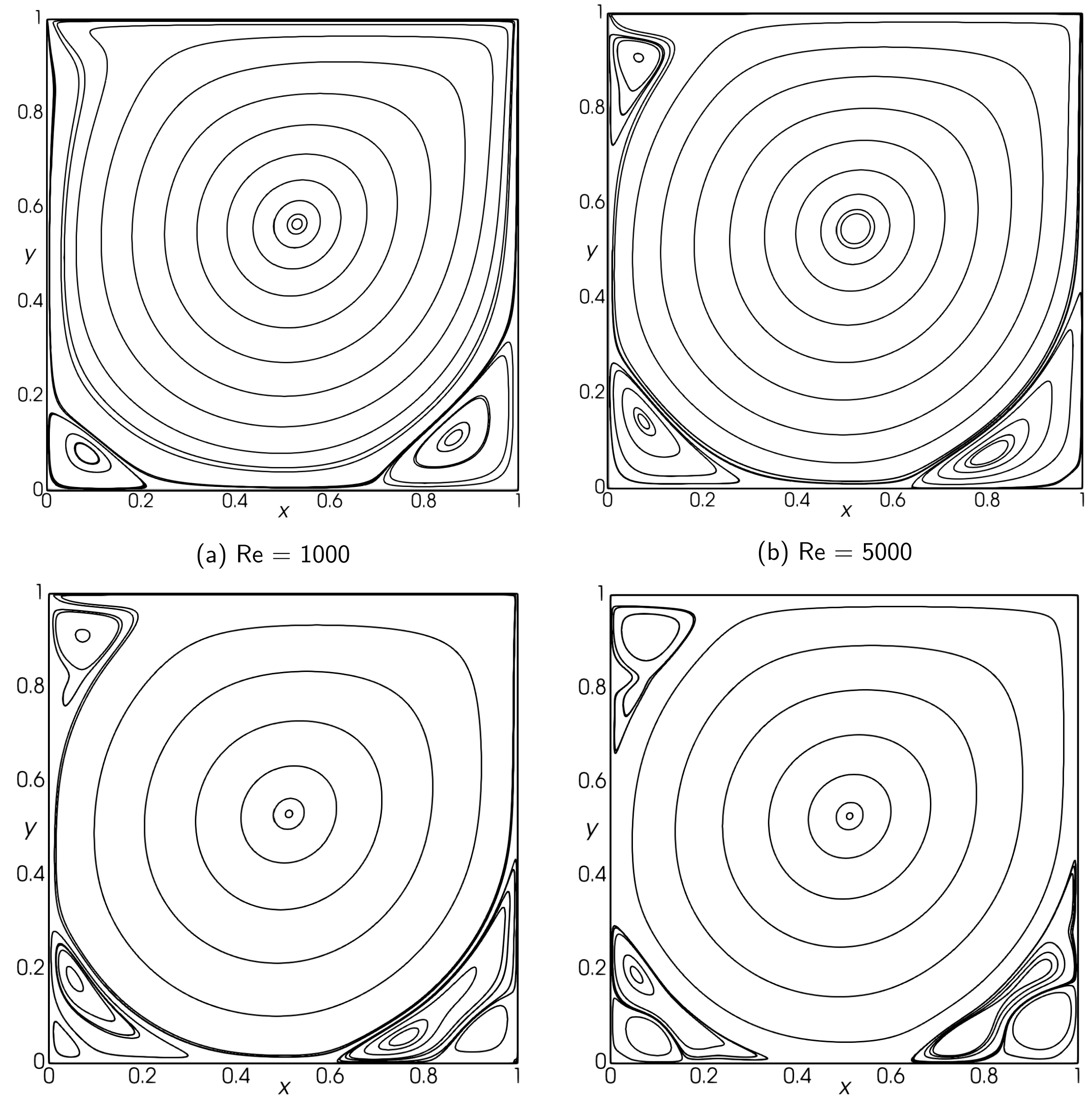
(d) Re = 10 000

(e) Re = 20 000

(f) Re = 20 000

Figure 7: Velocity profiles

An important role of applying the LES model is to increase the stability at higher Reynolds numbers. Although the MRT model has superior numerical stability to the SRT model, instabilities may arise as $\tau \to 1/2$. Adding the LES model increases the relaxation time at regions of high fluctuations and thus increases the stability. For a 128×128 grid, MRT encounters negative densities at around Re 100 000, while MRT-LES are stable up to Re 250 000. Results for Re $>$ 20 000 are not presented



(a) Re = 1000

(b) Re = 5000

(c) Re = 10 000

(d) Re = 20 000

Figure 8: Streamlines at different Reynolds numbers, using the MRT-LES model.

Re	model	centre vortex	left vortex	right voretx
1000	SRT	0.001	0.001	0.002
	MRT	0.001	0.002	0.002
	MRT-LES	0	0	0
5000	SRT	0.002	0.001	0.002
	MRT	0.004	0.005	0.001
	MRT-LES	0.004	0.004	0.003
10000	MRT	0.000	0.015	0.016
	MRT-LES	0.000	0.015	0.003
20000	MRT	0.003	0.009	0.035
	MRT-LES	0.001	0.006	0.029

Table 1: L^2 norm of the distance to the vortex centre.

here due to the accuracy limitations at high Reynolds numbers pointed out and limited numerical results in literature. The Smagorinsky model was used in the present study, which has the adjustable Smagorinsky constant, C_s. Numerical experiments indicated that C_s has a significant impact on the time-averaged numerical solution at Re $\geq$ 10 000. A larger C_s increases the magnitude of the velocities, which results in a higher accuracy at Re 20 000 but lower at Re 10 000. A potential mechanism behind this effect is that higher C_s corresponds to a larger eddy viscosity (i.e. larger relaxation time), allowing a greater momentum transfer. In the present study, C_s was kept constant at 0.1 to balance accuracy and stability, but further work on evaluating its impact at high Reynolds numbers is needed.

4.3 GPU performance

A key motivation of using the LBM is the computational and parallel capabilities. It is therefore essential to evaluate the performance of the models and GPU implementation. To compare the CPU and GPU code performance, a benchmark test was completed for the three models. The test used the lid-driven cavity simulation, a domain of 512x512 and a Reynolds number of 1000. The results were averaged over 10^4 iterations on the CPU and 10^5 iterations on the GPU. As the main interest is on running the simulation and not I/O, the performance test was completed without writing the macroscopic properties to file. The numerical experiments for the CPU version were generated with an Intel Core i7-8650U Processor. The GPU experiments were conducted on an NVIDIA GeForce GTX 1060 6 GB GPU, with 1280 CUDA cores and Pascal architecture.

The results will be compared to the memory bandwidth to evaluate the GPU implementation's efficiency. The computational performance is here quantified using the common metric of million lattice updates per second (MLups), defined as:

$$MLups = \frac{\text{iterations} \times \text{grid size}}{\text{runningtime} \times 10^6} \tag{18}$$

The GPU used has a listed theoretical memory bandwidth of $192\,GBs^{-1}$, which is calculated from the number of memory buses and the frequency of the memory transfers. In practice, the effective bandwidth will be lower. Following the approach in Delbosc et al. (2014), the performance will instead be compared to the measured effective memory bandwidth, which for the GPU used here, was measured to $146.6\,GBs^{-1}$. The maximum effective performance of the LBM model can be calculated by realising that each node requires nine reads and writes per iteration, which for single-precision (4 bytes) gives a memory requirement per node of $4 \cdot 9 \cdot 2 = 72$ bytes. Thus the maximum effective performance of the model can be obtained as $146.6\,GBs^{-1}/(72B) = 2036$ *MLups*.

The performance results are summarised in Table 2. The significant increase in performance from the SRT to the MRT model, was not expected as the MRT model has been reported to have around a 15 % decrease in performance to the SRT model (D'Humières et al. 2002). To investigate this result, the code was profiled by running separate parts of the code in isolation. For streaming, boundary conditions and collisions, the time spent on each individual part as a fraction of the sum of the three individual parts amounted to 70 %, 15 %, 15 % for the SRT model and 68 %, 15 %, 18 % for the MRT model. The streaming step for the MRT model, had an equal running time to the full simulation, which was not the case for the SRT model. This indicates that the MRT model effectively makes us of the shared memory, and is mostly limited by the memory broadband when writing the cached result in shared memory to global memory. The fact that the MRT CPU code is 25 % slower than the SRT CPU code, when using the same collision equations, further supports this argument. Further optimisation of memory access in the SRT GPU implementation is required to better utilise the shared memory on the GPU, but was not prioritised in the current implementation due to the stability issues of the SRT model.

The MRT model achieves 96 % of the effective maximum performance of the GPU, which indicates that a high degree of coalesced memory access is achieved and that the shared memory is efficiently used to cache the calculation before writing to global memory. Adding LES to the MRT model decreases the performance by 31 % due to the extra computations required. For the MRT-LES model, the GPU offers a speedup of 79x, demonstrating the parallel efficiency of the LBM. The lower speedup factor for the SRT model, again indicates that the SRT model is not efficiently implemented on the GPU.

Model	CPU (MLUPS)	GPU (MLUPS)	Speedup
SRT	35	1153	$33X$
MRT	26	1960	$75X$
MRT-LES	17	1350	$79X$

Table 2: Performance comparison of different models on a CPU and GPU.

5 Conclusion

The simulation of turbulent fluid flows is a computationally intensive task that is of great interest to applications across several fields. The LBM offers an attractive approach to fluid simulation that has superior parallel capabilities to conventional methods. This study aimed to evaluate the LBM for simulating high Reynolds number flows, using three different approaches: SRT and MRT, with and without, a Smagorinsky model LES. The numerical computation was accelerated using a GPU via NVIDIA's CUDA framework. Since the LBM algorithm is mainly memory-bound, the global memory accesses were minimised, and the memory access coalescence increased by using the SoA data layout and the pull-in streaming method. These optimisations resulted in speedup factors up to 79, compared to the CPU implementation and achieved 96 % of the effective memory broadband for the MRT model.

The implementation is validated using a 2D lid-driven cavity flow. At Re $\geq$ 10 000, the flow turns unsteady, and a time-averaged numerical solution was used. The results show good agreement with literature up to Reynolds numbers up to 20 000 before the first-order accurate boundary conditions affected the accuracy. The study demonstrated the MRT models' superior stability and found LES to increase the accuracy and stability at high Reynolds numbers. Further work is needed to consider higher-order boundary condition schemes, that are stable at high Reynolds numbers. In addition, to the effect of the Smagorinsky constant on the accuracy of the model.

References

Argyropoulos, C. D. & Markatos, N. C. (2015), 'Recent advances on the numerical modelling of turbulent flows', *Applied Mathematical Modelling* **39**(2), 693–732. 2

Ayachit, U., Cedilnik, A., Hoffman, B., King, B., Martin, K. & Neundorf, A. (2021), 'ParaView: An End-User Tool for Large Data Visualization, release: 5.9.1'. 10

Bailey, P., Myre, J., Walsh, S. D. C., Lilja, D. J. & Saar, M. O. (2009), Accelerating Lattice Boltzmann Fluid Flow Simulations Using Graphics Processors, *in* '2009 International Conference on Parallel Processing', pp. 550–557. 2

Bemaschi, M., Fatica, M., Melchionna, S., Succi, S. & Kaxiras, E. (2010), 'A flexible high-performance Lattice boltzmann GPU code for the simulations of fluid flows in complex geometries', *Concurrency Computation Practice and Experience* **22**(1), 1–14. 2

Bhatnagar, P. L., Gross, E. P. & Krook, M. (1954), 'A model for collision processes in gases. I. Small amplitude processes in charged and neutral one-component systems', *Physical Review* **94**(3), 511–525. 2

Cedilnik, A., Hoffman, B., King, B., Martin, K. & Neundorf, A. (2021), 'CMake, release: 3.21.0'. **URL:** *https://cmake.org* 10

Chen, H., Chen, S. & Matthaeus, W. H. (1992), 'Recovery of the Navier-Stokes equations using a lattice-gas Boltzmann method', *Physical Review A* **45**(8), R5339. 2, 3

Chen, S. (2009), 'A large-eddy-based lattice Boltzmann model for turbulent flow simulation', *Applied Mathematics and Computation* **215**(2), 591–598. 11

Coreixas, C., Chopard, B. & Latt, J. (2019), 'Comprehensive comparison of collision models in the lattice Boltzmann framework: Theoretical investigations', *Physical Review E* **100**, 33305. 2

Delbosc, N., Summers, J. L., Khan, A. I., Kapur, N. & Noakes, C. J. (2014), 'Optimized implementation of the Lattice Boltzmann Method on a graphics processing unit towards real-time fluid simulation', *Computers & Mathematics with Applications* **67**(2), 462–475. 2, 15

D'Humières, D., Ginzburg, I., Krafczyk, M., Lallemand, P. & Luo, L.-S. (2002), 'Multiple-relaxation-time lattice Boltzmann models in three dimensions', *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences* **360**(1792), 437–451. 5, 15

D'Humieres, D., Shizgal, B. D. & Weaver, D. P. (1992), Generalized Lattice-Boltzmann Equations, 18th International symposium, Rarefied gas dynamics, *in* 'Progress in Astronautics and Aeronautics, Rarefied gas dynamics, 18th International symposium, Rarefied gas dynamics', Vol. 159, AIAA;, Washington, DC, pp. 450–458. 4

Erturk, E., Corke, T. C. & Gökçöl, C. (2005), 'Numerical solutions of 2-D steady incompressible driven cavity flow at high Reynolds numbers', *International Journal for Numerical Methods in Fluids* **48**(7), 747–774. 11, 12

Ginzburg, I. & D'Humières, D. (2003), 'Multireflection boundary conditions for lattice Boltzmann models', *Phys. Rev. E* **68**(6), 66614. 6

Hegele, L. A., Scagliarini, A., Sbragaglia, M., Mattila, K. K., Philippi, P. C., Puleri, D. F., Gounley, J. & Randles, A. (2018), 'High-Reynolds-number turbulent cavity flow using the lattice Boltzmann method', *Physical Review E* **98**(4). 12

Higuera, F. J. & Jiménez, J. (1989), 'Boltzmann approach to lattice gas simulations', *EPL* **9**(7), 663–668. 3

Hou, S., Sterling, J., Chen, S. & Doolen, G. (1995), 'A lattice Boltzmann subgrid model for high Reynolds number flows', *Pattern Formation and Lattice gas Automata* pp. 151–166. 2

Krafczyk, M., Tölke, J. & Luo, L. S. (2003), 'Large-eddy simulations with a multiple-relaxation-time LBE model', *International Journal of Modern Physics B* **17**(1-2), 33–39. 2, 5, 18

Krüger, T., Kusumaatmaja, H., Kuzmin, A., Shardt, O., Silva, G. & Viggen, E. M. (2017), *The Lattice Boltzmann Method*, Graduate Texts in Physics, Springer International Publishing, Cham. 6

Lallemand, P. & Luo, L.-S. (2000), 'Theory of the lattice Boltzmann method: Dispersion, dissipation, isotropy, Galilean invariance, and stability', *Phys. Rev. E* **61**(6), 6546–6562. 2, 4, 5, 18

Latt, J., Chopard, B., Malaspinas, O., Deville, M. & Michler, A. (2008), 'Straight velocity boundaries in the lattice Boltzmann method', *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics* **77**(5), 056703. 6

Li, W., Wei, X. & Kaufman, A. (2003), 'Implementing lattice Boltzmann computation on graphics hardware', *Visual Computer* **19**(7-8), 444–456. 2

NVIDIA, Vingelmann, P. & Fitzek, F. H. P. (2021), 'CUDA, release: 11.4.48'.
URL: *https://developer.nvidia.com/cuda-toolkit* 10

Obrecht, C., Kuznik, F., Tourancheau, B. & Roux, J. J. (2011), 'A new approach to the lattice Boltzmann method for graphics processing units', *Computers and Mathematics with Applications* **61**(12), 3628–3638. 2

Qian, Y. H., D'Humières, D. & Lallemand, P. (1992), 'Lattice bgk models for navier-stokes equation', *EPL* **17**(6), 479–484. 3

Schroeder, W., Martin, K. & Lorensen, B. (2006), 'The Visualization Toolkit (4th ed.)'. 10

Smagorinsky, J. (1963), 'General Circulation Experiments with the Primitive Equations', *Monthly Weather Review* **91**(3), 99. 5

Tölke, J. (2008), 'Implementation of a Lattice Boltzmann kernel using the Compute Unified Device Architecture developed by nVIDIA', *Comput Visual Sci* **13**, 29–39. 2, 9

Wellein, G., Zeiser, T., Hager, G. & Donath, S. (2006), 'On the single processor performance of simple lattice Boltzmann kernels', *Computers & Fluids* **35**(8-9), 910–919. 10

Yu, H., Luo, L.-S. & Girimaji, S. S. (2006), 'LES of turbulent square jet flow using an MRT lattice Boltzmann model', *Computers & Fluids* **35**(8), 957–965. 2

Zhen-Hua, C., Bao-Chang, S. & Lin, Z. (2006), 'Simulating high Reynolds number flow in two-dimensional lid-driven cavity by multi-relaxation-time lattice Boltzmann method', *Chinese Physics* **15**(8), 1855. 2, 11, 12

Zou, Q. & He, X. (1997), 'On pressure and velocity boundary conditions for the lattice Boltzmann BGK model', *Physics of Fluids* **9**(6), 1591–1596. 6

A Calculating the local strain rate for D2Q9

From Krafczyk et al. (2003), the components of tensor Q is given by the equation

$$\mathrm{Q}_{mn} \equiv \frac{1}{3}\delta\rho\delta_{mn} + j_m j_n - \mathrm{P}_{mn}, \quad m, n \in \{x, y\}. \tag{19}$$

The second-order moments, P_{ij}, of the distribution function is given by

$$\mathrm{P}_{ij} = \sum_{\alpha} e_{\alpha i} e_{\alpha j} f_{\alpha}. \tag{20}$$

where $e_{\alpha i}$ is the discrete velocity of the i-th component. To obtain the second-order monomials, $\{e_{\alpha i} e_{\alpha j} \mid i, j \in \{x, y\}\}$, consider the orthogonal basis vectors for the MRT D2Q9 velocity model, given by Lallemand & Luo (2000):

$$
\begin{aligned}
|\rho\rangle_\alpha &= |\mathbf{e}_\alpha|^0 = 1, \\
|e\rangle_\alpha &= -4\,|\mathbf{e}_\alpha|^0 + 3\left(e_{\alpha,x}^2 + e_{\alpha,y}^2\right) \\
|\varepsilon\rangle_\alpha &= 4\,|\mathbf{e}_\alpha|^0 - \frac{21}{2}\left(e_{\alpha,x}^2 + e_{\alpha,y}^2\right) + \frac{9}{2}\left(e_{\alpha,x}^2 + e_{\alpha,y}^2\right)^2 \\
|j_x\rangle_\alpha &= e_{\alpha,x}, \\
|q_x\rangle_\alpha &= \left[-5\,|\mathbf{e}_\alpha|^0 + 3\left(e_{\alpha,x}^2 + e_{\alpha,y}^2\right)\right] e_{\alpha,x} \\
|j_y\rangle_\alpha &= e_{\alpha,y}, \\
|q_y\rangle_\alpha &= \left[-5\,|\mathbf{e}_\alpha|^0 + 3\left(e_{\alpha,x}^2 + e_{\alpha,y}^2\right)\right] e_{\alpha,y} \\
|p_{xx}\rangle_\alpha &= e_{\alpha,x}^2 - e_{\alpha,y}^2 \\
|p_{xy}\rangle_\alpha &= e_{\alpha,x} e_{\alpha,y}.
\end{aligned}
\tag{21}
$$

Projecting the second-order monomials to these vectors, we obtain the expressions

$$
\begin{aligned}
e_{\alpha x}^2 &= \frac{1}{6}\left(4|\rho\rangle_\alpha + |e\rangle_\alpha + 3\,|p_{xx}\rangle_\alpha\right), \\
e_{\alpha x}^2 &= \frac{1}{6}\left(4|\rho\rangle_\alpha + |e\rangle_\alpha - 3\,|p_{xx}\rangle_\alpha\right), \\
e_{\alpha x} e_{\alpha y} &= |p_{xy}\rangle_\alpha
\end{aligned}
\tag{22}
$$

thus the componets of Q can be calculated directly from the moments using:

$$
\begin{aligned}
\mathrm{P}_{xx} &= \frac{1}{6}\left[(4\rho + e + 3p_{xx}\right] \\
\mathrm{P}_{yy} &= \frac{1}{6}\left[(4\rho + e - 3p_{xx}\right] \\
\mathrm{P}_{xy} &= \mathrm{P}_{yx} = p_{xy}.
\end{aligned}
\tag{23}
$$

B Vortex positions

Re	model	x_c	y_c	x_l	y_l	x_r	y_r
1000	SRT	0.530	0.566	0.084	0.078	0.863	0.114
	MRT	0.530	0.566.	0.084	0.080	0.863	0.114
	MRT-LES	0.530	0.565	0.083	0.078	0.863	0.112
	Erturk et al.	0.5300	0.5650	0.0833	0.0783	0.8633	0.1117
5000	SRT	0.517	0.534	0.072	0.137	0.807	0.074
	MRT	0.513	0.532	0.076	0.133	0.806	0.074
	MRT-LES	0.513	0.532	0.074	0.141	0.802	0.072
	Erturk et al.	0.5150	0.5350	0.0733	0.1367	0.8050	0.0733
10000	MRT	0.512	0.530	0.055	0.178	0.761	0.055
	MRT-LES	0.512	0.530	0.054	0.178	0.779	0.059
	Erturk et al.	0.5117	0.5300	0.0583	0.1633	0.7767	0.0600
20000	MRT	0.513	0.526	0.055	0.188	0.761	0.054
	MRT-LES	0.511	0.527	0.053	0.186	0.755	0.052
	Erturk et al.	0.5100	0.5267	0.0483	0.1817	0.7267	0.0450

Table 3: Comparison of the centre position of the centre vortex (x_c, y_c), left vortex (x_l, y_l) and right vortex (x_r, y_r).