

Turbulence Simulations using the Generalized Lattice Boltzmann Equation on Massively Parallel Architectures

Sanjoy Banerjee^{1*}, Kannan N. Premnath^{1,2}, Martin J. Pattison²

¹ Department of Chemical Engineering, University of California, Santa Barbara, CA 93106, USA

² MetaHeuristics LLC, Santa Barbara, CA 93105, USA

e-mail: banerjee@engineering.ucsb.edu, nandha@metah.com, martin@metah.com

Abstract The lattice Boltzmann method (LBM) is becoming increasingly popular for the computational simulation of fluid flow. This approach is based on kinetic theory, and considers the evolution of distributions of particles on a lattice whose collective behaviour represents that of the equations governing the motion of fluids. The use of the LBM is attractive as it has a relatively fast execution speed and is highly suited to parallel implementation on multiprocessor computers. In this paper, the most advanced formulation currently available, the generalized lattice Boltzmann equation (GLBE), or multiple relaxation time model, is discussed. The results of large eddy simulations of fully developed turbulent channel flow obtained using this model are presented and shown to be in good agreement with benchmark data. Studies to compare the speed and numerical stability of the GLBE with other methods have been undertaken and clearly demonstrate its superiority over earlier versions of the LBM. Finally, results from performance tests using up to 1024 processors of a massively parallel supercomputer are provided, showing the expected near-linear scaling for large problems.

Key words: lattice Boltzmann method, multiple relaxation time, parallel computing, turbulence, CFD

INTRODUCTION

The simulation of fluid flow involves the numerical solution of the hydrodynamic equations of motion, the Navier–Stokes equations (NSE). This is commonly done through the use of finite difference or finite volume methods, in which the NSE are discretized and solved on a grid. Most schemes, such as the widely used projection method, involve the solution of a Poisson equation for the pressure field. Due to its elliptical nature, the computationally intensive Poisson equation solver, which can take as much as 80–90% of the overall computational time [1, 2], does not scale well on parallel distributed memory architectures for large numbers of processors.

An emerging alternative to traditional computational fluid dynamics (CFD) methods is the lattice Boltzmann method (LBM). This approach, which is based on kinetic theory, considers the evolution of particle population distributions on a lattice, representing discrete velocity directions [3]. The algorithm comprises two main steps, the propagation step, in which particle populations move to adjacent nodes on the lattice, and the collision step, in which the distribution function relaxes to its local equilibrium. With the choice of an appropriate model for the collision operator and with adequate lattice symmetry, the collective behavior of the populations will emulate the flow behavior represented by the NSE [4]. Most implementations of the LBM use the so-called BGK model [4, 5, 6], in which a single timescale is used in the collision operator, but in the more recently developed multiple relaxation time (MRT) model (using the *generalized* lattice Boltzmann equation, GLBE), this scalar timescale is replaced by a matrix [7, 9, 10]. The use of the MRT model as opposed to the BGK model not only enables a better representation of the physical system being modeled, but also offers significantly improved numerical stability.

The LBM does not require the solution of a Poisson equation and this enables it to run substantially faster than finite difference types of solvers (on a per time step per grid point basis). In addition, the nature of the scheme allows it to run very efficiently on parallel computers with distributed memory architectures. During each iteration, each processor only needs to communicate with those processors assigned to adja-

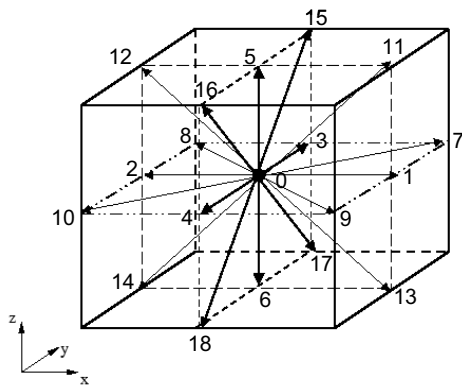
cent parts of the computational domain; if, for example, two dimensional domain decomposition is used, then each processor will need to exchange data with a maximum of eight other processors. This means that LBM based solvers can demonstrate almost linear parallel scaling on massively parallel supercomputers, even up to thousands of processors, as has been recently demonstrated for computation of magnetohydrodynamic flow on the Earth Simulator [8]. In contrast, with traditional CFD methods involving elliptic solvers, each processor must communicate with every other processor, leading to a limit on the number of processors which can usefully be employed.

In this paper, the formulation of the MRT–LBM and the optimization strategies used will be discussed. The MRT–LBM has been implemented in our MetaCFD code, a CFD package based on lattice Boltzmann methods which incorporates the latest developments in this field. Results obtained for both internal and external turbulent flows will be presented and compared with benchmark data from pseudo-spectral direct numerical simulations (DNS). A set of performance tests to investigate the execution speed and numerical stability has been undertaken, and comparisons with a finite difference solver and a BGK–LBM version of the MetaCFD code are made. The ability of the LBM to parallelize efficiently is demonstrated here by the results of tests with up to 1024 processors performed on the US NERSC’s IBM Nighthawk cluster which show almost linear scaling for LES of turbulent flow in a duct.

LATTICE BOLTZMANN METHOD

The approach taken in the LBM consists of solving the lattice Boltzmann equation for the evolution of a distribution function $f(\mathbf{x}, \mathbf{v}, t)$ of particles as they move and collide on a lattice. The solution of the equation involves two main steps, which represent streaming and collision of particle populations. Usually the collision process is represented by the Bhatnagar–Gross–Krook, or single relaxation time (SRT) model [5] where the particle populations relax to a local equilibrium state at a rate determined by a characteristic relaxation time parameter. More recently, the multiple relaxation time model has been introduced, which involves several relaxation times and proves to have substantially better numerical stability [7].

The LBE is discretised and solved on a lattice grid. The number of discrete velocity directions representing the lattice is chosen to respect certain symmetry requirements so as to recover the isotropy of the viscous stress tensor of the fluid flow [3]. In three dimensional implementations, a cubic grid is used and 15 or 19 particle velocity models are commonly used, though other models are occasionally found. In this work, the 19 velocity model was used, due to its superior numerical stability, and this is shown in Fig. 1.



$$\mathbf{e}_\alpha = \begin{cases} (0, 0, 0) & \alpha = 1 \\ (\pm 1, 0, 0), (0, \pm 1, 0), (0, 0, \pm 1) & \alpha = 2, \dots, 7 \\ (\pm 1, \pm 1, 0), (\pm 1, 0, \pm 1), (0, \pm 1, \pm 1) & \alpha = 8, \dots, 19 \end{cases}$$

Figure 1: Three dimensional, nineteen velocity (D3Q19) lattice.

The Cartesian component c of the particle velocity \mathbf{e}_α is given by $c = \delta_x / \delta_t$, where δ_x is the lattice spacing and δ_t the time step. In this paper, the convention is that Greek symbols (α, β, \dots) will be used to represent particle velocity directions and Roman symbols (i, j, k) will be used for Cartesian directions. The corresponding vector of distribution functions \mathbf{f} at a location may be written as

$$\mathbf{f} = [f_0, f_1, f_2, \dots, f_{18}]^T. \quad (1)$$

where the superscript “ T ” denote the transpose.

As previously mentioned, the distribution function is calculated using a two step procedure comprising a so-called collision step and a streaming step. In the most advanced formulation currently available, the MRT-LBM with forcing terms is used [7, 9, 10]:

$$\tilde{f}_\alpha(\mathbf{x}, t) = f_\alpha(\mathbf{x}, t) - \sum_\beta \Lambda_{\alpha\beta} (f_\beta - f_\beta^{eq}) + \sum_\beta \left(\mathbf{I}_{\alpha\beta} - \frac{1}{2} \Lambda_{\alpha\beta} \right) S_\beta \delta_t, \quad (2-a)$$

$$f_\alpha(\mathbf{x} + \mathbf{e}_\alpha \delta_t, t + \delta_t) = \tilde{f}_\alpha(\mathbf{x}, t). \quad (2-b)$$

The first term on the right hand side (RHS) of Eq. (2-a) represents the cumulative effect of particle collisions on the evolution of the distribution function f_α , and can be thought of as representing the effects of viscosity, as well as other processes. Collision is considered as a relaxation process in which f_β relaxes to its local equilibrium value f_β^{eq} at a rate determined by the relaxation time matrix $\Lambda_{\alpha\beta}$. The MRT model has a generalized collision matrix with multiple relaxation times corresponding to the underlying physics: the macroscopic fields such as densities, momentum and stress tensors are given as various kinetic moments of the distribution function. For example, collision does not alter the densities ρ and momentum $\rho \mathbf{u}$, while the stress tensors relax during collisions at rates determined by fluid properties such as the shear and bulk viscosities. Thus certain relaxation times forming components of the collision matrix $\Lambda_{\alpha\beta}$ in the MRT model are developed to reflect the underlying physics, while those which do not affect hydrodynamics are chosen to enhance the numerical stability of the approach. For more details, the reader is referred to Refs. [7, 9, 10].

The second term on the RHS of Eq. (2-a) introduces changes in the evolution of distribution function from driving forces \mathbf{F} , such as gravity, $\rho \mathbf{g}$, through a source term S_α . This source term may be written [9]:

$$S_\alpha = \frac{(e_{\alpha j} - u_j) F_j}{\rho c_s^2} f_\alpha^{eq,M}(\rho, \mathbf{u}), \quad (3)$$

where $f_\alpha^{eq,M}(\rho, \mathbf{u})$ is the local Maxwellian, which is dependent on the local density and velocity as

$$f_\alpha^{eq,M}(\rho, \mathbf{u}) = \omega_\alpha \rho \left\{ 1 + \frac{\mathbf{e}_\alpha \cdot \mathbf{u}}{c_s^2} + \frac{(\mathbf{e}_\alpha \cdot \mathbf{u})^2}{2c_s^4} - \frac{1}{2} \frac{\mathbf{u} \cdot \mathbf{u}}{c_s^2} \right\}, \quad \omega_\alpha = \begin{cases} \frac{1}{3} & \alpha = 1 \\ \frac{1}{18} & \alpha = 2, \dots, 7 \\ \frac{1}{36} & \alpha = 8, \dots, 19. \end{cases} \quad (4)$$

and $c_s = (1/\sqrt{3})c$ is the speed of sound of the model. In Eq. (2-a), $\mathbf{I}_{\alpha\beta}$ is the identity matrix.

At this point is worth noting that the commonly used SRT model uses a scalar relaxation parameter in place of the tensors $\Lambda_{\alpha\beta}$ and there is no summation for the terms on the right. Also note that the implementation of Eq. (2-a) does not involve direct summation as shown, but uses a highly optimized procedure that involves transformations into moment space and exploits certain properties of $\Lambda_{\alpha\beta}$ [7, 9, 10]. With these optimizations, it is found that despite its much greater complexity, the MRT only takes about 10–30% more CPU time than the BGK model [7, 9, 10].

Equation (2-b) is known as the advection, or streaming, step and deals with the change in the distribution function during a time interval δ_t , as the particles propagate from location \mathbf{x} to their adjacent location $\mathbf{x} + \mathbf{e}_\alpha \delta_t$, with a velocity \mathbf{e}_α along the characteristic direction α .

The local macroscopic density and velocity fields are then given by

$$\rho = \sum_{\alpha=0}^{18} f_\alpha, \quad \rho \mathbf{u} = \sum_{\alpha=0}^{18} f_\alpha \mathbf{e}_\alpha + \frac{1}{2} \mathbf{F} \delta_t, \quad (5)$$

and the pressure field p can be calculated from $p = c_s^2 \rho$.

The behaviour of the populations represented by the distribution function \mathbf{f} corresponds to that of fluid flow, and the incompressible Navier–Stokes equations can be recovered from the lattice Boltzmann equations for the case of low Mach number ($v \ll c_s$, where c_s is the speed of sound) [4, 9]. The discussion of the MRT model presented here has been relatively brief, but more detailed descriptions can be found elsewhere [7, 9, 10].

Two important points to note are that there is no pressure Poisson equation to solve, and that the solution scheme is explicit, with information required from neighbouring nodes only. The solution of the Poisson equation is time consuming and typically takes 80–90% of the CPU time in traditional CFD solvers [1]; its absence means that LBM codes are relatively fast on a per time step per grid point basis. The explicit nature of the computations means that codes based on the LBM can run very efficiently on parallel architectures, and this is one of the main motivations for its use.

In its basic form, the LBM only allows the use of cubic grids. However, it is frequently desirable to use variable grid spacing to obtain higher resolution in regions of high gradients without the penalty of large numbers of nodes in other regions. One means of achieving this within the lattice Boltzmann framework is the interpolation supplemented LBM (ISLBM) which uses variable grid spacings and involves an additional interpolation step [11]. This has proven very effective for laminar flows where there are very narrow wall layers to be resolved, as may occur in magnetohydrodynamic flows [12], and for Reynolds-averaged turbulent flows [13]. However no studies for turbulence simulations in which the time dependent motion of the eddies or equivalently fluctuations in the velocity field are resolved could be found in the literature. Indeed tests using our MetaCFD code with an ISLBM implementation yielded very poor results, probably as a result of interpolation step causing too much dissipation.

Another approach is to use local grid refinement, in which the grid remains cubic, but the size of the cells used in different regions is changed. This method has been widely used for direct numerical simulations (DNS) of turbulence [15], in contrast to the ISLBM. A number of different techniques for coupling the different regions have been formulated, and can broadly be divided into two types. One uses co-located grid [14] and the other uses a staggered arrangement [15] – the latter has been used for MetaCFD as conservation of mass and momentum at the interface can readily be enforced.

TURBULENT SIMULATIONS

A common test case used for CFD codes is fully developed turbulent flow in a channel. The case considered here involves the simulation of a flow bounded by a no-slip wall at the lower surface and a free slip boundary at the upper surface, with periodic boundaries in the other directions. A shear Reynolds number $Re_* = 184$ was used, where $Re_* = u_*H/\nu$, H is the channel depth, ν is the molecular viscosity, and u_* the friction velocity. The flow was driven by a pressure gradient, and this can be related to the friction velocity through $-dP/dx = \rho u_*^2/H$. The extent of the domain was approximately $6H$ and $3H$ in the streamwise and spanwise directions respectively.

Two simulations were performed with MetaCFD using the MRT formulation of the LBM. Case 1 used a grid with a node separation of 4 wall units throughout. Case 2 had a grid spacing of 4 wall units in the region near the wall ($z^+ < 60$) and a spacing of 8 wall units in the rest of the domain by using a local grid refinement strategy [15]. These resolutions are not sufficiently fine to resolve all the eddies, and a Smagorinsky model with van Driest wall damping was used to account for the effect of the unresolved motions. The initial velocity was set by taking the mean velocity found from prior simulations and superimposing a divergence free perturbation to initiate turbulence [16]. The computation was run until a statistically steady state was achieved, then continued for a sufficiently long time to collect statistics.

Figure 2 shows the computed mean velocity profile as a function of the distance from the wall and compares it with predictions for the viscous layer and log law regions. The axes are normalised with wall units (ν/u_*) and friction velocity. The coordinates x , y and z represent the streamwise, spanwise and wall-normal directions respectively, and a “+” superscript denotes the use of wall units. Comparisons are made with benchmark data from pseudo-spectral DNS [17], and it can be seen that the computations agree reasonably well, though there is a slight overprediction, but this is generally expected with LES. Root mean square (rms) pressure fluctuations are presented in Fig. 3. The LBM predicts higher intensity fluctuations than the DNS, and this may be due to the compressibility effects, which may be significant in the LBM. Slight overprediction of the pressure fluctuations by LBM has also been found elsewhere in the context of DNS and has been attributed to the different way in which pressure is treated [18].

Turbulent intensities and Reynolds stress are shown in Fig. 4. The results are in good agreement, though, as is commonly seen with LES, there is some overprediction of the streamwise intensities and underprediction of the wall-normal intensities, particularly for Case 2 which used a lower resolution in the

region farther from the wall. The different predictions of Cases 1 and 2 may also be due in part to turbulent motions being somewhat suppressed at the interface between the fine and coarse regions in Case 2; further investigation as to whether this indeed does happen and whether better methods of coupling the coarse and fine grids can be found is warranted.

An example of the application of the LBM to an external flow is that of flow over a cavity. This type of flow is of interest due to the development of self sustained oscillations and the resulting acoustic effects. Tone generation in cavities is generally due to a feedback mechanism involving shear layer instabilities occurring in the mouth of the cavity that is reinforced by acoustic pulses generated by the impingement of the vortical structures on the downstream cavity edge [19]. They can also be due to resonant waves generated by the geometry of the cavity.

Such a case was performed with for a shallow cavity. The simulation used a domain of 550×110 and was run in parallel using eight processors. A turbulent boundary layer was set upstream of the cavity as inlet conditions and the behaviour downstream can be seen in Fig. 5, where the instability in the shear layer is evident. The three dimensional nature of the flow is evident in Fig. 6, which shows intense mixing of vortical structures in the cavity opening. Quantification of the oscillations can be obtained by taking the power spectrum of the velocity at a location near the cavity opening and Fig. 7 shows this as a function of Strouhal number, and this very clearly shows the presence of a dominant frequency, whose magnitude is in agreement with theoretical predictions and experimental data.

PERFORMANCE

Execution speed

One attribute of the LBM is that it runs relatively fast, and some tests were performed to compare the speed of the MRT-LBM code with that of a finite difference code. The code selected for comparison was an established LES solver which used a second order finite difference (FD) scheme [20]. The simulations were set to run the same number of grid points for the same problem, channel flow with a Smagorinsky turbulence model. Both the codes were run on a single processor of AMD Athlon 2000 dual processor computer running the Linux operating system. The GNU compiler with the -O3 optimization was used and checks were made to ensure that the results were reproducible – it was found that the times taken did not vary significantly (within about 1%). However, tests conducted on the Microsoft Windows operating system had shown large variations in time from one identical run to another.

Table 1: Computational times of MRT and FD codes running LES of a turbulent channel flow.

Grid Size	LB code	Finite Difference	Ratio
$270 \times 135 \times 47$	516	2015	3.9
$180 \times 90 \times 32$	153	505	3.3
$135 \times 135 \times 92$	510	1824	3.6
$135 \times 270 \times 47$	516	1849	3.6
$90 \times 45 \times 32$	38	88	2.3

Table 1 shows the times required for 100 time steps for each code. It shows that as far as the computational cost per grid node and per time step is concerned, the MRT code runs faster than the FD code employed. Despite being substantially more complex, the MRT only takes 10–30% more time than the SRT, depending on the platform and compiler, mainly because the nature of the MRT permits many optimizations to be performed. The limiting time step was also examined, for the MRT it was found to be at a CFL number of about 0.42, whereas the FD code became unstable at about 0.35. However the accuracy at larger time steps may not be so good. In any case, the LBM implementation is competitive as compared with the FD code. An important point to note here is the elimination of the need to solve the time consuming Poisson-type equation for pressure field by the LBM, but which is inherent in most of the FD methods. It is also interesting to observe that a recent work based on a LBM for DNS of turbulence flows reported

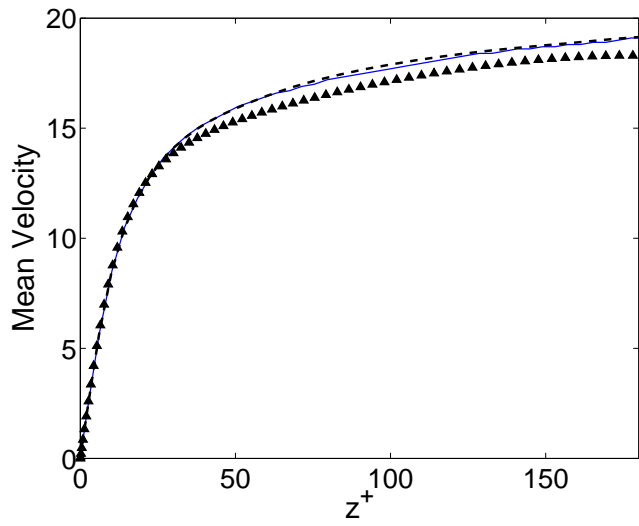


Figure 2: Mean velocity profiles compared with benchmark DNS data. — Case 1; - - - Case 2; symbols DNS [17]. Velocities normalised by u_* .

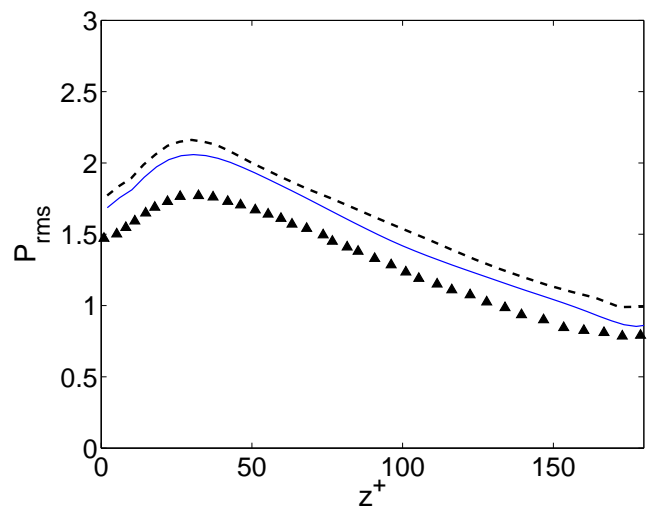


Figure 3: Root mean square pressure fluctuations. [17]. — Case 1; - - - Case 2; symbols DNS [17]. Pressure normalised by ρu_*^2 .

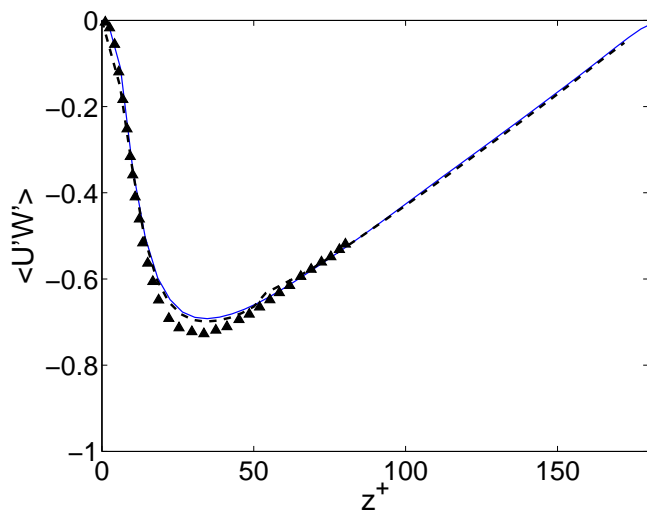
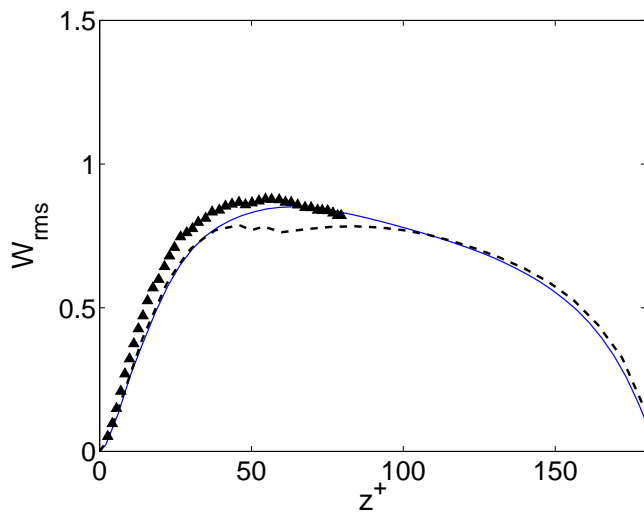
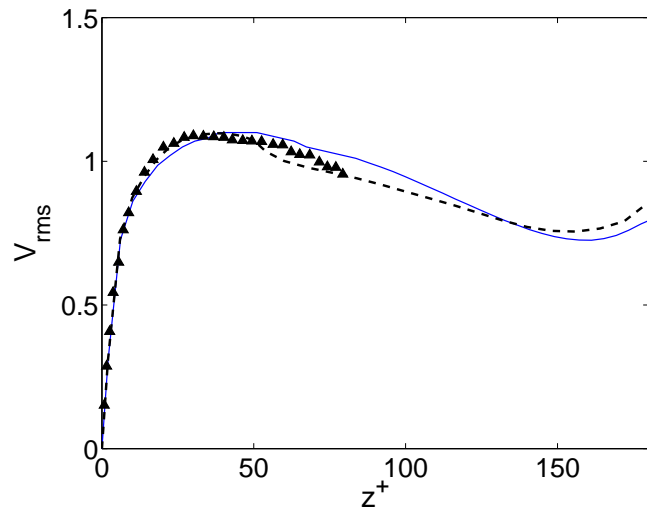
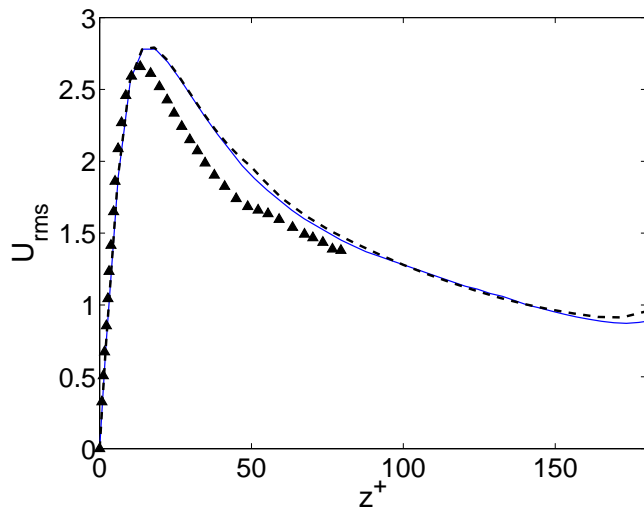


Figure 4: Components of rms velocity fluctuations normalized by friction velocity. — Case 1; - - - Case 2; symbols DNS [17].

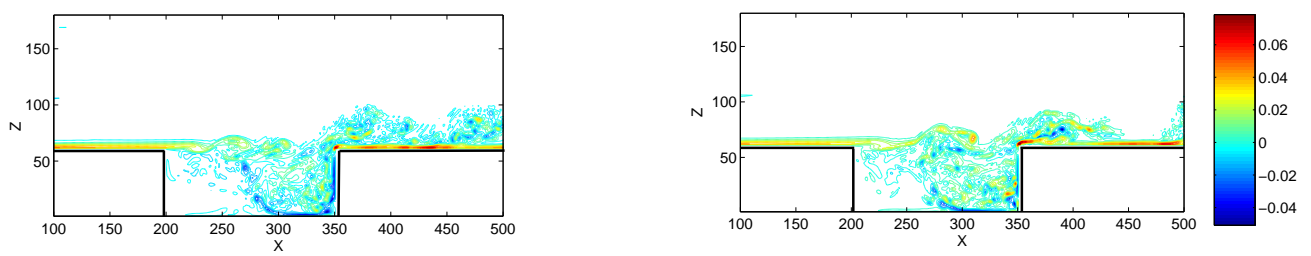


Figure 5: Spanwise vorticity contours in spanwise plane at $t=80\,000$ (left) and $t=85\,000$ (right)

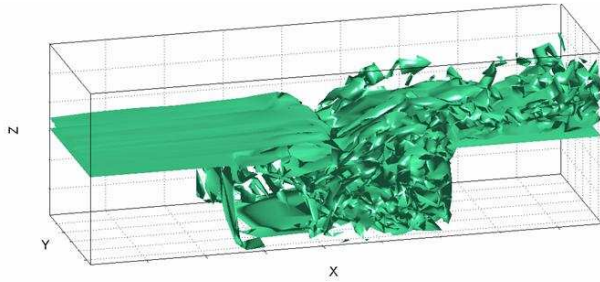


Figure 6: Instantaneous iso-surfaces of spanwise vorticity in shallow cavity.

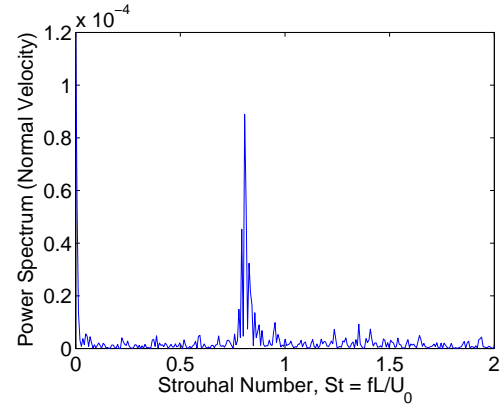


Figure 7: Power spectrum of normal velocity in terms of Strouhal number, based on cavity opening width L and free stream velocity U_0 .

it to be competitive in terms of run-times when compared with the Chebyshev-spectral methods on single processors [18].

Stability

Following the implementation of the MRT model, it soon became clear that it was much more stable than the earlier SRT model. Simulations of some of the more challenging cases suggested that it was more stable by a factor of about three, a value in keeping with previous work. A simple example of a case for which the MRT model demonstrates its superior stability is that of turbulent channel flow with a no-slip boundary condition at the bottom and a rigid free surface at the top. Figure 8 shows the rms velocity fluctuations obtained using both the SRT and the MRT model; spurious spikes in turbulent fluctuations can be clearly seen near the wall, as can evidence of numerical problems near to the free surface. Note that the results from the SRT model were obtained from a relatively small sampling time, which may contribute to the difference between the mean values. The stability is dependent on the spacing of the grid points – finer resolutions are more stable, and the problems can be eliminated through the use of more grid points. The grid size, Δ required to ensure stable behaviour with the SRT model has previously been given as $\Delta^+ \leq 2.5$ for channel flow [18]; however it was shown in the previous section that using $\Delta^+ = 4.0$ yielded good results (though the smallest structures would not have been completely resolved). This suggests the maximum grid spacing that can be used with the SRT-LBM is likely to be dictated by stability constraints rather than considerations of accuracy in many cases.

To obtain a more quantitative assessment of the stability differences between the SRT and MRT models, tests were made with a three dimensional lid-driven cavity flow. Following the example of other researchers [7], the lid velocity was imposed by setting the distribution functions at the lid to the equilibrium distribution function. This method is more stable than alternative methods for setting the velocity, but it should be noted that it is unsuitable for setting velocities where there is a component normal to the wall, for which alternative schemes are available, as discussed in [12]. In these tests, the velocity of the lid was gradually increased until the computation failed (i.e. crashed or showed unphysical behaviour). If the velocity was increased slowly enough, the point at which this occurred was found to be reproducible. Table 2 shows the velocities at which the computation became unstable for different grid sizes and viscosities

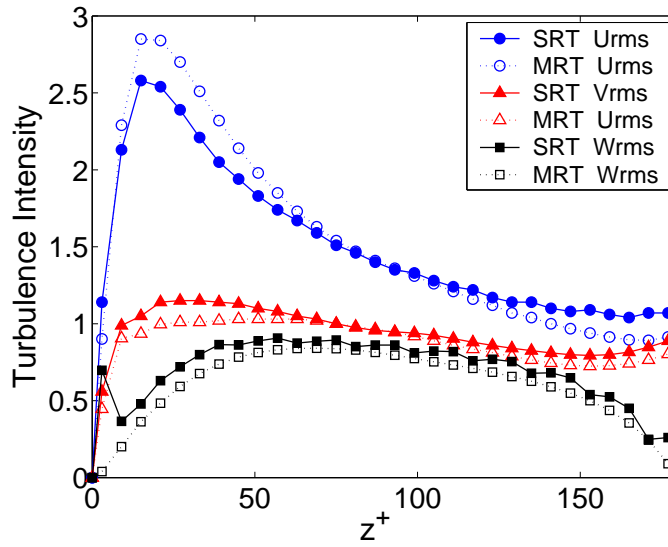


Figure 8: Components of rms velocity fluctuations normalized by friction velocity for fully developed turbulent channel flow with a rigid free surface at the top for $Re_* = 180$ obtained with MRT-LBM (dashed lines with open symbols) and SRT-LBM (solid line with filled symbols).

Table 2: Point at which LBM became unstable for lid driven cavity flow.

Grid	Viscosity	Maximum velocity		Maximum Re		Ratio
		SRT	MRT	SRT	MRT	
64^3	0.0005	0.0295	0.149	3776	19100	5.1
64^3	0.001	0.059	0.232	3776	14848	3.9
64^3	0.002	0.123	0.385	3936	12320	3.1
96^3	0.0005	0.0265	0.188	5088	36096	7.1

(given in lattice units) – it is clear that the MRT approach performed much better than the SRT model on this test. Furthermore, it should be noted that the transition to turbulence takes place at a Reynolds number of about 12 000, so the SRT version was unable to reach the turbulent region on any of the runs, whereas the MRT implementation was, when similar grid resolutions are maintained.

Parallel performance

A key advantage of using the LBM over finite difference/finite volume methods is that they are very well suited to parallel execution on machines with distributed memory architectures. When running computations on parallel machines, the computational domain is split into a number of subdomains, each of which is assigned to different processor. After each time step, each processor exchanges data with a number of different processors. With codes based on the LBM, computations generally only require information from adjacent points, or points two nodes away. This means that a processor only needs to communicate with those processors dealing with adjacent subdomains. If slab decomposition has been used (i.e. the domain has been divided in one direction), then each processor need only transfer data to two other processors, and with 2D block decomposition, data needs to be transferred to at most eight other processors. This contrasts with other schemes where each processor may effectively have to communicate with every other processor, in such cases the proportion of time spent passing data increases with the number of processors, placing a limit on how many processors can usefully be employed.

As an example, Fig. 9a shows the results of some parallel performance tests using the MetaCFD code. The case simulated was turbulent flow through a cylindrical pipe using a Smagorinsky turbulence model modulated by a wall damping function. A grid size of $2400 \times 120 \times 120$ was used and data were obtained by recording the time taken to perform a fixed number (about 200) of time steps. The code was run on between 8 and 128 processors on the US Department of Energy's supercomputer BASSI and the speeds shown in the plot are normalised such that the mean speed on one 8 processor node is eight. For the 128

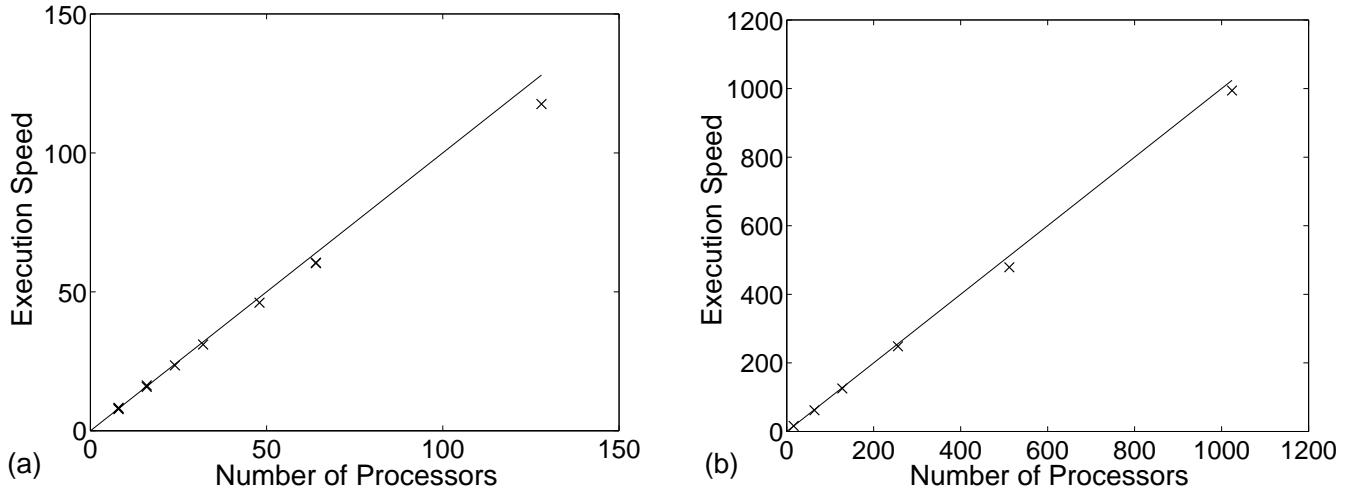


Figure 9: Execution speeds as a function of the number of processors; lines are for linear scaling and crosses data obtained with MetaCFD. (a) LES of pipe flow with fixed domain size of $2400 \times 120 \times 120$. (b) LES of duct flow with fixed subdomain size of $20 \times 256 \times 256$.

processor case, the computational speed was 14.7 times that with 8, corresponding to 92% of that which would be obtained with linear scaling. It was found that for the tests involving just one node, the times taken varied significantly, by up to 6%, though when using many nodes the run times were more or less constant. This suggests that about 3% of the fall off in performance could be accounted for by variations in the speed of the different nodes. Similar performance figures using this code have been found for other flows [12].

Another large-scale investigation involved the turbulent flow through a square duct and was conducted on the US Department of Energy's supercomputer Seaborg, using up to 1024 processors. In these tests, rather than keeping the overall computational domain constant, the size of each subdomain was held constant, at $20 \times 256 \times 256$, or 1.3 million, nodes. The results are shown in Fig. 9b and almost linear scaling is again evident, with the speed for 1024 processors being 97% of that for the 16 processor case.

CONCLUSIONS

The application of the MRT-LBM to the large eddy simulation of turbulent fluid flow in a channel has been tested using the MetaCFD code. Results for the mean flow and turbulence statistics have been compared with prior benchmark data from direct numerical simulations found to be in good agreement. The computational speed of the method has been compared with that of an existing finite difference code and has been found to be faster by a factor of about three, on a per time step per node basis. Comparisons with an earlier version of the LBM, the SRT model, have shown the MRT model to be far superior in terms of numerical stability. However, although the MRT formulation appear to have far greater complexity, the use of appropriate optimization strategies results in an increase in execution time of only 10–30%, depending on the platform used. The principal advantage of the LBM over finite difference types of methods is that its explicit nature enables efficient parallel execution on multiprocessor machines. This has demonstrated here with tests on a massively parallel distributed memory computer in which MetaCFD exhibit almost linear scaling up to 1024 processors. It appears that the advanced formulation of the LBM based on multiple relaxation times offers a promising tool for wall-bounded turbulence simulations on very large computer clusters.

Acknowledgements The development of the MetaCFD has been funded by the US Department of Energy (DOE) under Grant No. DE-FG02-03ER83715, by NASA under Contract Nos. NNL06AA34P and NNL07AA04C, and by NSF under Award No. OII-0610893. The work used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of DOE under Contract DE-AC03-76SF00098, and the National Center for Supercomputing Applications (NCSA) under Award CTS 060027.

REFERENCES

- [1] R.K. Madabhushi, S.P. Vanka, *Large eddy simulation of turbulence-driven secondary flow in a square duct*, Phys. Fluids A, 3, (1991), 2734–2745.
- [2] K.N. Premnath, V. Magi, J. Abraham, *Parallelization of a multidimensional code for the simulation of flows in engines: performance with OpenMP programming mode*, HPC paper 0148, *High Performance Computing (HPC) Symposium, Advanced Simulation Technologies Conference (ASTC)*, San Diego, California (2002).
- [3] S. Chen and G.D. Doolen, *Lattice Boltzmann method for fluid flows*, Annu. Rev. Fluid Mech., 30, (1998), 329–364.
- [4] Y.H. Qian, D. d’Humières and P. Lallemand, *Lattice BGK models for the Navier-Stokes equation*, Europhys. Lett., 17, (1992), 479–484.
- [5] P. Bhatnagar, M. Gross and E. Krook, *A model for collision processes in gases. I. Small amplitude processes in charged and neutral one-component systems*, Phys. Rev., 94, (1954), 511–525.
- [6] H. Chen, S. Chen, W.H. Matthaeus, *Recovery of the Navier–Stokes equations using a lattice-gas Boltzmann method*, Phys. Rev. A, 45, (1992), 5339–5342.
- [7] D. d’Humières, I. Ginzburg, M. Krafczyk, P. Lallemand, L.-S. Luo, *Multiple-relaxation-time lattice Boltzmann models in three-dimensions*, Phil. Trans. R. Soc. Lond. A, 360, (2002), 437–451.
- [8] J. Carter et al., *Magnetohydrodynamic turbulence simulations on the Earth Simulator using the lattice Boltzmann method*, Supercomputing 2005.
- [9] K.N. Premnath, J. Abraham, *Three dimensional multi-relaxation-time (MRT) lattice Boltzmann models for multiphase flow*, J. Comput. Phys., 224, (2007), 539–559.
- [10] K.N. Premnath, M.J. Pattison, S. Banerjee, *Generalized lattice Boltzmann equation with forcing term for LES of bounded turbulent flows: accuracy, stability and computational efficiency*, To be submitted to Phys. Rev. E.
- [11] X. He, L.-S. Luo, M. Dembo, *Some progress in the lattice Boltzmann method. Part I: Non-uniform mesh grids*, J. Comp. Phys., 129, (1996), 357–363.
- [12] M.J. Pattison, K.N. Premnath, N.B. Morley, M.A. Abdou, *Progress in lattice Boltzmann methods for magnetohydrodynamic flows relevant to fusion applications*, Fusion Eng. Des. (accepted).
- [13] K.N. Premnath, J. Abraham, *Discrete lattice BGK Boltzmann equation computations of transient incompressible turbulent jets*, Int. J. Modern Phys. C, 15, (2004), 699–719.
- [14] D. Yu, R. Mei, W. Shyy, *A multi-block lattice Boltzmann method for viscous fluid flows*, Int. J. Numer. Meth. Fluids, 39, (2002), 99–120.
- [15] H. Chen, O. Filippova, J. Hoch, K. Molvig, R. Shock, C. Teixeira, R. Zhang, *Grid Refinement in Lattice Boltzmann methods based on volumetric formulation*, Physica A, 362, (2006), 158–167.
- [16] K. Lam, *Numerical investigation of turbulent flow*, PhD Thesis, Univ. California, Santa Barbara, (1989).
- [17] J. Kim, P. Moin, R. Moser, *Turbulence statistics in fully developed channel flow at low Reynolds numbers*, J. Fluid Mech., 177 (1987), 133–166.
- [18] P. Lammers, K.N. Beronov, R. Volkert, G. Brenner, F. Durst, *Lattice BGK direct numerical simulation of fully developed turbulence in incompressible plane channel flow*, Computers and Fluids, 35, (2006), 1137–1153.
- [19] D. Rockwell, E. Naudascher *Review – self sustaining oscillations of flow past cavities*, Trans. ASME, 100, (1978), 152–165.
- [20] M.V. Salvetti, S. Banerjee, *A priori tests of a new dynamic subgrid-scale model for finite-difference large-eddy simulations*, Phys. Fluids, 7, (1995) 2831–2847.