

NUMERICAL SIMULATION OF FLUIDS USING THE LATTICE BOLTZMANN SCHEME

M.J. Pattison¹ and S. Banerjee²

1: MetaHeuristics LLC, Santa Barbara, USA martin@metah.com

2: MetaHeuristics LLC, Santa Barbara, USA sanjoy@engineering.ucsb.edu

KEY WORDS

Lattice Boltzmann, parallel processing, turbulence, LES

ABSTRACT

The lattice Boltzmann scheme for solving the equations governing the flow of fluids has become increasingly popular in recent years. In this method, a probability distribution function for the velocity of fluid elements is computed at each grid point, which contrasts with more established computational fluid dynamics methods, which solve for an average velocity. Advantages of the lattice Boltzmann scheme are that it can deal with complex geometries relatively easily and the numerical solution scheme readily lends itself to processing on parallel architectures. In this work, the lattice Boltzmann scheme was implemented in a computer code and a number of models for different flow phenomena were incorporated. A model for multicomponent flow was included and an eddy viscosity turbulence model was incorporated in order that large eddy simulations could be performed. Evolution of a passive scalar field was simulated by using an explicit Lax-type finite difference scheme. Geometries used in the simulations described include flow around a cylinder and over a backward-facing step. The speed of computation has been compared with that of a more traditional CFD code, and the performance using parallel processing on a computer cluster has been investigated, with the lattice Boltzmann code clearly demonstrating its strength in this regard.

1. INTRODUCTION

Most codes used for the simulation of fluid flows solve the equations governing the fluid behaviour by using the finite difference approach. An alternative methodology, which has gained popularity in recent years, is to use the lattice Boltzmann (LB) method. In this approach the fluid at every point on a grid is considered to comprise fluid elements with a distribution of different velocities. The general idea behind this scheme is that fluid motion can be represented by a probability distribution function $f_i(\mathbf{x}, \mathbf{v}, t)$, where f is the probability of finding a fluid element with velocity vector \mathbf{v} at location \mathbf{x} at a particular time t . In practical implementations of this approach, the distribution function is discretised in time and space with a finite set of velocities being used. The use of several fixed velocities with probability distribution functions (sometimes called digital fluid dynamics) contrasts to more conventional CFD models in which one variable velocity is used. A comprehensive review of the lattice Boltzmann method is given by Chen and Doolen (1998), and a brief description is also presented in the following section.

One advantage of the lattice Boltzmann method is that it is very well suited to parallel processing on distributed memory architectures. At each time step, it is only necessary for a processor to communicate with a fixed number of other processors, typically between two and eight depending on

how the computational domain is divided up, and for large problems, the execution speed is roughly proportional to the number of processors used. This contrasts with other methods where each processor has to communicate with all, or many other, processors involved in the computation; in this situation, when large numbers of processors are used, the time spent communicating can severely reduce the execution speed compared to the lattice Boltzmann method where the execution speed is almost proportional to the number of processors used.

Another benefit is that solution of the Poisson equation is not required in LB methods. Since this takes up 80-90% of the CPU time in traditional solution methods, LB codes are also much faster per time step. In addition, LB methods can handle flows in complex geometries relatively easily, and deal with multiphase flows and complex chemistry.

2. THE LATTICE BOLTZMANN METHOD

In this approach to solving the equations of fluid dynamics, the fluid at every point on a grid is considered to comprise fluid elements with a distribution of different velocities. The general idea behind this approach is that the fluid flow can be represented by a probability distribution function $f_i(\mathbf{x}, \mathbf{v}, t)$, where f is the probability of finding a fluid element with velocity \mathbf{v} at location \mathbf{x} at a particular time t . In practical implementations of this approach, the distribution function is discretised in time and space with a finite set of velocities being used.

In three dimensional implementations of this approach, both 15 velocity and 19 velocity models are commonly used. The 15 velocity model is illustrated in Figure 1. In this model, the velocity vectors correspond to the directions from the centre of the cube to the corner, the cube centre to the middle of the faces, with one zero velocity vector.

The equations used in the lattice Boltzmann method are normally non-dimensionalised such that these components have magnitude 0, 1 or $\sqrt{3}$. For example, the velocity vectors for the directions corresponding to $i = 0, 1$ and 8 are then:

$$e_0 = (0, 0, 0) \quad e_1 = (1, 0, 0) \quad e_8 = (1, 1, 1) \quad (1)$$

with similar relations for the other directions.

The following relations for the density, ρ , and the velocity, \mathbf{u} , can be written as:

$$\rho = \sum_i f_i \quad (2a)$$

$$\rho \mathbf{u} = \sum_i \mathbf{e}_i f_i \quad (2b)$$

where the summation is performed over all the different directions.

To calculate the evolution of the flow field with time, two steps are used. The first is known as the collision step and represents the exchange of momentum due to interactions between fluid elements with different velocities and other sources; this can be expressed as:

$$f_i^c(\mathbf{x}, t+1) = f_i(\mathbf{x}, t) + \Omega_i(\mathbf{x}, t) \quad (3)$$

where Ω is known as the collision operator. The other step deals with the advection and is given by:

$$f_i(\mathbf{x} + \mathbf{e}_i, t+1) = f_i^c(\mathbf{x}, t+1) \quad (4)$$

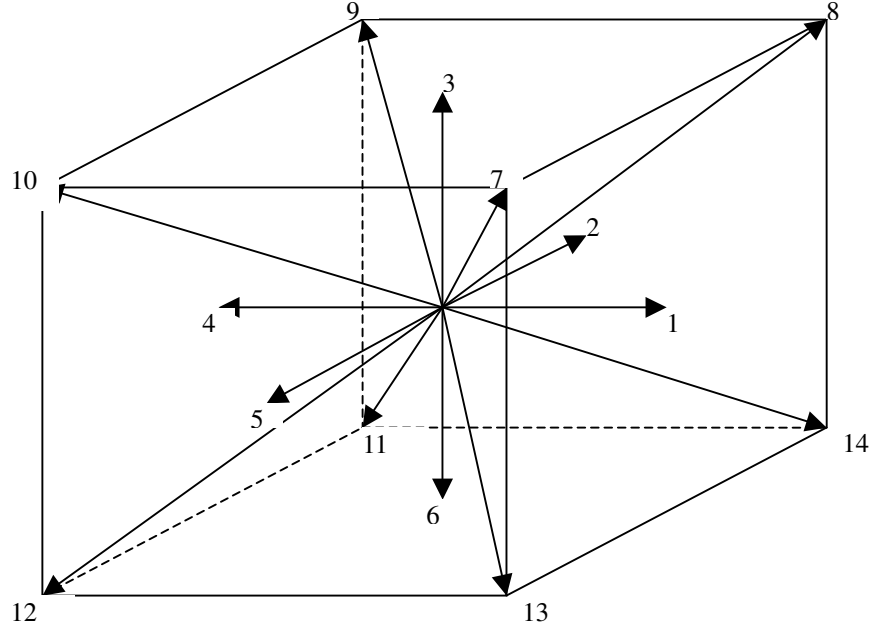


Figure 1. Three-dimensional, fifteen velocity lattice

In these equations, a dimensionless time is used, the normalisation parameter is chosen such that the fluid elements are advected to adjacent grid points in one timestep (with the exception of the rest element).

Defining a relaxation time, τ , the collision operator can be expressed as:

$$\Omega_i = -(f_i(\mathbf{x}, t) - f_i^{eq}(\mathbf{x}, t)) / \tau \quad (5)$$

where f_i^{eq} is an equilibrium distribution function. This then leads to the following form for the finite difference lattice Boltzmann equation:

$$f_i(\mathbf{x} + \mathbf{e}_i, t+1) = f_i(\mathbf{x}, t) + \frac{f_i(\mathbf{x}, t) - f_i^{eq}(\mathbf{x}, t)}{\tau} \quad (6)$$

In order to solve this equation, we need an appropriate formulation for the equilibrium distribution function. A commonly used formulation for this is (Qian et al., 1992; Sankaranarayanan et al, 2002):

$$f_i^{eq} = w_i \rho \left[1 + 3\mathbf{e}_i \cdot (\mathbf{u} + \tau \mathbf{a}) + \frac{9}{2} (\mathbf{e}_i \cdot (\mathbf{u} + \tau \mathbf{a}))^2 - \frac{3}{2} (\mathbf{u} + \tau \mathbf{a}) \cdot (\mathbf{u} + \tau \mathbf{a}) \right] \quad (7)$$

where \mathbf{a} is an acceleration term accounting for forces acting on the fluid, such as gravity. The relaxation time can be related to the dynamic viscosity, ν , by

$$\tau = 3\nu + 0.5 \quad (8)$$

The w_i coefficients are weighting factors which determine the distribution of the particles among the different velocities. For example, in the 15 velocity model the following values are used:

$$\begin{aligned} w_i &= 2/9 & i &= 0 \\ w_i &= 1/9 & i &= 1, \dots, 6 \\ w_i &= 1/72 & i &= 7, \dots, 14 \end{aligned}$$

3. PARALLEL PROCESSING

In order to test the performance of the lattice Boltzmann method on a multiprocessor system, the LB code was set up to run on a networked computer cluster. The cluster used had AMD processors running at about 2GHz, and the networking permitted data transfer speeds of 1Gbit/sec – as such it can be regarded as being typical of a modern computer cluster. Ten machines in the cluster were made available to us for our tests.

The example used was one of plane Couette flow, and two sizes of grid were used. One was $480 \times 40 \times 28$ points (~ 0.5 million in total) and the other $120 \times 40 \times 28$. The parallelisation involved splitting the computational domain into a number of slabs, then assigning each slab to a different computer. Thus when the first domain was run on ten machines, each processor was used to perform the calculation on a sub-domain of size $48 \times 40 \times 28$ grid points. In order to proceed with the simulation, the computations at each mesh point require data from adjacent nodes. For points on slab boundaries, these data lie on other machines, so following each iteration, data packets are exchanged between the computers processing adjacent slabs.

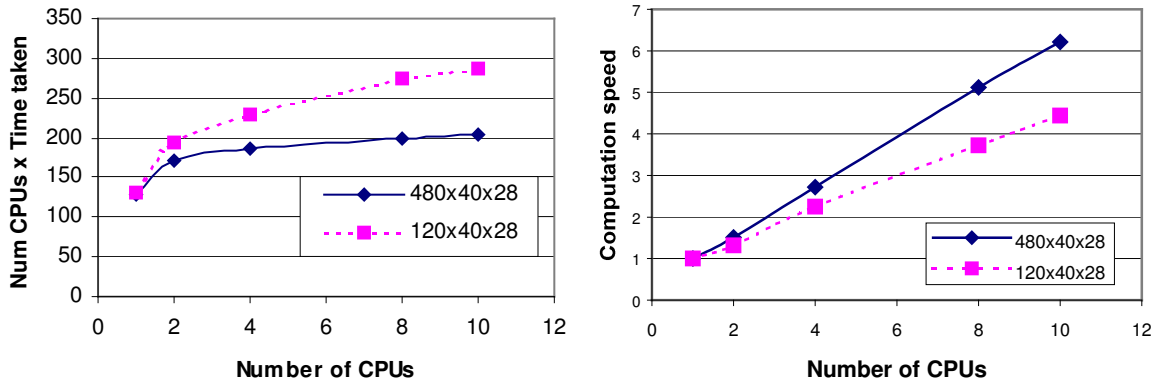


Figure 2. (a) Total computational time taken and (b) overall speed (Mflops normalised to 1) for running a LBM code on a parallel cluster. Slab decomposition used, with cuts along the x-axis (the longest one)

Figure 2 shows the total amount of computational time taken (i.e. the time taken times the number of processors) and the overall speed (reciprocal of elapsed time) against the number of processors used; the results are normalised so that the times of both cases are the same for one processor. If parallelisation were perfect, and data transfer between computers instantaneous, the total computing time would remain constant and the lines in Figure 2a would be flat. As it is, they rise somewhat as the number of processors is increased. There is a significant rise in the amount of

time taken when two machines are used instead of one, which is due to the computational overhead associated with the threading used to control the communication. As the number of CPUs used increases, there is a modest increase in the amount of time taken. This is because while the number of floating point operations performed by each sub-domain is inversely proportional to the number of processors used, the amount of data needing to be exchanged remains constant. This is reflected by the fact that the larger domain shows a much lower drop in performance, and one can see that the time taken using 2 CPUs and the smaller grid is the same as that taken for 8 CPUs and the larger grid; in both cases the sub-domains were the same size. What determines the proportion of the time spent exchanging data is the ratio of the number of grid points on the boundaries to the total number of points in the slab.

In order to assess the actual speed of the LB computations, a simple flow through a rectangular duct was simulated using both the LB code and a finite difference code using the projection method to solve the pressure Poisson equation. Figure 3 plots the ratio of the computational speed (timesteps per unit time) of the LB code with that of the projection method (PM) code. Clearly, the LB code is much faster, especially for larger grids.

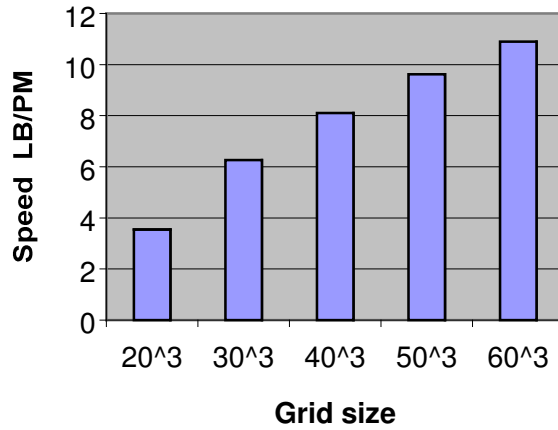


Figure 3. Ratio of LB code speed to that of conventional CFD code

These results of these tests confirm that LB codes run very efficiently, and indicate that scalability on multiprocessor systems is good, particularly with larger domains.

4. MULTICOMPONENT FLOWS

To calculate the velocity and density fields requires only one distribution function for the case of single-component flows. To extend the model to multi-component systems, a natural approach is to use separate functions for each component, and it was this method that was used in this work. At this point it is useful to briefly present the basic equations used to calculate these distributions. Equation 6 can be modified to give:

$$f_i^\sigma(\mathbf{x} + \mathbf{e}_i, t + 1) = f_i^\sigma(\mathbf{x}, t) + \frac{f_i^\sigma(\mathbf{x}, t) - f_i^{eq,\sigma}(\mathbf{x}, t)}{\tau} + S_i^\sigma(\mathbf{x}, t) \quad (9)$$

where f^σ represents the number of fluid particles of component σ , τ is a relaxation time and S_i^σ is source term. \mathbf{e}_i is the direction vector connecting the grid point at \mathbf{x} with its neighbouring points, in our code we used models with 15 directions or 19 directions. The equilibrium distribution f_i^{eq} is calculated using:

$$f_i^{eq,\sigma} = w_i \rho_\sigma \left[1 - d_i + 3\mathbf{e}_i \cdot (\mathbf{u} + \tau \mathbf{a}^\sigma) + \frac{9}{2} (\mathbf{e}_i \cdot (\mathbf{u} + \tau \mathbf{a}^\sigma))^2 - \frac{3}{2} (\mathbf{u} + \tau \mathbf{a}^\sigma) \cdot (\mathbf{u} + \tau \mathbf{a}^\sigma) \right] \quad (10)$$

w_i is a weighting factor and determines how the fluid particle velocities are distributed among the different directions and ρ the number density. The parameter d_i is used to take into account the difference in density of the fluids. \mathbf{a} is an acceleration term resulting from forces on the fluid element. Note that the velocity \mathbf{u} is the bulk liquid velocity and will be the same for each component.

Sources may be due to chemical reactions creating a particular component or result from capture of ions. Dealing with these sources is a relatively straightforward matter of incorporating them into the final term in Equation (9). However, in the LBM it is necessary to assign a velocity distribution to these sources and this may be done using Equation (10) to apportion the source according to the equilibrium distribution.

The main consideration for multicomponent flows is the interactions between the components. For example, to simulate immiscible fluids the effects of the molecular repulsion cause the fluids to separate. There are a number of ways of dealing with this and the method tested in Phase I was based on the work of Shan and Chen (1993, 1994) and was to introduce a force on the fluid elements calculated as:

$$\mathbf{F}_i^\sigma(\mathbf{x}) = -\varphi^\sigma(\mathbf{x}) \sum_{\sigma'} \sum_i G_i^{\sigma\sigma'} \varphi^{\sigma'}(\mathbf{x} + \mathbf{e}_i) \quad (11)$$

where the summation is taken over each component and around neighbouring cells. φ is a function of the component density; in the studies undertaken here, it was simply taken to be proportional to the density of each component. The magnitude of the interaction force is controlled by $G_i^{\sigma\sigma'}$ and is related to the surface tension. A similar equation can be written for fluid-wall interactions. The use of this equation effectively models the effects of surface tension and can be used to keep the components separate. It should be noted that care must be used when selecting values for G_i – some authors have suggested using the same constant value for each direction, but in these studies it was found that this could result in interfaces preferentially aligning along grid directions or at an angle of 45° to them. In the case of 2-D simulations this tended to result in octagonal droplets. In our 3-D 15 velocity model, setting the values of G_i for the diagonal directions to be one eighth of those parallel to axes ensured isotropic behaviour.



Figure 4. Flow of droplet down wall. Profile across interface of circled area shown in next figure.

An example obtained using this model is illustrated in Figure 4. In this case, an initially semi-circular droplet attached to a wall with the surrounding fluid stationary. A gravitational force was applied and the code was able to simulate the movement of the droplet. Initially the droplet spread out along the wall due to capillary effects, then as the fluid accelerated it moved along the wall. It should be noted that the angle of contact between the wall and the droplet is determined by surface tension. Figure 5 shows the concentrations across the interface, i.e. a blow-up of the interfacial region circled in Figure 4 – values are normalised to one, and it can be seen that the interface is about three grid points across. As with all the computations we performed, a 3-D LB model was used – since this was a 2-D problem, only one grid point was used in the third direction, with periodic boundary conditions imposed.

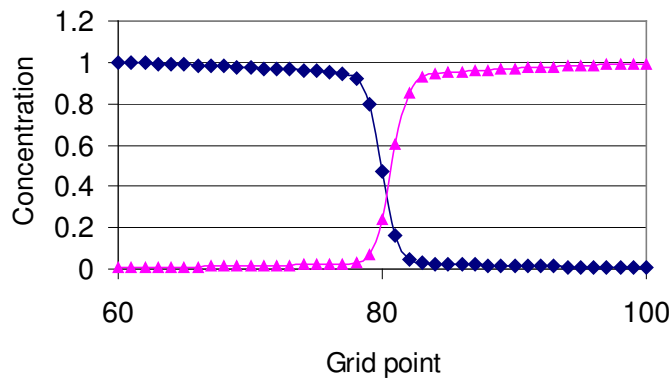


Figure 5. Variation of concentrations of fluids perpendicular to interface

3. THERMAL AND SCALAR TRANSPORT

The transport of heat or any other passive scalars in a prescribed velocity field, \mathbf{u} , can be modelled by using a transport equation:

$$\frac{\partial \phi}{\partial t} + \nabla \cdot (\mathbf{u}\phi) = \nabla \cdot (D\nabla \phi) + \Psi \quad (12)$$

where ϕ is the scalar, D is a diffusion coefficient and Ψ is a source term. The variable diffusion coefficient introduces an effective advection term which must be handled appropriately to avoid instabilities (Succi et al., 1999).

Solution of this equation can be achieved by introducing a pseudo-velocity defined by:

$$\mathbf{U} = \mathbf{u} - \nabla D \quad (13)$$

If the flow is turbulent, the thermal diffusivity is given by:

$$D = \frac{\kappa(\nu + \nu_t)}{\rho C_p \nu} \quad (14)$$

where κ is thermal conductivity, ν is the viscosity and ρ the density. The turbulent viscosity ν_t would be calculated by the turbulence model in the LB code.

A Lax type scheme was used for the finite differencing to give:

$$\begin{aligned} \phi(\mathbf{x}, t + \delta t) = & \frac{1}{2} \sum_i \left[\left(\frac{1}{3} + U_i^- \right) \phi(\mathbf{x} - \delta \mathbf{x}_i, t) + \left(\frac{1}{3} - U_i^+ \right) \phi(\mathbf{x} + \delta \mathbf{x}_i, t) \right] + \Psi(\mathbf{x}, t) \delta t \\ & + \sum_i [\phi(\mathbf{x} + \delta \mathbf{x}_i, t) - 2\phi(\mathbf{x}, t) + \phi(\mathbf{x} - \delta \mathbf{x}_i, t)] \hat{D}_i \\ & - \sum_i [D(\mathbf{x} + \delta \mathbf{x}_i, t) - 2D(\mathbf{x}, t) + D(\mathbf{x} - \delta \mathbf{x}_i, t)] \phi(\mathbf{x}, t) \delta t / (\delta \mathbf{x}_i)^2 \\ & - \frac{1}{8} \sum_{i,j \neq i} \left\{ U_i^- U_j (\mathbf{x} - \delta \mathbf{x}_i, t) [\phi(\mathbf{x} - \delta \mathbf{x}_i + \delta \mathbf{x}_j, t) - \phi(\mathbf{x} - \delta \mathbf{x}_i - \delta \mathbf{x}_j, t)] \right\} \\ & \left\{ U_i^+ U_j (\mathbf{x} + \delta \mathbf{x}_i, t) [\phi(\mathbf{x} + \delta \mathbf{x}_i + \delta \mathbf{x}_j, t) - \phi(\mathbf{x} + \delta \mathbf{x}_i - \delta \mathbf{x}_j, t)] \right\} \end{aligned} \quad (15)$$

where

$$\hat{D}_i = D \delta t / (\delta \mathbf{x}_i)^2 - (1 - 3U_i^2) / 6$$

and

$$U_i^\pm = U_i(\mathbf{x} \pm \delta \mathbf{x}_i, t)$$

There are a couple of points to be made regarding this approach. First, the thermal equation is decoupled from the equations governing the velocity. Second, the finite difference equation is explicit – this has the advantage over an implicit formulation that it is easily parallelizable, retaining

the advantage related to parallelisability of the LB method. The finite difference formulation has been linked to the LB velocity field solver. This does not impact computing speed which is usually determined in CFD codes by the velocity field solver in this case. In conventional/continuum CFD codes it is the solution of the pressure Poisson equation which takes most of the CPU time per timestep.

A heat transfer scenario which was used to test this procedure was the Graetz problem. In this case flow through a pipe with walls held at a constant temperature is simulated. The inlet temperature is held constant, and in this simulation, there was a fully developed parabolic velocity profile. Figure 6 shows temperature profiles across the diameter of the pipe at three different locations from the inlet. These are in agreement with the analytical solution and indicates that the LB code incorporating the passive scalar formulation discussed is suitable for the prototype in Phase II. The method can be used for both heat transfer and as impurity transport.

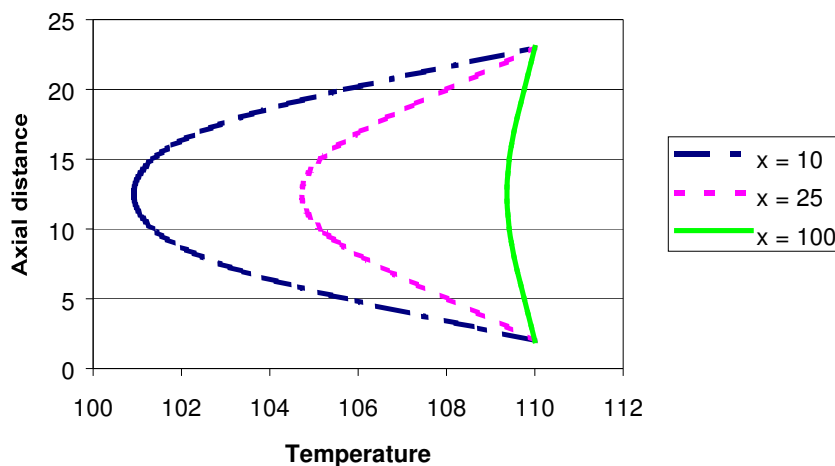


Figure 6. Temperature profiles downstream of entrance. Inlet temperature of 100, walls at 110.

5. COMPLEX BOUNDARIES

When calculating the fluid velocity at a grid point, values of the distribution functions at neighbouring points are used. In cases where the grid point is adjacent to a boundary, this presents problems when data from the wall side are required. For example, in Figure 7, values at point d are required by points 'a', 'b' and 'c', but since there is no fluid at 'd', values are not available. The solution to this is to use an extrapolation scheme in distribution functions on the fluid side and the distance of the points from the wall are used to set artificial values at 'd'. This is very straightforward for straight boundaries that are parallel to the grid axes – the simple “bounceback” scheme is used for this (Succi, 2001); for more complex shapes, the scheme of Mei et al. (2002) was used. This approach is considerably easier than using a body-fitted grid, as would be done by finite difference codes, as we shall show, works very well, confirming the capability of LB codes to handle complex geometries without body fitting.

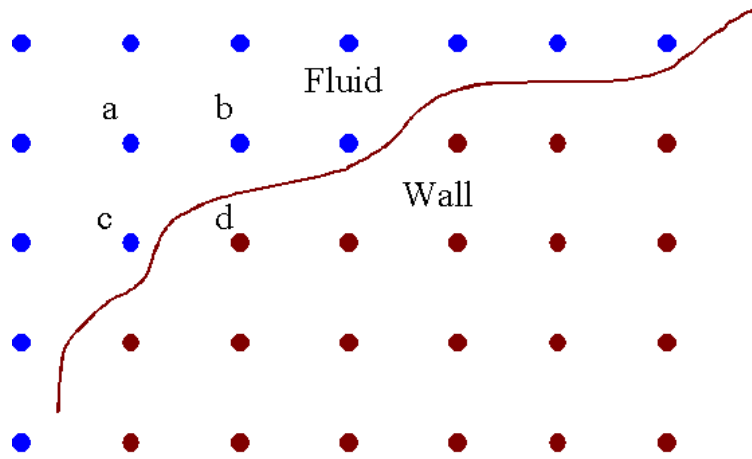


Figure 7. Arrangement of grid points around boundary

A number of tests with different geometries were performed as a validation of the code, the object being both to assess both the of the performance of the boundary conditions used and to verify that the basic equations used by the LBM had been properly implemented. The first test involved flow over a backward facing step. The flow at the inlet had a prescribed parabolic profile and there was a free outlet far downstream of the step as shown in Figure 8 below:

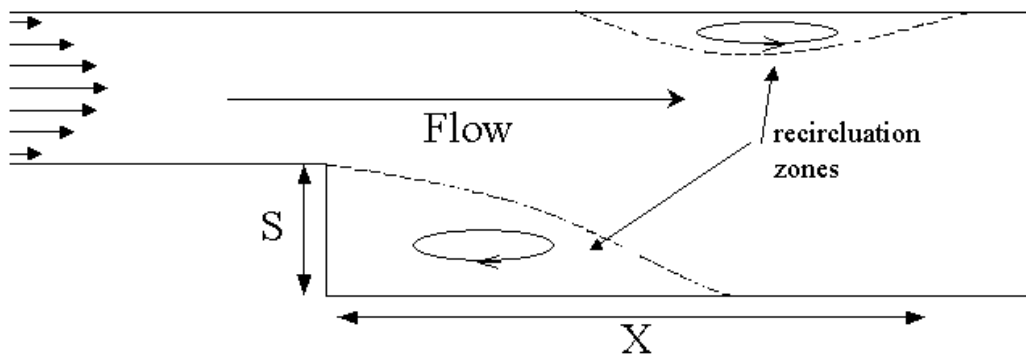


Figure 8. Flow over a backward facing step

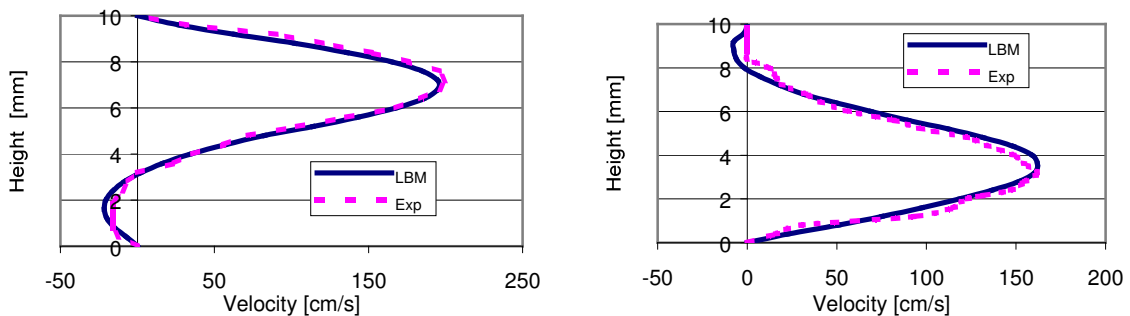


Figure 9. Velocity profiles downstream of step. Left at $x/S = 6$, right at $x/S = 20$.

Figure 9 shows the velocity profiles calculated at two locations downstream from the step and compares them with the experimental results of Armaly et al. (1983) for a Reynolds number of 1000 (based on the mean inlet velocity and twice the inlet height). Results are in close agreement, though

differences can be seen. Simulations at lower Reynolds numbers gave much closer agreement with experimental data, and it was suggested by Armaly that three dimensional motion at the higher Reynolds number influenced the experimental results.

At higher Reynolds numbers, flow around a cylinder becomes unstable and produces vortices that are carried downstream by the flow. Figure 10 shows a snapshot of instantaneous vorticity contours for a flow past a cylinder with a Reynolds number of 81.6. In this case the flow is bounded at the top and bottom by no-slip walls and a parabolic inlet profile was used, and we again used Mei et al.'s method. Because of the flat walls and the cylinder, the geometry would normally be quite complex to body fit, but is straightforward for the LB code. The plot shows clearly the presence of the vortices, a measurement of the frequency with which these vortices were produced gave a value of 0.25 for the Strouhal number – in very close agreement to those found by other workers (Gergova, 2002).

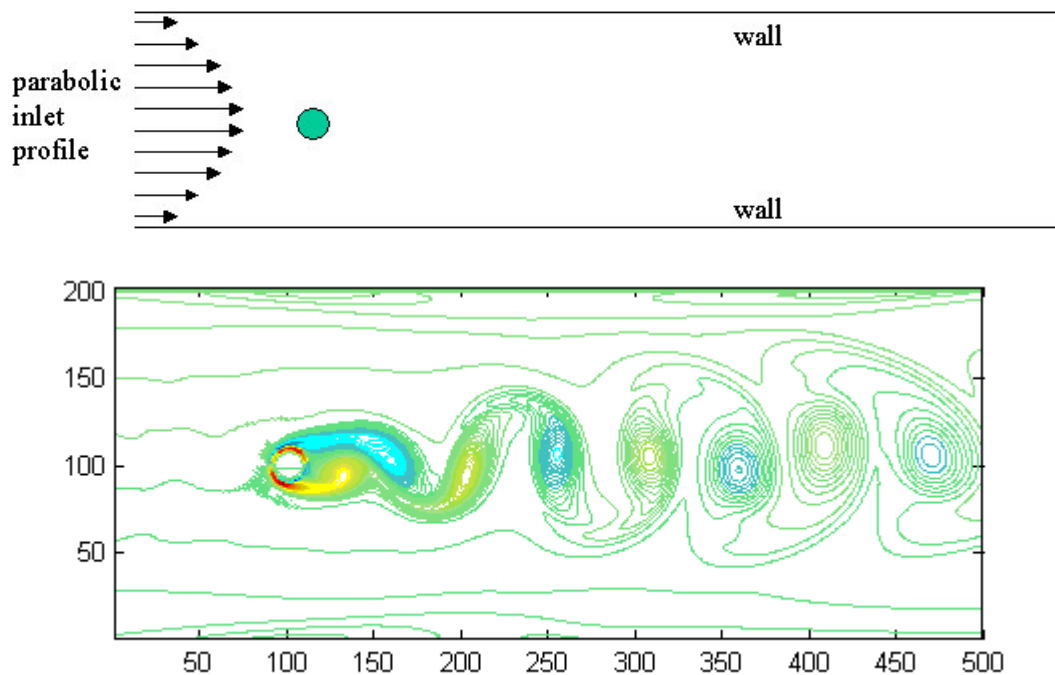


Figure 10. Vortex shedding from a cylinder. Flow is to right. Cylinder centred at (100, 100)

It was found that as well as giving good results, Mei et al.'s method was relatively easy to use. Unlike conventional CFD methods, there is no need to generate a grid, whether body fitted or finite volume that conforms to the shape of the boundaries – usually a major task. Instead a rectangular grid is used, with the model requiring the distance of the boundary from adjacent grid points. Numerical stability was very good when the 19 velocity model was used – for some cases numerical oscillations were found when using the less isotropic 15-velocity model, an effect previously reported in the literature. This confirms the expectation that the LB code could be applied in a straightforward manner to complex geometries.

5 FREE SURFACES

One question which arises for a variety of flows in fusion systems, e.g. the liquid surface module in the divertor experiment, is the behaviour of the free surface. In the case of flat surfaces, this is relatively straightforward and a simple condition similar to the bounceback scheme can be applied.

However, the situation is more complicated for cases where the surface deforms, and one part of Phase I was to address how best to deal with this.

The approach adopted involves the use of an interaction force:

$$\mathbf{F}_i(\mathbf{x}) = -\varphi(\mathbf{x}) \sum_i G_i \varphi(\mathbf{x} + \mathbf{e}_i) \quad (16)$$

This is similar to Equation (11) – indeed it is the special case of just one component. By using appropriate relations for G_i and the function φ , one can simulate a fluid which exhibits a phase transition with a dense and a light state (Nourgaliev et al., 2002, 2003). This equivalent to the situation in real fluid where intermolecular attraction causes them to have a liquid and a gaseous state. This approach was applied using the following form for φ :

$$\varphi = \rho_0 (1 - \exp(\rho / \rho_0)) \quad (17)$$

Simply using a relation where φ was proportional to the density caused numerical instabilities for values of G_i large enough to cause phase separation, but the above relation avoided these problems. Using this approach, it is able to simulate separated flow without there being any need to explicitly keep track of the interface – the boundary evolves automatically.

An example is shown in Figure 11. In this case, the simulation is started with a dense fluid layer lying over a lighter fluid. Symmetry boundary conditions are applied to the sides, with solid walls on the top and bottom. A small perturbation is applied to the interface, and a column of dense fluid flows into the light fluid. The bulge which develops near the end of the column is due to Kelvin-Helmholtz instability, and development of the shape of the interface is similar to that found by He et al. (1999).



Figure 11. Penetration of a dense fluid (top) into a light fluid.

6 TURBULENT FLOW

Many flows of industrial interest are turbulent in nature and it is these that frequently pose the greatest challenges for computational modelling. One of the simplest ways of simulating turbulent flow is to use an algebraic eddy viscosity model. In this type of model, the effect of turbulence on the dynamics of the flow is captured by substituting the molecular viscosity with an *effective* or *eddy* viscosity. The Baldwin-Lomax and the Cebeci-Smith models are of this type and are discussed by Granville (1987). In such models, the eddy viscosity is typically a function of the local vorticity, the wall shear stress, and the distance from the boundary. A simple model of this type was implemented in our code without any problems being encountered.

The more accurate way of modelling turbulence is to run a simulation at a high enough resolution that all the turbulent eddies are captured, a technique known as direct numerical simulation (DNS). No additional turbulence models are required and these simulations can be done with an LBM. The principal drawback of DNS is that the very high resolution required results in large computational demands.

An alternative to DNS is large eddy simulation (LES) in which the grid resolution is set to be large enough to resolve the larger scale eddies, and the effects of the smaller scale eddies are accounted for by introducing an eddy viscosity. A number of models for calculating the eddy viscosity are available and in this work the model of Smagorinsky (1963) was used, in which an eddy viscosity is calculated from the strain rate tensor, S_{ij}

$$v_t = (C_s \Delta)^2 (2S_{ij}S_{ij})^{1/2} \quad (18)$$

where C_s is a constant and Δ is the grid spacing.

This model was tested by simulating an open channel flow with a no-slip condition at the bottom, a flat surface (free slip) at the top and periodic boundary conditions in the other directions. Turbulence statistics for r.m.s. velocity fluctuations were compiled and found to be in keeping with what is normally found in shear flow. Figure 12 compares the r.m.s. velocity fluctuations in each direction with the channel flows results of Pan (1996). The sets of data are in good qualitative agreement, showing similar trends. From a quantitative perspective, there are significant differences; this was attributed to the fact that a much lower grid resolution was used in the LB calculations.

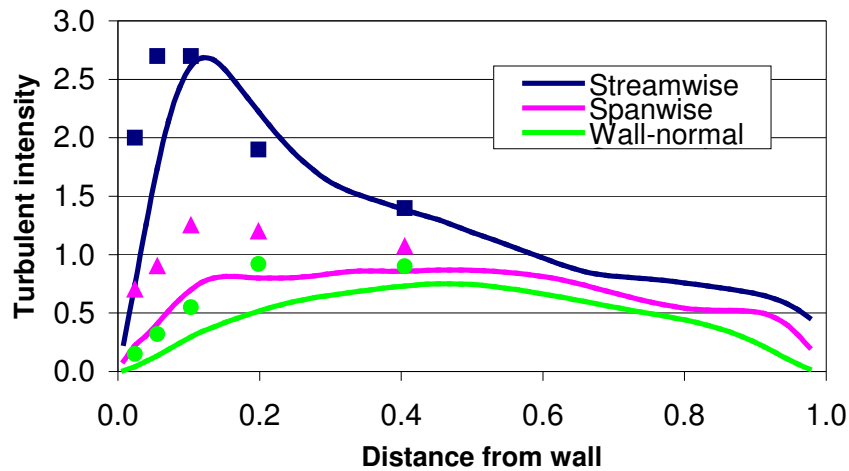


Figure 12. Turbulence intensities – values are normalised by the friction velocity

7. CONCLUSIONS

During this work, we were able to implement basic working models for many of the key aspects of fluid behaviour into a lattice Boltzmann model. This resulted in a fully 3-D transient LB code capable of handling complex boundaries, multicomponent flows, heat and passive scalar transport, free surfaces and turbulence (using either a turbulence model or LES).

- It was verified that the LB code is highly parallelizable and scales almost linearly with the number of processors, with scaling improving with domain size.
- It was shown that the use of potential functions to represent the effects of intermolecular forces enabled modelling multicomponent, immiscible fluids and the effects of surface tension – applications include flows with capillary effects and droplet/bubble behaviour. In addition, this technique can be used to simulate condensable gases. The interface evolves automatically using this method, removing the need to explicitly track the boundary – as such it can be (and has been) applied to free surface flows.

- A suitable finite difference scheme for heat transport was formulated. This scheme is also applicable to the flow of passive scalars such as trace component transport. It is an explicit scheme and only a knowledge of conditions at neighbouring grid points is required and is one-way coupled to the parallelised LB velocity solver.
- A scheme for dealing with irregular and complex boundaries was implemented. This method enabled flow past curved objects (cylinders and spheres were tested) to be accurately simulated without developing a grid to conform to the boundaries and the objects and the confining walls. It was clear that the LBM has great potential for modelling complex geometries and this is one very strong point in its favour.
- The code has successfully been able to model a turbulent flow, resolving the eddies and turbulence structures, indicating sufficiently low numerical diffusion to sustain turbulence – often a problem with low order finite difference methods.

ACKNOWLEDGEMENT

This work described in this paper was funded by the US Department of Energy, under grant DE-FG02-03ER83715

REFERENCES

1. B.F Armaly, F. Durst, J.C.F Pereira and B. Schönung (1983) Experimental and theoretical investigation of backward-facing step flow. *J. Fluid Mech.*, **127**, pp. 473-496
2. S. Chen and G.D. Doolen (1998) Lattice Boltzmann method for fluid flows. *Annu. Rev. Fluid Mech.*, **30**, 329-364
3. M. Gergova (2002) *Evaluation of improved boundary conditions for the lattice Boltzmann approach: Investigation of the laminar vortex sheet behind a circular cylinder*. Bachelor's Thesis, Friedrich-Alexander University, Erlangen-Nuremberg
4. P.S. Granville (1987) Baldwin-Lomax factors for turbulent boundary layers in pressure gradients. *AIAA Journal*, **25**, pp. 1624-162
5. X. He, S. Chen and R. Zhang (1999) A lattice Boltzmann scheme for incompressible multiphase flow and its application in Rayleigh-Taylor instability. *J. Comp. Phys.*, **152**, pp. 642-663
6. A. R. Mei, D. Yu and W. Shyy (2002) Force evaluation in the lattice Boltzmann method involving curved geometry. *Phys. Rev. E*, **65**, paper 041203
7. R.R Nourgaliev, T.N. Dinh and B.R. Sehgal (2002) On lattice Boltzmann modeling of phase transition in an isothermal non-ideal fluid. *Nuclear Eng. and Design*, **211**, pp. 153-171
8. R.R. Nourgaliev, T.G. Theofanous and D. Joseph (2003) The lattice Boltzmann equation method: theoretical interpretation, numerics and implications. *Int. J. Multiphase Flow*, **29**, pp. 117-169
9. Y. Pan (1996) *Numerical Investigation of Particle-Containing Turbulent Open Channel Flows*. PhD Thesis, Dept. of Chemical Engineering, Univ. California, Santa Barbara
10. Y.H Qian, D. d'Humières, P. Lallemand (1992) Lattice BGK models for the Navier-Stokes equation. *Europhys. Lett.*, **17**, pp. 479-484.
11. K. Sankaranarayanan, X. Shan, I.G. Kevrekidis and S. Sundaresan (2002) Analysis of drag and virtual mass forces in bubbly solutions using an implicit formulation of the lattice Boltzmann method. *J. Fluid Mech.*, **452**, 61-96
12. X. Shan and H. Chen (1993) Lattice Boltzmann model for simulating flows with multiple phases and components. *Phys. Rev. E*, **47**, 1815-1819

13. X. Shan and H. Chen (1994) Simulations of nonideal gases and liquid-gas phase transitions by the lattice Boltzmann equation. *Phys. Rev. E*, **49**, 2941-2948
14. J. Smagorinsky (1963) General circulation experiments with the primitive equations: 1. The basic equations. *Mon. Weather Rev.*, **91**, pp. 99
15. S. Succi, H. Chen, C. Teixeira, G. Bella, A. De Maio and K. Molvig (1999) An integer lattice realization of a lax scheme for transport processes in multiple component fluid flows. *J. Comp. Phys.*, **152**, pp. 493-516
16. S. Succi (2001) *The Lattice Boltzmann Equation for Fluid Dynamics and Beyond*. Clarendon Press, Oxford, UK