42nd European Rotorcraft Forum
Lille, France, 5–8 September, 2016
Paper 85

TOWARDS REAL TIME WAKE COMPUTATIONS USING LATTICE BOLTZMANN METHOD FOR FLIGHT DYNAMICS SIMULATIONS

Mark A. Woodgate
Mark.Woodgate@glasgow.ac.uk
CFD Laboratory, School of Engineering
James Watt South Building
University of Glasgow, G12 8QQ, U.K.

Rene Steijl
Rene.Steijl@glasgow.ac.uk
CFD Laboratory, School of Engineering
James Watt South Building
University of Glasgow, G12 8QQ, U.K.

George N. Barakos
George.Barakos@glasgow.ac.uk
CFD Laboratory, School of Engineering
James Watt South Building
University of Glasgow, G12 8QQ, U.K.

Gavin J. Pringle
g.pringle@epcc.ed.ac.uk
EPCC, University of Edinburgh
James Clerk Maxwell Building
Edinburgh, EH9 3FD, U.K.

Abstract

1 INTRODUCTION

Computational Fluid Dynamic (CFD) methods have become increasingly sophisticated and accurate over the past 20 years, however they are orders or magnitude too slow for real time flow computation and so, analytical models or simplified aerodynamic models are still used if real time estimates are necessary.

There are a number of methods to represent vortical wakes in real time flight simulations. The first is to use an analytical model or a set of velocity vectors in tabular form. A second method for real time simulation is obtained by reducing the computational cost of the calculation by using a low fidelity free wake model such as shown by Horn *et al.* [1] who performed a parametric study of the wake parameters to achieve real time execution with minimal differences from a spatially and temporally converged response, which at the time did not achieve real time execution. Lastly, a method suggested by Spence *et al.* [2] used an implicit large eddy simulation (ILES) to build a database which is accessed in real time. This was achieved through the use of a data compression schemes via mesh simplification, and the use of kd-trees for fast data queries.

In recent years the numbers of cores in both Central Processing Units (CPUs) and Graphic Processing Units (GPUs) have been increasing rapidly and currently stand at a few thousand cores for a high end commodity GPU. This number of cores makes running real time simulations much more feasible but the schemes will have to take advantage of such a large number of processors by

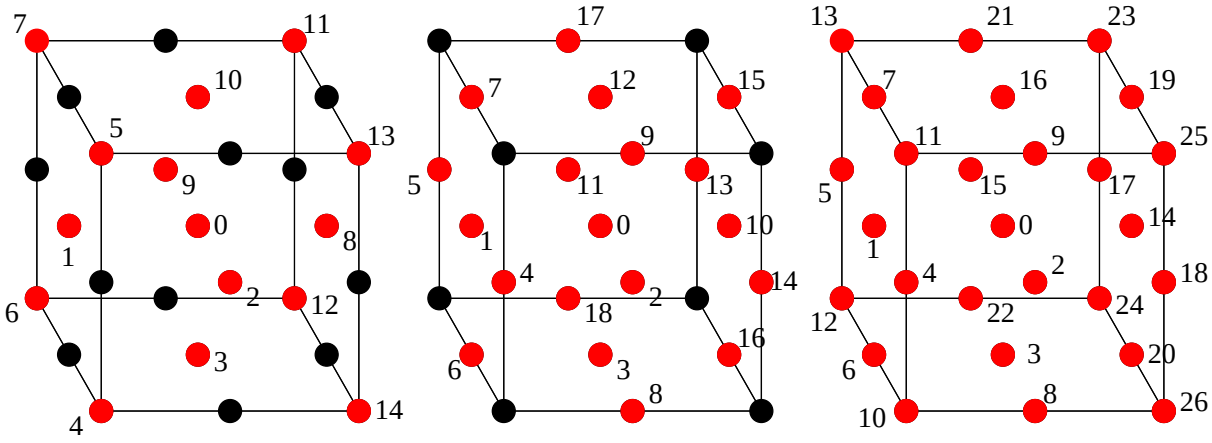


Figure 1: The common three dimensional lattices with the indexes re-order so as to minimise the traversal of memory and hence reduce the memory bandwidth requirements of the the LBM

carefully choosing algorithms that decompose into a large number of semi independent operations. Recently, lattice Boltzmann methods (LBM) have emerged as an alternative to the more traditional methods for simulating fluid flow. The method was developed as an extension to lattice gas automata [3, 4] and reviews of the developments since then can be found in [5, 6].

Recently Khan, *et al.* [7] demonstrated the use of the lattice Boltzmann method, implemented on a graphic processor unit (GPU), running real time simulations for indoor environments.

2 THE LATTICE BOLTZMANN METHOD

The Lattice Boltzmann Method (LBM) solves macroscopic fluid dynamics problems and sits at the boundary between the molecular and continuum views. The LBM uses the discrete Boltzmann equation to simulate flow using the Bhatnagar-Gross-Krook (BGK) collision model to calculate the flow of the fluid across a limited number of particles and directions. The stating point is the Boltzmann Equation

$$\frac{\partial f}{\partial t} + \frac{p}{m} \cdot \nabla f + F \cdot \frac{\partial f}{\partial p} = L(f, f) \tag{1}$$

where m is the mass of the particles, F is the force field acting on the particles and the distribution function $f = f(x, p, t)$ is a function of position x, momentum p, and time t. The right hand side describes the collisions between particles and encapsulates the behaviour of the particles. One simplification is (BGK) where

$$L(f, f) = \tau(f_0 - f). \tag{2}$$

where f_0 is the local equilibrium value of the particles and τ is a relaxation time related to the viscosity of the fluid. The finite discrete velocity model of the Boltzmann equation is where the evolution of the fluid particles is confined to a discrete set of Q velocities $E = e_0, e_1, ..., e_Q$, such that the conservation of equation 1 is given by

$$\partial_t f_i(x, p, t) + e_i \cdot \nabla f_i(x, p, t) = L_i(f, f) \tag{3}$$

where the discrete one-particle probability distribution function $f_i(x, p, t)$ is the probability that a fluid particle has velocity e_i, at time t, at position x. The transport of particles is then balanced by the interactions represented by the collision operator $L_i(f, f)$.

In the lattice Boltzmann method the space is discretised into a regular Cartesian lattice where each node represents a small portion of the fluid. The lattices are commonly labelled DdQq, where d is the spatial dimension and q are the number of microscopic velocities. Some common three dimensional lattice constructions for fluid flows are D3Q15, D3Q19 and D3Q27 as shown in figure 1. The D3Q19 model has been chosen to keep the computational cost low while maintaining an isotropic lattice.

The equations are solved numerically by a two step process. First, there is a collision step where;

(4)
$$f_i^t(x, t+\delta t) = f_i(x,t) + \frac{1}{\tau_f}\left[f_i^{eq}(\rho,u) - f_i(x,t)\right]$$
$$= (1 - \frac{1}{\tau_f})f_i(x,t) + \frac{1}{\tau_f}f_i^{eq}(\rho,u),$$

with f_i representing the particle distribution function which is the fraction of particles located at position x, at time t, moving with the microscopic velocity e_i, and i are the discrete directions of momentum which are the q chosen collocation points of the velocity-discrete Boltzmann equation and determine the basic structure of the numerical grid.

This is followed by a streaming step where the value of $f_i^t(x, t+\delta t)$ is shifted in space along the lattice velocity e_i

(5) $$f_i(x + ce_i\delta t, t + \delta t) = f_i^t(x, t + \delta t).$$

where c is the lattice speed. The relaxation time τ determines how fast the equilibrium position is approached and is also related to the kinematic viscosity of the fluid. The equilibrium state $f_i^{eq}(\rho, u)$ itself is a low Mach number approximation of the Maxwell-Boltzmann equilibrium distribution function where ρ is the the macroscopic value of the density and u is the value of the velocity.

The density ρ and the velocity u are obtained from the zero and first moments of the distribution functions

(6) $$\rho = \sum_{i=0}^{18} f_i, \qquad \rho u = \sum_{i=0}^{18} ce_i f_i$$

and the discrete velocity set e_i is defined as follows

(7) $$e_i = \begin{cases} (0,0,0) & i = 0 & w_i = 1/3 \\ (\pm1,0,0) & i = 1-2 & w_i = 1/18 \\ (0,\pm1,0) & i = 3-4 & w_i = 1/18 \\ (0,0,\pm1) & i = 5-6 & w_i = 1/18 \\ (\pm1,\pm1,0) & i = 7-10 & w_i = 1/36 \\ (\pm1,0,\pm1) & i = 11-14 & w_i = 1/36 \\ (0,\pm1,\pm1) & i = 15-18 & w_i = 1/36 \end{cases}$$

The equilibrium state is calculated by

(8)
$$f_i^{eq}(\rho,u) = \rho w_i\left(1 + \frac{3e_i\cdot u}{c} + \frac{9(e_i\cdot u)^2}{2c^2} - \frac{3u^2}{2c^2}\right)$$

where the w_i are the weight co-efficients defined in equation 7.

It can be shown through a Chapman-Enskog expansion [8, 9] that the Navier-Stokes equations can be obtained from the lattice BGK model.

It should be noted that equation 4 is local in the sense that only information at point x is needed to calculate the system state. The streaming process is limited to the nearest neighbours and requires no computation. This makes the Lattice Boltzmann method ideal for running on many cores simultatiously.

Although the lattice Boltzmann method is expressed as a decomposition into two steps this is a suboptimal in terms of software implementation. More optimised implementations can be found for both CPUs [10, 11] and graphics processing units (GPUs) [12]. The disadvantages of the LBM is that it is limited to low Mach numbers with non-trivial implementations of the boundary conditions. However both these drawbacks are not an issue when just considering wake calculations.

2.1 High Reynolds Number Flows

The relaxation time τ is related to the viscosity of the fluid by

(9) $$\tau = 0.5 + 3\nu_{lb} = 0.5 + u_{lb}(N-1)/Re$$

where ν_{lb} and u_{lb} are the viscosity and speed in lattice units with Reynolds number Re. However as τ approaches $1/2$ the scheme becomes unstable as the lattice viscosity is too low to dissipate the shortest wavelengths. The Reynolds number can be increased by several orders of magnitude by use of the Entropic Lattice Boltzmann method [13, 14] which allows the Lattice Boltzmann models to support a discrete H-theorem through the use of a modified equilibrium distribution function

(10)
$$f_i^{eq} = \rho w_i \sum_{\alpha=1}^{3}\left(2 - \sqrt{1+u_\alpha^2}\right)\left(\frac{2u_\alpha + \sqrt{1+3+u_\alpha^2}}{1-u_\alpha}\right)^{e_{i\alpha}}$$

The relaxation process is also modified with an adjustable parameter β at every simulation step by means of the solution of the h-function monotonicity constraint

(11) $$H(f) = H(f - \beta(f - f^{eq}))$$

which produces an unconditional stable numerical scheme.

2.2 Turbulence Model

For turbulent calculations a Smagorinsky sub-grid scale model [15] is used locally to modify the fluid viscosity by adding a term ν_t which is dependent on the magnitude of the strain rate tensor S.

$$\nu = \nu_0 + \nu_t \tag{12}$$

In the smagorinsky model, the relaxation time τ_t is calculated using the momentum flux tensor:

$$Q_{\alpha\beta} = \sum_i e_{i\alpha} e_{i\beta} (f_i - f_i^{eq}), \tag{13}$$

and

$$\tau_t = \frac{1}{2}\left(\sqrt{\tau_0^2 + 4c_s^{-4} C_S^2 (Q_{\alpha\beta} Q_{\alpha\beta})^{1/2}} - \tau_0\right) \tag{14}$$

where C_S is the smagorinsky constant. This increases the computational of the scheme, as well as removing the single relaxation time, since it is now both spatially and temporally varying dependent on the gradients of the velocity, but it is still local to the node.

More recently Malaspinas [16] introduced two new methods, the *consistent strain* model and the *consistent smagorinsky* model. Although the second method gave superior results it is not as computationally efficient as the first since it involves both second order Hermite polynomial as well as the strain rate is calculated with finite differences.

3 FINITE VOLUME METHOD LBM COUPLING

Coupling between a Navier-Stokes solver and a Lattice Boltzmann method were considered in [17]. The connection between the lattice Boltzmann variables and the finite volume method was achieved through a first order expansion of the lattice Boltzmann variables around the local equilibrium term. The LBM is a molecular level method and so contains more information then the FVM macroscopic variables. The zeroth order terms relate to the the macroscopic quantities (FVM variables) while the first order terms are related to the gradients. Since in this case the FVM was an incompressible solver the pressure field of the FVM was related to the density field of the LBM with the constant calculated assuming the density average on the interface was constant. Recently Tong and He [18] proposed a framework to unify the different coupling schemes.

3.1 Parallel implementation

Implementing the interface between FVM and LBM solvers not only requires that the variables are transformed from macroscopic to molecular and back again but the communication does not have to be two way. In general the two unsteady schemes will not be progressing at the same real time step and real require different numbers of iterations per synchronisation. If this number is fixed then it still possible to use a two way communication, however to increase the flexibility between the coupling methods a one sided communication using remote memory access (RMA) model has been used in this case.

MPI [19] has two memory models for RMA, as shown in figure 2, unified where the public and private windows are identical and separate where they are different. In the unified model the whole of the solution vector would need to be in a window instead of just the interface values.

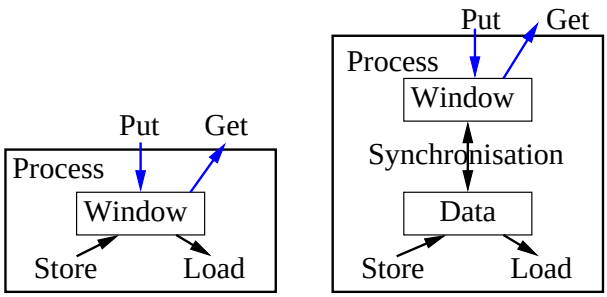


Figure 2: The two memory models for RMA, unified on the left and separate on the right.

This model was chosen for its flexibility since there is both a conversion between the molecular variables and the macroscopic variables, and an interpolation required before the data is in the correct format for the other code. Both of these steps can be done within the synchronisation process hence allowing either the molecular or macroscopic variables to be *put* or *get* depending which has the smaller message size.

To fully exploit this, the windows are set up as shown in figure 3. In this configuration, multiple windows are contained within a single processor and each window will have its own synchronisation process so it's quite possible that not all windows on the same process have the variables stored in the same format. Another advantage of laying out the windows in this way, is that at no time do multiple processes try and access them same region of

the window and this mean that the windows do not have to be locked.

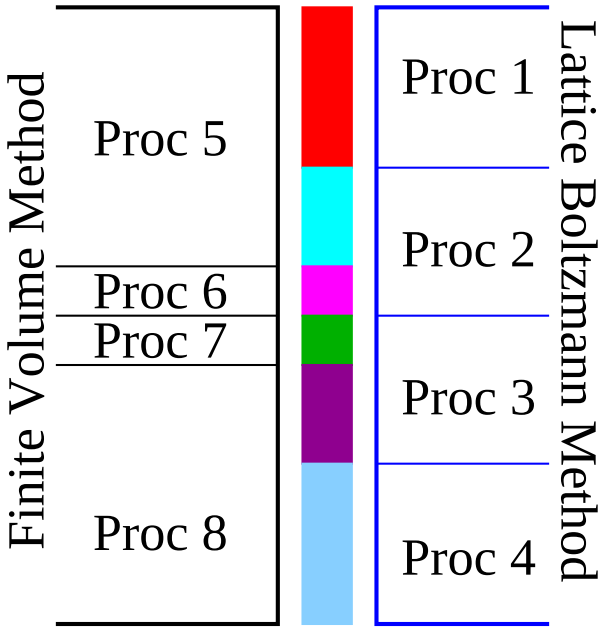


Figure 3: The layout of the windows on the interface between the HMB and LBM.

4 REAL TIME COMPUTATION

A simulation is at least real time, if the physical time between two simulation steps is no less than the time taken to compute them. For this to occur, means there is a balance between the size of the lattice which effects both the size of the domain as well as the accuracy of the results, and the computational cost of updating each lattice point. Consider the following idealised wake problem. Assuming a computational domain of $20m \times 20m \times 20m$ with a flow of $10m/s$ the Reynolds number would be around half a million. Assuming a $20cm$ resolution within the computational domain, implies there are 1 Millions points in the lattice. Taking a lattice speed of 0.05, 1000 lattice time steps are required for each real second of time. Hence a LBM needs to be able to update 1 Billion lattice points per second to obtain a real time computation. Currently state of the art solvers, Palabos [20] can reach about 120 Million update per second on an ARCHER [21], the UK national supercomputing service, node, two 2.7GHz 12-core E5-2697 v2 processors, while current GPU performance is around 900 Million update per second [7] meaning that a rate of 1 Billion is within reach. However this should be noted that this is for the most basic of lattice Boltzmann methods and the addition of both the turbulence model and the entropic lattice Boltzmann method will add to the cost of the algorithm and hence reduce the number of lattice updates per second.

Secondly, a dynamical load balancing is used to maintain balance between the FVM and the LBM and run them at maximum efficiently on a given number of processors. The load balance can be seen in figure 4. With N processes, these are split into 3 groups. The first is a single process which is the controller. The second is the group of processes associated with the FVM code and the third is the group of processes associated with the LBM code. Within the last two groups statistics are keep on how close to real time the computation is. These are fed back to the controller via a single processor to decides if a re-balancing is needed. Both the FVM and the LBM have been modified to run as subroutines, avoiding the need to use MPI spawning. The main changes required were to rename `MPI_COMM_WORLD` in both codes and clear up all the memory leaks. To maintain a single code base, a compiler directive is used in the top level function to distinguish if the code base will be used alone or within with controller framework.

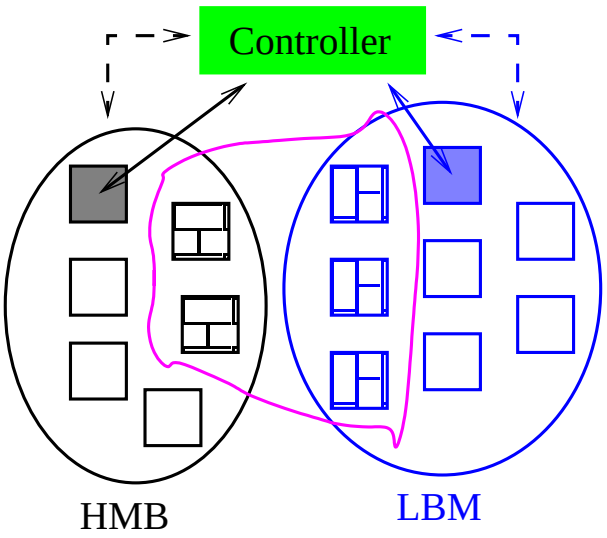


Figure 4: The different processor types in the real time simulation.

The main task of the controller is to compute the new interface groups and interpolation factors needed in the synchronisation process. This is done while the fluid computation continues so as to reduce the downtime during the restart. At this point the controller tells process zero of the FVM and LBM groups that a restart is required. The solution data is then either dumped to disk in a processor independent format or send directly to the controller to avoid the I/O.

Function	Percentage run time
Collide	32%
Halo exchange plus boundary conditions	21%
Calculating continuum velocities	12%
Calculating equilibrium distribution	8%
Steaming	8%
Updating the continuum conditions	7%
Calculating residual norm	4%

Table 1: The top 7 most computational expense functions in LBM code accounting for over 90% of the runtime of the code.

5 PERFORMANCE

A key aspect of obtaining real time computations is optimising both the serial and parallel performances of the algorithm.

Table 1 shows the time spent in LBM code for a test case of a lattice of size $11 \times 101 \times 101$ split into 16 blocks with periodic boundary conditions on all sides. The streaming and the collision parts of the algorithm currently take up some 40% of the runtime of the code. The one part of the code which needs more optimisation is the exchange of data between the blocks, this requires no computation an only works on the 6 block faces, 12 block edges and 8 block corners. The remained are the cost in calculating the continuum variables from equation 6 and the equilibrium distribution from equation 8

No. of Procs	Time	Efficiency
1	2.850s	100.0%
2	1.540s	92.5%
4	0.921s	77.4%
8	0.601s	59.2%
12	0.607s	39.1%

Table 2: Wall time and efficiency of LBM on a 2.7 GHz, 12-core E5-2697 v2 Archer CPU

As can be seen from table 2 the LBM has a very different parallel scaling compared to a FVM which would still be around 75% all 12 cores of a E5-2697 CPU. The main difference between the codes is the computational cost per lattice point is much lower than in an approximate Riemann of the FVM. Due to this low amount of work per load/store the memory bandwidth within the processor becomes a critical bottleneck in the performance when running on all the cores. In fact the increase in performance between 8 and 12 cores is minimal. In fact the streaming part of the algorithm does no floating point operations and is just loads and stores.

A new improved memory layout, shown in table 3, allocates a single block of memory, then aliases the lattice points instead of allocating each lattice point separately. This increases the single node performance at the expense of some parallel efficiency however the over runtime of the code has been reduced even at the maximum number of cores.

No. of Procs	Time	Efficiency
1	2.200s	100.0%
2	1.180s	93.2%
4	0.714s	77.0%
8	0.489s	56.2%

Table 3: Wall time and efficiency on a 2.7 GHz, 12-core E5-2697 v2 Archer CPU using improved memory layout

Figure 5 shows the parallel performance of the current LBM compared to that of Palabos [20]. A state of the art LBM code which has been developed since 2009 when it branched from the OpenLB [22]. The LBM of Glasgow is not as efficient on single node performance as Palabos, for example Palabos uses only q variables per node instead of $2q$, as well as, combining both the collision and streaming into one process which means the memory just has to be stepped through once instead of twice. These performance enhancements will be introduced at a latter stage when the couple frameworks have been shown to be robust. However the speedup curves between both codes show a very similar drop of in performance at the higher number of nodes after the memory bandwidth of the processor has been saturated.

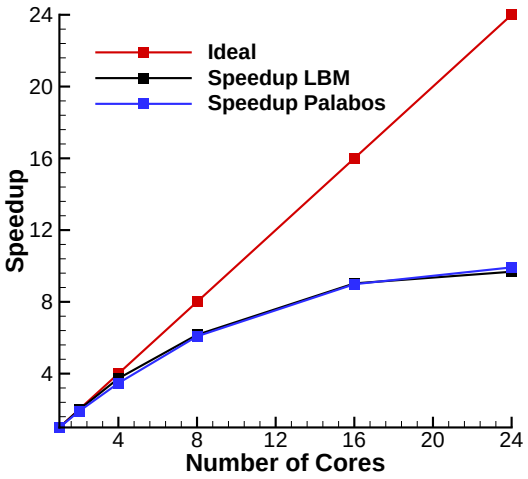


Figure 5: The speedup curve for running up to 24 cores within an ARCHER compute node for the LBM of Glasgow compared to that of Palabos [20] - (Two 2.7GHz 12-core E5-2697 v2 Processors).

6 RESULTS

The following results example how well the LBM can be used to resolve and maintain vortical structures at the resolutions required for for real time computations

6.1 Two counter-rotating vortices

For a counter-rotating vortex pair in a viscous fluid their mutual induced velocity causes them to remain a fixed distance apart, and to move together in a fixed direction. Figure 6 shows the initial solution on the baseline 101×201 lattice and the fine 201×401 lattice. Due to the maximum vorticity begin at a lattice node in both lattices the peak magnitude of vorticity is the same while the behaviour near the peak is less well represented in the the baseline lattice. Figure 7 shows the comparison between the vorticity on the baseline and fine lattice computations after cycling through the domain 4 times. The solutions are very similar with the baseline lattice is only slightly lagging behind the finer grid solution with also a slightly reduced peak vorticity. This means that the resolution of 101×201 baseline lattice is a lattice converged solution. Figures 8 and 9 shown the time history of the counter-rotating vortex pair for the baseline and fine lattice respectively. They clearly show that the behaviour of the system is similar on both lattice sizes and so the baseline lattice is fine enough for a lattice converged result. The figures also clearly show the vortex pair to diffuse over time due to the low Reynolds number used in the calculation. This diffusion also causes the vortex pair to slow down over time.

6.2 Two co-rotating vortices

This test case consists of a pair of co-rotating equal vortices. Before merging, the vortex pair can be described as follows. Firstly, the two vortices rotate around the centroid of the total vortex. Next the vorticity contours are deformed elliptically by the local strain induced by each vortex on the other. Thirdly, vorticity waves propagate from each vortex rearranging the field and lastly the core of each vortex increases due to viscous effects. The viscosity reduces the ratio between vortex radius and vortex separation until a critical value is reached and the vortices merge. Figure 10 clearly shows the correct merging behaviour of a co-rotating vortex pair on a 401×401 lattice.

6.3 Vortex shedding behind a cylinder

The final test case is a real time calculation of flow based a cylinder. The lattice size was $601 \times 101 \times 3$ with a Reynolds number of 1000, lattice velocity of 0.1. Hence the lattice spacing is 0.01 and time step of 0.001 equates to 1000 iteration per real second. The I/O was excluded from the calculation as it represented a substantial, 35%, total run time. Figure 11 shows the behaviour of the wake throguh a single shedding cycle.

7 CONCLUSIONS

This paper presented initial results of implementing a three dimensional LBM within the University of Glasgow solver suite. It has shown good promise at being able to maintain flow features on a mesh resolution which could be run in real time without a massively parallel computing. Both HMB and LBM has been modified to allow to be used under a controller with dynamic load balancing.

Currently the best method to couple the interface variables between the is under investigation between the molecular level lattice Boltzmann method and the FVM macroscopic variables for a

compressible solver. The high Reynolds number Entropic Lattice Boltzmann method is being developed to investigate its possible to maintain real time flow calculation for more realistic Reynolds numbers.

8 ACKNOWLEDGMENTS

This work is funded under the Engineering and Physical Sciences Research Council Embedded CSE (EPSRC/eCSE) support grant eCSE05-04 which provides funding to develop software to run on ARCHER. The use of the UK National Supercomputing Service ARCHER, and the West of Scotland Computing service ARCHIE-WeSt are all gratefully acknowledged.

9 REFERENCES

[1] Horn, J. F., Bridges, D. O., Wachspress, D. A., and Rani, S. L., "Implementation of a free-vortex wake model in real-time simulation of rotorcraft," *Journal of Aerospace Computing, Information, and Communication*, Vol. 3, No. 3, 2006, pp. 93–107.

[2] Spence, G. T., Moigne, A. L., Allerton, D. J., and Qin, N., "Wake vortex model for real-time flight simulation based on Large Eddy Simulation," *Journal of aircraft*, Vol. 44, No. 2, 2007, pp. 467–475.

[3] Frisch, U., Hasslacher, B., and Pomeau, Y., "Lattice-Gas Automata for the Navier-Stokes Equation," *Phys. Rev. Lett.*, Vol. 56, Apr 1986, pp. 1505–1508.

[4] McNamara, G. R. and Zanetti, G., "Use of the Boltzmann Equation to Simulate Lattice-Gas Automata," *Phys. Rev. Lett.*, Vol. 61, Nov 1988, pp. 2332–2335.

[5] Chen, S. and Doolen, G. D., "Lattice Boltzmann method for fluid flows," *Annual review of fluid mechanics*, Vol. 30, No. 1, 1998, pp. 329–364.

[6] Aidun, C. K. and Clausen, J. R., "Lattice-Boltzmann method for complex flows," *Annual review of fluid mechanics*, Vol. 42, 2010, pp. 439–472.

[7] Khan, M. A. I., Delbosc, N., Noakes, C. J., and Summers, J., "Real-time flow simulation of indoor environments using lattice Boltzmann method," *Building Simulation*, Vol. 8, No. 4, 2015, pp. 405–414.

[8] Chapman, S. and Cowling, T. G., *The mathematical theory of non-uniform gases: An account of the kinetic theory of viscosity, thermal conduction, and diffusion in gases*, Cambridge University Press, 1991.

[9] He, X. and Luo, L.-S., "Lattice Boltzmann model for the incompressible Navier–Stokes equation," *Journal of statistical Physics*, Vol. 88, No. 3-4, 1997, pp. 927–944.

[10] Wellein, G., Zeiser, T., Hager, G., and Donath, S., "On the single processor performance of simple lattice Boltzmann kernels," *Computers & Fluids*, Vol. 35, No. 89, 2006, pp. 910 – 919.

[11] Mattila, K., Hyvluoma, J., Timonen, J., and Rossi, T., "Comparison of implementations of the lattice-Boltzmann method," *Computers & Mathematics with Applications*, Vol. 55, No. 7, 2008, pp. 1514 – 1524.

[12] Obrecht, C., Kuznik, F., Tourancheau, B., and Roux, J.-J., "The TheLMA project: A thermal lattice Boltzmann solver for the GPU," *Computers & Fluids*, Vol. 54, 2012, pp. 118 – 126.

[13] Chikatamarla, S., Ansumali, S., and Karlin, I., "Entropic lattice Boltzmann models for hydrodynamics in three dimensions," *Physical review letters*, Vol. 97, No. 1, 2006, pp. 010201.

[14] Karlin, I. V., Ferrante, A., and Öttinger, H. C., "Perfect entropy functions of the Lattice Boltzmann method," *Europhys. Lett.*, Vol. 47, 1999, pp. 182–188.

[15] Yu, H., Girimaji, S. S., and Luo, L.-S., "{DNS} and {LES} of decaying isotropic turbulence with and without frame rotation using lattice Boltzmann method," *Journal of Computational Physics*, Vol. 209, No. 2, 2005, pp. 599 – 616.

[16] Malaspinas, O. and Sagaut, P., "Consistent subgrid scale modelling for lattice Boltzmann methods," *Journal of Fluid Mechanics*, Vol. 700, 2012, pp. 514–542.

[17] Latt, J., Chopard, B., and Albuquerque, P., "Spatial coupling of a lattice Boltzmann fluid model with a finite difference Navier-Stokes solver," *arXiv preprint physics/0511243*, 2005.

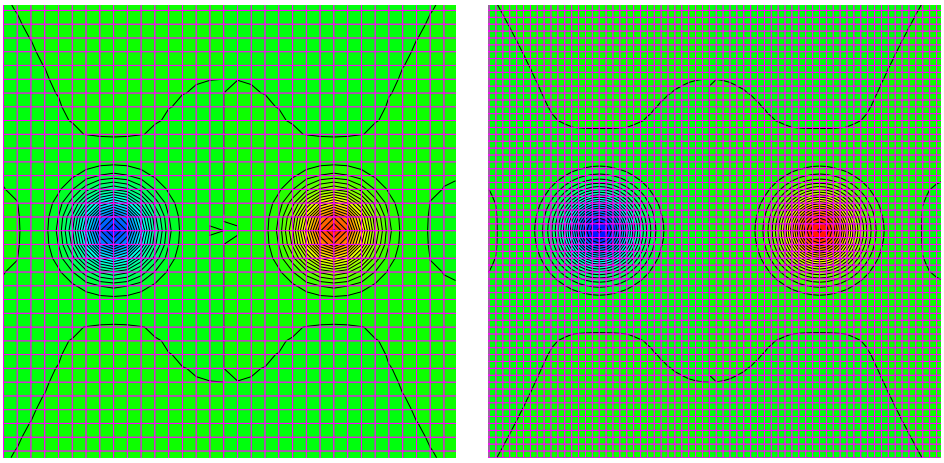
[18] Tong, Z.-X. and He, Y.-L., "A unified coupling scheme between lattice Boltzmann method and finite volume method for unsteady fluid flow and heat transfer," *International Journal of Heat and Mass Transfer*, Vol. 80, 2015, pp. 812 – 824.

[19] "Message Passing Interface Forum," http://www.mpi-forum.org/, Accessed: 01-07-2016.

[20] Latt, J., "Palabos an open-source CFD solver based on the lattice Boltzmann method," http://www.palabos.org/, Accessed: 01-07-2016.

[21] "UK National Supercomputing Service ARCHER," http://www.archer.ac.uk/, Accessed: 01-07-2016.

[22] Krause, M. J., "OpenLB Open source lattice Boltzmann code," http://www.optilb.org/openlb, Accessed: 01-07-2016.



Baseline Lattice Fine Lattice

Figure 6: The initial solution on the baseline and fine meshes for the Counter-rotating Vortex Pair

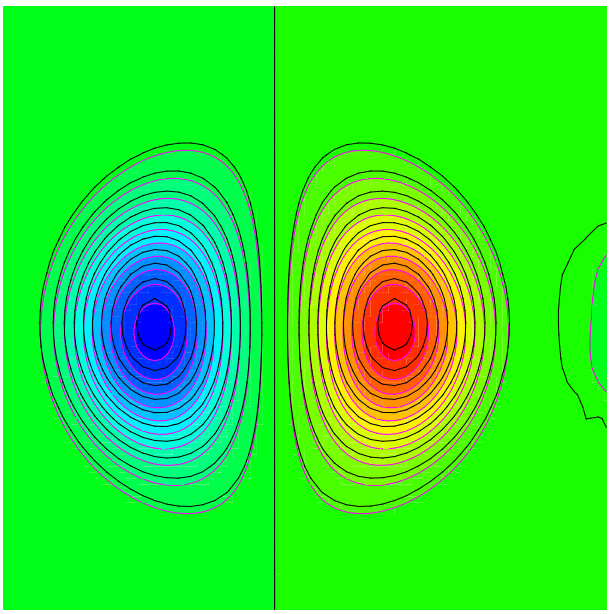


Figure 7: A comparison between the vorticity of the baseline, black contours, and fine lattice, purple contours, for the Counter-rotating Vortex Pair after four loops

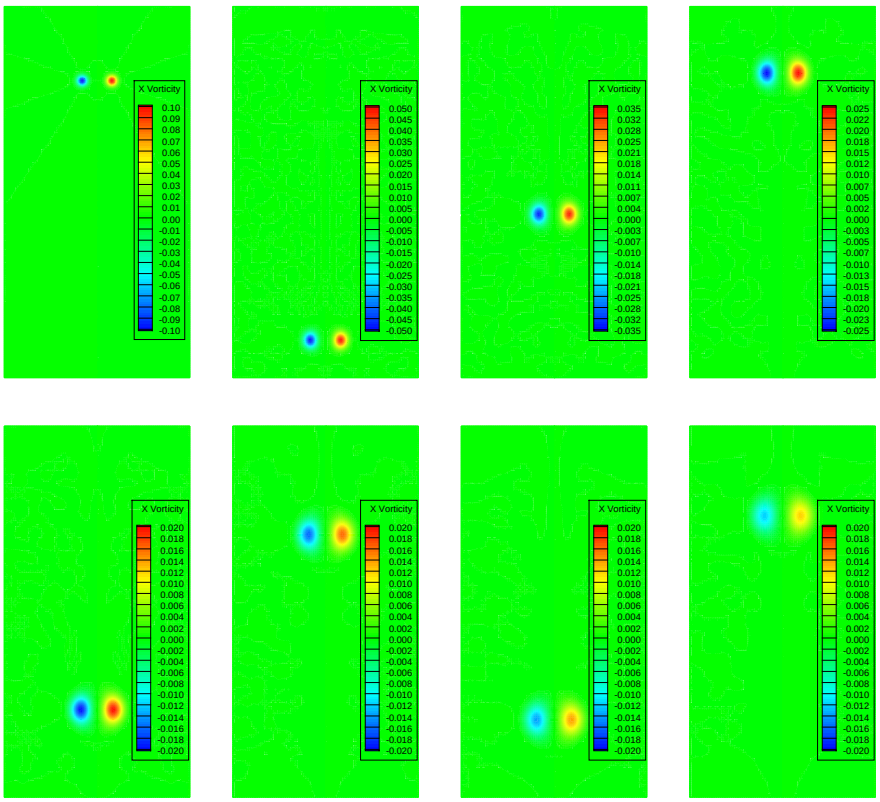


Figure 8: The vorticity of the baseline lattice for the Counter-rotating Vortex Pair

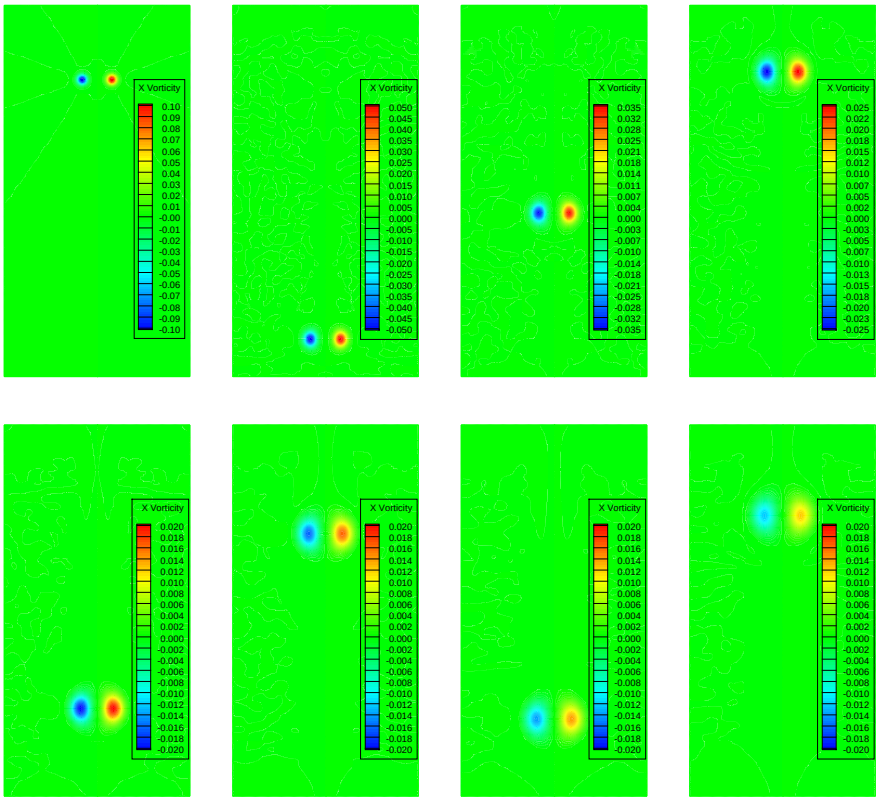


Figure 9: The vorticity of the fine lattice for the Counter-rotating Vortex Pair

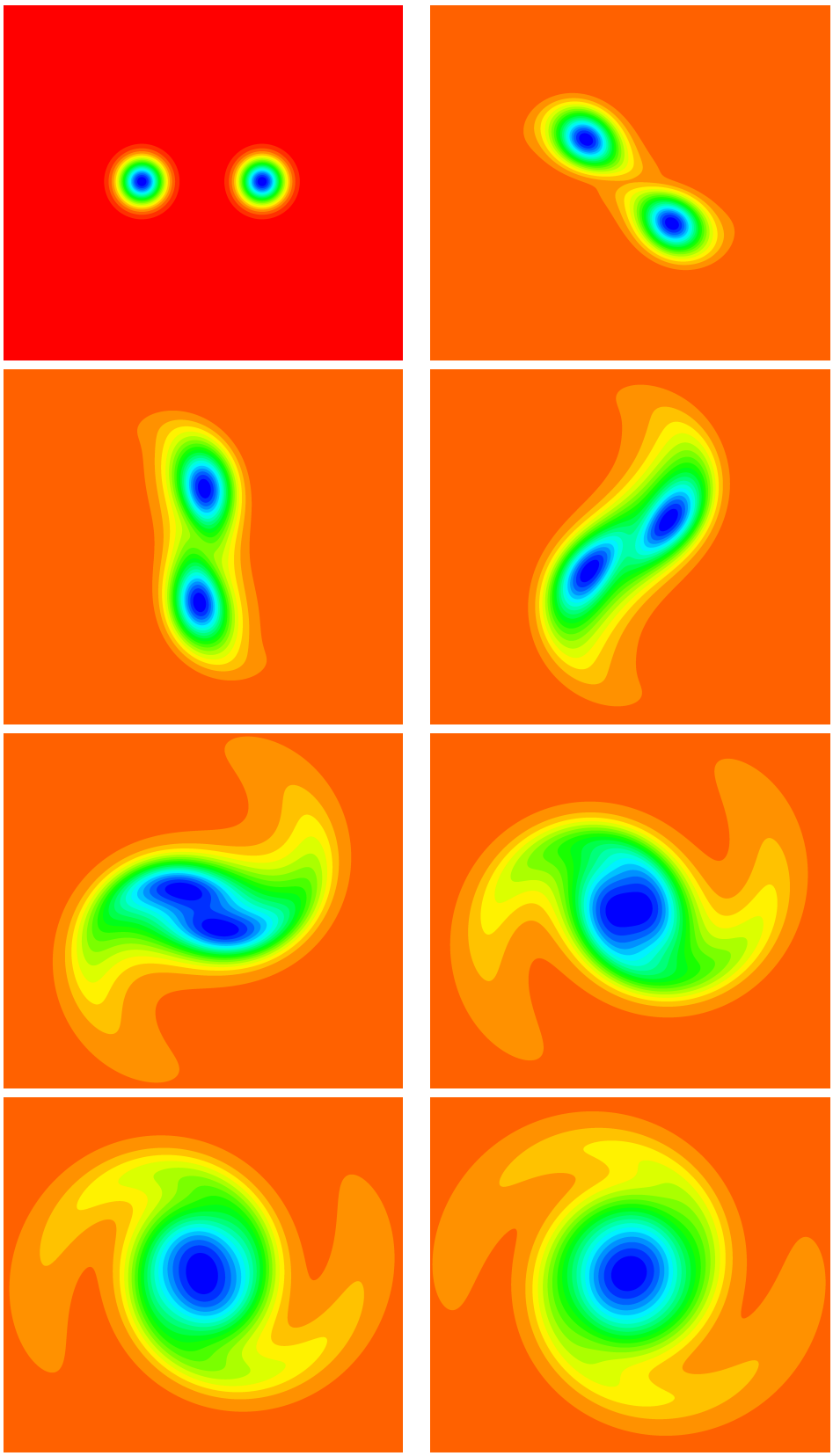


Figure 10: The merging of a Co-rotating Vortex Pair

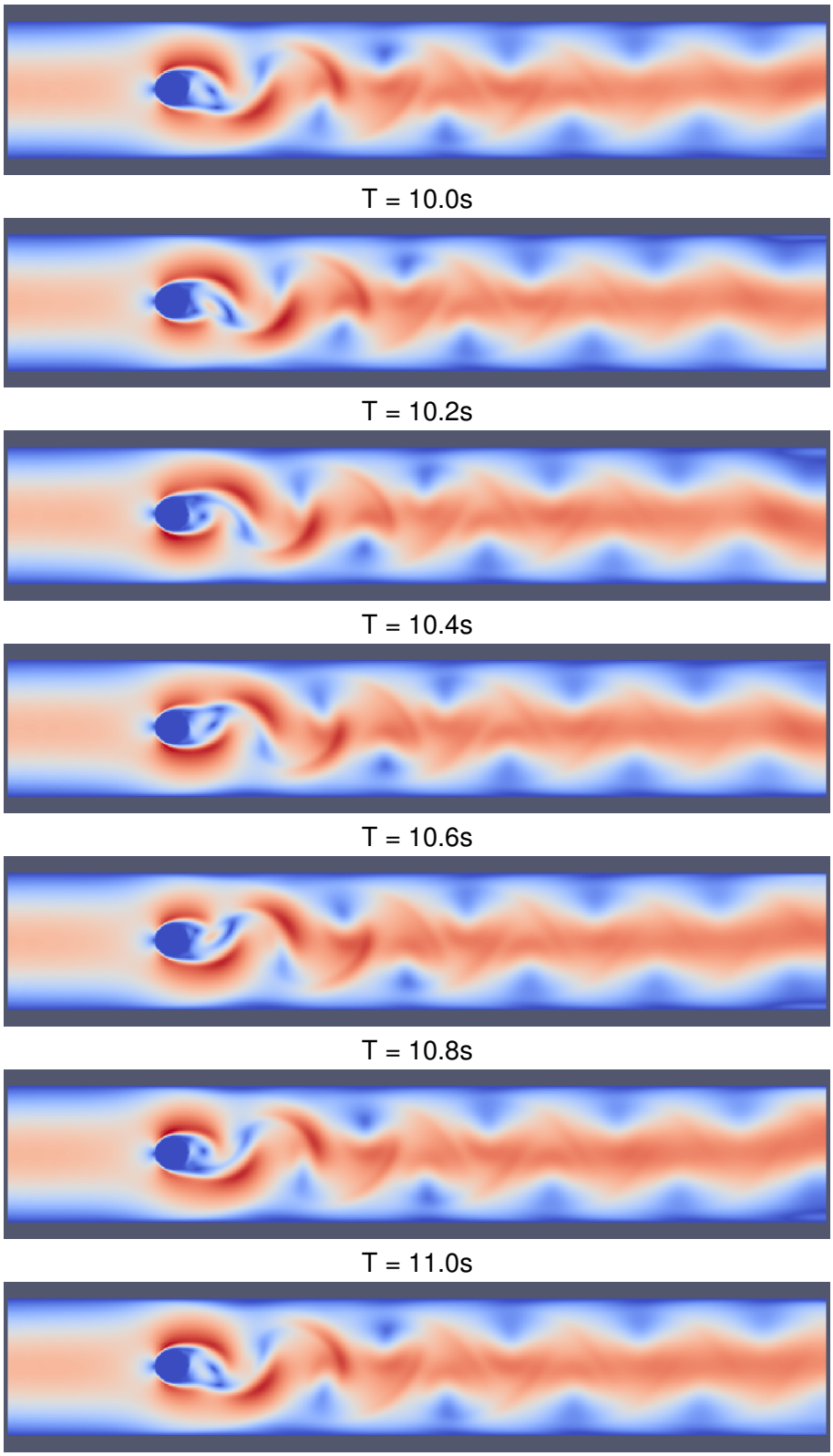


Figure 11: Flow past a cylinder Re=1000.0