43rd European Rotorcraft Forum
September 12-15, Milan, Italy, 2017
Paper 520

Real Time Wake Computations using Lattice Boltzmann Method on Many Integrated Core Processors

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.

Abstract

This paper puts forward an efficient Lattice Boltzmann method for use as a wake simulator suitable for real-time environments. The method is limited to low speed incompressible flow but is very efficient and can be used to compute flows “on the fly”. In particular, many-core machines allow for the method to be used with the need of very expensive parallel clusters. Results are shown here for flows around cylinders and simple ship shapes.

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, simplified aerodynamic models, and linearized CFD-based reduced order 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. Recent developments have included the use of the free vortex wake method on graphic cards in order to achieve real-time capability [3] and the use of dynamic inflow model extracts the inflow velocities from a real-time Lattice-Boltzmann fluid simulation and passes them to a blade element based flight dynamics code to capture the rotorcraft motion [4]

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. INTELs second generation Many Integrated Core Architecture (MIC) uses between 64 and 72 Airmont (Atom) cores with four threads per cores and so has a core count between CPUs and GPUs. The Intel Xeon Phi is currently installed on 7 of the top 20 fastest HPC systems according to the 49th Top500 List of June 2017. Out of these, 5 are the second generation Knights Landing units.

This increasing number of cores makes running real time simulations much more feasible but the employed schemes will have to take advantage of such a large number of processors by carefully choosing algorithms that decompose into a large number of semi-independent operations as well as being able to exploit the underlying core architecture to the full. In the last 20 years the lattice Boltzmann method (LBM) has emerged as an alternative to the more traditional methods for simulating fluid flow. The LBM was developed as an extension to lattice gas automata [5, 6] and reviews of the developments since then can be found in [7, 8]. The LBM method has good parallelism with some benchmark running on 2^{14} processor cores but the discrete probability distribution functions require more memory for their storage than the hydrodynamic variables of the Navier-Stokes equations (19 real valued quantities per node against 7 for 3D flows). Some versions of the LBM also involve only a very limited amount of floating points operations per computational node resulting in the method being limited by memory bandwidth rather than arithmetic performance when computed by using general-purpose processors.

Recently Khan, *et al.* [9] 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 LBM uses a simplified kinetic model which includes the essential microscopic effects to encapsulate the macroscopic averaged quantities of the Navier-Stokes equations which is achieved by solving the discrete-velocity Boltzmann equation. A regular lattice is used for the domain and a particle distribution function represent the probability of a particle having a given velocity at each lattice point. The movement of the particles is restricted to a subset of neighbouring lattice points. The discrete collision rule is replaced by an approximate collision operator with the Bhatnagar-Gross-Krook (BGK) model being the most widely used (see [7]). A common labelling for the lattices in the LBM is 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 in this work to keep the computational cost low while maintaining an isotropic lattice.

This is solved numerically by decomposing it into a two step process. First the collision step where

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

where f_i represents the particle distribution function which is the fraction of particles located at position $\bar{x}$ at time t moving with the microscopic velocity $\bar{e}_i$, 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 then followed by a streaming step where the value of $f_i^t(\bar{x}, t+\delta t)$ is shifted in space along the lattice velocity $\bar{e}_i$,

(2) $$f_i(\bar{x} + c\bar{e}_i\delta t, t+\delta t) = f_i^t(\bar{x}, t+\delta t),$$

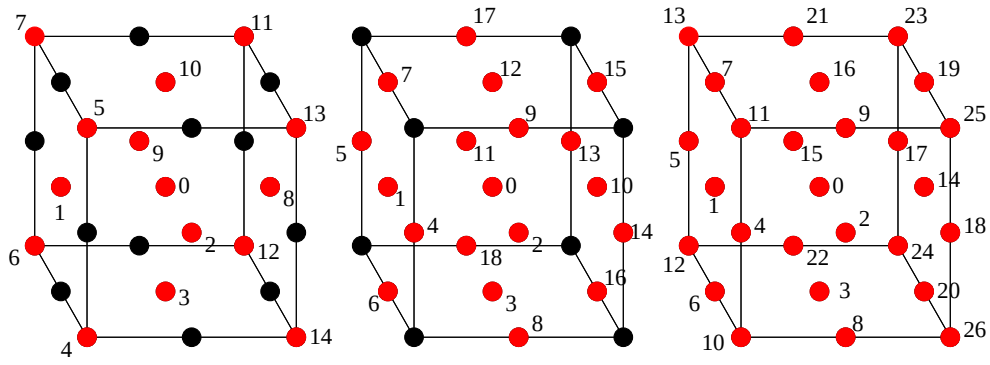


Figure 1: The common three dimensional lattices with the indices re-ordered to minimize the traversal of memory and hence reduce the memory bandwidth requirements

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 $f_i^{eq}(\rho, \bar{u})$ 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 $\bar{u}$ is the value of the velocity.

The density ρ and the velocity $\bar{u}$ are obtained from the zeroth and first moments of the distribution functions

$$\rho = \sum_{i=0}^{18} f_i, \qquad \rho\bar{u} = \sum_{i=0}^{18} c\bar{e}_i f_i, \tag{3}$$

and the discrete velocity set $\bar{e}_i$ is defined as follows:

$$\bar{e}_i = \begin{cases} (0,0,0) & i = 0 & w_i = 1/3 \\ (\pm 1,0,0) & i = 1-2 & w_i = 1/18 \\ (0,\pm 1,0) & i = 3-4 & w_i = 1/18 \\ (0,0,\pm 1) & i = 5-6 & w_i = 1/18 \\ (\pm 1,\pm 1,0) & i = 7-10 & w_i = 1/36 \\ (\pm 1,0,\pm 1) & i = 11-14 & w_i = 1/36 \\ (0,\pm 1,\pm 1) & i = 15-18 & w_i = 1/36 \end{cases}. \tag{4}$$

The equilibrium state is calculated by

$$f_i^{eq}(\rho, \bar{u}) = \rho w_i \left(1 + \frac{3\bar{e}_i \dot{} u}{c} + \frac{9(\bar{e}_i \dot{} u)^2}{2c^2} - \frac{3\bar{u}^2}{2c^2}\right) \tag{5}$$

where the w_i are the weight coefficients defined in equation 4.

It can be shown through a Chapman-Enskog expansion (see [10]) that the Navier-Stokes equations can be obtained from the lattice BGK model. First by using a 2nd order Taylor series expansion about the left hand side of equation 1 the particle distribution function is split into equilibrium and non equilibrium components. After using the Chapman-Enskog expansion, which expands the non equilibrium part in a power series of the Knudsen number, the Taylor series can be decomposed into different orders of magnitude of the Knudsen number to obtain the continuum equations which recover the Navier-Stokes equation assuming the density variation is small.

2.1 High Reynolds Number Flows

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

$$\tau = 0.5 + 3\nu_{lb} = 0.5 + u_{lb}(N-1)/Re \tag{6}$$

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 [11, 12] which allows the Lattice Boltzmann models to support a discrete H-theorem through the use of a modified equilibrium distribution function

$$f_i^{eq} = \rho w_i \prod_{\alpha=1}^{3} \left(2 - \sqrt{1+u_\alpha^2}\right) \left(\frac{2u_\alpha + \sqrt{1+3+u_\alpha^2}}{1-u_\alpha}\right)^{e_{i\alpha}} \tag{7}$$

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

$$H(f) = H(f - \beta(f - f^{eq})) \tag{8}$$

which produces an unconditional stable numerical scheme.

2.2 Turbulence Model

For turbulent flow calculations a Smagorinsky subgrid scale model [13] 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.

$$\nu = \nu_0 + \nu_t \tag{9}$$

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

$$Q_{\alpha\beta} = \sum_i e_{i\alpha} e_{i\beta} (f_i - f_i^{eq}), \tag{10}$$

and

$$\tau_t = \frac{1}{2}\left(\sqrt{\tau_0^2 + 4c_s^{-4} C_S^2 (Q_{\alpha\beta} Q_{\alpha\beta})^{1/2}} - \tau_0\right) \tag{11}$$

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.

Malaspinas [14] proposed a consistent way of including sub-grid closure models in the BGK Boltzmann equation for large eddy simulations of turbulent flows. The derived the terms based on a Hermite expansion of the velocity distribution function and showed a connection between the new models and the current standard practice showing that a single modified scalar relaxation time to account for the sub-grid viscosity is not consistent in the compressible case.

2.3 Bounce-back boundary conditions

These boundary conditions are used to implement both slip/symmetry and no-slip wall boundary conditions. In this boundary condition when the distribution function streaming reaches the boundary node it will scatter back into the fluid. The two boundary types are implemented by changing the direction in which the distribution function is scattered.

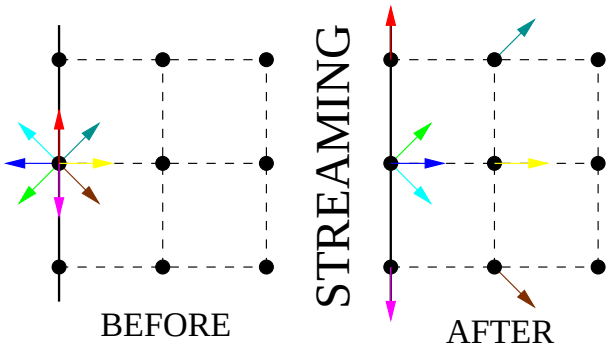


Figure 2: The distribution function for a boundary node for the full bounce-back condition before and after streaming.

For the full bounce-back the incoming directions of the distribution function are reversed when they hit a boundary node and this process does not require the orientation of the boundary. So complex geometries require no extra computation. The streaming of the full bounce-back can be seen in figure 2. It should be noted that this boundary condition acts half way between nodes and not at the boundary node. For a general geometry the lattice points inside the solid need to be flagged as such. Both a lattice and a STereoLithography (STL) file of the geometry are needed, and a simple utility code can be written to return all the lattice points contained inside the geometry. An example of this can be seen for the Simple Frigate Shape 2 (SFS2) used by [15] in figure 4. Since the geometry has been rotated by 15 degrees none of the surfaces align with the lattice an so a ”staircase” formation is obtained on every surface. At the current resolution there are just enough points to resolve the stack on the superstructure of the SFS2.

3 Knights Landing

The Knights Landing chip is etched in 14 nanometer manufacturing processes with over 8 billion transistors. The chip contains 36 tiles interconnected by a two dimensional mesh. Each tile consists of 2 Cores, 4 AVX512 512-bit Vector processing units (VPUs) and 1MB 16-Way of L2 Cache, which is coherent across all tiles, as shown in figure 3. Each of the computer nodes contains a 64-core KNL processors (model 7210) running at 1.3Ghz. These offer a large amount of floating point performance, (3TFlops peak using double precision) and the hardware is a significant step forward from the previous generation of Xeon Phis.

Figure 3: Schematic of a tile within a KNL processor.

The code was evaluated on nodes configured in cache mode with all 16GB of the on-chip Multi-Channel DRAM (MCDRAM) used to cache the system memory, and job sizes were small enough so all the data could fit within the cache. The MCDRAM is a high bandwidth memory which fits well with the needs of a LBM.

4 PERFORMANCE

The following section first presents the LBM parallel performance on a $121 \times 241 \times 241$ lattice containing 7 million lattice points with periodic boundary condition in all three directions. The lattice used a Cartesian partition of $NX_p \times NY_p \times NZ_p$ equal sized blocks. The total number of blocks equalled the total number of processors to maximize the parallel performance.

The details of the current implementation of the Helicopter Lattice Boltzmann Method (HLBM) code can be found in [16]. Table 1 shows the parallel performance of the HLBM code computed within node of the Advanced Research Computing High End Resource (ARCHER) which is the UK National Supercomputing Service. There is a marked drop off in parallel performance when running on more that 4 cores per CPU - 8 cores in total. This is because the method is very memory bandwidth intensive and general memory bandwidth has not kept pace with the ever increasing number of cores on CPUs.

Cores	Time	Efficiency
1	2.1249s	N/A
2	1.0721s	99.1%
4	0.56952s	93.2%
8	0.34484s	77.0%
16	0.23538s	56.4%
24	0.21953s	41.3%

Table 1: Performance scaling of HLBM within a node of two Intel 2.7GHz 12-core E5-2697 v2 Processors

As stated earlier the HLBM code was evaluated on KNL nodes configured in cache mode with all 16GB of the on-chip Multi-Channel DRAM (MCDRAM) used to cache the system memory, and job sizes were small enough so all the data could fit within the cache. The MCDRAM is a high bandwidth memory which fits well with the needs of a LBM. The results can be seen in table 2 and although the single core performance of a KNL processor was three times slower, mainly due to the lower clock speed, the parallel scaling was much better at high number of cores. Hence 24 nodes on an ARCHER computer node run the same as 32 on a KNL node. This results in the KNL nodes being 80% faster when both nodes are full utilized.

Cores	Time	Efficiency
1	6.772s	N/A
2	3.503s	96.6%
4	1.743s	97.1%
8	0.880s	96.2%
16	0.442s	95.6%
32	0.226s	93.6%
64	0.126s	84.0%

Table 2: Performance scaling of HLBM within a 64-core KNL processor (model 7210) running at 1.3Ghz

It should be noted that the current version of HLBM has extensive inner loops unrolling and hence make no use of the 2 vector processing units (VPU) per core. A listing which takes full advantage of the VPUs might increase the single core performance and is currently under investigation.

When the KNL nodes are configured in flat mode the 16GBytes of local MCDRAM becomes available for explicit use within the code instead of caching the system memory. Since the memory footprint of the test case is smaller than 16GBytes is possible to run totally within the local MCDRAM and this results in a similar run times as running in cached mode. However if the system memory is used instead of the local MCDRAM memory the runtime is increased from 0.126 seconds to 0.2942 seconds which highlights the large detrimental on the algorithm when using a slower memory sub system.

Finally table 3 shows the performance of HLBM when run across multiple nodes. For the Intel Xeon nodes the performance across nodes shows linear speedup because the number of lattice points per process dropped from $288,000$ to just 4500 when

on 64. This means a much larger percentage of the data could be stored in the cache which increases the core performance. It has been found the HLMB performs about twenty percent faster when running on a small block size and this sequential performance gain offsets the communication costs. The scaling across the nodes is not quite as good for the KNL nodes but us still above 90%.

5 RESULTS

For the real time flow around the cylinder, the lattice size was set to 600×100 with a lattice spacing of 0.01, a lattice velocity of 0.1 and a Reynolds number of 1000. This means that 1000 time steps are required to simulate a second of real time. With the given lattice size, at least 60 Million lattice updates a second are required which is possible within a single ARCHER node. It should be noted that this ignores data IO which is around 25% of the total run time due to 24 flow fields being outputted per second for flow animation. The full cycle of the shedding is shown in Figure 5. The inlet was free steam, and the boundary layer starts to develop on the upper and lower walls. Due to the vortex shedding and the close proximity of the walls the vortical structures interact with the wall boundary layer. The lattice spacing is enough to resolve the flow features while the lattice speed is high, the equilibrium function was truncated to second order and so terms of the size lattice speed cubed have been dropped.

A second real time test case is the flow around the planform of the simple frigate shape 2 [15] a larger domain was used being 1600×400 due to the much larger object in the flow. The lattice spacing was set to 1 with the lattice velocity equal to 0.1. The Reynolds number based on the length of the frigate was 1500. Since the lattice velocity is the same as in the cylinder case but the lattice itself is ten times larger a real time computation requires 640 Million lattice updates a second. This required 4 ARCHER KNL nodes to maintain this performance. It should be noted that to convert the flow field data into TECPLOT format, calculate the vorticity and output the frame as a picture was of the order of 20 seconds each meaning the post processing of the complete flow field cannot even keep up when running on a single core. Hence the I/O needs to be reworked so firstly it is written in the native format of the post processing tools to avoid time consuming conversions between formats, and secondly output a restricted subset of the domain.

The computation was run with wall above and below the channel with inflow and outflow boundary conditions. The results for the zero degree headwind can be seen in figure 6 and clearly shows an unsteady wake forming behind the vessel. For the case with the frigate at 10 degrees to the flow, shown in figure 7 the vortical structures are excited faster and so shedding is initiated more quickly. There are many more vortical structures with a stream of vortices generated from the leading edge and passing alongside the frigate. It should be noted that the top and bottom walls are close enough that they effect the trajectory of the vortical structures aft of the vessel. Since only the angle of the frigate was changed this does not effect the computational cost of the calculation.

For a three dimensional case the top and bottom wall were change from no-slip to slip so as to remove the boundary layer formation since at the current Reynolds number of 650 most of the vessel would have been contained within the boundary layer. The inflow and outflow boundary conditions where also replaced, with periodic conditions, since the restriction they placed in the Reynolds number caused the flow to become steady. This also allows for a smaller τ to be used and hence a higher Reynolds number. The third dimension also had periodic boundaries. The last change that was the lattice velocity was reduced to 0.07 since the maximum speed in the two dimensional simulation was around 0.15 which is probably too high for the approximation for the equilibrium equation. This does mean that 30% more time steps are needed for each flow second, and so this adds 30% to the computational resource required to obtain real time computations.

The lattice size was $900 \times 100 \times 240$, so the require performance needed to obtain a real time computation was of the order of 30800 million updates a second. With the current performance of just over 130 million updates per second for an ARCHER node this means of the order of 240 nodes or some 5760 cores. And while these numbers are feasible they are not within the reach for facilities currently linked flight simulators. For KNL processors this would be brought down to 130 nodes with the current implementation with the chance of better performance if the VPUs are utilized.

Results are presented on the three cut planes

shown in figure 8. The first plane is parallel to the boundary half way up the ship hull. The second cut plane is 66% along the landing deck at the rear of the vessel while the last plane is a cut through the center line. The results at 15 degrees can be seen in figures 9 for behind the vessel, figure 10 for the cut through deck and figure 11 for a cut through the center line of the vessel. The results shown many more vortical structures on the starboard side due to the wakes generated by both the bow and superstructure of the vessel. Theses vortical structures are then advected aft. Due to the low Reynolds number and coarseness of the lattice, the flow field above the deck is almost steady but does show a pocket above the deck with very low vorticity levels.

6 CONCLUSIONS

This paper presents the details of implementing the LBM method efficiently on several parallel platforms. The main algorithm was re-written to allow for the maximum LB updates per second. Additional modification were put in place to allow for the exploitation of modern CPU's like the KNL systems. The proposed coding is both efficient and easy to understand, and stems naturally out of a straightforward LBM implementation. Real time execution is possible on large computer clusters and the use of KNL opens the gate for linking high performance clusters with real-time wakes in flight simulators.

Future work is currently directed towards furthering the real-time flow capabilities using VPUs and improving the the algorithm. As an investigation in using a Hybrid MPI/OpenMP code with dynamic thread scheduling to allow for a more balanced work-load, is also planned.

7 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 and carried out in collaboration with Dr. Gavin Pringle of the EPCC. The use of the UK National Supercomputing Service ARCHER is gratefully acknowledged.

8 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] Rubenstein, G., Moy, D. M., Sridharan, A., and Chopra, I., "Python-based Framework for Real-time Simulation using Comprehensive Analysis," *Proceedings of the AHS International 72nd Annual Forum*, West Palm Beach, Florida, USA, May 2016.

[4] Bludau, J., Rauleder, J., Friedmann, L., and Hajek, M., "Real-Time Simulation of Dynamic Inflow Using Rotorcraft Flight Dynamics Coupled With a Lattice-Boltzmann Based Fluid Simulation," *55th AIAA Aerospace Sciences Meeting*, Grapevine, Texas, 9-13 Jan 2017, AIAA 2017-0050.

[5] 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.

[6] 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.

[7] 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.

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

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

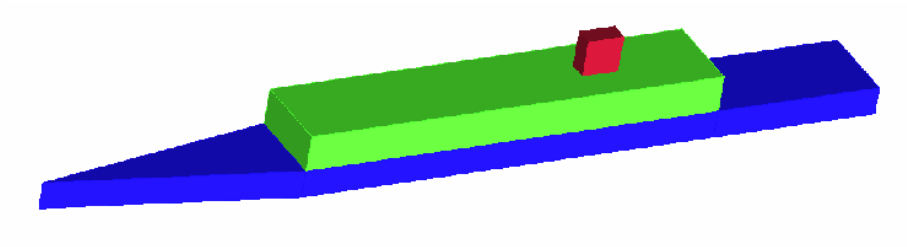
[14] Malaspinas, O. and Sagaut, P., "Consistent subgrid scale modelling for lattice Boltzmann methods," *Journal of Fluid Mechanics*, Vol. 700, 2012, pp. 514–542.

[15] Crozon, C., Steijl, R., and Barakos, G. N., "Numerical Study of Helicopter Rotors in a Ship Airwake," *Journal of Aircraft*, Vol. 51, No. 6, 2014, pp. 1813–1832.

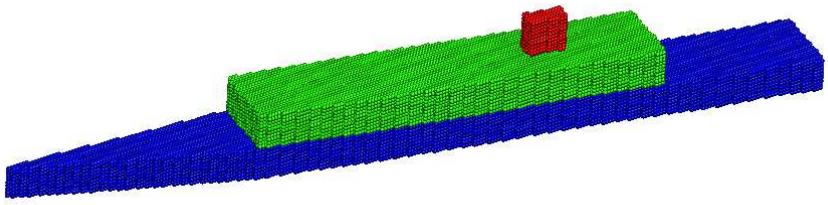
[16] Woodgate, M. A., Barakos, G. N., Steijl, R., and Pringle, G. J., "Parallel Performance for a Real Time Lattice Boltzmann Code," *29th Parallel CFD Conference*, Glasgow, Scotland, 15-17 May 2017.

Number of Nodes	Xeon Nodes (24 Cores)		KNL Nodes (64 Cores)	
1	0.21195s	N/A	0.12835s	N/A
2	0.10560s	104%	0.06350s	101%
4	0.05050s	108%	0.03305s	97%
8	0.02586s	106%	0.01729s	93%
64	0.00319s	107%	N/A	N/A

Table 3: Performance scaling of HLBM across both Intel Xeon and KNL cores

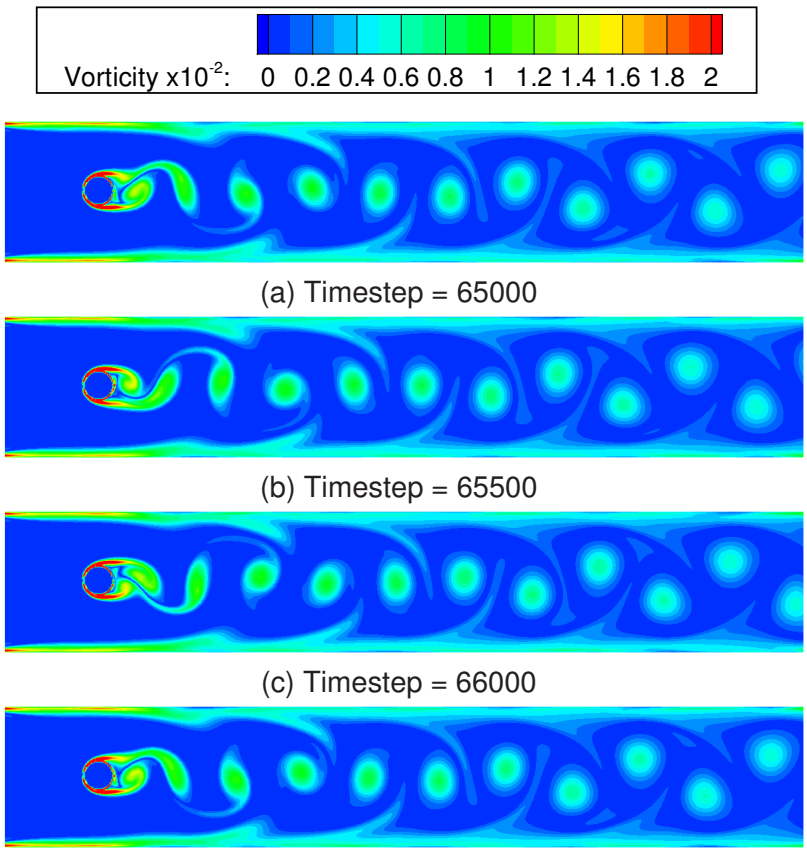


(a) STereoLithography file of SFS2.



(b) Flagged lattice points for SFS2 after rotation by 15 degrees.

Figure 4: Flagging of bounce-back lattice points for an STL geometry.



(a) Timestep = 65000

(b) Timestep = 65500

(c) Timestep = 66000

(d) Timestep = 66500

Figure 5: The vorticity magnitude for flow around a cylinder within a channel

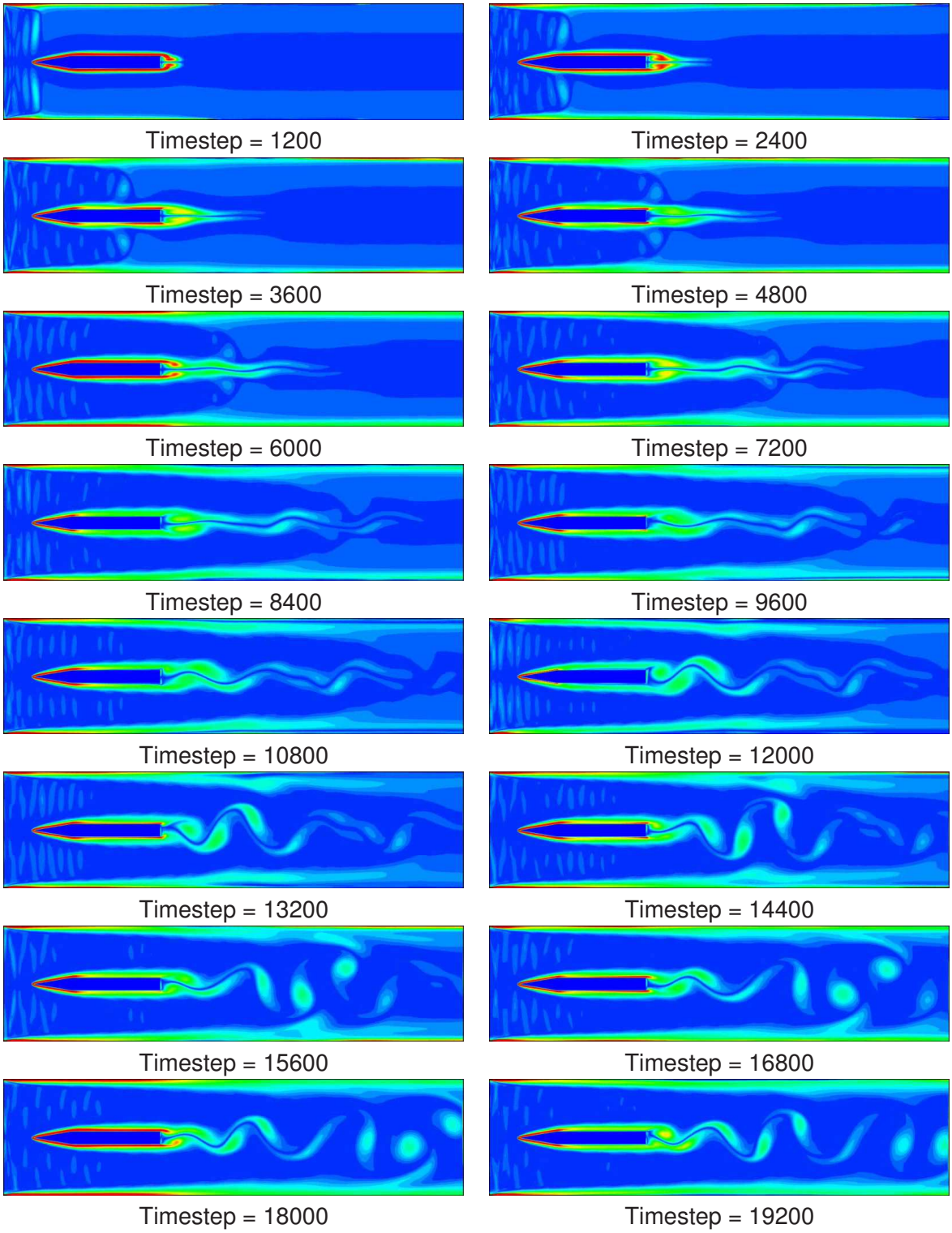


Figure 6: The vorticity magnitude for flow around the planform of the SFS2 at zero degrees.

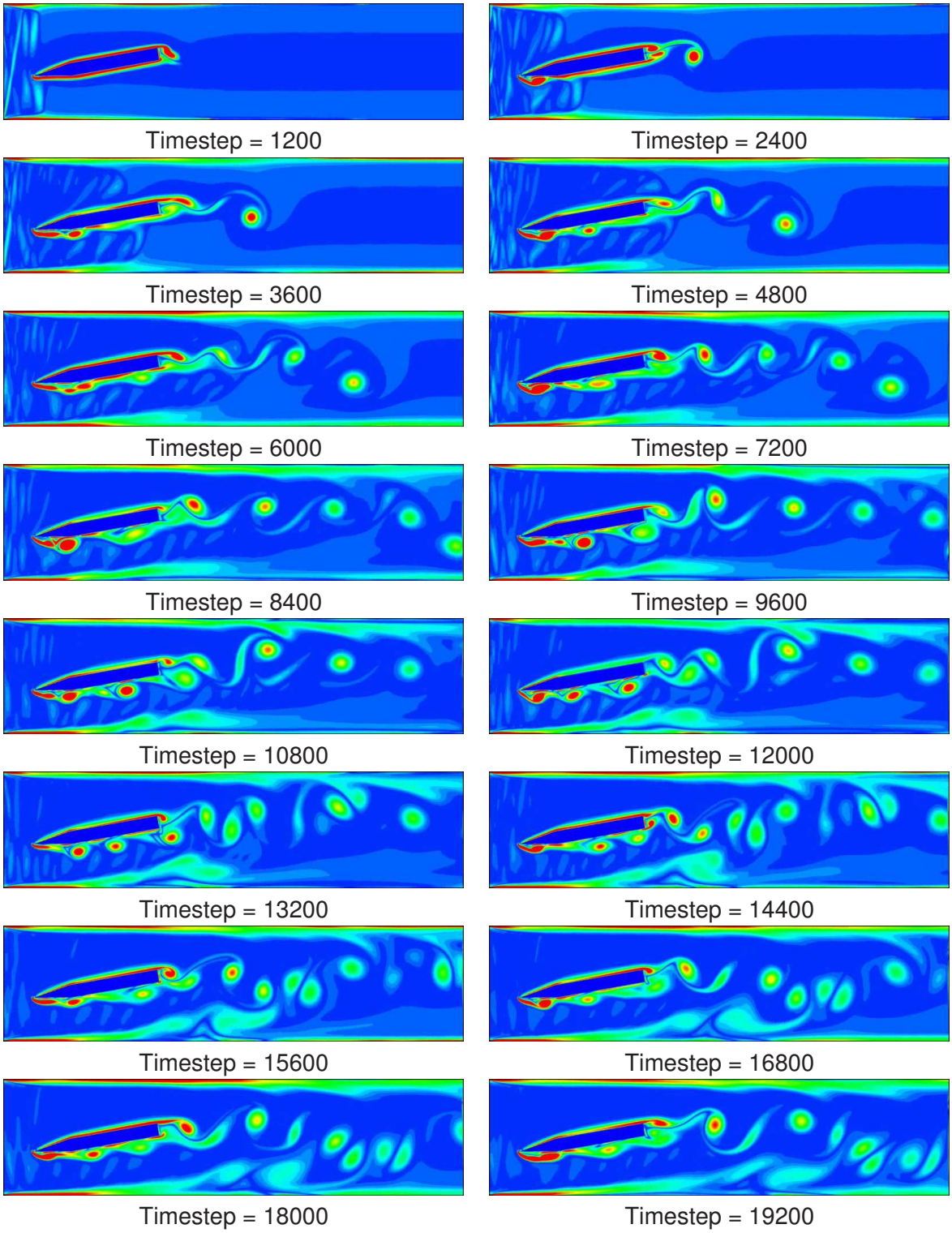


Figure 7: The vorticity magnitude for flow around the planform of the SFS2 at 10 degrees.

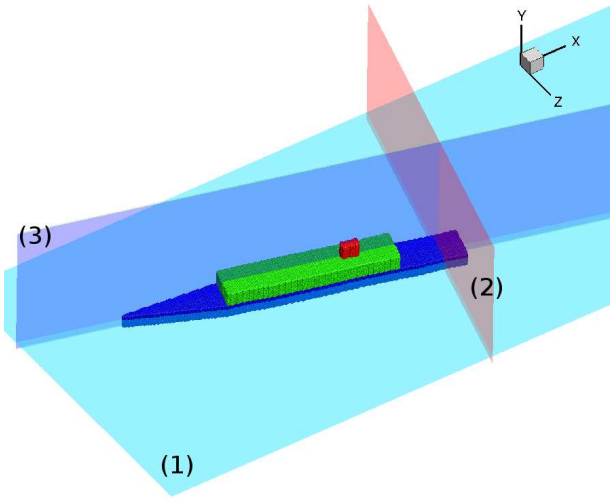


Figure 8: Position of the three cut planes with respect to the SFS2 geometry.

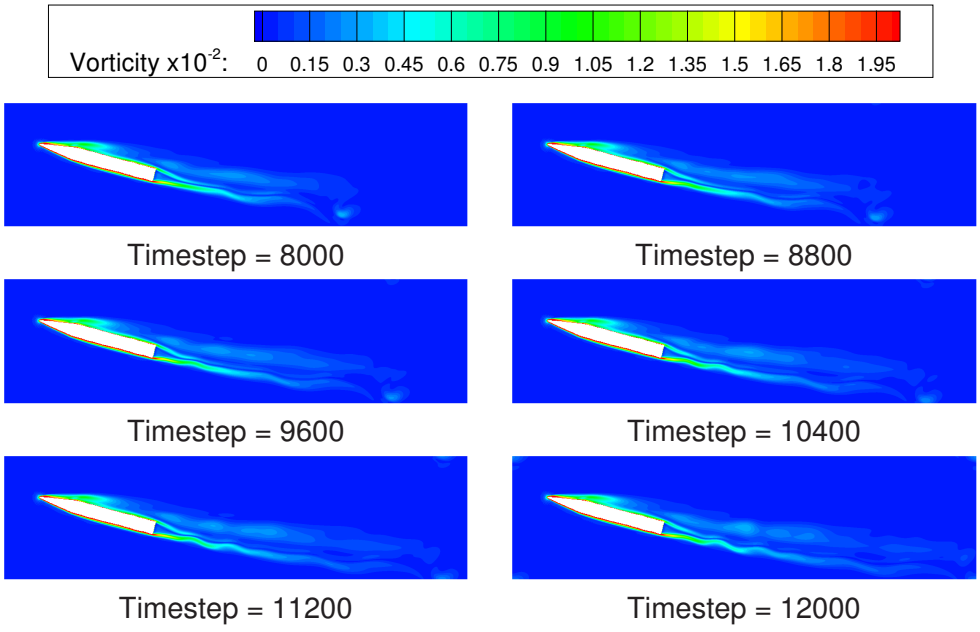


Figure 9: The vorticity magnitude for flow around the SFS2 at 15 degrees on the first cut plane.

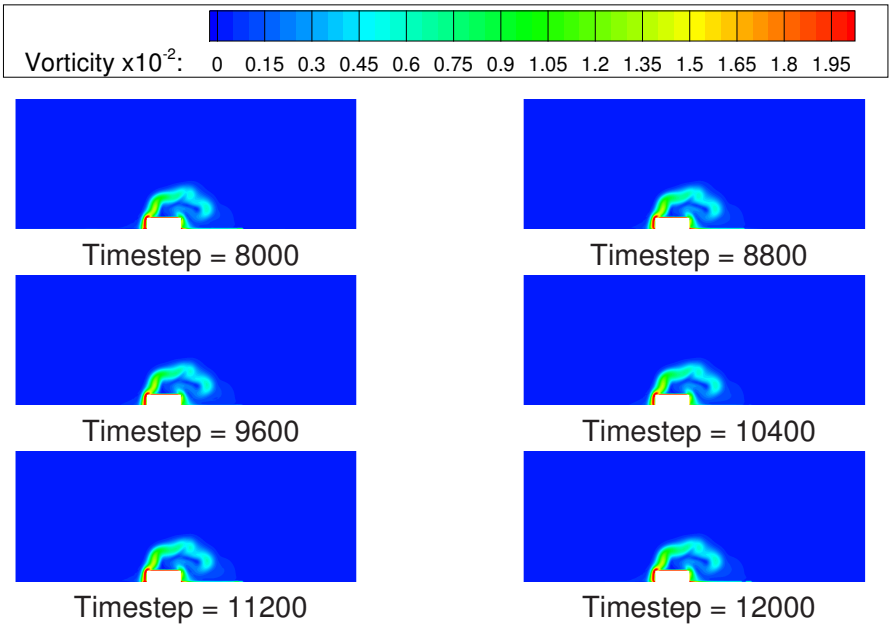


Figure 10: The vorticity magnitude for flow around the SFS2 at 15 degrees on the second cut plane.

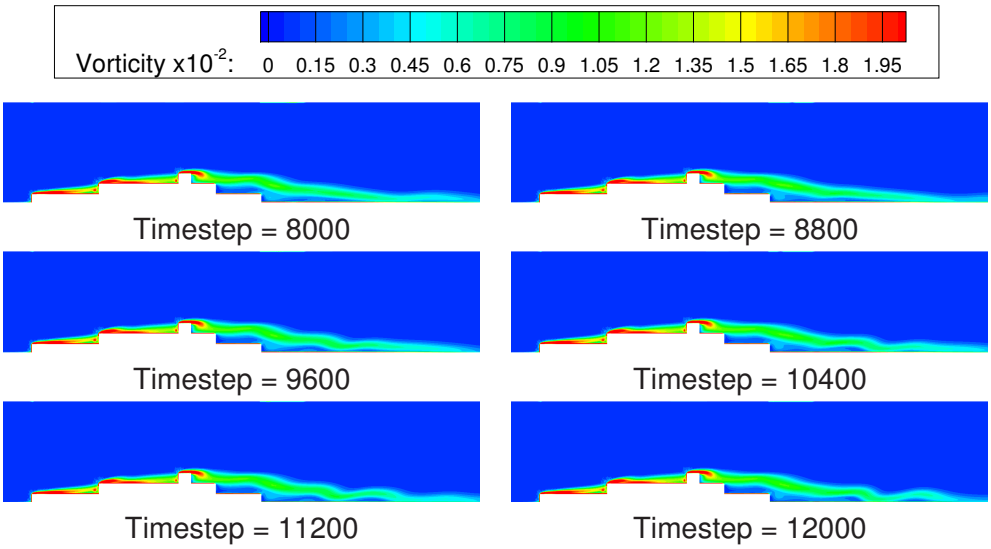


Figure 11: The vorticity magnitude for flow around the SFS2 at 15 degrees on the third cut plane.