6th European Conference on Computational Mechanics (ECCM 6)
7th European Conference on Computational Fluid Dynamics (ECFD 7)
11–15 June 2018, Glasgow, UK

A REAL-TIME MODELLING AND SIMULATION PLATFORM FOR VIRTUAL ENGINEERING DESIGN AND ANALYSIS

ADRIAN R. G. HARWOOD[1], PETRA WENISCH[2] and ALISTAIR J. REVELL[1]

[1] School of Mechanical, Aerospace & Civil Engineering
The University of Manchester, Oxford Road, Manchester, M13 9PL, UK
adrian.harwood@manchester.ac.uk

[2] University of Applied Sciences Potsdam, Faculty of Civil Engineering
Kiepenheuerallee 5, 14469 Potsdam, Germany
wenisch@fh-potsdam.de

Key words: Virtual Reality, Game Engines, Mobile Devices, GPU Computing, Lattice-Boltzmann Method

Abstract. The ability to perform credible CFD simulations at accelerated speeds has opened up the potential for a new use-mode for CFD as a tool in engineering: the application of CFD for first-order parameter-space exploration, analysis, and design communication. When coupled with a suitable real-time rendering and interaction capability for in-situ visualisation and manipulation of 3D results, CFD may be used as part of an interactive design tool in virtual engineering. These steerable applications represent a paradigm shift in the application of CFD for engineering and offer the potential to transform the way CFD is used within the industry.

This article presents developments towards a production-ready virtual wind tunnel including presentation of an integrated, interactive modelling and simulation tool for aerodynamic design and analysis built using the Unreal Engine 4 game engine. The virtual wind tunnel application provides a mechanism for integrating virtual reality observation, navigation, visualisation and in-game interaction with a flow field simulated using our own GPU-accelerated CFD library based on the lattice-Boltzmann method. Objects may be imported from CAD or reconstructed using Microsoft Kinect-based 3D scanning. Simulation parameters may be modified at run-time by the user.

The flow solver has been validated against experimental data for a representative turbulent flow and demonstrates excellent agreement with available data.

1 INTRODUCTION

Modelling and simulation is an essential part of engineering. Engineers can conduct a wide range of tests virtually, avoiding the effort and expense of physical testing. Modelling

and simulation in engineering is typically an activity requiring significant time, effort and computing power, with high-accuracy the principal aim. However, there is a growing acknowledgement in industrial circles that lower order, broad parameter space mapping is sometimes more valuable than higher order analysis of far fewer design points.

A real-time simulation platform is a manifestation of this concept, allowing run-time manipulation of geometrical and physical simulation variables. This enables users to rapidly and intuitively investigate different scenarios and design configurations. Ultimately, a complete interactive simulation package would be capable of simulating a multi-physics 3D environment in real-time to an application-appropriate degree of accuracy. However, significant inter-disciplinary research is required in reduced-order physical modelling, numerical methods, and software integration to realise a solution.

In this paper we present progress towards the production of a 3D interactive, real-time simulation for fluid flow. We present a recent realisation of a virtual wind tunnel which uses GPU-accelerated CFD and virtual reality to facilitate an interactive 3D flow environment for automotive design.

1.1 Real-time Simulation

The definition of real-time simulation needs establishing in the current context. In the field of computing, a real-time system is one which responds to requests within an application-specific time window, usually of the order of milliseconds. Throughout this paper, when we refer to real-time simulation, we mean a simulation which updates at a rate suitable for an observer to see appreciable change in the state of the simulation, sometimes termed 'interactive' simulation. A more strict definition of a real-time simulation would be one where the simulation takes 1 seconds to simulate 1 second of physical behaviour. As discussed by Harwood and Revell,[1] this can be difficult to achieve without significant compromises in accuracy, stability or simulation domain size.

2 A GPU-ACCELERATED LBM SOLVER

In order to achieve real-time flow simulation, numerical methods need to be selected carefully such that they can make full use of the capabilities of accelerated computing hardware. Our work focuses on the use of the lattice-Boltzmann Method (LBM):[2] a CFD method ideally suited to acceleration on GPUs due to its spatial and temporal locality. GPU-LBM simulations have extremely high computational throughput compared with traditional CFD methods.[3,4]

The LBM solves a lattice-discretised Boltzmann transport equation Eq. (1) in two steps – the 'streaming' step and the 'collision' step as indicated in Fig. 1. The transport quantity f_i represents the probability of finding particles at a given lattice node with velocity $\vec{c}_i$. These are termed probability density functions or simply 'populations'. Particle collisions are modelled through the collision operator Ω.

$$f_i(\vec{x}, t) - f_i(\vec{x} + \vec{c}_i \Delta t, t + \Delta t) = \Omega(f_i, f_i^{eq}) \tag{1}$$

A number of efficient implementations of LBM on GPU have emerged in recent years.[5–9]

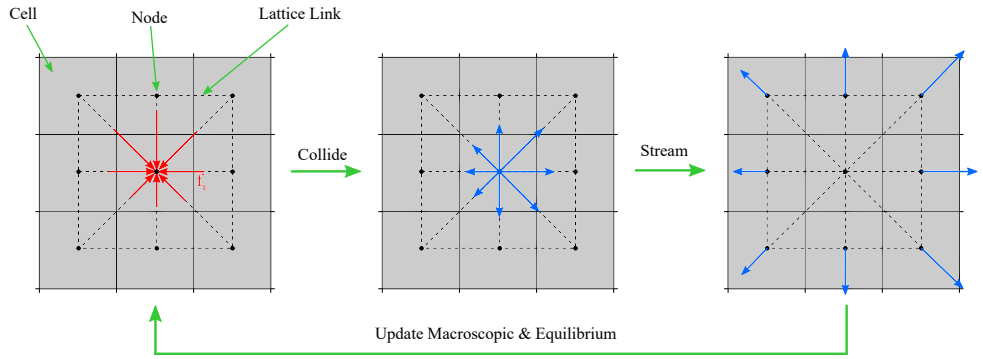


Figure 1: Illustration of the LBM with discrete cells shown as shaded blocks, lattice links as dashed lines the populations f_i as coloured arrows. Populations before collision are red, populations after are blue.

These implementations rely on following established guidelines for programming LBM on GPU[10] to ensure hardware limitations are managed appropriately. Existing interactive GPU-LBM CFD applications[1,11–13] are capable solutions but limit interaction and visualisation to traditional, flat interfaces. These do not convey a truly immersive or intuitive simulation environment for the broader user base. The work presented later in Section 4 incorporates virtual reality and uses its inherent ability for head and controller tracking to provide a more immersive experience for a user.

To maximise performance, our LBM configuration uses the BGK collision operator.[14] Rigid wall boundaries are implemented as simple bounce-back boundary conditions[15] and a Smagorinsky turbulence model[16] is used to provide additional stability. Flow may be introduced using either the forcing scheme of Guo[17] or a forced-equilibrium inlet/outlet boundary.

2.1 Performance

LBM solver performance may be expressed in terms of million lattice updates per second (MLUPS). Our solver demonstrates a peak, 3D, single-precision performance of between 1000 and 1700 MLUPS running on an NVIDIA GTX 1080Ti GPU, depending on the modelling options chosen. It also supports a 1D domain decomposition strategy to split the calculation across multiple GPUs if available. The thickness of the overlap (halo cells) between blocks is selected based on Fig. 2. Even with only 4 GPUs, we are able to simulate over 20M cells in 3D with a throughput amounting to over 3600 MLUPS.

2.1.1 Real-Time Ratio: A Better Metric

The standard LBM is a quasi-incompressible method so non-negligible fluid compressibility introduce errors. In order to control this compressibility error as the resolution is increased, the time step needs to shrink at rate at least that of the grid size. Therefore, increases in resolution require more iterations to be completed to simulate the same physical time. If the throughput of a given GPU is saturated, the 'perceived flow rate' of the simulated flow by a real-time observer will reduce with resolution.

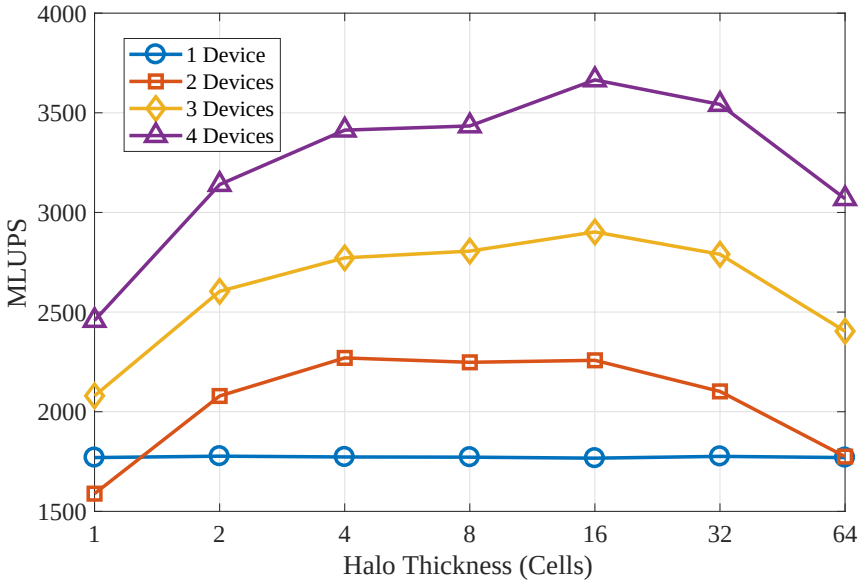


Figure 2: Weak scaling of the LBM solver for different halo thicknesses. Grid size per device was chosen as 193^3 which is sufficient to saturate the GPU memory bandwidth and give the highest possible MLUPS per device.

A more suitable measure of real-time performance is thus the *real-time ratio*[1] which includes the effects of spatial and temporal scaling. This metric is computed as the ratio of wall clock time to simulated time. The simulated time depends on both the lattice throughput, as well as the spatial and temporal discretisation of the simulation. The tests performed in the cited article illustrate that in order to make GPU-LBM scalable for real-time simulation, relaxing the hardware memory-bandwidth limit is only a partial solution and efforts should be concentrated in: relaxing the numerical restrictions of the method; combining LBM with other numerical methods in hybrid simulation approaches; or simply parallelising the simulation across a greater number of GPUs.

2.2 Validation

A validation of the solver is performed by simulating the turbulent channel case of Zecevic *et al.*[18] Direct simulation of an $Re_\tau = 180$ using $H = 46$ is performed giving a $y^+ = 2$ for first lattice site and $y^+ = 4$ for each site thereafter. DNS results of Kim, Moin and Moser (KMM)[19] are used for comparison. Figure 3 shows that the LBM achieves excellent agreement even at modest resolution.

3 TOWARDS AN INTERACTIVE, REAL-TIME CFD ENVIRONMENT

There are many different computing resources which may be used in isolation or combination to enable interactive simulation. The interactive simulation eco-system shown in Fig. 4 illustrates the interoperability of different computing and visualisation devices including mobile devices, virtual reality systems and high performance computing (HPC)

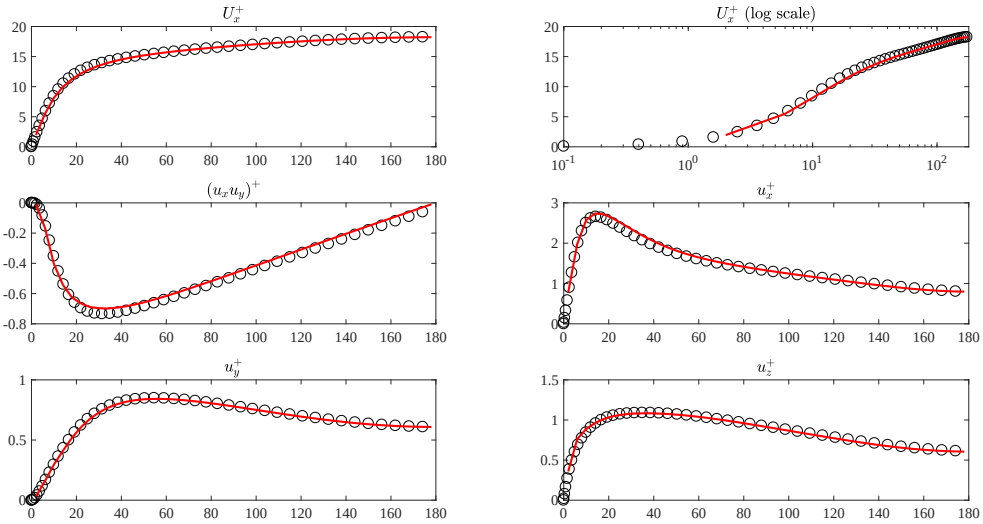


Figure 3: Time-averaged velocity and Reynolds stress profiles measured from the wall to the centre of the channel computed from a 2D spatial average over the domain. KMM (circles), LBM (lines)

systems in delivering interactive simulation.

Geometry can be acquired from depth sensing cameras attached to mobile devices or from computer aided design packages. Simulation may be conducted in one or more of three modes – *local compute*, *local-offload*, *remote off-load*. In the remainder of this section we will discuss some of our implementations of these modes.

3.1 Interactive CFD on Mobile Devices

Recent work[1,13] develops a mobile application framework for implementing GPU-LBM on suitable mobile devices. The authors demonstrate a 2D mobile GPU-LBM implementation that uses the touch screen for run-time interaction as well as the presentation of visual flow information. Readers are referred to these articles for more details.

Our current research is exploring resource sharing models, using clusters of peer-to-peer-connected mobile devices to distribute computation across a local network of mobile GPUs. Thus realising the 'local off-load' part of the eco-system.

3.2 Interactive Steering on HPC

The thesis of Wenisch[20] developed an interactive simulation application using the 'remote off-load' approach. The LBM can be implemented very efficiently on CPU-based HPC systems by exploiting vectorisation. With appropriate optimisation, demanding problems can be simulated at real-time rates.[21] Steering of these compute kernels requires remote on-the-fly visualisation and hence a local user interface (UI) must be connected in an efficient way. Local steering may range from a tablet to virtual reality projection systems (CAVEs).

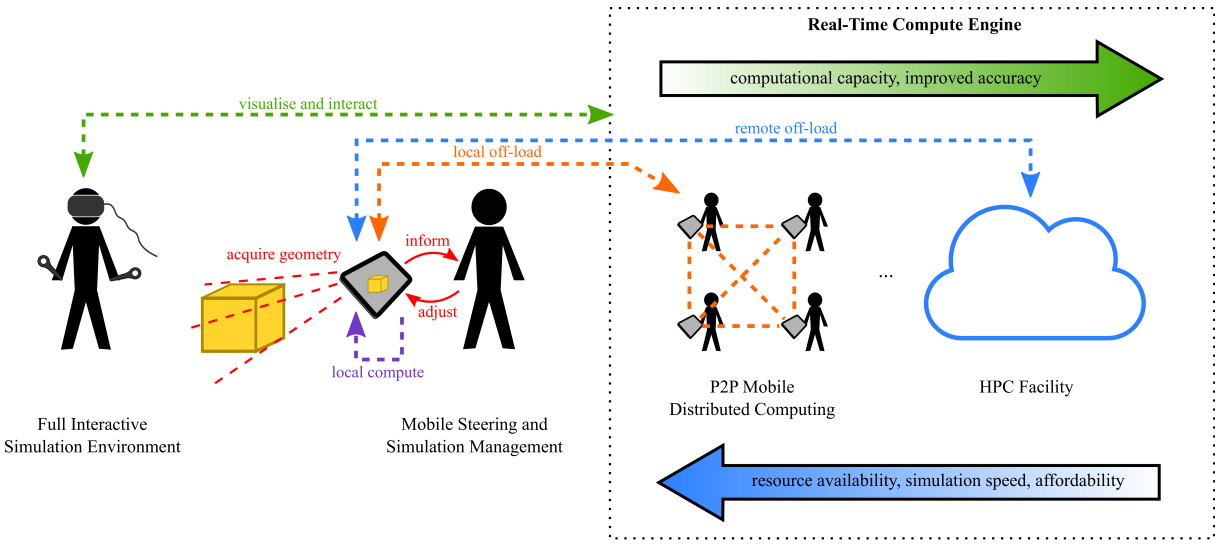


Figure 4: Interactive simulation eco-system using a combination of mobile devices as well as high performance computing systems (adapted from Harwood and Revell[13]).

3.2.1 On-the-fly Visualisation and Steering

To offer smooth, *interactive* data exploration the client must support multi-threading. This facilitates interaction with data representation objects while the underlying data is simultaneously updated by the simulation process. In the case of receiving new external data (usually at irregular intervals dependent on user interaction) the datasets must be updated automatically. Therefore, it is necessary to introduce an interface thread to manage the UI separately. Additionally,this enables the visualisation to connect to other data services, as well as supporting multi-client extensions.

Interaction with the simulation can either be in the form of grid independent modifications or modification of the geometric model. The latter triggers a regeneration of the computational grid. Both types of modifications need to be transferred to the remote computational kernel. Furthermore, the exchange of steering parameters and simulation results takes place 'on-demand' and in real-time. Therefore, the main challenges for the communication process arise from potentially irregular communication patterns on heterogeneous platforms connected by networks with variable bandwidth.

3.2.2 Communication Layout

Details of the communication layout and its data streams is shown in Fig. 5. Modifications pertaining to a user interaction are sent to the simulation engine and are immediately incorporated into the simulation configuration. The HPC system then continues computing based on the updated configuration. As soon as simulation results are available the data are sent to the visualisation client, where the user can observe the adaptation of the fluid.

On the visualisation (VIS) and steering (STEER) side, a background communication

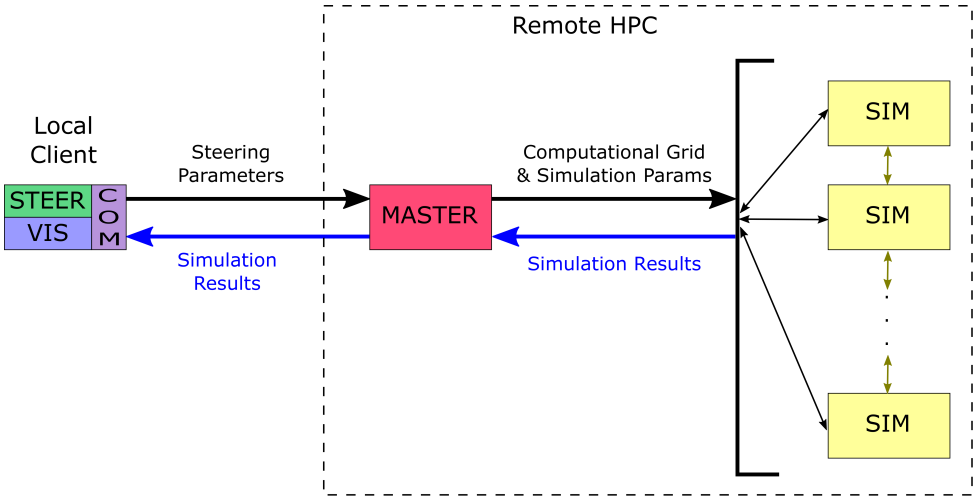


Figure 5: On the visualisation and steering workstation, current flow data received from the running simulation on the HPC System are displayed. The steering UI consists of three threads, one for visualization (VIS), one for user interaction (STEER), and the communication thread (COM). Modifications are sent to the simulation master (MASTER) where they are incorporated into the simulation model immediately and forwarded to the simulation slaves (SIM). By introducing the communication threads (COM and MASTER), data transfer can be overlapped with computation and visualisation for increased efficiency.

thread (COM) monitors for incoming results and sends user modifications. This multi-threading implementation avoids interruptions in steering and post-processing. To keep the data transfer as short and infrequent as possible, only modifications to the set up are forwarded. Therefore, the transmission process is not triggered until after the user has completed all modifications. Since the results are not necessarily sent at regular intervals either, the receipt of data is, in essence, an event-driven process in both directions.

The simulation master (MASTER) can be seen more or less as the communication interface connecting the visualisation and steering UI to the simulation. When the master receives modifications due to user interaction, they are incorporated into the global computational model. Subsequently, the master performs domain decomposition and sends the computational grid (if regenerated) and all further necessary information to the simulation slaves (SIM). The results computed by the slaves are then gathered by the master and sent back to the visualisation and steering client. Data transfer to the visualisation client overlaps the computation of the slaves eliminating communication dependencies between simulation slaves and the steering terminal.

4 VIRTUAL WIND TUNNEL

Visualisation and interaction may also be facilitated using virtual reality (VR), with a local or remote compute engine. We integrated our own GPU-LBM library into a 3D game built using the Unreal Engine 4 (UE4) game engine which supports the HTC Vive VR headset. The LBM solver is written in CUDA C, wrapped in a C++ interface class and compiled as a standalone library. We built a virtual wind tunnel game using the UE4

editor. A custom game actor class represents the simulation domain in the game world, and acts as an interface to the simulation library.

4.1 Game Design

A recreation of a wind tunnel environment is constructed in the UE4 editor using custom meshes designed in CAD. A single VR player is added to the game and additional logic for movement and interaction using the controllers implemented within the editor. Particle-based visual effects are placed upstream in the tunnel to provide smoke streak visualisation of flow velocity.

4.1.1 Solver Interface

It is not practical or necessary to simulate the entire tunnel. Instead, only a limited section of the tunnel is simulated, assuming a uniform upstream flow. A new actor class (`LbmPhysics` actor) is developed and added at the centre of tunnel. The bounding box of the actor coincides with the limits of the region in which the simulation takes place.

In order to have the particle systems describe the simulated velocity field, a `VectorField` actor is attached to our `LbmPhysics` actor. `VectorField` actors are capable of storing a 3D vector field from which particle systems are able to infer their velocity when passing through its region of influence. The contents of the vector field are updated continuously by `LbmPhysics`.

4.1.2 Tunnel Objects

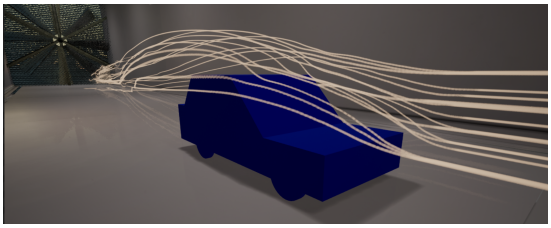
Objects can be added to the tunnel as long as they have a suitable surface mesh representation (i.e. an STL file). This can be obtained directly from CAD or by using a Microsoft Kinect camera to scan an object. This mesh may be imported as an asset through the UE4 editor. Users may then cycle the objects in the tunnel at run-time, selecting any of the available game assets.

The LBM solver does not use the STL mesh directly but pre-computes a voxel-grid-filtered representation of the geometry which is then used to apply the bounce-back boundary conditions within the solver.

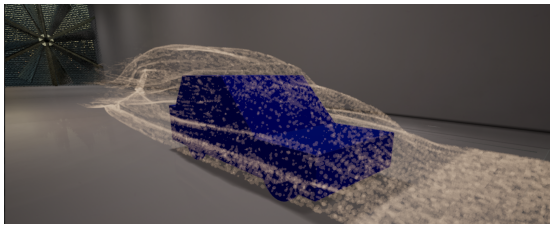
4.1.3 Player Capabilities

An in-game menu is used for player interaction, although some capabilities are mapped to controller buttons. Using the controllers and the menu and run-time, players may:

- cycle through available objects;
- rotate objects in the flow;
- teleport around the wind tunnel (in addition to walking with VR);
- spray smoke using controllers as smoke wands;



(a) Streak Line configuration



(b) Smoke Sheet configuration

Figure 6: Demonstration of the two different tunnel smoke configurations. The smoke sheet may be moved in the vertical plane at run-time.

- switch between two different tunnel smoke representations (Fig. 6);
- adjust the Reynolds number of the flow.

The addition of further player capabilities based on desired use-cases is straightforward to implement using the current framework requiring the addition of a new UI element and adaptation of the game-solver interface class.

5 CONCLUSIONS AND FUTURE WORK

In this paper we have presented an interactive CFD eco-system as well as a brief overview of some recent contributions to advancing real-time CFD simulations. These developments include the application of local mobile devices for local computation and visualisation, the use of remote off-load approaches for interactive simulation using HPC, and the inclusion of an GPU-LBM kernel inside a VR game engine environment.

In order to address inherent limitations of using GPU-LBM, other avenues of research need to be explored including extending calculations to a network of GPUs. This will increase throughput, offsetting the damaging effect of mesh refinement on the real-time ratio.

We have also presented our recent implementation of a real-time simulation platform in the form of a interactive, virtual wind tunnel utilising a virtual reality interface and a local GPU-LBM solver. This game has demonstrated the potential for automotive applications very well but has highlighted a number of challenges with the approach that must be addressed in order to extend its capabilities further.

A tighter integration between simulation and visualisation should be sought. In order to use the `VectorField` actor in UE4, a new vector field must be constructed anew from GPU data introducing a well-known bottleneck in GPGPU computing due to the passing of information between device and host. In the present implementation, this bottleneck restricts the performance of the platform for large-scale problems. This data transfer bottleneck will be removed in a future iteration of the virtual wind tunnel by modifying the particle simulation classes in the game engine source to allow direct swapping of vector field source data resources.

Acknowledgements

This work was supported by Engineering and Physical Science Research Council Impact Accelerator Account (grant number: EP/K503782/1).

REFERENCES

[1] A. R. G. Harwood and A. J. Revell. Interactive flow simulation using Tegra-powered mobile devices. *Advances in Engineering Software*, 115(Supplement C):363 – 373, 2018.

[2] S. Chen and G. D. Doolen. Lattice Boltzmann Method for Fluid Flows. *Annual Review of Fluid Mechanics*, 30(1):329–364, 1998.

[3] M. Mawson. *Interactive Fluid-Structure Interaction with Many-core Accelerators*. PhD thesis, School of Mechanical, Aerospace & Civil Engineering, The University of Manchester, 2013.

[4] N. Delbosc, J. Summers, A. Khan, N. Kapur, and C. Noakes. Optimized implementation of the Lattice Boltzmann Method on a graphics processing unit towards real-time fluid simulation. *Computers & Mathematics with Applications*, 67(2):462 – 475, 2014. Mesoscopic Methods for Engineering and Science (Proceedings of ICMMES-2012, Taipei, Taiwan, 23-27 July 2012).

[5] J. Tölke. Implementation of a lattice boltzmann kernel using the compute unified device architecture developed by nvidia. *Computing and Visualization in Science*, 13(1):29, Jul 2008.

[6] C. Obrecht, F. Kuznik, B. Tourancheau, and J.-J. Roux. Scalable lattice Boltzmann solvers for CUDA GPU clusters. *Parallel Computing*, 39(6):259 – 270, 2013.

[7] M. J. Mawson and A. J. Revell. Memory transfer optimization for a lattice Boltzmann solver on Kepler architecture nVidia GPUs. *Computer Physics Communications*, 185(10):2566–2574, 2014.

[8] Y. Koda and F.-S. Lien. The Lattice Boltzmann Method Implemented on the GPU to Simulate the Turbulent Flow Over a Square Cylinder Confined in a Channel. *Flow, Turbulence and Combustion*, 94(3):495–512, Apr 2015.

[9] N.-P. Tran, M. Lee, and S. Hong. Performance Optimization of 3D Lattice Boltzmann Flow Solver on a GPU. *Scientific Programming*, 2017, 2017.

[10] N. Delbosc. *Real-time simulation of indoor air flow using the lattice Boltzmann method on Graphiscs Processing Units*. PhD thesis, School of Mechanical Engineering, The University of Leeds, 2015.

[11] J. Linxweiler, M. Krafczyk, and J. Tölke. Highly interactive computational steering for coupled 3D flow problems utilizing multiple GPUs. *Computing and Visualization in Science*, 13(7):299–314, 2010.

[12] M. S. Glessmer and C. F. Janßen. Using an Interactive Lattice Boltzmann Solver in Fluid Mechanics Instruction. *Computation*, 5(3), 2017.

[13] A. R. G. Harwood and A. J. Revell. Parallelisation of an interactive lattice-Boltzmann method on an Android-powered mobile device. *Advances in Engineering Software*, 104(1):38–50, 2017.

[14] P. L. Bhatnagar, E. P. Gross, and M. Krook. A Model for Collision Processes in Gases. I. Small Amplitude Processes in Charged and Neutral One-Component Systems. *Phys. Rev.*, 94:511–525, May 1954.

[15] D. P. Ziegler. Boundary conditions for lattice Boltzmann simulations. *Journal of Statistical Physics*, 71(5):1171–1177, 1993.

[16] H. Yu, S. S. Girimaji, and L.-S. Luo. DNS and LES of decaying isotropic turbulence with and without frame rotation using lattice Boltzmann method. *Journal of Computational Physics*, 209(2):599 – 616, 2005.

[17] Z. Guo, C. Zheng, and B. Shi. Discrete lattice effects on the forcing term in the lattice Boltzmann method. *Physical Review E*, 65:046308, Apr 2002.

[18] V. Zecevic, M. P. Kirkpatrick, and S. W. Armfield. The lattice Boltzmann method for turbulent channel flows using graphics processing units. In W. McLean and A. J. Roberts, editors, *Proceedings of the 15th Biennial Computational Techniques and Applications Conference, CTAC-2010*, volume 52 of *ANZIAM J.*, pages C914–C931, 2011.

[19] J. Kim, P. Moin, and R. Moser. Turbulence statistics in fully developed channel flow at low Reynolds number. *Journal of Fluid Mechanics*, 177:133–166, 1987.

[20] P. Wenisch. *Computational Steering of CFD Simulations on Teraflop Supercomputers*. PhD Thesis, Technische Universität München, 2008.

[21] P. Wenisch, O. Wenisch, and E. Rank. Design and Performance Aspects of a CFD Computational Steering Application. In *The Second International Conference on Parallel, Distributed, Grid and Cloud Computing for Engineering*, Ajaccio, Corsica, France, 2011.