GPU-ACCELERATED REAL-TIME VISUALIZATION AND INTERACTION FOR COUPLED FLUID DYNAMICS

F. DE VUYST, C. LABOURDETTE AND C. REY

Abstract. *For real-time applications (dynamic data-driven applications systems like computer-assisted surgery, command and control, etc.), it is necessary to design fast or strongly-accelerated computational approaches. Reduced-order modeling (ROM) is a candidate methodology that summarizes all the parameter-dependent PDE solutions into an easy-to-compute condensed form. ROM usually requires an offline learning process that returns the essential components of the solutions. However, it is known that ROM methodology is not suitable for all problems, especially problems with a large Kolmogorov n-width, like for example dynamical problems involving a continuous multiscale spectrum (like turbulence). In this case, direct simulation is needed and one has to find acceleration strategies. Graphics Processing units (GPU) are a cheap but relevant way to parallelize computations on thousands of cores leading to speedups of order 200 for some algorithms. This paper talks about real-time CFD computations allowing for real time visualization and flow interaction.*

Keywords. Graphics processing unit, real time CFD, interaction.

1. Introduction

Today's computer-assisted Engineering, especially for structural and fluid Mechanics faces great challenges in terms of computing power. Indeed, the exploration of parameter-dependent solutions, sensitivity analyzes w.r.t the entries or uncertainty quantification require hundred of evaluations, and each "PDE" evaluation is already time consuming. For real-time applications like dynamic data-driven application systems, this is more critical because a real-time response is expected. Order reduction methodology is a candidate solution to achieve real-time simulations. It usually requires an offline learning process that returns the essential components of the solutions (by proper orthogonal decomposition [1] or greedy reduced basis construction for example). However, it is known that ROM methodology is not suitable for all problems, for example problems with a large Kolmogorov n-width, like for example dynamical problems involving a continuous multiscale spectrum (like turbulence).

In that case, we need alternative solutions. Parallel computing is a way to accelerate computing, but up to recent years, was reserved to the scientific community or to privileged companies. With the emergence of relatively cheap graphics processing units (GPU) and manycore coprocessors (Intel Xeon Phi), one can now consider highly parallel high-performance computing on a workstation or even a laptop. For example, the new NVidia Tesla K20X (year 2013) has 14 multiprocessors units for a total of 2688 CUDA cores, 6 GB GDDR5 memory and low power consumption of 225 Watt. Tesla K20X also offers 1.31 TFLOPS of double-precision power and nearly 4 TFLOPS of single-precision power (see figure 1).

FIGURE 1. NVidia TESLA K20X coprocessor (KEPLER family) composed of 2688 CUDA cores, offering 1.31 TFLOPS of double-precision power (image by NVIDIA).

In this work, we have evaluated GPU-accelerated fluid dynamics computations. A first conclusion is that GPU performance is strongly dependent on the numerical method in use. Numerical schemes with a low byte-per-flop ratio are much more suitable. Lattice Boltzmann methods have this property and are easily implemented on GPU, allowing for fast solution of the Navier-Stokes equations, even for high Reynolds numbers. Our experiments show that GPU acceleration can achieve real time visualization of fluid dynamics with even real-time fluid interaction. We also give some details on the extension of lattice Boltzmann methods for thermal-fluid coupled problems.

2. Lattice Boltzmann methods for complex flows

Lattice Boltzmann (LB) methods are numerical schemes for simulating viscous compressible flow in the subsonic regime. Instead of solving the continuum hydrodynamic equations, the LB method makes use of a simplified kinetic representation of the fluid and computes the evolution of some discrete distribution functions. The coupled discrete Boltzmann equations are nothing else but transport-reaction equations and the time discretization along the lattice trajectories is particularly simple to derive, thus providing a high level of parallelism. Standard LB methods are LBGK methods based on a BGK collision operator representation, but more complex multiple relaxation time collision operators as proposed in [2] provide more flexibility and improved numerical stability.

As an example, let us consider a thermal-CFD coupled problem modelized by the Navier-Stokes equations and the Boussinesq approximation. The reference fluid mass density at reference temperature T_0 will be denoted by ρ_0, the velocity vector by $\boldsymbol{u}$, the pressure by p, the dynamic viscosity by μ, the constant gravity vector by $\boldsymbol{g} = (0, -g)$, the mean temperature by T, the fluid thermal conductivity by κ, the thermal expansion coefficient by β. We get the coupled problem made of partial differential equations:

$$\nabla \cdot \boldsymbol{u} = 0 \quad \text{in } \Omega \times (0,T], \tag{1}$$

$$\partial_t \boldsymbol{u} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} + \nabla p - \frac{\mu}{\rho_0} \Delta \boldsymbol{u} = \boldsymbol{g}\,(1 - \beta(T - T_0)) \quad \text{in } \Omega \times (0,T], \tag{2}$$

$$\partial_t T + \boldsymbol{u} \cdot \nabla T - \nabla \cdot (\kappa \nabla T) = 0 \quad \text{in } \Omega \times (0,T], \tag{3}$$

added by some convenient initial and boundary conditions. Using either stabilized Finite Element methods, discontinuous Galerkin methods or Finite Volume methods, time discretization is usually written into a semi-implicit form where diffusive terms are treated in a fully implicit manner. As a

consequence, time iterations require the numerical solution of a large sparse linear system. Although linear algebra and sparse data GPU libraries do exist, the performance today is still limited and has to be improved.

On the other hand, LB method are purely explicit methods: their GPU implementation is relatively straightforward with minimal code effort. LB methods are subject to accuracy constraints with a time step $\Delta t = O(h^2)$, h being the characteristic mesh element size. This could let us think that such methods are not suitable for fast time advancing. But, the explicit feature is largely compensated by the high parallel computing decomposition. In the literature, speedups of order 200 at comparable accuracy are often reported.

For the Boussinesq model, a 2D Lattice Boltzmann formulation may be for example:

$$\partial_t f_i + c\boldsymbol{e}_i \cdot \nabla_x f_i = \frac{f_i^{eq} - f_i}{\tau \Delta t} + \frac{\boldsymbol{e}_i \cdot \boldsymbol{g}}{2c}(\delta_{i2} + \delta_{i4})\left(1 - \beta(T - T_0)\right), \quad i = 0, ..., 8, \tag{4}$$

on the standard D2Q9 (dimension 2, 9 velocities) lattice for the Boussinesq Navier-Stokes equations where $\boldsymbol{e}_0 = (0,0)$, $\boldsymbol{e}_1 = (1,0) = -\boldsymbol{e}_3$, $\boldsymbol{e}_2 = (0,1) = -\boldsymbol{e}_4$, $\boldsymbol{e}_5 = (1,1) = -\boldsymbol{e}_7$, $\boldsymbol{e}_6 = (-1,1) = -\boldsymbol{e}8$, and

$$\partial_t k_i + c\boldsymbol{e}'_i \cdot \nabla_x k_i = \frac{k_i^{eq} - k_i}{\tau' \Delta t}, \quad i = 1, ..., 4 \tag{5}$$

on a D2Q4 lattice for the convection-diffusion thermal equation, with $\boldsymbol{e}'_1 = (1,0)$, $\boldsymbol{e}'_2 = (0,1)$, $\boldsymbol{e}'_3 = -\boldsymbol{e}'_1$ and $\boldsymbol{e}'_4 = -\boldsymbol{e}'_2$. The numerical "speed of sound" c is the speed of propagation of the information on the lattice for a given constant time step Δt: $c = h/\Delta t$. The so-called lattice Knudsen numbers τ and τ', $\tau, \tau' > 0$, $\tau, \tau' = O(1)$ are calibrated in order to be consistent with the macroscopic equations. This will be discussed a little bit later.

Macroscopic quantities are then retrieved according to the first discrete moments of the distributions: the temperature is simply given by

$$T = \sum_{i=1}^{4} k_i. \tag{6}$$

For the Navier-Stokes equation, we decided to use the so-called ID2Q9 model [3] referred to as the incompressible D2Q9 LB method, which as been built to be less sensitive to the compressibility effects than the standard D2Q9 LBKG model. Macroscopic quantities are retrieved according to the formulas

$$\boldsymbol{u} = \sum_{i=0}^{8} c\boldsymbol{e}_i \, f_i, \quad p = \frac{c^2}{4\sigma}\left[\sum_{i=1}^{8} f_i - \frac{2}{3}\frac{|\boldsymbol{u}|^2}{c^2}\right]. \tag{7}$$

Equilibrium distributions are computed from the macroscopic quantities, i.e. $f_i^{eq} = f_i^{eq}(\boldsymbol{u}, p)$ and $k_i^{eq} = k_i^{eq}(T)$. For the temperature distribution, the equilibrium expression is very simple:

$$k_i^{eq} = \frac{T}{4}\left(1 + 2\frac{\boldsymbol{u} \cdot \boldsymbol{e}'_i}{c}\right).$$

For the distributions f_i, the ID2Q9 model [3] provides the expressions:

$$f_i^{eq} = \xi_i \frac{p}{c^2} + \omega_i \left[3\frac{\boldsymbol{e}_i \cdot \boldsymbol{u}}{c} + \frac{9}{2}\frac{(\boldsymbol{e}_i \cdot \boldsymbol{u})^2}{c^2} - \frac{3}{2}\frac{|\boldsymbol{u}|^2}{c^2}\right] \tag{8}$$

where $\xi_0 = -4\sigma$, $\xi_{1,2,3,4} = \lambda$ and $\xi_{5,6,7,8} = \gamma$, with some weights $\omega_0 = \frac{4}{9}$, $\omega_{1,2,3,4} = \frac{1}{9}$ and $\omega_{5,6,7,8} = \frac{1}{36}$. To be consistent with the incompressible Navier-Stokes equations, one asks $\lambda + \gamma = \sigma$ and $\lambda + 2\gamma = 1/2$. In [3], the following admissible values are suggested: $\sigma = 5/12$, $\lambda = 1/3$ and $\gamma = 1/12$.

Finally, the coupling terms

$$s_i := \frac{\boldsymbol{e}_i \cdot \boldsymbol{g}}{2c}(\delta_{i2} + \delta_{i4})\,(1 - \beta(T - T_0))$$

at the r.h.s. of the LB equation (4) allow us to retrieve the macroscopic buoyancy term since

$$\sum_{i=0,8} c\boldsymbol{e}_i\, s_i = \boldsymbol{g}\,(1 - \beta(T - T_0))\,.$$

2.1. **Chapman-Enskog multiscale analysis, consistency analysis.** From a multiscale Chapman-Enskog expansion w.r.t. the relaxation times $\varepsilon = \tau\Delta t$ and $\varepsilon' = \tau'\Delta t$, it can be shown that the kinetic lattice Boltzmann equations are consistent with the Boussinesq Navier-Stokes equations as soon as the two compatibility identities hold

$$\tau\frac{h^2}{\Delta t} = \frac{\mu}{\rho_0}, \quad \tau'\frac{h^2}{\Delta t} = \kappa. \tag{9}$$

So there is a direct link between diffusivity and the relaxation times toward local statistical equilibrium. For $\tau, \tau' = O(1)$, we find out the consistency constraint on the time step $\Delta t = O(h^2)$.

In can also be proved that the order of accuracy of the LB approximation is $O(M\Delta t)$, where $M = |\boldsymbol{u}|/c$ is the numerical Mach number. The speed of sound can always be adapted to the maximum flow velocity so that M is rather small. If needed, one can choose $M = O(\Delta t)$ in such a way that the LB method is second-order accurate.

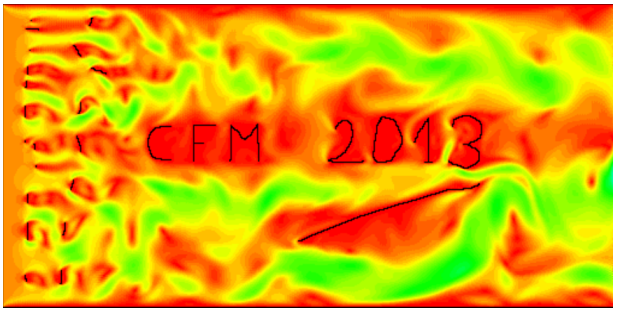
Then the Lattice Boltzmann equations (LBE) are fully discretized on the lattice using either the fully explicit Euler scheme, or using the trapezoid rule along streamlines. This last option directly leads to a second order discretization but is implicit. The explicit feature can be easily recovered by a classical change of variable (see [4]).

2.2. **Implementation on GPU.** A Lattice Boltzmann method is particularly suitable for parallel GPU computing for many reasons: first, the method is explicit, thus it does not require any solution of large linear systems. Second, the time advance scheme on the lattice Boltzmann equation can be seen as a two-step method "stream-and-collide" involving a point-wise collision process which is fully parallelizable, and a streaming process that propagates information to the first lattice neighbors. There is also optimized code data structure and programming methods on GPU for streaming. Third, the spatial propagation pattern – the lattice – is uniform, what is very important for efficient concurrent memory access and high performance. Today, one can find numerous papers about the optimal GPU implementation of Lattice Boltzmann methods, see for example C. Obrecht and co-authors [5] for a recent reference.

2.3. **Visualization, interaction.** GPU allows not only for high-performance computing, but also for high-performance real-time visualization. GPU programming languages, like NVIDIA CUDA (Compute Unified Device Architecture) allow to bind some data arrays with visual objects like pixel

buffer objects. Then the OpenGL graphic language is used to render images OpenGL also allow for mouse event handling (move, mouse button clicked and released, etc). It is used to add obstacles into the flow. Adding obstacles means that some computational points change of state and become wall boundary conditions. An embedded boundary condition is then used to handle both fluid points and wall boundary conditions. For the present time, wall boundaries are treated as step-like broken lines, but there is no theoretical difficulty to improve embedded boundary conditions and achieve second-order accuracy, by embedding CAD objects for example.

Another technological achievement is the use of an infra-red tracking device connected to a PC or a laptop allowing for real time interaction on any display device. A U-pointer converts any projected image into an interactive workstation and captures U-pointer pen's movement on display and analyzes the movement. Thus this enables live simulation demonstrations into classrooms or during scientific meetings or conferences (see figure 2).

a) 

b) 

FIGURE 2. a) Visualization and interaction. Adding some wall obstacles on-the-fly into the live simulation with the mouse. b) U-pointer PC infra-red tracking device. This tool converts any projected image into an interactive workstation and captures the U-pointer pen's movement on display and analyzes the movement. Thus it enables live simulation demonstrations into classrooms or during scientific meetings or conferences.

2.4. **Performance assessment.** It is always a difficult task to compare performance of GPU-accelerated codes to standard CPU-codes because there is neither absolute measure of performance nor standard protocols ([6]). Moreover, a CPU-based code can be optimized for CPU using compiler optimization options (Intel SSE optimization for example, etc.). Finally, today both CPU and GPU are evolving very fast, so even an absolute performance comparison measure would become obsolete rapidly. In our opinion, the implementation effort should also be taken into account into the performance assessment.

Roughly speaking, what can be said is that GPU accelerates computations of lattice Boltzmann methods by a speedup factor of about 200 on a standard HPC GPGPU (Nvidia Tesla boards for example) compared to a mono-core computation on a standard CPU (like Intel Xeon). Of course, one can use SMP parallel computing on a CPU with openMP for example. But anyway GPU still notably accelerates computations compared to a (say) 6-core CPU processor.

It has been observed that GPU-acceleration allows for live runtime evolution of flow dynamics, including high-Reynolds number Navier-Stokes equations, on grid sizes of typical order 1024×512. This has been presented at different recent schools (MFN2013), conferences (SMAI 2013) and international Conferences (Trends in multiphase flows modeling 2012, THESIS 2013) arousing surprises, questioning and interest w.r.t. such cheap computational power.

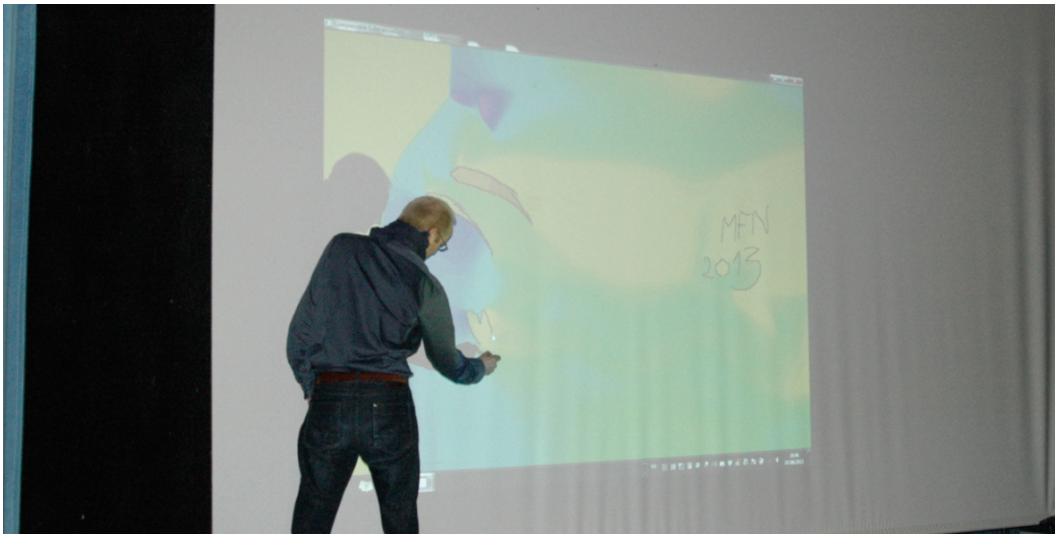


FIGURE 3. Live GPU-accelerated CFD demo at the Spring CFD School MFN 2013 (organized by LIMSI, U. Orsay). Here, an attendee is welcome to draw some obstacles on-the-fly on the board.

3. Ongoing works

We definitely believe that high-performance GPU computing will open a new era for scientific computing and computer engineering, with new ways of use and new models of collaborative works. In the academic context, GPU enables live demonstrations of real-time simulations into classrooms, for a better understanding of physical phenomenons, dynamics of fluid Mechanics, etc. From the Engineering point of view, direct human interaction into live simulation probably will allow for collaborative design, human-in-the-loop optimization, parameter space exploration, etc.

Current developments in our team are going into this direction: for example for thermal-flow computing of heat convection, we plan to setup a software workshop of optimal design with online multicriteria evaluations (pressure loss vs Nusselt number), plot into the output objective space, Pareto surface, etc. We also intend to use use such a real-time software environment into a large displaywall showroom for exploring new collaborative engineering processes.

4. Conclusions

GPU parallel computing is yet another mean to accelerate complex coupled fluid dynamics problems. We have developed a GPU hardware and software environment able to render fluid dynamics simulation on two-dimensional grids of order 1024^2, with possibly direct flow interaction. The use

of a pointer tracking device allows us to achieve live CFD demonstrations into classrooms or during meetings and conferences. Today's GPU are still limited in memory capacity (up to 6 GB DRAM) and bandwidth (about 250 GB/s), but the next NVIDIA Maxwell and Volta generations (scheduled for 2014 and 2016 respectively) will allow unified virtual memory with direct access memory, and the Volta GPU is expected to gain access to up to 1 TB/s memory bandwidth, probably making real 3D high performance computing possible.

Acknowledgments

This work is part of the scientific Program ENSC *CUDA Research Center* supported by NVIDIA. It is also partly supported by the Farman Institute for complex systems, ENS Cachan, by FMJH, Labex Fondation de Mathématique Hadamard, Université Paris-Saclay and DIGISCOPE Datawall Equipment consortium.

References

[1] B. Haasdonk, Convergence rates of the POD-Greedy method, 2013, ESAIM: Mathematical Modelling and Numerical Analysis Volume 47, 3:859-873.

[2] D. d'Humières, I. Ginzburg, M. Krafczyk, P. Lallemand and L.S. Luo, Multiple relaxation-time lattice Boltzmann models in three-dimensions, 2002, Philo. Trans. R. Soc. Lond. A, 360:437-451.

[3] Z. Gho, B. Shi, C. Zheng, A coupled lattice BGK for the Boussinesq equations, 2002, Int. J. Numer. Meth. Fluids, 39:325–342.

[4] I. V. Karlin, S. Ansumali, C. E. Frouzakis, and S. S. Chikatamarla, Elements of the Lattice Boltzmann Method I: Linear Advection Equation, 2006, Communications in Computational Physics, 1:616-655.

[5] C. Obrecht, F. Kuznik, B. Tourancheau and J.-J. Roux, The TheLMA project : a thermal lattice Boltzmann solver for the GPU, 2012, Computers and Fluids, 54:118-126.

[6] J.-M. Etancelin, G.H. Cottet, C. Picard, F. Pérignon, F. De Vuyst, C. Labourdette, Is GPU the Future of Scientific Computing ?, 2013, Annales Mathématiques Blaise Pascal, Actes CANUM 2012, Mini-symposium "Is GPU the future of Scientific Computing ?" (org. L. Gouarin, A. Hérault, V. Louvet), online paper by *Centre de diffusion des revues académiques de Mathématiques*.

[6] F. De Vuyst and F. Salvarani, GPU-accelerated numerical simulations of the Knudsen gas on time-dependent domains, 2013, Comput. Phys. Comm., 184, 3:532-536.

F.D.V, C.L.: Centre de Mathématiques et leurs Applications, CMLA UMR 8536, ENS Cachan, 61, avenue du Président Wilson, 94235 Cachan cedex France

E-mail address: devuyst,labour@cmla.ens-cachan.fr

C.R.: Laboratoire de Mécanique et Technologie, LMT UMR 8535, ENS Cachan, 61, avenue du Président Wilson, 94235 Cachan cedex France

E-mail address: rey@lmt.ens-cachan.fr