Efficient Aeroacoustic Simulations of Jet Engine Noise using GPU acceleration

Benjamin de Brye[1], Markus Brandstetter[1], Eloi Gaudry[1],Yves Detandt[1], Aurélien Mosson[2]
[1] *Free Field Technologies, Mont-Saint-Guibert, Belgium, E-Mail:benjamin.debrye@fft.be*
[2] *Airbus France SAS, Toulouse, France*

Introduction

The computational cost of aeroacoustic simulations is driven by considerations on the highest frequency to be addressed, the size of the physical domain, the complexity of the mean flow supporting the acoustic propagation and the complexity of the noise source which increases with the frequency. Realistic jet engine simulations involve all these difficulties and therefore lead to a very challenging model in terms of size for the design and optimization of rear fan noise. To reduce the computational time, modern solvers are considering both optimized computational sequences through parallel solutions and optimization with respect to the hardware architecture. In this paper, a Discontinuous Galerkin method is selected to solve the linearized Euler equations in time-domain, which provides an element-wise storage well suited to efficient parallel solutions. Also, an emerging technique is to use the Graphical Processing Units (GPU) for accelerating computational operations. The paper presents these different optimizations to enhance the efficiency of the aeroacoustic simulations. Timings and computational requirements are presented in the paper as obtained by the commercial software Actran DGM. Interesting perspectives are given for larger models (higher frequencies, larger domains) as well as for the numerical design of new jet engine exhaust systems.

The Discontinuous Galerkin Method for the Linear Euler Equations

The discontinuous Galerkin method (DGM) combines the highly parallelisable aspects of the finite volume schemes to the accuracy of the finite element techniques. Degrees of freedom (dofs) are stored at the element level, meaning that dofs are duplicated at the element interfaces (Figure 1). This discontinuity allows for an efficient parallel handling.

The time domain propagation operator is solved using a 4th order explicit Runge-Kutta that leads to a block wise system easy to inverse. The spatial operators are computed using a quadrature free implementation that allows for a fast solving with less memory consumption.

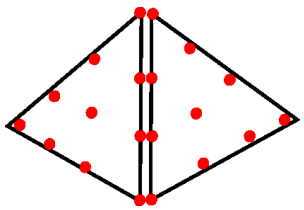

Figure 1: Sketch of location of degrees of freedom. Dofs are duplicated at the element interface, making the method discontinuous.

The Linear Euler Equations (LEE) are solved to deliver accurate predictions in aeroacoustic modelling of complex systems by means of Discontinuous Galerkin Method [1,2,3,4,5].

Model Description

The model is an isolated jet engine exhaust (nacelle only) with static sideline conditions. The mean flow is computed with an external CFD solver as illustrated in Figure 2.

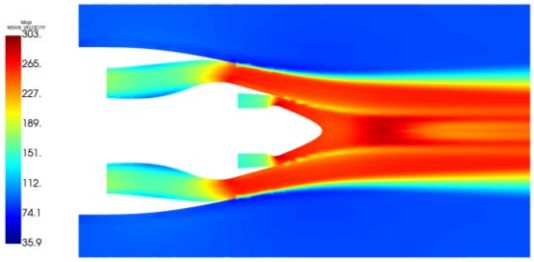

Figure 2: Axial mean velocity norm in 3h-9h cross-section

The acoustic waves are emitted from the fan surface as a set of duct modes propagating in the secondary exhaust. The acoustic solution propagates in the near field and a buffer region is used to damp the outgoing waves. An FWH solver computes the far field solution based on the acoustic fields provided at the interface between the physcial and the buffer region. The simulation is run for a single frequency : 1147Hz. The order of the elements is automatically computed in order to enforce a satisfying dispersive and dissipative error over the whole mesh (a priori error estimation). The mesh is composed of 440k elements, leading to 132Mdofs.

The Figure 3 represents the acoustic pressure level in dB. The reader is referred to [5] for a complete description of the exhaust model.

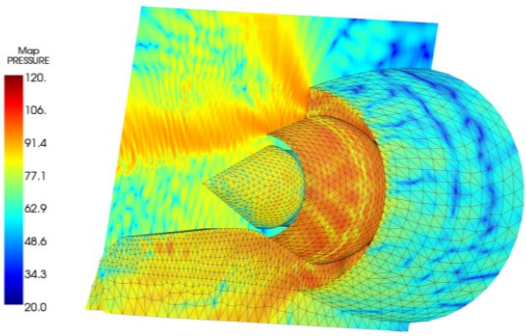

Figure 3: Map of acoustic pressure [dB] at 1174 Hz of the isolated exhaust computed with Actran DGM accelerated with 2 Tesla K80.

Implementation

The GPU acceleration inside Actran DGM has been implemented in the CUDA framework (only for NVidia accelerators). Many functions had to be re-factored for fitting with this framework. The GPU multithreading is done at the element and element-node level for volume and surface integral contributions. Intense use of the cuBLAS [6] libraries is made in order to accelerate the matrix-matrix multiplications. Data structures have been restructured in order to improve the memory access efficiency. Finally, multiple GPUs acceleration is available through domain parallelism (distributed memory parallelism).

Performances

This section presents some performance indicators concerning the GPU acceleration of the exhaust model presented previously. Performance indicators are not absolute since they may depend on the reference machine. So different performance indicators concerning different machines are presented.

The machine boarding the GPU accelerators is a 6 cores Haswell Intel ® Xeon ® E5-1650 v3@3.50GHz (named HASWELL-GPU). Both NVidia Tesla K40 and K80 cards have been tested on this machine. The K80 card has the particularity to contain two GPUs. A second machine is used for comparing the timings. This is a cluster of 24 nodes Ivybridge Intel ® E3 with each 4 cores connected with InfiniBand (named IVB4).

The speedups are all based on a mean time per iteration. Another important indicator of performance is the efficiency which is defined as $E=T_1/(nT_n)$, where T_1 indicates the time per iteration for the sequential run, n is the number of processes and T_n denotes the time per iteration for the n processes simulation. The closest to 1, the better is the scalability of the application.

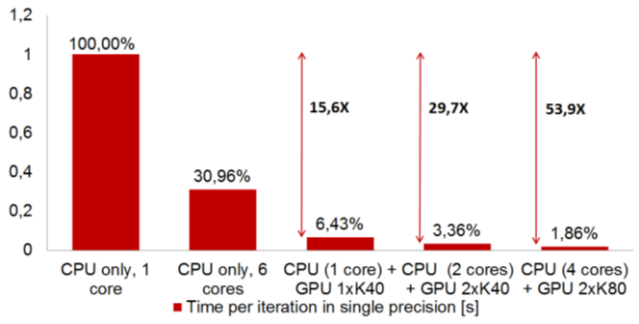

Figure 4: Relative time for iteration for a single precision simulation on a 6 cores cores Haswell node (Intel® Xeon® CPU E5-1650 v3 @ 3.50 GHz) equipped with 2 Tesla K40 or 2 Tesla K80. Speedup ratio compares the sequential (1 core) and the parallel (6 cores) full CPU simulation with the 1,2 or 4 GPUs.

The first results compare full CPU sequential and parallel computations with CPU+GPU(s) computations using the same (6 cores) node (HASWELL-GPU). Figure 4 presents the mean time per time step (from left to right) for a full sequential CPU simulation, a full CPU 6 processes parallel simulation, a sequential CPU simulation accelerated with 1 K40, a 2 processes parallel simulation with one GPU per process (2 K40) and finally a 4 processes simulation with one GPU per process (2 K80, 4 GPUs).

With single precision, one can observe that the CPU reference sequential simulation takes 23s per time step (defined as a reference time, i.e. 100%). When using the 6 cores of HASWELL-GPU, the time per iteration decreases to 31%. The corresponding efficiency reaches 48%; this low value can be explained by the concurrency of the memory access inside the node. When using one GPU, the sequential simulation time per iteration falls to 6.4%, leading to a speedup of 15.6 compared to the sequential simulation and 4.8 compared to the 6 processes simulation. When doubling the number of GPUs (and processes), i.e. 2 processes and 2 GPUs, the speedup reaches 29.7 and 9.2. Using 2K80 (4GPUs), the speedup comes to 53.9. The performance for the double precision simulations (not shown here) are of the same order.

The second performance analysis will compare the HASWELL-GPU machine (6 cores, 2K40) with the IVB4 cluster. Figure 5 presents the time per iteration versus the number of MPI processes for the single precision (red) and double precision simulations (blue). The continuous lines with the crosses represent the time per iteration obtained with IVB4 while the rounds and triangles indicate the time per iteration for the HASWELL-GPU machine using respectively 1 and 2 K40 GPU(s). It can be seen that the single precision run with 1 GPU is equivalent to the parallel run using 31 processes. The parallel run with 2 GPUs is as fast as a 62 processes simulation on IVB4 (for single and double precision). The 1GPU double precision results are not presented since the amount of RAM was not sufficient on the GPU. The efficiency here is over 90% due to the small number of cores and their interconnection with InfiniBand.

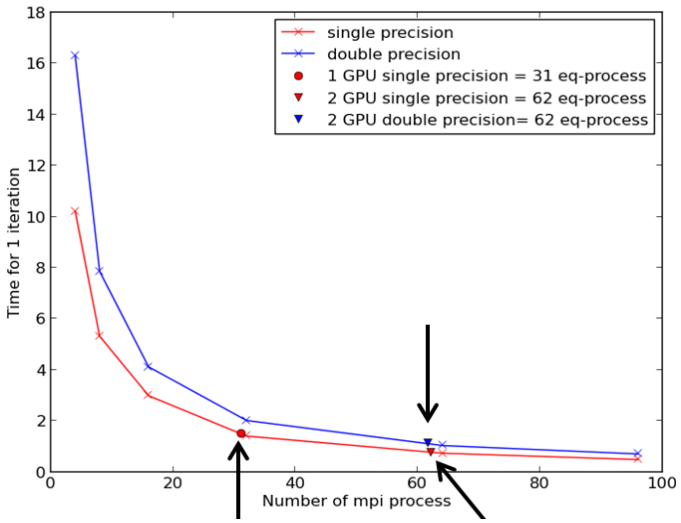

Figure 5: Comparison of the CPU+GPU iteration time with a CPU cluster. Iteration time vs number of MPI processes

The last analysis was realised using the PSG cluster provided by NVidia for benchmarking applications. We used 2 nodes with each 20 Ivybridge Intel ® cores and 6 Tesla K40. The time per iteration in single precision reaches 35s

for the CPU sequential reference simulation. Using up to 12 GPUs (so 2 nodes with each 6 cores and 6 GPUs) we reach a speedup of 215. Comparing on the one hand a full CPU simulation using the 20 cores of the node (efficiency of 70%) and on the other hand a sequential CPU+GPU simulation, the speedup is 1.7. Figure 6 presents the speedup as a function of the number of GPUs, leading to a satisfying efficiency of 76% for the 12 GPUs simulation.

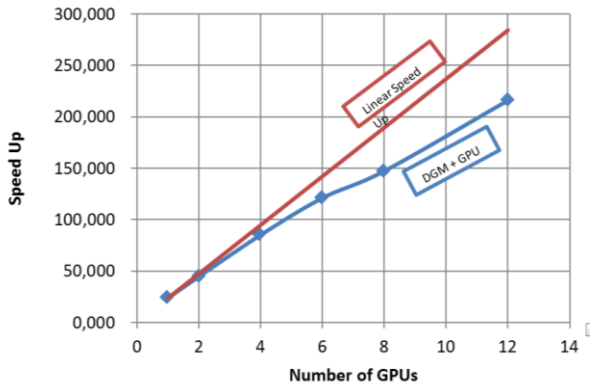

Figure 6: Comparison of scaling of Actran DGM+GPU with the perfect linear scaling. Actran DGM efficiency varies between 93% (2 GPUs) and 76% (12 GPUs).

Perspective for larger models

This jet engine exhaust case has 132 Mdofs. The ratio between the model length and the wavelength ($ka = L/\lambda$) reaches 16 and the model volume amounts to 2 m³. This paper shows that it is possible to run this model using single precision on one single K40 (12GB), but not in double precision due to GPU memory limits. Actually, using 12 GB it is possible to run up to 125Mdofs in double precision and 250 Mdofs in single precision. So using 2K80, possible 1Gdofs simulation in single precision is possible.

With 2 Tesla K80, ActranDGM has been used to solve successfully :

- A truck exhaust 5 kHz radiation totalizing 220 Mdofs, 36 m³ and ka = 40. Simulation time: 2h23' (Figure 7);
- A car inner cavity with/without acoustic treatment at 15 kHz totalizing 383 Mdofs, 2.95m³ and ka = 80. Simulation time: 4h00';
- A car parking ultra sound sensor (50 kHz) for 984 Mdofs, 0.1 m³ and ka = 135. Simulation time: 14h40'.

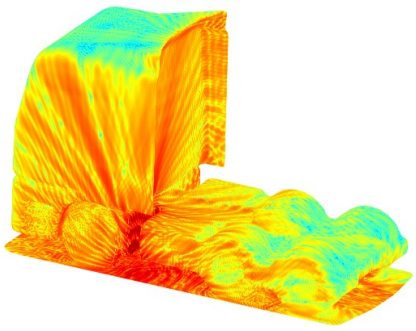

Figure 7: Truck exhaust radiation pressure map at 5kHz.

Conclusions

This paper presents the acceleration of the Actran DGM solver using GPUs. Since Discontinuous Galerkin Methods are suited for GPU acceleration, a speedup between 15 and 20 is achievable depending on the hardware in single precision and between 28 and 40 for a Tesla K80. Indeed GPU acceleration allows (for the same performance) to restrict the simulation to a single node equipped with several Tesla cards instead of several interconnected nodes counting hundreds of cores that is more expensive and require more maintenance.

Currently, simulations with 1Gdofs size are feasible (car parking sensor at 50kHz with less than 15 hours of simulation using only 2 Tesla K80). So even if the amount of RAM for the GPUs is restricted, very large industrial models can be tackled using only a few GPUs.

Acknowledgments

This study has been realised thanks to the support of NVIDIA providing us the access to their benchmarking infrastructure (PSG cluster).

Literature

[1] Free Field Technologies, Actran 16 User's guide: Actran DGM, 2015.

[2] Leneveu, R., Schiltz, B., Manera, J. and Caro, S., 2007. Parallel DGM scheme for LEE applied to exhaust and bypass problems. *AIAA Paper*, *3510*, p.2007.

[3] Angeloski, A., Discacciati, M., Legendre, C., Lielens, G. and Huerta, A., 2014. Challenges for time and frequency domain aeroacoustic solvers. In *11th World Congress on Computational Mechanics (WCCM XI); 5th European Conference on Computational Mechanics (ECCM V); 6th European Conference on Computational Fluid Dynamics (ECFD VI): July 20-25, 2014, Barcelona, Spain* (pp. 1-12).

[4] Rarata, Z., 2014. *Application and assessment of time-domain DGM for intake acoustics using 3D linearized Euler equations* (Doctoral dissertation, University of Southampton).

[5] A. Mosson, D. Binet and J. Caprile, Simulation of Installation Effects of Aircraft Engine Rear Fan Noise with ACTRAN/DGM, *20th AIAA/CEAS Aeroacoustics Conference*, Atlanta, 2014

[6] NVIDIA CUDA Basic Linear Algebra Subroutines (cuBLAS) URL: https://developer.nvidia.com/cublas