DTU Informatics
Department of Informatics and Mathematical Modeling

Development of Desktop Computing Applications and Engineering Tools on GPUs

Hans Henrik B. Sørensen†, Stefan Glimberg, Toke Jansen Hansen, Jeppe Frisvad, Allan P. Engsig-Karup {†: hhs@imm.dtu.dk}

Indtroduction and Background

GPULab - A competence center and laboratory for research and collaboration within academia and partners in industry has been established in 2008 at section for Scientific Computing, DTU informatics, Technical University of Denmark. In GPULab we focus on the utilization of Graphics Processing Units (GPUs) for high-performance computing applications and software tools in science and engineering, inverse problems, visualization, imaging, dynamic optimization. The goals are to contribute to the development of new state-of-the-art mathematical models and algorithms for maximum throughout performance, improved performance profiling tools and assimilation of results to academic and industrial partners in our network. Our approaches calls for multi-disciplinary skills and understanding of hardware, software development, profiling tools and tuning techniques, analytical methods for analysis and development of new approaches, together with expert knowledge in specific application areas within science and engineering. We anticipate that our research in a near future will bring new algorithms and insight in engineering and science applications targeting practical engineering problems.

Fast Simulation of Unsteady Nonlinear Water Waves

For analysis and prediction of unsteady dispersive nonlinear water waves over uneven bottoms from shallow to deep water and for wave-structure interactions in ocean and offshore engineering it is important to have fast simulations tools. In a current project, we focus on designing new or improved algorithms that are massively parallel and can achieve a high effective arithmetic throughput in simulations based on state-of-the-art algorithms in computational fluid dynamics problems for ocean and offshore engineering. This will enable opportunity for accurate and fast analysis and prediction of flow evolution and kinematics, e.g. in large areas, over long times and for doing fast parameter studies based on the efficient solution of many problems.

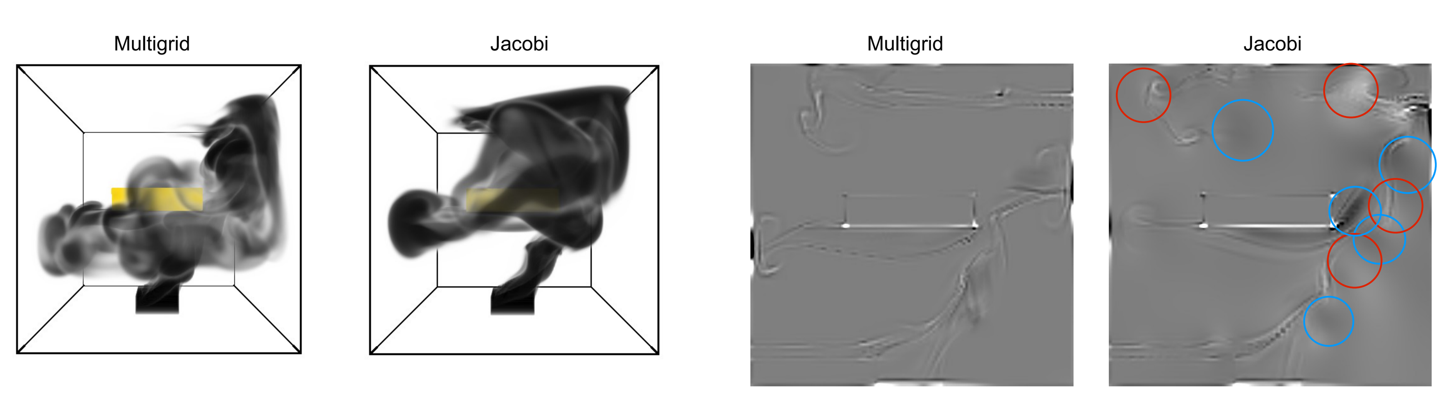

Fig. 2: Simulation comparison of the multigrid and the Jacobi methods. The number of iterations was chosen so that the total time consumption are the same for both methods. Left: Details along with the rotational motions, are much more apparent with the multigrid method. Right: The divergence field at the center slice. Medium gray equals zero divergence. Only the Jacobi method suffers from low-frequency errors.

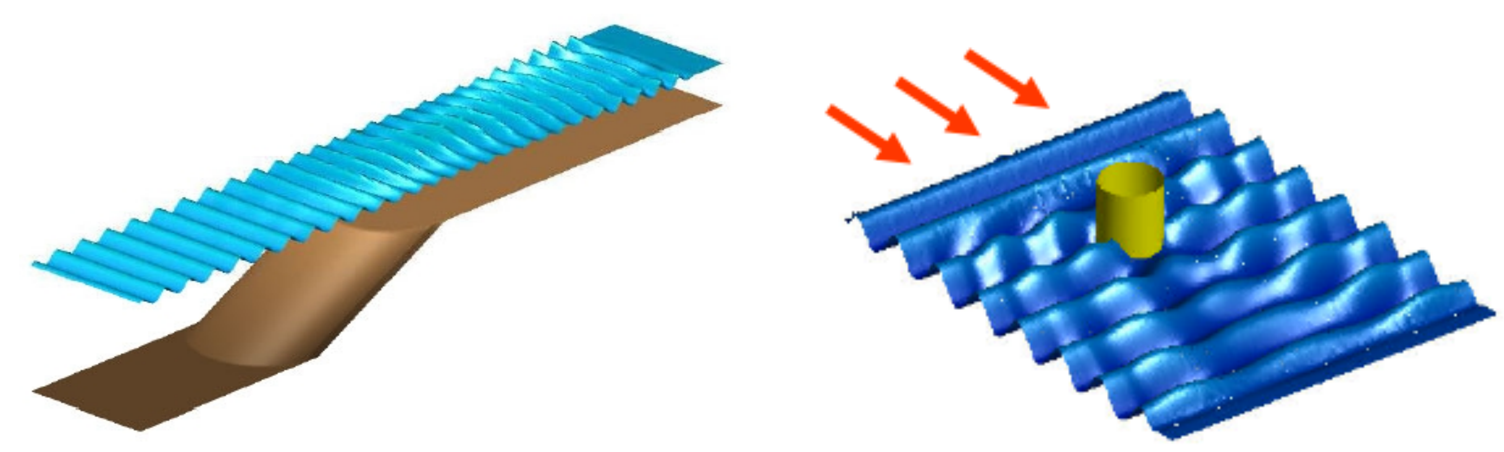

Fig. 1: Snapshot of free surface for an instant of time. Left: Wave focussing due to depth refraction. Right: Wave diffraction caused by wave-structure interaction.

Smoke Simulation in Fire Engineering using GPUs

A Computational Fluid Dynamics (CFD) solver, for smoke propagation, using the parallel architecture of graphics hardware has been developed. The purpose is to investigate the possibilities of fast approximation techniques in combination with the powers of GPUs, and test usability from an engineering point of view. CFD tools for smoke propagation help engineers analyze security risks at fire scenes. Based on several case studies, our solver has shown to be an efficient tool for interactive smoke simulation. Even at high resolutions the solver performs at interactive rates on a single GPU. The parallel architecture of the GPU makes it an excellent computational unit for compute intensive tasks like CFD.

Kernel based searchlight Heuristic for realtime fMRI

A novel searchlight heuristic that yields similar results to resampling based searchlight approaches has been developed. The reduced computational complexity of the suggested heuristic enables searchlight approaches to be applied in a real-time setting preventing the need for functional localizer scans. The suggested heuristic also enables the use of dynamic searchlight procedures capable of adapting to changes in the subjects strategy, performance or brain state during the experiment. Finally, the absence of data dependencies between distinct searchlight regions and the low memory footprint, makes the heuristic highly suitable for modern multi core architectures.

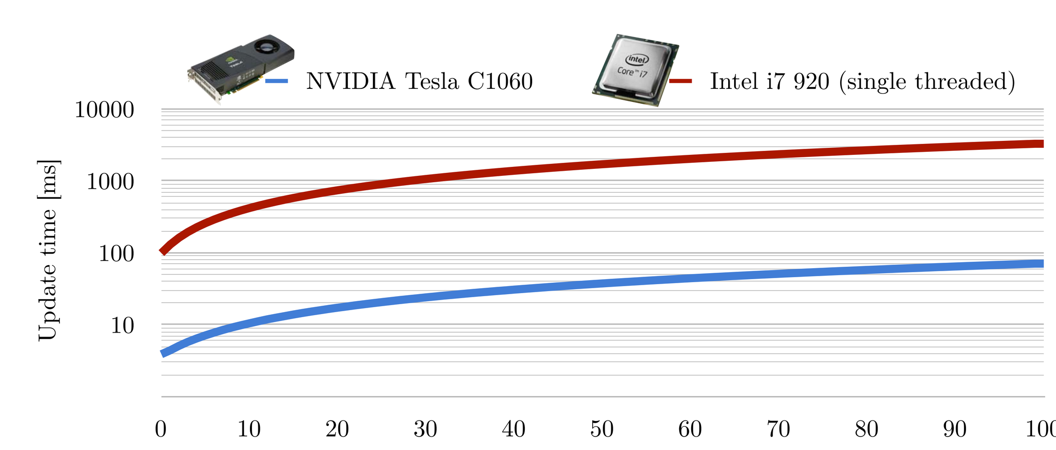

Fig. 3: Compares the performance of the heuristic on respectively a GPU and CPU. The single threaded CPU implementation can handle a time window of ≈ 50 samples, before the update reaches the bound defined by the repetition time, whereas the GPU is ≈ 45 times faster.

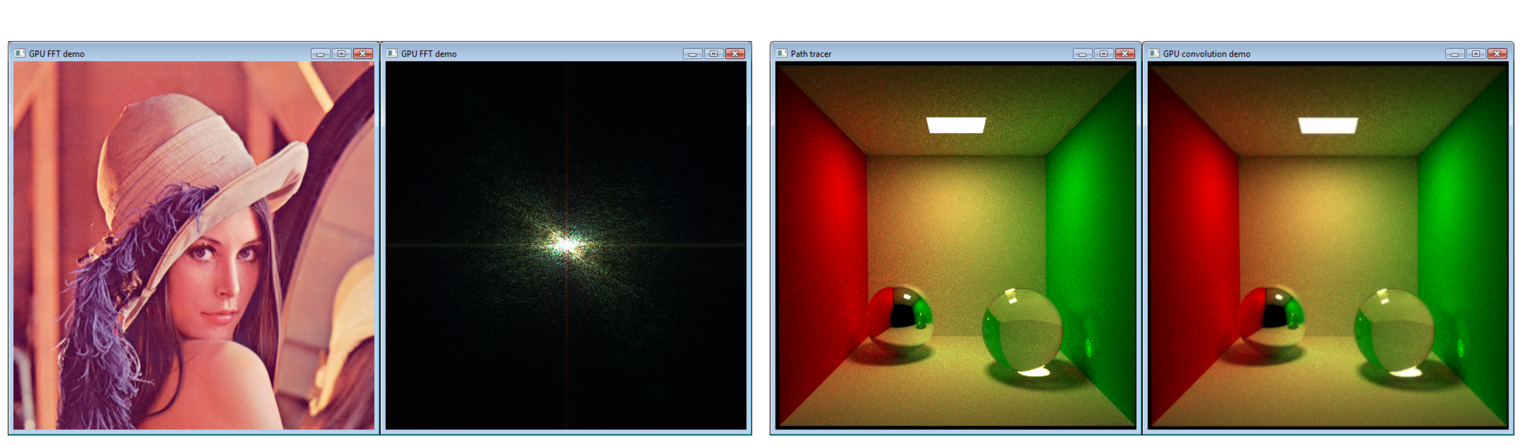

Fig. 4: Left: we compute the 2D FFT of an 512x512 RGB image in 2 milliseconds using only a laptop GPU (NVIDIA Quadro FX 1600M). Right: illustrates how we can use convolution to reduce the high-frequency noise which often appears in images computed using classic Monte Carlo path tracin. Convolution of two 512x512 RGB images, which takes only 5.5 milliseconds using the above mentioned laptop.

GPU FFT and Convolution

The GPU is extremely efficient at computing the Discrete Fourier Transform. Because of its roots in graphics, it is particularly well-suited for working with 4-vectors. This means that two DFTs in parallel is a very good idea (two complex numbers in each 4-vector). We implement two parallel FFTs in GLSL using the classic Cooley-Tukey algorithm, which is now available in the CUDA library called CUFFT. Using the GPU FFT implementation, we can easily implement convolution on the GPU. The two parallel FFTs become a great advantage since we can compute the FFT of corresponding color bands in parallel and multiply them immediately after. We need nine 2D FFTs to convolve two RGB images.

Performance Modeling and Automatic Tuning

Recent advances in computer architecture and computing systems, such as multicore processors and hybrid systems with GPU accelerators, have made the effort required to maximize the performance of applications on such architectures relatively high. We have implemented an auto-tuning framework that can automate the performance tuning process by running a large set of empirical evaluations to configure applications and libraries on the targeted computing platform. A performance model can then be tested or used as a supporting module to narrow the possibly large optimization space of complicated kernels.

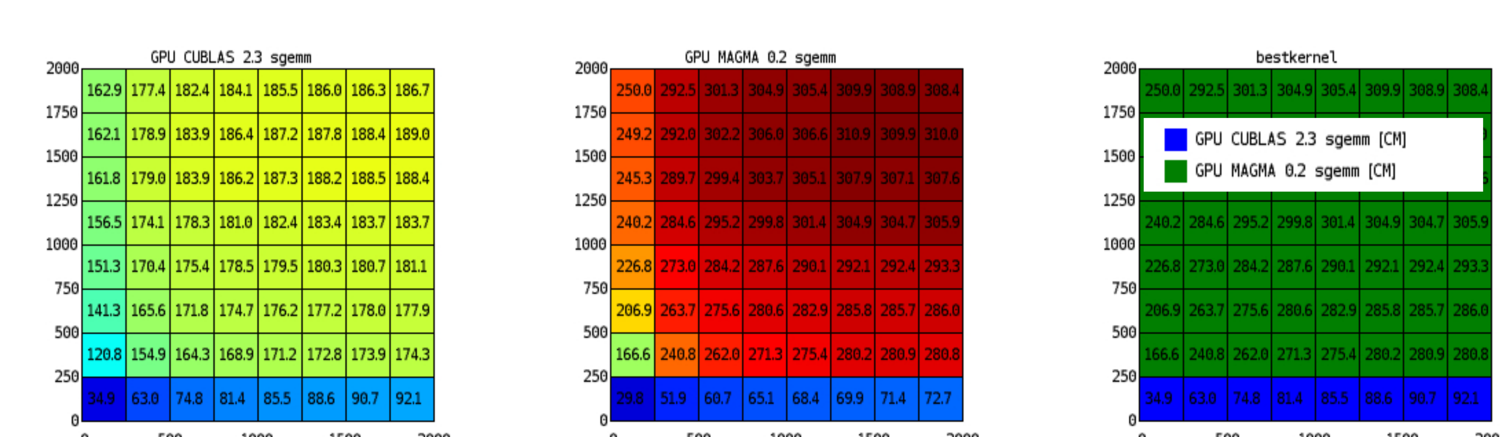

Fig. 3: Auto-tuning performed for the DGEMM routine on the Tesla C1060 device at different matrix sizes up to 2000x2000, where the numbers on the figure represent Gflops obtained. Candidates are the CUDA libraries CUBLAS 2.30 and MAGMA 0.2. We see that for all but the smallest row indexes, the latter is the better choice.

GPU-Lab
DTU Informatics

Per Christian Hansen
Jan Hesthaven
Bernd Dammann
John Bagterp Jørgensen

Allan Engsig-Karup
Jeppe Frisvad
Boyuan Lazarov
Hans Henrik Brandenborg Sørensen

Stefan Glimberg
Rikhardur Einarson
Toke Jansen Hansen
Morten Gorm Madsen