Homework: Curvilinear finite element analysis

Elian NEPPEL

January 12, 2025

Guidance

Pick up one journal article (either of fluid/solid and any other topic, either numerical scheme or application are fine) for numerical study using a high-order / polynomial-type scheme on unstructured mesh

- Solid: Discontinuous Galerkin, hybridized DG
- Fluid: Discontinuous Galerkin, Flux Reconstruction, Spectral Difference, Spectral Volume

Please summarize the content in one-page A4 paper.

Chosen paper

F.D. Witherden, A.M. Farrington, P.E. Vincent, PyFR: An open source framework for solving advection–diffusion type problems on streaming architectures using the flux reconstruction approach, Computer Physics Communications, Volume 185, Issue 11, 2014, Pages 3028-3040, ISSN 0010-4655, https://doi.org/10.1016/j.cpc.2014.07.011.

1 Summary

Tackling high-order numerical methods for fluid advection-diffusion problems, PyFR is an open source flux reconstruction (FR) solver for unstructured mesh. Its main strength lies in its deep parallel computing abilities, unlocking tremendous performances thus making it suitable for industrial applications. It is indeed capable of multi CPU computations as well as GPU computations.

In the past Reynolds Averaged Navier–Stokes (RANS) was the main computation method, due to its compute efficiency. However, it does not perform well for separated flow. FR however is a new, more capable, high-order general method that encompasses several previously discovered technics such as discontinuous Galerkin or spectral difference. Those high order methods typically require more computations, but much less memory accesses. In opposition to the past, current computing hardware is memory-bound, especially when parallelizing tasks, hence this interest in FR that reduces memory load. Modern hardware also supports deep parallelization with vector processing units and GPU, to leverage this PyFR divides the computation in independent tasks.

PyFR uses typical FR equations, while adding a few steps to improve compute. Notably, normalization and standardization of quantities and elements must be sought after. The goal is to later reduce the number of parameters necessary to the computation kernels, this reduces the memory load by effectively replacing memory transfers with computations.

Given the resulting equations and aside from a few compile-time or start-time computations, FR can be expressed using solely matrix multiplication and independent point-wise operations. Matrix multiplication are left to highly optimised BLAS libraries. For point-wise operations, the kernels are compiled at runtime and writen an intermediate language developped by PyFR, instead of a distributing a pre-compiled binary. This allows for the best compiler optimization relative to hardware and given problem. Finaly, great care has been take to break down tasks into independent sub-problems that can execute in parallel while keeping memory requirement low.

At the time of the paper, are natively implemented in PyFR for 2D and 3D: the Euler equations for inviscid compressible flow, and the compressible Navier–Stokes equations for viscous compressible flow. Both require a specific state vector and flux matrix, plus a Reimann solver should be specified. Navier-Stokes is a bit more involved, considerating of the impact of viscosity on the boundaries of the elements.

To study the order of spatial accuracy of the resulting euler method, a vortex was simulated. After waiting for the solution to stabilise in time, the order of accuracy of PyFR using several flux reconstruction method was deduced and corresponds to those found in the litterature for FR. Despite using polynomials of degree 3, the accuracy can reach order 7 – this phenomenon is call super-accuracy. Similar study with a Navier-Stokes specific problem gave similar result, although a slightly less impressive super-accuracy. The capacity of PyFR to handle unsteady flow was also proven.

Performance of the algorithm was evaluated on a single GPU. 90% of the reference maximum memory load was used by PyFR while having very low compute load. The compute load was mainly attributed to the BLAS library, using between 50-75% of the maximum peak. Although it is not perfect, it nevertheless is a very good balance, leveraging most of the hardware. Finaly PyFR was evaluated on a cluster of GPU, performing extremely well in weak scalability metric and a little less so for the strong scalability metric.

Ultimately, PyFR manages to merge the advanced capabilities of FR with modern high-end computing architectures.