# Homework: Curvilinear finite element analysis

Elian NEPPEL (C3TD9115)

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 meshes. Its primary strength lies in its deep parallel computing capabilities, unlocking tremendous performance and making it suitable for industrial applications.

In the past, Reynolds-Averaged Navier–Stokes (RANS) was the primary computational method due to its efficiency. However, it has significant limitations, particularly in handling separated flows. FR, on the other hand, is a novel, more capable high-order method that encompasses several previously discovered techniques, such as discontinuous Galerkin and spectral difference. These high-order methods typically require more computations but significantly fewer memory accesses. Unlike earlier eras, modern computing hardware is memory-bound, especially when parallelizing tasks. This makes FR particularly appealing, as it reduces memory load. Additionally, modern hardware supports deep parallelization, which PyFR leverages to great effect.

PyFR uses standard FR equations while introducing additional mathematical steps to improve computation. Notably, normalization and standardization of quantities and elements are emphasized. These steps aim to reduce the number of parameters required by the computation kernels, effectively replacing memory transfers with computations.

The resulting equations, aside from a few compile-time or start-time computations, can be expressed using solely matrix multiplication and independent point-wise operations. Matrix multiplications are handled by highly optimized BLAS libraries. For point-wise operations, the kernels are written in an intermediate language developed by PyFR and compiled at runtime rather than distributing a precompiled binary. This approach allows for optimal compiler performance tailored to specific hardware and problems. Additionally, great care has been taken to decompose tasks into independent sub-problems that execute in parallel while maintaining low memory requirements.

At the time of writing, PyFR natively implements 2D and 3D versions of the Euler equations for inviscid compressible flow and the compressible Navier–Stokes equations for viscous compressible flow. Both require specific state vectors, flux matrices, and Riemann solvers. Navier–Stokes is more complex due to the viscosity on the element boundaries.

To evaluate the spatial accuracy of the Euler method, a vortex simulation was conducted. After allowing the solution to stabilize in time, the order of accuracy of PyFR using various flux reconstruction methods was found to match existing FR literature. Despite using polynomials of degree 3, the accuracy achieved order 7, a phenomenon known as super-accuracy. A similar study with a Navier–Stokes-specific problem gave comparable results, although with slightly reduced super-accuracy. PyFR's capacity to handle unsteady flow was also demonstrated.

The algorithm's performance was evaluated on a single GPU. PyFR utilized 90% of the reference maximum memory load while maintaining a relatively low compute load. The computational load was mainly attributed to the BLAS library, which operated at 50–75% of the maximum peak performance. While not perfect, this represents a very efficient balance, leveraging most of the available hardware. PyFR was further tested on a GPU cluster, performing exceptionally well in weak scalability metrics and slightly less impressively in strong scalability metrics.

Ultimately, PyFR successfully combines the advanced capabilities of flux reconstruction methods with the potential of modern high-end computing architectures.