# PyFR: An Open Source Python Framework for High-Order CFD on Heterogeneous Platforms

http://www.pyfr.org @PyFR_Solver

F. D. Witherden, B. C. Vermeire, P. E. Vincent

*Department of Aeronautics, Imperial College London, SW7 2AZ, UK*

## Introduction

Theoretical studies and numerical experiments suggest that high-order methods for unstructured grids can solve hitherto intractable fluid flow problems in the vicinity of complex geometrical configurations. The flux reconstruction (FR) approach, developed by Huynh [1], is a simple yet efficient high-order scheme that is particularly amenable to the requirements of modern hardware architectures—specifically by having a large degree of structured computation. Using the FR approach it is possible to unify various popular high-order methods such as nodal discontinuous Galerkin and spectral difference schemes.

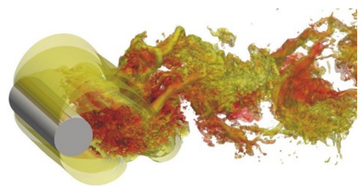
The majority of operations within an FR step can be cast as block-by-panel type matrix multiplications [2]. Depending on the polynomial order these multiplications account for between 50% and 85% of wall-clock time. All remaining operations are pointwise and so are amenable to acceleration.

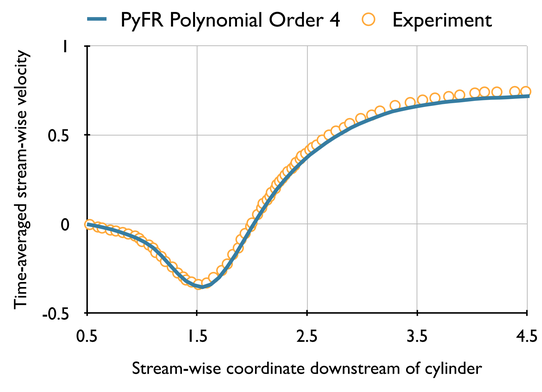
## Implementation

Our implementation of FR is called PyFR. As suggested by its namesake PyFR is written almost entirely in Python. The current version of PyFR is capable of solving the compressible Navier-Stokes equations on unstructured grids of hexahedra, tetrahedra and prismatic elements and is explicit in time. To achieve both performance and portability PyFR makes extensive use of run-time code generation. Pointwise operations are specified in a templating language, based around the Mako templating engine, which are dynamically translated into either CUDA, OpenCL, or C/OpenMP. Using MPI PyFR is capable of running on distributed memory clusters. To eliminate bottlenecks on the PCIe bus PyFR performs all computations on the device. Data is only ever copied for the purposes of exchanging halo data with other MPI ranks or when writing a solution to disk.

## Validation

The numerics of PyFR have been verified against both analytical and experimental results for a range of benchmark flow problems. This validation has been performed across a range of polynomial orders and mesh types. Turbulent flow over a cylinder at Reynolds number 3900 and Mach number 0.2 has been analysed and compared against the experimental data of [3]. Simulations were undertaken using a hexahedral mesh with 118070 elements.






## Performance Portability

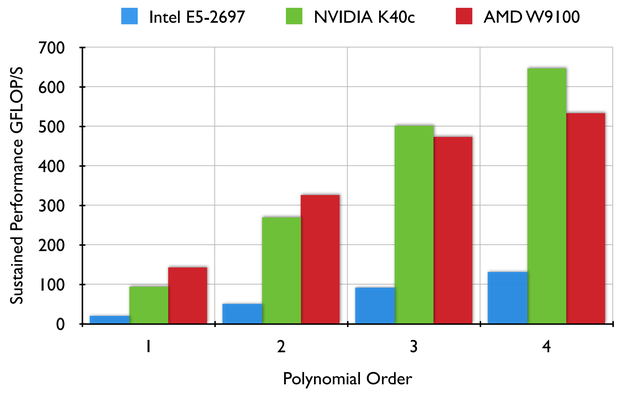
Matrix multiplication is an extremely well studied performance primitive. High performance routines are readily available through vendor-provided BLAS libraries. The figure on the left shows the sustained double-precision performance of PyFR at a range of polynomial orders for the hexahedral cylinder simulation described above. Configuration: an Intel Xeon E5-2697 v2 CPU employing the C/OpenMP backend and MKL; an NVIDIA Tesla K40c GPU using the CUDA backend and cuBLAS; and an AMD FirePro W9100 using the OpenCL backend with clBLAS.

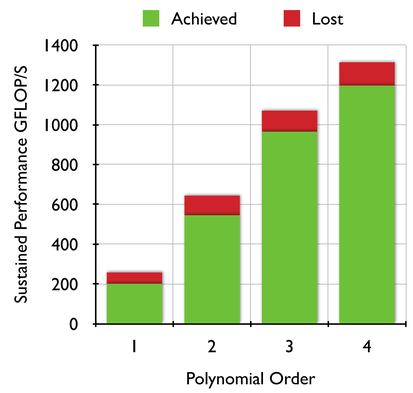
In the case of fourth order solution polynomials PyFR on the Intel E5-2697 and the NVIDIA K40c can be seen to obtain close to 50% of the theoretical peak FLOPS (280 and 1430 GFLOP/S, respectively).

## Heterogenous Operation

By design the nature of the halo data that is exchanged between MPI ranks in PyFR is independent of the backend being used. PyFR is therefore capable of heterogenous operation with different ranks employing different backends. As a demonstration of this the aforementioned cylinder mesh was suitably decomposed into three partitions and run across the Intel E5-2697, the NVIDIA K40c, and the AMD W9100 in a single desktop workstation. The resulting performance numbers are shown on the right figure. 'Lost' FLOP/S indicate the difference between the 'Achieved' FLOP/S and the theoretical limit indicated by the figure on the left.

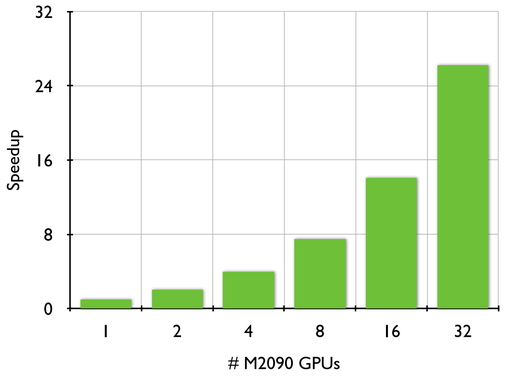
Looking at the fourth order simulation we can see PyFR achieving 1200 GFLOP/S out of a possible 1300 GFLOP/S for a heterogeneous efficiency of 92%.

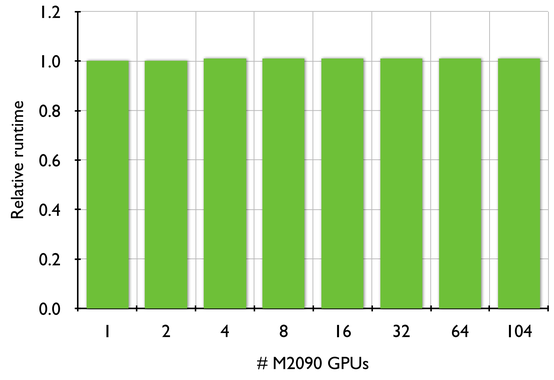



## Scalability

The scalability of PyFR for the three dimensional compressible Navier-Stokes equations on hexahedral mesh elements was evaluated on the Emerald GPU cluster with third order solution polynomials. A mesh with 114688 elements was generated and run on a single NVIDIA M2090 with error correcting codes (ECC) enabled. To evaluate the strong scalability of PyFR this mesh was partitioned into *N* segments of equal size. The resulting speedups compared to a single M2090 can be seen in the left figure. We note that with 32 GPUs that a speedup of ~26 times can be observed. As a means of evaluating the weak scalability of the code another series of meshes were created with the number of elements being proportional to the number of GPUs. Simulation times relative to a single M2090 can be seen to the right. Weak scalability is observed as being near-perfect. The 104 GPU run contained almost four billion degrees of freedom and a working set of ~485 GiB.




## Acknowledgements

The authors would like to thank the Engineering and Physical Sciences Research Council for their support via a Doctoral Training Grant and an Early Career Fellowship (EP/K027379/1). The authors would also like to thank AMD, Intel, and NVIDIA for hardware donations.

## References

[1] H. T. Huynh. A flux reconstruction approach to high-order schemes including discontinuous Galerkin methods. AIAA Paper 2007-4079. 2007

[2] Freddie D Witherden, Antony M Farrington, and Peter E Vincent. Pyfr: An open source framework for solving advection-diffusion type problems on streaming architectures using the flux reconstruction approach. arXiv preprint arXiv:1312.1638, 2013.

[3] Parnaudeau, Philippe and Carlier, Johan and Heitz, Dominique and Lamballais, Eric. Experimental and numerical studies of the flow over a circular cylinder at Reynolds number 3900. Physics of Fluids, 2008.