PyFR: Heterogeneous Computing from a Homogeneous Codebase

P. E. Vincent

Department of Aeronautics
Imperial College London

20th November 2014 (Antony’s Birthday!)
Overview

- Motivation
- Flux Reconstruction
- Modern Hardware
- PyFR
- Results
- Summary
Current industry standard CFD tools have limited capabilities
Technology is decades old and designed for solving steady flow problems (using RANS approach)
Technology is decades old and designed for solving steady flow problems (using RANS approach).
Need to expand the ‘industrial CFD envelope’
• “reliable use of CFD has remained confined to a small but important region of the operating design space due to the inability of current methods to reliably predict turbulent separated flows” [2]

• Objective of our research is to advance industrial CFD capabilities from their current ‘RANS plateau’
Motivation

• We aim to develop the *de facto* industry standard technology for *affordable* (and hence industrially relevant) high-fidelity scale-resolving simulations of *unsteady flow* phenomena within the vicinity of *complex geometric configurations*
• Achieved by intelligently leveraging benefits of (and synergies between) high-order Flux Reconstruction (FR) methods for unstructured grids and massively-parallel modern hardware platforms
Motivation

Flux Reconstruction + Modern Hardware

[Images of various scientific and technological themes]
Flux Reconstruction (FR) approach to high-order methods was first proposed by Huynh in 2007 [3]

- **High-order** accurate in space
- Works on **unstructured grids**

• So ...

High Accuracy + Complex Geometry
Flux Reconstruction

- Nature of FR scheme depends on location of solution points, interface flux, **correction function**

- Can recover a wide range of schemes via judicious choice of correction function [4]

- A one-parameter family of provably stable FR schemes have been identified [5]


We have recently identified a new multi-parameter family of stable FR schemes.
Flux Reconstruction

- Plot stable region of parameter space …
• Original one-parameter family ...
• Perform a whole bunch of numerical experiments …
• And record which ones were stable/unstable …
Modern Hardware
Modern Hardware
Motivation | Flux Reconstruction | Modern Hardware | PyFR | Results | Summary

Modern Hardware

[Images of various modern hardware components, including Intel Xeon, AMD Opteron processors, and NVIDIA graphics cards.]
• **FLOPS** increasing faster than **memory bandwidth**

---

• Also **FLOPS** come in **parallel** …
Modern Hardware

- Multiple **Cores** each with **Vector Unit**

- Parallelism exposed via **MIMD** (Multiple Instruction Multiple Data) + **SMT** (Simultaneous Multi Threading) + **SIMD** (Single Instruction Multiple Data) + **ILP** (Instruction Level Parallelism)
Modern Hardware

- **Nvidia GPUs** - multiple **Streaming Multi-Processors** each with **CUDA Cores**

- Parallelism exposed via **SIMT** (Single Instruction Multiple Thread)
Modern Hardware

- **AMD GPUs** - multiple **Compute Units** each with Stream Processors
- Parallelism exposed via **SIMT** (Single Instruction Multiple Thread)
Modern Hardware

• Also, different programming languages for different devices
Modern Hardware

• So a **challenging** environment ...
Modern Hardware

• So a challenging environment ...
• But significant **FLOPS** now available if they can be harnessed …

2.6TFLOPS (Double Precision)
Flux Reconstruction
+
Modern Hardware

PyFR
PyFR

Motivation | Flux Reconstruction | Modern Hardware | PyFR | Results | Summary
### PyFR

• **Features** (**v0.2.3 - current release**)  

| Governing Equations | Compressible Euler  
| Compressible Navier Stokes |
|----------------------|----------------------|
| **Spatial Discretisation** | Arbitrary order FR on mixed unstructured grids (tris, quads, hexes, tets, prisms) |
| **Temporal Discretisation** | Range of explicit Runge-Kutta schemes |
| **Platforms** | CPU clusters (C-OpenMP-MPI)  
| | Nvidia GPU clusters (CUDA-MPI)  
| | AMD GPU clusters (OpenCL-MPI) |
| **Precision** | Single  
| | Double |
| **Input** | Gmsh |
| **Output** | Paraview |
**PyFR**

- **Features** *(v0.2.3 - current release)*

<table>
<thead>
<tr>
<th>Governing Equations</th>
<th>Compressible Euler</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Compressible Navier Stokes</td>
</tr>
<tr>
<td>Spatial Discretisation</td>
<td>Arbitrary order FR on mixed unstructured grids (tris, quads, hexes, tets, prisms)</td>
</tr>
<tr>
<td>Temporal Discretisation</td>
<td>Range of explicit Runge-Kutta schemes</td>
</tr>
<tr>
<td>Platforms</td>
<td>CPU clusters (C-OpenMP-MPI)</td>
</tr>
<tr>
<td></td>
<td>Nvidia GPU clusters (CUDA-MPI)</td>
</tr>
<tr>
<td></td>
<td>AMD GPU clusters (OpenCL-MPI)</td>
</tr>
<tr>
<td>Precision</td>
<td>Single</td>
</tr>
<tr>
<td></td>
<td>Double</td>
</tr>
<tr>
<td>Input</td>
<td>Gmsh</td>
</tr>
<tr>
<td>Output</td>
<td>Paraview</td>
</tr>
</tbody>
</table>
• Python Outer Layer (Hardware Independent)

  Python Outer Layer
  (Hardware Independent)

  • Setup
  • Distributed memory parallelism
  • Outer ‘for’ loop and calls to
    Hardware Specific Kernels
• Need to generate the **Hardware Specific Kernels**

**Python Outer Layer**  
(Hardware Independent)

• Setup
• Distributed memory parallelism
• Outer ‘for’ loop and calls to **Hardware Specific Kernels**
PyFR

- Two types of kernel are required ...

  - Python Outer Layer (Hardware Independent)
    - Setup
    - Distributed memory parallelism
    - Outer ‘for’ loop and calls to Hardware Specific Kernels

  - Matrix Multiply Kernels
    - Data interpolation/extrapolation etc.

  - Point-Wise Nonlinear Kernels
    - Flux functions, Riemann solvers etc.
• For **matrix multiply kernels** it is pretty easy …

- **Python Outer Layer** (Hardware Independent)
  - Setup
  - Distributed memory parallelism
  - Outer ‘for’ loop and calls to Hardware Specific Kernels

- **Matrix Multiply Kernels**
  - Data interpolation/extrapolation etc.

- **Point-Wise Nonlinear Kernels**
  - Flux functions, Riemann solvers etc.

Use DGEMM from vendor supplied BLAS
• Harder for **point-wise nonlinear kernels** ...
• These can now be called

Python Outer Layer (Hardware Independent)
- Setup
- Distributed memory parallelism
- Outer ‘for’ loop and calls to Hardware Specific Kernels

Matrix Multiply Kernels
- Data interpolation/extrapolation etc.

Point-Wise Nonlinear Kernels
- Flux functions, Riemann solvers etc.

C/OpenMP Hardware Specific Kernels
CUDA Hardware Specific Kernels
OpenCL Hardware Specific Kernels

Use DGEMM from vendor supplied BLAS

Pass Mako derived kernel templates through Mako derived templating engine
PyFR

• These can now be called

Python Outer Layer (Hardware Independent)
- Setup
- Distributed memory parallelism
- Outer ‘for’ loop and calls to Hardware Specific Kernels

C/OpenMP Hardware Specific Kernels
CUDA Hardware Specific Kernels
OpenCL Hardware Specific Kernels

Matrix Multiply Kernels
- Data interpolation/extrapolation etc.

Use DGEMM from vendor supplied BLAS

Point-Wise Nonlinear Kernels
- Flux functions, Riemann solvers etc.

Pass Mako derived kernel templates through Mako derived templating engine
• ~5.5k lines of Python
• Open source ‘3 Clause New Style BSD License’
• Website: www.pyfr.org
• Twitter: @PyFR_Solver
• Paper: Computer Physics Communications [3]

• **Taylor-Green** vortex breakdown

• Re = 1600

• Ma = 0.1

• Compare with van Rees et al. [7]

• A movie …
• A movie …
- van Rees et al. spectral DNS + PyFR (2nd order hex)
• van Rees et al. spectral DNS + PyFR (3rd order hex)
van Rees et al. spectral DNS + PyFR (4th order hex)
• van Rees et al. spectral DNS + PyFR (5th order hex)
• van Rees et al. spectral DNS + PyFR (6th order hex)
Results

- van Rees et al. spectral DNS + PyFR (6th order hex)
• $L_\infty$ error in decay rate
- $L_{\infty}$ error in decay rate
• $L_\infty$ error in decay rate
Results

- $L_\infty$ error in decay rate
Results

- $L_\infty$ error in decay rate
Results

- $L_\infty$ error in decay rate
- $L_\infty$ difference between decay rate and enstrophy
• $L_\infty$ difference between decay rate and enstrophy
Results

- $L_\infty$ difference between decay rate and enstrophy

![Graph showing log(error) vs log(work)]
• $L_\infty$ difference between decay rate and enstrophy
• \( L_\infty \) difference between decay rate and enstrophy
• $L_\infty$ difference between decay rate and enstrophy
• Flow over a circular cylinder

• Re = 3900

• Ma = 0.2

• Compare with Parnaudeau et al. [6]

• A movie …
• A movie …
• Parnaudeau et al. experiment
• Parnaudeau et al. experiment + Parnaudeau et al. LES
Results

• Parnaudeau et al. experiment + PyFR (5th order hex)
• Parnaudeau et al. experiment
• Parnaudeau et al. experiment + Parnaudeau et al. LES
• Parnaudeau et al. experiment + PyFR (5th order hex)
• Parnaudeau et al. experiment
Results

- Parnaudeau et al. experiment + Parnaudeau et al. LES
• Parnaudeau et al. experiment + PyFR (5th order hex)
Results

- Flow over a NACA 0021 at 60 degree AoA
- Re = 270,000
- Ma = 0.2
• A movie …
• A movie …
Results

- Performance on a **heterogeneous workstation**
Results

- **Single-node** on prism/tetrahedral mesh

<table>
<thead>
<tr>
<th>Polynomial Order</th>
<th>W9100 (OpenCL)</th>
<th>K40c (CUDA)</th>
<th>E5-2697 (C/OpenMP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>175</td>
<td>350</td>
</tr>
<tr>
<td>2</td>
<td>350</td>
<td>525</td>
<td>700</td>
</tr>
<tr>
<td>3</td>
<td>525</td>
<td>700</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>700</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Results

- Multi-node heterogeneous on prism/tetrahedral mesh
Summary

Flux Reconstruction + Modern Hardware
Funding

EPSRC
Pioneering research and skills

Technology Strategy Board

AIRBUS

nvidia

CUDA RESEARCH CENTER

nvidia

AMD

FIREPRO TECHNOLOGY

intel
• The authors would like to acknowledge the work presented here made use of the EMERALD HPC facility provided by the Centre for Innovation.