Examining the performance of Nektar++ for fusion applications

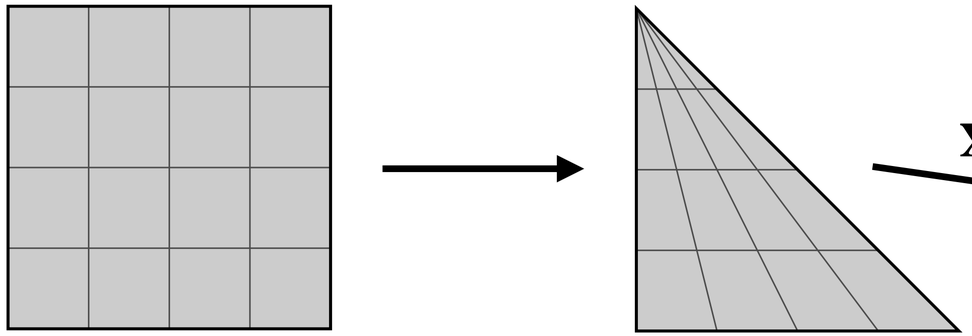
David Moxey
College of Engineering, Maths & Physical Sciences, University of Exeter

Chris Cantwell & Spencer Sherwin
Department of Aeronautics, Imperial College London

Overview

- The overall aim of this project is to examine the use & challenges for high-order spectral element methods within NEPTUNE.
- In particular, we will focus on the development and numerical requirements for a prototypical 2D anisotropic heat transport problem (output 2.1.5).
- This work is split into three tasks to align with the requirements:
  - **Task 1:** Curvilinear mesh generation (lead: UoE).
  - **Task 2:** Developing flexible and performance-portable proxyapps (lead: ICL).
  - **Task 3:** Community engagement with NEPTUNE partners (lead: UoE/ICL).

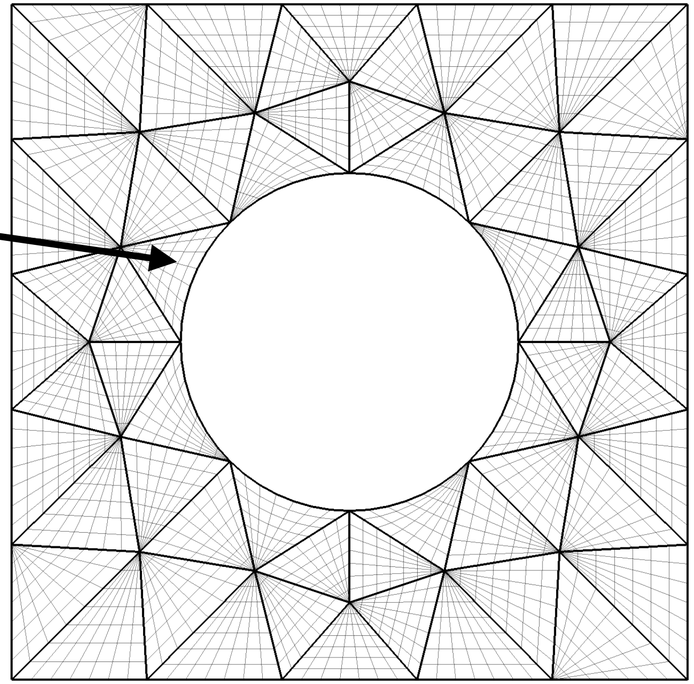
Spectral element formulation

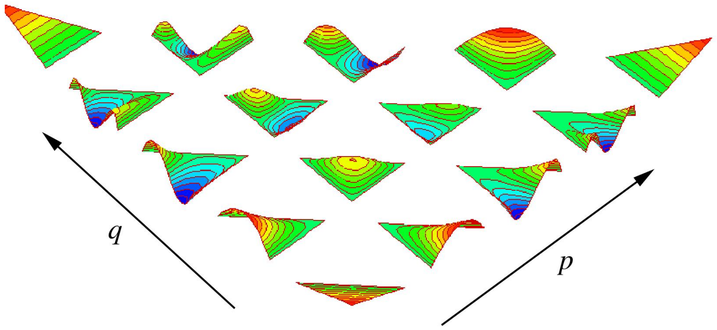


collapsed coordinates
$(\eta_1, \eta_2) \in [-1,1]^2$

reference element
$(\xi_1, \xi_2) \in \Omega_{\text{st}}$

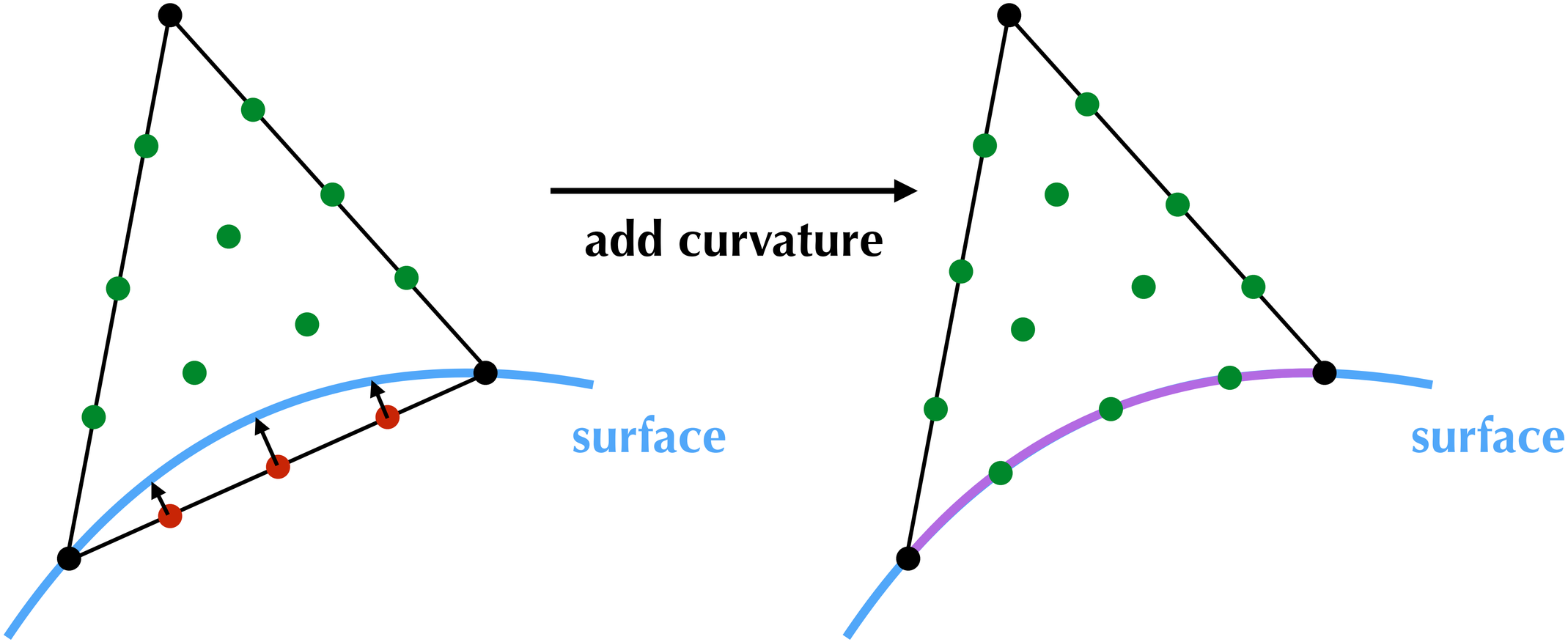
$\mathbf{x} = \chi^e(\xi)$





(C⁰) tensor product basis
$\phi_p^a(\eta_1)\,\phi_{pq}^b(\eta_2)$

Task 1: high-order mesh generation



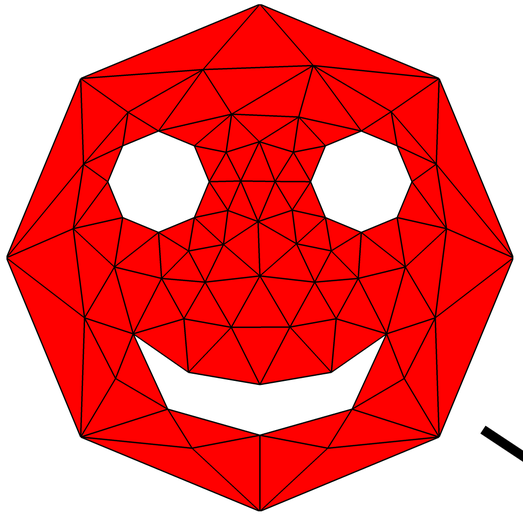
Task 1: high-order mesh generation

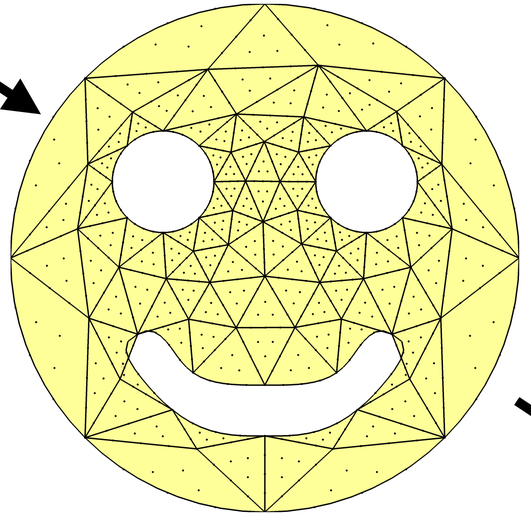
- Under task 1 we will investigate high-order mesh generation challenges of relevance to the fusion community:
  - **D1.1:** Generation of meshes to cluster around X-point configurations.
  - **D1.2:** Extension to quadrilateral-based meshes (using cross-field solver).
  - **D1.3:** Quantification of surface mesh quality.
  - **D1.4:** End-user interfaces & workflow integration.

Variational optimisation

$$\min_{\phi} \mathscr{E}(\phi) = \min_{\phi} \int_{\Omega_I} W(\nabla\phi)\, dy$$

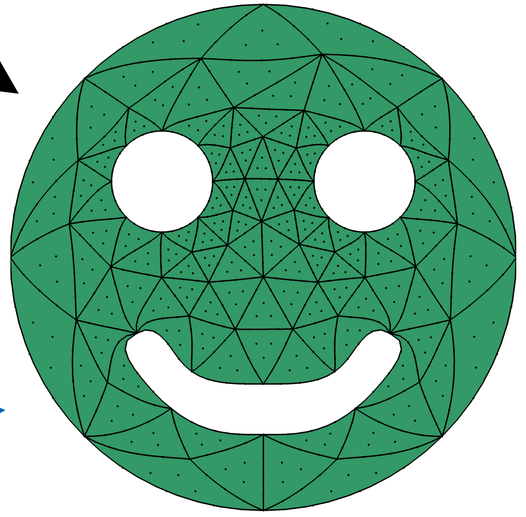
Straight-sided mesh





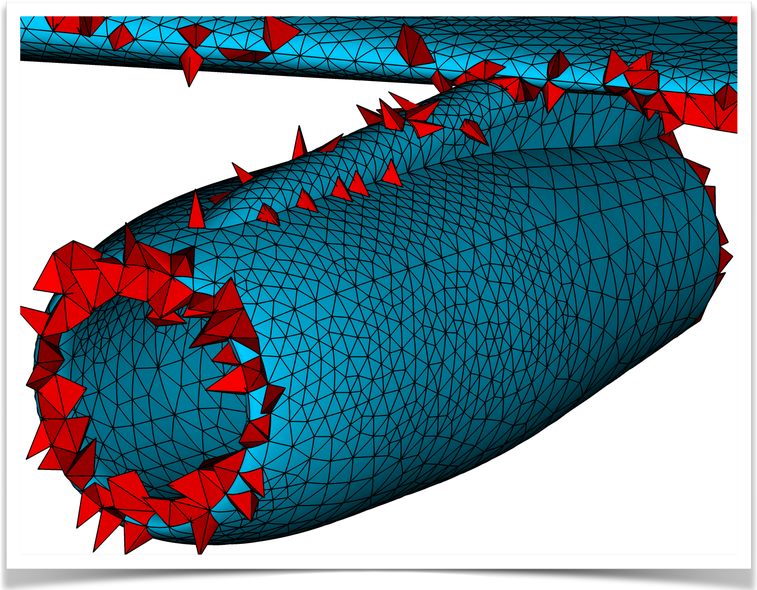
Boundary projection

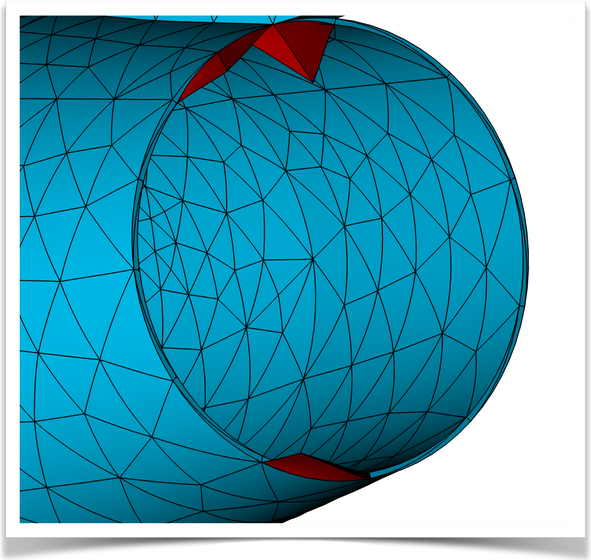
ϕ

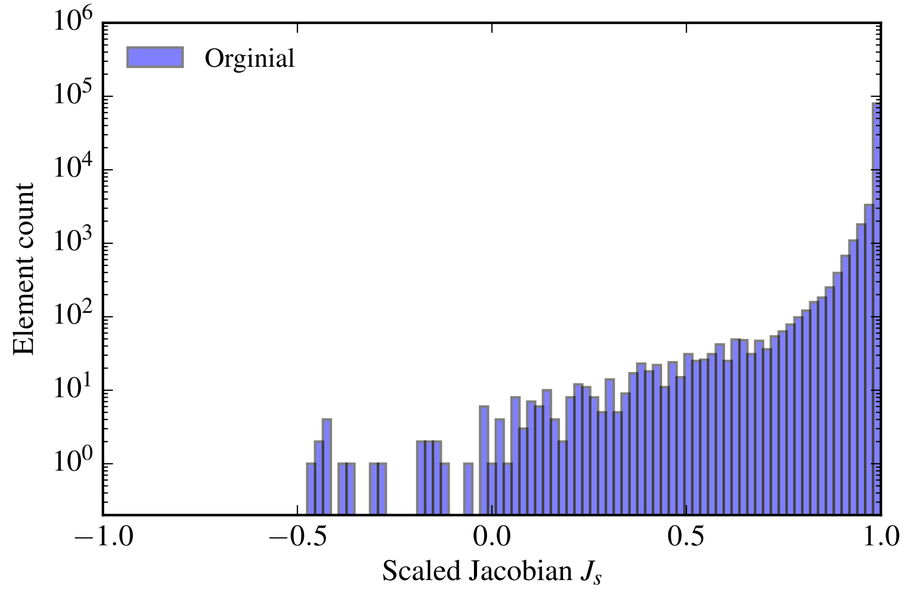


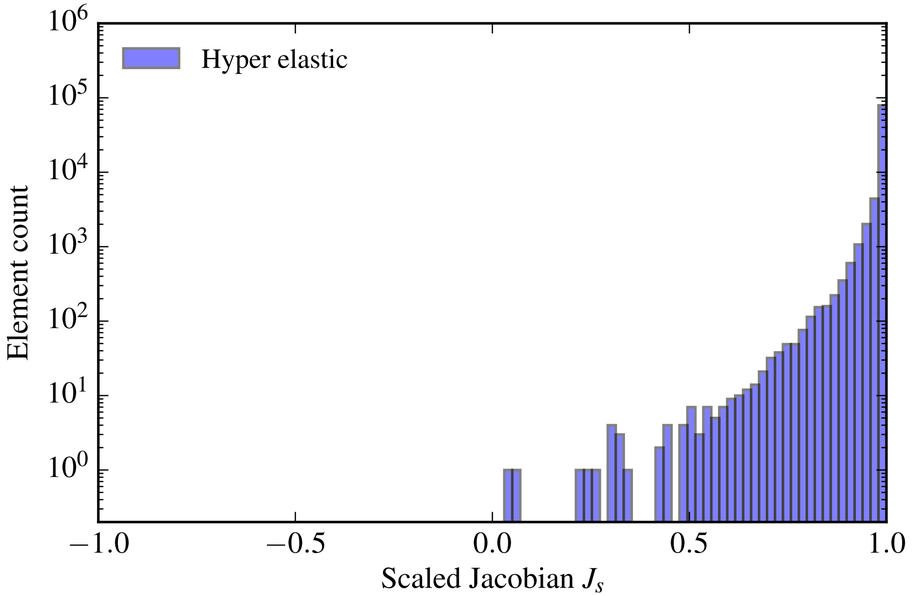
Deformed mesh

Example: DLR F6 engine



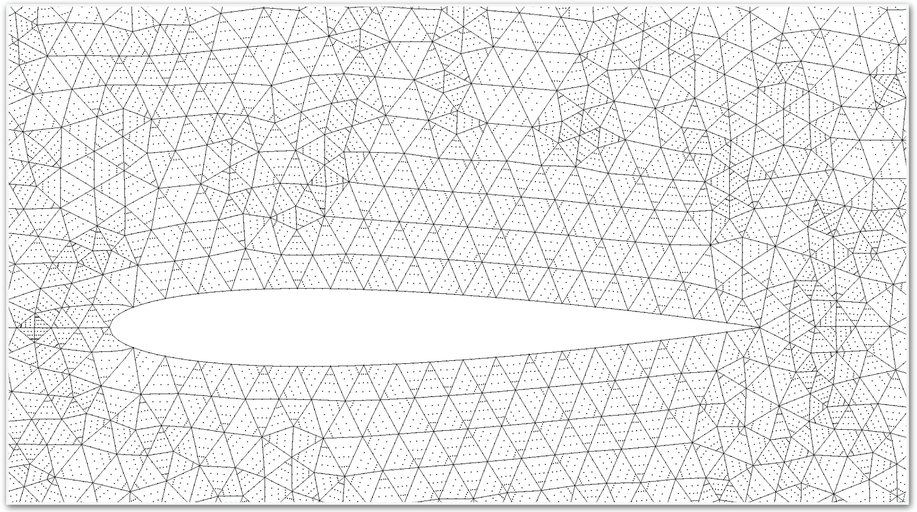


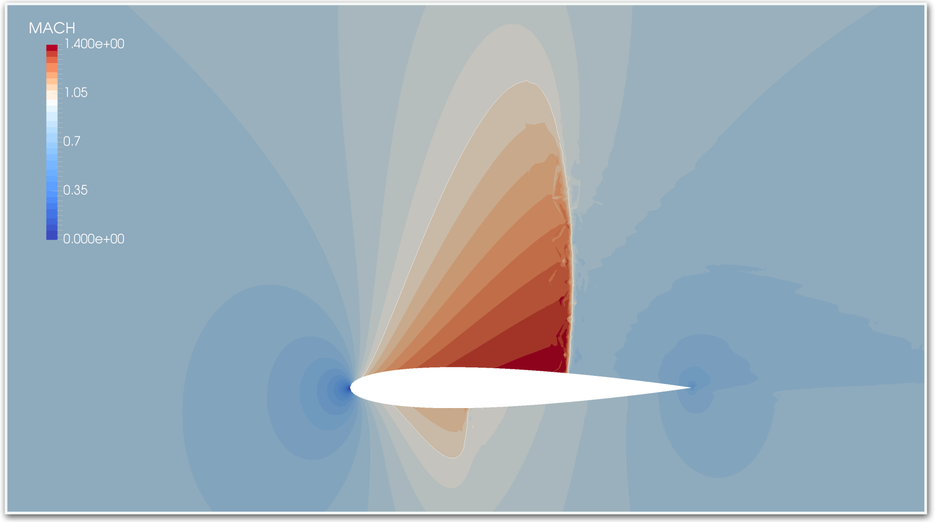




Use in adapted meshes

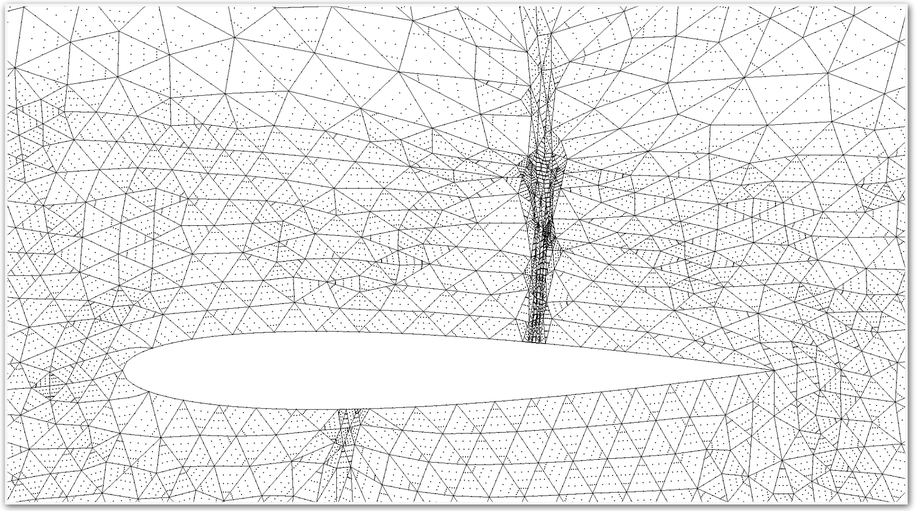
Starting mesh

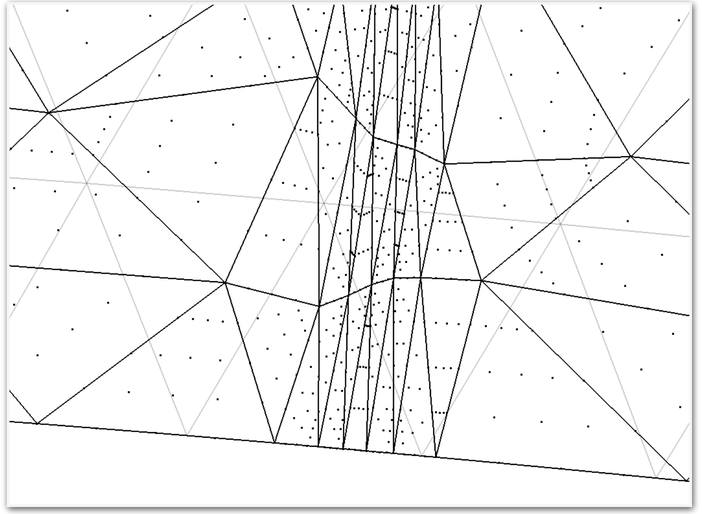




Initial simulation

Calculate target size & do r-adaptation





Use of CAD sliding

Task 2: Developing flexible & performance portable proxyapps

- Focus of this task will be development of performance portable solver for anisotropic heat transport problem.

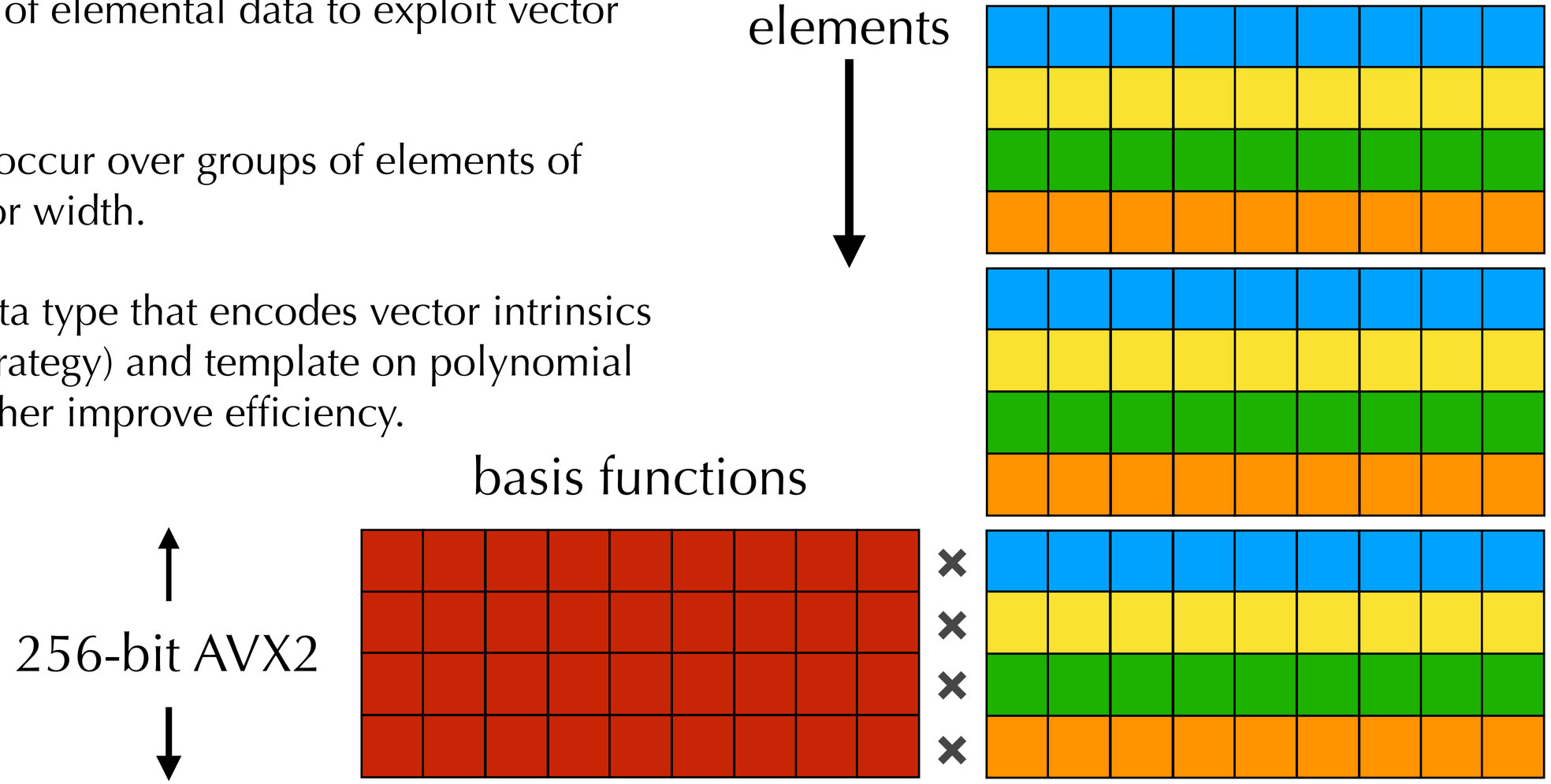
- Key to attaining peak performance on most modern hardware is exploiting arithmetic intensity & avoid memory bandwidth bottlenecks as much as possible.

- High-order methods have been shown to be highly performant in matrix-free formulations, combined with sum-factorisation/tensor contraction, particularly for quad/hex based meshes.

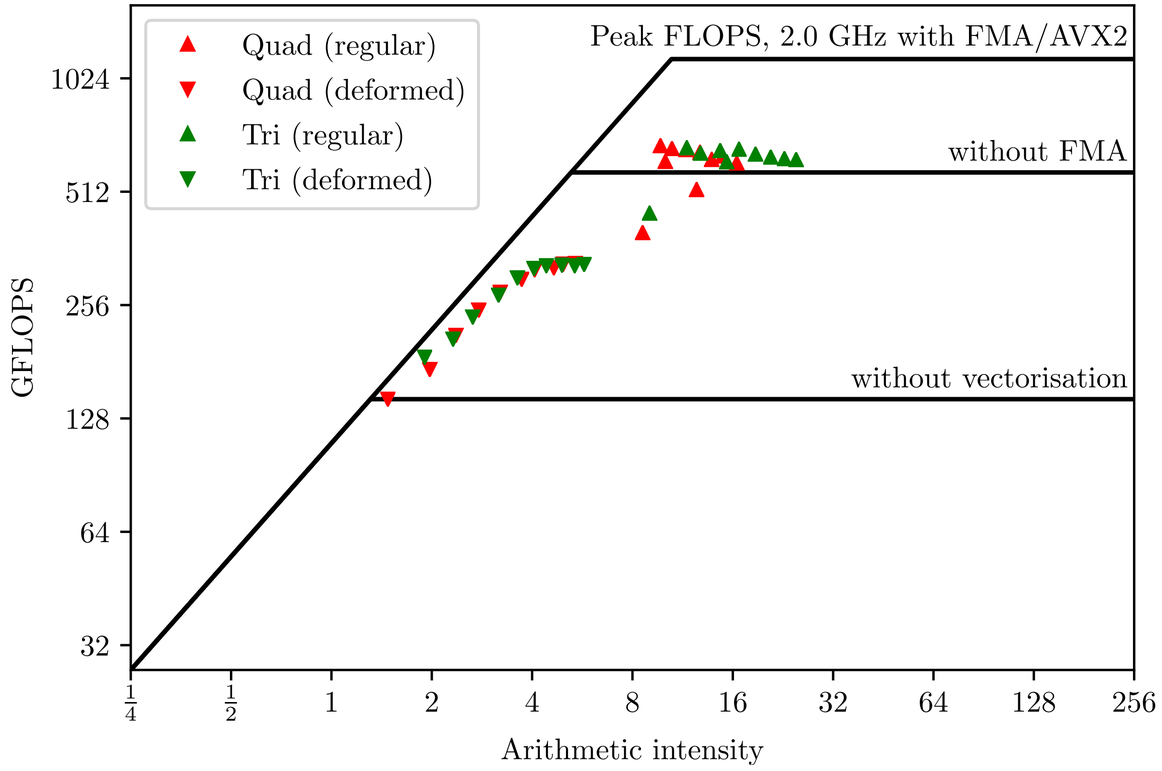
- Aim in this task is to start from a baseline x86 app, integrate matrix-free methods and extend this to multiple architectures (ARM/GPU); liaise with other groups to investigate preconditioners.

Data layout

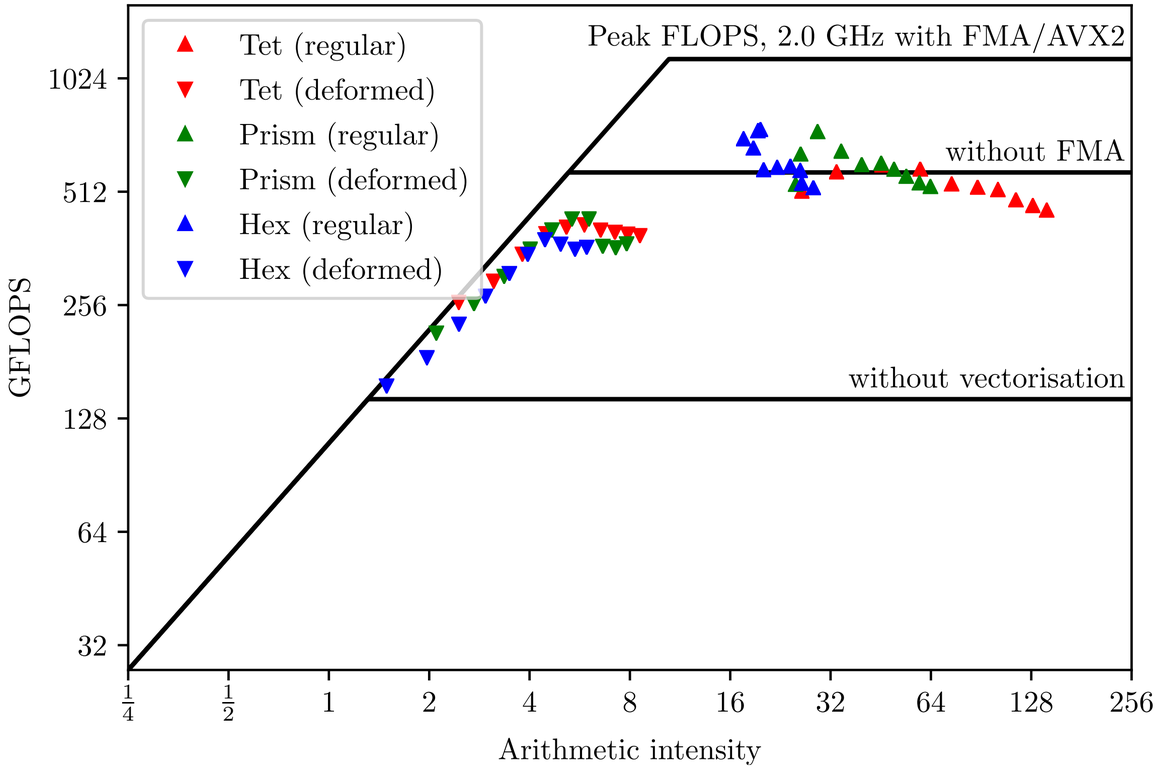
- Interleaving of elemental data to exploit vector instructions.
- Operations occur over groups of elements of size of vector width.
- Use C++ data type that encodes vector intrinsics (common strategy) and template on polynomial order to further improve efficiency.



Roofline results



2D: Quads, triangles



3D: Hexahedra, prisms, tetrahedra

Use of ~50-70% peak FLOPS for regular elements

Task 3: engagement with NEPTUNE partners

- This task focuses on collaborative efforts with other partners in NEPTUNE.
  - **D3.1:** Proof-of-concept proxyapp coupling via CWIPI.
  - **D3.2:** Training materials, tutorials, documentation, continuous integration & delivery.
  - **D3.3:** Flexible coordination efforts (e.g. workshops/seminars/tutorials) to engage with other NEPTUNE partners.

Nektar++

spectral/hp element framework

- Nektar++ is an **open source framework** for high-order methods.
- Although fluid dynamics is a key application area, we try to make it easier to use these methods in other areas.
- C++ API, with ambitions to bridge current and future hardware diversity (e.g. many-core processors, GPUs).
- Modern development practices with continuous integration, git, etc.

Recent RSE developments

Docker images for latest master

Complete parallel build, incl. tutorials

```
docker pull nektarpp/nektar      # binaries
docker pull nektarpp/nektar-dev # dev image
```

Jupyter notebook, contains Python interface as well as utilities for pre- and post-processing

```
docker pull nektarpp/nektar-workbook
```

Working on pre-built binaries using continuous delivery: easier installs and releases

Python interface

```cpp
#include <LibUtilities/BasicUtils/SessionReader.h>
#include <SpatialDomains/MeshGraph.h>

session = SessionReader::CreateInstance(argc, argv);
mesh    = SpatialDomains::Read(session);
cout << mesh->GetMeshDimension() << endl;
```

C++

```python
from NekPy.LibUtilities import SessionReader
from NekPy.SpatialDomains import MeshGraph

session = SessionReader.CreateInstance(sys.argv)
mesh    = MeshGraph.Read(session)
print(mesh.GetMeshDimension())
```

Python

- Nektar++ is quite a complex code; trying to lower barrier to entry with Python interface.
- Uses boost::python, good support for inheritance, shared pointers
- The interface is a work-in-progress: needs improvements which this project will support.

Summary

- Upcoming deliverable by end of FY20/21 is D1.3 (surface mesh quality report). Main thrust of efforts will be in FY21/22.
- Recruitment open for a PDRA to support these efforts.
- Lots of challenges and lots to do!