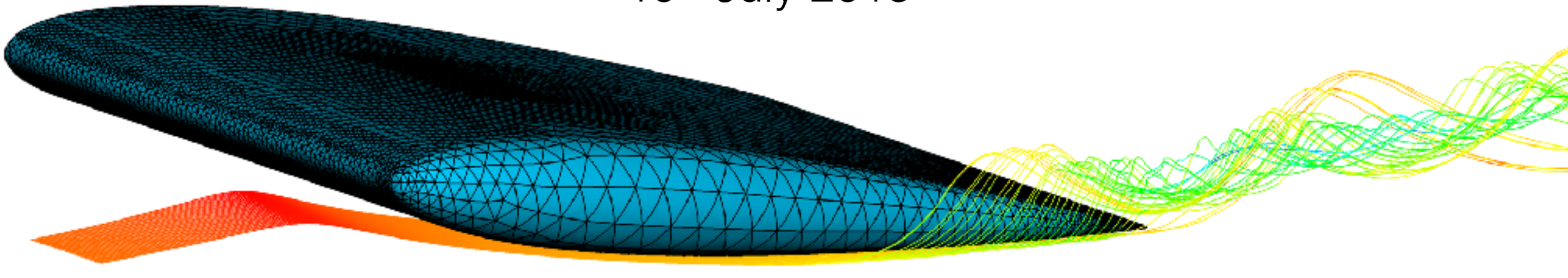
Towards automatic high-order mesh generation for industrial geometries

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

Michael Turner, Jan Eichstädt, Julian Marcon, Joaquim Peiró
Department of Aeronautics, Imperial College London

ICOSAHOM 2018, London, UK
10th July 2018

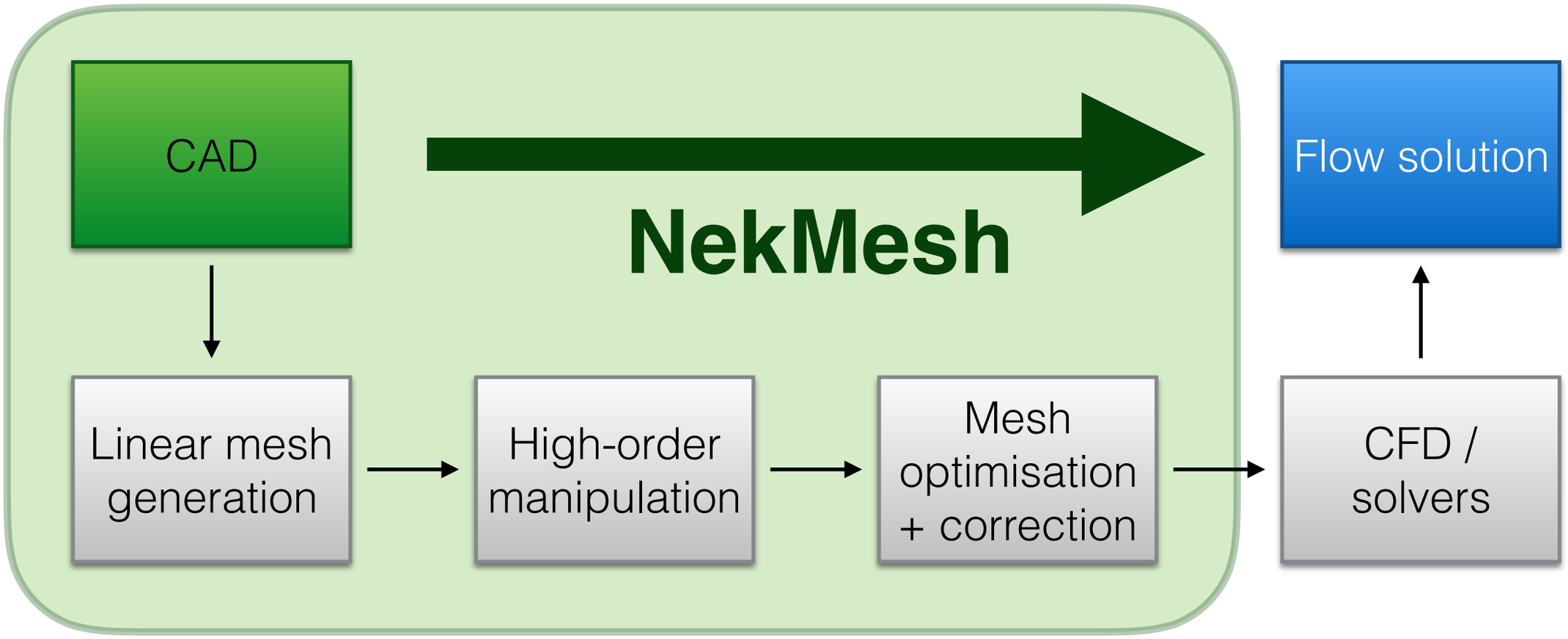


Overview

- NekMesh
- Mesh correction & optimisation with GPUs
- Moving meshes for *rp*-adaptation
- Summary

NekMesh goals

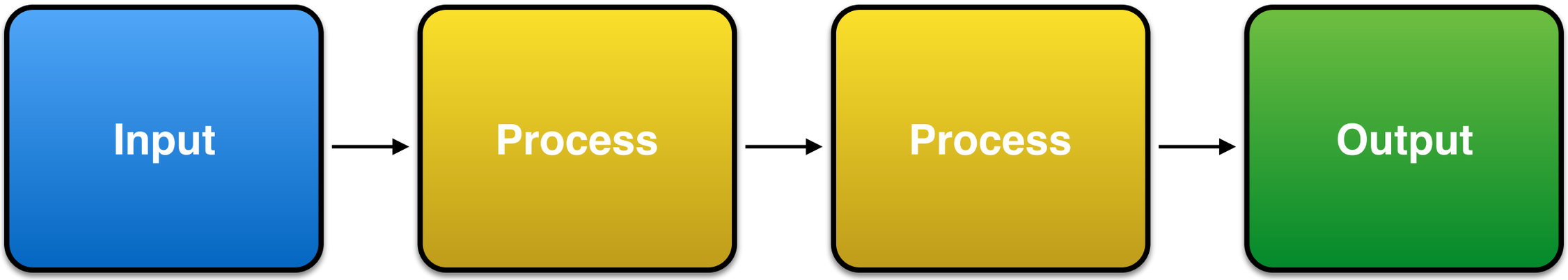
Single step process from CAD to flow solution
As little user interaction throughout
Preserve CAD throughout



NekMesh goals

- Be capable of generating 3D curvilinear meshes of complex geometries with boundary layers
- CAD interface to different CAD engines
- Be capable of optimising quality and correcting invalid meshes where present
- Support export and import to/from various external file formats
- Support high-order manipulation of linear meshes
- Open source

Preprocessing



Nektar++
CAD
Nektar
Gmsh
Tecplot
StarCCM+
PLY
VTK
Semtex

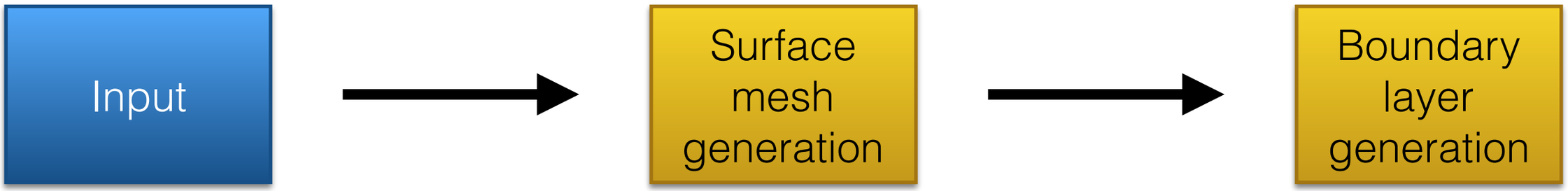
High-order
smoothing

Boundary
refinement

Nektar++
Gmsh
VTK (linear)

Mesh translated through pipeline of processing modules

NekMesh: workflows & modules



- OpenCascade
- CFI (CADFix)
- Many others (Gmsh, Star-CCM+)

- In-house code based on octree + triangle
- CFI (CADFix)

- In-house code
- Star-CCM+

- In-house code
- Star-CCM+
- CFI (CADFix)

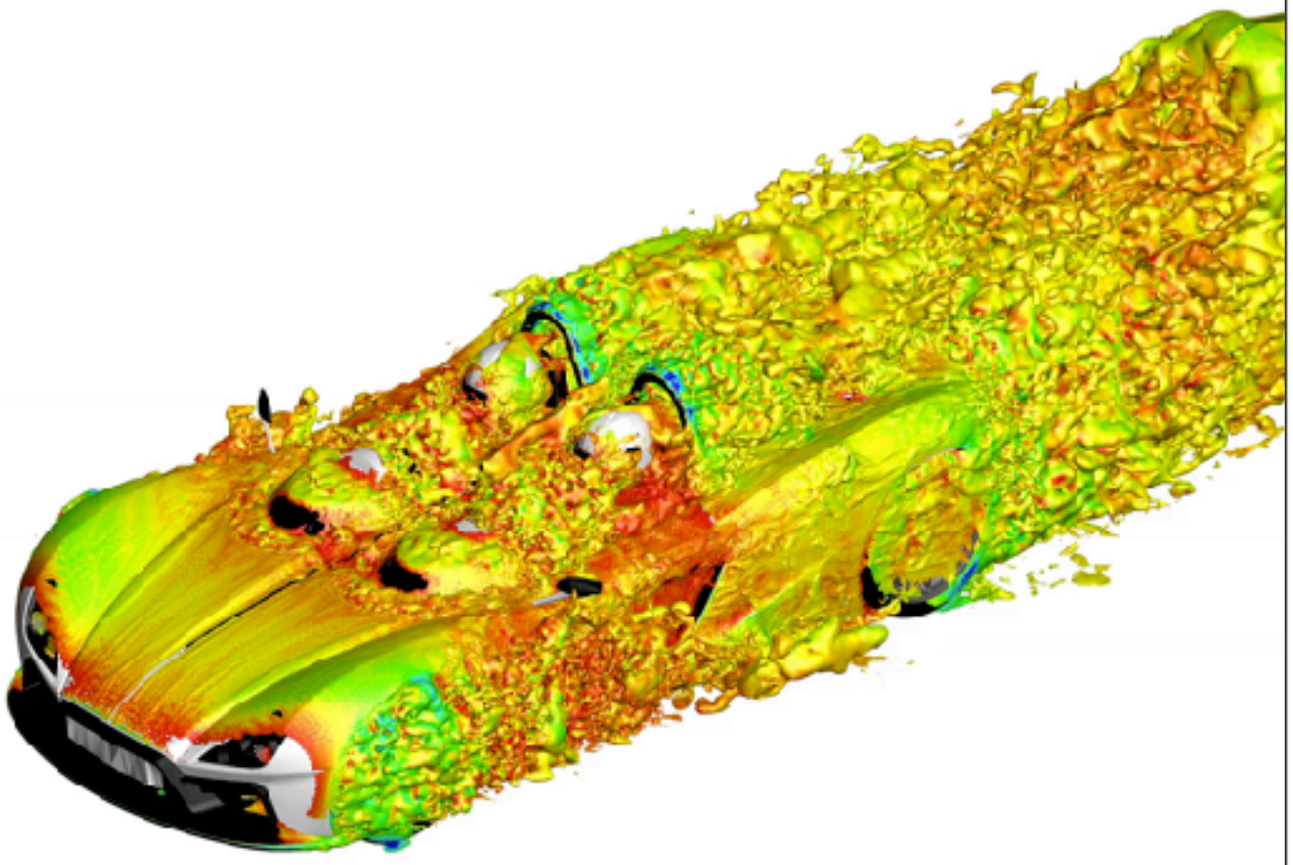
- In-house tools

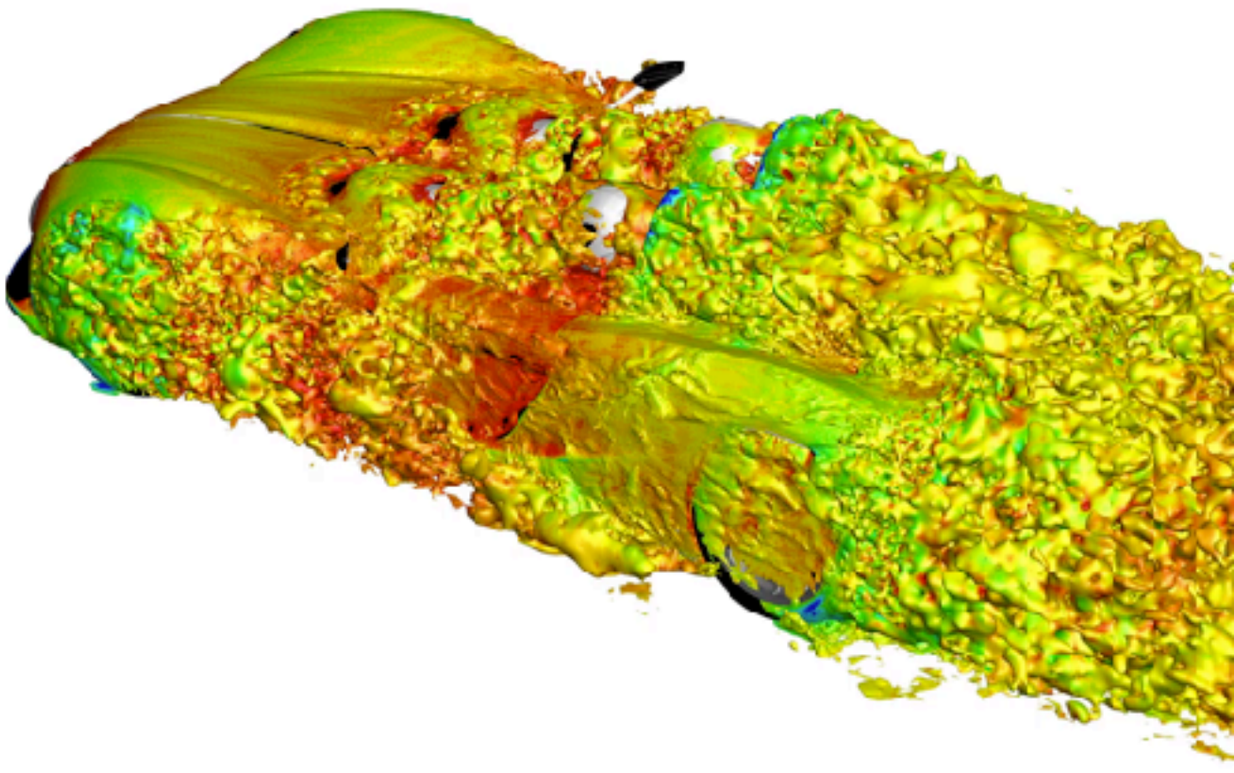
- Nektar++ format
- Gmsh format
- +others?

Road car

$P = 4$

Simulations





$P = 5$ around 1000 million dof

BL through Star-CCM+ Re = 50k

Mesh optimisation/correction

PDE solutions

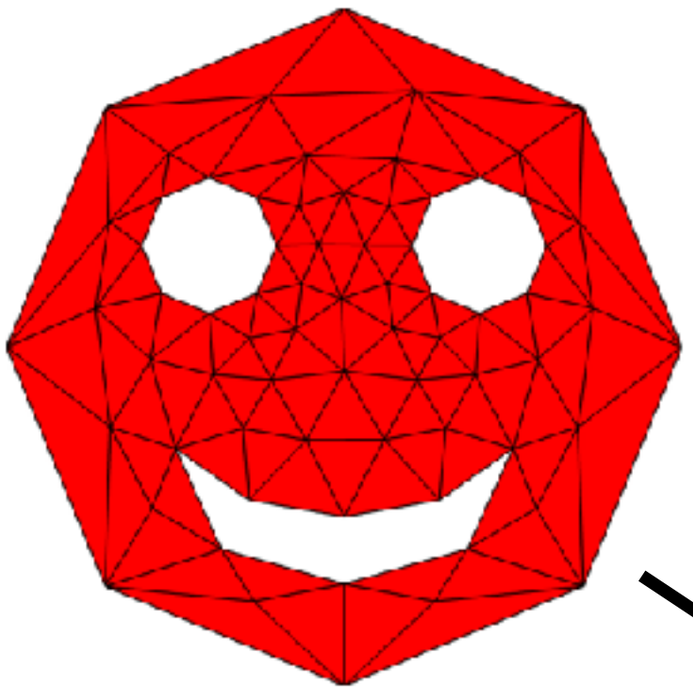
- Non-linear elasticity (Persson & Peraire 2009)
- Linear elasticity (Xie et al 2013; Hartmann & Leicht 2015)
- Thermo-elasticity (Moxey et al 2015)
- Winslow (Fortunato & Persson 2016)

Direct optimisation

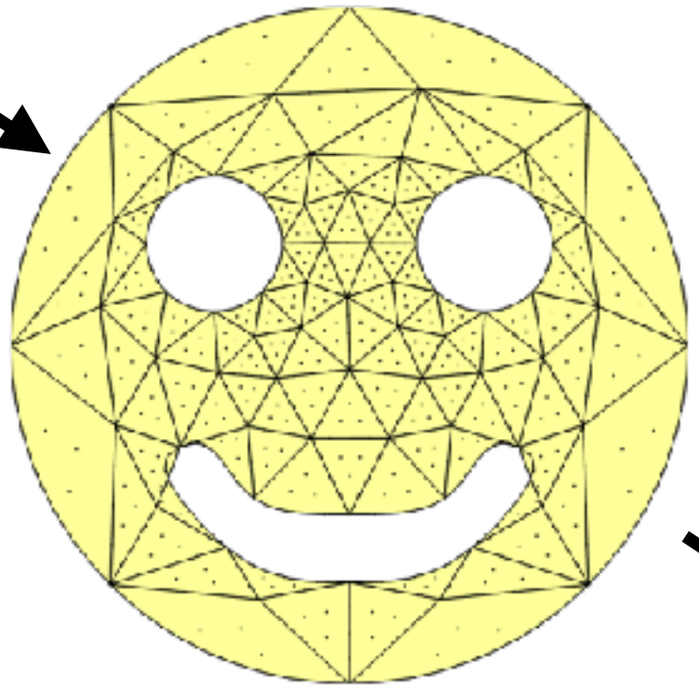
- Log barrier optimisation (Toulorge et al 2013)
- Distortion metric (Roca et al 2014)

Optimisation & correction

Straight-sided mesh



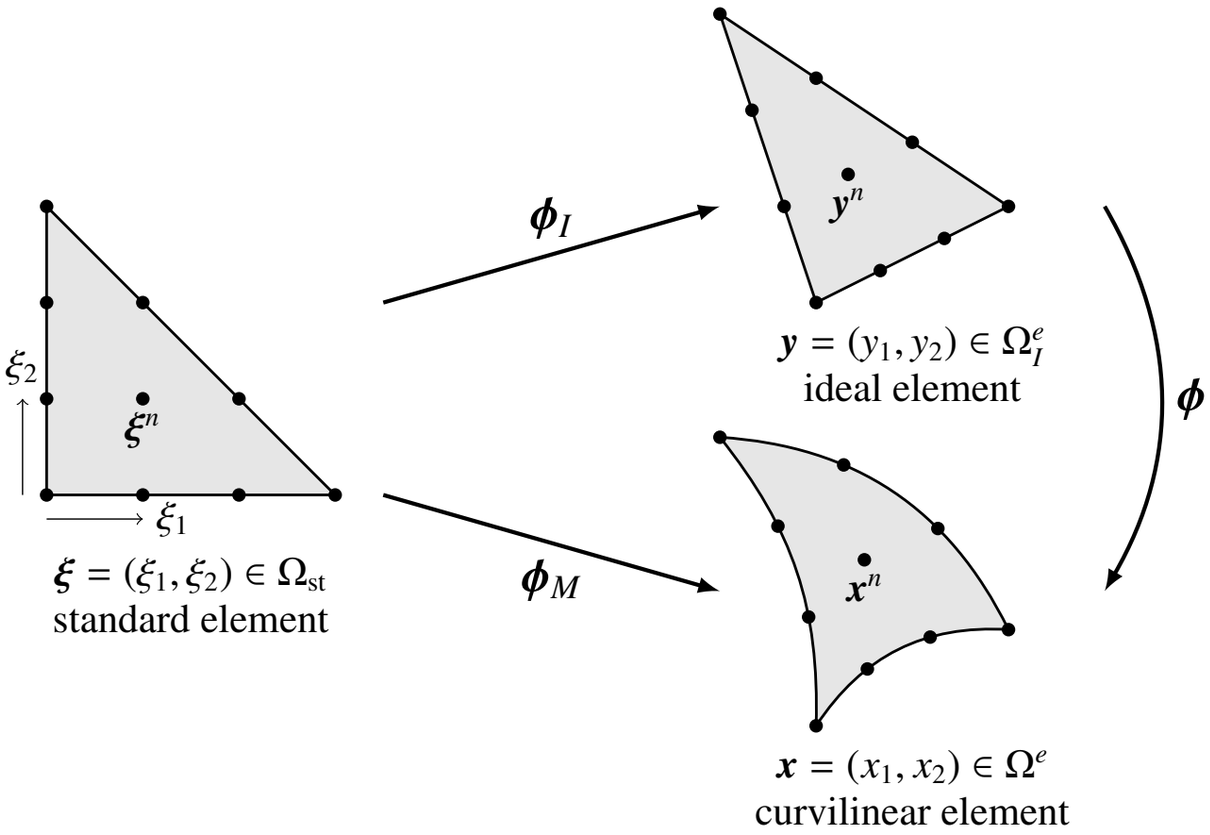
Boundary projection



Deformed mesh

ϕ

Variational approach



Recast PDE as energy minimisation and solve

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

Different W give PDE and optimisation methods in a single framework

M. Turner, J. Peiró, D. Moxey, *Curvilinear mesh generation using a variational framework*, Computer Aided Design **103** 73-91 (2018)

Choice of functional

$$\mathsf{F} = \nabla\phi \qquad J = \det \mathsf{F}$$

- Linear elasticity: $W = \frac{\kappa}{2}(\ln J)^2 + \mu\,\mathbf{E} : \mathbf{E}; \quad \mathbf{E} = \frac{1}{2}(\mathbf{F}^t\mathbf{F} - \mathbf{I})$
- Non-linear elasticity: $W = \frac{\mu}{2}(\mathbf{F} : \mathbf{F} - 3) - \mu \ln J + \frac{\lambda}{2}(\ln J)^2$
- Winslow: $W = J^{-1}\,(\mathbf{F} : \mathbf{F})$
- Distortion: $W = \frac{1}{d}|J|^{-d/2}(\mathbf{F} : \mathbf{F})$

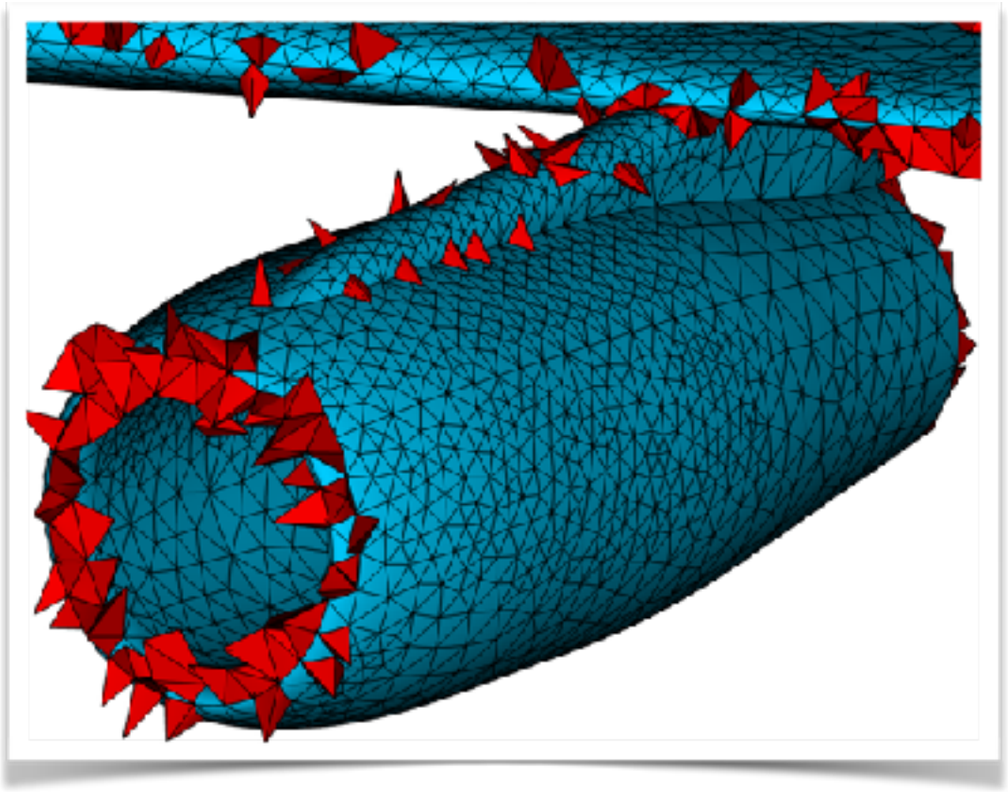
Solving the minimisation

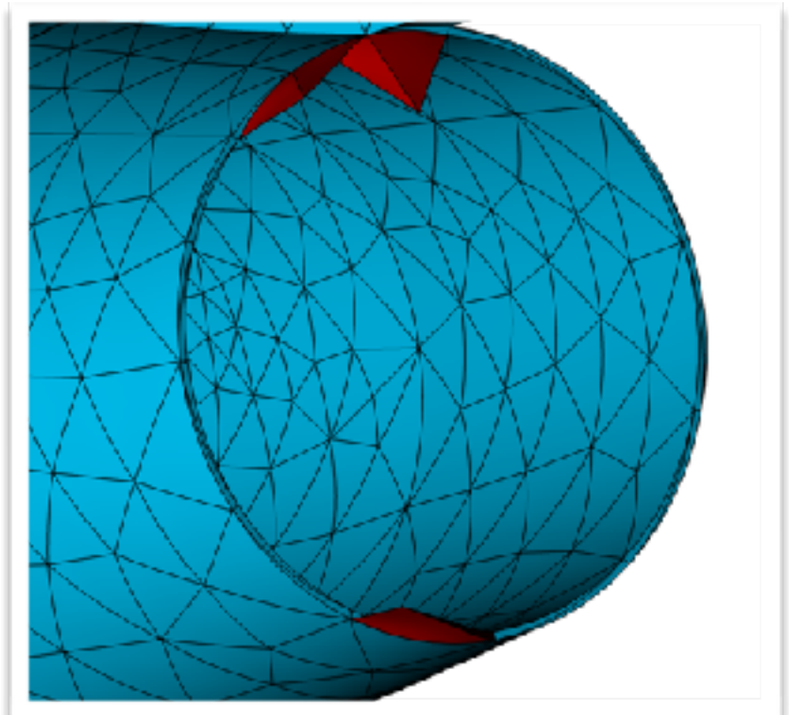
- Need to solve non-linear optimisation, where at each stage we move the mesh nodes. Can choose
  - **Global:** move all the nodes of the mesh at the same time
  - **Local:** split into lots of local problems for each node (relaxation)

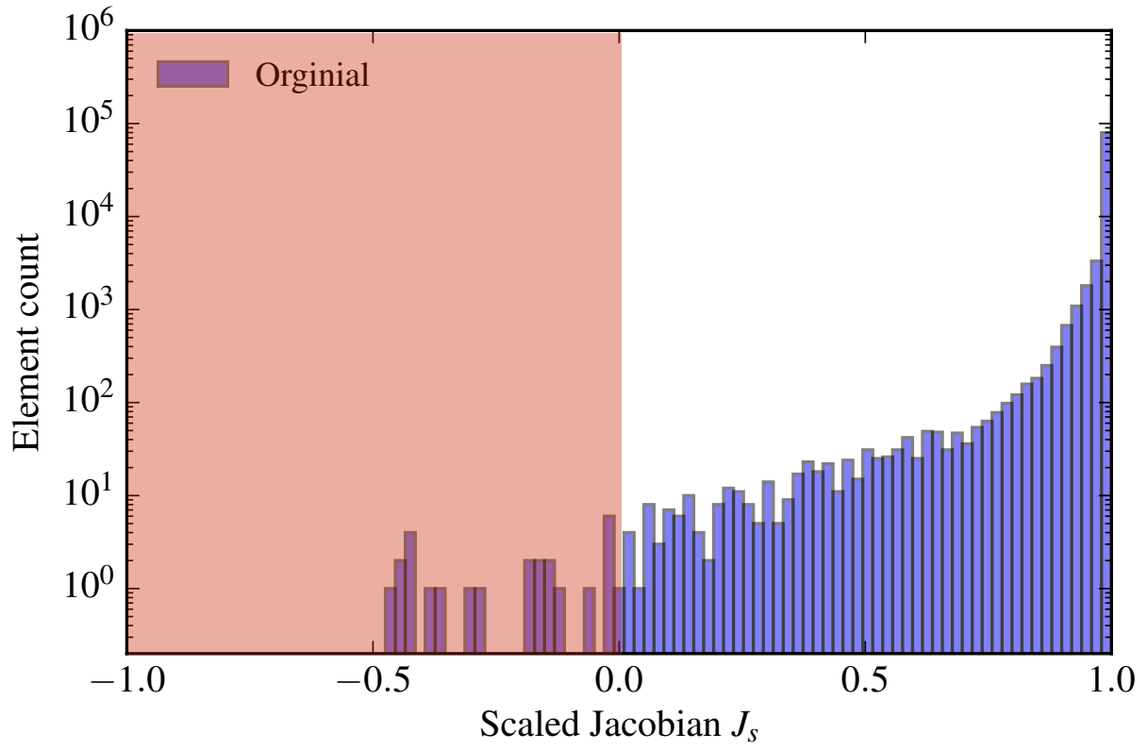
- Global approach needs nonlinear solver + large matrix solves per step but more robust

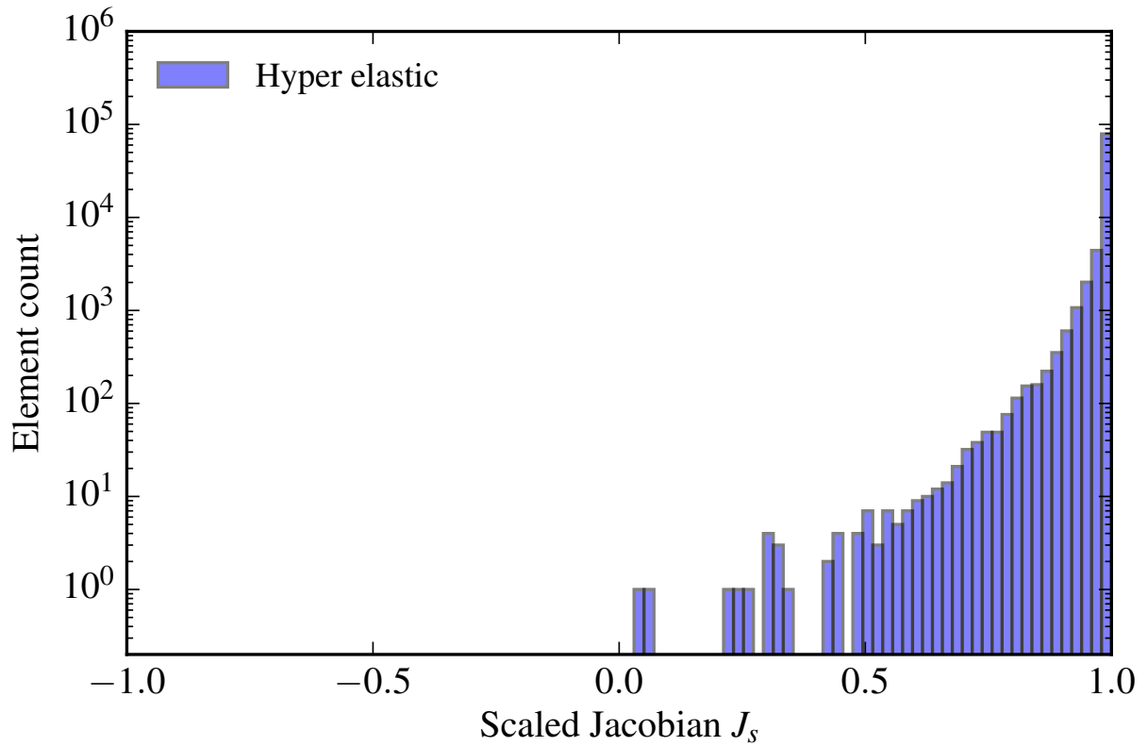
- Local approach is much cheaper, maybe less robust + more iterations, but extremely parallelisable

Example: DLR F6 engine





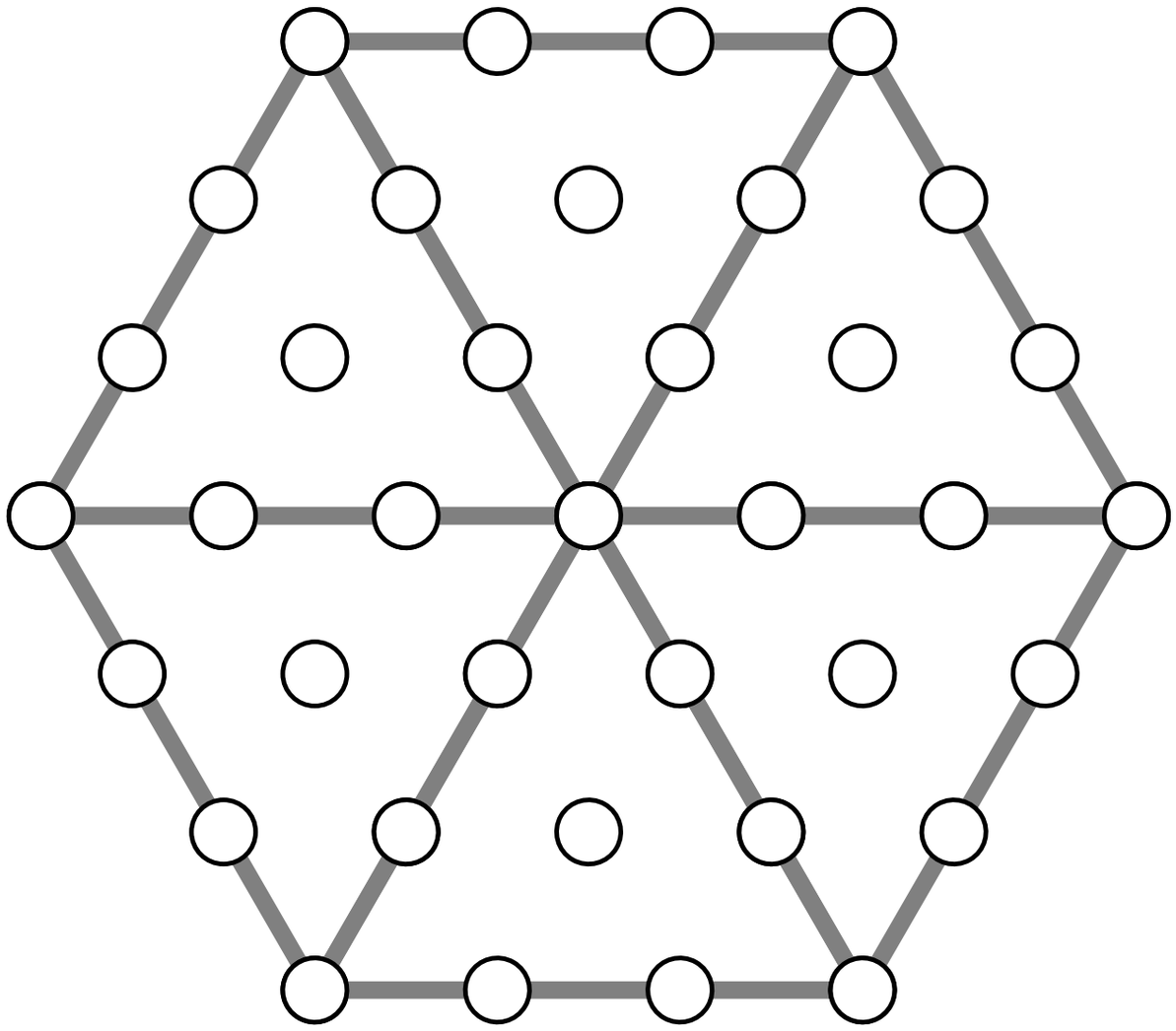




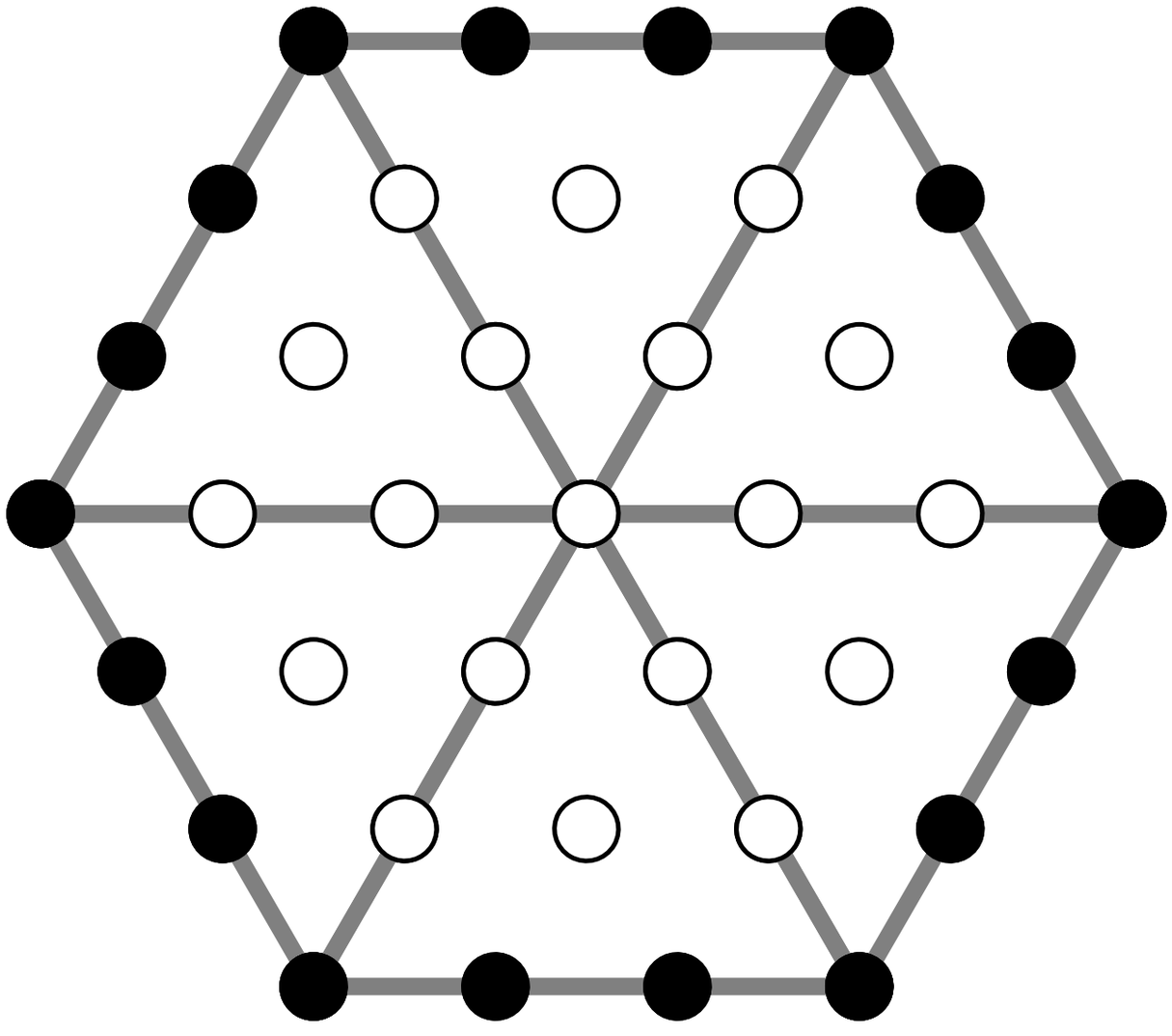
Speeding up optimisation

- Meshing usually accomplished on a single workstation, require many design iterations
- Optimisation process is resource intensive
- GPUs have lots of compute density
- Can we leverage parallelism of the method effectively on a GPU?
- How do we do this in a code-friendly way?

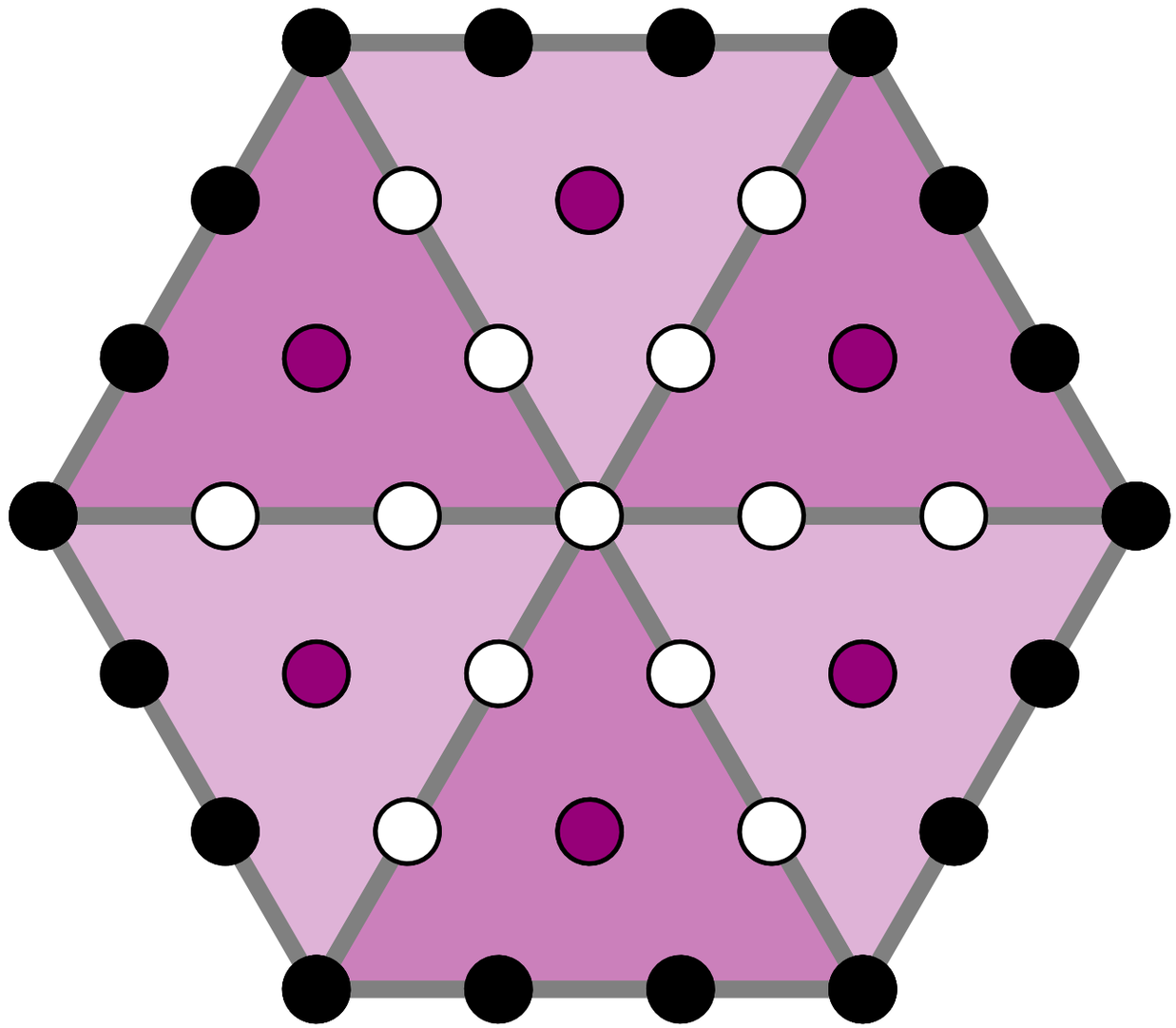
Node colouring



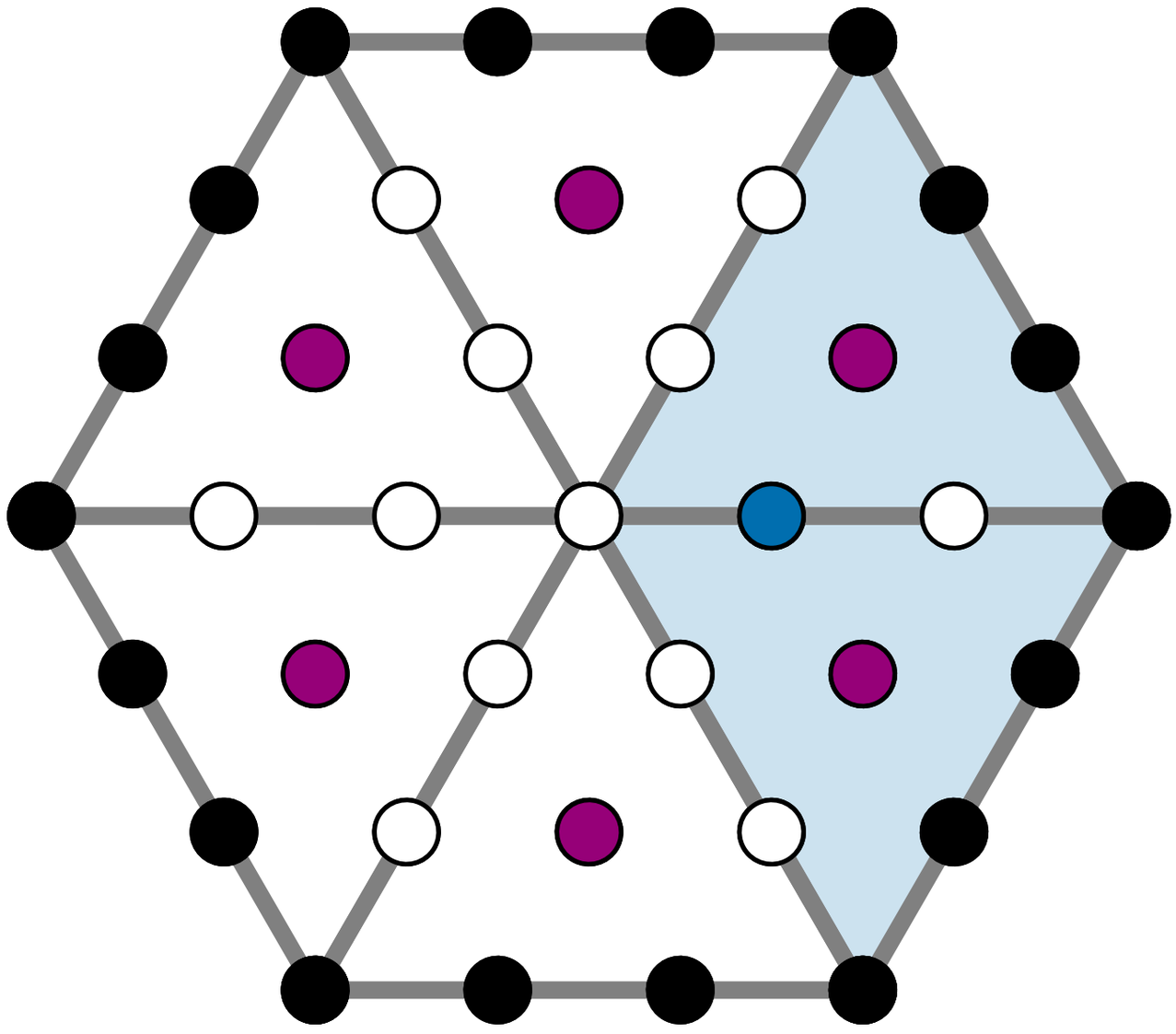
Node colouring



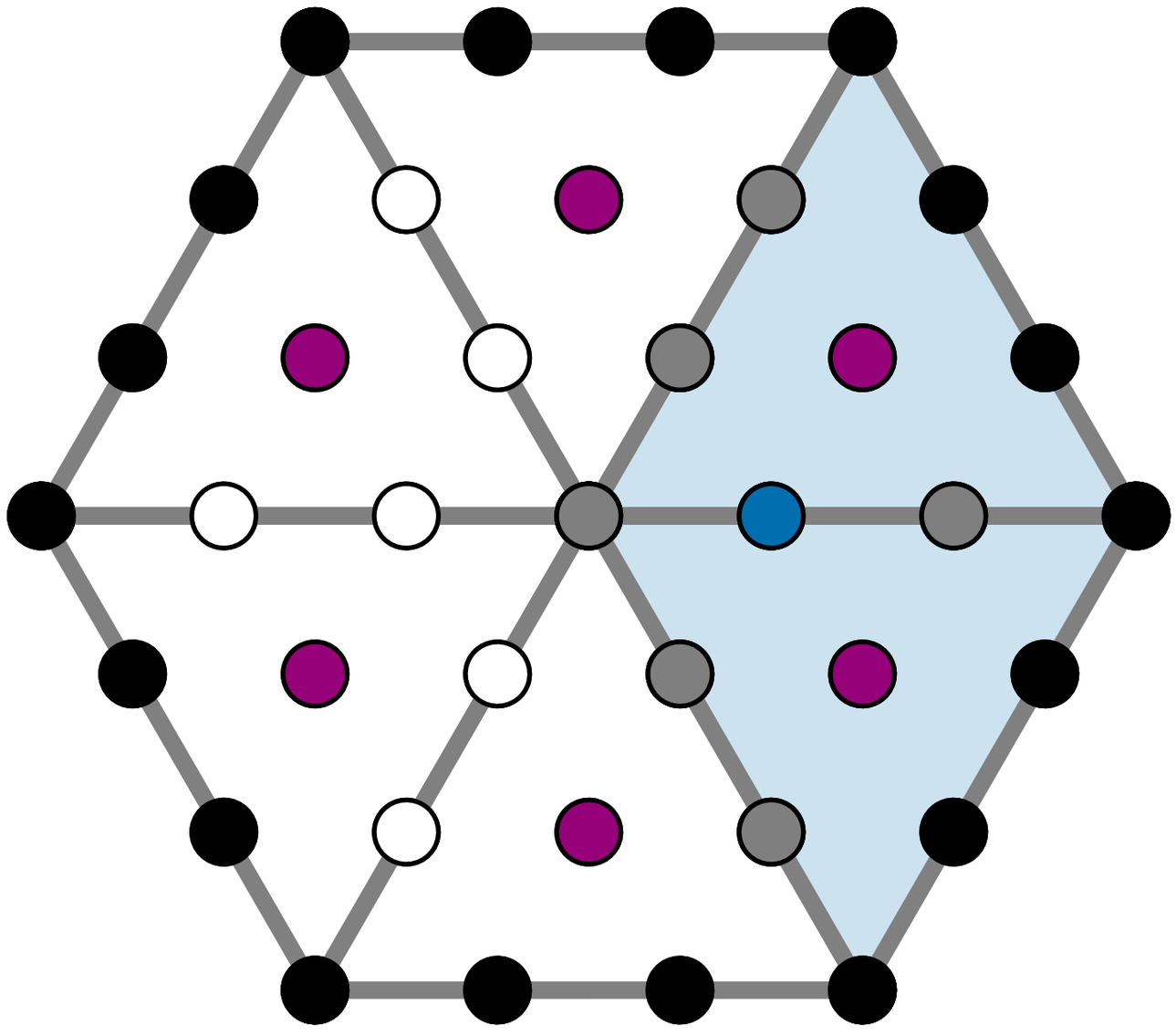
Node colouring



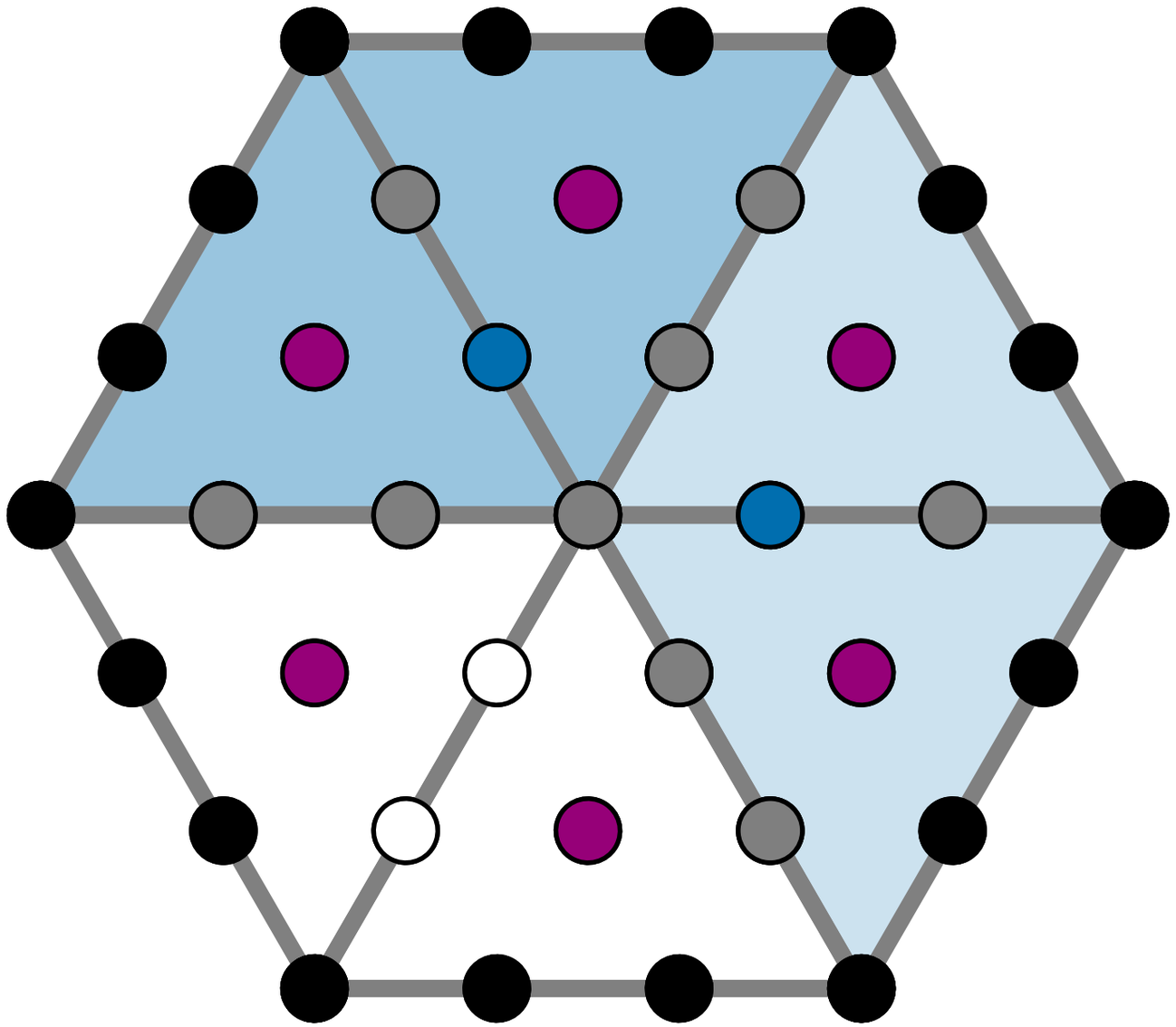
Node colouring



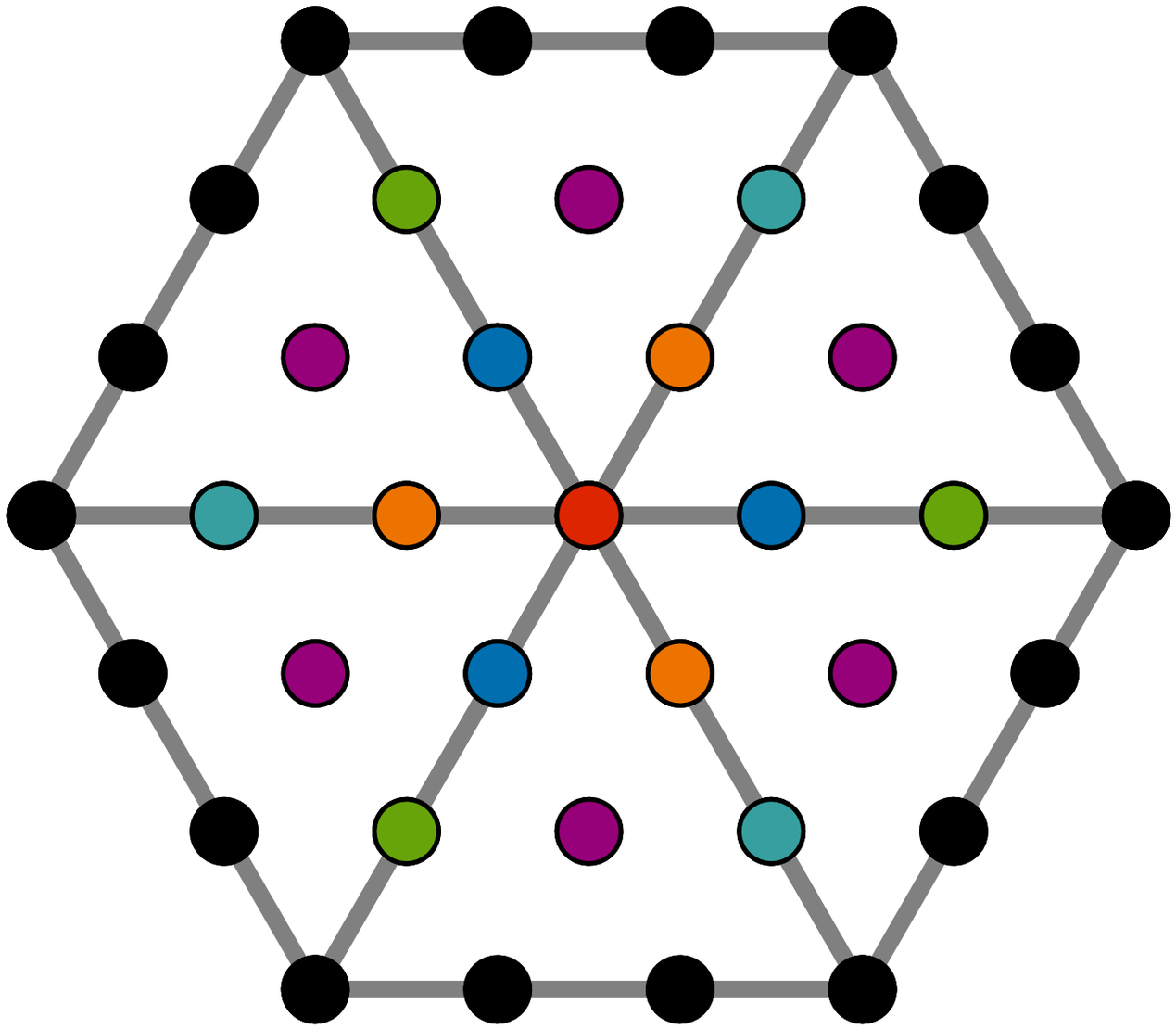
Node colouring



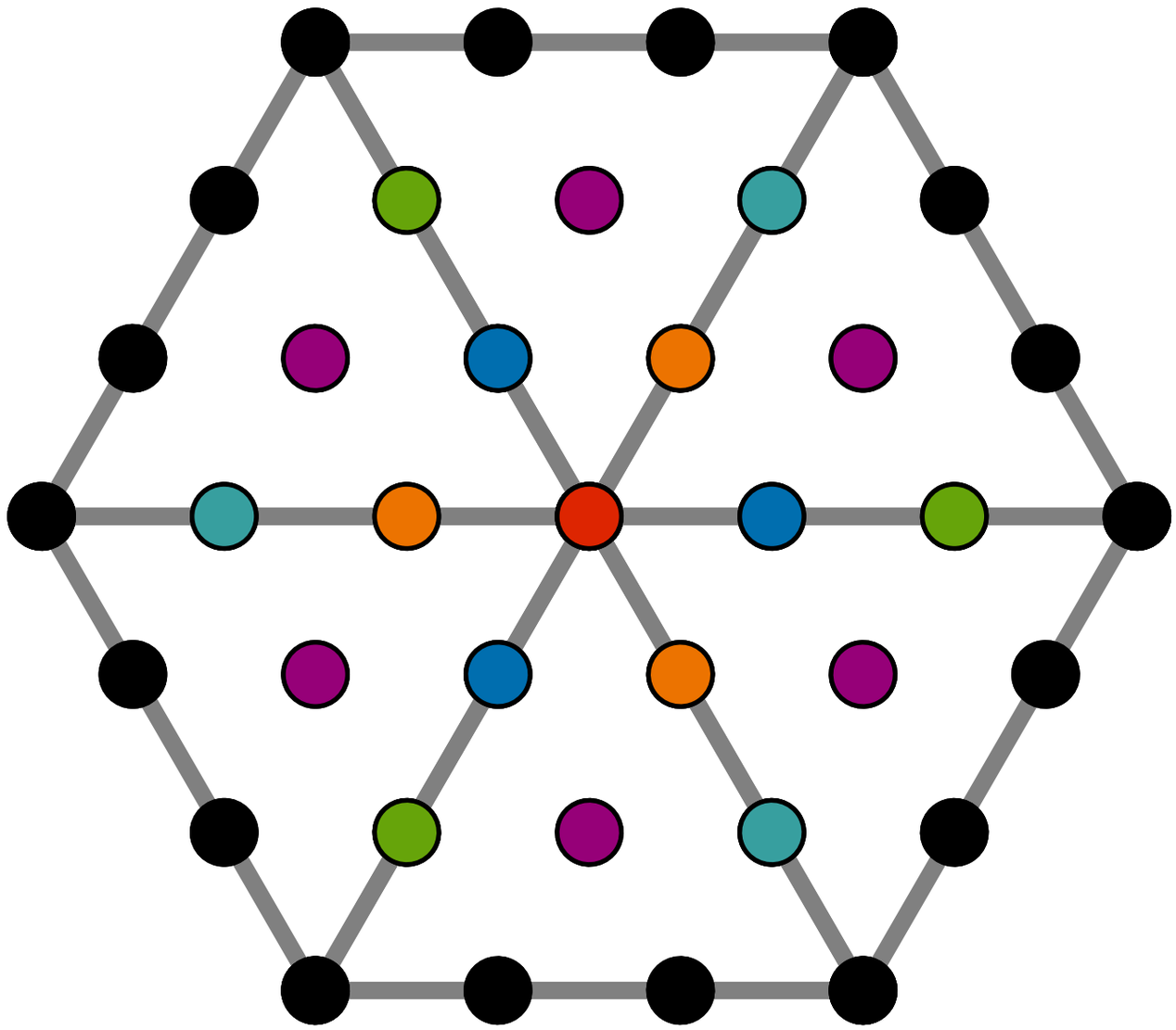
Node colouring



Node colouring



Node colouring



For each node solve local minimisation problem

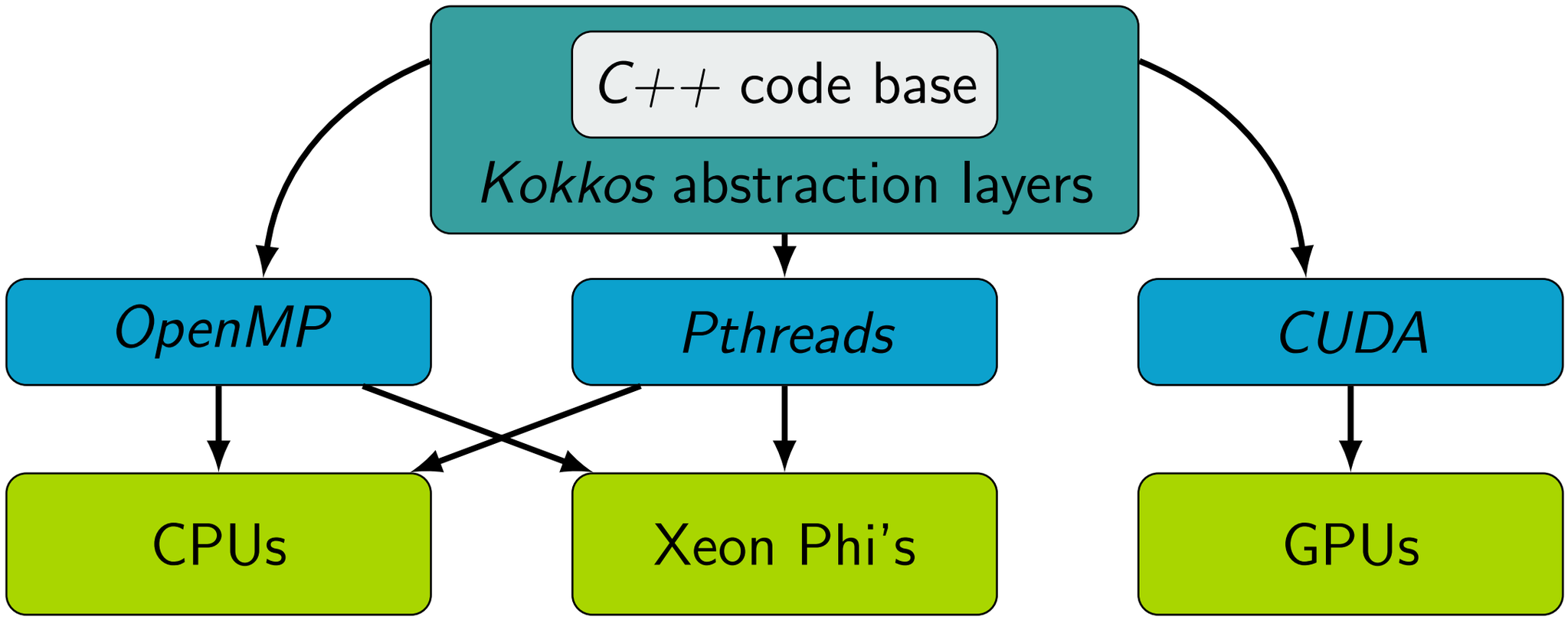
Calculate functional + gradients

Uses multi-level threading to exploit GPU hierarchy

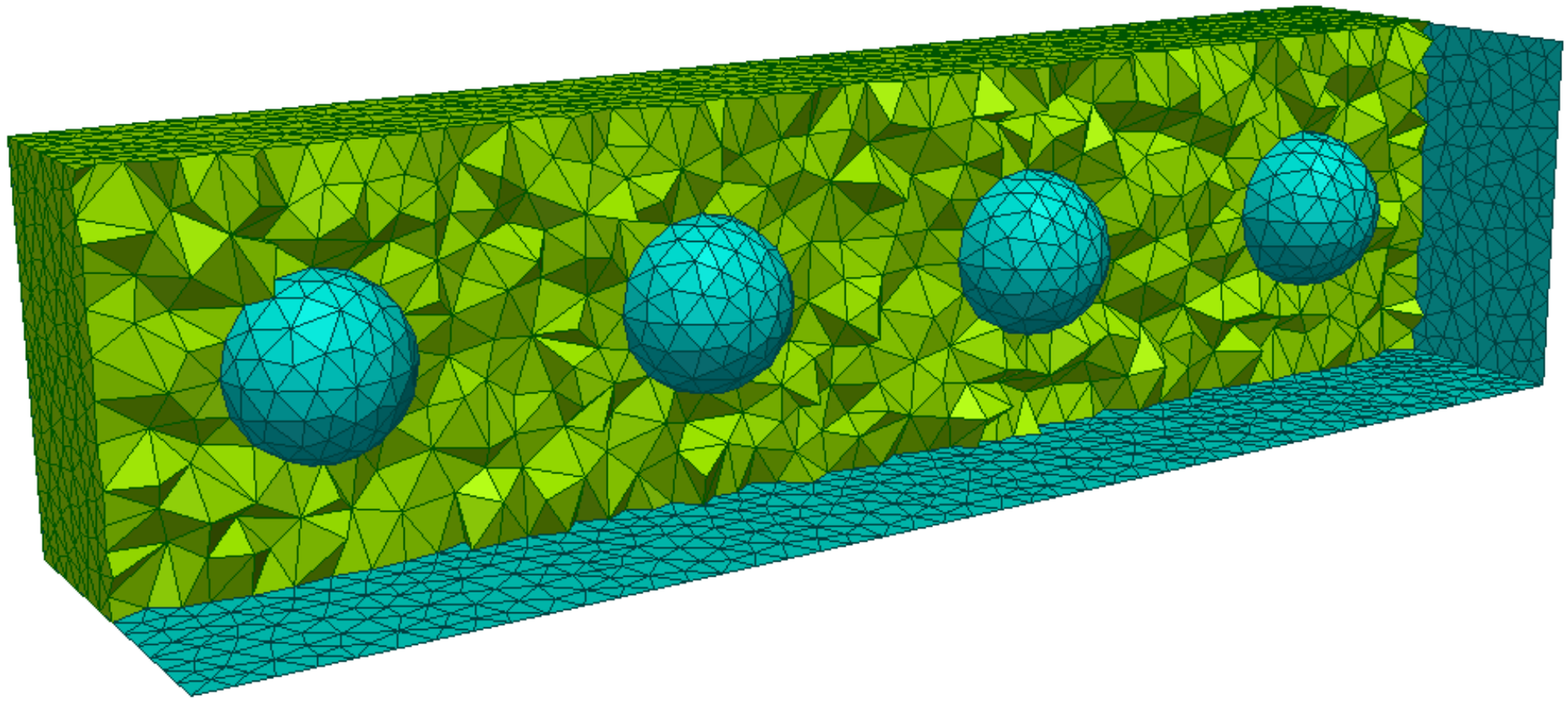
Iterate until global functional residual is small

Frameworks

Investigated use of Kokkos framework
to integrate with existing C++ infrastructure

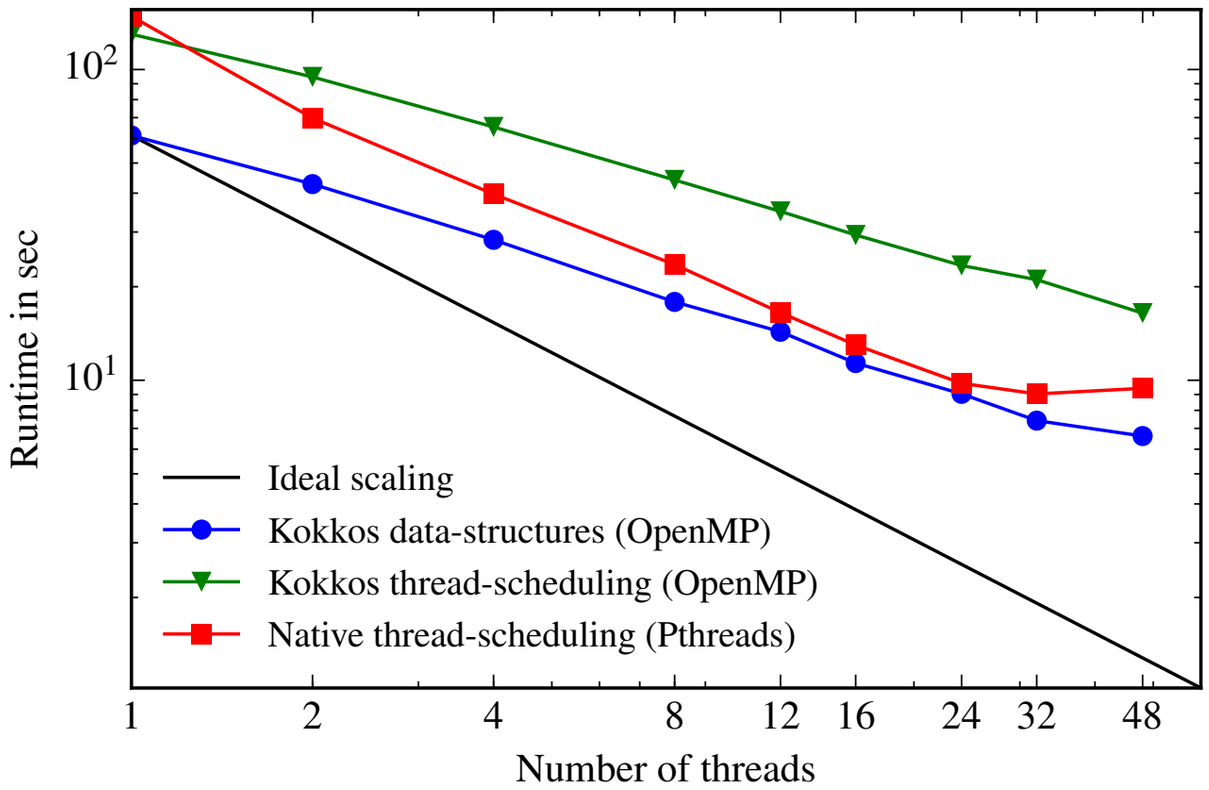


Results: CPU scaling

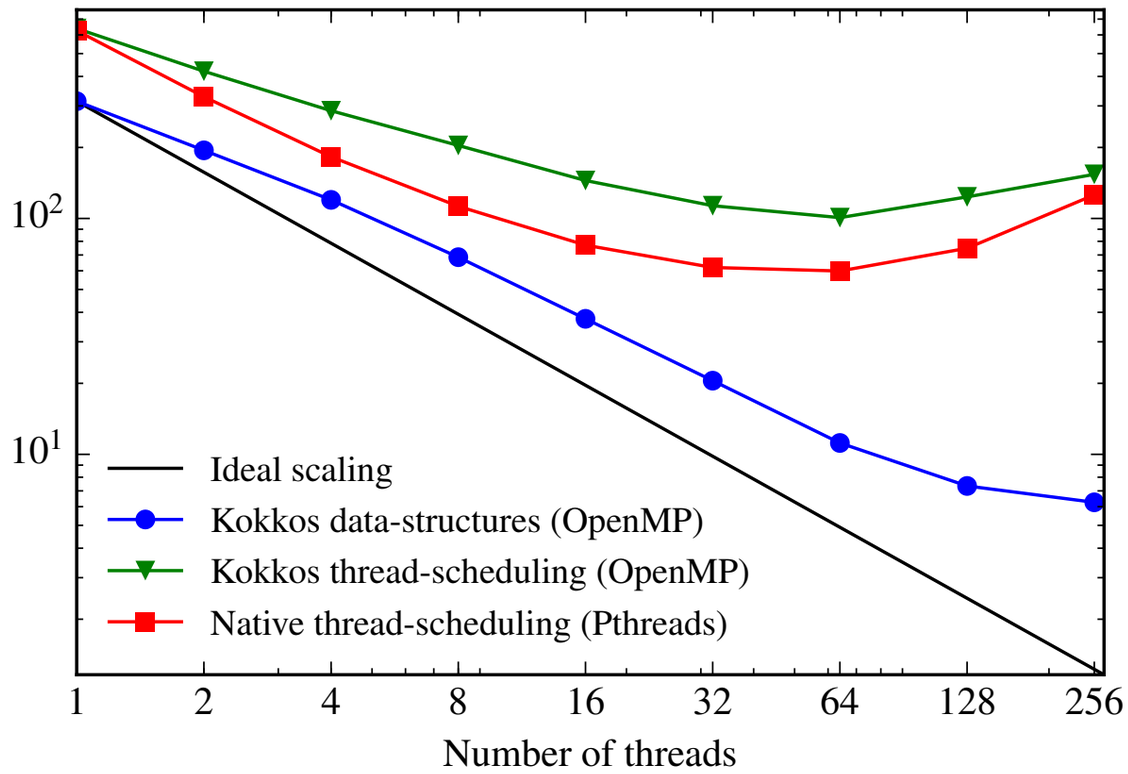


Four spheres in a box, 33k tetrahedra, ~400k nodes
at $p = 5$

Results: CPU scaling

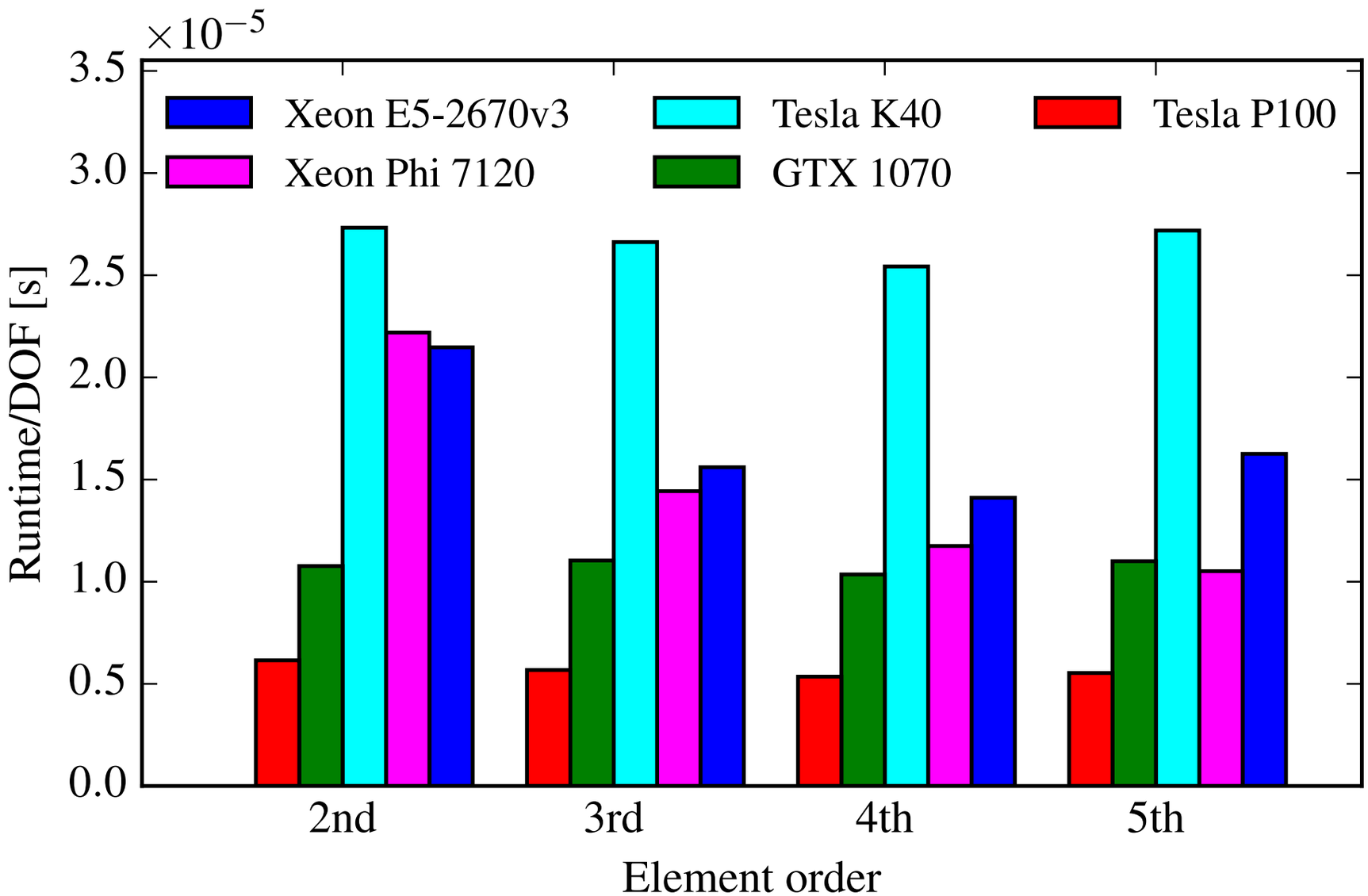


2 x Intel E5-2670v3
Haswell nodes

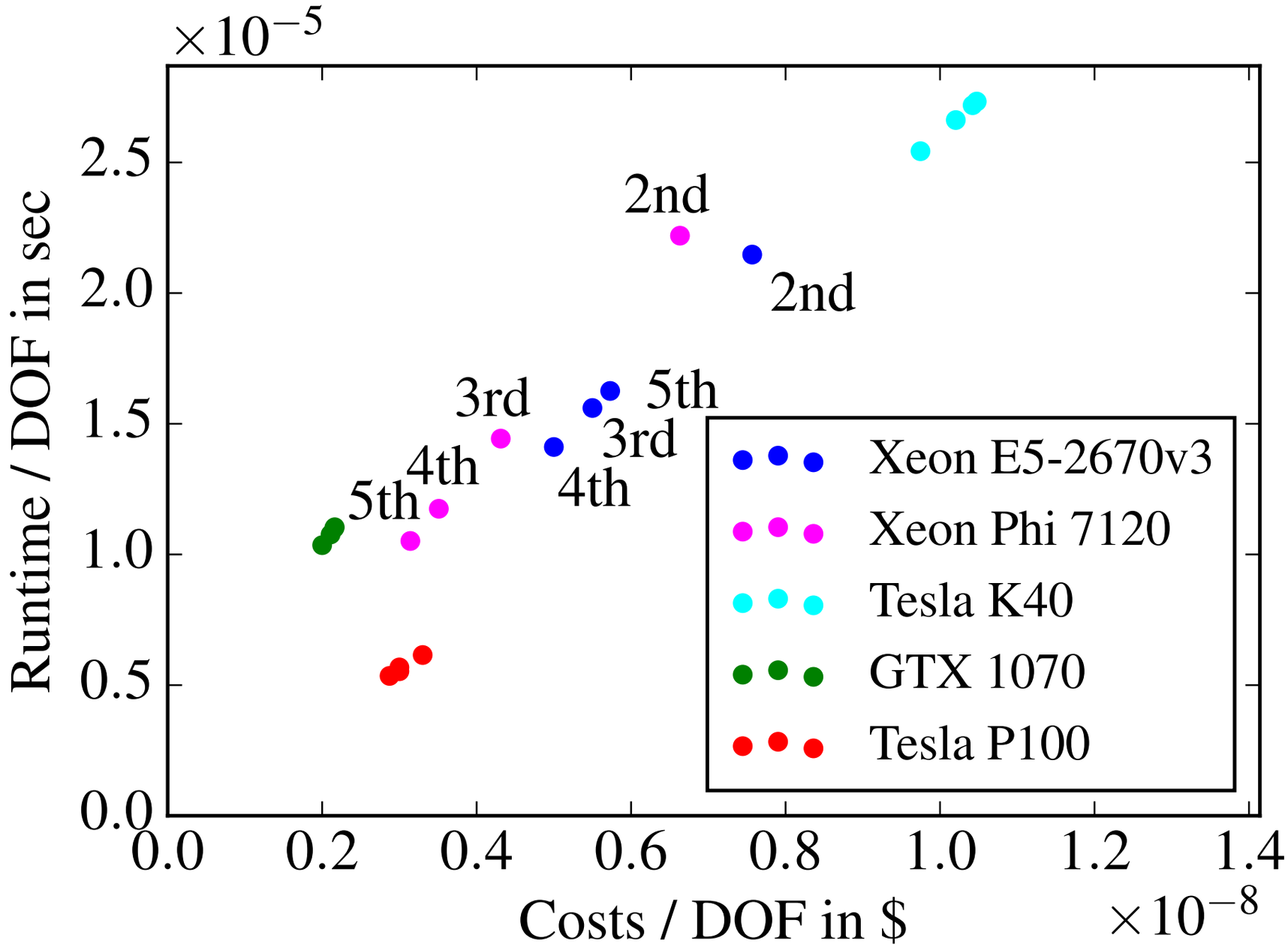


Xeon Phi 7210
KNL node

Results: many-core



Results: many-core



Lessons learned

- Good performance on CPU and GPU
- Execution cost and time savings to GPU
- Significant code reworking to get functioning efficiently
- Potential for thread scheduling improvements in Kokkos to enhance performance on CPU
- Kokkos certainly not the only choice

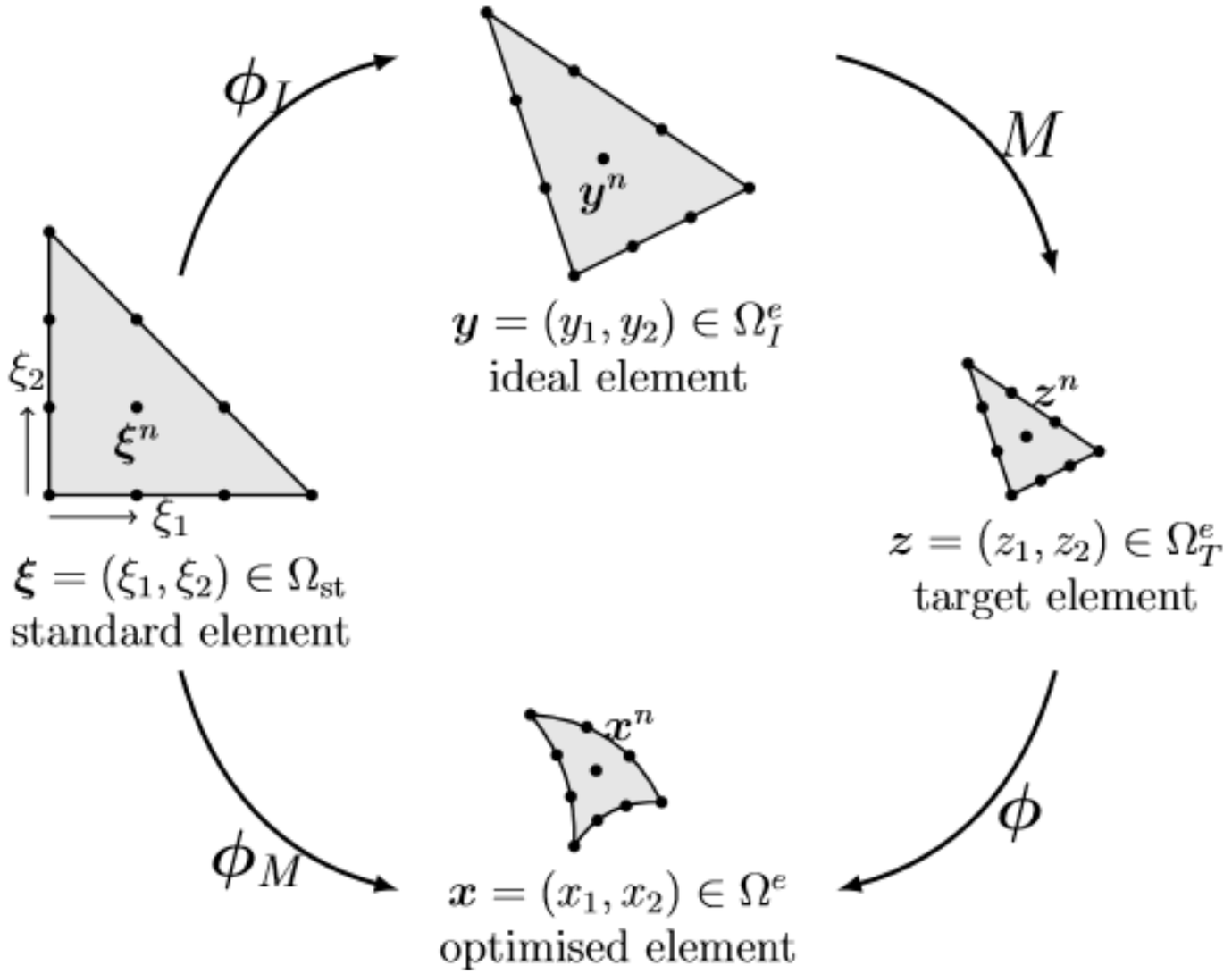
Towards solver interaction

- A key goal of NekMesh is to bring CAD and solver closer together
- Enable dynamic mesh movement even for complex geometries
- Small wrapper around CAD to enable different CAD systems
- Example application: shock capturing

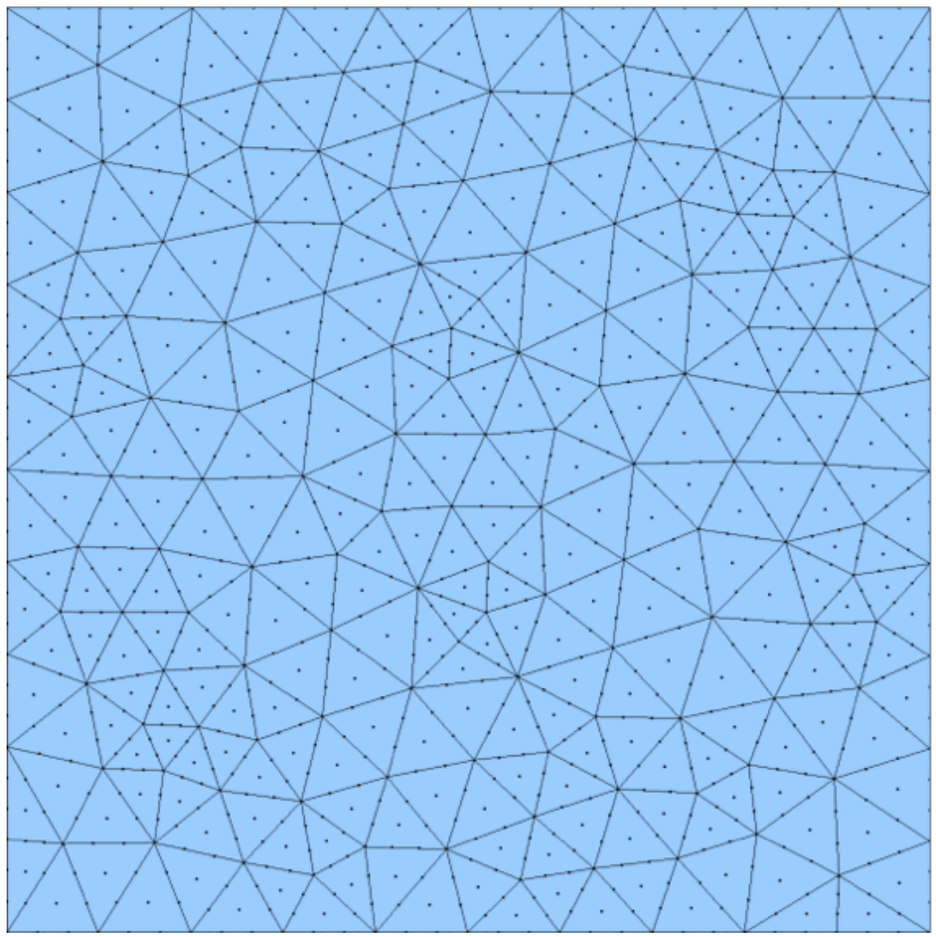
Towards solver interaction

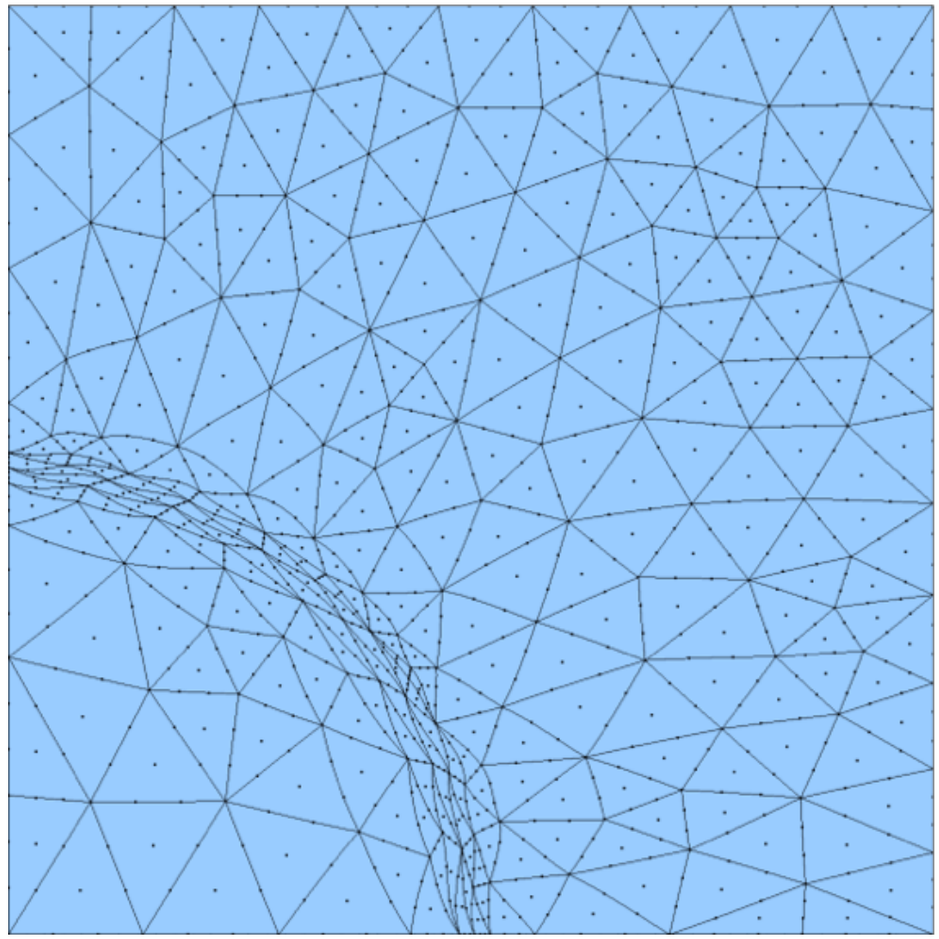
- Want to combine:
  - *r*-adaptation (moving mesh) to resolve shock
    - *p*-adaptation to add additional resolution where required
- Aim to achieve highly efficient compressible flow simulations with shocks
- Problem: how do we deal with complex geometries?

Adapting to element size

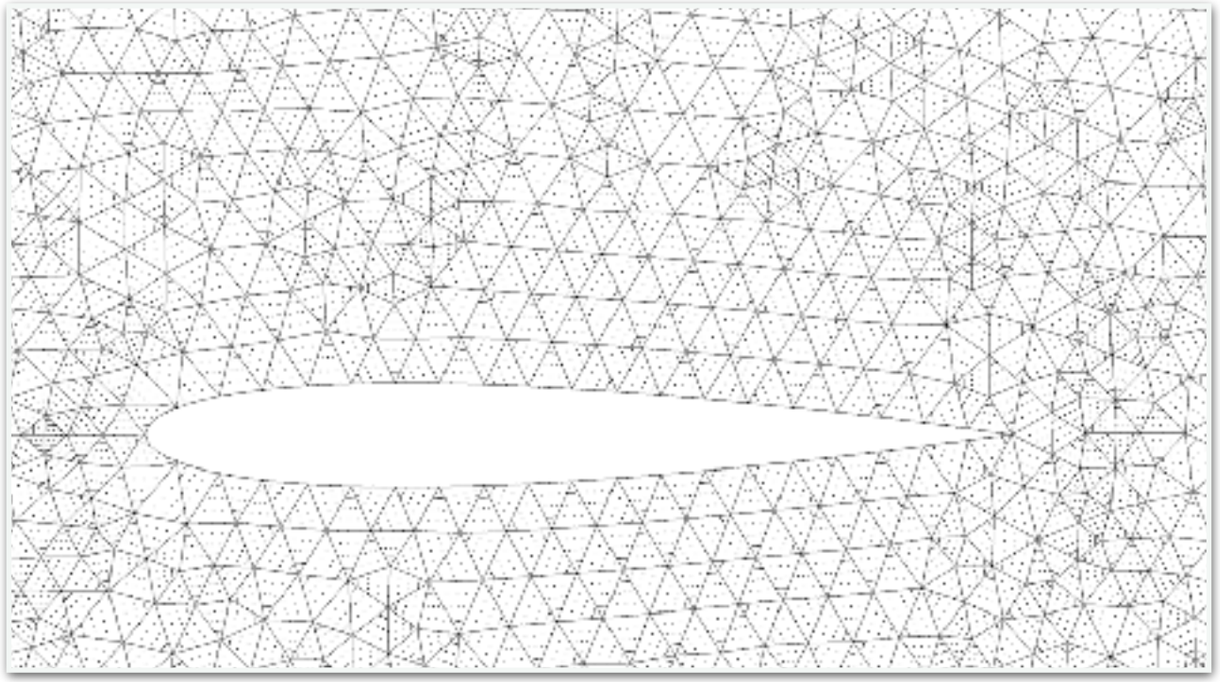


Adapting to element size

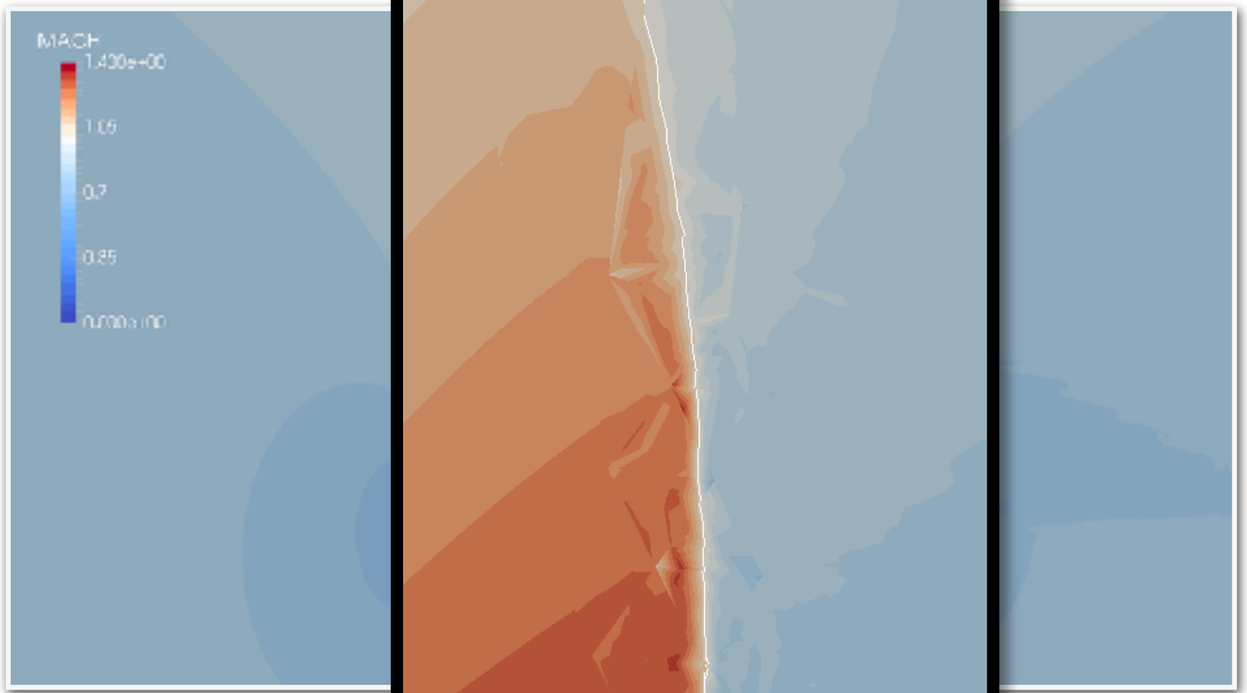




Results: NACA 0012 transonic

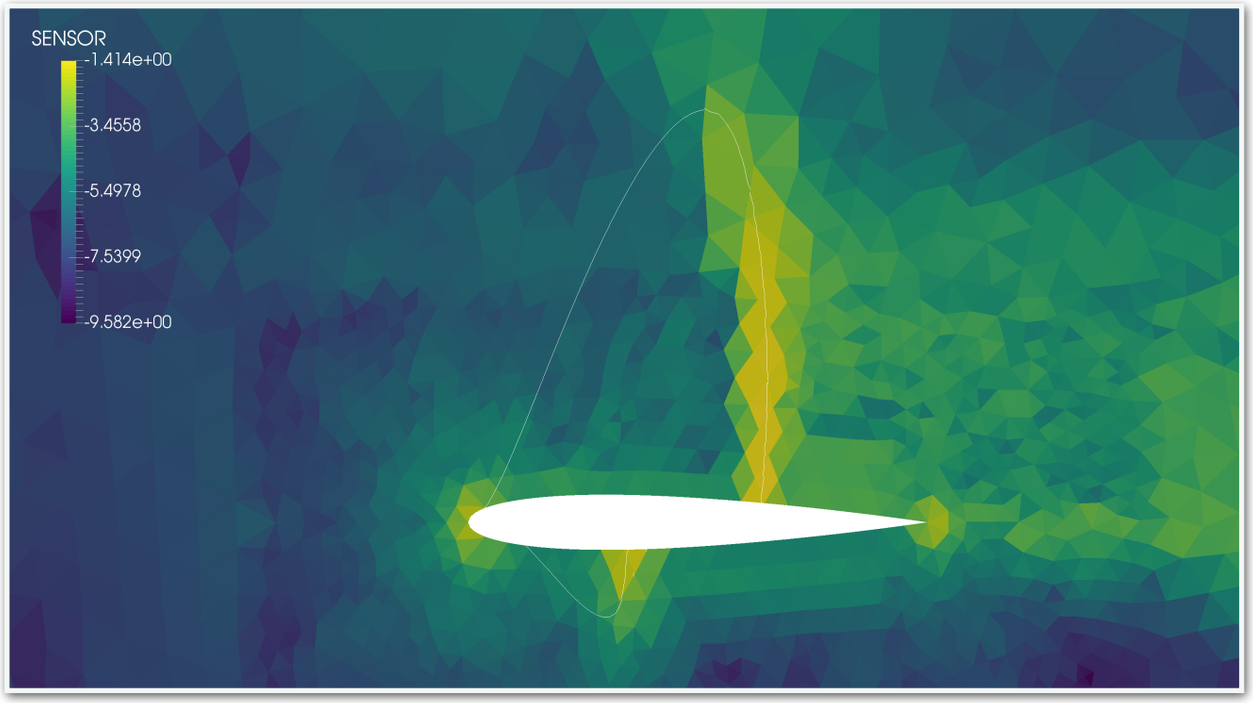


Starting mesh

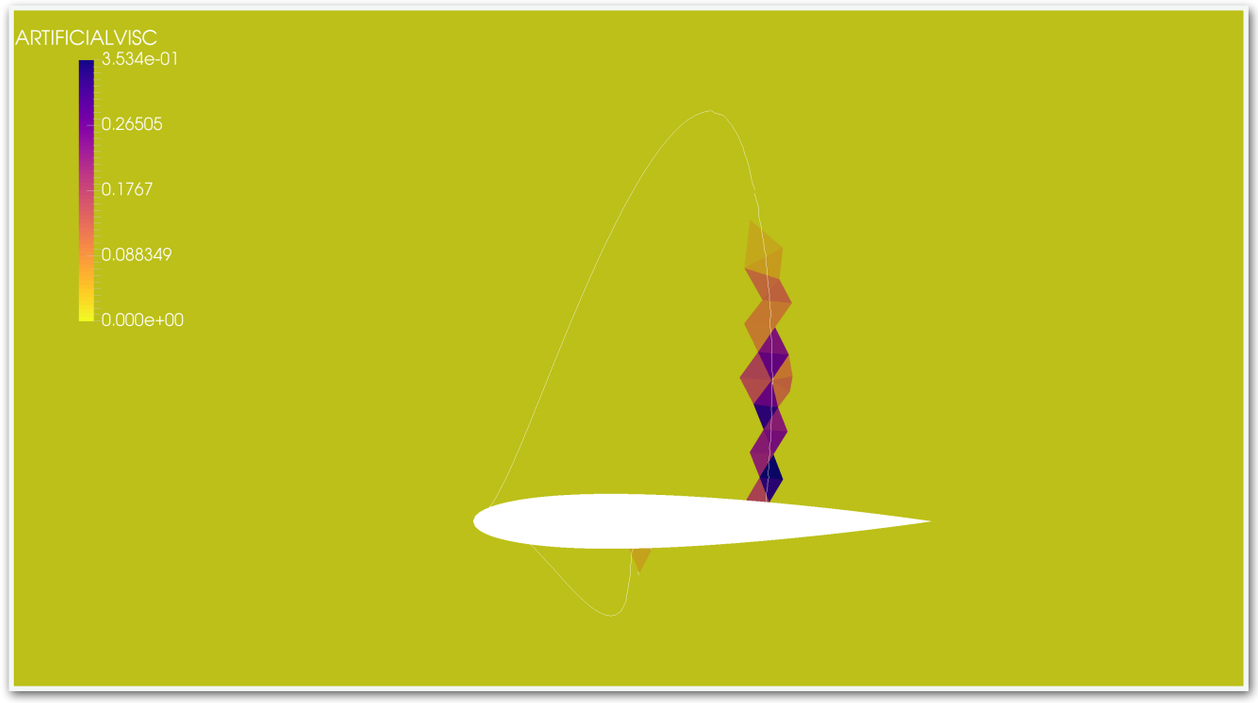


Ini

Results: NACA 0012 transonic

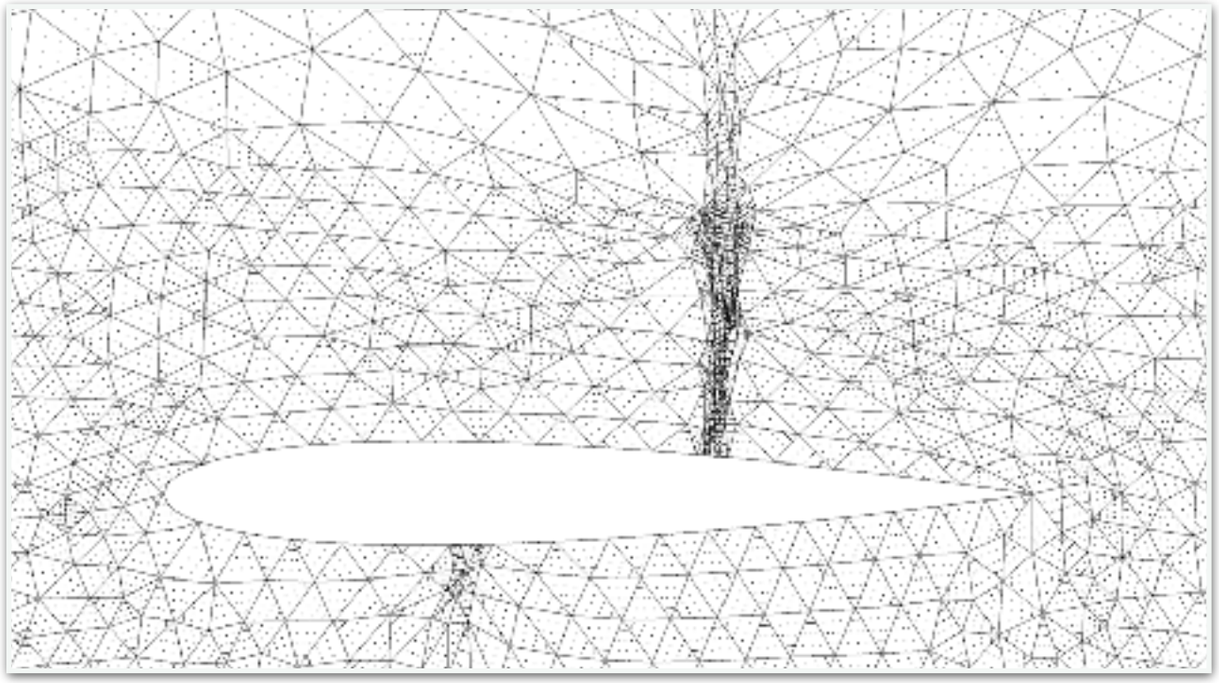


Discontinuity sensor

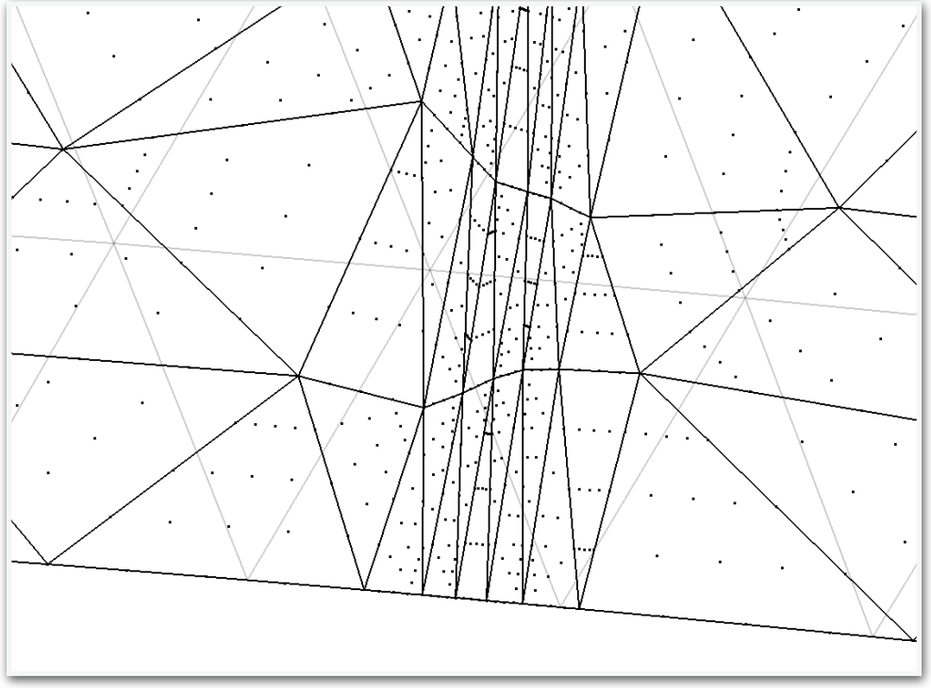


Artificial viscosity

Results: NACA 0012 transonic

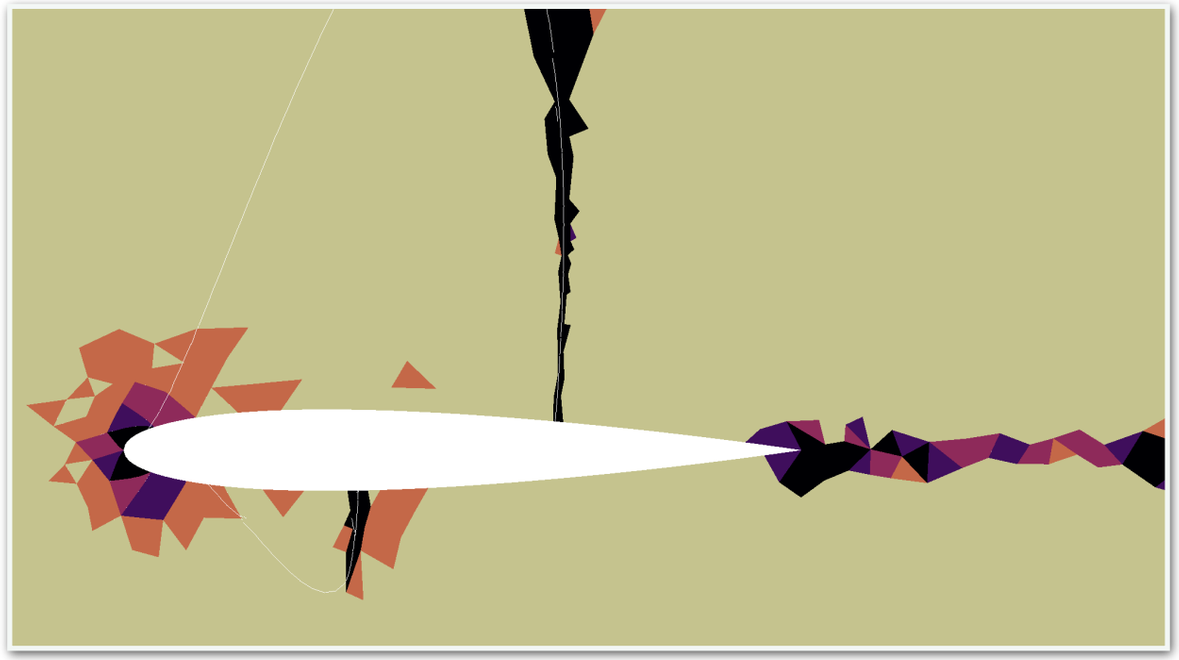


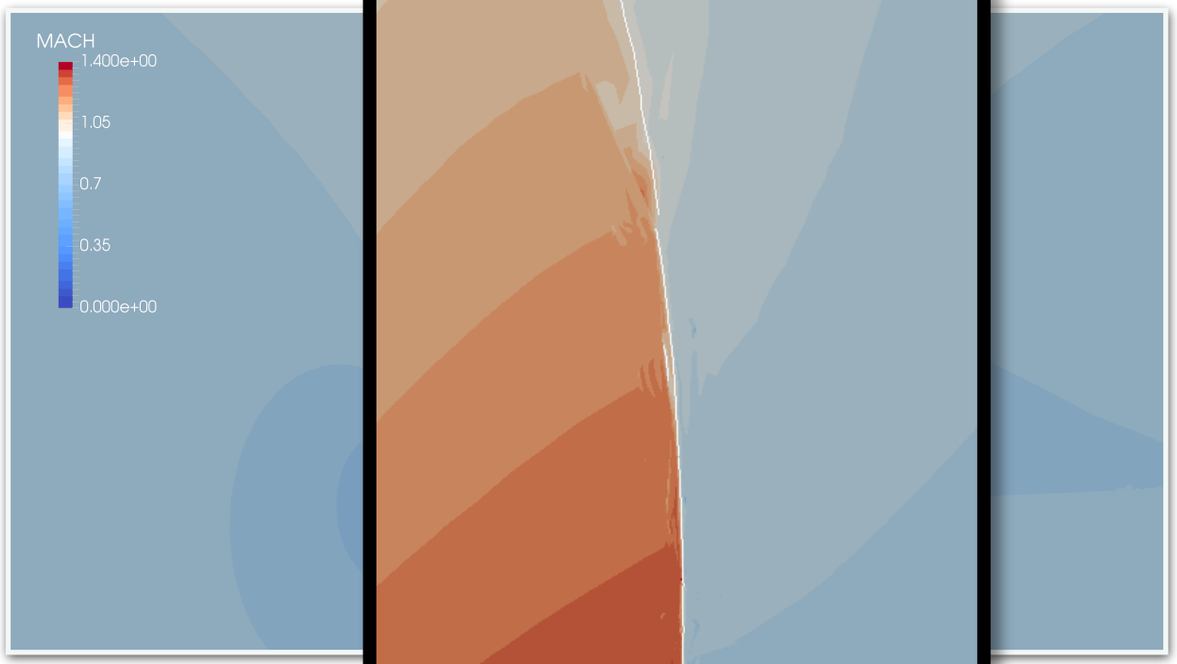
Calculate target size
& do r-adaptation



Use of CAD sliding

Results: NACA 0012 transonic





Translate to variable p

Improve ing

Summary

- Promising route to efficient computational performance on many-core architectures
- Careful thought required on architecting code
- Combination of *r* and *p* adaptation provides good accuracy at reduced degree of freedom
- CAD sliding crucial for dealing with complex geometries
- Lots to do!

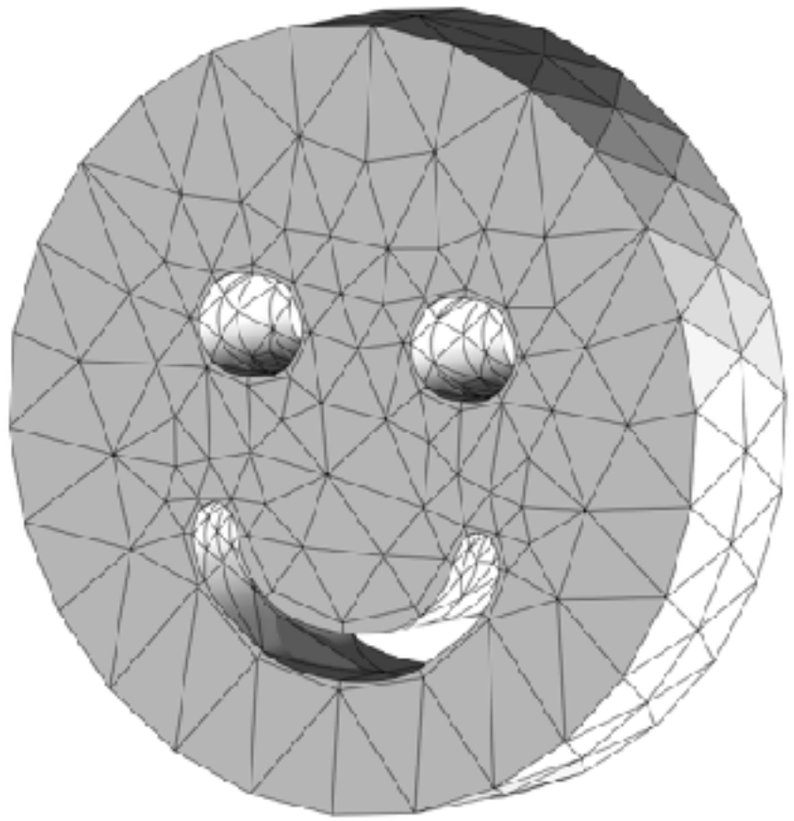
Other talks

Nektar++ talk in MS38

Today, 5:00-5:30pm (Room 200)

Chris Cantwell

Thanks for listening!



https://davidmoxey.uk/ @davidmoxey

d.moxey@exeter.ac.uk

www.nektar.info