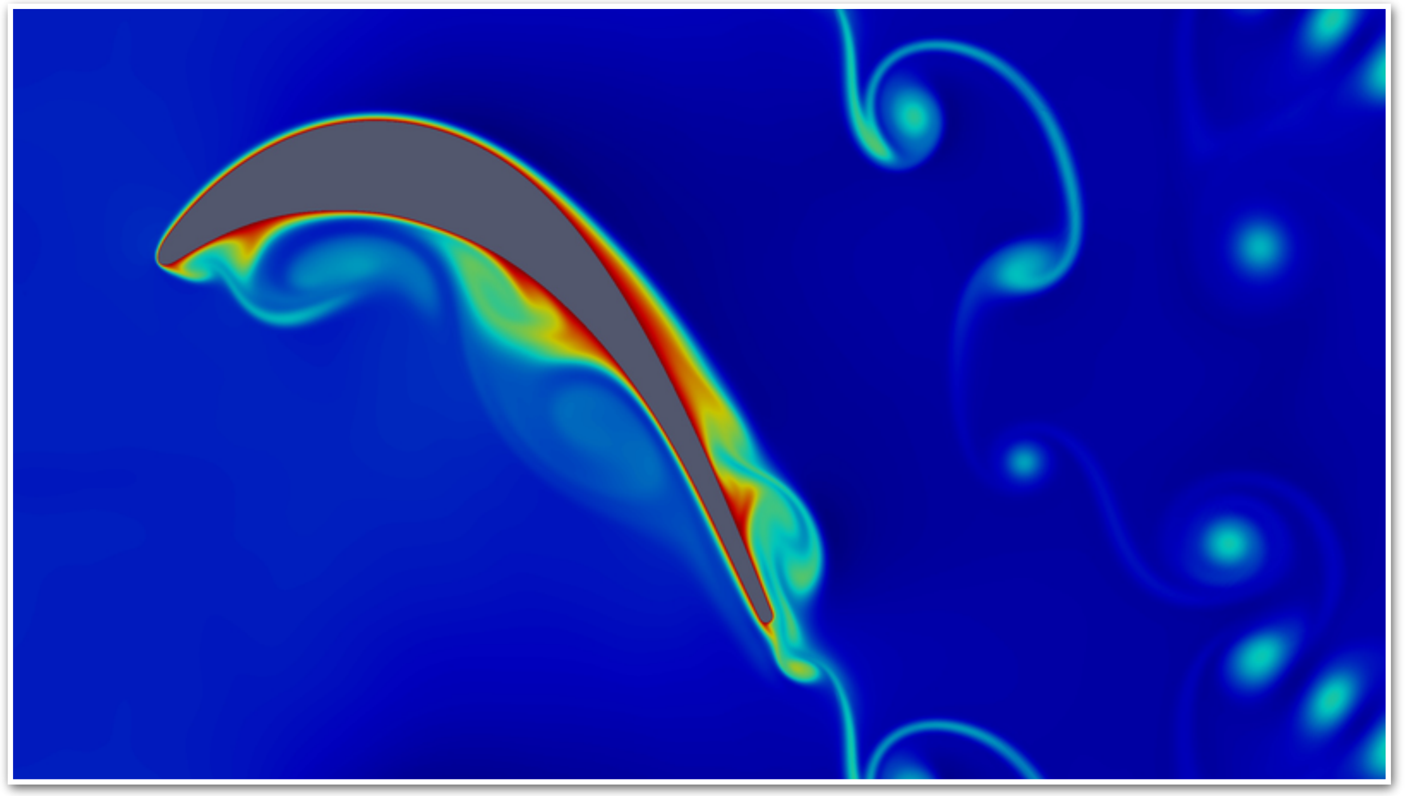
Interdisciplinary Research Computing Seminar, Imperial College London, 17th January 2022.

Tackling real-world problems with modern software

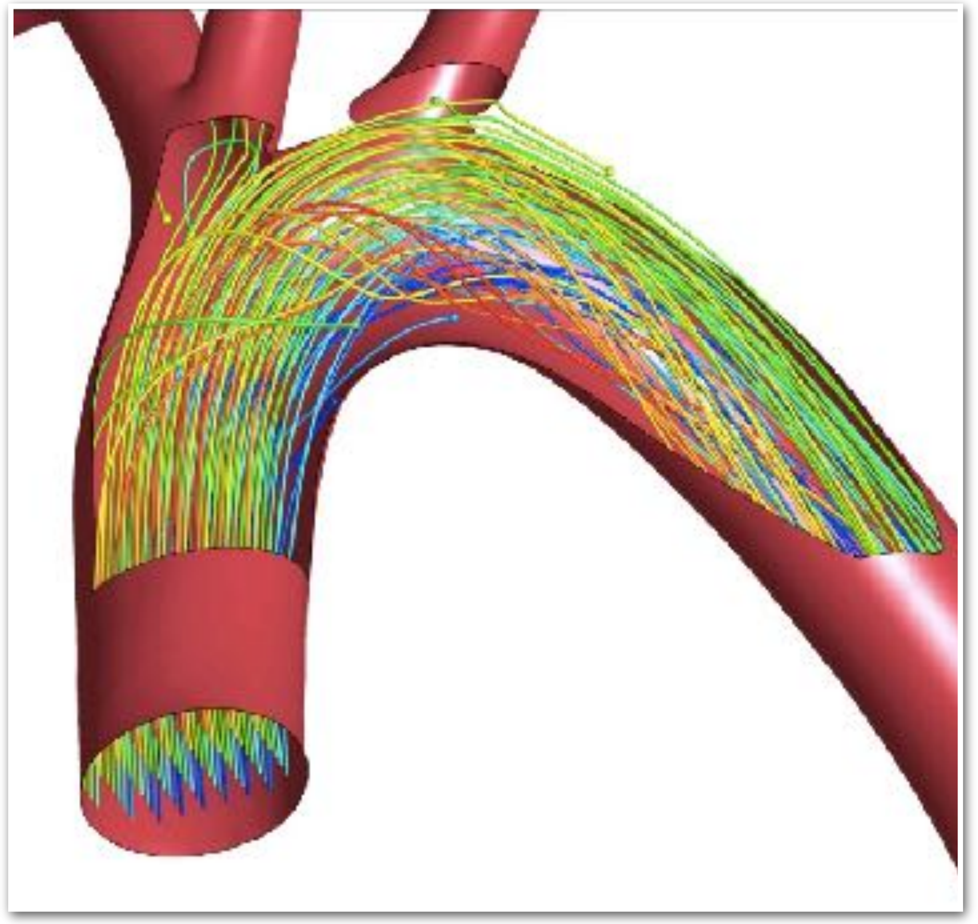
David Moxey, Department of Engineering, King's College London
Chris Cantwell, Department of Aeronautics, Imperial College London

Overview

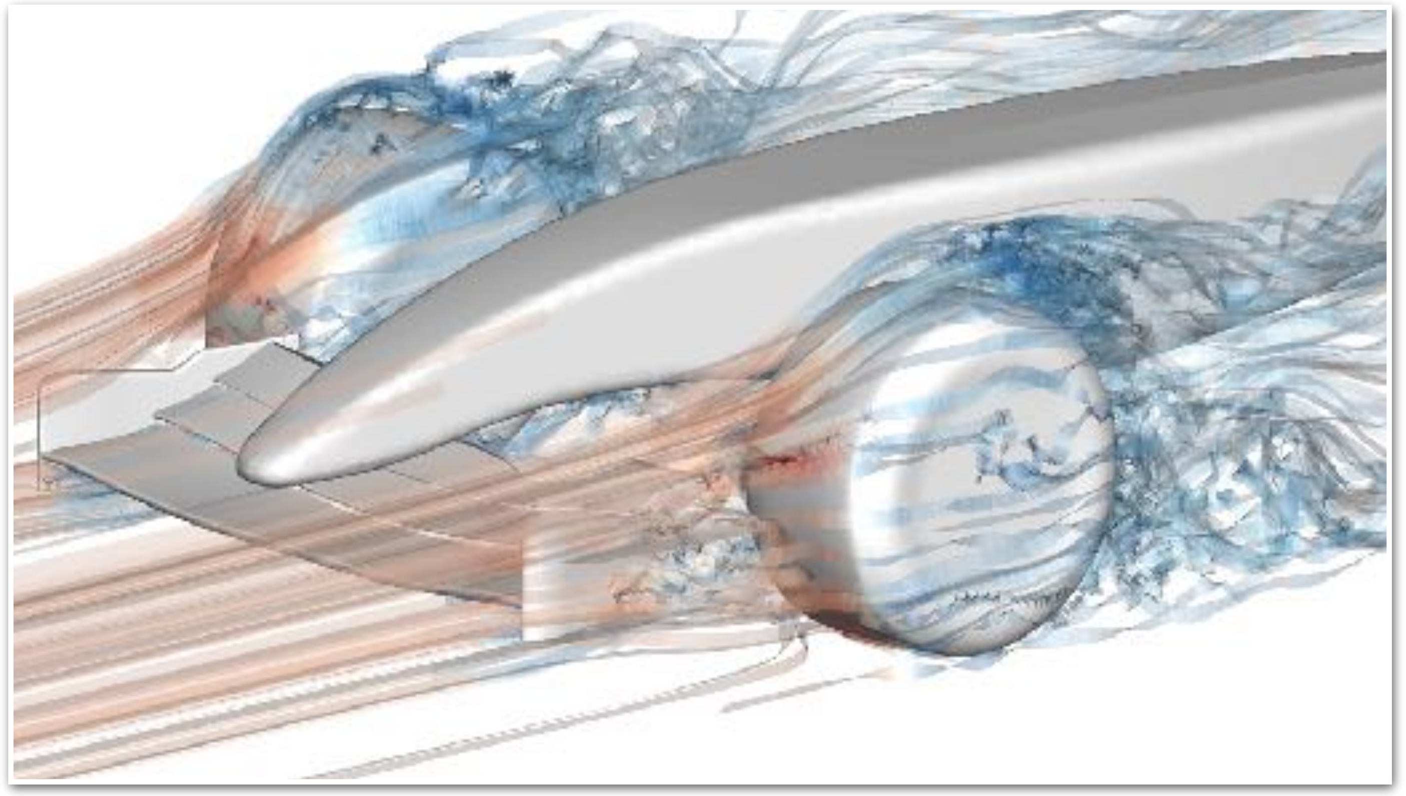
- Computational modelling plays a key role in many scientific fields.
- The increasing power of computing hardware means:
  - ➡ more detailed simulations, finer resolution of physics;
  - ➡ look at wider ranges of parameters.
- Today I'll talk about two fairly disconnected fields: fluid dynamics & cardiac electrophysiology.
- How do we design software that's flexible enough to tackle multiple areas & can exploit power on offer?



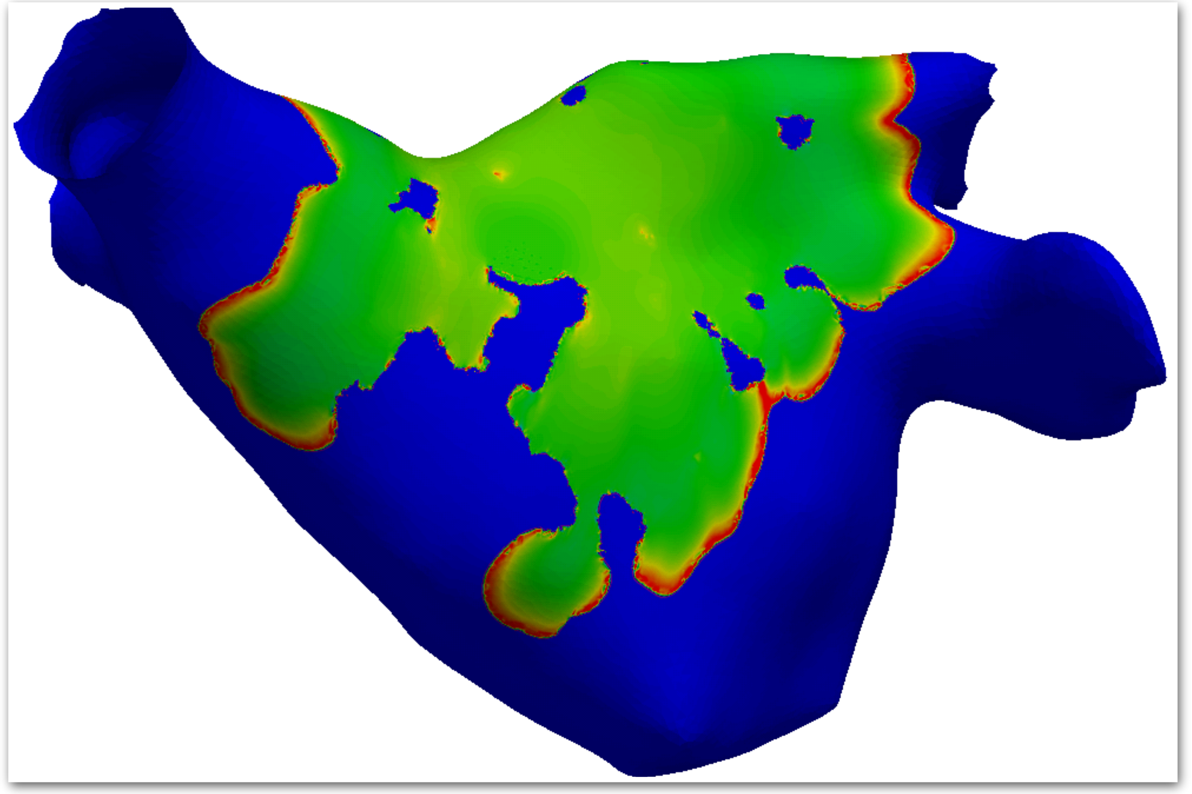
aviation and aerospace



bioflows & bioengineering



motorsport



cardiac electrophysiology

Lombard, Moxey et al., *AIAA J* (2016)

Increasing desire for **high-fidelity** simulation in high-end engineering applications.

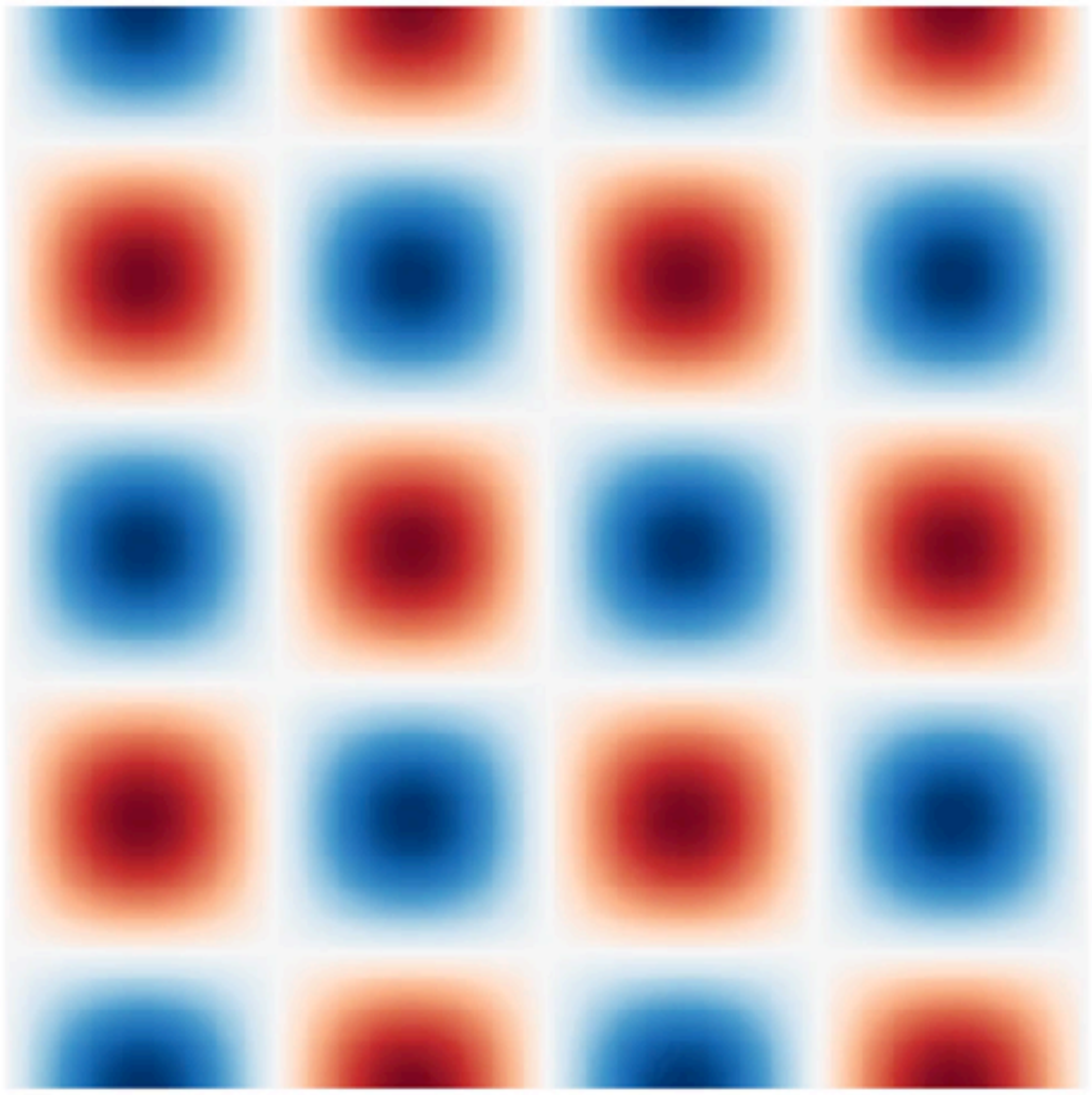
Want to accurately model difficult features:

- strongly separated flows
- feature tracking and prediction
- vortex interaction

Computational power is nearly there: but we need the software to drive this.

How do we model a physical system?

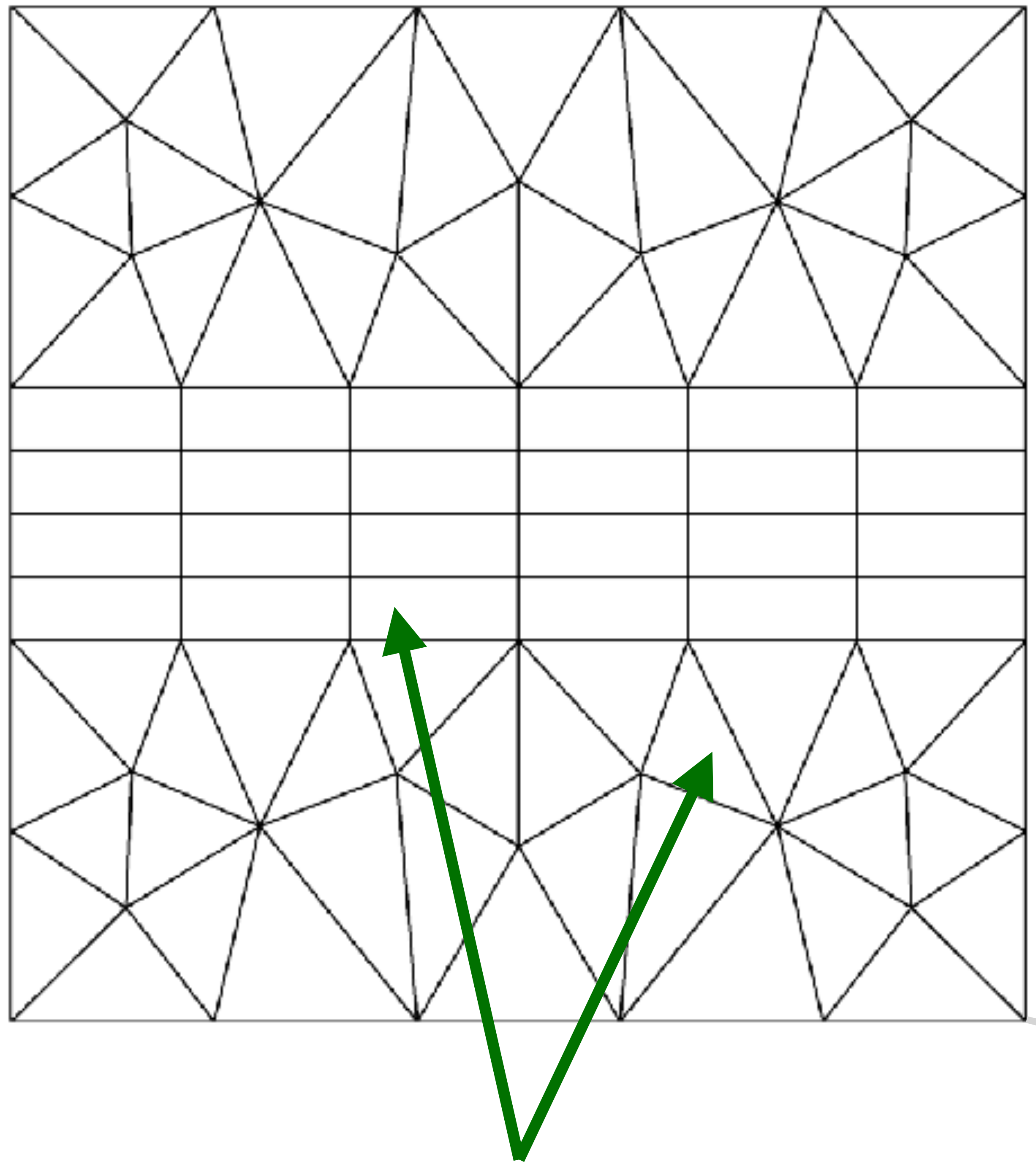
form underlying equations



advection velocity $\mathbf{v}$

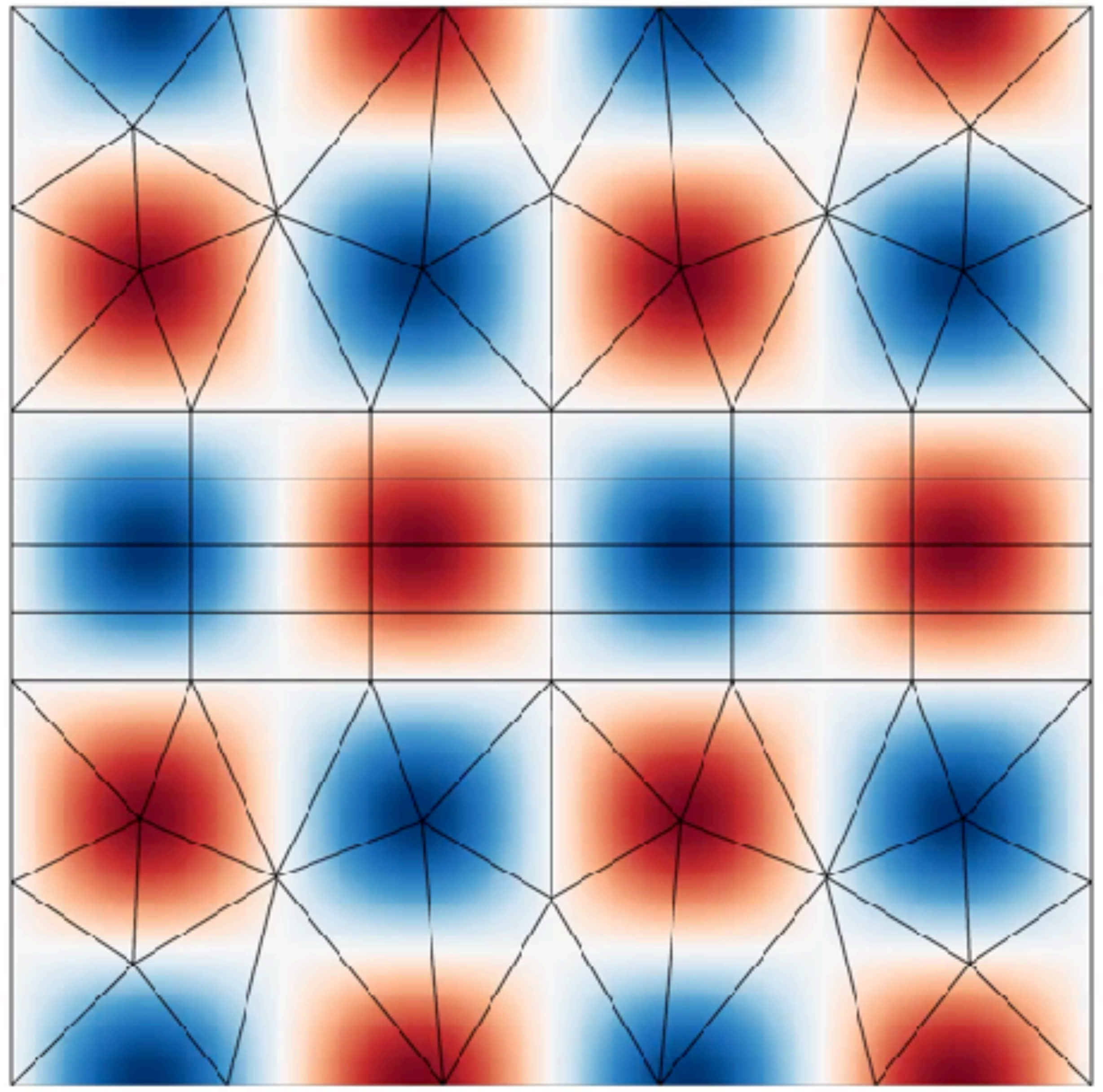
$$\frac{\partial u}{\partial t} + \mathbf{v} \cdot \nabla u = 0$$

discretise (typically) complex domain



simpler ***elements*** on which the equations can be solved

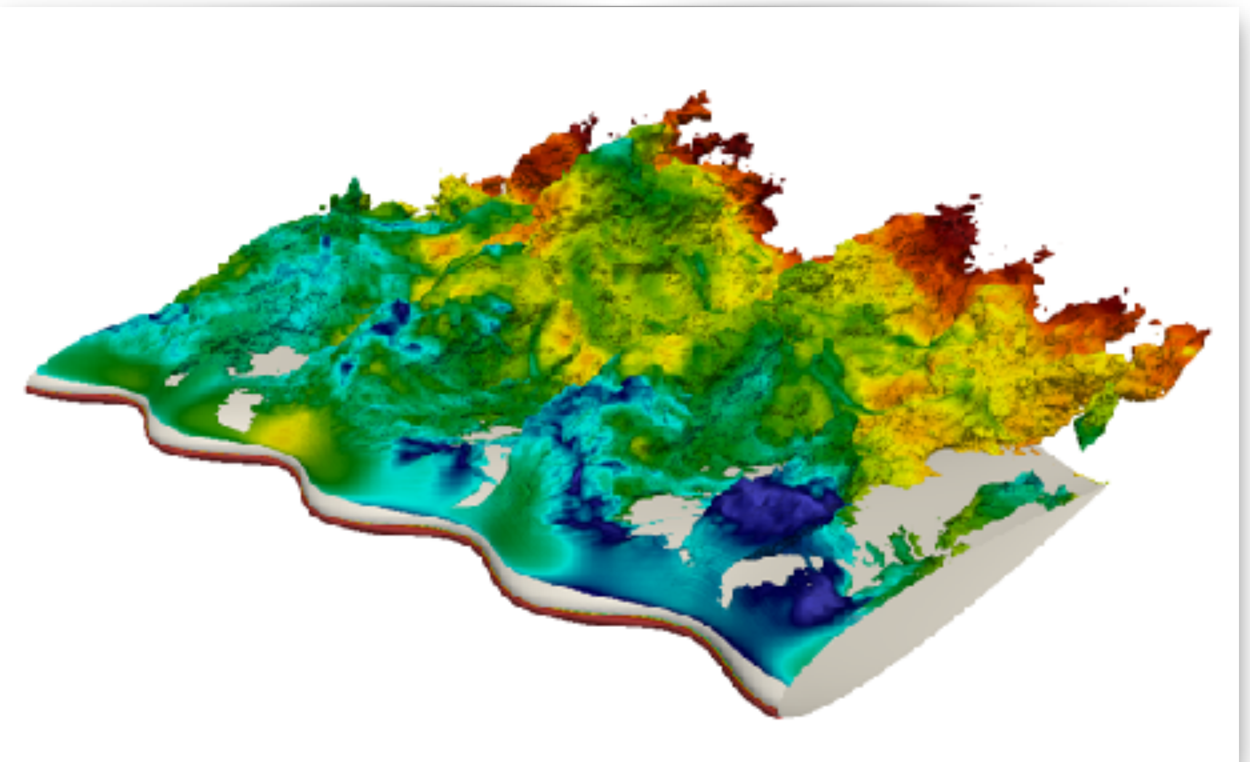
solve resulting system



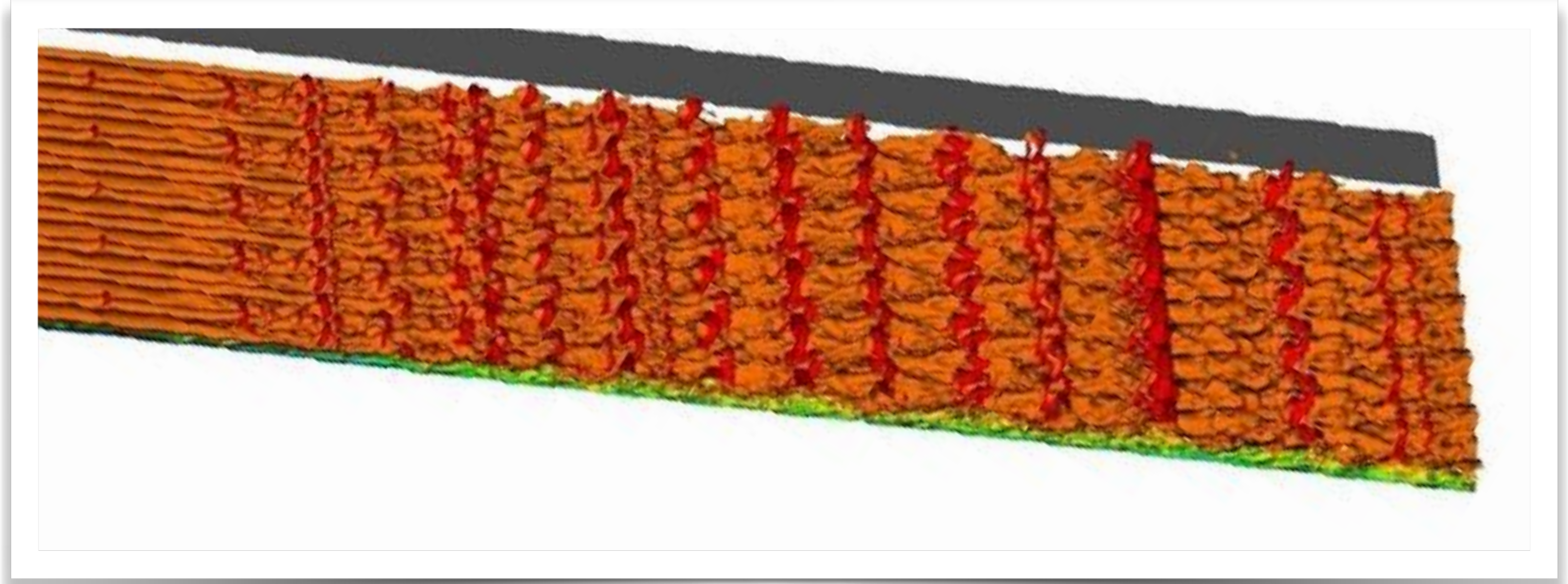
$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

(+ timestepping)

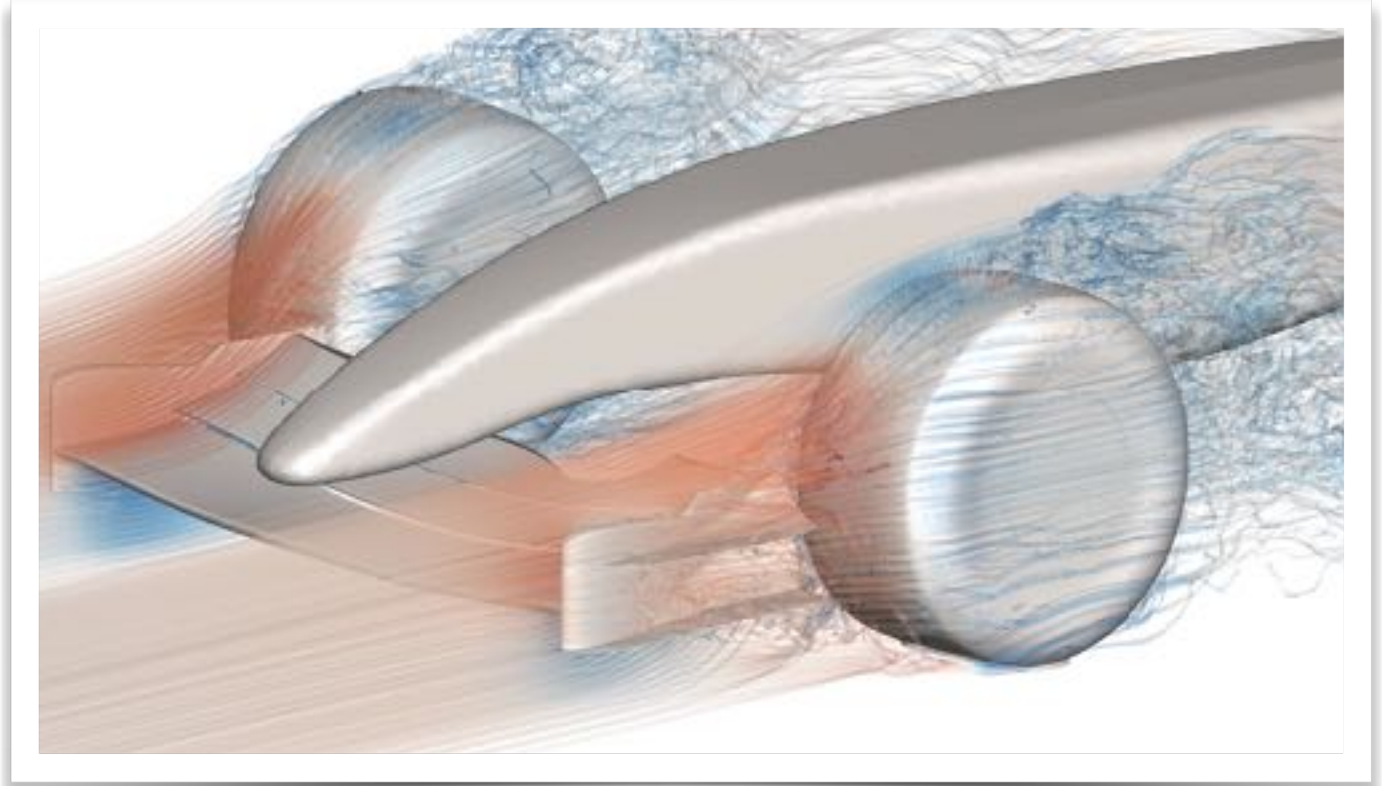
Applications in aeronautics



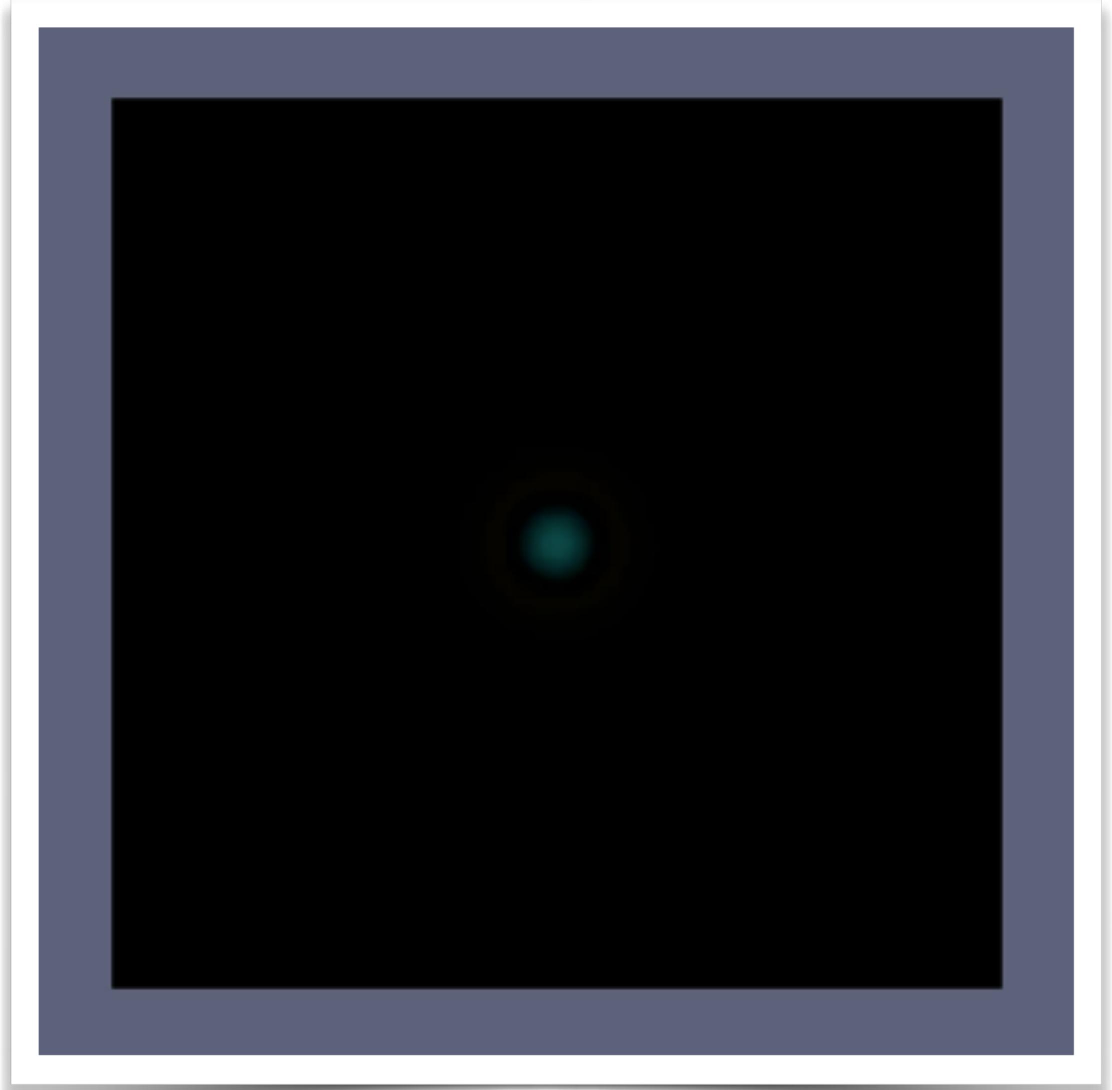
Wavy wing (Serson)



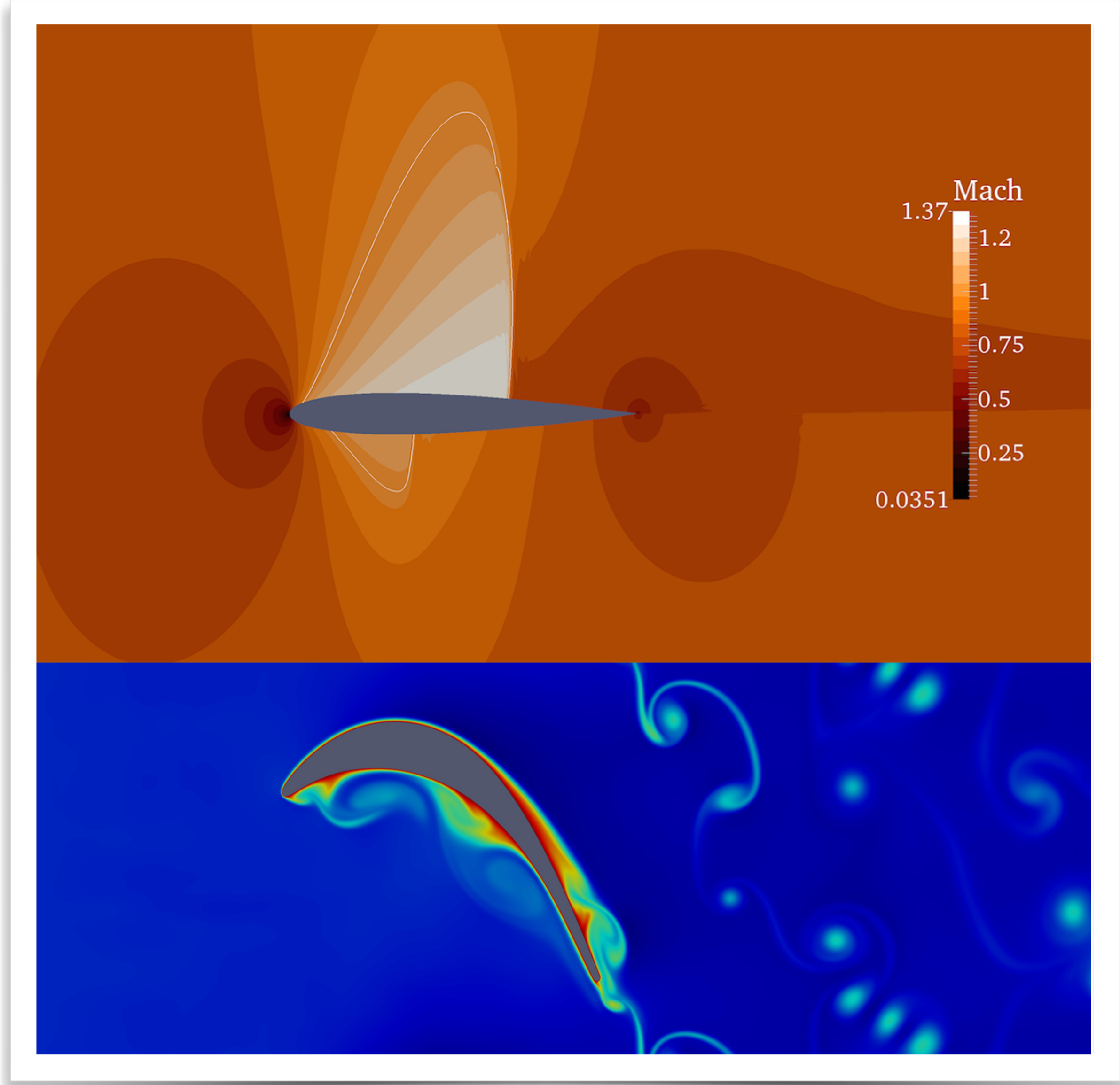
Transition to turbulence (Xu)



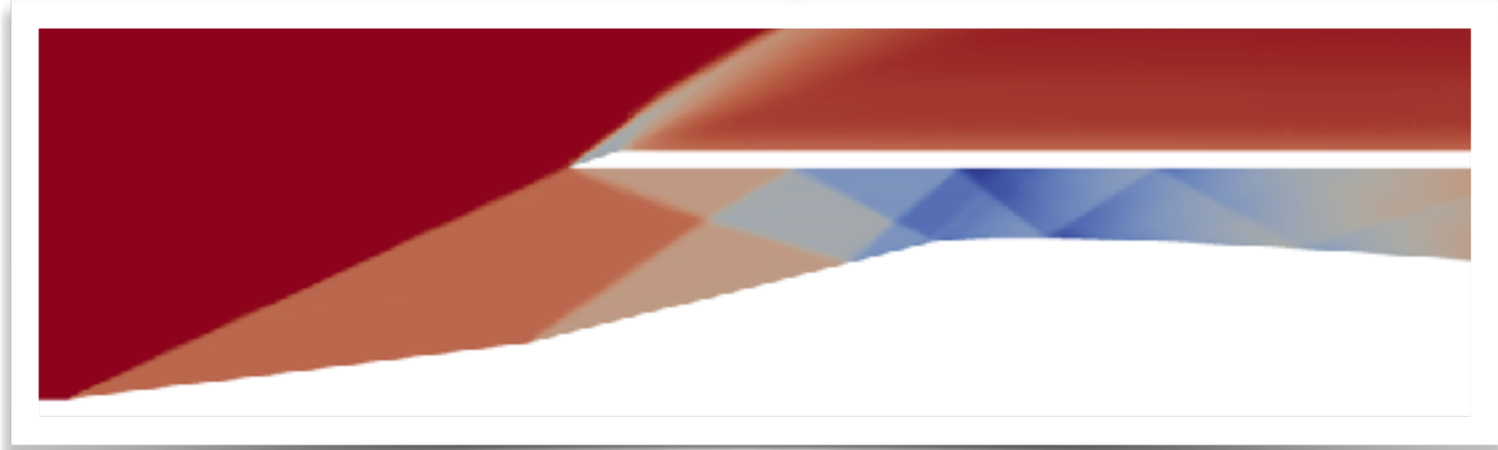
F1 (Lombard, Moxey, Sherwin)

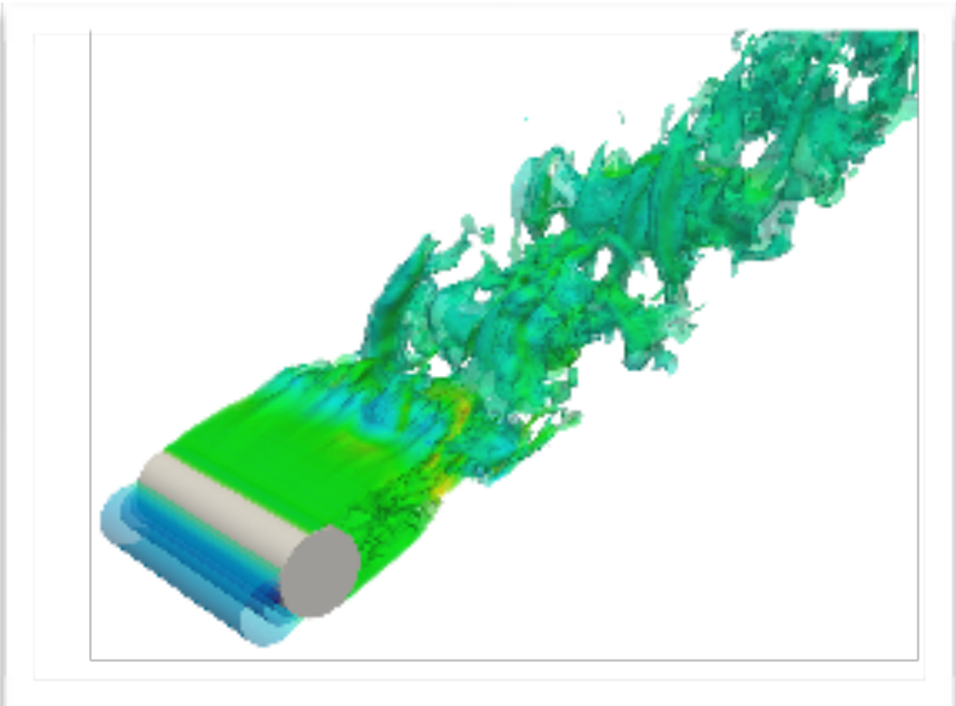


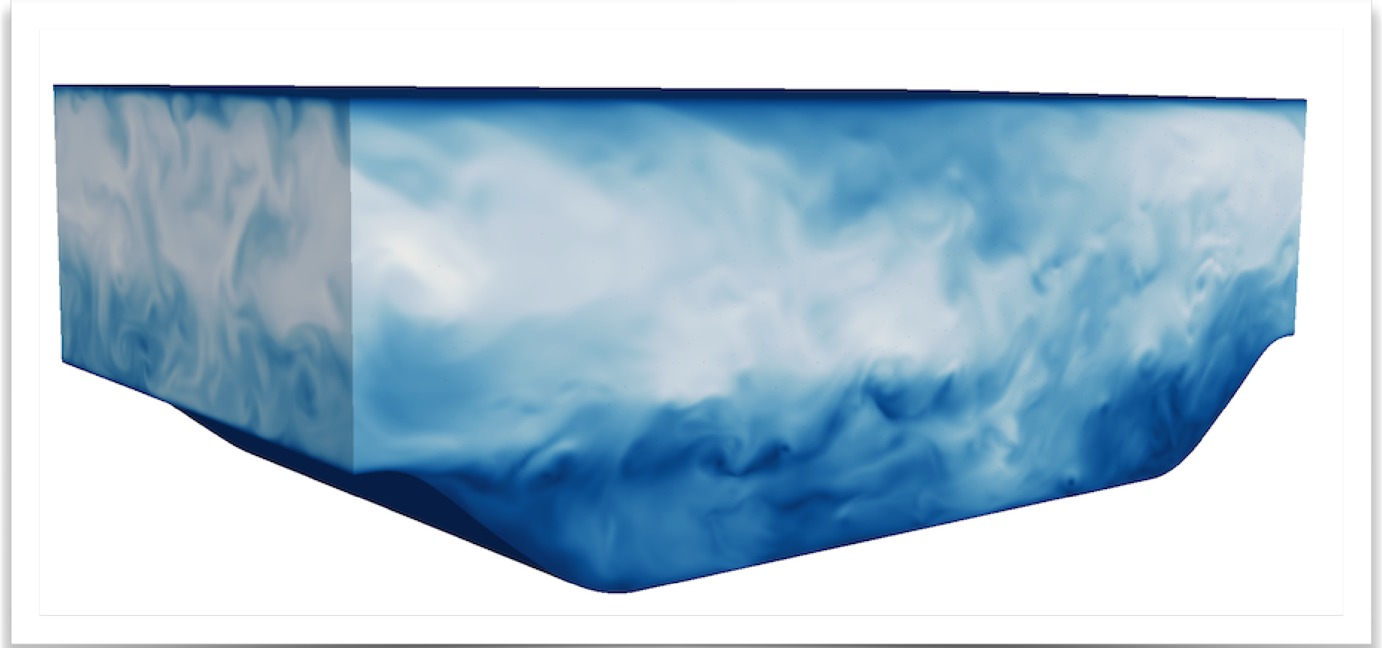
Magnetic turbulence (Moxey)



Compressible flows (Yan, Pan, Mengaldo, Sherwin, Moxey)

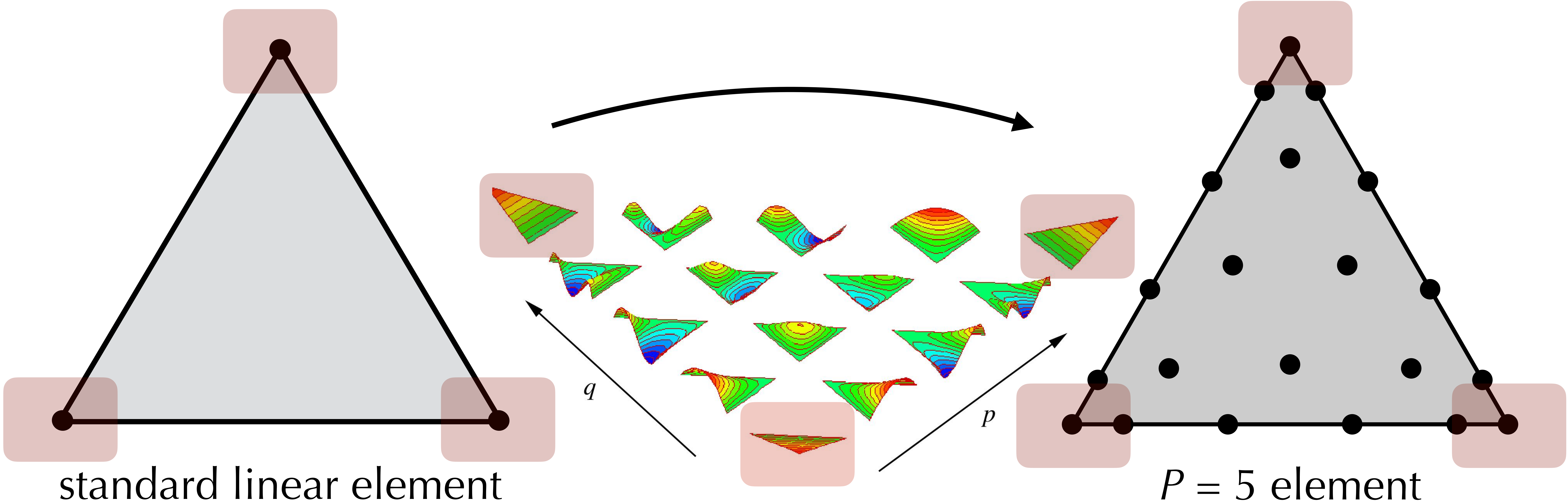






Turbulent hill (Moxey)

High-order methods

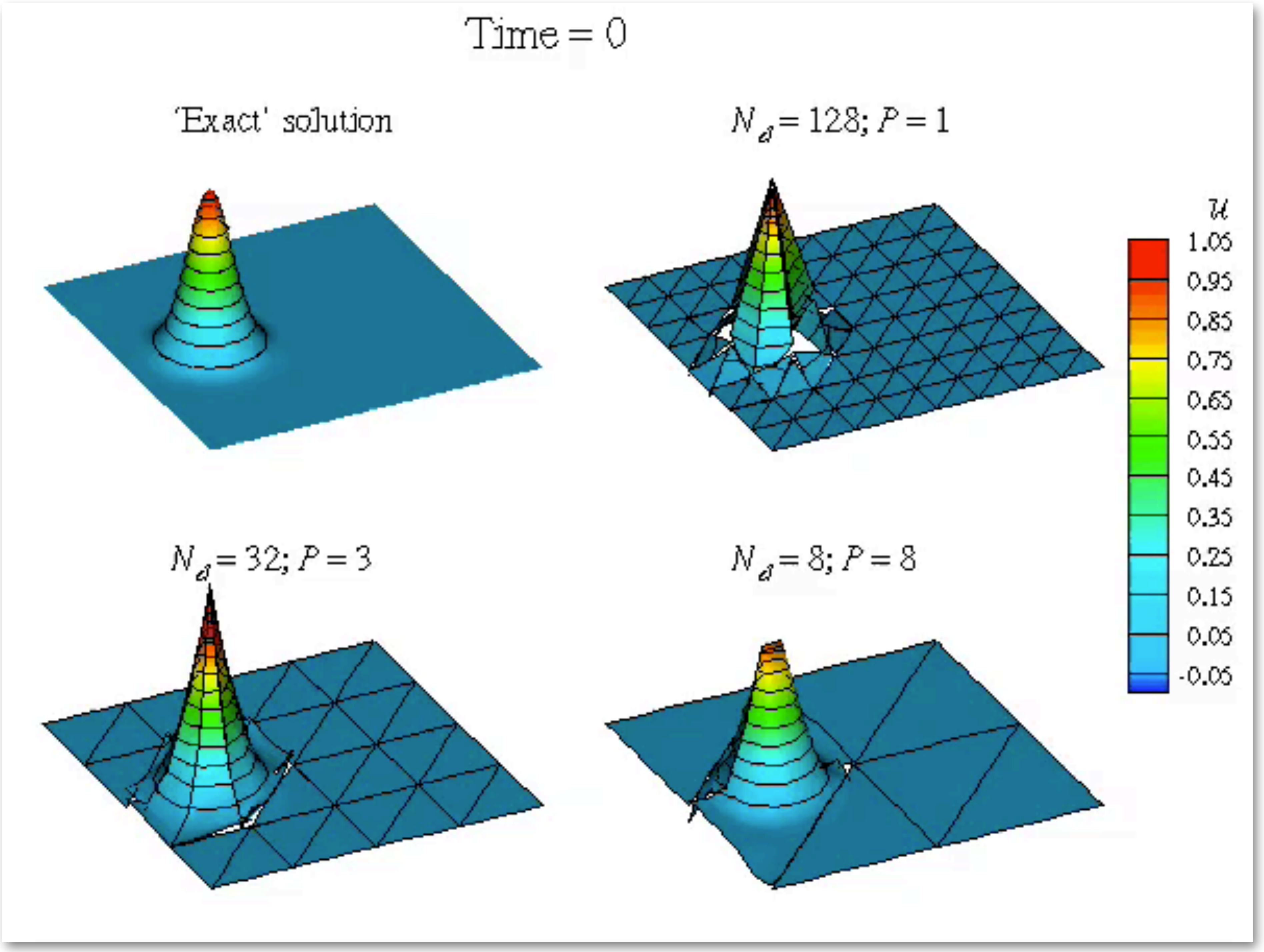


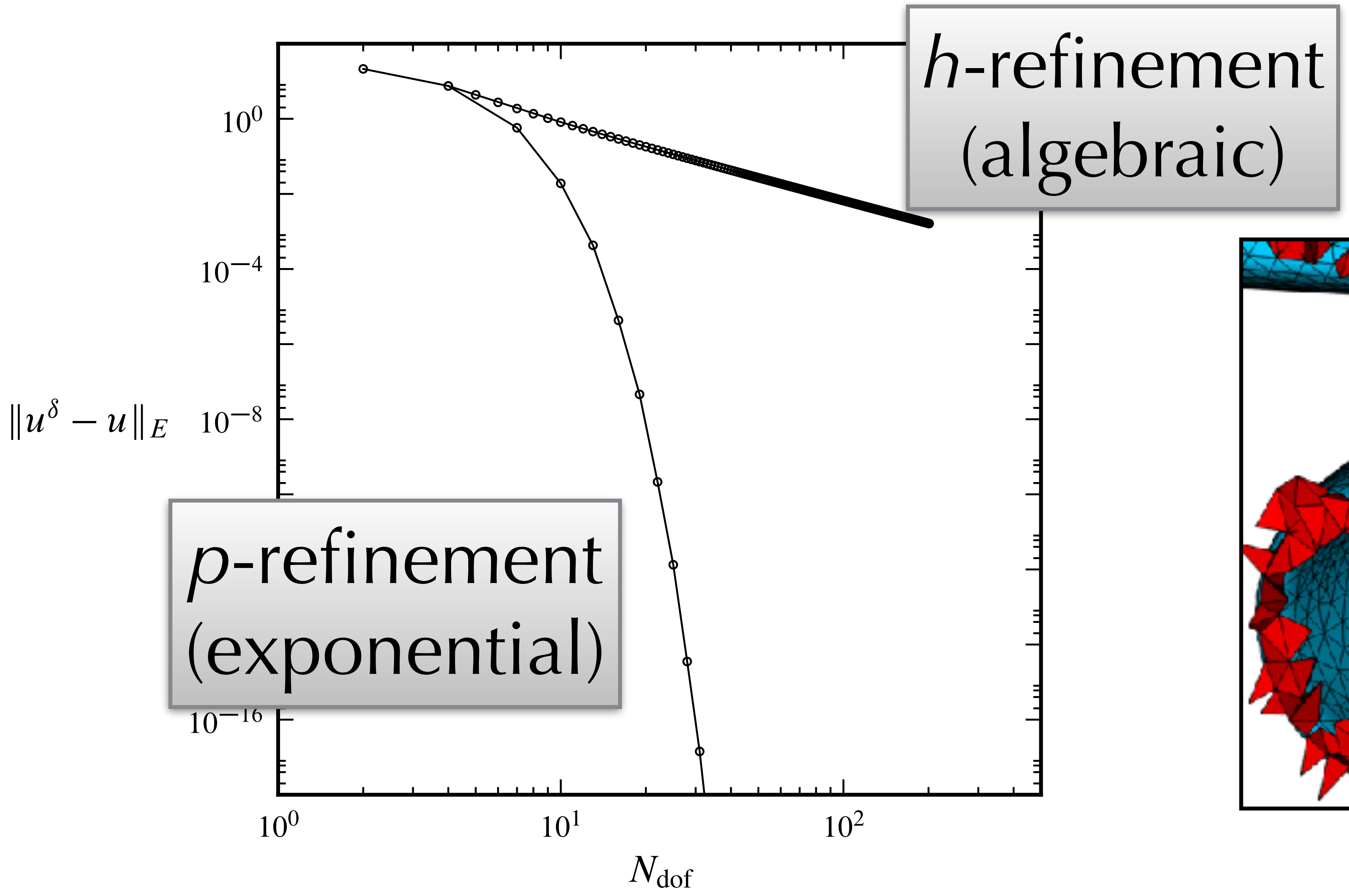
standard linear element
3 degrees of freedom
one per vertex (or cell)

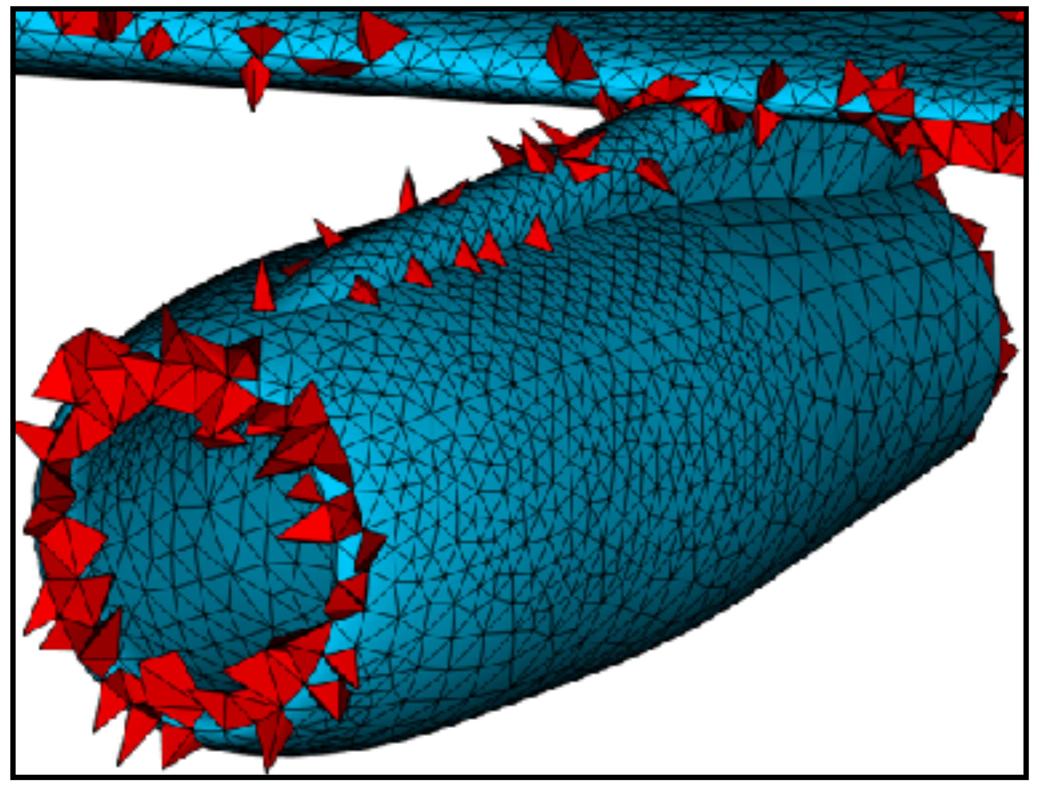
$P = 5$ element
21 degrees of freedom
vertex, edge and interior 'modes'

High-order methods for fluid dynamics

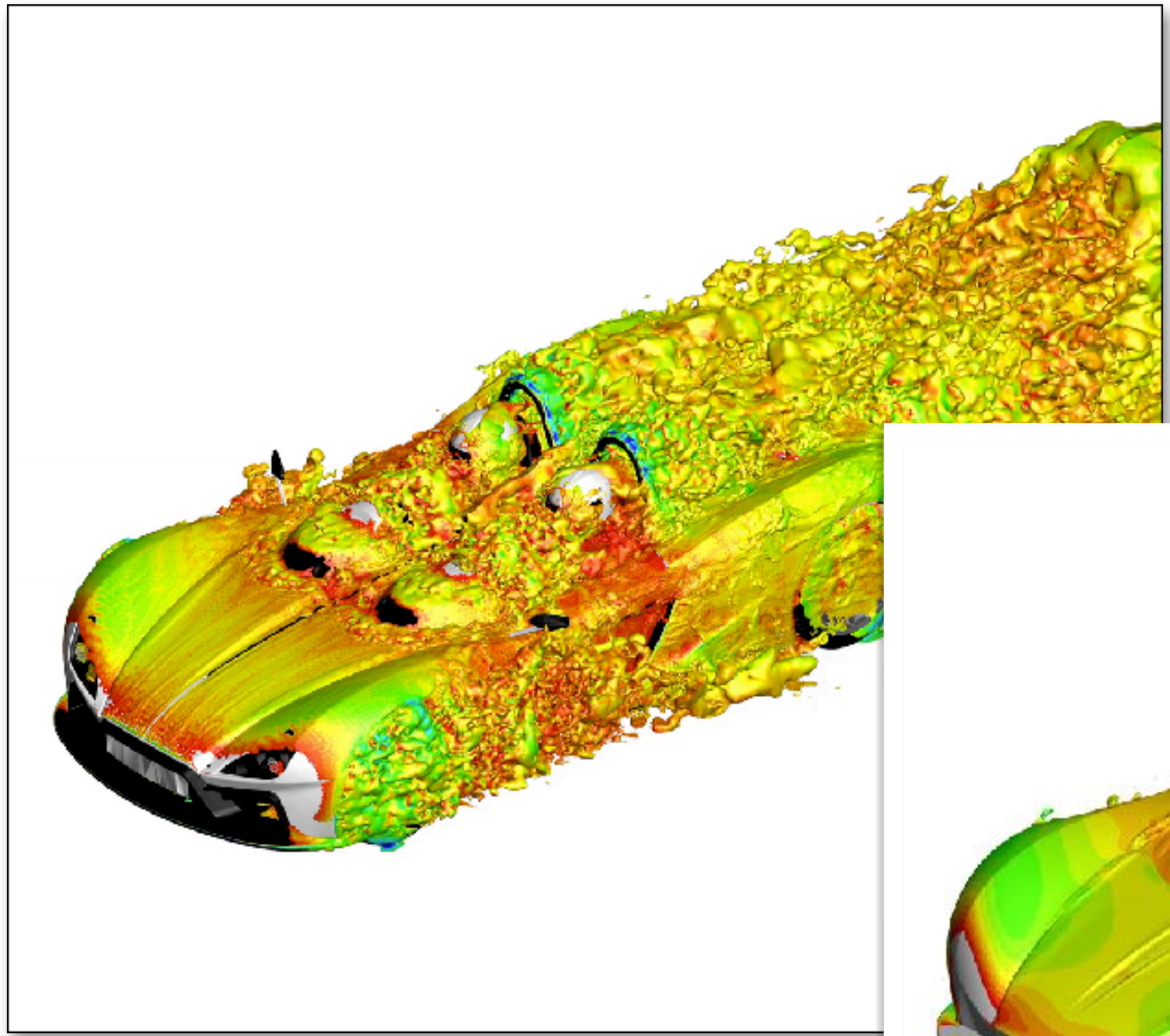
- ✓ error decays *exponentially* (smooth solutions);
- ✓ favorable diffusion & dispersion characteristics;
- ✓ model complex domains
- ✓ computational advantage: reduced memory bandwidth, better use of hardware.



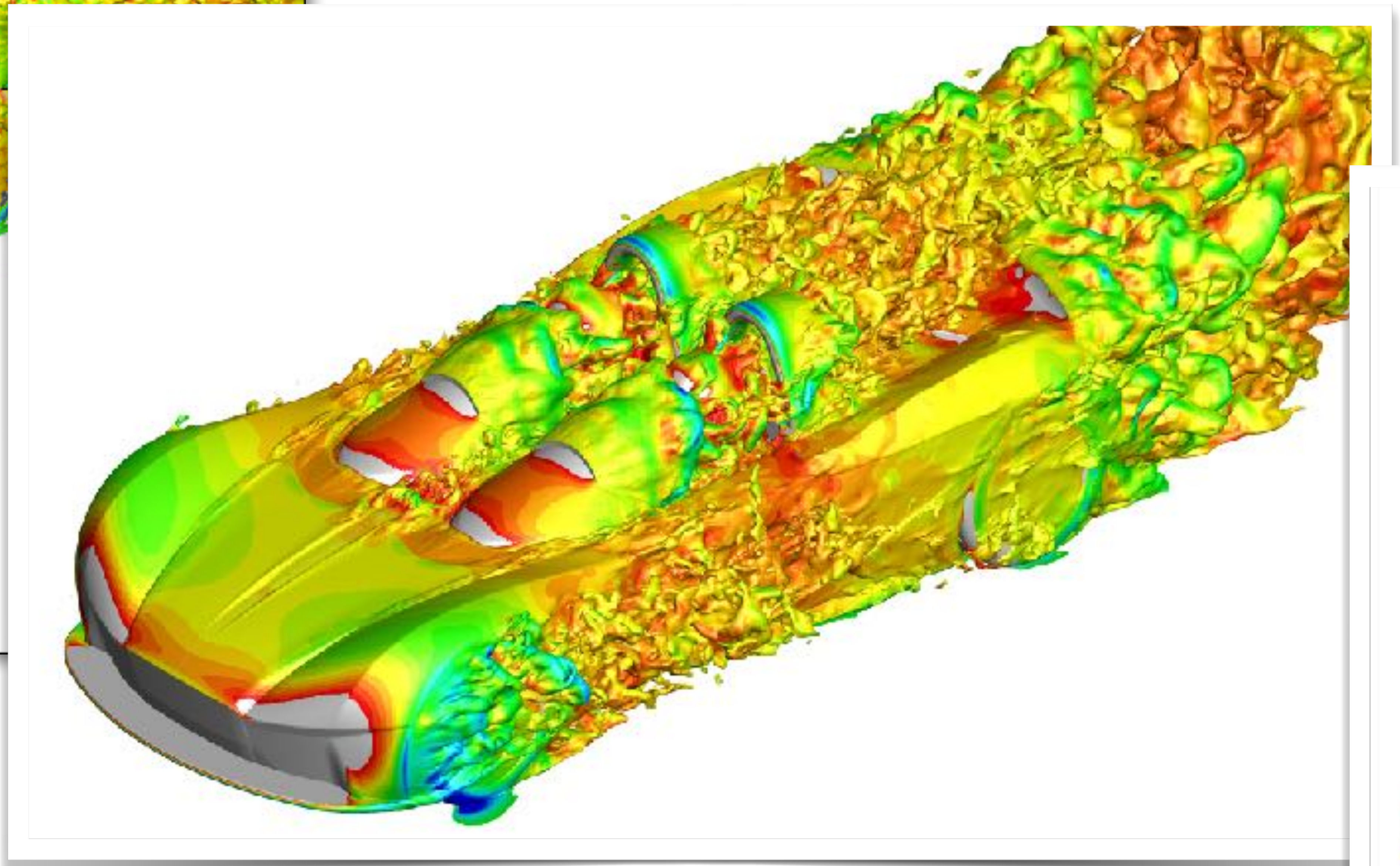




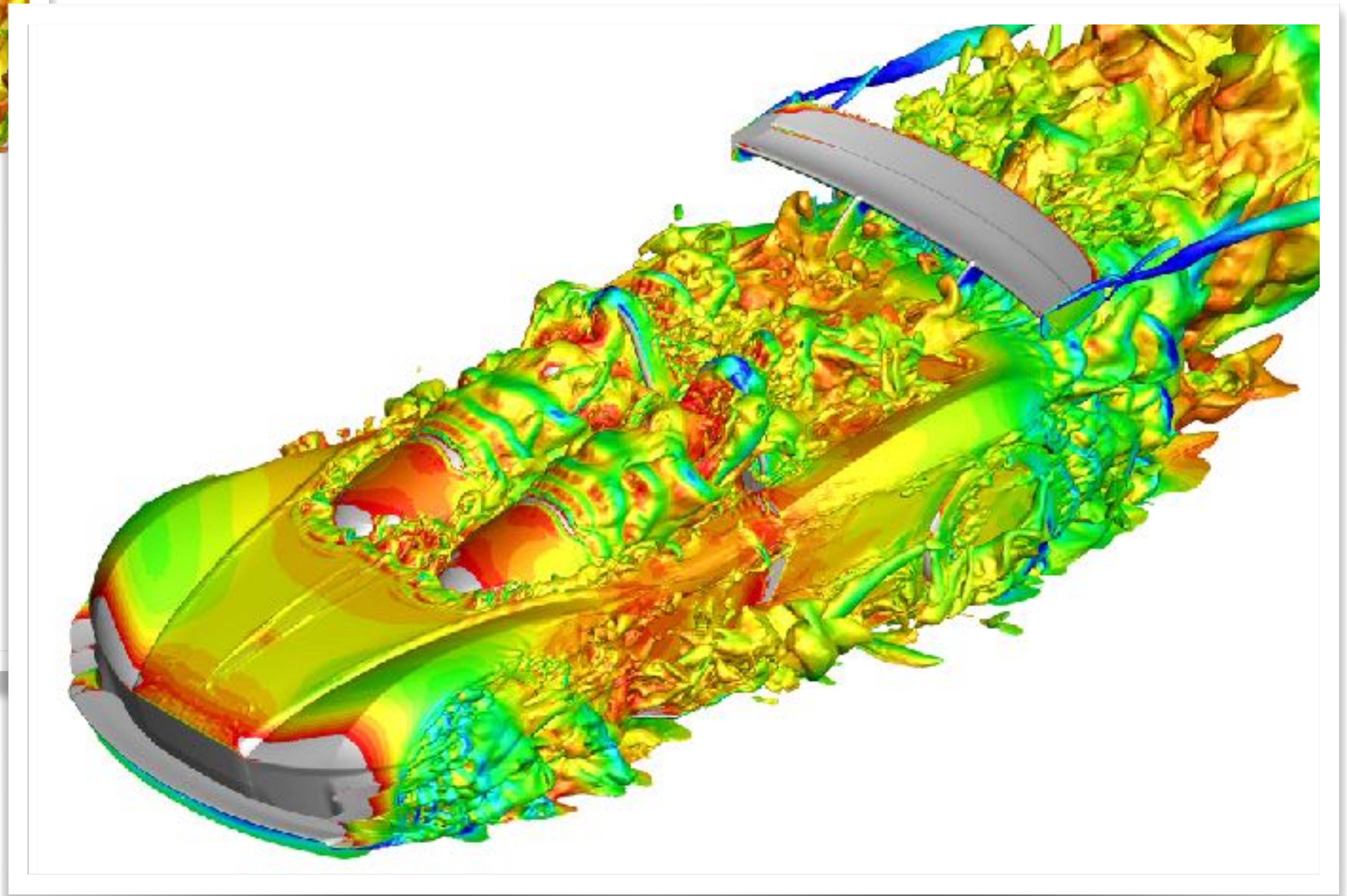
Virtual wind tunnel: Elemental road race car



1bn degrees of freedom
Uses only CFD for design



Design 2: +33% Downforce



Design 3: +270% Downforce

Mengaldo, Moxey, Turner, Jassim, Taylor, Peiro & Sherwin, SIAM Review (2021)

Moving to problems outside of fluids

- Original spectral/*hp* implementation for fluid dynamics was made ~20 years ago in a code called **Nektar**.
- We wanted to expand these methods to things outside of just fluid dynamics. But there were a lot of software problems trying to adapt Nektar for this:
    - × **No version control:** everyone had their own versions.
    - × **No testing:** no way to know if code changes/new features broke things.
    - × **Code structure:** code was tied to fluids & hard to make work outside of this area for more general problems.
    - × **Revision was hard:** Difficult to get started if you were new to the code.
- Motto of the story: it's not enough to write working software: you need to write *maintainable* software. Modern development practices are essential.

Nektar++

spectral/hp element framework

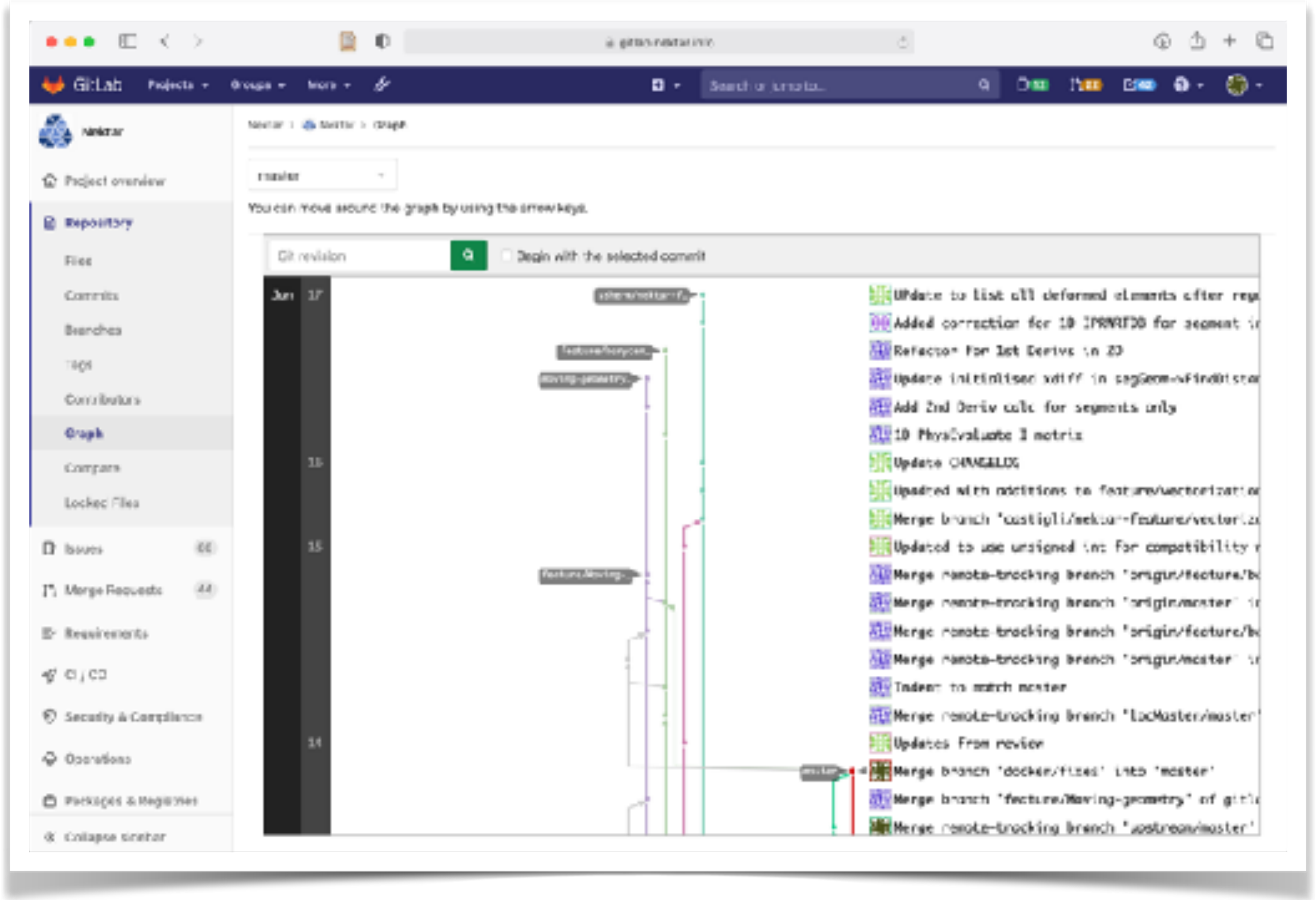
High-fidelity numerical methods
Highly parallel, designed for unsteady flows

Nektar++

spectral/hp element framework

- Nektar++ is an **open source framework** for high-order methods.
- Although fluids is a key application area, we try to make it easier to use these methods in many areas, **not just fluids**.
- Modern C++ design; runs at variety of scales, from desktops (1-128 cores) through to supercomputers (100k+). Uses pure MPI for parallelisation.
- Extensive use of modern software development practices: continuous integration/delivery, git, containerisation.

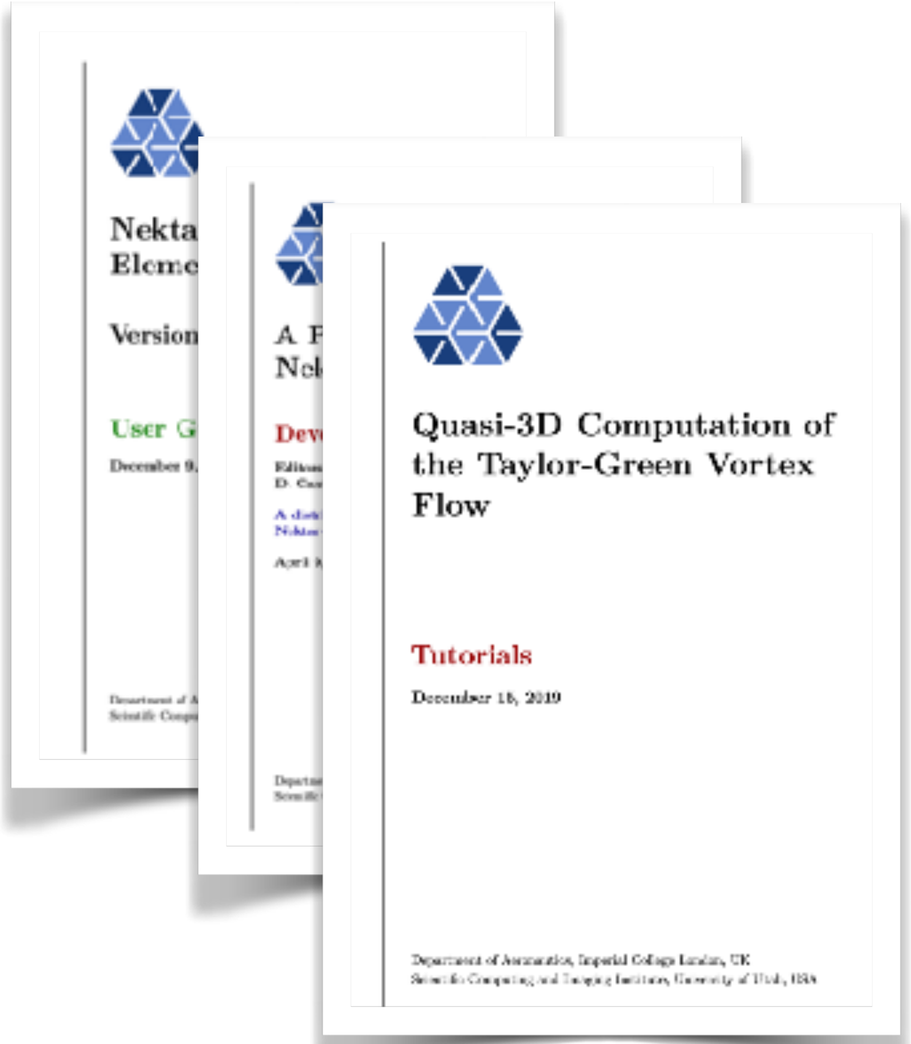
What practices do we use in Nektar++?



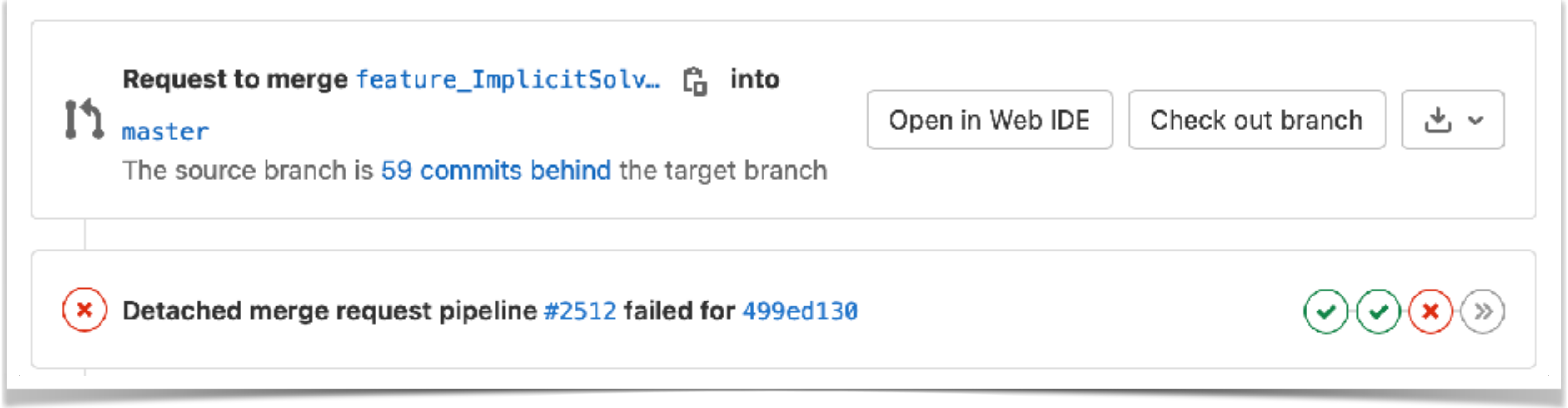
git (hosted on gitlab) for version control

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

docker for containerised builds (enables use of singularity for HPC)



documentation + tutorials



gitlab CI for continual integration (testing)

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

What does this give us?

- **Testing + code review:** Allows control over the source, make sure that before new features or fixes are merged, we aren't introducing *new* problems.

- **Docker:** generates an image of the code at each new commit.

  ✓ Reproducibility is much easier.

  ✓ Anyone can pull up-to-date master without compiling.

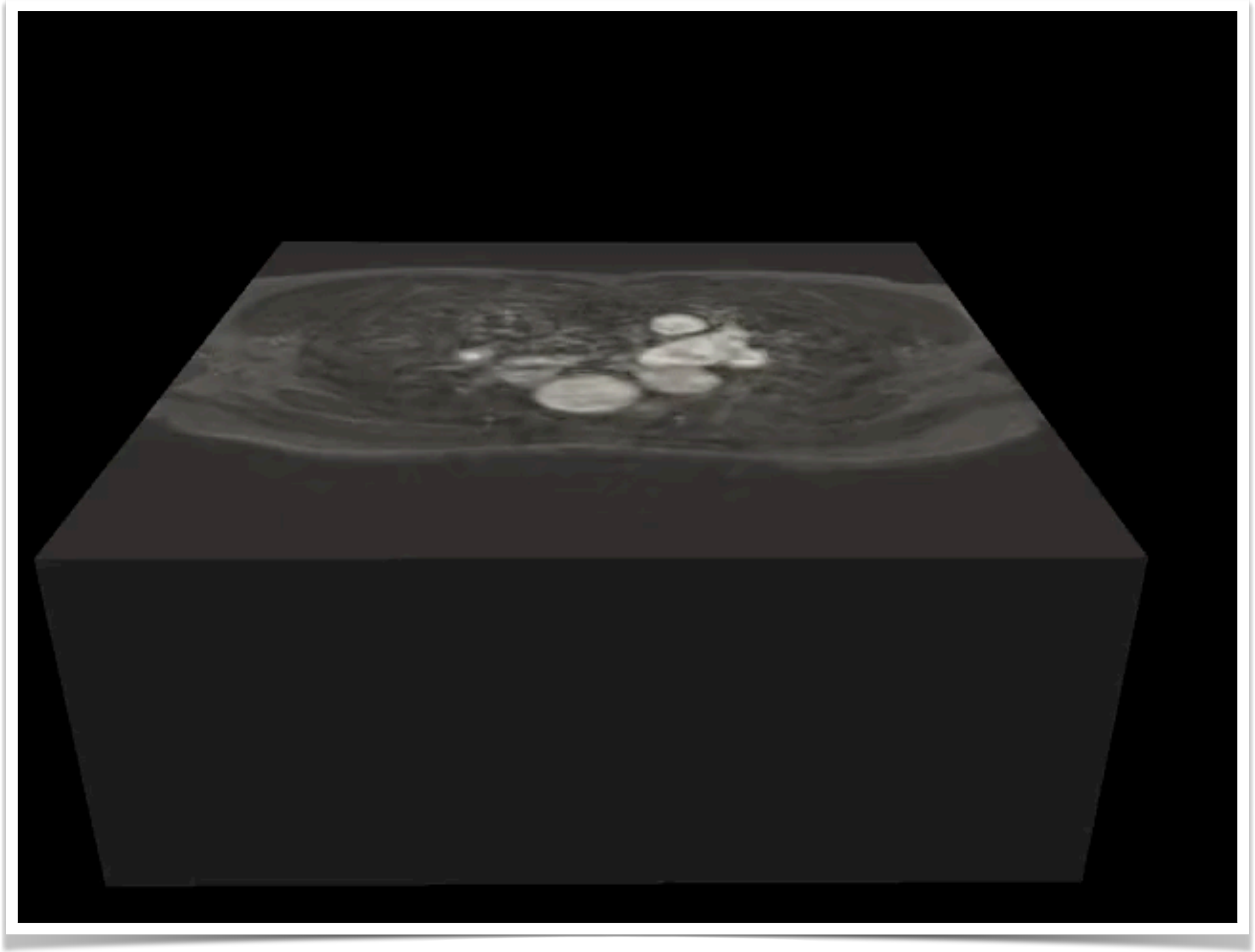
  ✓ CI built on top of this means we can quickly pull images where things fail and investigate without recompiling.

- **Interfaces:** C++ is pretty hard; friendly interfaces like a Python interface/Jupyter notebook makes it hopefully easier for people to get started with a complex codebase!

- Properly designed software means we can **extend outside of fluids!**

Modelling cardiac electrophysiology

- Cardiac electrophysiology is the study of how electrical activity occurs in the heart.
- Improve our mechanistic understanding.
    - Examine phenomena in controlled environments.
    - Ask questions which cannot be (easily) tested biologically.
    - Conduct experiments which cannot be (ethically) be performed in vivo.
- Develop emerging clinical tools.
    - Enrich existing clinical mapping technologies.
    - Precision diagnosis.
    - Personalised treatement.



Models for cardiac electrophysiology

Cellular Models of the Action Potential

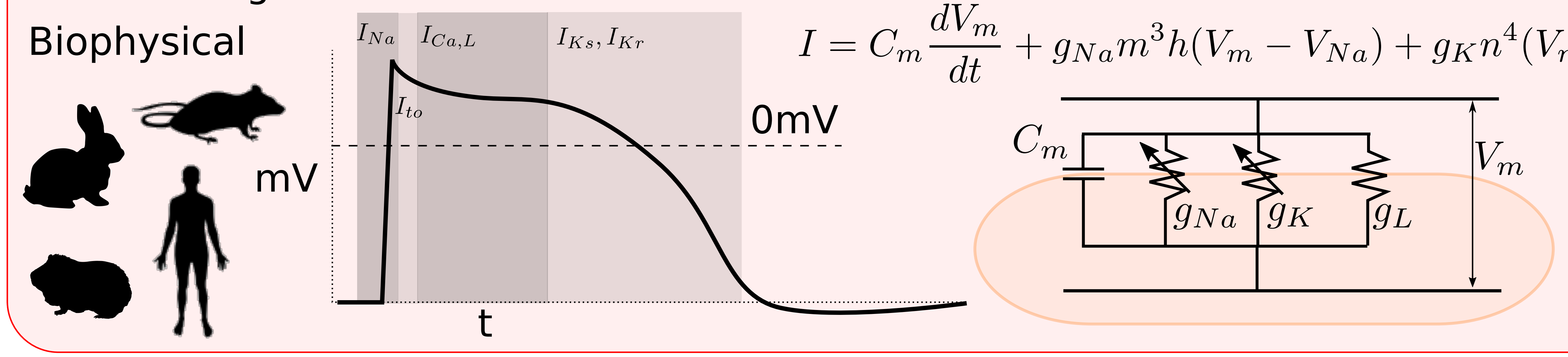
Phenomenological

Biophysical

$$\frac{dm}{dt} = \alpha_m(V_m)(1-m) - \beta_m(V_m)m$$

Hodgkin-Huxley

$$I = C_m \frac{dV_m}{dt} + g_{Na} m^3 h (V_m - V_{Na}) + g_K n^4 (V_m - V_K) + g_l (V_m - V_l)$$



Stochastic

Derive models for individual cells

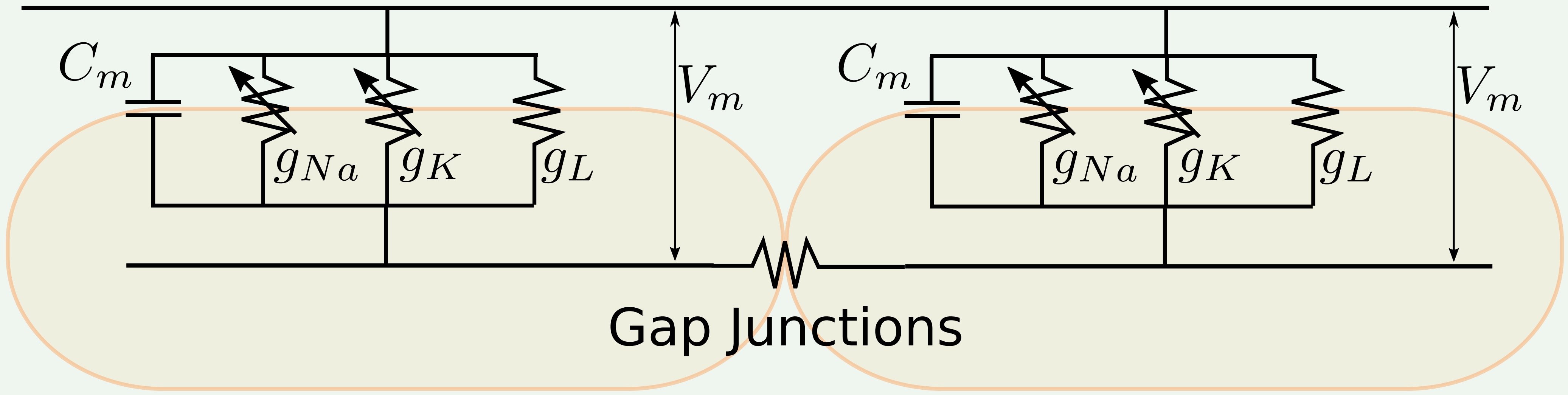
Monodomain Tissue Model

Assumption: $\mathbf{D}_e = \lambda \mathbf{D}_i$

$$\beta \left(C_m \frac{\partial V_m}{\partial t} + I_{ion} \right) = \nabla \cdot \mathbf{D} \nabla V_m$$

...which can be simplified (in some cases)

Inter-cellular coupling

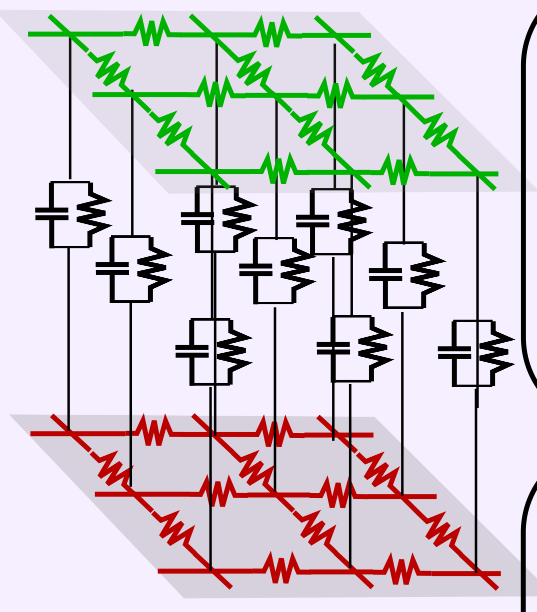


Cells coupled through gap junctions ...however too many of them, so homogenize

Bidomain Tissue Model

$$V_m = V_i - V_e$$

Extracellular



Intracellular

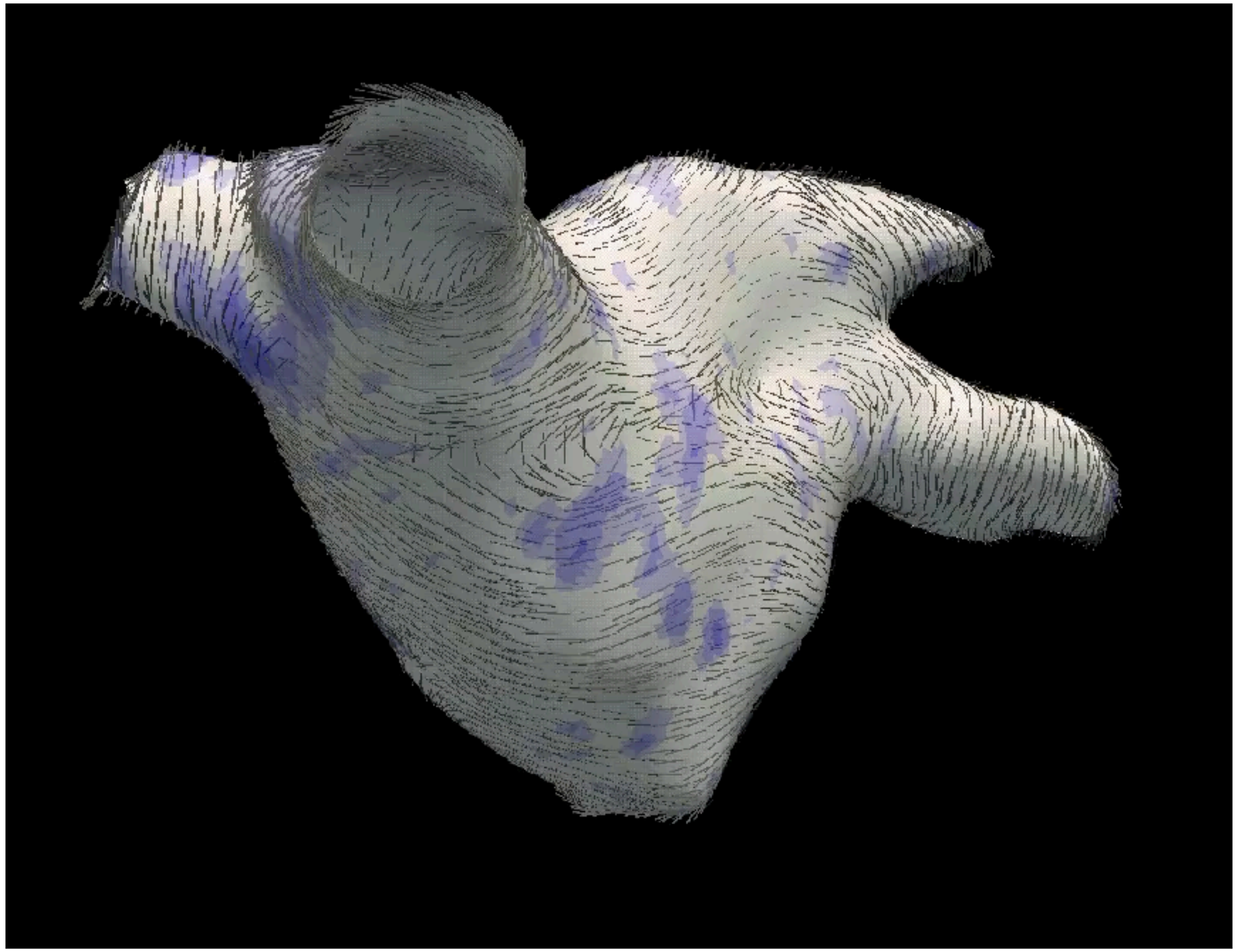
$$\nabla \cdot D_i \nabla V_m + \nabla \cdot D_e \nabla V_e = \beta \left(C_m \frac{\partial V_m}{\partial t} + I_{ion} \right)$$

$$\nabla \cdot D_i \nabla V_i + \nabla \cdot (D_e) \nabla V_e = 0$$

Leads to a PDE

Personalised models

- Great potential in computational modelling for *personalised* models to tailor treatment for specific patients.
- Incorporate multi-model clinical data:
  - ➡ Imaging (MR/CT)
    - Chamber geometry
    - Location of scar (LGE-CMRI)
  - ➡ Electroanatomic mapping
    - Activation patterns
    - Nature of arrhythmia
- Use these to tailor simulation parameters (e.g. diffusion tensor), understand how different treatments may affect activation patterns.



Summary

- Combination of cutting-edge models and high-quality software can give deep insights into challenging problems: from increasing downforce to understanding life-altering diseases.

- Learning about research software development and how to effectively use modern practices can be hugely beneficial.

- Lots of challenges still to overcome!

    - How do we make tools easier to set up and use?

    - How do we deal with the wealth of parameters, meshing, etc?

    - How do we do a better job of handling uncertainty?

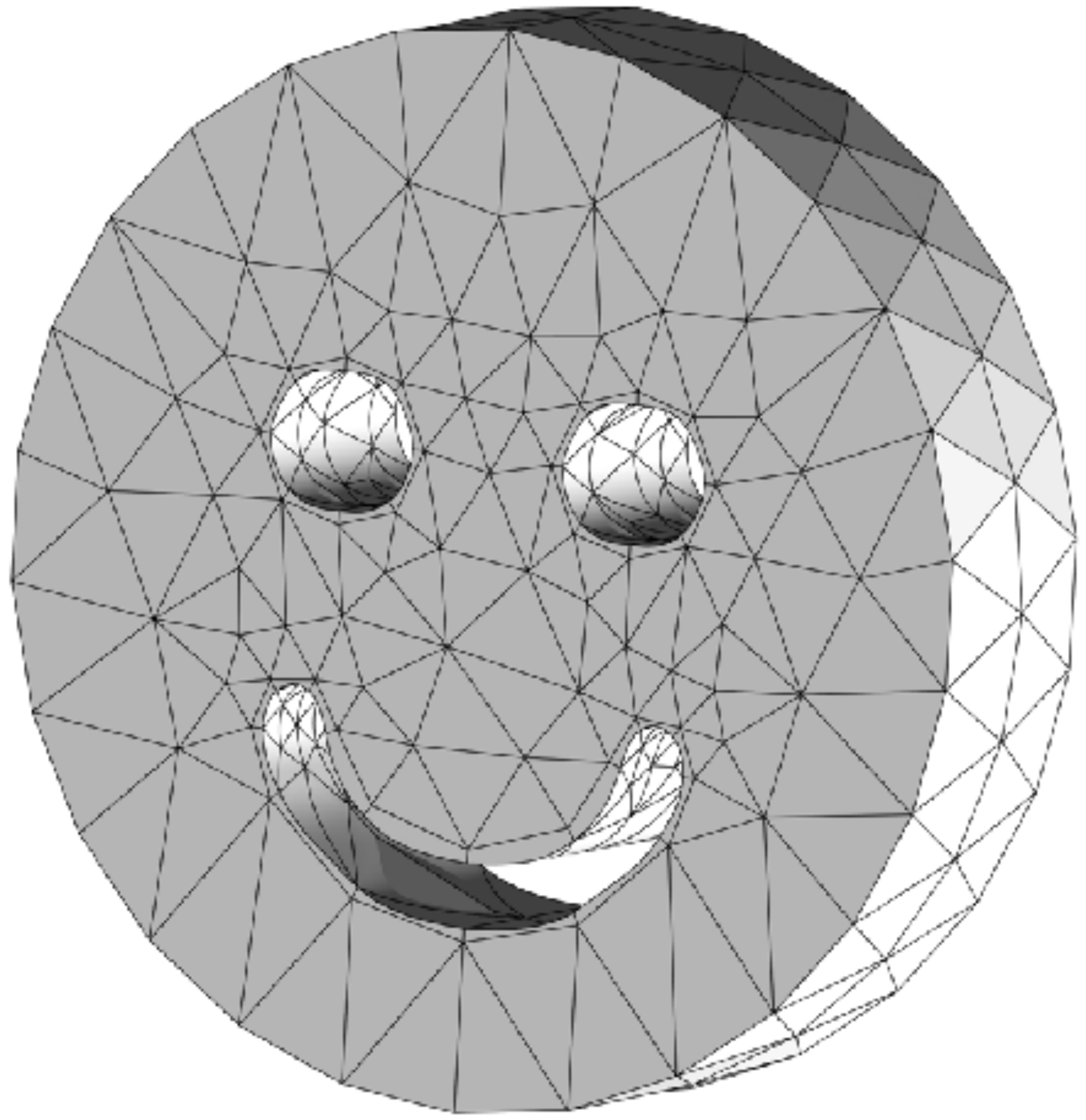
Thanks for listening!

https://davidmoxey.uk/

d.moxey@kcl.ac.uk

www.nektar.info

https://prism.ac.uk/



EPSRC
Pioneering research and skills

Nektar++: enhancing the capability and application of high-fidelity spectral/*hp* element methods

David Moxey[1], Chris D. Cantwell[2], Yan Bao[3], Andrea Cassinelli[2], Giacomo Castiglioni[2], Sehun Chun[4], Emilia Juda[2], Ehsan Kazemi[4], Kilian Lackhove[6], Julian Marcon[2], Gianmarco Mengaldo[7], Douglas Serson[2], Michael Turner[2], Hui Xu[5,2], Joaquim Peiró[2], Robert M. Kirby[8], Spencer J. Sherwin[2]