Imperial College London

Compatible finite element methods for numerical weather prediction

Colin Cotter
Professor of Computational Mathematics
Imperial College London

15th September 2020

Plan for the talk

[The lectures] should contain both research and educational material appealing to an early career scientist audience.

1. Compatible finite elements: what you need to know.
2. Some advances that we have made since 2013 in analysis and applications.

Imperial College
London

Compatible finite elements: an introduction

Some educational materials

- Introduction to implementation and theory of finite elements: `https://finite-element.github.io`
- Proceedings from my 2013 talk (1D introduction): Search for Cotter and McRae (2016) on the ArXiV (or ECMWF website).
- Gibson et al (2019), "Compatible Finite Element Methods for Geophysical Flows: Automation and Implementation Using Firedrake", Mathematics of Planet Earth Springer Briefs.

What are the ingredients of a finite element method?

1. A weak form of the unapproximated equations.
2. A choice of finite element space(s) for the numerical approximation.

We will describe both of these ingredients in the following slides.

For more details head to `https://finite-element.github.io`

What's a weak form?

e.g. Poisson's equation.

Strong form

$$-\nabla^2 p = f \text{ on } \Omega, \qquad p = 0 \text{ on } \partial\Omega.$$

What's a weak form?

e.g. Poisson's equation.

Strong form

$-\nabla^2 p = f$ on Ω, $\quad p = 0$ on $\partial\Omega$.

A weak form

Find p in V such that $-\int_\Omega q\nabla^2 p\,\mathrm{d}x = \int_\Omega fq\,\mathrm{d}x, \quad \forall\, q \in V$ (for some suitable set of functions V).

What's a weak form?

e.g. Poisson's equation.

Strong form

$-\nabla^2 p = f$ on Ω, $\quad p = 0$ on $\partial\Omega$.

A weak form

Find p in V such that $-\int_\Omega q\nabla^2 p\,\mathrm{d}x = \int_\Omega fq\,\mathrm{d}x, \quad \forall q \in V$ (for some suitable set of functions V).

Another weak form

Find p in $\mathring{V} = \{p \in V : p = 0 \text{ on } \partial\Omega\}$ such that
$\int_\Omega \nabla q \cdot \nabla p\,\mathrm{d}x = \int_\Omega fq\,\mathrm{d}x, \quad \forall q \in \mathring{V}$.

What makes a good weak form?

Find $p \in V$ such that $a(p; q) = F(q)$ for all $q \in V$.

Consistency

A weak form is consistent if it vanishes when you substitute a solution of the strong form into the weak form.

What makes a good weak form?

Find $p \in V$ such that $a(p; q) = F(q)$ for all $q \in V$.

Consistency

A weak form is consistent if it vanishes when you substitute a solution of the strong form into the weak form.

Uniqueness

Only one solution to the weak form (depends on the structure of the form and the choice of V).

What makes a good weak form?

Find $p \in V$ such that $a(p; q) = F(q)$ for all $q \in V$.

Consistency

A weak form is consistent if it vanishes when you substitute a solution of the strong form into the weak form.

Uniqueness

Only one solution to the weak form (depends on the structure of the form and the choice of V).

Stability

Small perturbations to the weak form result in small perturbations in the solution (ditto).

What's a finite element space?

Finite element space

A finite dimensional vector space of functions (i.e. expandable in a basis) defined on a mesh.

What's a finite element space?

Finite element space

A finite dimensional vector space of functions (i.e. expandable in a basis) defined on a mesh.

A specific finite element space $V_h \subset V$ is defined by two aspects:

1. The type of functions we allow in each mesh cell (shape space), and
2. The type of continuity we allow between neighbouring mesh cells.

Finite element approximation

Recall the weak form: find $p \in V$ such that

$$a(p; q) = F(q), \qquad \forall\, q \in V.$$

Finite element approximation

Recall the weak form: find $p \in V$ such that

$$a(p; q) = F(q), \qquad \forall q \in V.$$

Finite element approximation using $V_h \subset V$

Find $p_h \in V_h$ such that

$$a(p_h; q) = F(q) \qquad \forall q \in V_h.$$

Finite element approximation

Recall the weak form: find $p \in V$ such that

$$a(p; q) = F(q), \qquad \forall q \in V.$$

Finite element approximation using $V_h \subset V$

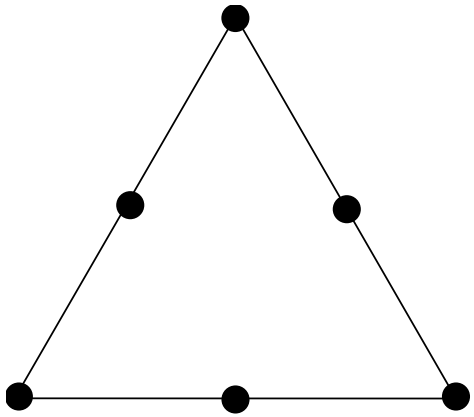
Find $p_h \in V_h$ such that

$$a(p_h; q) = F(q) \qquad \forall q \in V_h.$$

'Finite difference methods approximate the equation, but finite element methods approximate the solution instead.'

Bill Morton

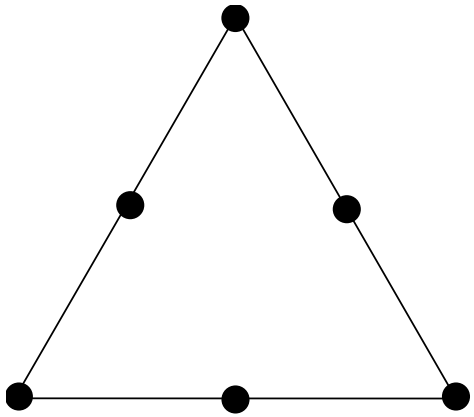
How do we build bases for finite element spaces?



Nodal variables

A set of mappings from the shape space to numbers, $\mathcal{N}_1, \ldots, \mathcal{N}_n$.

How do we build bases for finite element spaces?



Nodal variables

A set of mappings from the shape space to numbers, $\mathcal{N}_1, \ldots, \mathcal{N}_n$.

Nodal basis for the shape space

$$\mathcal{N}_i[\phi_j] = \delta_{ij}.$$

What are compatible FE spaces?

$$\mathbb{V}^0 \xrightarrow{\nabla^\perp} \mathbb{V}^1 \xrightarrow{\nabla\cdot} \mathbb{V}^2$$

What are compatible FE spaces?

$$\begin{array}{ccccc} H^1 & \xrightarrow{\nabla^\perp=(-\partial_y,\partial_x)} & H(\mathrm{div}) & \xrightarrow{\nabla\cdot} & L^2 \\ \downarrow{\scriptstyle \pi_0} & & \downarrow{\scriptstyle \pi_1} & & \downarrow{\scriptstyle \pi_2} \\ \mathbb{V}^0 & \xrightarrow{\nabla^\perp} & \mathbb{V}^1 & \xrightarrow{\nabla\cdot} & \mathbb{V}^2 \end{array}$$

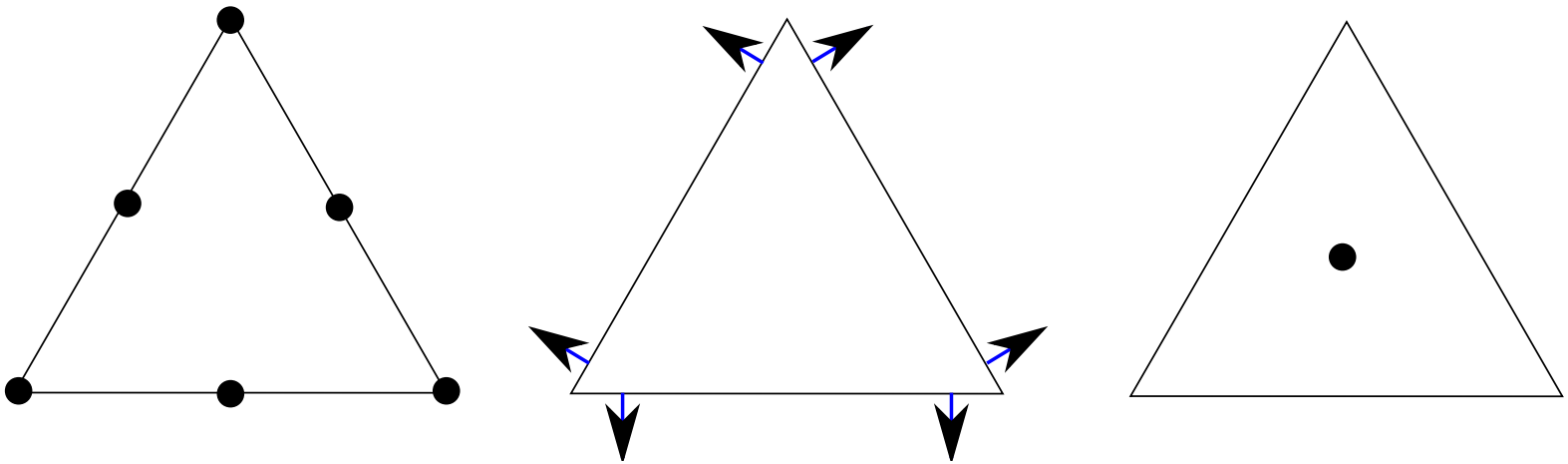
Requirements

1. $\nabla\cdot$ maps from $\mathbb{V}^1$ onto $\mathbb{V}^2$, and $\nabla^\perp$ maps from $\mathbb{V}^0$ onto kernel of $\nabla\cdot$ in $\mathbb{V}^1$.
2. Commuting, bounded surjective projections π_i exist.

Arnold, Falk and Winther, Acta Numerica (2006).

Compatible FE spaces

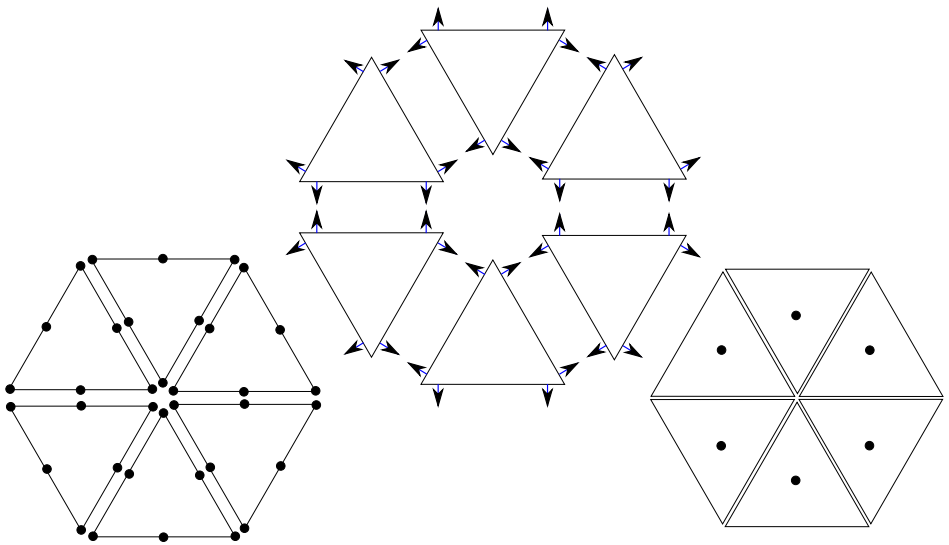
$$\underbrace{\mathbb{V}_0 = P2}_{\text{Quadratic, Continuous}} \xrightarrow{\nabla^\perp} \underbrace{\mathbb{V}_1 = BDM1}_{\text{Linear, Continuous normals}} \xrightarrow{\nabla\cdot} \underbrace{\mathbb{V}_2 = P0}_{\text{Constant, Discontinuous}}$$



Brezzi, Douglas and Marini (1985)

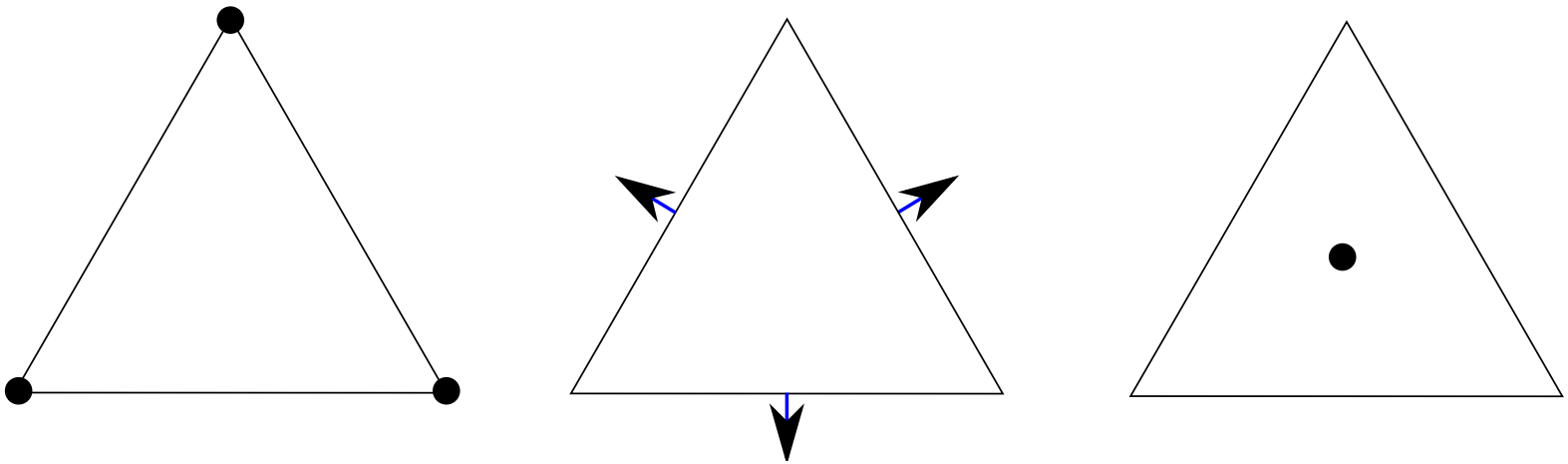
Compatible FE spaces

$$\underbrace{\mathbb{V}_0 = P2}_{\text{Quadratic, Continuous}} \xrightarrow{\nabla^\perp} \underbrace{\mathbb{V}_1 = BDM1}_{\text{Linear, Continuous normals}} \xrightarrow{\nabla\cdot} \underbrace{\mathbb{V}_2 = P0}_{\text{Constant, Discontinuous}}$$



Compatible FE spaces

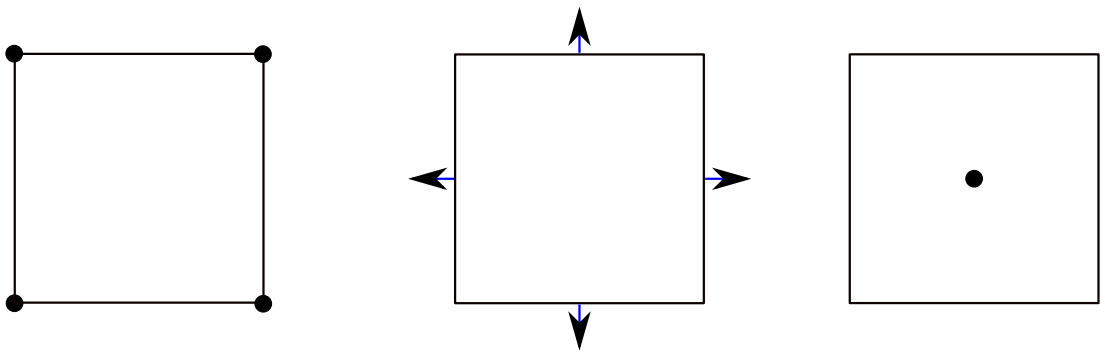
$$\underbrace{\mathbb{V}_0 = P1}_{\text{Linear, Continuous}} \xrightarrow{\nabla^\perp} \underbrace{\mathbb{V}_1 = RT0}_{\text{Constant (+1 Linear), Cont. normals}} \xrightarrow{\nabla\cdot} \underbrace{\mathbb{V}_2 = P0}_{\text{Constant, Discontinuous}}$$



Raviart and Thomas (1977)

Compatible FE spaces

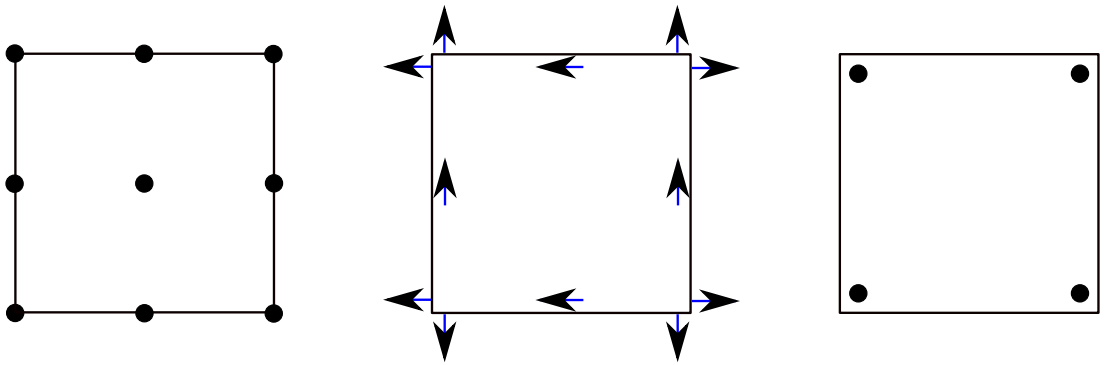
$$\underbrace{\mathbb{V}_0 = Q1}_{\text{Bilinear Continuous}} \xrightarrow{\nabla^\perp} \underbrace{\mathbb{V}_1 = RT0}_{\text{Constant/Linear, Cont. normals}} \xrightarrow{\nabla\cdot} \underbrace{\mathbb{V}_2 = Q0_{DG}}_{\text{Constant, Discontinuous}}$$



Raviart and Thomas (1977)

Compatible FE spaces

$$\underbrace{\mathbb{V}_0 = Q2}_{\text{Biquadratic Continuous}} \xrightarrow{\nabla^\perp} \underbrace{\mathbb{V}_1 = RT1}_{\text{Bilinear/Biquadratic, Cont. normals}} \xrightarrow{\nabla\cdot} \underbrace{\mathbb{V}_2 = Q1_{DG}}_{\text{Bilinear, Discontinuous}}$$



Raviart and Thomas (1977)

Helmholtz decomposition

Helmholtz decomposition (Arnold, Falk and Winther (2006))

Any $\boldsymbol{u} \in \mathbb{V}_1$ can be uniquely written as

$$\boldsymbol{u} = \nabla^\perp \psi + \mathfrak{h} + \tilde{\nabla} \phi,$$

with $\psi \in \mathbb{V}_0$, $\mathfrak{h} \in \mathfrak{h}_1$, $\phi \in \mathbb{V}_2$,

$$\mathfrak{h}_1 = \{\mathfrak{h} \in \mathbb{V}_1 : \nabla \cdot \mathfrak{h} = \tilde{\nabla}^\perp \cdot \mathfrak{h} = 0\}.$$

Orthogonality:

$$\langle \nabla^\perp \gamma, \mathfrak{h} \rangle = \langle \nabla^\perp \gamma, \tilde{\nabla} \beta \rangle = \langle \tilde{\nabla} \beta, \mathfrak{h} \rangle = 0, \quad \forall \gamma \in \mathbb{V}_1,\ \mathfrak{h} \in \mathfrak{h}_1,\ \beta \in \mathbb{V}_2.$$

L^2 inner products: $\langle \boldsymbol{u}, \boldsymbol{v} \rangle = \int_\Omega \boldsymbol{u} \cdot \boldsymbol{v} \, \mathrm{d}x$, $\langle \phi, \beta \rangle = \int_\Omega \phi \beta \, \mathrm{d}x$.

Building a finite element discretisation of the shallow water equations

$$\boldsymbol{u}_t + f\boldsymbol{u}^{\perp} + g\nabla h = 0,$$
$$h_t + H_0\nabla\cdot\boldsymbol{u} = 0.$$

Building a finite element discretisation of the shallow water equations

$$\boldsymbol{u}_t + f\boldsymbol{u}^{\perp} + g\nabla h = 0,$$
$$h_t + H_0 \nabla \cdot \boldsymbol{u} = 0.$$

Mixed finite element discretisation: seek $\boldsymbol{u}_h \in \mathbb{V}_1$, $D_h \in \mathbb{V}_2$ s.t.

$$\begin{aligned} \langle \boldsymbol{w}, (\boldsymbol{u}_h)_t \rangle + \langle \boldsymbol{w}, f\,(\boldsymbol{u}_h)^{\perp} \rangle - \langle \nabla \cdot \boldsymbol{w}, gD_h \rangle &= 0, \quad \forall \boldsymbol{w} \in \mathbb{V}_1, \\ \langle \gamma, (D_h)_t \rangle + \langle \gamma, H_0 \nabla \cdot \boldsymbol{u}_h \rangle &= 0, \quad \forall \gamma \in \mathbb{V}_2. \end{aligned}$$

Building a finite element discretisation of the shallow water equations

$$\boldsymbol{u}_t + f\boldsymbol{u}^\perp + g\nabla h = 0,$$
$$h_t + H_0\nabla\cdot\boldsymbol{u} = 0.$$

Mixed finite element discretisation: seek $\boldsymbol{u}_h \in \mathbb{V}_1$, $D_h \in \mathbb{V}_2$ s.t.

$$\begin{aligned}\langle \boldsymbol{w}, (\boldsymbol{u}_h)_t\rangle + \langle \boldsymbol{w}, f\,(\boldsymbol{u}_h)^\perp\rangle - \langle \nabla\cdot\boldsymbol{w}, gD_h\rangle &= 0, \quad \forall\, \boldsymbol{w}\in\mathbb{V}_1,\\ \langle \gamma, (D_h)_t\rangle + \langle \gamma, H_0\nabla\cdot\boldsymbol{u}_h\rangle &= 0, \quad \forall\,\gamma\in\mathbb{V}_2.\end{aligned}$$

See Natale, Shipton and CJC (2016) for survey of: energy conservation (from antisymmetry), local mass conservation (compatibility), inf-sup condition (compatibility), correct number of harmonic functions (compatibility$\star$), exact geostrophic balance (compatibility, CJC and Shipton (2012)).

Implementing the discretisation on a computer: 1

Seek $\boldsymbol{u}_h \in \mathbb{V}_1$, $D_h \in \mathbb{V}_2$ s.t.

$$\begin{aligned} \langle \boldsymbol{w}, (\boldsymbol{u}_h)_t \rangle + \langle \boldsymbol{w}, f\,(\boldsymbol{u}_h)^{\perp} \rangle - \langle \nabla \cdot \boldsymbol{w}, g D_h \rangle &= 0, \quad \forall \boldsymbol{w} \in \mathbb{V}_1, \\ \langle \gamma, (D_h)_t \rangle + \langle \gamma, H_0 \nabla \cdot \boldsymbol{u}_h \rangle &= 0, \quad \forall \gamma \in \mathbb{V}_2. \end{aligned}$$

Implementing the discretisation on a computer: 1

Seek $\boldsymbol{u}_h \in \mathbb{V}_1$, $D_h \in \mathbb{V}_2$ s.t.

$$\begin{aligned} \langle \boldsymbol{w}, (\boldsymbol{u}_h)_t \rangle + \langle \boldsymbol{w}, f\,(\boldsymbol{u}_h)^{\perp} \rangle - \langle \nabla \cdot \boldsymbol{w}, g D_h \rangle &= 0, \quad \forall \boldsymbol{w} \in \mathbb{V}_1, \\ \langle \gamma, (D_h)_t \rangle + \langle \gamma, H_0 \nabla \cdot \boldsymbol{u}_h \rangle &= 0, \quad \forall \gamma \in \mathbb{V}_2. \end{aligned}$$

Basis expansions: $\boldsymbol{u}_h = \sum_{i=1}^{N_1} u_i \boldsymbol{N}_i(x)$, $\boldsymbol{w}_h = \sum_{i=1}^{N_1} w_i \boldsymbol{N}_i(x)$, $D_h = \sum_{i=1}^{N_1} D_i P_i(x)$, $\gamma_h = \sum_{i=1}^{N_1} g_i P_i(x)$ lead to

Implementing the discretisation on a computer: 1

Seek $\boldsymbol{u}_h \in \mathbb{V}_1$, $D_h \in \mathbb{V}_2$ s.t.

$$\begin{aligned} \langle \boldsymbol{w}, (\boldsymbol{u}_h)_t \rangle + \langle \boldsymbol{w}, f\,(\boldsymbol{u}_h)^\perp \rangle - \langle \nabla \cdot \boldsymbol{w}, g D_h \rangle &= 0, \quad \forall \boldsymbol{w} \in \mathbb{V}_1, \\ \langle \gamma, (D_h)_t \rangle + \langle \gamma, H_0 \nabla \cdot \boldsymbol{u}_h \rangle &= 0, \quad \forall \gamma \in \mathbb{V}_2. \end{aligned}$$

Basis expansions: $\boldsymbol{u}_h = \sum_{i=1}^{N_1} u_i \boldsymbol{N}_i(x)$, $\boldsymbol{w}_h = \sum_{i=1}^{N_1} w_i \boldsymbol{N}_i(x)$, $D_h = \sum_{i=1}^{N_1} D_i P_i(x)$, $\gamma_h = \sum_{i=1}^{N_1} g_i P_i(x)$ lead to

$$\begin{pmatrix} M_1 & 0 \\ 0 & M_2 \end{pmatrix} \begin{pmatrix} \dot{\mathbf{u}} \\ \dot{\mathbf{D}} \end{pmatrix} + \begin{pmatrix} K & -gC^T \\ H_0 C & 0 \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \mathbf{D} \end{pmatrix} = 0.$$

$$(M_1)_{ij} = \int_\Omega \boldsymbol{N}_i(x) \cdot \boldsymbol{N}_j(x)\,\mathrm{d}x, \; C_{ij} = \int_\Omega P_i(x) \nabla \cdot \boldsymbol{N}_j(x)\,\mathrm{d}x,$$

$$(M_2)_{ij} = \int_\Omega P_i(x) P_j(x)\,\mathrm{d}x, \; K_{ij} = \int_\Omega f\,\boldsymbol{N}_i(x) \cdot \boldsymbol{N}_j(x)^\perp\,\mathrm{d}x.$$

Implementing the discretisation on a computer: 2

Building matrices e.g. $M_{ij} = \int_\Omega \boldsymbol{N}_i \cdot \boldsymbol{N}_j \, \mathrm{d}x$.

Elemental assembly by numerical quadrature

$(M_1)_{ij} = \sum_{e \in \Omega} \int_e \boldsymbol{N}_i \cdot \boldsymbol{N_j} \, \mathrm{d}x$.

Implementing the discretisation on a computer: 2

Building matrices e.g. $M_{ij} = \int_\Omega \boldsymbol{N}_i \cdot \boldsymbol{N}_j \, \mathrm{d}x$.

Elemental assembly by numerical quadrature

$(M_1)_{ij} = \sum_{e \in \Omega} \int_e \boldsymbol{N}_i \cdot \boldsymbol{N_j} \, \mathrm{d}x$.

Sparsity

M_1 is SPARSE. Use a sparse matrix format.
On any given cell e, almost all basis functions are zero. To build M_1, don't loop over i, j. Instead, loop over cells, incrementing the entries in M_1 relevant to each cell.

Implementing the discretisation on a computer: 2

Building matrices e.g. $M_{ij} = \int_\Omega \boldsymbol{N}_i \cdot \boldsymbol{N}_j \, \mathrm{d}x$.

Elemental assembly by numerical quadrature

$(M_1)_{ij} = \sum_{e \in \Omega} \int_e \boldsymbol{N}_i \cdot \boldsymbol{N_j} \, \mathrm{d}x$.

Sparsity

M_1 is SPARSE. Use a sparse matrix format.
On any given cell e, almost all basis functions are zero. To build M_1, don't loop over i, j. Instead, loop over cells, incrementing the entries in M_1 relevant to each cell.

For more details head to `https://finite-element.github.io`

Imperial College
London

Advances in compatible finite element methods since 2013

Firedrake: https://firedrakeproject.org

- Runtime automated code generation from weak forms described in UFL (a DSL expressed as a Python library), e.g. `inner(w, u)*dx`

Firedrake: https://firedrakeproject.org

- Runtime automated code generation from weak forms described in UFL (a DSL expressed as a Python library), e.g. `inner(w, u)*dx`
- Runtime automated code generation for parallel matrix and form assembly.

Firedrake: https://firedrakeproject.org

- Runtime automated code generation from weak forms described in UFL (a DSL expressed as a Python library), e.g. `inner(w, u)*dx`
- Runtime automated code generation for parallel matrix and form assembly.
- Mesh data structures wrap PETSc DMPlex and parallel nonlinear/linear solvers via PETSc.

Firedrake: https://firedrakeproject.org

- Runtime automated code generation from weak forms described in UFL (a DSL expressed as a Python library), e.g. `inner(w, u)*dx`
- Runtime automated code generation for parallel matrix and form assembly.
- Mesh data structures wrap PETSc DMPlex and parallel nonlinear/linear solvers via PETSc.
- Automated adjoint via PyAdjoint.

Firedrake: https://firedrakeproject.org

- Runtime automated code generation from weak forms described in UFL (a DSL expressed as a Python library), e.g. `inner(w, u)*dx`
- Runtime automated code generation for parallel matrix and form assembly.
- Mesh data structures wrap PETSc DMPlex and parallel nonlinear/linear solvers via PETSc.
- Automated adjoint via PyAdjoint.
- Flexible support for Python preconditioners in PETSc, geometric multigrid.

Firedrake: https://firedrakeproject.org

- Runtime automated code generation from weak forms described in UFL (a DSL expressed as a Python library), e.g. `inner(w, u)*dx`
- Runtime automated code generation for parallel matrix and form assembly.
- Mesh data structures wrap PETSc DMPlex and parallel nonlinear/linear solvers via PETSc.
- Automated adjoint via PyAdjoint.
- Flexible support for Python preconditioners in PETSc, geometric multigrid.
- Domain specific languages (UFL, GEM, Loopy, Slate) allow machine specific optimisations (vectorisation, data-padding, loop inlining, loop-invariant code motion).

The success of Firedrake

Main take-home messages

1. It is always easier to generalise than write specific code.
2. Developer productivity is just as important as code efficiency.
3. If you need sparse parallel (non)linear solvers and you aren't using PETSc, you need your head examining.

https://firedrakeproject.org, https://firedrakeproject.org/gusto

Gibson, Thomas H., Andrew TT McRae, CJC, Lawrence Mitchell, and David A. Ham. Compatible Finite Element Methods for Geophysical Flows: Automation and Implementation Using Firedrake. Mathematics of Planet Earth Springer Briefs, 2019.

Vertical discretisation

Given 2D finite element spaces $(U_0 \overset{\nabla^\perp}{\to} U_1 \overset{\nabla\cdot}{\to} U_2)$ and 1D finite element spaces $(V_0 \overset{\partial_x}{\to} V_1)$, we can generate a product in three dimensions:

$$W_0 \xrightarrow{\nabla} W_1 \xrightarrow{\nabla\times} W_2 \xrightarrow{\nabla\cdot} W_3,$$

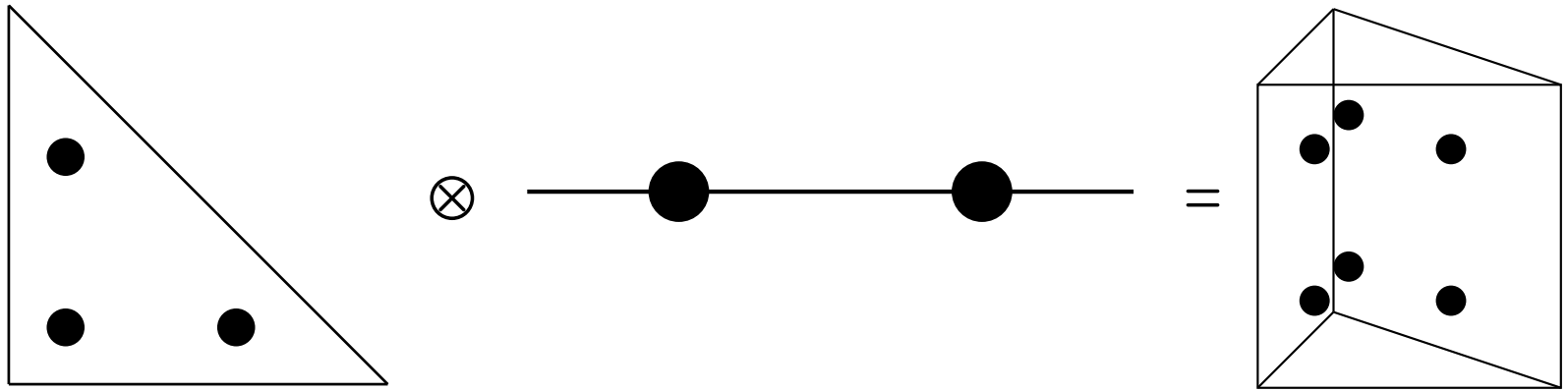
where

$$\begin{aligned}
W_0 &:= U_0 \otimes V_0, \\
W_1 &:= \texttt{HCurl}(U_0 \otimes V_1) \oplus \texttt{HCurl}(U_1 \otimes V_0), \\
W_2 &:= \texttt{HDiv}(U_1 \otimes V_1) \oplus \texttt{HDiv}(U_2 \otimes V_0), \\
W_3 &:= U_2 \otimes V_1,
\end{aligned}$$

with $W_0 \subset H^1, W_1 \subset H(\text{curl}), W_2 \subset H(\text{div}), W_3 \subset L^2$

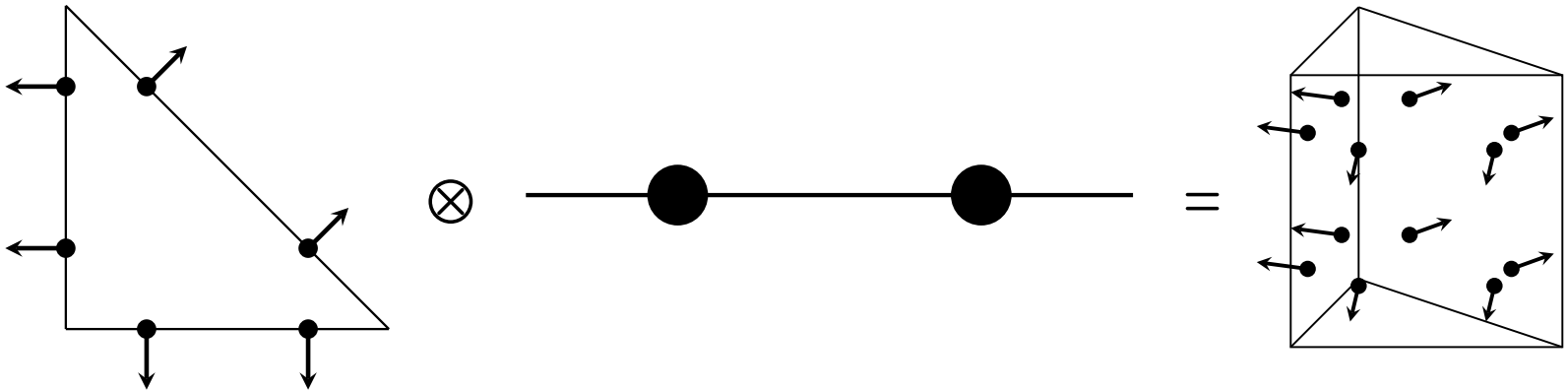
3D spaces

Density space



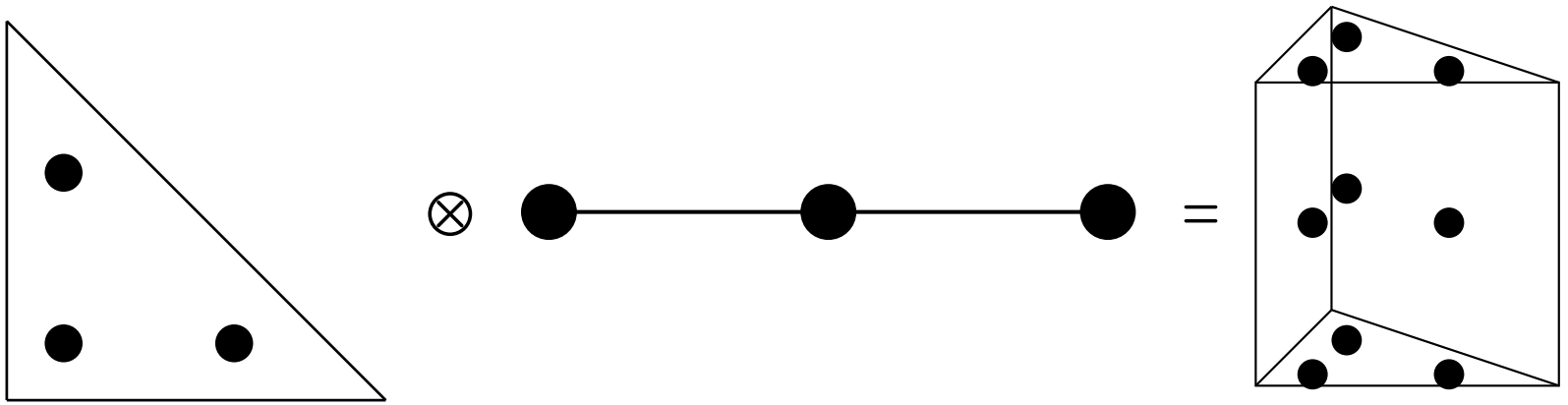
3D spaces

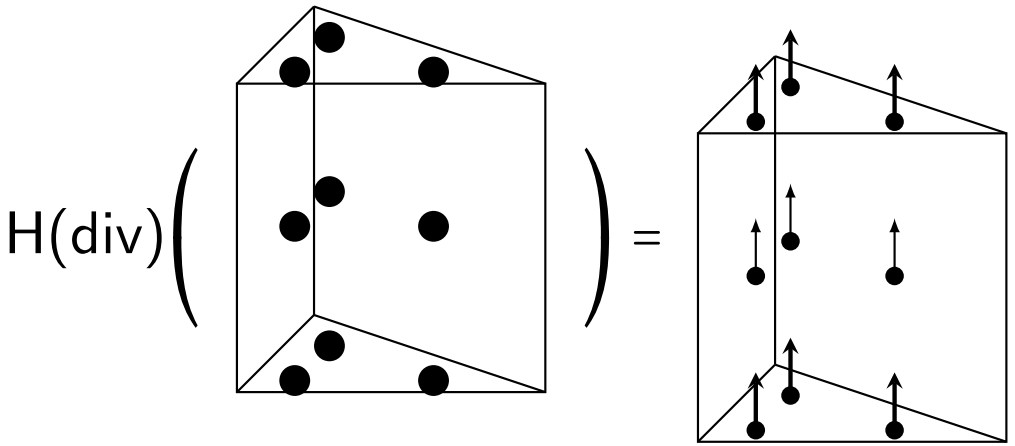
Horizontal part of velocity space



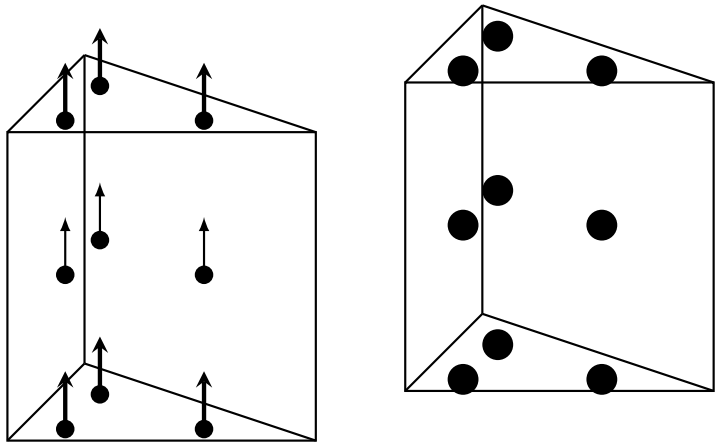
3D spaces

Vertical part of velocity space





Where to store temperature?

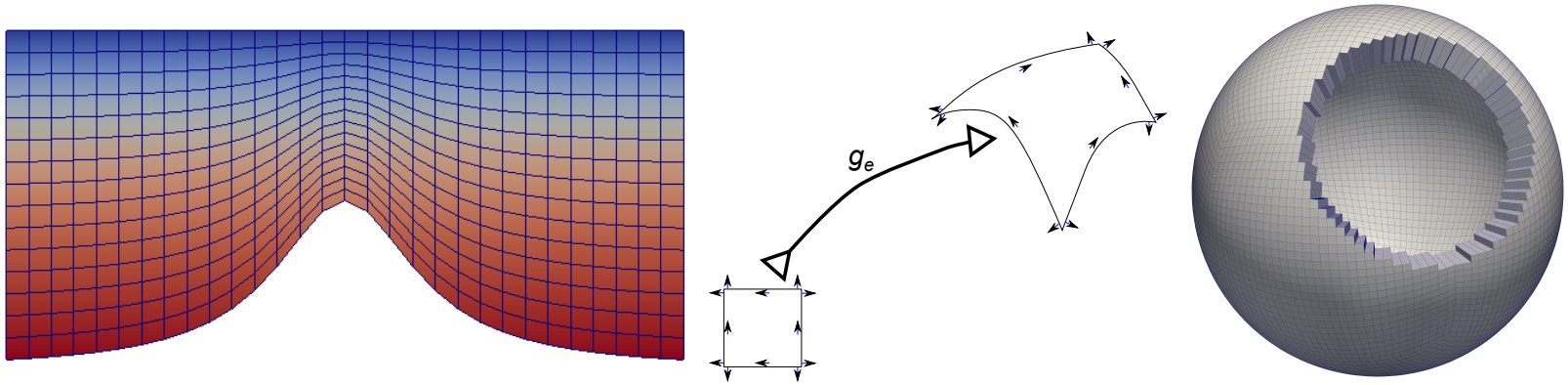


Charney-Phillips for FEM: Guerra and Ullrich (2016)

We use the same node locations for temperature as the vertical part of velocity.

Proof of exact hydrostatic balance: Natale, Shipton and CJC (2016).

Approximation theory



- On non-affine elements, compatible finite element spaces become non-polynomial.
- Cubed sphere quadrilateral meshes, higher-order sphere triangle meshes, topography, 3D global meshes all call for non-affine elements.

How does this impact approximation properties?

Approximation theory

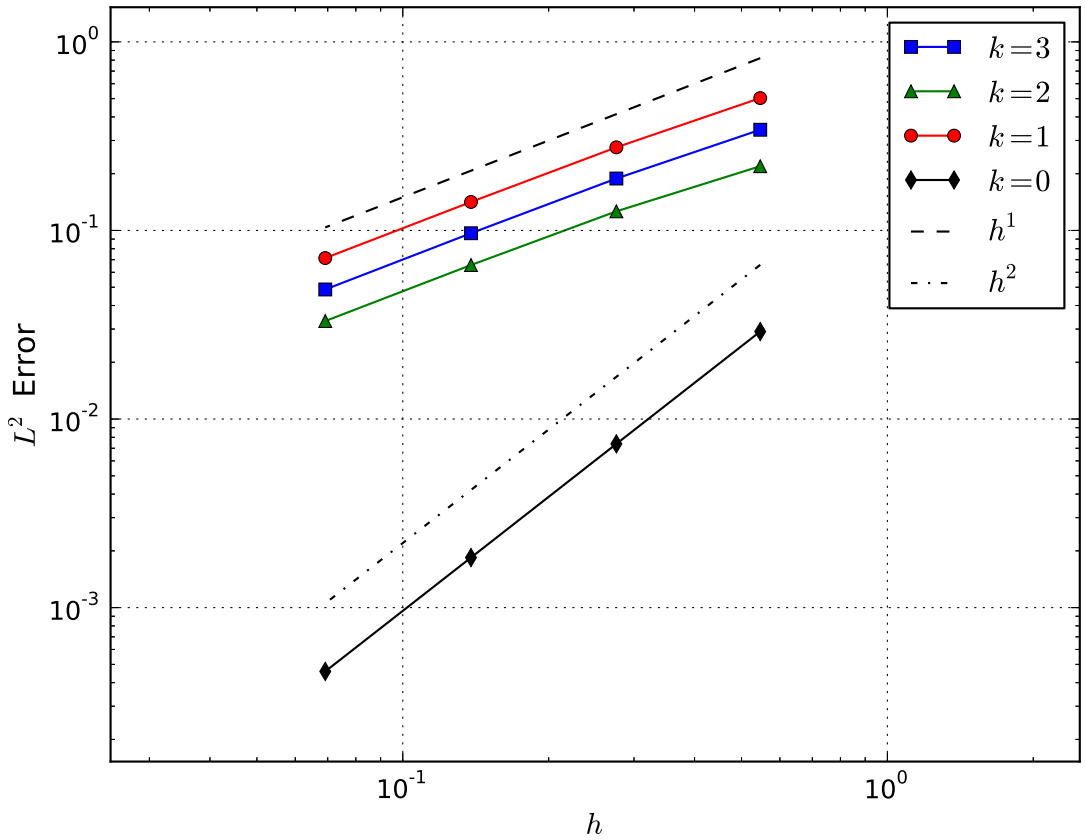
- Holst and Stern, FoCM (2012) showed that approximation order is not reduced provided that mesh is obtained from piecewise C^∞ global transformation from an affine mesh.
- The mapping $G : \Omega \to \mathbb{R}^4$ given by

$$\boldsymbol{x} = (x, y, z) \mapsto G(\boldsymbol{x}) = \left(\frac{x}{|\boldsymbol{x}|}, \frac{y}{|\boldsymbol{x}|}, \frac{z}{|\boldsymbol{x}|}, |\boldsymbol{x}| \right),$$

  defines a domain that can be meshed using affine elements.
- For (piecewise C^∞) terrain-following meshes, C can be composed with the global mapping to the spherical annulus.
- Similar construction works for cubed sphere.

A. Natale, Shipton and CJC (2016)

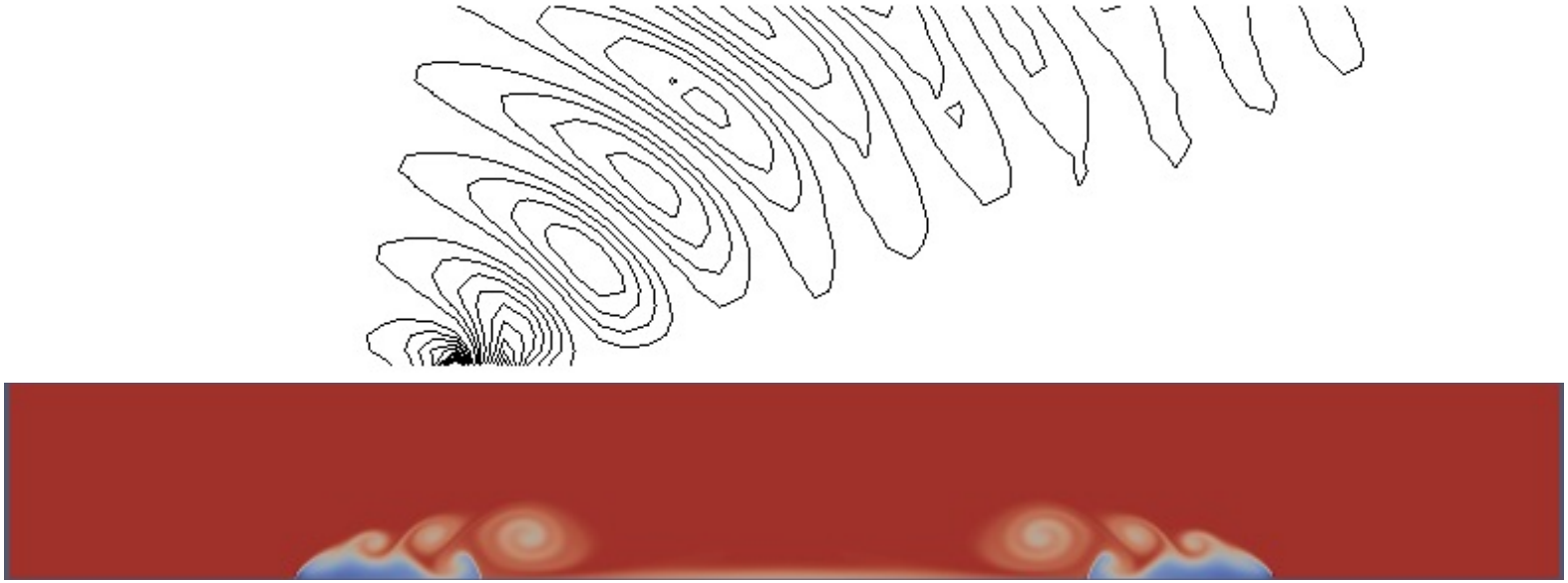
Approximation theory



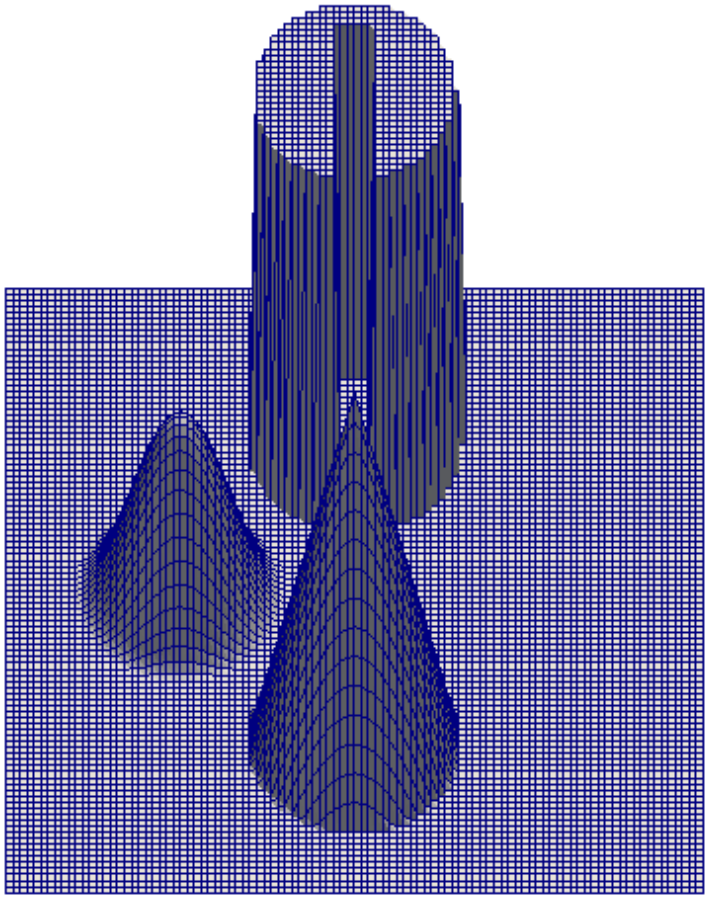
Nonlinear terms in rotating compressible Euler equations

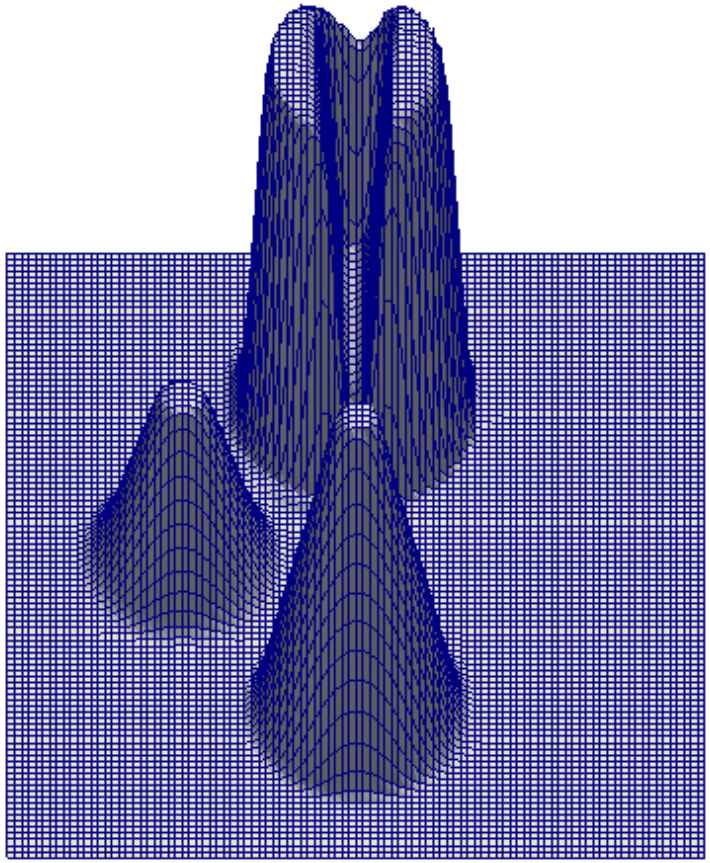
- Velocity advection: upwinded advective form (Guzmán et al, 2016), upwinded vector invariant /energy conserving form (Natale and CJC, 2016).
- Density advection: upwind discontinuous Galerkin method.
- Temperature advection: hybrid SUPG/discontinuous Galerkin method (Natale, Shipton and CJC (2016), embedded discontinuous Galerkin method (CJC and Kuzmin (2016)).
- All fields (lowest order): recovered space schemes (Bendall, CJC and Shipton (2019)).
- Nonlinear pressure gradient term ($\theta\nabla\Pi$): Natale, Shipton and CJC (2016).

Vertical slices

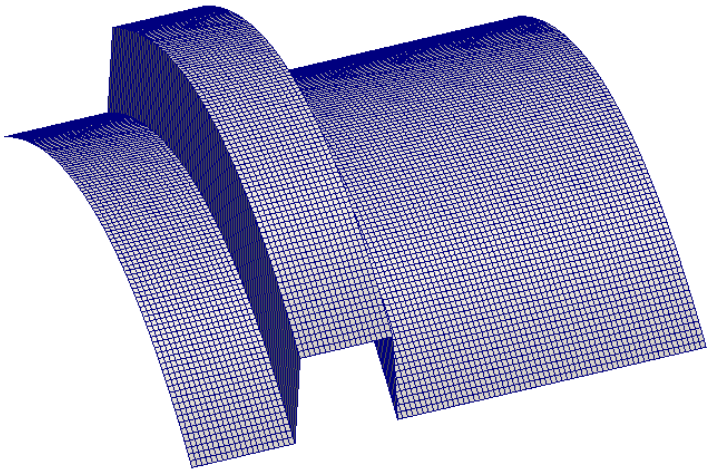


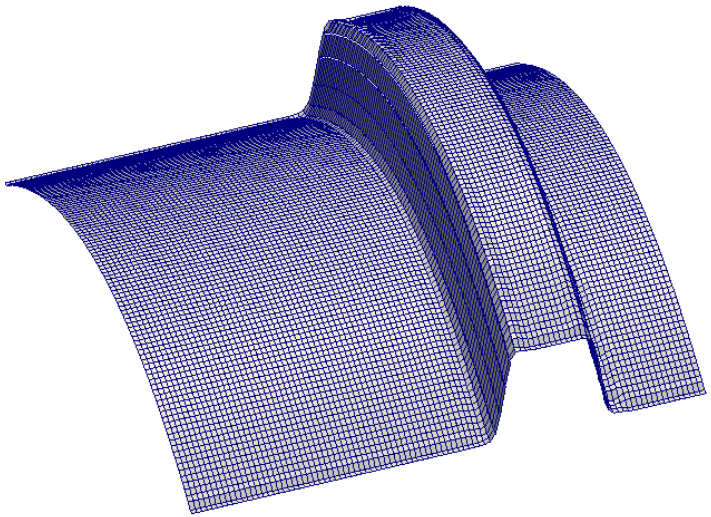
Bounded temperature space transport





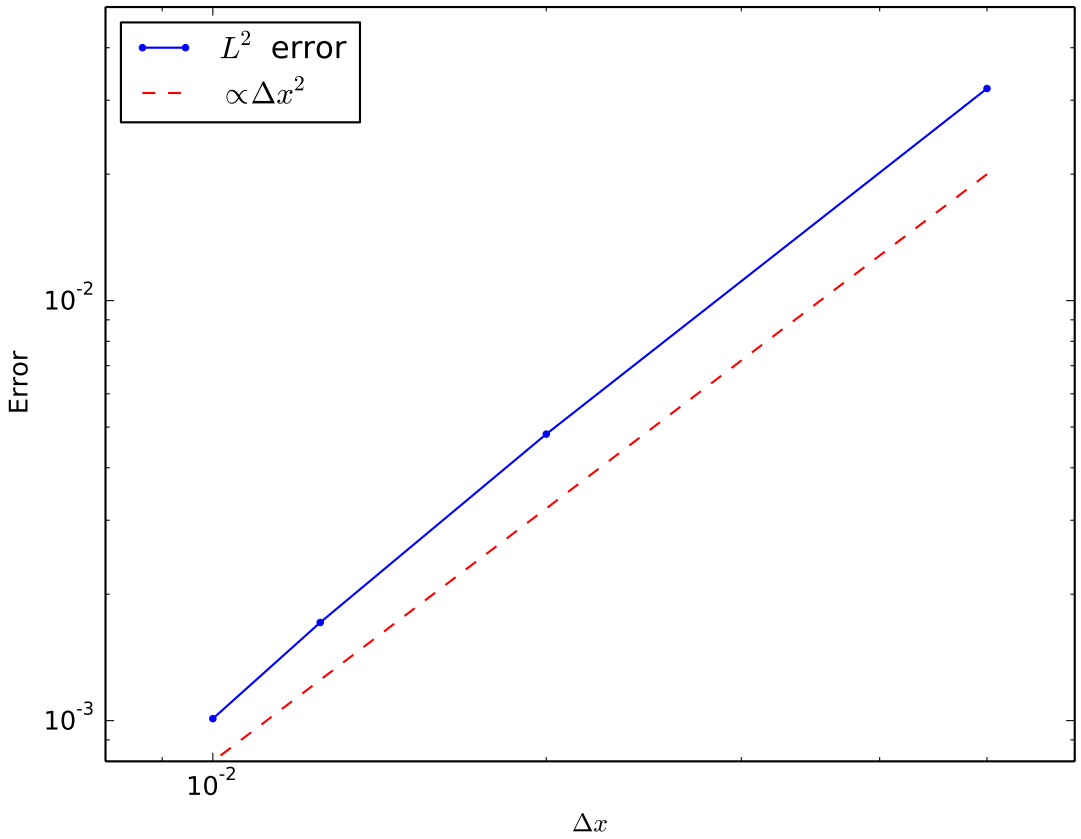
Bounded temperature space transport



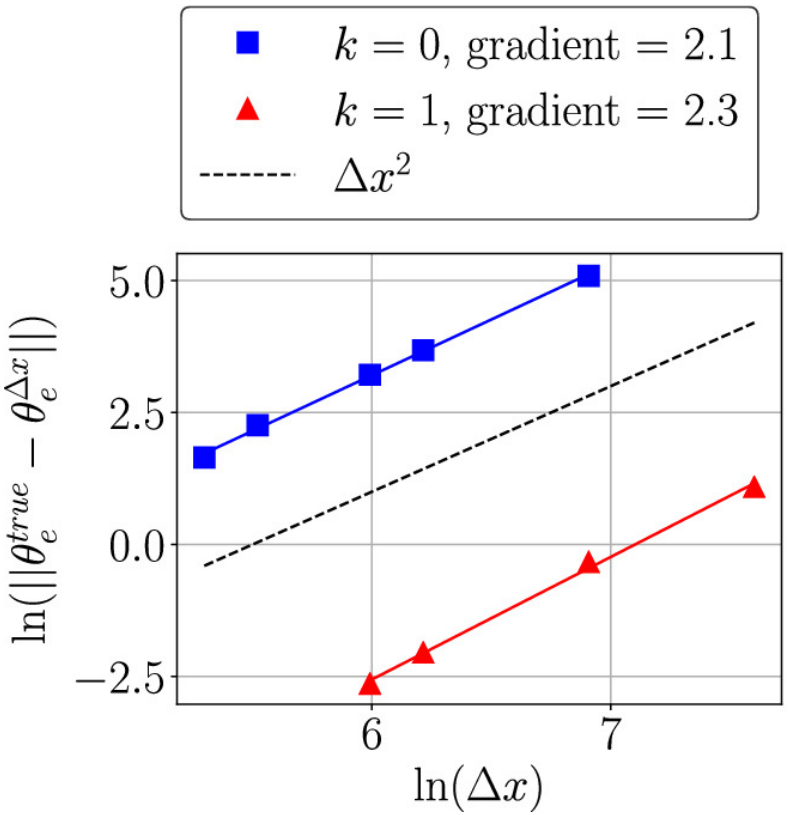


CJC and Kuzmin (2016).

Bounded temperature space transport

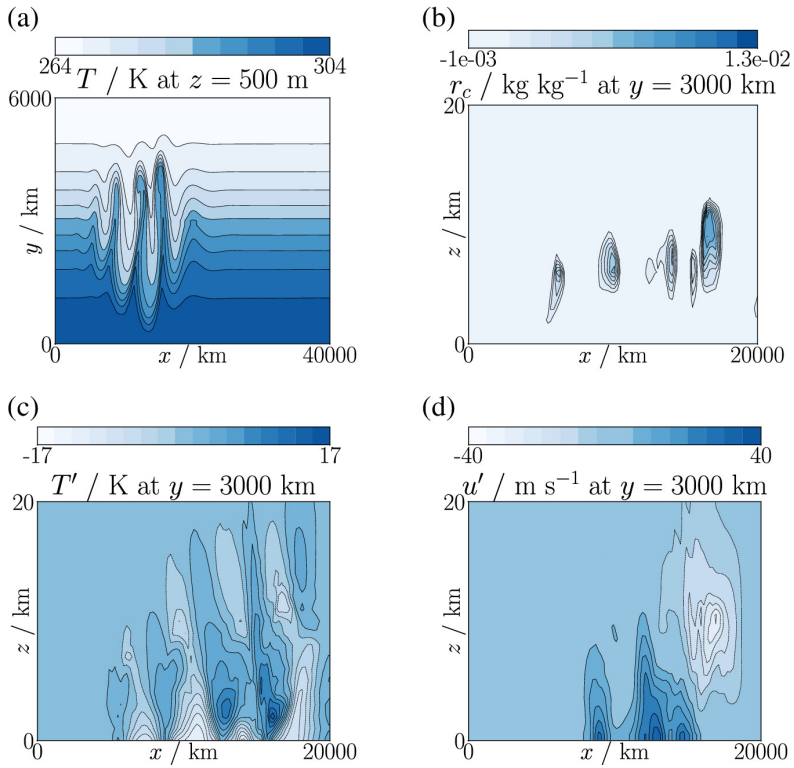


Adding moisture



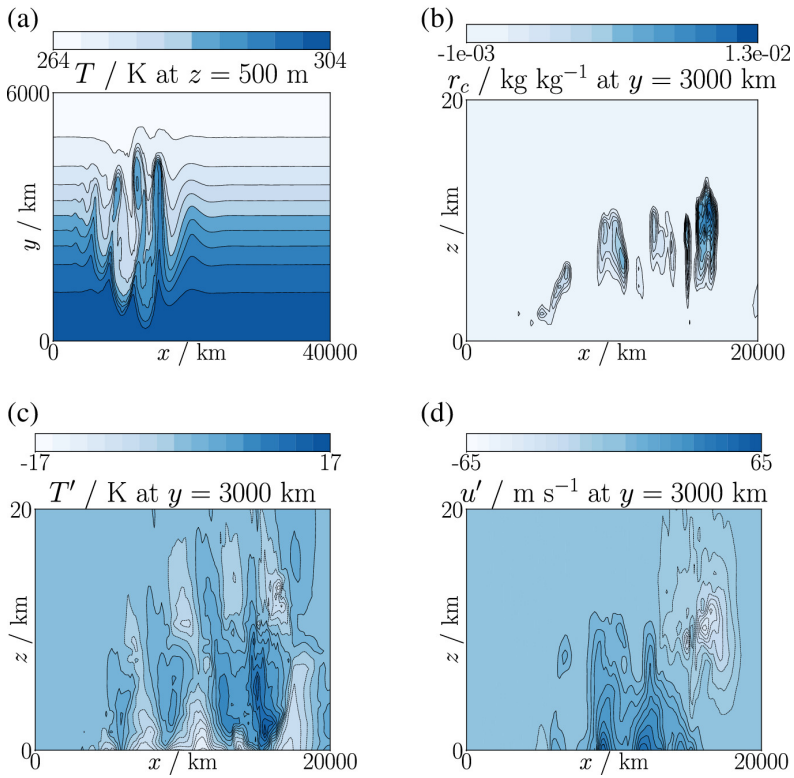
- Tracer variables in temperature space
- Conservation possible (but not yet implemented).
- Lowest order spaces (with remapped transport) and next-to-lowest order spaces (with embedded DG transport).
- Simple cloud moisture/air moisture scheme (see Bendall et al (2020))

Moist baroclinic wave (lowest order)



12 days, $\Delta x = \Delta y$ =200km, Δz =1 km, Δt = 300s.

Moist baroclinic wave (next-to-lowest order)



12 days, $\Delta x = \Delta y$ =250km, Δz =1.5 km, Δt = 300s.

Iterative solvers

Recall the linear SWE discretisation: Seek $\boldsymbol{u}_h \in \mathbb{V}_1$, $D_h \in \mathbb{V}_2$ s.t.

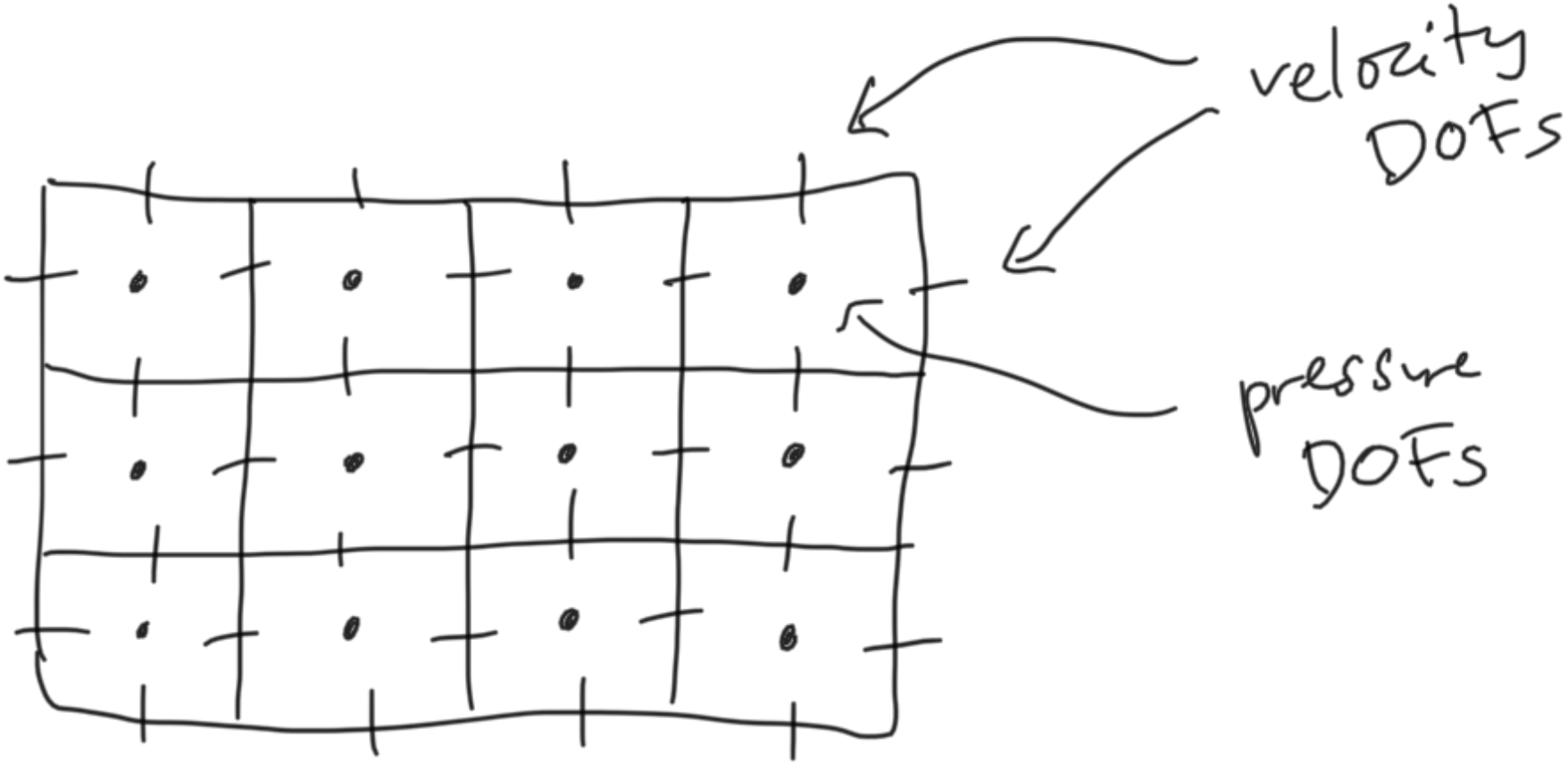
$$\begin{aligned}
\langle \boldsymbol{w}, (\boldsymbol{u}_h)_t \rangle + \langle \boldsymbol{w}, f\,(\boldsymbol{u}_h)^\perp \rangle - \langle \nabla \cdot \boldsymbol{w}, g D_h \rangle &= 0, \quad \forall\, \boldsymbol{w} \in \mathbb{V}_1, \\
\langle \gamma, (D_h)_t \rangle + \langle \gamma, H_0 \nabla \cdot \boldsymbol{u}_h \rangle &= 0, \quad \forall\, \gamma \in \mathbb{V}_2.
\end{aligned}$$

$$\begin{pmatrix} M_1 & 0 \\ 0 & M_2 \end{pmatrix} \begin{pmatrix} \dot{\mathbf{u}} \\ \dot{\mathbf{D}} \end{pmatrix} + \begin{pmatrix} K & -gC^T \\ H_0 C & 0 \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \mathbf{D} \end{pmatrix} = 0.$$

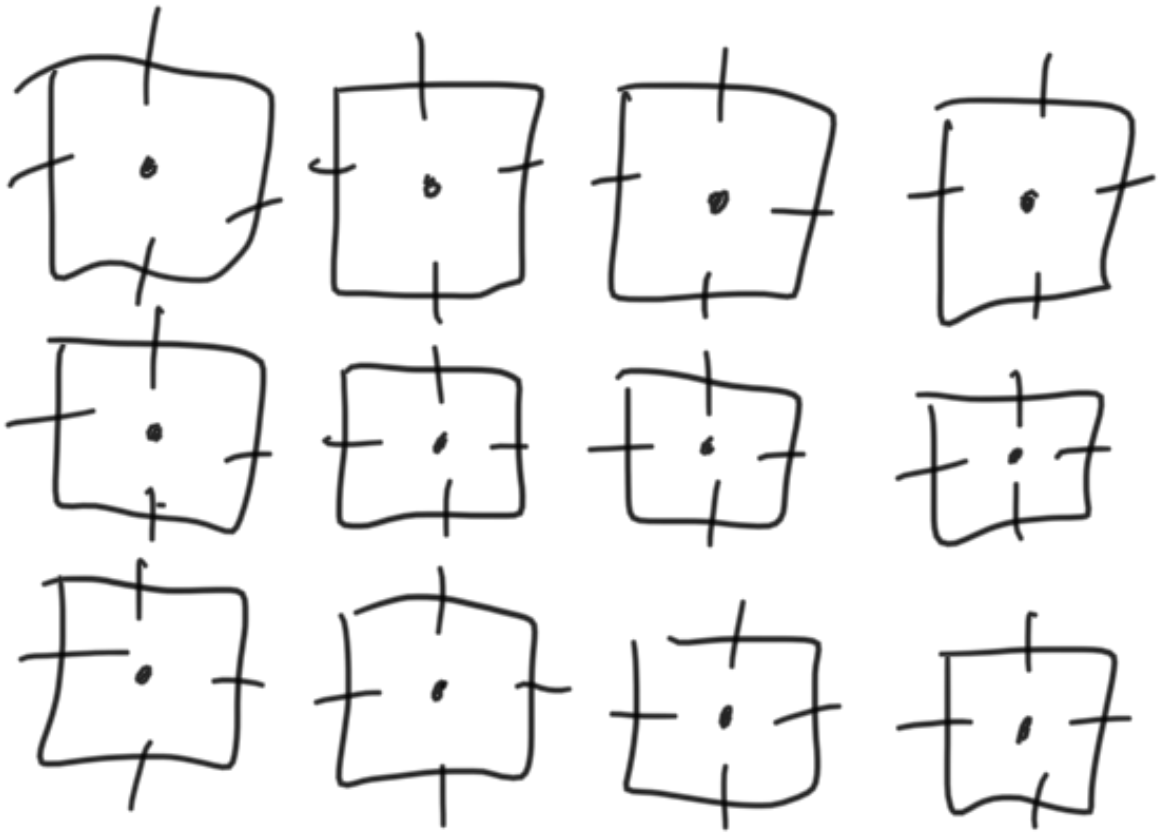
$$(M_1)_{ij} = \int_\Omega \boldsymbol{N}_i(x) \cdot \boldsymbol{N}_j(x)\,\mathrm{d}x,\; C_{ij} = \int_\Omega P_i(x) \nabla \cdot \boldsymbol{N}_j(x)\,\mathrm{d}x,$$

$$(M_2)_{ij} = \int_\Omega P_i(x) P_j(x)\,\mathrm{d}x,\; K_{ij} = \int_\Omega f\,\boldsymbol{N}_i(x) \cdot \boldsymbol{N}_j(x)^\perp\,\mathrm{d}x.$$

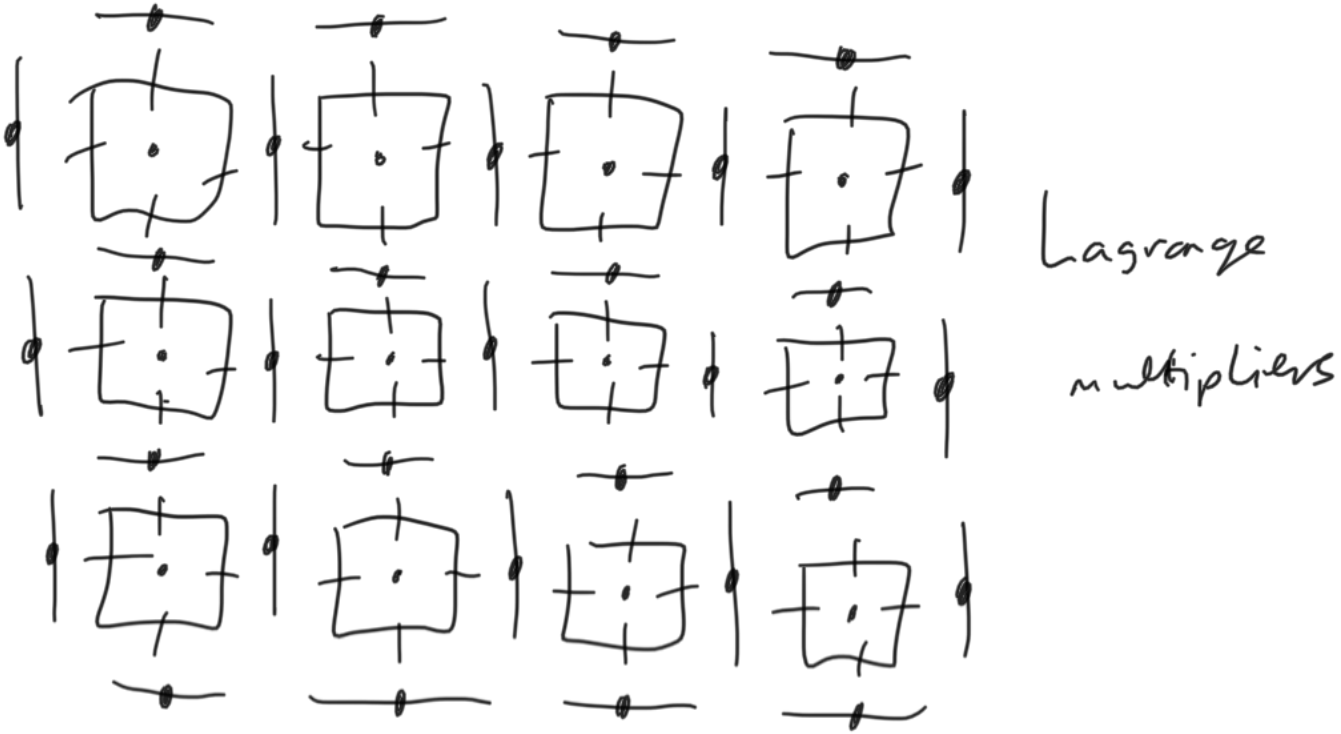
Finite element spaces



Breaking the continuity ...



... and fixing it again



Eliminate $\boldsymbol{u}$ and D to get sparse system for the Lagrange multipliers λ (hybridisation).

Non-hybridisable formulation for rotating compressible Euler

For Picard iteration using linearisation about state of rest.

From (Natale, Shipton and CJC (2016), Melvin et al (2019)).

$$
\begin{aligned}
&\langle \boldsymbol{w}, \boldsymbol{u}' \rangle - \Delta t \langle (\nabla_h)_V \cdot (\boldsymbol{w}\theta'), \bar{\Pi} \rangle \\
&-\Delta t \langle \nabla_h \cdot (\boldsymbol{w}\bar{\theta}), \Pi' \rangle + \Delta t \left\langle\!\!\left\langle \left[\!\left[\boldsymbol{w}\bar{\theta} \right]\!\right] \{\Pi'\} \right\rangle\!\!\right\rangle = \ldots, \qquad \forall\, \boldsymbol{w} \in \mathbb{W}_2, \\
&\langle \gamma, \theta' \rangle + \Delta t \langle \gamma, \boldsymbol{u}' \cdot \hat{\boldsymbol{z}} N^2 \rangle = \ldots, \qquad \forall\, \gamma \in \mathbb{W}_\theta, \\
&\langle \phi, \rho' \rangle - \Delta t \langle \nabla_h \phi, \boldsymbol{u}' \rangle + \Delta t \left\langle\!\!\left\langle \left[\!\left[\phi \boldsymbol{u}' \right]\!\right] \{\bar{\rho}\} \right\rangle\!\!\right\rangle = \ldots, \qquad \forall\, \phi \in \mathbb{W}_3.
\end{aligned}
$$

Hybridisable formulation

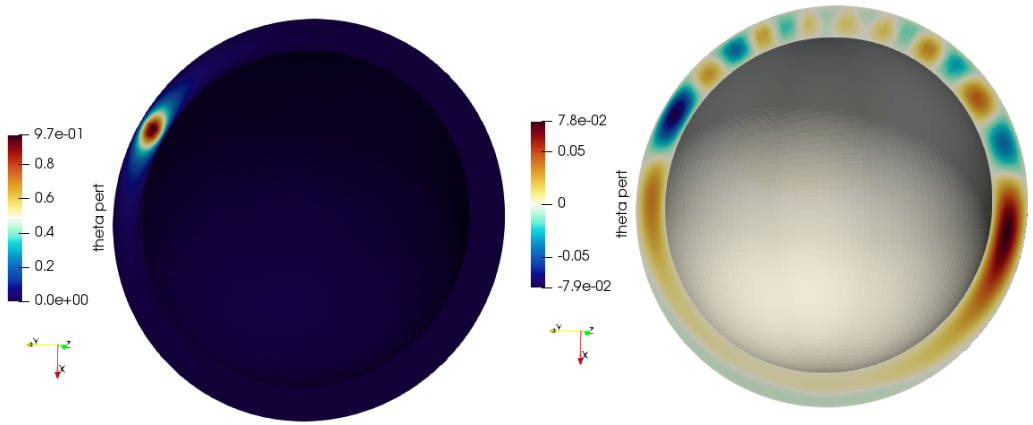
For Picard iteration using linearisation about state of rest.

From Thomas Gibson's PhD thesis (2019).

$$
\begin{aligned}
\langle \boldsymbol{w}, \boldsymbol{u}' \rangle + \Delta t \langle \boldsymbol{w}, 2\Omega \times \boldsymbol{u}' \rangle & \\
-\Delta t \langle (\nabla_h)_V \cdot (\boldsymbol{w}\theta'), \bar{\Pi} \rangle + \Delta t \left\langle\!\!\left\langle \left[\!\left[\boldsymbol{w}_V \theta' \right]\!\right] \{\bar{\Pi}\} \right\rangle\!\!\right\rangle & \\
-\Delta t \langle \nabla_h \cdot (\boldsymbol{w}\bar{\theta}), \Pi' \rangle + \Delta t \left\langle\!\!\left\langle \left[\!\left[\boldsymbol{w}\bar{\theta} \right]\!\right] \lambda \right\rangle\!\!\right\rangle &= \ldots, \quad \forall\, \boldsymbol{w} \in \hat{\mathbb{W}}_2, \\
\langle \gamma, \theta' \rangle + \Delta t \langle \gamma, \boldsymbol{u}' \cdot \hat{\boldsymbol{z}} N^2 \rangle &= \ldots, \quad \forall\, \gamma \in \mathbb{W}_\theta, \\
\langle \phi, \rho' \rangle - \Delta t \langle \nabla_h \phi, \boldsymbol{u}' \rangle + \Delta t \left\langle\!\!\left\langle \left[\!\left[\phi \boldsymbol{u}' \right]\!\right] \{\bar{\rho}\} \right\rangle\!\!\right\rangle &= \ldots, \quad \forall\, \phi \in \mathbb{W}_3, \\
\left\langle\!\!\left\langle \left[\!\left[\boldsymbol{u}'\bar{\theta} \right]\!\right], \mu \right\rangle\!\!\right\rangle &= 0, \quad \forall\, \mu \in Tr.
\end{aligned}
$$

Hybridisation test case

DCMIP 3.1, 6144 horizontal quads/5120 triangles, 64 layers, $\Delta t = 100$s, no advection. AMG on reduced system with scaled Richardson/ILU smoothing.

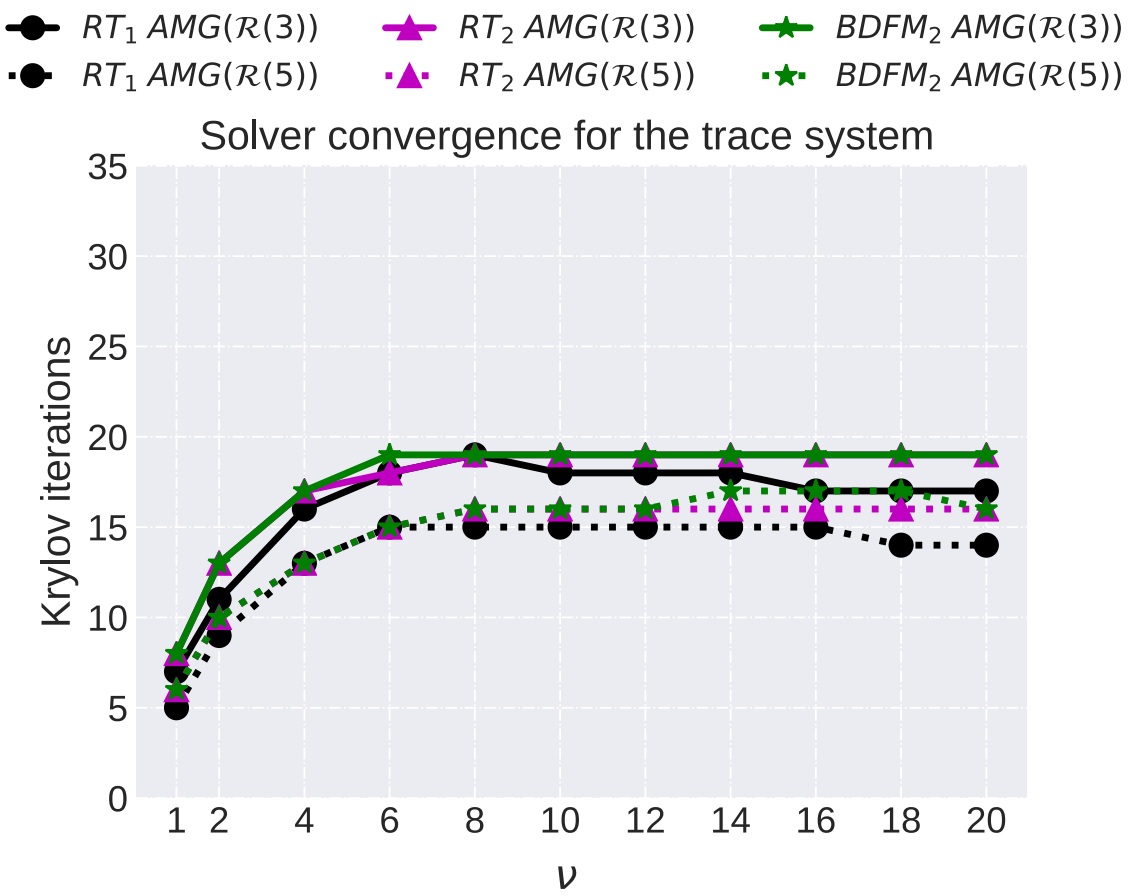


(A) Initial temperature perturbation.

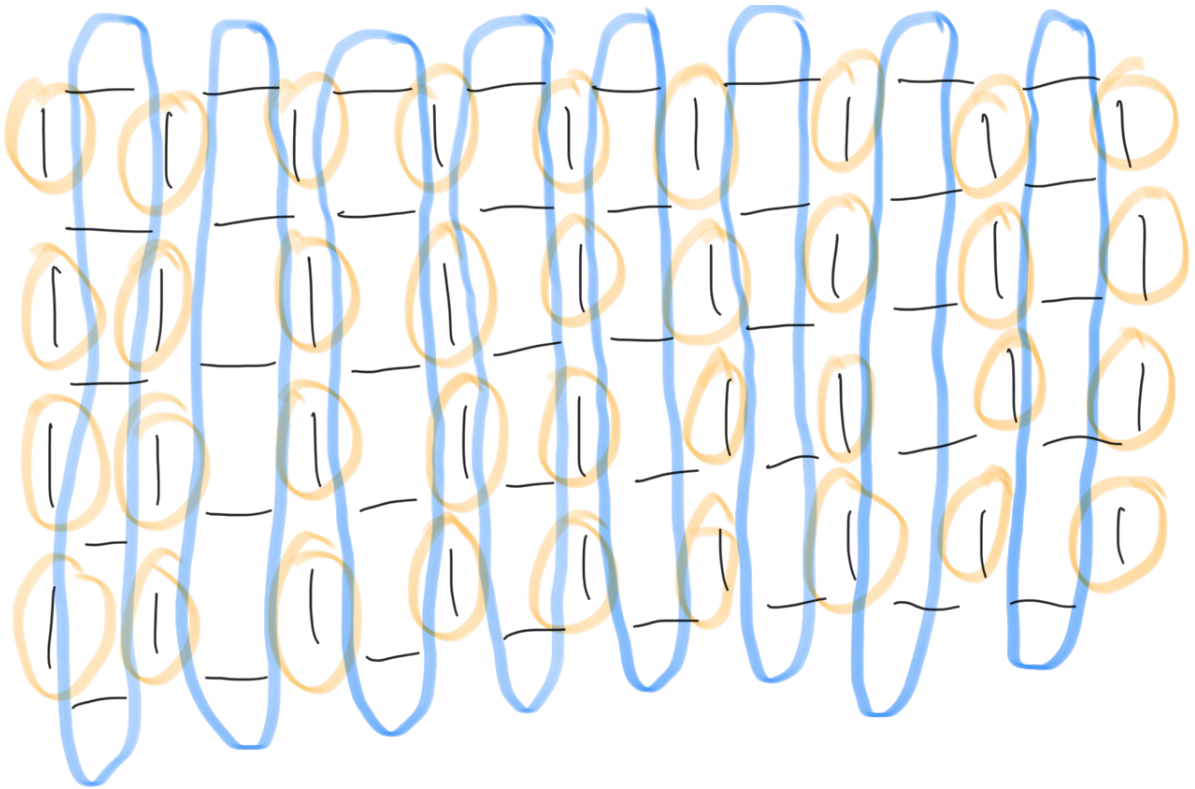
(B) Evolution of the potential temperature at $t = 3000$ s.

FIGURE 5.8: Plots of the potential temperature perturbation at $t = 0$ s and $t = 3000$ s for the non-orographic gravity wave test on a condensed planet.

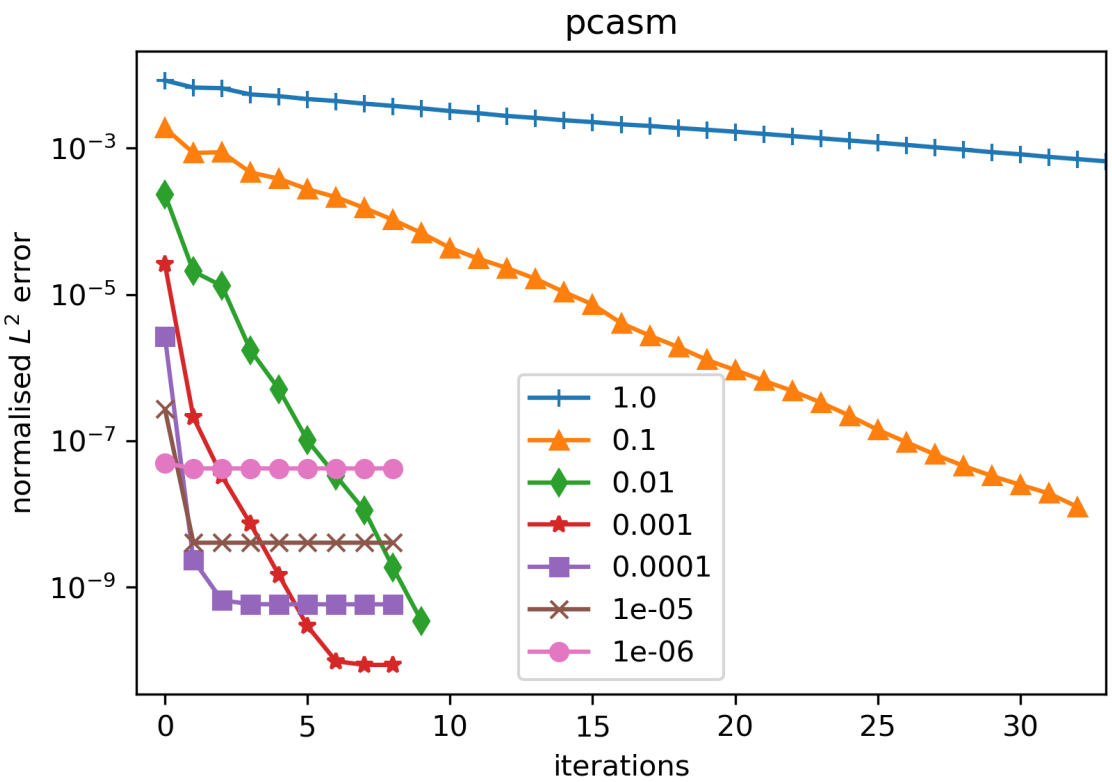
Hybridisation test case



Line smoothers



Line smoothers



With Jack Betteridge and Lawrence Mitchell, "in preparation"

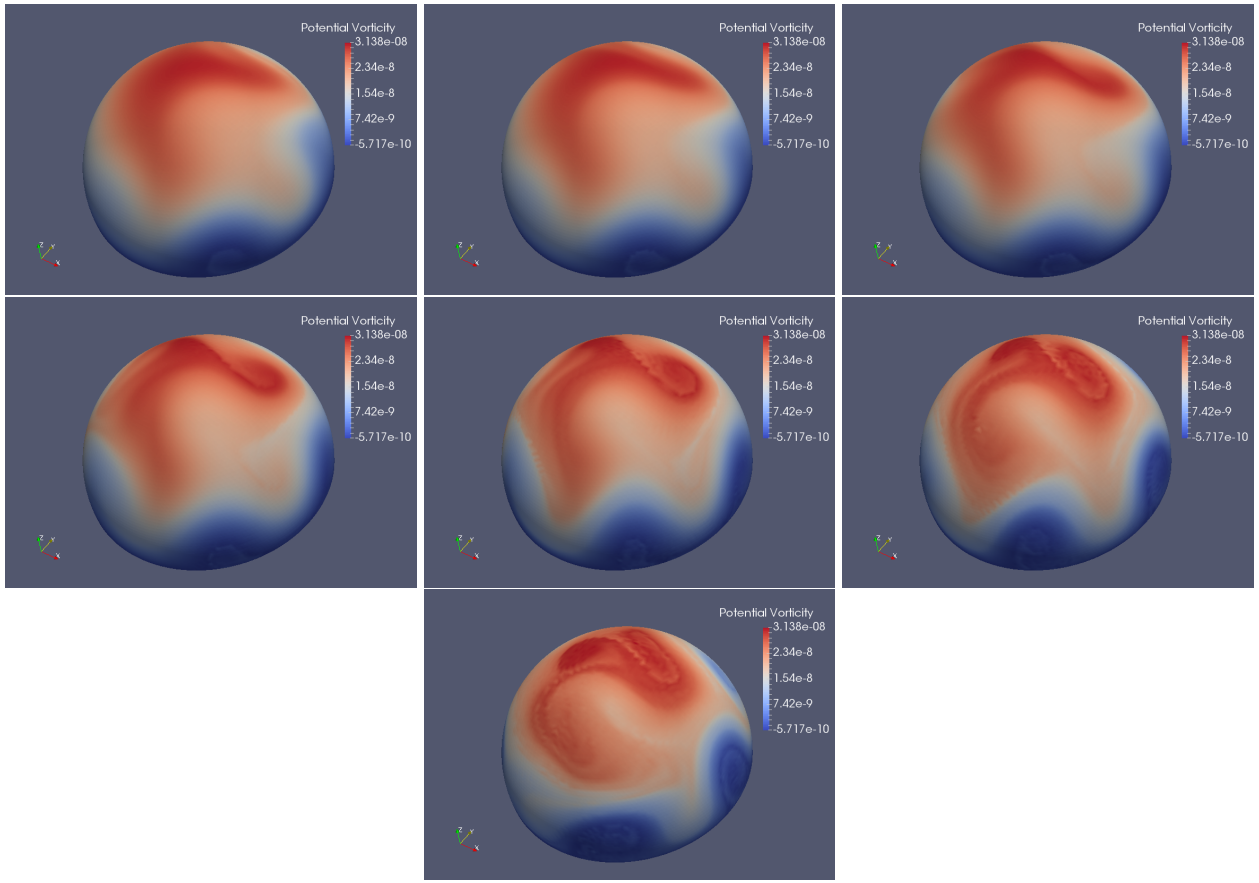
Conservative formulations

Inspired by the C-grid crowd, compatible FEM facilitates Poisson bracket discretisations with:

- Global conservation of energy,
- Global conservation of enstrophy (for SWE),
- Consistent embedded transport scheme for:
  - Potential vorticity (for SWE),
  - 3D vorticity (for 3D, with baroclinic terms).

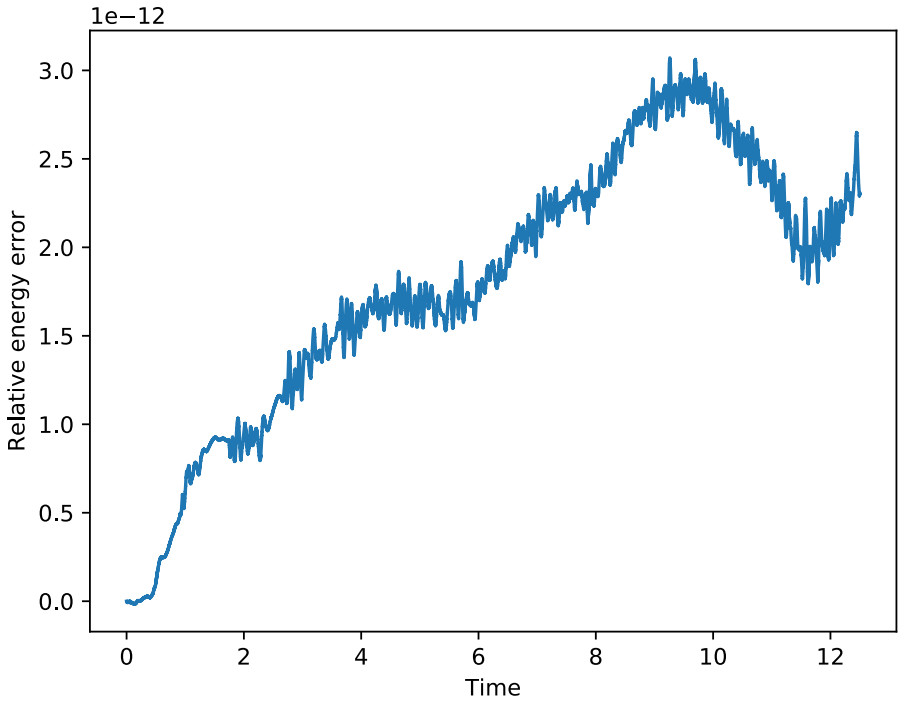
In 2013, I reported on McRae and CJC (2014) - SWE on sphere. Lots of work since then! (see also work by: Dubos, Eldred, Palha, Lee, etc.)

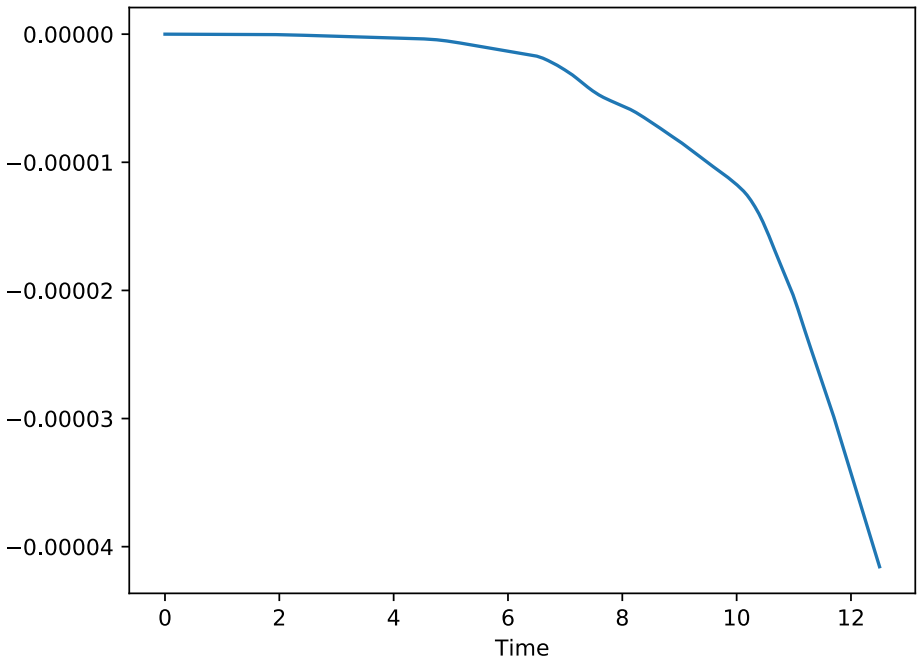
Domains with boundaries



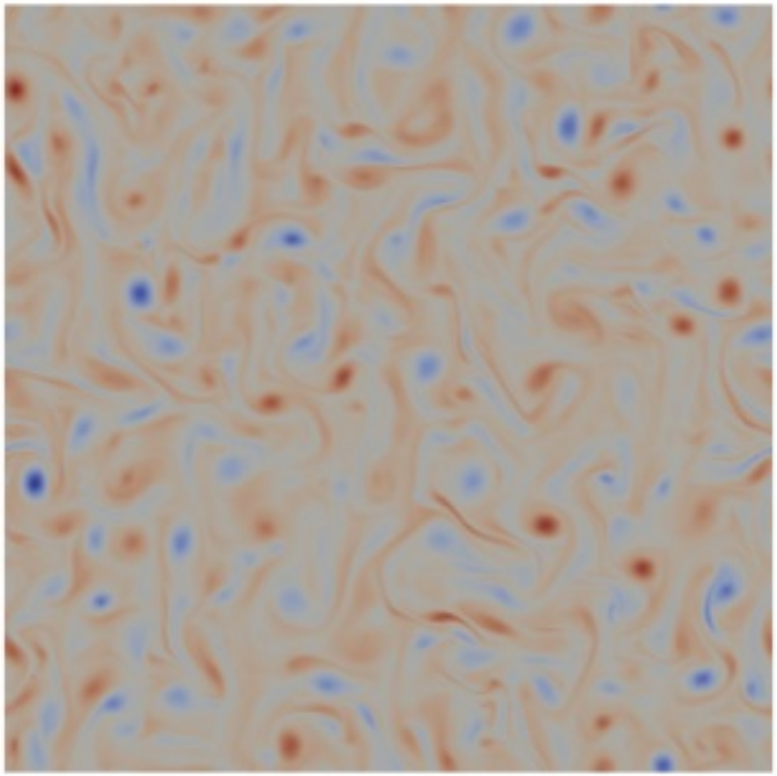
SUPG upwinding, Poisson integrator: Bauer and CJC (2018)

Domains with boundaries



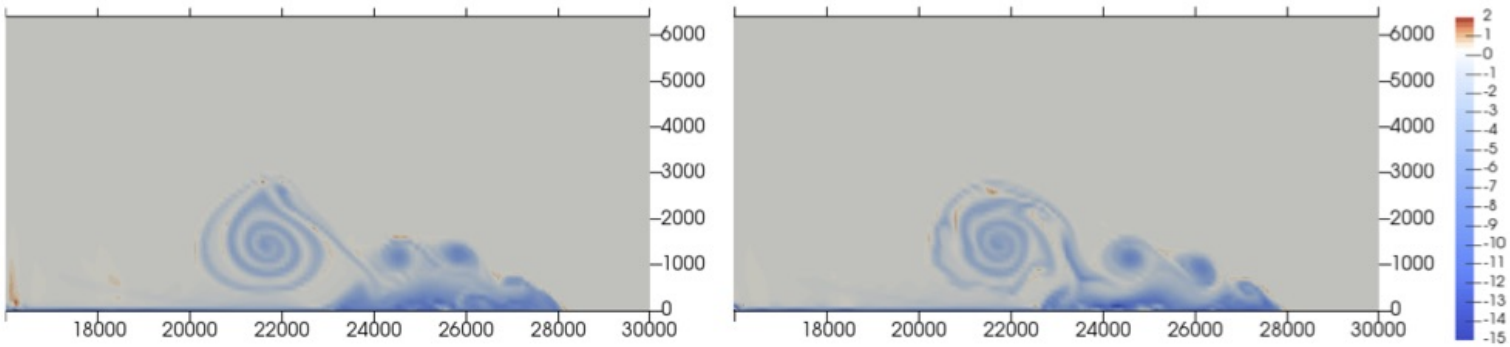


Energy conservation with upwinding: 2D incompressible Euler



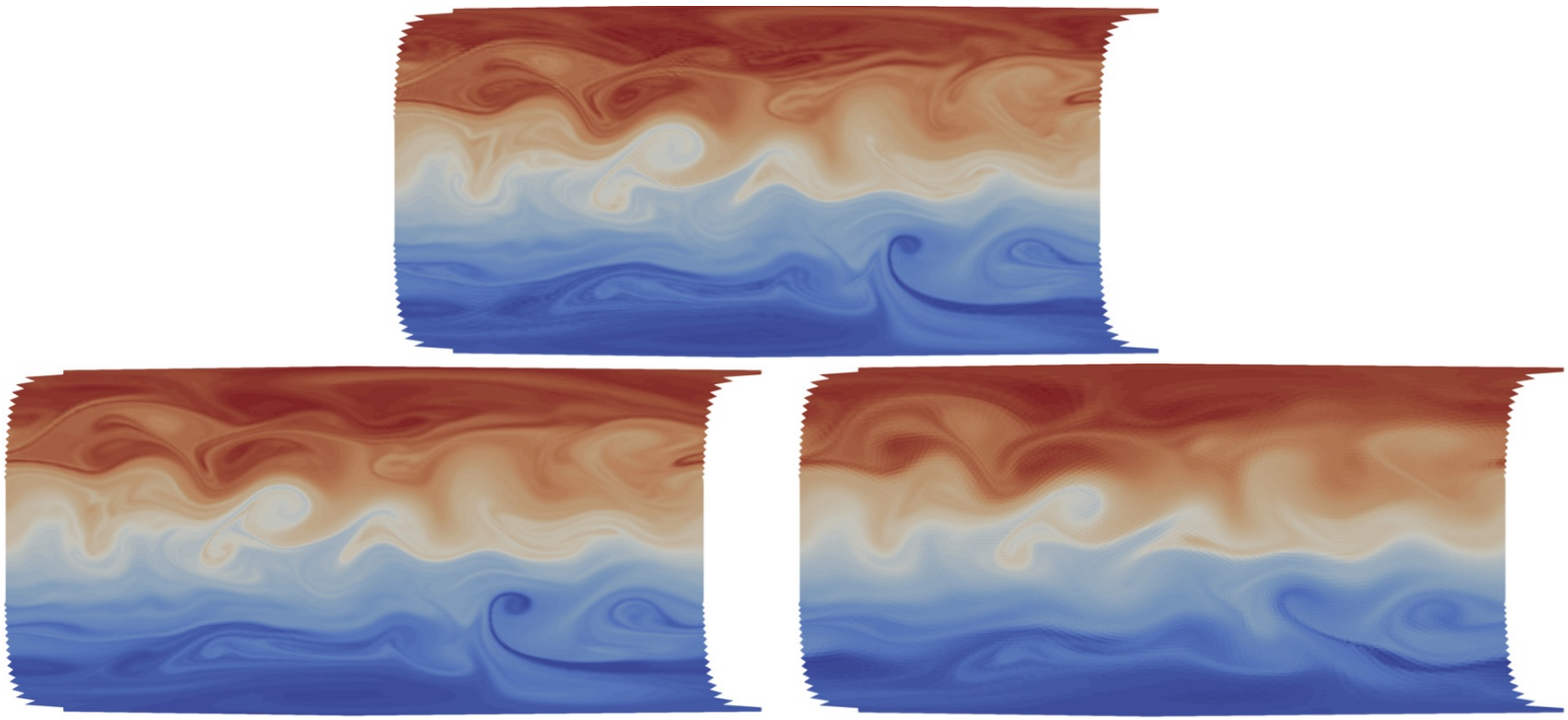
Natale and CJC (2016, QJRMS)

Energy conservation with temperature upwinding: compressible Euler



Wimmer, CJC and Bauer (submitted 2020, ArXiV)

Energy conservation with temperature/vorticity upwinding: TSW



Golo Wimmer PhD Thesis 2020

Energy conservation with temperature/vorticity upwinding: 3D rising bubble

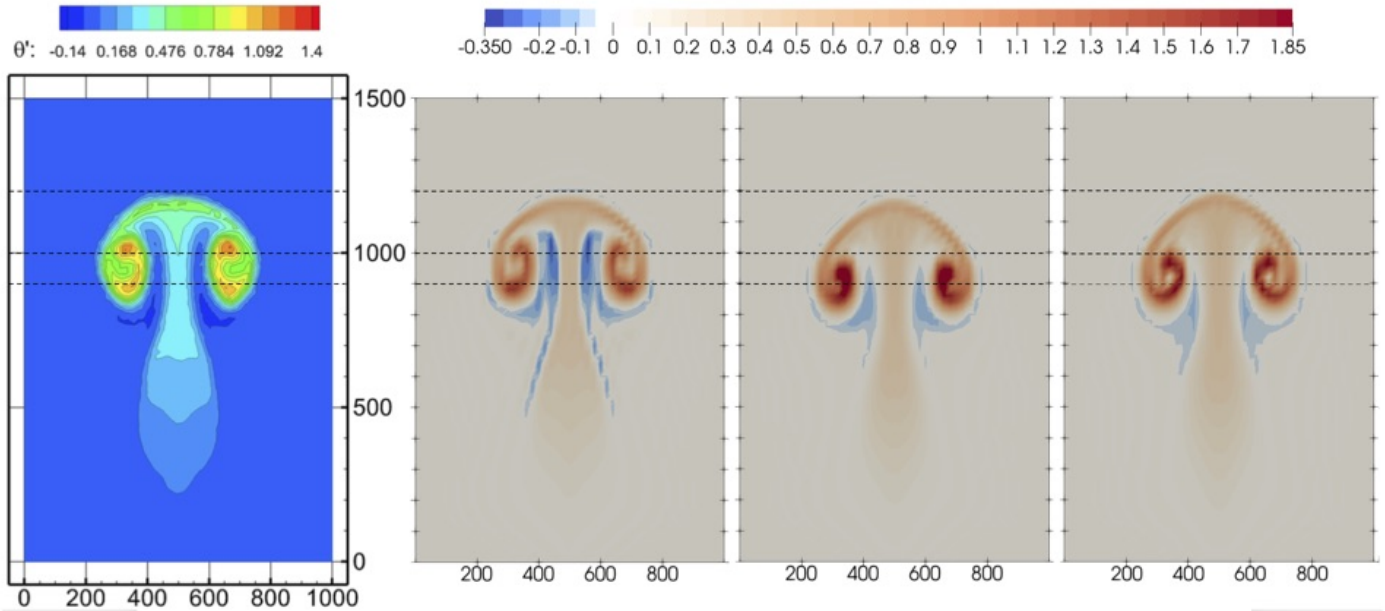


Figure 4.6: Potential temperature fields for 3D rising bubble test case after 300 seconds. Left: reference run from [TD16]. Middle left: energy conserving setup with vorticity SUPG. Middle right: energy conserving setup with velocity upwinding. Right: non-energy conserving setup with velocity upwinding. For the right three images, $\Delta x = \Delta y = \Delta z = 20m$, $\Delta t = 0.8s$, with 4 Picard iterations per time step, and $\lambda_\theta = 1/4$, $\lambda_\omega = 1/2$. Dashed horizontal lines are placed at 900, 1000, and 1200 metres.

Golo Wimmer PhD Thesis 2020

Imperial College London

Energy conservation with temperature/vorticity upwinding: 3D rising bubble

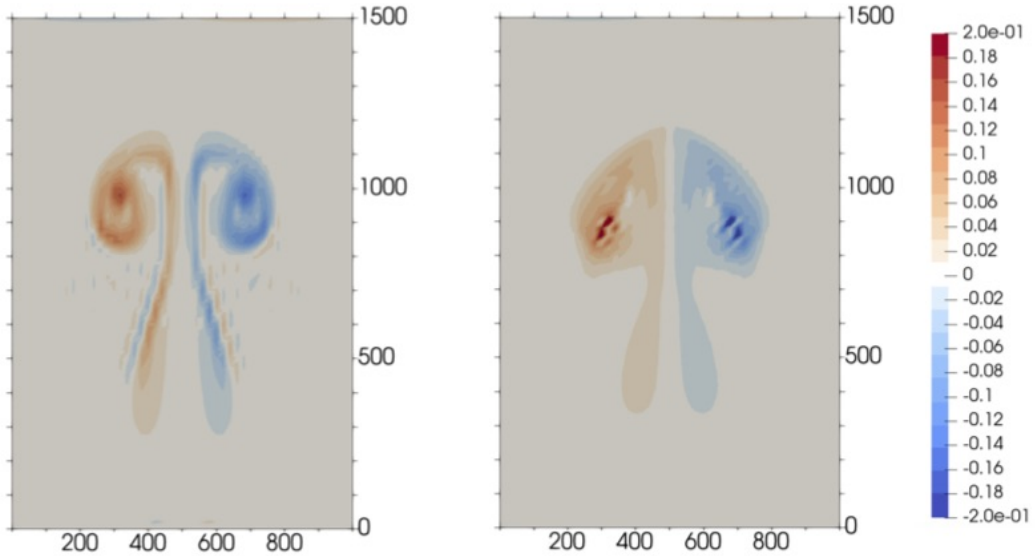


Figure 4.7: Vorticity fields for 3D rising bubble test case after 300 seconds. Left: energy conserving setup with vorticity SUPG. Right: energy conserving setup with velocity upwinding. $\Delta x = \Delta y = \Delta z = 20m$, $\Delta t = 0.8s$, with 4 Picard iterations per time step, and $\lambda_\theta = 1/4$, $\lambda_\omega = 1/2$

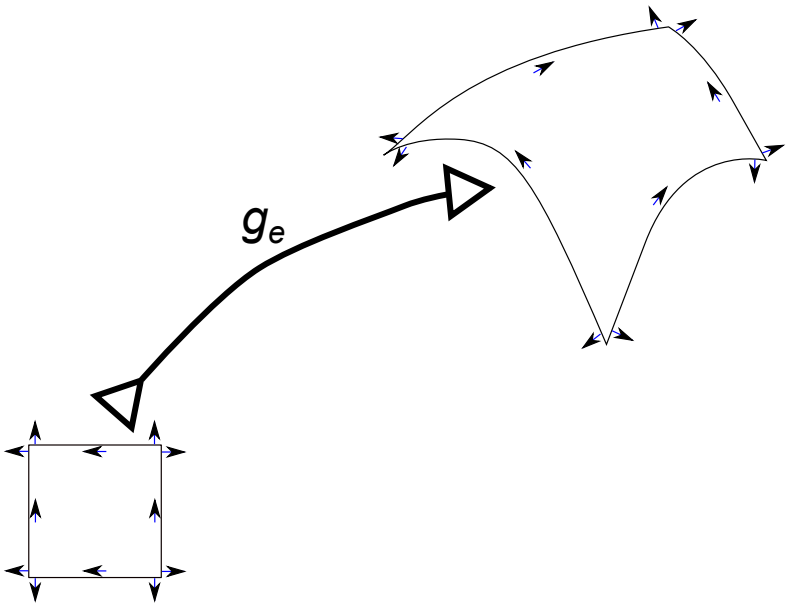
Golo Wimmer PhD Thesis 2020

Some things I'm working on at the moment

- Parallel-in-time methods (phase averaging, ParaDIAG) with Jemma Shipton, Werner Bauer, Hiroe Yamazaki
- Geometric multigrid for iterative solvers (via Gopalakrishnan and Tan (2008)) and for nonhydrostatic ocean with Thomas Gregory
- Scalable methods for solving the full Jacobian
- Truly horizontal and vertical spaces for terrain-following meshes (with Karina Kowalczyk)

The End

Construction of $\mathbb{V}_1$



$\boldsymbol{g}_e : e' \to e,\ \boldsymbol{x} = \boldsymbol{g}_e(\boldsymbol{x}')$.

Definition (Piola transformation)

The Piola transformation $\boldsymbol{u}' \mapsto \boldsymbol{u}$:

$$\boldsymbol{u} \circ \boldsymbol{g}_e = \frac{1}{\det \frac{\partial \boldsymbol{g}_e}{\partial \boldsymbol{x}'}} \frac{\partial \boldsymbol{g}_e}{\partial \boldsymbol{x}'} \boldsymbol{u}'$$

$$\boldsymbol{u}' \cdot \boldsymbol{n}' \, \mathrm{d}x' = \boldsymbol{g}_e^*(\boldsymbol{u} \cdot \boldsymbol{n} \, \mathrm{d}x)$$

Normal components

$\int_f \boldsymbol{u}' \cdot \boldsymbol{n}' \, \mathrm{d}x = \int_{\boldsymbol{g}_e(f)} \boldsymbol{u} \cdot \boldsymbol{n} \, \mathrm{d}x.$

M. Rognes, CJC, D. Ham and A. McRae, Automating the solution of PDEs on the sphere and other manifolds (2013).