Cell-vertex discretization on mixed meshes

S. Danilov and A. Androsov,
Alfred Wegener Institute, Bremerhaven, Germany

Finite-volume cell-vertex discretization on meshes made of triangles and quads:

Why?

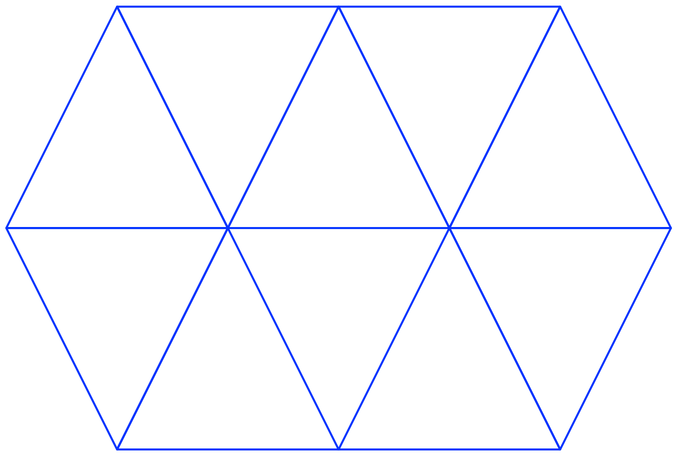
Less difficulty with spurious modes
Higher CPU efficiency

Main numerical issue: control of noise in regimes with grid-scale Reynolds number

Plan:

- cell-vertex setup at AWI
- How to stabilize it in highly transient flows
- Combining triangles and quads

Triangles vs. quads vs. hexagons



Triangles:

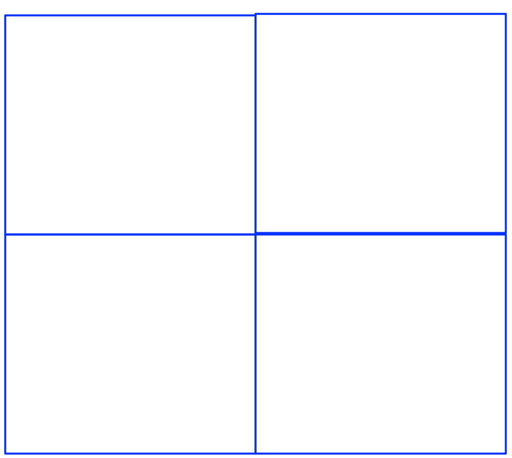
Vertices:cells:edges=1:2:3

Hexagons:

Vertices:cells:edges=2:1:3

Spurious modes:

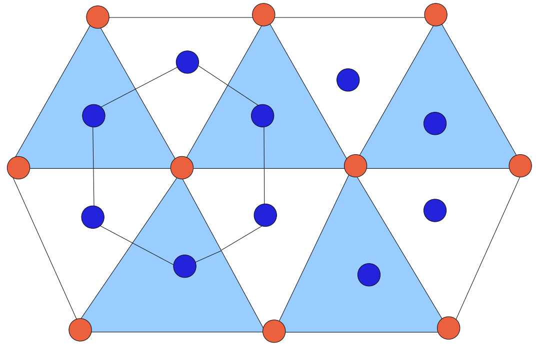
The art is handling these modes without introducing too strong dissipation

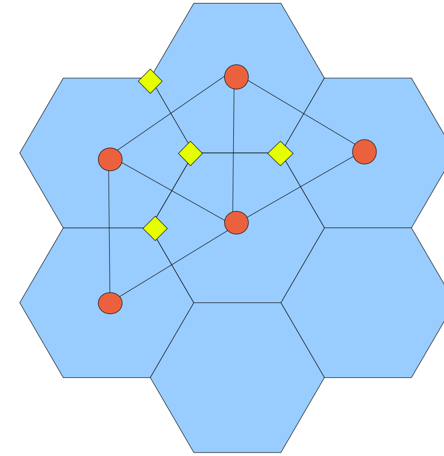


Quads:

Vertices:cells:edges=1:1:1

Finite-volume setup at AWI:

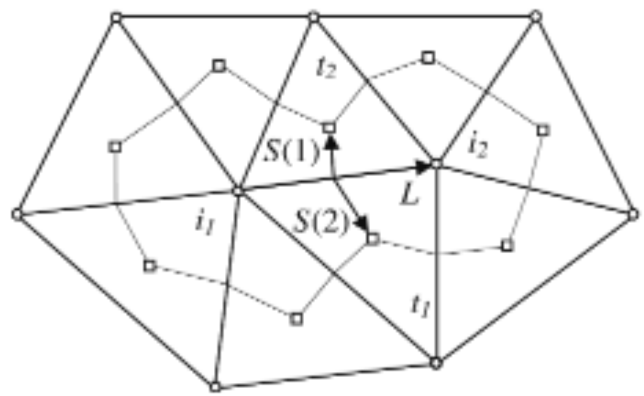


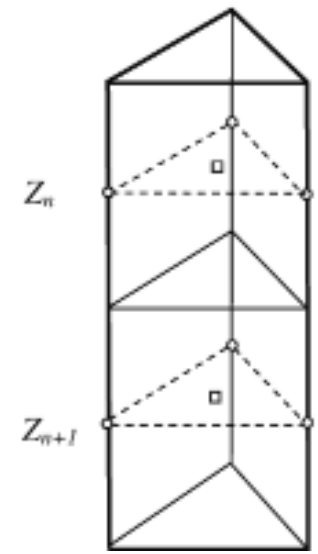


- cell-vertex discretization
- triangular surface mesh
- scalar part is similar to the hexagonal C-grid MPAS

Motivation: computational efficiency (no mass matrices) -- 2-3 times faster than FESOM

Status: running, coupled to sea-ice model, tests vs. FESOM





Velocities on cells (squares), scalars on vertices (circles)
Medial-dual control volumes around vertices are hexagonal on equilateral meshes

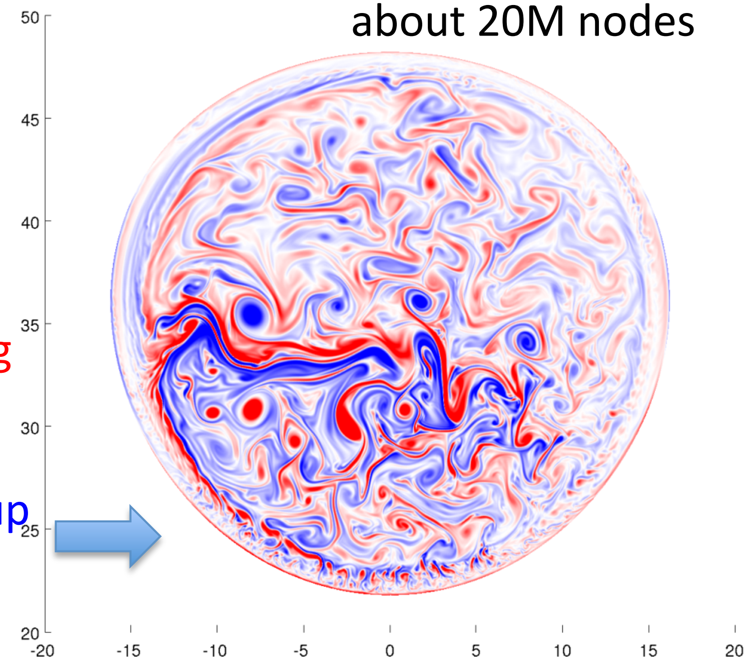
SOMA (Simulation of Ocean Mesoscale Activity)
(wind-driven flow in a temperature-stratified ocean, 40 layers). Resolution from 32 to 4 km)

Participants: MPAS (hexagons), FV AWI (triangles)
POP (B-grid), ROMS (C-grid)

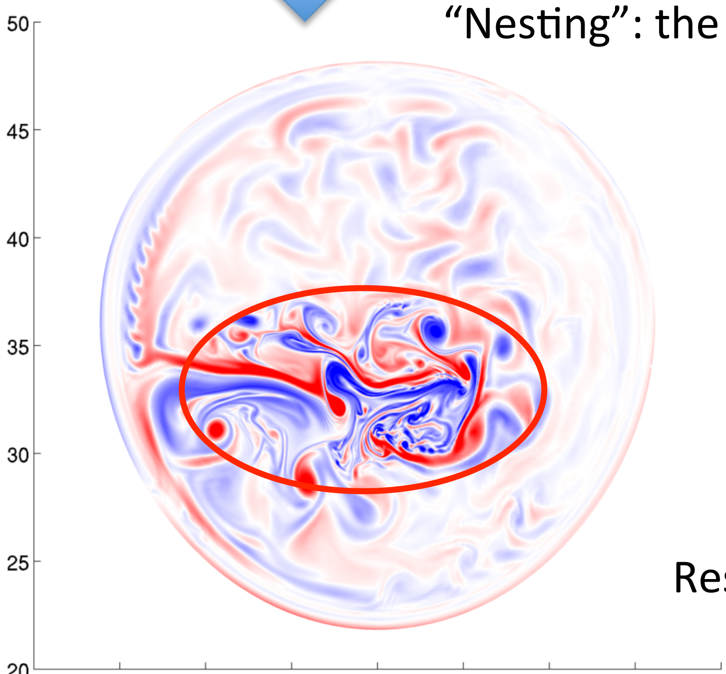
Question: How numerics affects the dynamics of eddying regimes

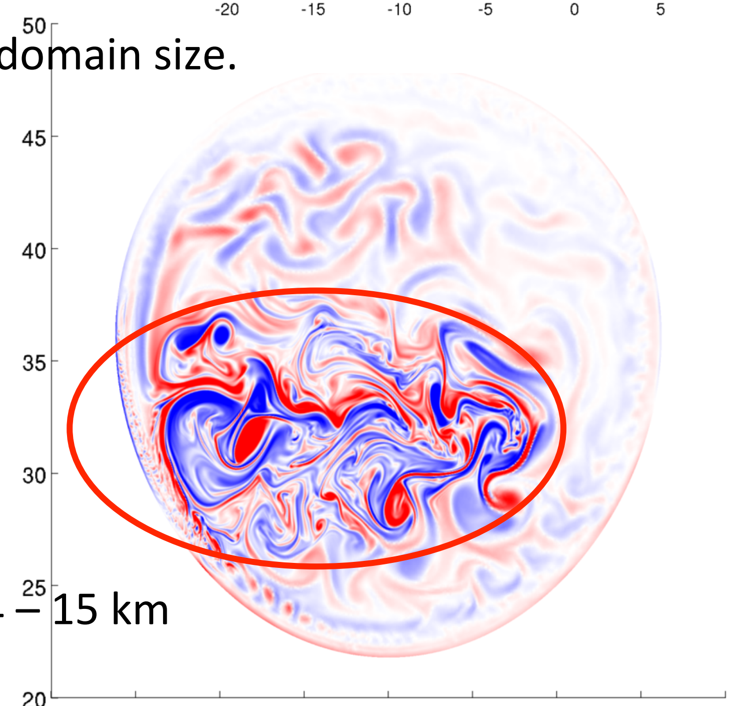
Relative vorticity at 100 m, simulations with FV AWI setup

Resolution 4 km,
about 20M nodes



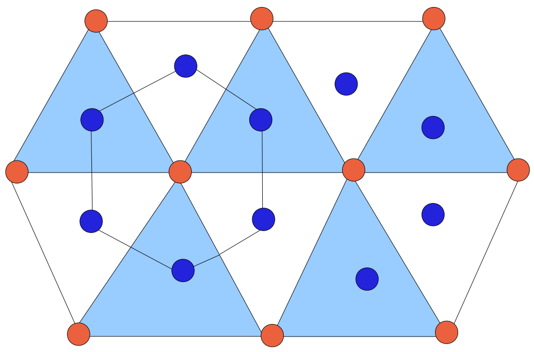
“Nesting”: the effect of domain size.





Resolution 4 – 15 km

Efficient work of triangular cell-vertex discretization requires stabilization:
(i) momentum advection
(ii) viscosity operator



Momentum advection:

(i) Standard flux implementation (linear upwind reconstruction) on cell control volumes

$$\int \nabla(\boldsymbol{v}\boldsymbol{u})\,dS = \sum (\boldsymbol{n}\boldsymbol{v})_i \boldsymbol{u}_i l_i, \quad \boldsymbol{v} = (\boldsymbol{u}, w)$$

--- too dissipative (on resolved scales) and creates too much grid-scale noise

(ii) Project the horizontal velocity on P1 and compute fluxes then, but with centered approach --- performs much better

(iii) Compute momentum flux divergence at scalar control volumes, then average to velocity locations --- nice filtering of velocity noise

(iv) Use vector invariant form --- cheaper than (iii), with nearly equivalent performance.

$$(\boldsymbol{v}\nabla)\boldsymbol{u} = w\partial_z \boldsymbol{u} + curl\,\boldsymbol{u} \times \boldsymbol{u} + \nabla(\boldsymbol{u}\boldsymbol{u}/2)$$

Why: the relative vorticity and kinetic energy are computed at scalar points, which provides averaging.

Viscous operators for velocity

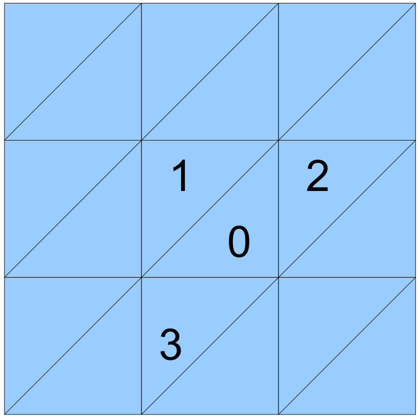
Since the velocity space is too large, effective dissipation of small scales is needed.

1. Standard viscosity implementation

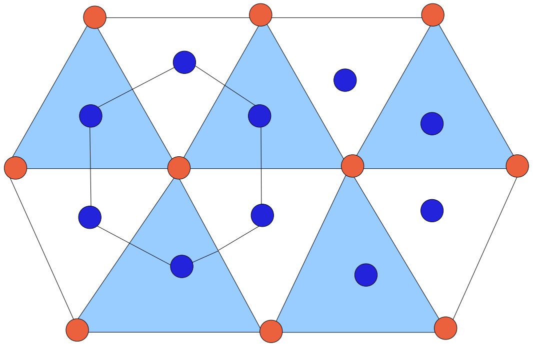
$$Lv_i = \partial_j A \partial_j v_i$$

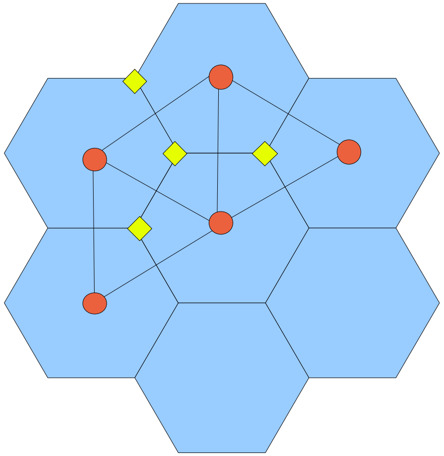
Bad

Laplacian at 0 does not involve velocity at neighboring points 1, 2 and 3. Noise on up and down triangles is decoupled.



2. Small-stencil Laplacian (Ringler and Randal, 2002)





In reality, on uniform meshes $(L_s\mathbf{u})_0 = (\mathbf{u}_1 + \mathbf{u}_2 + \mathbf{u}_3 - 3\mathbf{u}_0)/3$, i. e., it is a filter operator.
-- implement as filter (harmonic and biharmonic).
But: it deviates from Laplacian on general meshes.

3. Repair standard computations:

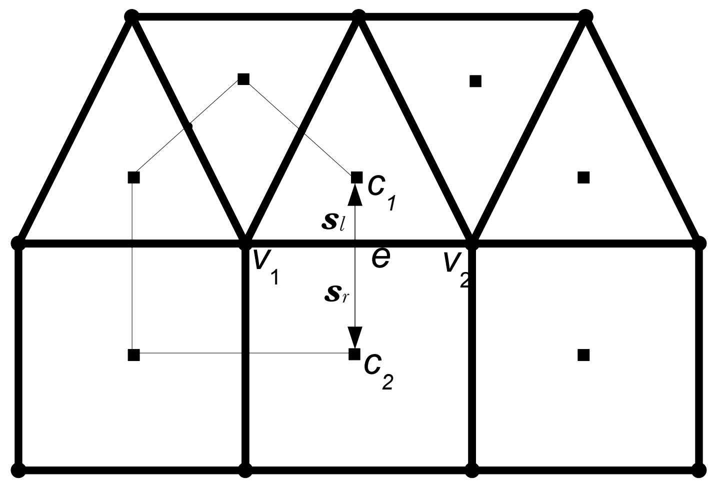
$$\mathbf{n}_{10} = \mathbf{r}_{10}/|\mathbf{r}_{10}| + (\mathbf{n}_{10} - \mathbf{r}_{10}/|\mathbf{r}_{10}|),\ldots$$
$$\mathbf{n}_{\mathbf{j}}\partial_{\mathbf{j}}\mathbf{v}_{\mathbf{i}} = \partial\mathbf{v}_{\mathbf{i}}/\partial\mathbf{r} + (\mathbf{n}_{\mathbf{j}} - \mathbf{r}_{\mathbf{j}}/|\mathbf{r}|)\partial_{\mathbf{j}}\mathbf{v}_{\mathbf{i}}$$

It is equivalent to L_s on equilateral triangles and approximates the Laplacian operator on general meshes.

4. Use the Leith and modified Leith viscosity

$$A = CS^{3/2}|\nabla\nabla\cdot\mathbf{u}|$$

Combine triangles and quads:
Why?
(i) Quads are less vulnerable to spurious modes
(ii) They are more efficient numerically (less edges)
(iii) Triangles can be used to make smooth transitions

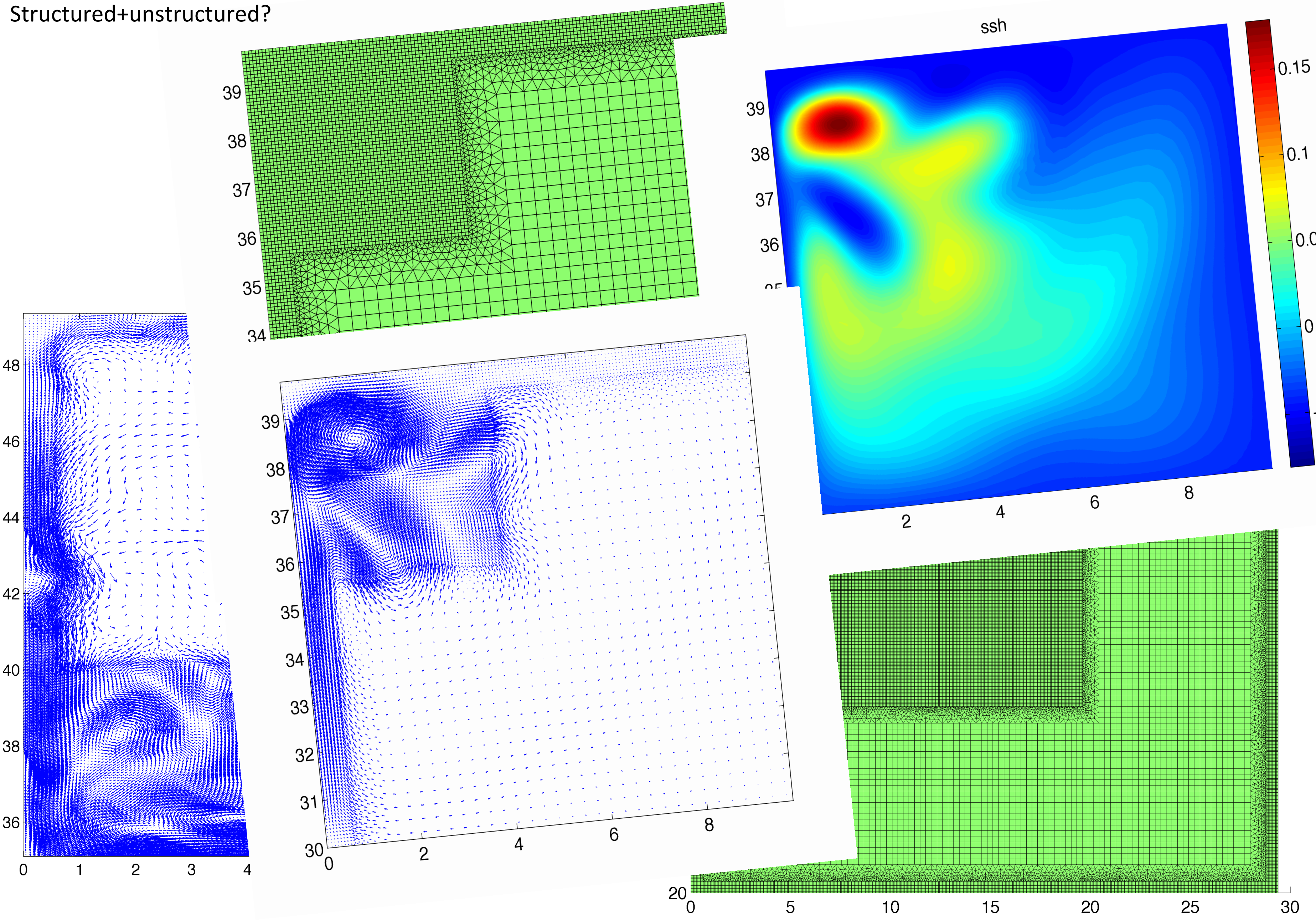


Caveat:
Jump in resolution --- not in reality! But still needs stabilization.

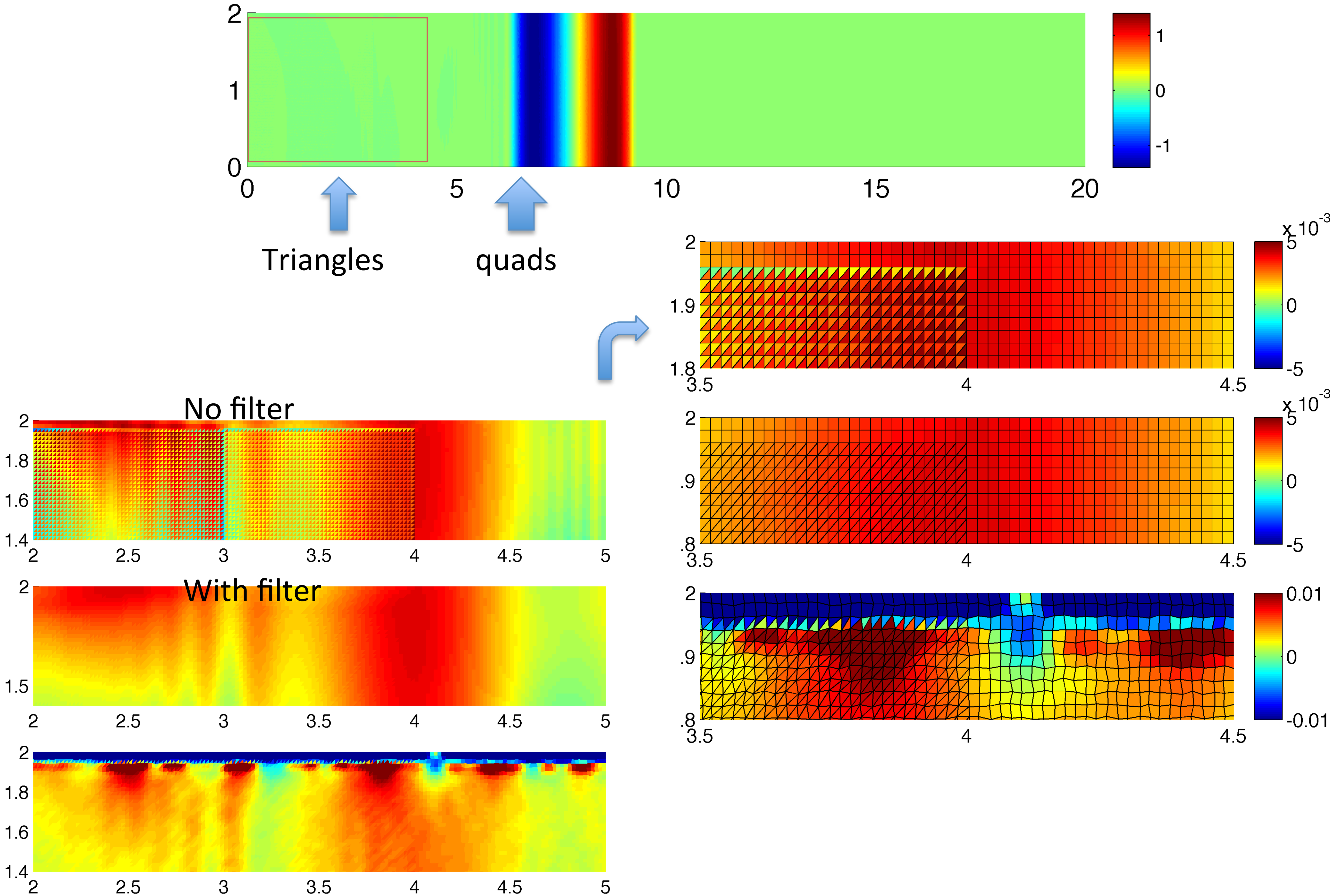
Is it better than 2-way nesting?
Yes, because it is consistent, and transition is smoother.

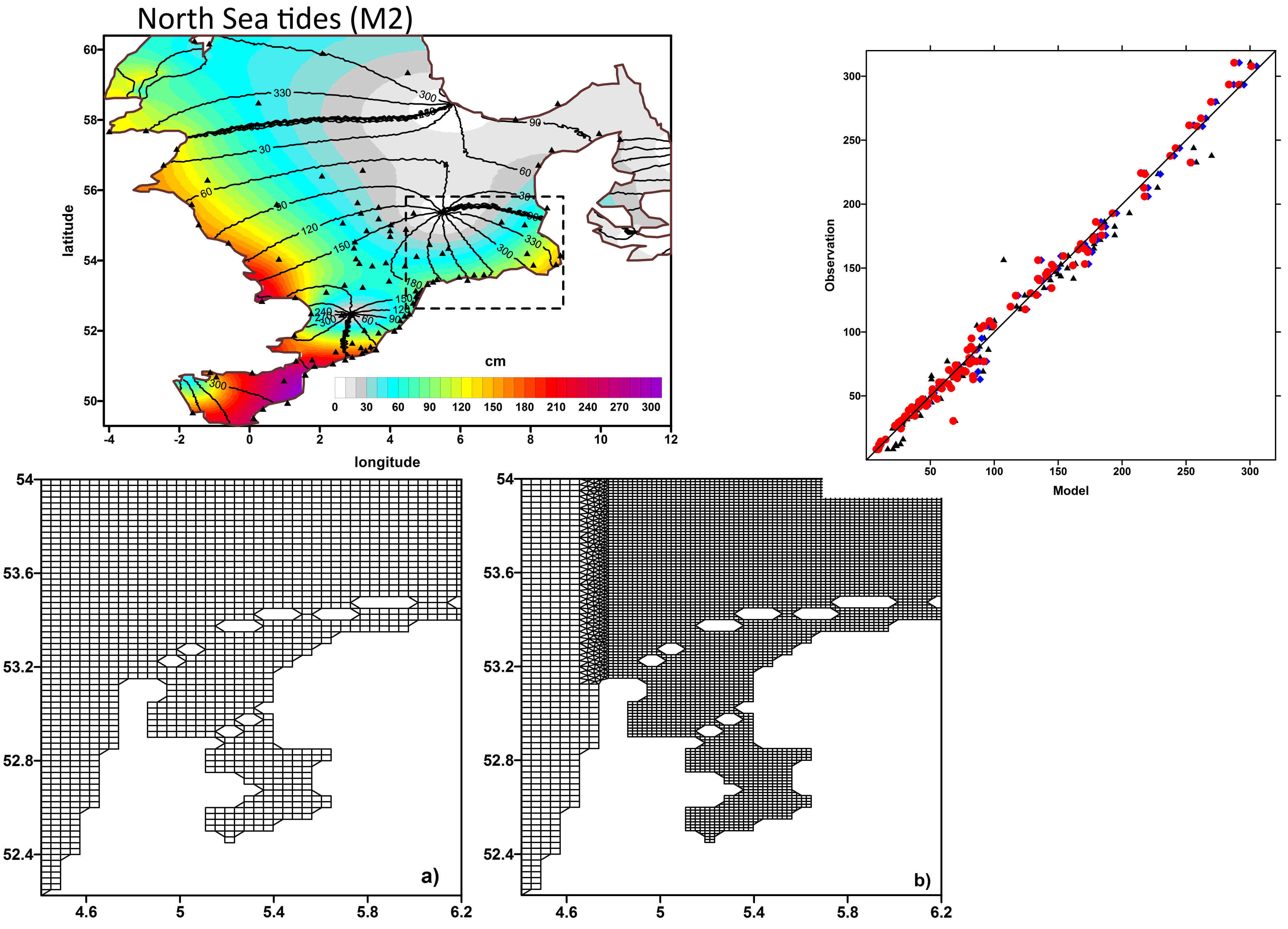
Mixed meshes (Playing games)

- Structured+unstructured?

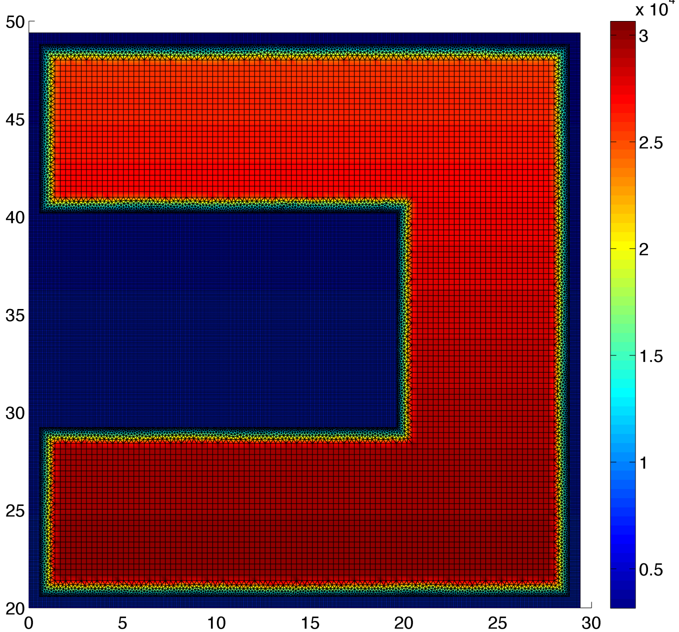


Wave propagation test: wave of 10 m in amplitude in a 500 m deep channel




North Sea tides (M2)
latitude
longitude
cm
0 30 60 90 120 150 180 210 240 270 300
Observation
Model
a)
b)

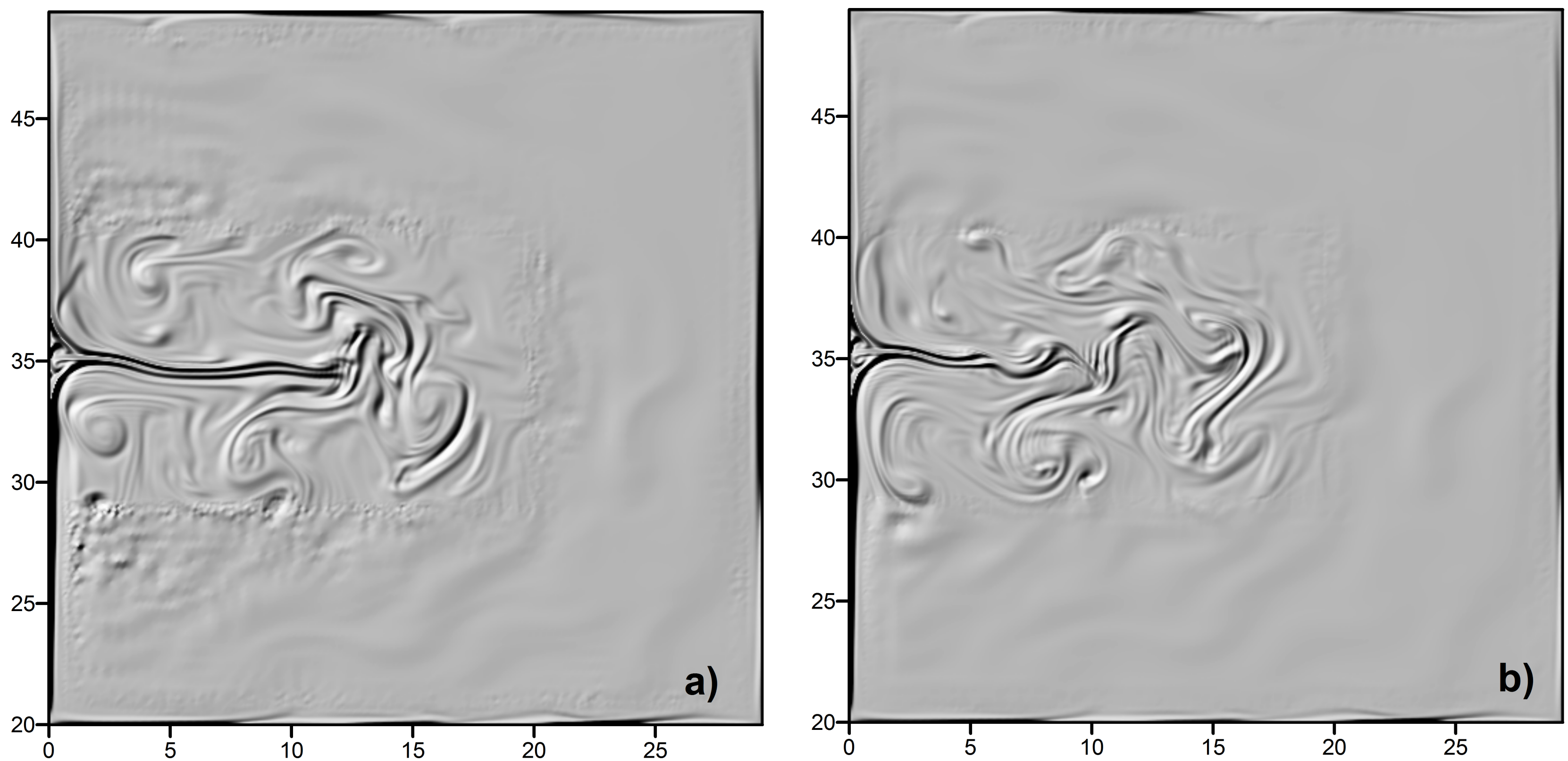
Wind-driven double-gyre flow



Mesh resolution

Relative vorticity pattern visualizes potential difficulties

Biharmonic filter is 3 times stronger only over triangles --- noise is almost gone!



Conclusions:
Cell-vertex discretization works well on triangular meshes, but needs appropriate implementation of momentum advection and tuned viscosity.

Triangles and quads can be easily combined. Stabilization needed on triangular meshes is sufficient to control noise at transitions.