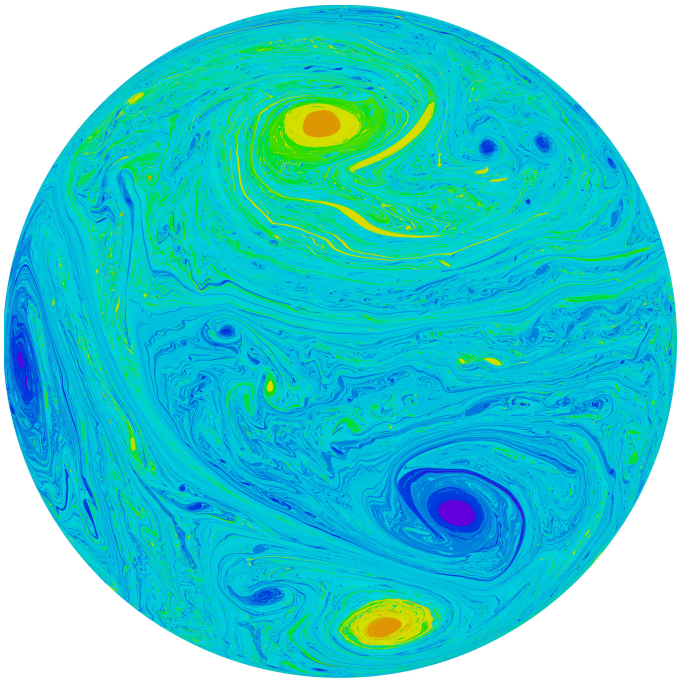
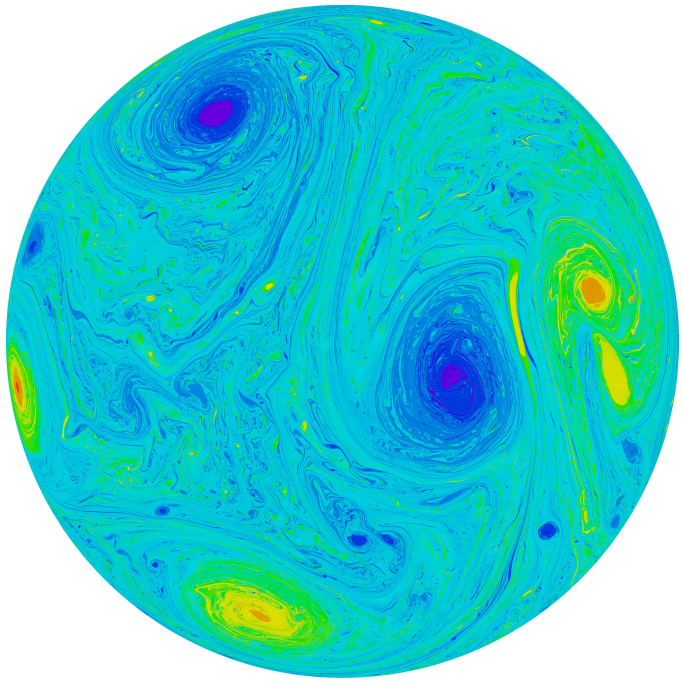
Lagrangian-based approaches in GFD, and associated advanced numerical methods

David Dritschel

November 2024





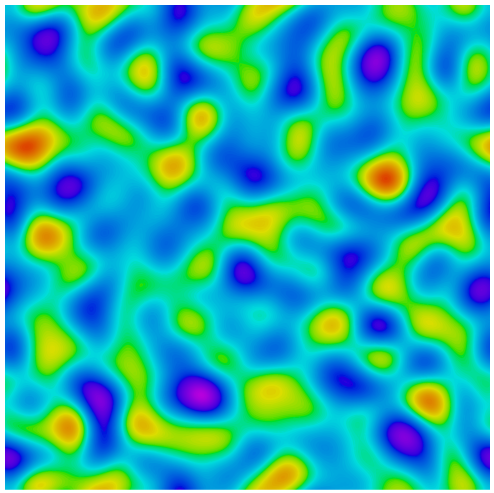
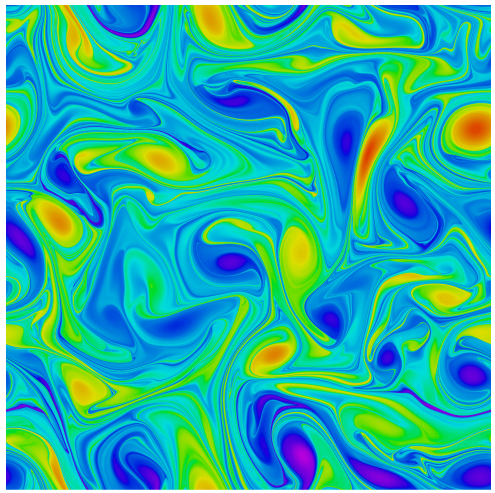
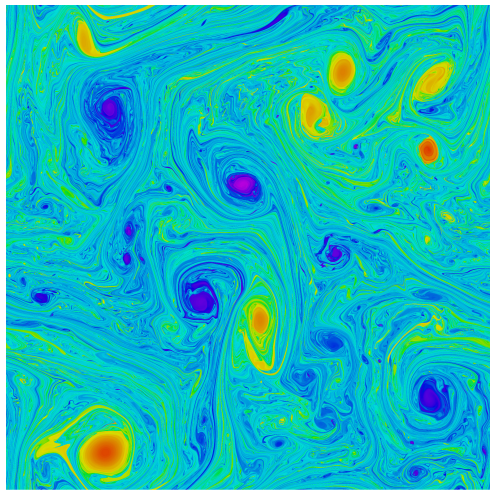
Outline

Numerical models: overview

Even using greatly simplified reduced models may still require powerful computers to "solve" (approximately).

For example, in two-dimensional flow — a flow independent of height *or* a quasi-geostrophic flow in the limit $L_D \to \infty$ — the equations are still difficult to solve since they are fundamentally *nonlinear*.

This nonlinearity allows the excitation of many scales of motion, leading to turbulence, which even in two dimensions is complicated!

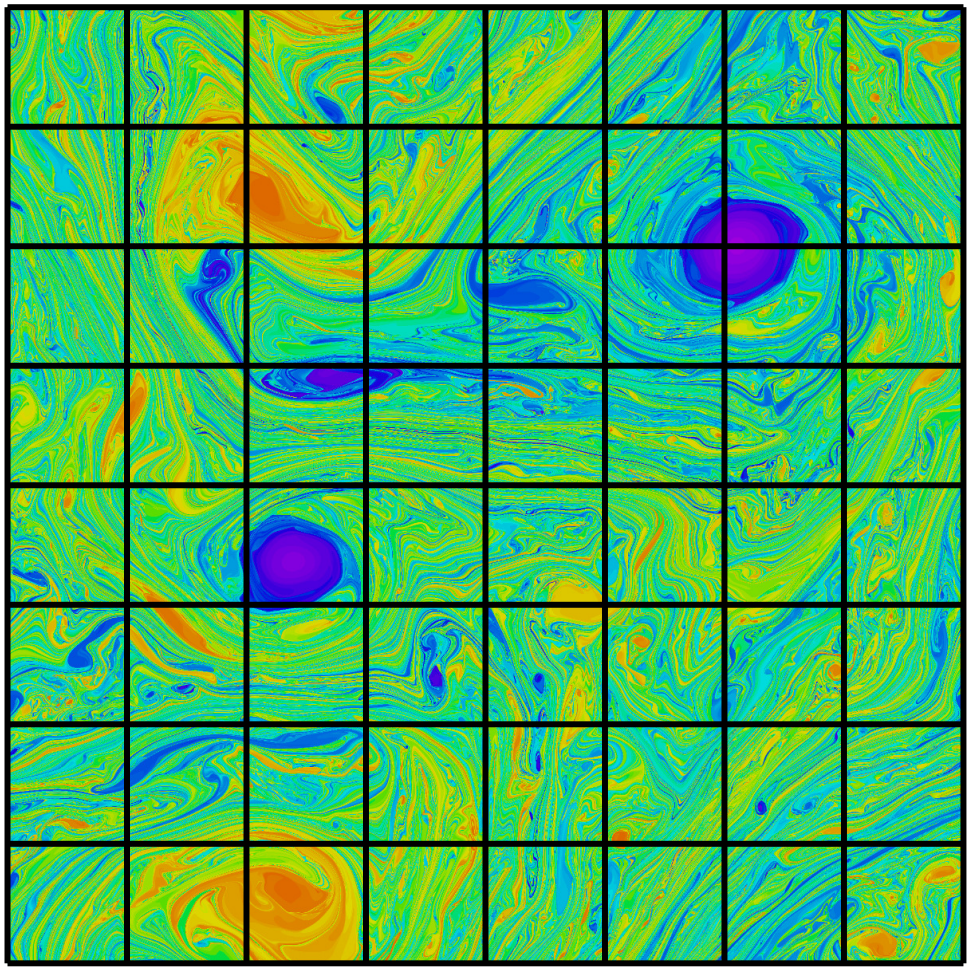
Discretisation

Ultimately, a computational model must discretise space and time in some way since resources are finite.

Space is often discretised into grid cells, labelled by position $\boldsymbol{x}$. In this approach, no detail is retained below the grid scale.

This can lead to serious errors.
Moreover, it is avoidable.

- *Other approaches exist.*



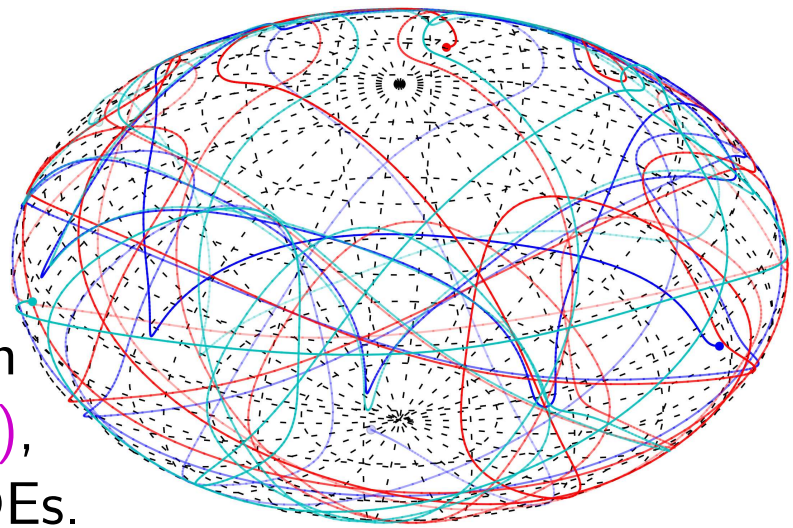
Alternative approaches: points

There are *many* approaches to this problem. The simplest *exactly* reduce, where possible, the PDEs to ODEs by using *points*, e.g. vortices, charges, masses, *etc*.

This is possible when the system possesses a material invariant, e.g. vorticity in two-dimensional (2D) flow (or potential vorticity in a quasi-geostrophic (QG) flow).

This can be concentrated into a finite number of points, each with finite circulation in the fluid dynamical context.

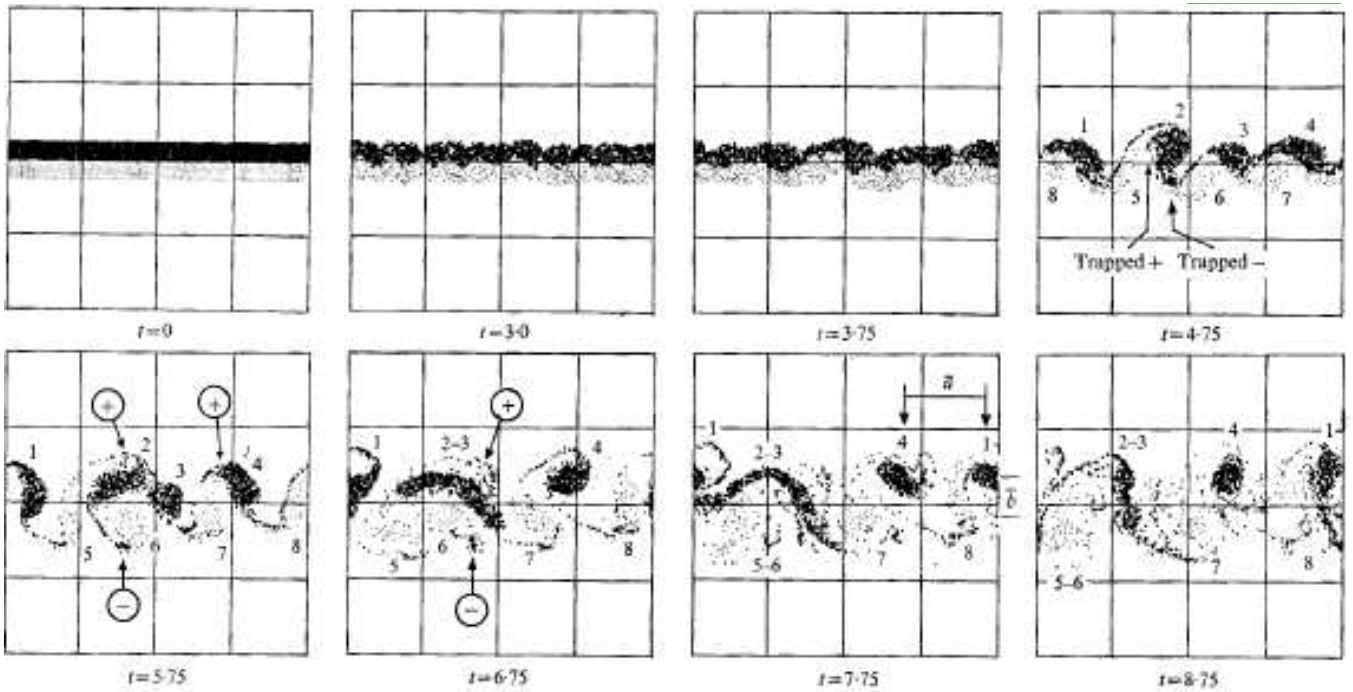
The system evolution then just depends on the point positions (and domain geometry), leading to a coupled set of (nonlinear) ODEs.



Alternative approaches: particle-in-cell

For a small number of points and simple surfaces/domains, the dynamics are analytically tractable. But normally one must solve the coupled nonlinear ODE system. For large numbers of points n, the numerical cost grows like n^2 and near collisions make the problem stiff.

The alternative is "Particle-In-Cell", whereby an underlying grid is used to *approximately* compute the point interactions. This is a *hybrid* method.



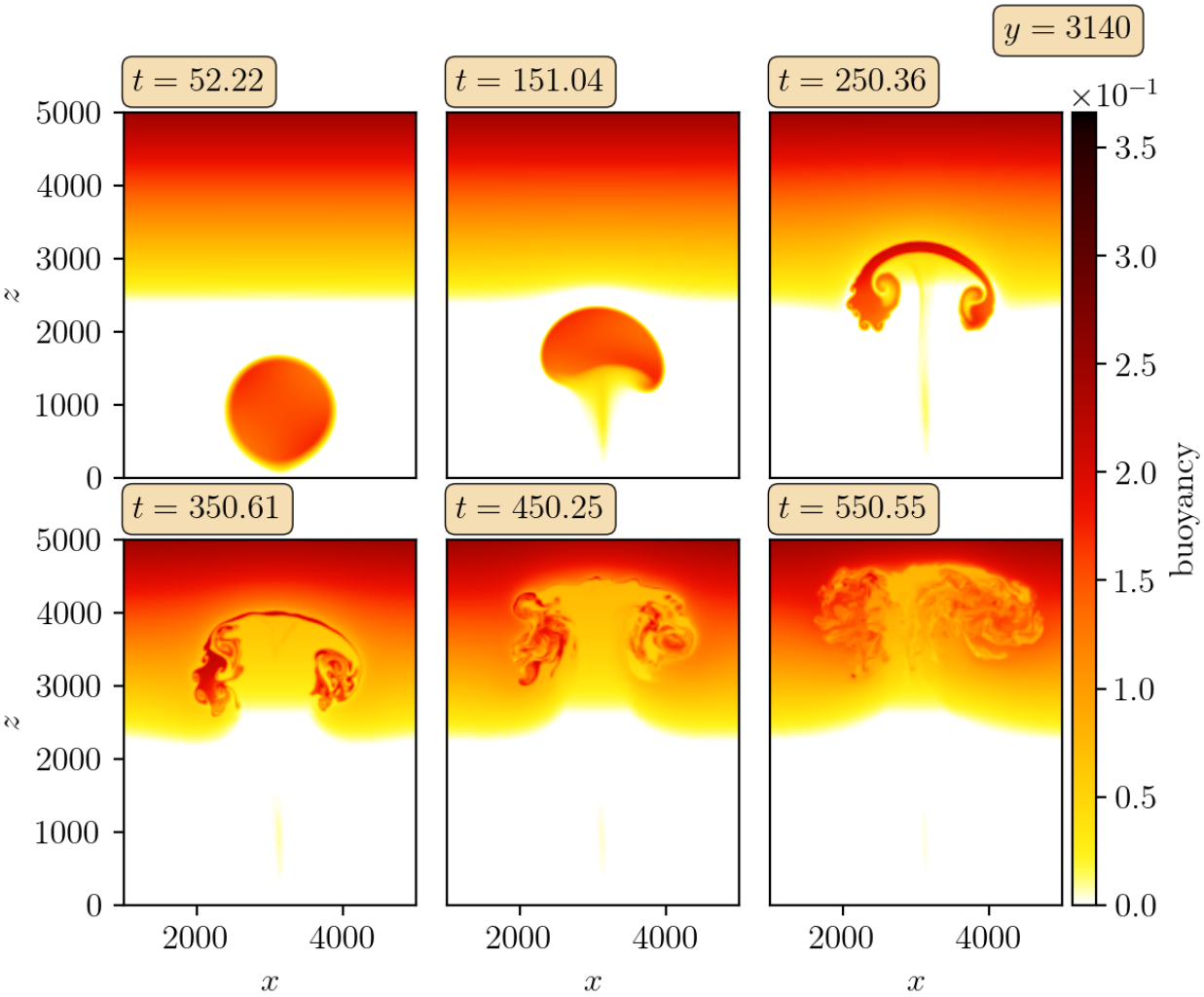
J. P. Christiansen and N. J. Zabusky

FIGURE 2. Experiment I. Instability of a laminar wake subject to a small random perturbation. Fixed y boundaries.

J. of Comput. Physics (1973)

Alternative approaches: elliptical parcel-in-cell (EPIC)

Recently, a major extension of PIC called "Elliptical Parcel-In-Cell" (EPIC) has shown great promise in modelling clouds in unprecedented detail — see Frey, Dritschel & Böing *J. Comput. Phys. X* **17** (2023) and below.

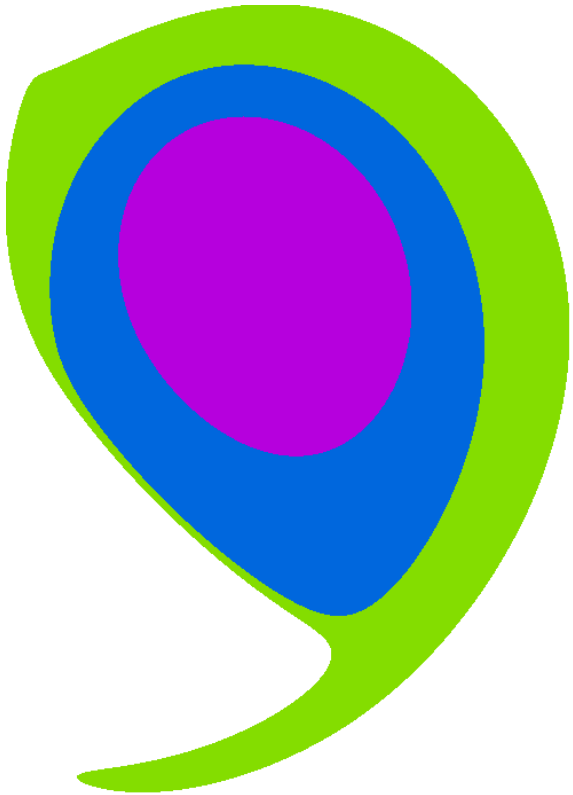


Here $> 10^{10}$ parcels used!

x-z cross section of a 3D simulation

Alternative approaches: contour dynamics

Another alternative approach, also exploiting material conservation, is "Contour Dynamics". The basic idea is to represent the conserved field, say q, by a set of contours between which q is *uniform*.



In this way, one *only* needs to follow the contours, *not all the points between*.

For quasi-geostrophic flow on $\mathbb{R}^2$, for example

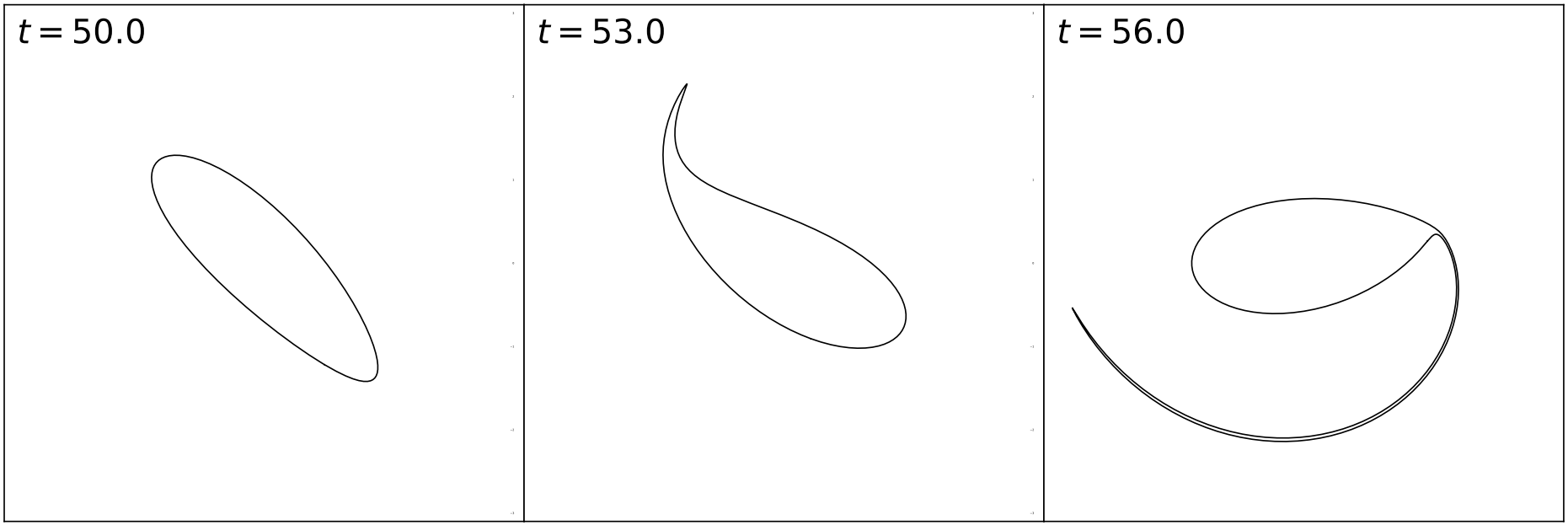
$$\frac{\mathrm{d}\boldsymbol{x}_j}{\mathrm{d}t} = \frac{1}{2\pi} \sum_k \Delta q_k \oint_{C_k} K_0(\|\boldsymbol{x}'_k - \boldsymbol{x}_j\|/L_D)\, \mathrm{d}\boldsymbol{x}'_k$$

where $\boldsymbol{x}'_k$ is a point on the kth contour C_k while $\boldsymbol{x}_j$ is a point on another contour C_j. Here Δq_k is the inward jump in q across C_k and L_D is the Rossby deformation length.

Alternative approaches: contour dynamics

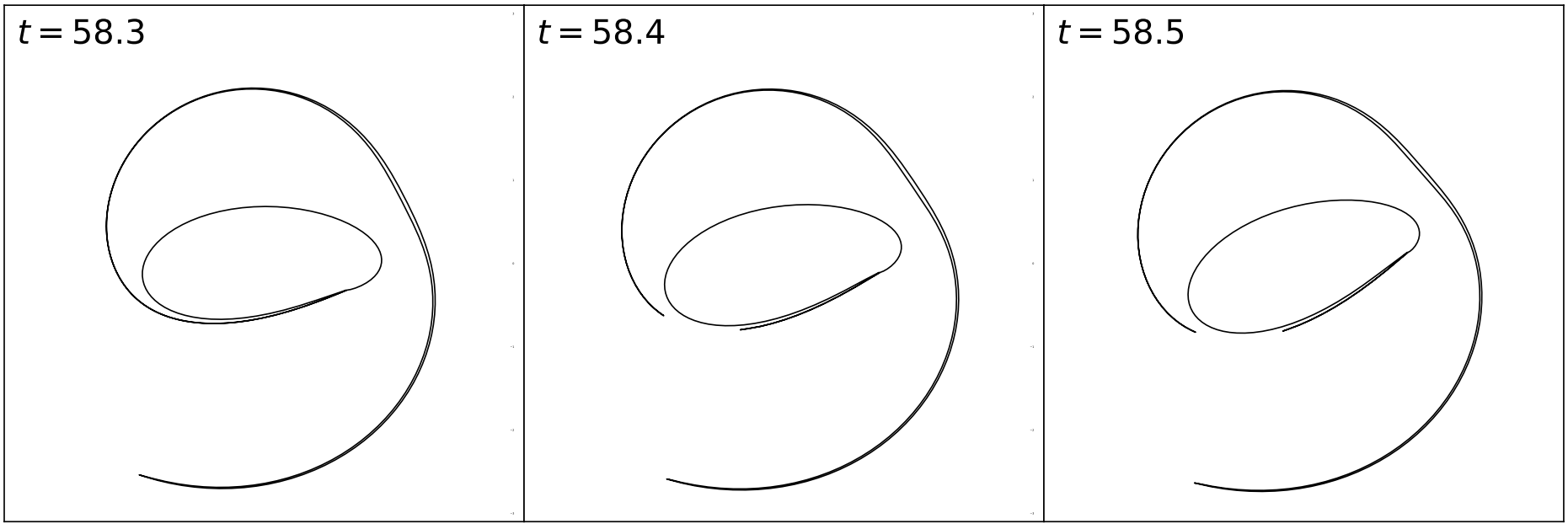
"Contour Dynamics" was developed for the 2D Euler equations (the limit $L_D \to \infty$ of the QG equations) by Zabusky, Hughes & Roberts (1979), though originates in the "Water Bag" model of Berk & Roberts (1965) used to study the Vlasov equations in Plasma Physics.

It is limited by the inevitable tendency for contours to filament, leading to a rapid growth in contour length and complexity:



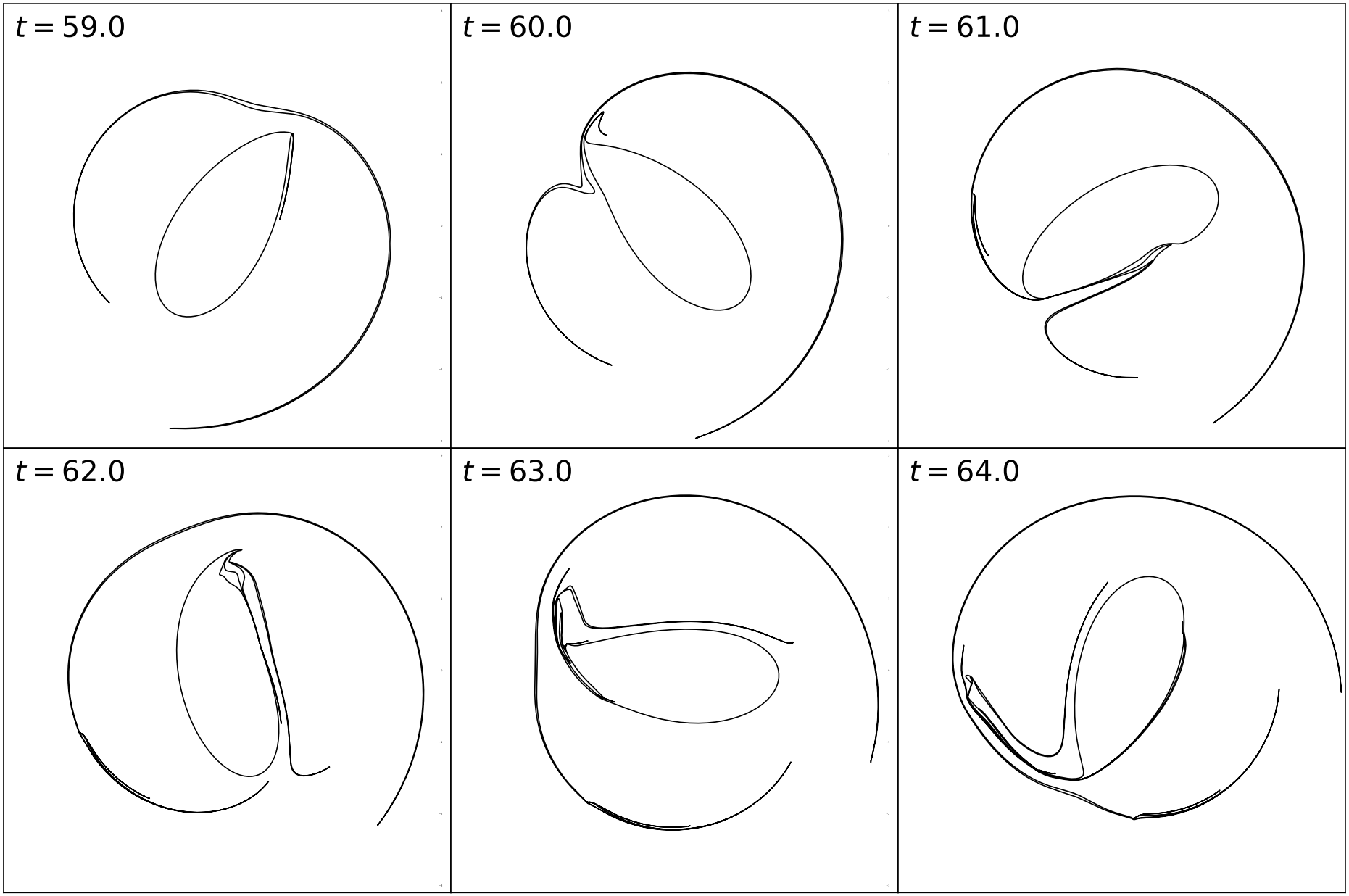
Alternative approaches: contour surgery

"Contour Surgery" (Dritschel *J. Comput. Phys.* **77**, 1988) was developed to cope with this situation, limiting contour complexity by permitting topological reconnections:

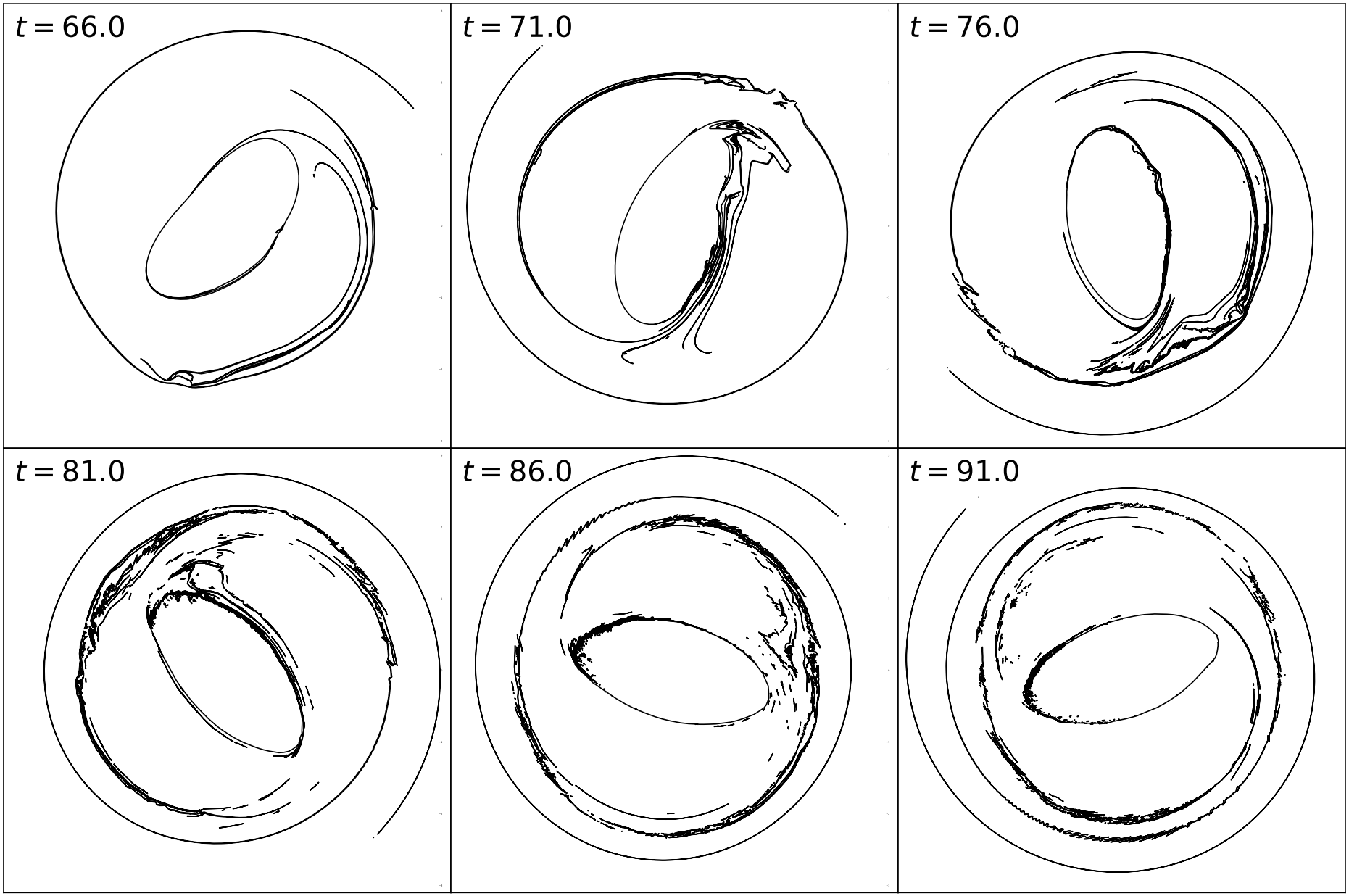


This enabled one to carry out long time simulations and determine the fate of vortex interactions.

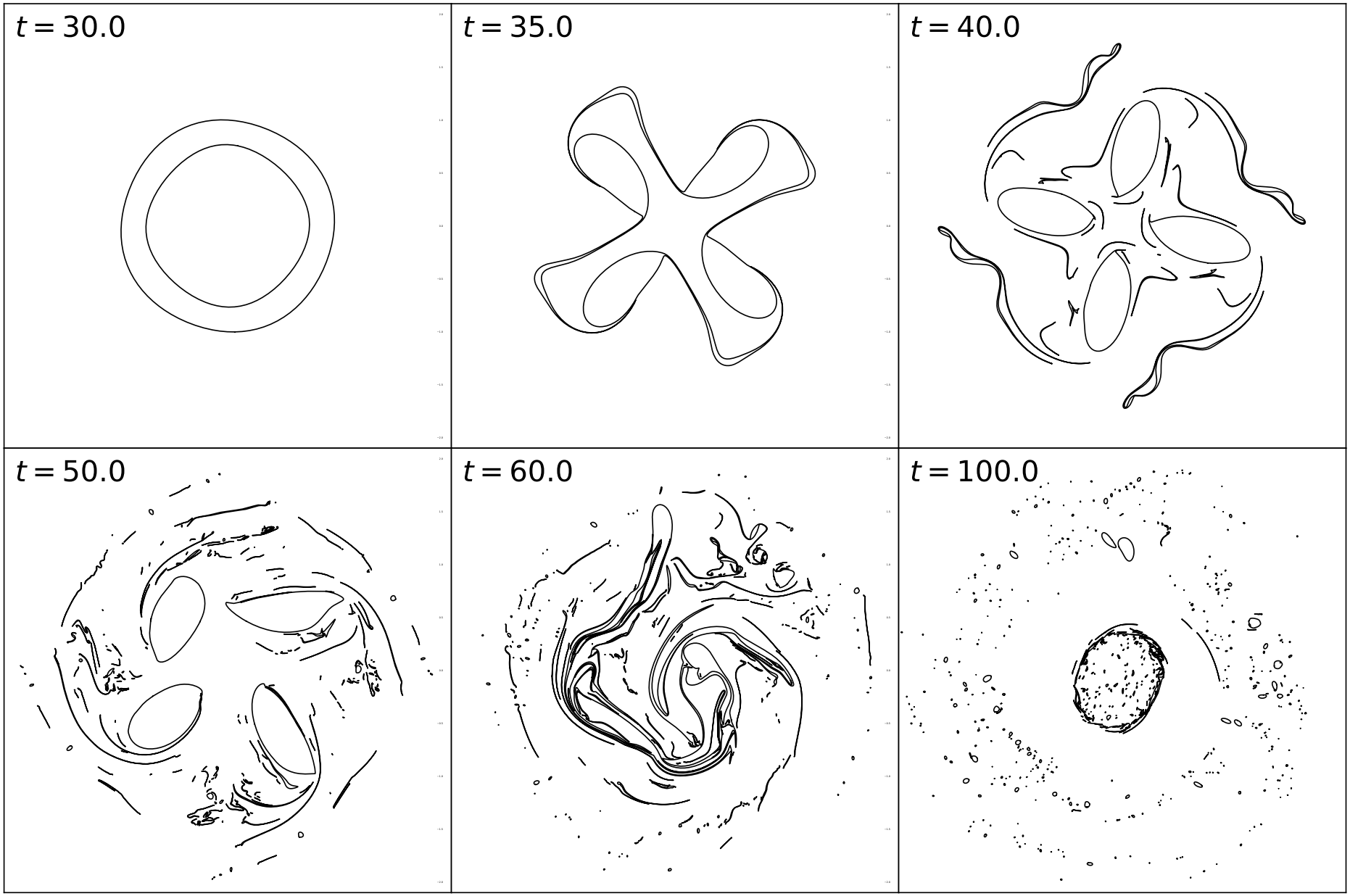
Initially, simply an unstable elliptical vortex patch with aspect ratio of 0.3.



Yes! *Simply* an unstable elliptical vortex patch with aspect ratio of 0.3.



Initially, simply an unstable annulus $0.75 \le r \le 1$ in QG with $L_D = 1$.

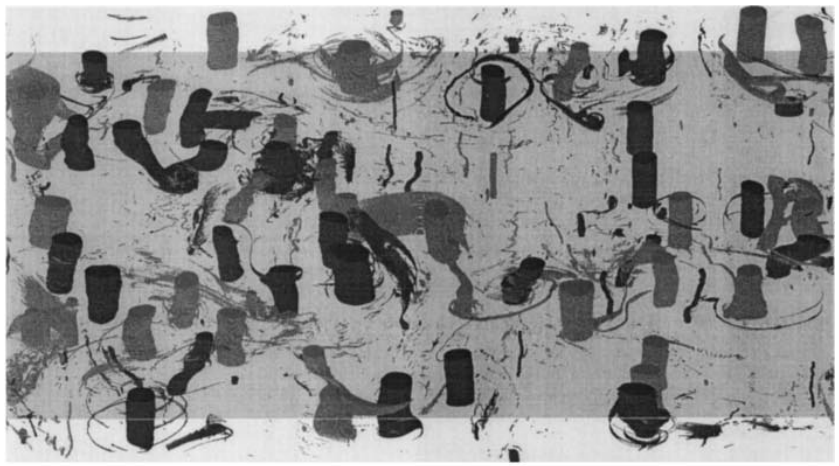


Alternative approaches: contour advection

The computational cost of Contour Surgery $\propto n^2$, where n is the total number of discrete nodes representing the contours. This is prohibitive for complex flows like *turbulence*.

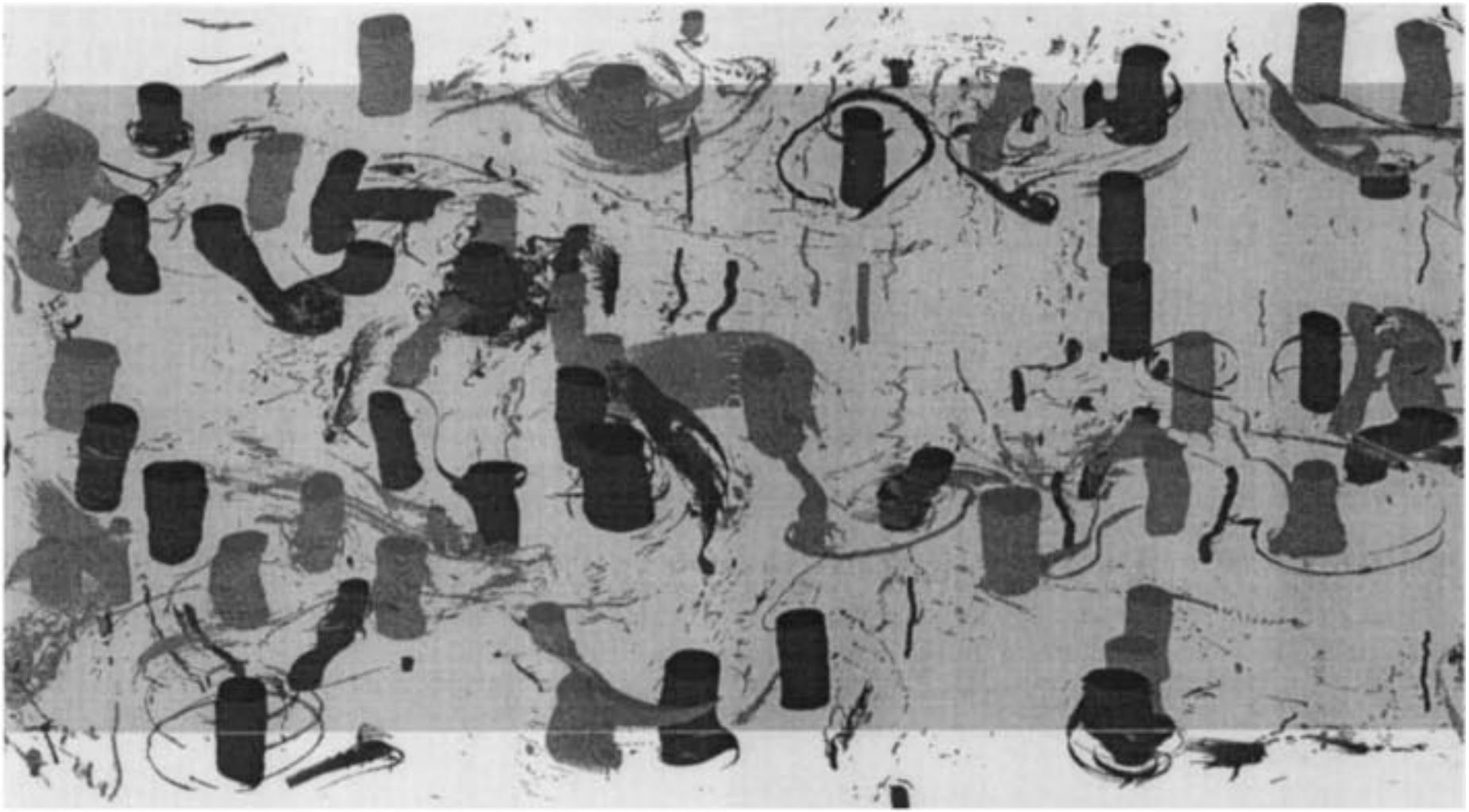
An alternative, hybrid approach is "Contour Advection" (Dritschel & Ambaum, Quart. J. Roy. Meteorol. Soc. 1997). Like PIC, Contour Advection makes use of an underlying grid to replace costly contour integrations by efficient grid-based methods, e.g. Pseudo-Spectral.

The result is a highly-efficient and accurate method, capable of studying a much wider range of physical systems. In particular, these may include models, like SW, where some of the prognostic variables are evolved *entirely on the grid*.



3D QG turbulence — PoF 1999

A closer look...



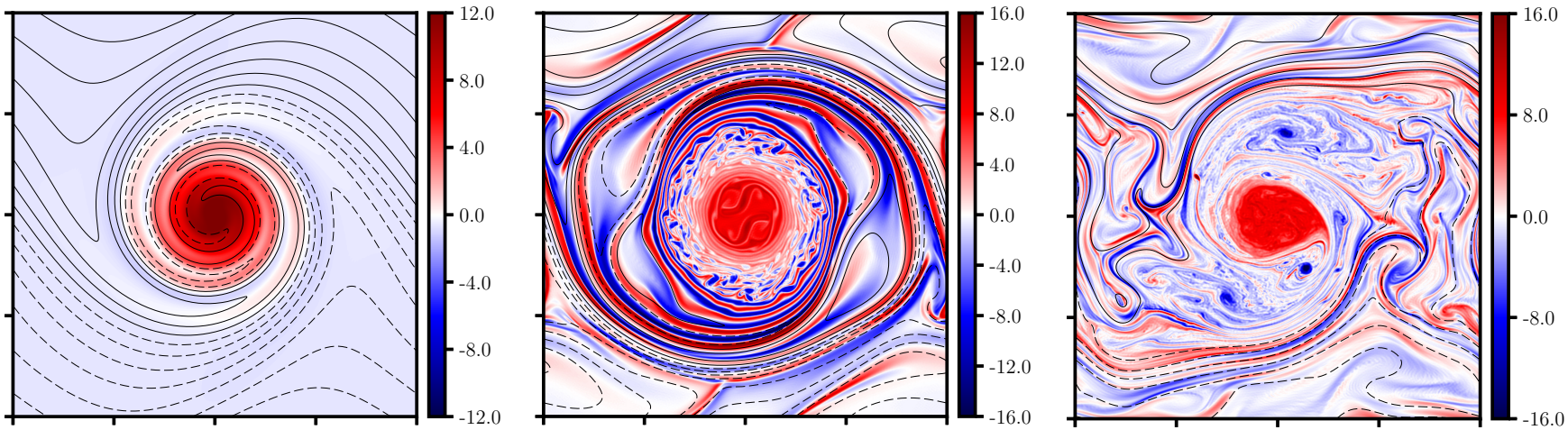
Dritschel, de la Torre Juárez & Ambaum, *Phys. Fluids* **11(6)**, (1999).

Alternative approaches: CLAM

The most advanced algorithm using Contour Advection is called the "Combined Lagrangian Advection Method" (CLAM) (Dritschel & Fontane, *J. Comput. Phys.* 2010).

CLAM goes further by evolving *gridded* representations of q alongside the contours. This is done to improve conservation of integral invariants like energy in extended, complex flow simulations characterised by frequent surgery (and hence strong dissipation of q variance or "enstrophy")

CLAM moreover permits general non-conservative effects, e.g. as induced by a magnetic field in astrophysical flows (Dritschel *et al*, *JFM* **857**, 2018).

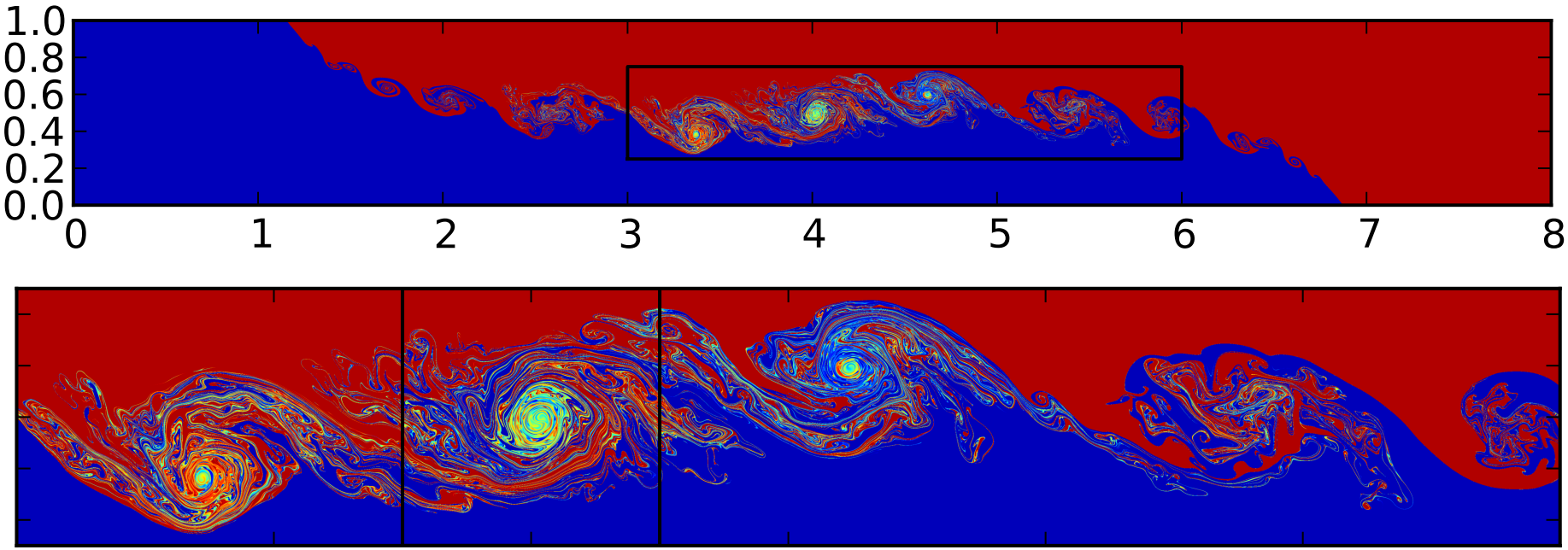


The lock-exchange problem (S. E. King & Dritschel)

2D stratified flow: $b = -g(\rho - \rho_0)/\rho_0$ is buoyancy, ω is (scalar) vorticity.

$$\frac{\mathrm{D}b}{\mathrm{D}t} = 0\,, \qquad \frac{\mathrm{D}\omega}{\mathrm{D}t} = \frac{\partial b}{\partial x}\,, \qquad \boldsymbol{\nabla}\cdot\boldsymbol{u} = 0\,.$$

At $t = 0$, (simply!) $\omega = 0$ and $b = \frac{1}{2}\tanh((x-4)/\ell)$, for $\ell \ll L_x = 8$.



Shown: buoyancy ... as simulated on a conventional computer workstation!

Numerical models: focus

Next, several of the above approaches are discussed in more detail.

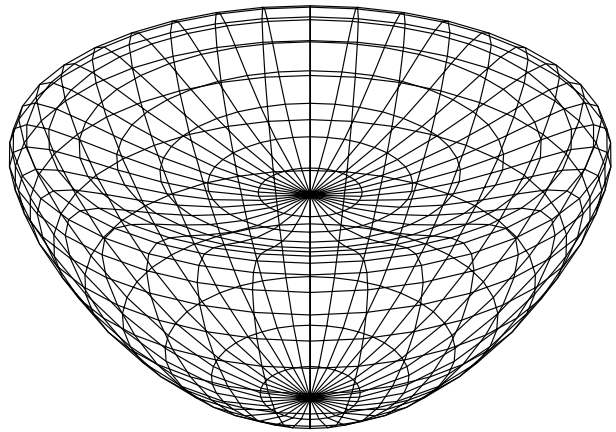
- Point vortices on closed surfaces of revolution
- Contour Dynamics & Contour Surgery
- Contour Advection & CLAM
- Hydra
- EPIC

Contact me for access to any of the related software.

Point vortices

We consider a 2D, incompressible, inviscid fluid flow on a surface S embedded in $\mathbb{R}^3$.

In general, just two coordinates $\boldsymbol{s} = (s_1, s_2)$ are required to specify any point on S.



The natural dynamical variable is the *vorticity* $\omega(\boldsymbol{s}, t)$ normal to S, since ω is conserved following fluid particles.

Incompressibility implies there exists a *streamfunction* ψ in terms of which the velocity field may be expressed as

$$\boldsymbol{u} = \boldsymbol{n} \times \nabla \psi$$

where $\boldsymbol{n}$ is the unit normal to S.

Point vortices

Then, given the definition of vorticity, $\omega = \boldsymbol{n} \cdot \nabla \times \boldsymbol{u}$, we arrive at the inversion problem $\Delta\psi = \omega$, where Δ is the *Laplace-Beltrami operator* (∇^2 restricted to S).

On the plane $\mathbb{R}^2$, Kirchhoff (1876) considered what happens if we take ω to be concentrated at a discrete set of points $\boldsymbol{x}_k(t)$, $k = 1, 2 \ldots n$. Each such point vortex has zero area A, infinite vorticity ω, but finite circulation Γ_k $(= \omega A)$.

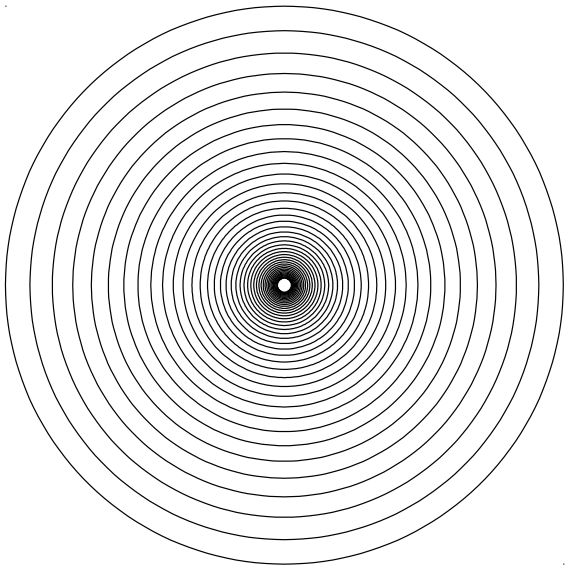
For each vortex, one can exactly solve $\Delta\psi = \omega$, using Green's function for ∇^2 on $\mathbb{R}^2$:

$$\psi(\boldsymbol{x}, t) = \frac{\Gamma_k}{2\pi} \ln \|\boldsymbol{x} - \boldsymbol{x}_k(t)\|$$

and *then* linear superposition gives the streamfunction *for any number* of vortices.

Point vortices on $\mathbb{R}^2$

The flow field is singular at each vortex $\boldsymbol{x} = \boldsymbol{x}_k(t)$, but the self-induced flow is purely azimuthal meaning it does not affect the vortex motion.



Each vortex, instead, moves according to the velocity field induced by all others.

This results in a *finite* Hamiltonian system,

$$\frac{\mathrm{d}x_k}{\mathrm{d}t} = -\frac{1}{2\pi}\sum_{j\neq k}\Gamma_j\frac{y_k - y_j}{\|\boldsymbol{x}_k - \boldsymbol{x}_j\|^2} \quad \& \quad \frac{\mathrm{d}y_k}{\mathrm{d}t} = +\frac{1}{2\pi}\sum_{j\neq k}\Gamma_j\frac{x_k - x_j}{\|\boldsymbol{x}_k - \boldsymbol{x}_j\|^2}$$

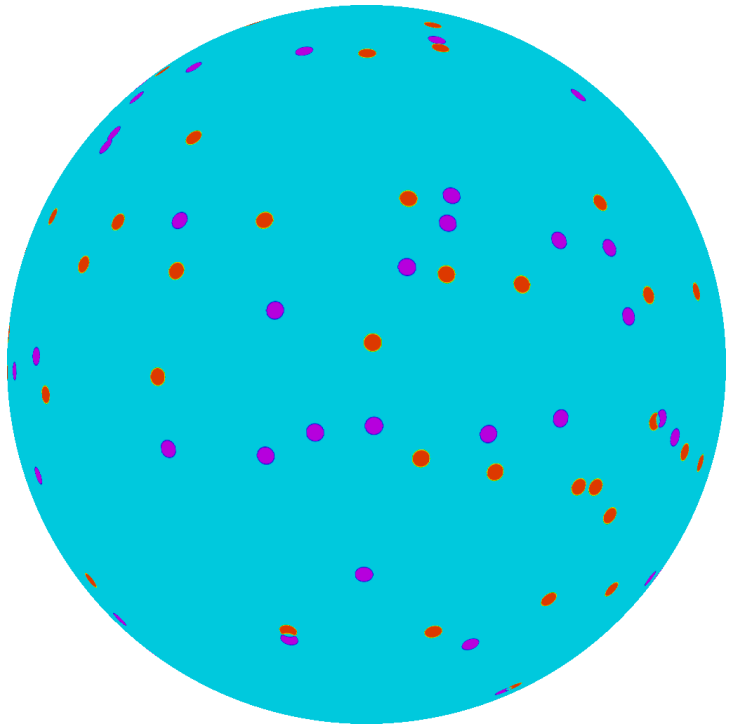
The Hamiltonian $\mathcal{H}$ is the vortex interaction energy,

$$\mathcal{H} = -\frac{1}{4\pi}\sum_k\sum_{j\neq k}\Gamma_j\Gamma_k \ln\|\boldsymbol{x}_j - \boldsymbol{x}_k\|$$

Point vortices on $\mathbb{S}^2$

Point vortex motion on the sphere $\mathbb{S}^2$ is closely analogous to that on $\mathbb{R}^2$.

Both share the same Hamiltonian if $\|\boldsymbol{x}_j - \boldsymbol{x}_k\|$ is regarded as the chord distance, i.e. $2(1 - \boldsymbol{x}_j \cdot \boldsymbol{x}_k)$ on the unit sphere.



The vortex evolution equations are also closely similar (Dritschel, *JFM* **78**, 1988):

$$\frac{\mathrm{d}\boldsymbol{x}_k}{\mathrm{d}t} = \frac{1}{4\pi} \sum_{j \neq k} \Gamma_j \frac{\boldsymbol{x}_j \times \boldsymbol{x}_k}{1 - \boldsymbol{x}_k \cdot \boldsymbol{x}_j}$$

$\Rightarrow$ For S^n, $n > 2$, see Dritschel, *Phil. Tran. Roy. Soc. A* **377** (2019).

Variable curvature surfaces

The sphere is *very* special in having constant curvature. This is part of the reason why the Hamiltonian structure is so similar to that in $\mathbb{R}^2$.

But, the sphere, and indeed any bounded surface S must satisfy the Gauss condition

$$\iint_S \nabla^2\psi \,\mathrm{d}S = \iint_S \omega \,\mathrm{d}S = 0$$

i.e. *the average vorticity must vanish*.

This is not automatically satisfied by a set of point vortices, *unless* the sum of their circulations Γ_k is zero.

The form of point vortices on closed surfaces

This is an undesirable restriction. To circumvent it, we associate with each point vortex a *uniform* compensating background vorticity, i.e. we take

$$\omega(\boldsymbol{s}) = \sum_{k=1}^{n} \Gamma_k \left(\delta(\boldsymbol{s} - \boldsymbol{s}_k) - \frac{1}{A} \right)$$

where $\boldsymbol{s}$ are coordinates on S, and A is the area of S.

This form of the vorticity, in fact, explains the close similarity between the sphere and $\mathbb{R}^2$.

Specifically, it explains why the Green function G is the same in Cartesian coordinates. *But how do we get G for arbitrary surfaces?*

Conformal transformations

This is done by an amazing (mathematical) trick!

If we can find a *conformal*, angle-preserving transformation from S to $\mathbb{R}^2$, *then* we can use the simple Green function for $\mathbb{R}^2$, re-expressed after transforming back to S.

A point in $\mathbb{R}^2$ may be represented as the complex number $z = re^{i\phi}$. The Green function for the plane, $G_p(z, z_o)$, satisfies Laplace's equation everywhere apart from the Dirac singularity at the point z_o, and we have

$$G_p(z, z_o) = \frac{1}{2\pi} \log |z - z_o|.$$

This Green function transforms conformally to any point on S — *except* one point! That is, it is the Green function $G_p(\boldsymbol{s}, \boldsymbol{s}_o)$ for the punctured surface S_p.

An example: surfaces of revolution

For a surface of revolution (about the vertical Z axis), we may express the Cartesian coordinates of any point on the surface S by

$$X = \rho(\theta)\cos\phi \qquad Y = \rho(\theta)\sin\phi \qquad Z = \zeta(\theta)$$

where $0 \le \phi \le 2\pi$ and $0 \le \theta \le \pi$ parametrize the curve $\rho(\theta)$, $\zeta(\theta)$.

We call ϕ the *longitude* and θ the *co-latitude*, by analogy with a spherical surface (for which $\rho = \sin\theta$ and $\zeta = \cos\theta$).

To find the conformal transformation from the surface S to the plane $\mathbb{R}^2$, we first compute the differential distance $\mathrm{d}s$ between two points on S:

$$\mathrm{d}s^2 = |\mathrm{d}\boldsymbol{X}|^2 = \mathrm{d}\rho^2 + (\rho\mathrm{d}\phi)^2 + \mathrm{d}Z^2 = [(\rho')^2 + (\zeta')^2]\mathrm{d}\theta^2 + \rho^2\mathrm{d}\phi^2$$

where $\rho' \equiv \mathrm{d}\rho/\mathrm{d}\theta$.

The goal is to express $\mathrm{d}s^2$ in planar polar coordinates (r,ϕ) as

$$\mathrm{d}s^2 = \lambda^2(\mathrm{d}r^2 + r^2\mathrm{d}\phi^2)$$

where λ^2 is a *conformal factor*.

Distances are not preserved but angles are under a conformal transformation.

Hence, equating the expressions for $\mathrm{d}s^2$ $\Rightarrow$

$$\lambda^2 \mathrm{d}r^2 = [(\rho')^2 + (\zeta')^2]\mathrm{d}\theta^2 \qquad \& \qquad \lambda^2 r^2 = \rho^2$$

which implies

$$\frac{r'}{r} = \frac{\sqrt{(\rho')^2 + (\zeta')^2}}{\rho} \qquad \& \qquad \lambda = \frac{\rho}{r}.$$

Starting from $r(0) = 0$, in principle these equations can be solved for $r(\theta)$ and $\lambda(\theta)$, given $\rho(\theta)$ and $\zeta(\theta)$.

Note that $r \to \infty$ when $\theta \to \pi$.
That is, the point $\theta = \pi$ is mapped to infinity.

More correctly, *the conformal transformation applies only to the punctured surface* S_p!

Hence, the Green function

$$G_p(s, s_o) = \frac{1}{2\pi} \log |z - z_o| ,$$

with $\boldsymbol{s} = \{\theta, \phi\}$ and $z = r(\theta)e^{i\phi}$, is valid *only* for S_p.

Repairing the hole

The Green function $G_p(\boldsymbol{s}, \boldsymbol{s}_o)$ does not satisfy the Gauss condition.

However, consider the function

$$G(\boldsymbol{s}, \boldsymbol{s}_o) = G_p(\boldsymbol{s}, \boldsymbol{s}_o) - \frac{1}{A}\iint_{S_p} G_p(\boldsymbol{s}, \boldsymbol{s}_o)\,\mathrm{d}S - \frac{1}{A}\iint_{S_p} G_p(\boldsymbol{s}, \boldsymbol{s}_o)\,\mathrm{d}S_o$$

where A is the area of S. *Note:* two extra terms are included to maintain the required symmetry of G. The second term on the r.h.s. is a function only of s_o, while the third is a function only of s.

Applying the ∇^2 operator, we find

$$\nabla^2 G(\boldsymbol{s}, \boldsymbol{s}_o) = \nabla^2 G_p(\boldsymbol{s}, \boldsymbol{s}_o) - \frac{1}{A} = \delta(\boldsymbol{s} - \boldsymbol{s}_o) - \frac{1}{A}$$

Resolution

So, integrating over S_p, we obtain

$$\iint_{S_p} \nabla^2 G(\boldsymbol{s}, \boldsymbol{s}_o) \mathrm{d}S = 0$$

as required. *We're in business!* This is our Green function!

This permits us to express the streamfunction $\psi(\boldsymbol{s})$ induced by the point vortices at $\boldsymbol{s} = \boldsymbol{s}_k$:

$$\psi(\boldsymbol{s}) = \sum_{k=1}^{n} \Gamma_k G(\boldsymbol{s}, \boldsymbol{s}_k).$$

But, this is not valid at a vortex position $\boldsymbol{s} = \boldsymbol{s}_j$ where G is singular.

Regularisation

The singular part, which does not induce any flow, is removed by subtracting

$$\frac{1}{2\pi}\log d(\boldsymbol{s},\boldsymbol{s}_j)$$

from $G(\boldsymbol{s},\boldsymbol{s}_j)$, where $d(\boldsymbol{s},\boldsymbol{s}_j)$ is the geodesic distance between $\boldsymbol{s}$ and $\boldsymbol{s}_j$.

Hence, we arrive at the expression for the streamfunction at the jth vortex:

$$\psi(\boldsymbol{s}_j) = \sum_{j=1,j\neq k}^{n} \Gamma_k G(\boldsymbol{s}_k,\boldsymbol{s}_j) + \Gamma_j R(\boldsymbol{s}_j)$$

where

$$R(\boldsymbol{s}_j) = \lim_{\boldsymbol{s}\to\boldsymbol{s}_j}\left(G(\boldsymbol{s},\boldsymbol{s}_j) - \frac{1}{2\pi}\log d(\boldsymbol{s},\boldsymbol{s}_j)\right)$$

is known as the *Robin function*. ⋆ ⋆ ⋆ *new* ⋆ ⋆ ⋆

Equations of Motion

Skipping many details, this streamfunction permits one to define the Hamiltonian $\mathcal{H}$, in terms of which one can derive the equations of motion for each vortex, i.e. for $\boldsymbol{s}_k(t)$ (Dritschel & Boatto, *Proc. Roy. Soc. A* **471**, 20140890).

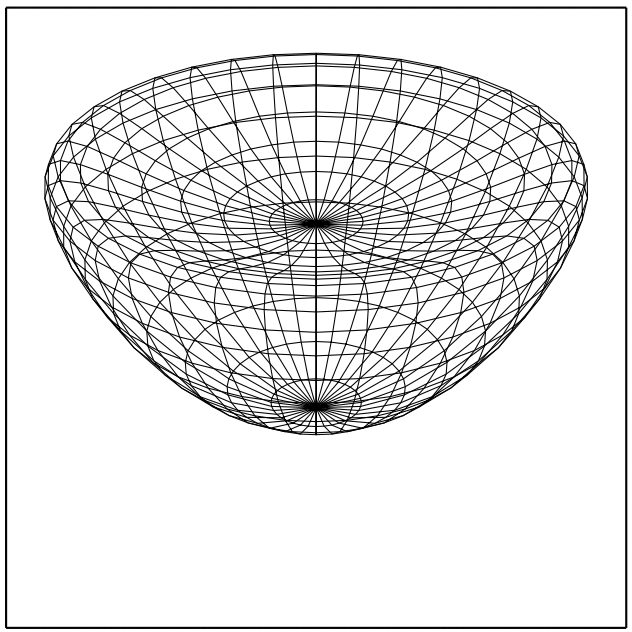
This is illustrated here for a surface of revolution, which is rotationally symmetric about the z axis.

For example, the 'bean-shaped' surface has

$$X = \rho(\theta)\cos\phi\,, \quad Y = \rho(\theta)\sin\phi\,, \quad Z = \zeta(\theta)$$

with $\rho(\theta) = \sin\theta$ & $\zeta(\theta) = a\sin^2\theta + b\cos\theta$

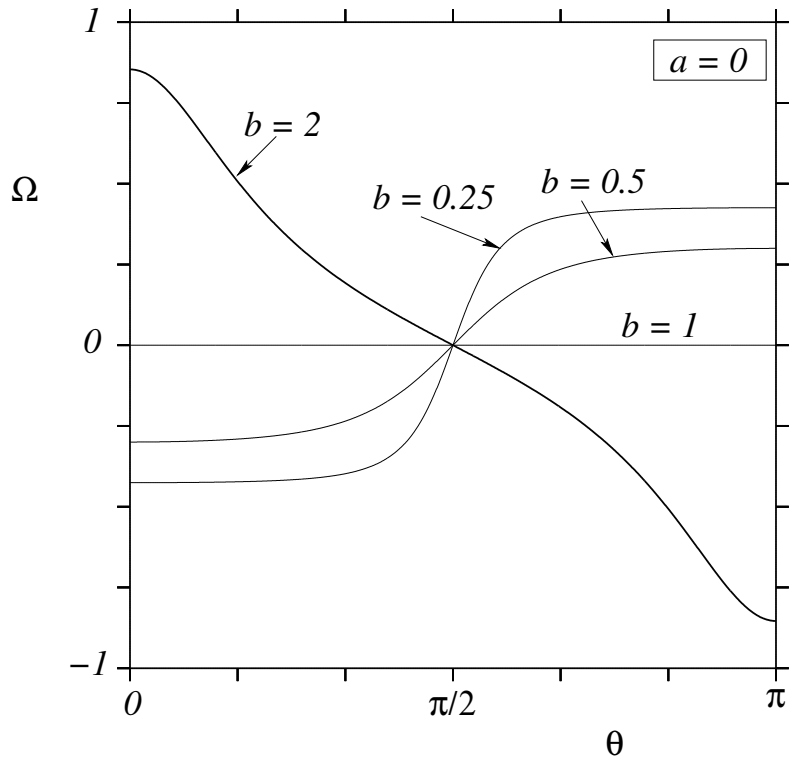
The ellipsoid is recovered by setting $a = 0$.



Point vortices on an ellipsoid of revolution

A single vortex on the plane $\mathbb{R}^2$ or on a sphere $\mathbb{S}^2$ is stationary, by symmetry. This is no longer true on surfaces with variable curvature.

On a surface of revolution, a vortex rotates at a rate Ω about the axis of symmetry, depending on its co-latitude θ. Below, $\Omega(\theta)$ is plotted for an ellipsoid, $x^2 + y^2 + z^2/b^2 = 1$, for a single vortex of circulation $\Gamma = 2\pi$.



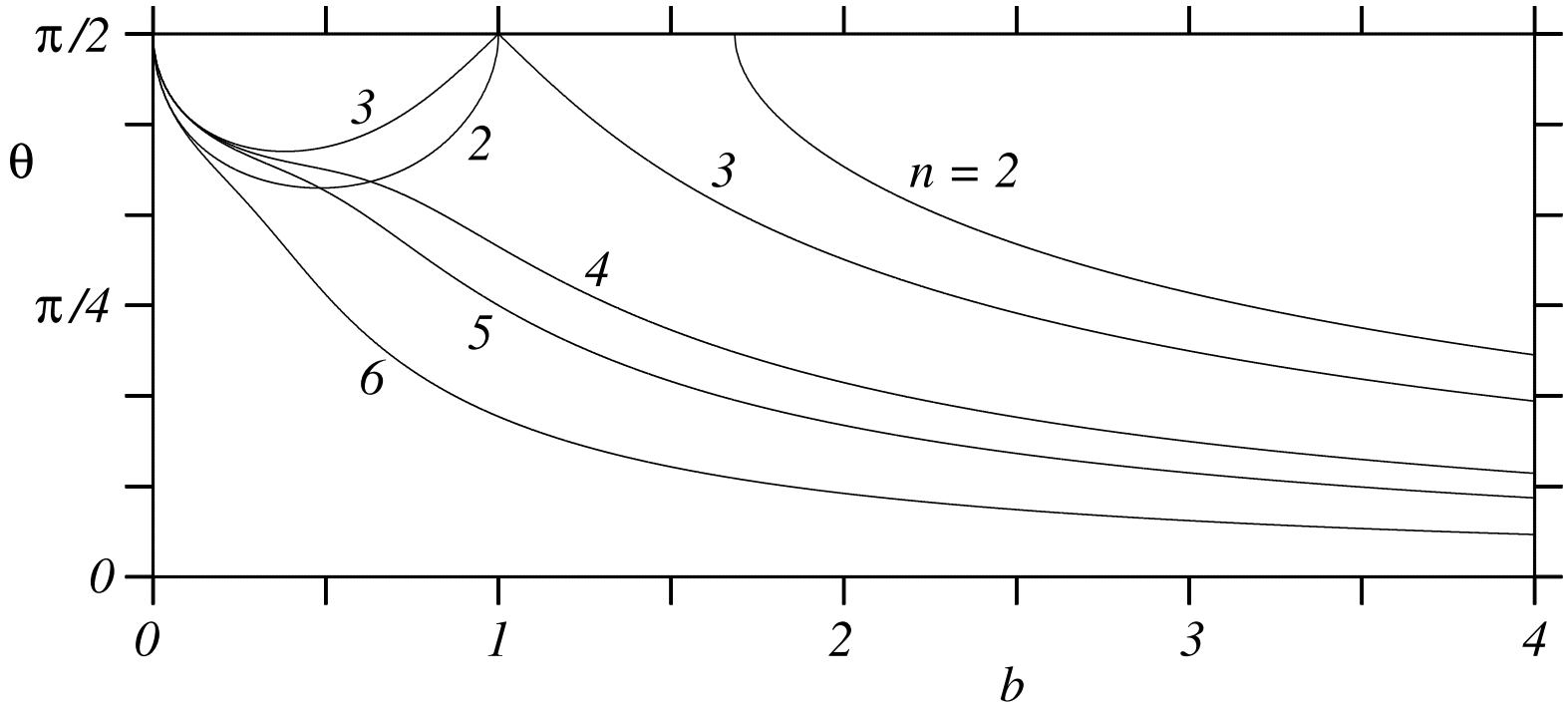
$b > 1$: prolate

$b = 1$: sphere

$b < 1$: oblate

Point vortices on an ellipsoid of revolution: stability

Instability domains for $n = 2$ to 6 vortices arranged on a ring of constant co-latitude θ, as a function of the aspect ratio of the ellipsoid b.

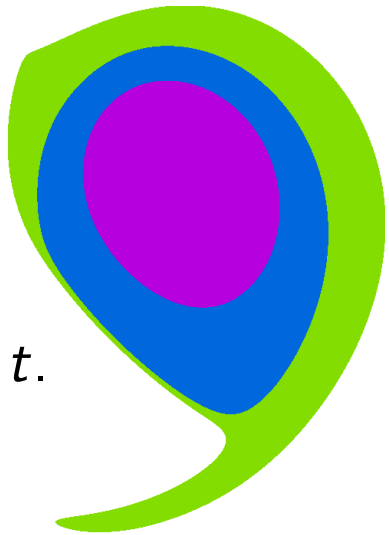


For a given n, there is linear instability for co-latitudes between $\theta_n(b)$ and $\pi - \theta_n(b)$, centred on the equator, wherever $\theta_n(b) < \pi/2$. [movies]

Contour Dynamics

We first begin by deriving the equations for Contour Dynamics for QG flow in $\mathbb{R}^2$.

We assume that the material invariant q, the "potential vorticity", is *piecewise-uniform*. Specifically, q jumps by Δq_k crossing contour C_k *inwards*.



Let points on C_k be denoted $\boldsymbol{x} = \boldsymbol{x}_k$.

If we assume $q = 0$ "at infinity", then the C_k and Δq_k uniquely specify the spatial distribution $q(\boldsymbol{x}, t)$ at any time t.

The first task is to find ψ from $\nabla^2\psi - \gamma^2\psi = q$ in $\mathbb{R}^2$, where here $\gamma = 1/L_D$ is the inverse of the Rossby deformation length.

Contour Dynamics

The solution makes use of the well-known Green function involving the modified Bessel function K_0:

$$\psi(\boldsymbol{x},t) = -\frac{1}{2\pi}\iint_{\mathbb{R}^2} q(\boldsymbol{x}',t)K_0(\gamma\|\boldsymbol{x}'-\boldsymbol{x}\|)\,\mathrm{d}x'\,\mathrm{d}y'\,.$$

But $q(\boldsymbol{x}',t)$ is piecewise-uniform: $q = q_k$ in regions R_k. Hence,

$$\psi(\boldsymbol{x},t) = -\frac{1}{2\pi}\sum_k q_k \iint_{R_k} K_0(\gamma\|\boldsymbol{x}'-\boldsymbol{x}\|)\,\mathrm{d}x'\,\mathrm{d}y'\,.$$

One *can* reduce this to contour integrals, but we don't need to!. Consider instead the velocity field $\boldsymbol{u} = (-\partial\psi/\partial y, \partial\psi/\partial x)$:

$$\boldsymbol{u}(\boldsymbol{x},t) = -\frac{1}{2\pi}\sum_k q_k \iint_{R_k} \left(-\frac{\partial}{\partial y}, \frac{\partial}{\partial x}\right) K_0(\gamma\|\boldsymbol{x}'-\boldsymbol{x}\|)\,\mathrm{d}x'\,\mathrm{d}y'\,.$$

Contour Dynamics

This looks like we are going nowhere; however, K_0 is symmetric in $\boldsymbol{x}$ and $\boldsymbol{x}'$. Hence, we can exchange derivatives of $\boldsymbol{x}$ for $\boldsymbol{x}'$ if we also change sign:

$$\boldsymbol{u}(\boldsymbol{x},t) = \frac{1}{2\pi}\sum_k q_k \iint_{R_k} \left(-\frac{\partial}{\partial y'}, \frac{\partial}{\partial x'}\right) K_0(\gamma\|\boldsymbol{x}' - \boldsymbol{x}\|)\,\mathrm{d}x'\,\mathrm{d}y' \,.$$

This is perfectly set up for Stokes' Theorem, yielding

$$\boldsymbol{u}(\boldsymbol{x},t) = \frac{1}{2\pi}\sum_k \Delta q_k \oint_{C_k} K_0(\gamma\|\boldsymbol{x}'_k - \boldsymbol{x}\|)\,\mathrm{d}\boldsymbol{x}'_k$$

where $\boldsymbol{x}'_k$ is a point on C_k. Note that q_k has now switched to Δq_k.

All we have used is that the Green function ($\propto K_0$) is symmetric in $\boldsymbol{x}$ and $\boldsymbol{x}'$. The above result thus generalises considerably (Dritschel, *Comput. Phys. Rep.* 1989).

Contour Dynamics

This expression for $\boldsymbol{u}$ is valid everywhere, *including on the same or other contours*.

Hence, evaluating at $\boldsymbol{x} = \boldsymbol{x}_j$, a point on C_j, we have

$$\frac{\mathrm{d}\boldsymbol{x}_j}{\mathrm{d}t} = \boldsymbol{u}(\boldsymbol{x}_j, t) = \frac{1}{2\pi} \sum_k \Delta q_k \oint_{C_k} K_0(\gamma \|\boldsymbol{x}'_k - \boldsymbol{x}_j\|)\, \mathrm{d}\boldsymbol{x}'_k$$

— using the definition of the point's velocity $\boldsymbol{u}(\boldsymbol{x}_j, t) = \mathrm{d}\boldsymbol{x}_j/\mathrm{d}t$.

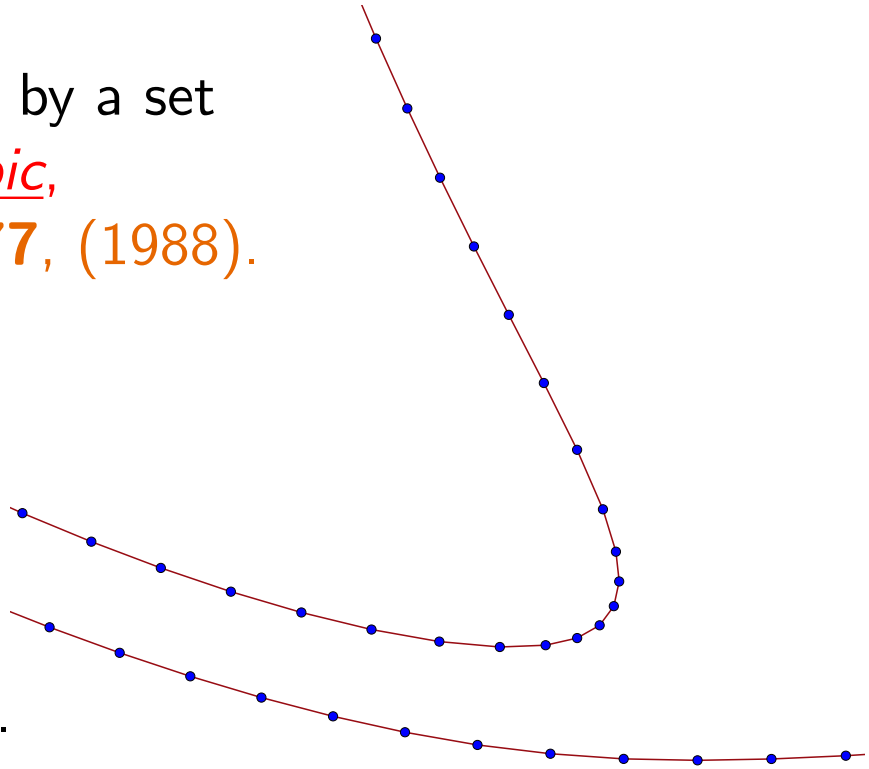
The above equations are the equations of Contour Dynamics. Notice they are entirely self-contained: this is a closed dynamical system in the points on the contours. The points between the contours are not relevant!

Nonetheless, it is an infinite order (Hamiltonian) system, as there are an uncountable number of points on the contours. (The Hamiltonian is the *total energy*, kinetic + potential.)

Contour Dynamics: numerics

In general, the contours deform in ways that cannot be calculated analytically.

Numerically, contours are approximated by a set of nodes connected by splines, here *cubic*, following Dritschel, *J. Comput. Phys.* **77**, (1988).



The node spacing is controlled by the local curvature, and is adjusted every few time steps.

The total number of nodes n may vary in time, typically considerably.

The computational cost rises like n^2. This rapidly becomes prohibitive.

Contour Surgery

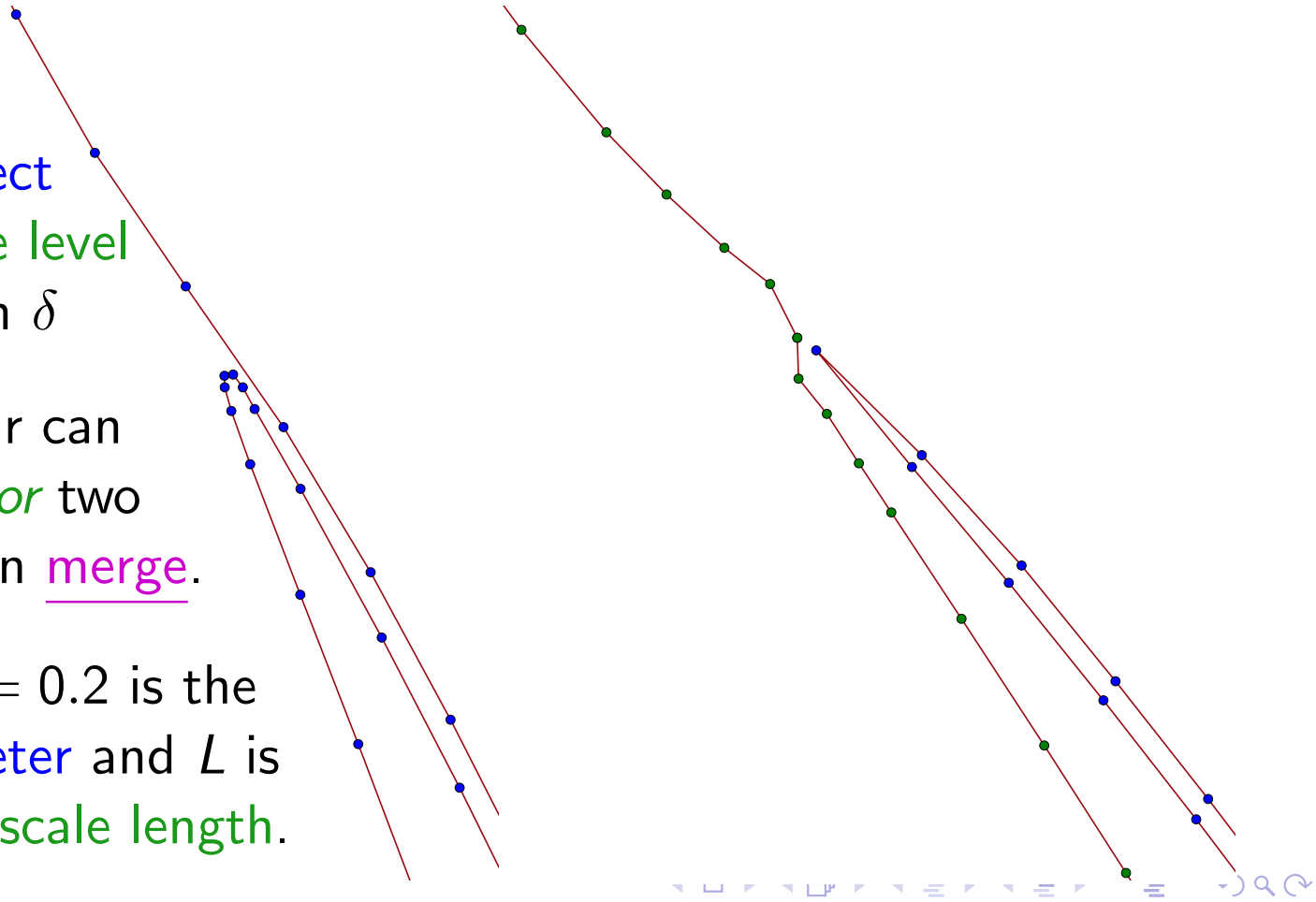
A solution is to *limit* the growth in complexity by fixing a smallest scale, the *surgical scale*, δ (Dritschel, *J. Comput. Phys.* **77**, 1988).

The node density is controlled to limit the maximum curvature to $\sim \delta^{-1}$.

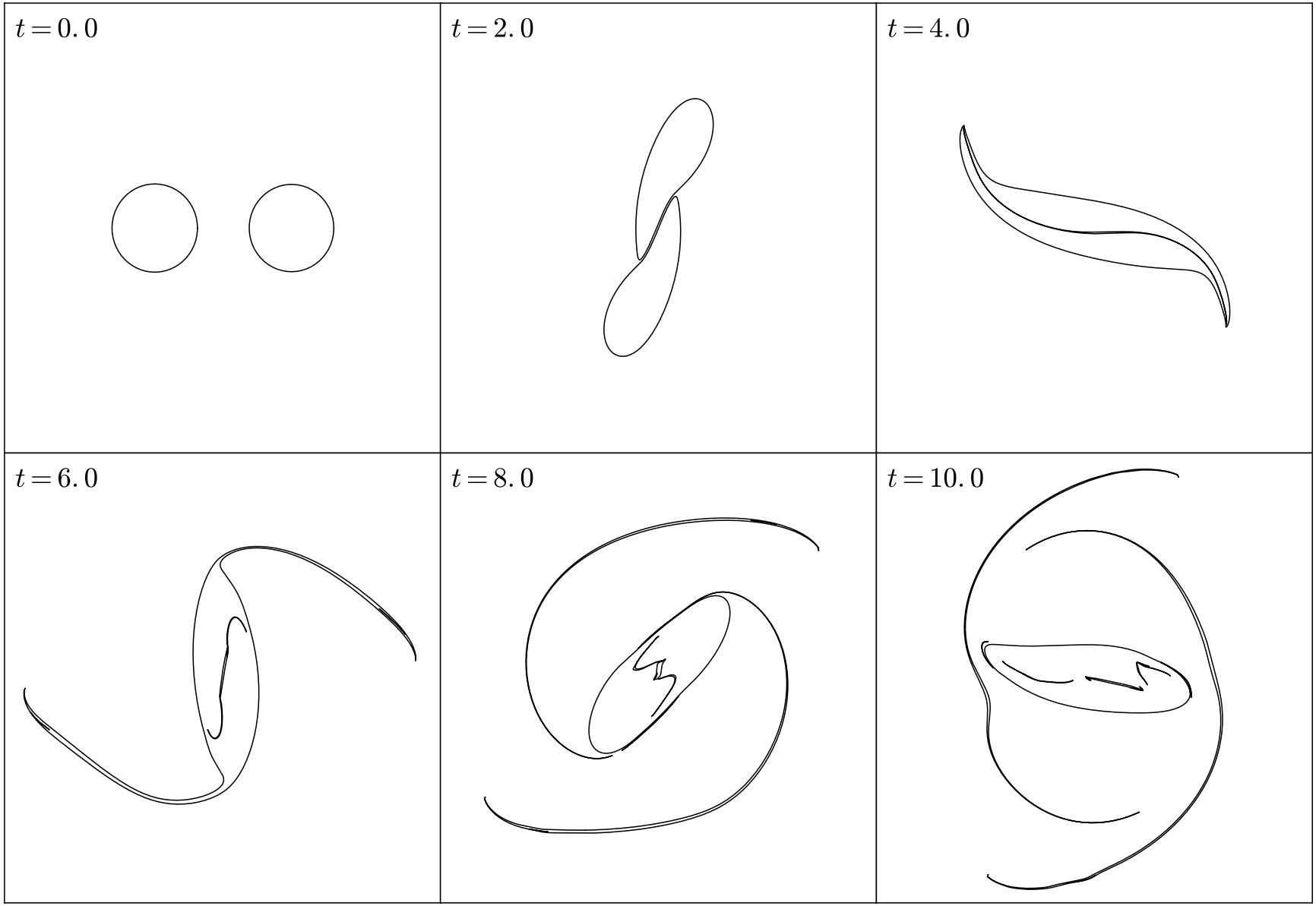
Surgery is used to topologically reconnect contours of the same level which are closer than δ

In this way, a contour can split into two parts, *or* two different contours can merge.

$\delta = \frac{1}{4}\mu^2 L$, where $\mu = 0.2$ is the node-spacing parameter and L is the (specified) large-scale length.



Initially circular patches, with radii $R_1 = 1$ & $R_2 = 0.99$, separated by $d = 3.2$. Here $\gamma = 0$. See also Dritschel & Waugh, Phys. Fluids (1992).



Contour Advection

For complex flows like 2D turbulence, or for flows in bounded domains or on surfaces, contour surgery is impractical. The computational cost may be excessive, and moreover a simple Green function *may not exist*.

The alternative is to replace the costly/impractical contour integrations by efficient grid-based methods. This requires the use of an underlying grid as in Particle-In-Cell.

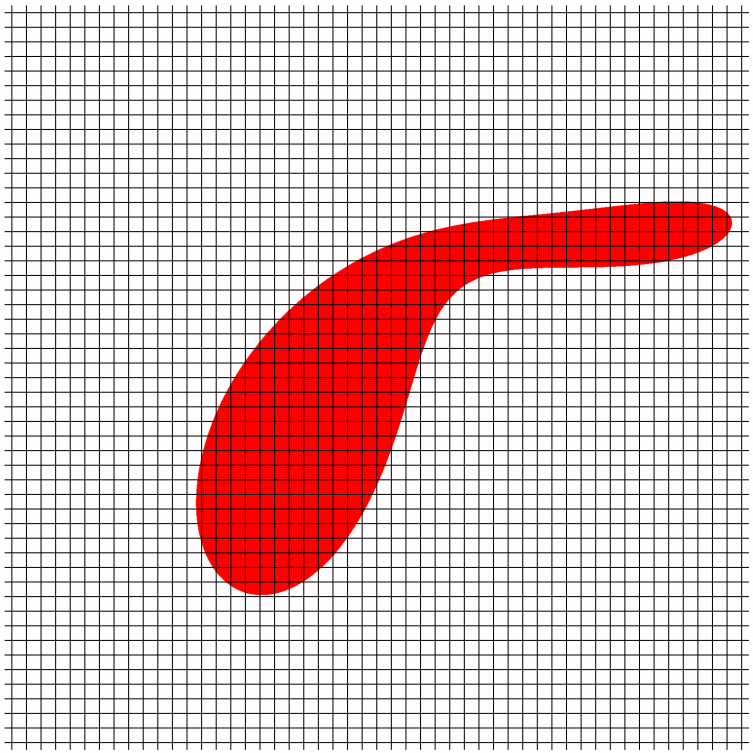
This is the approach taken by "Contour Advection".

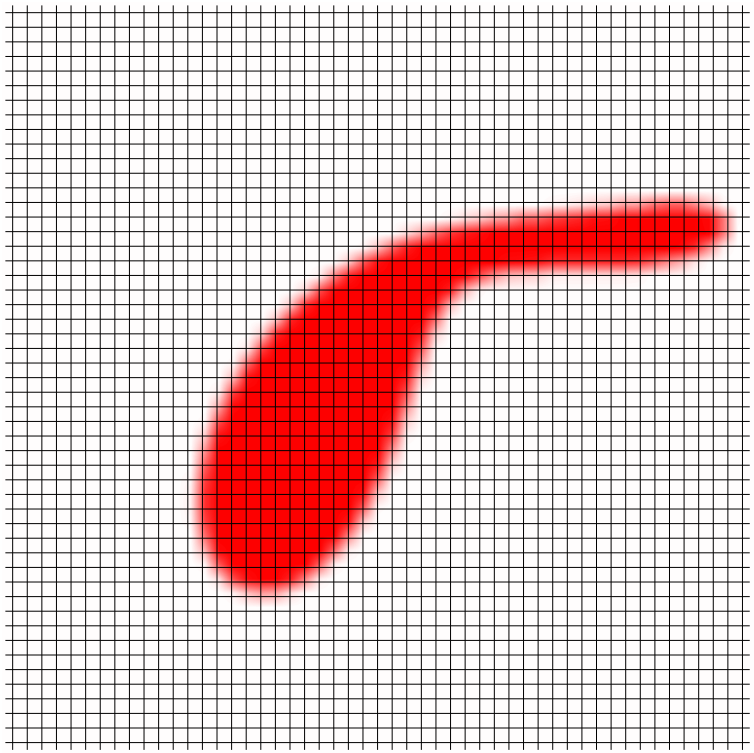
(Dritschel & Ambaum, Quart. J. Roy. Meteorol. Soc. 1997.)

Contour Advection (CA) furthermore permits one to study *a much wider range of models*, such as shallow-water, where only the potential vorticity q is represented by contours. Other fields are treated conventionally, i.e. by grid-based methods.

Contour Advection

Note: Only materially-conserved fields q are represented by contours.





CA makes use of a fast-fill algorithm, which creates a gridded representation of q for use in inversion, i.e. in finding $\boldsymbol{u}$ from q (and possibly other fields, as in shallow-water; see Dritschel, Polvani & Mohebalhojeh, *Mon. Wea. Rev.* 1999).

Contour Advection

The fast-fill algorithm uses a grid 4 times finer in each direction than the 'inversion grid', where e.g. the velocity field $\boldsymbol{u}$ is computed.

1–2–1 averaging is used to coarsen q back to the inversion grid.

While this procedure results in a loss of information at small scales, the impact on $\boldsymbol{u}$ is remarkably small. This is because $\boldsymbol{u}$ is obtained effectively from an integration over q: large scales contribute the most.

The velocity is computed conventionally, e.g. using Fast Fourier Transforms and spectral methods (in periodic domains). This is standard.

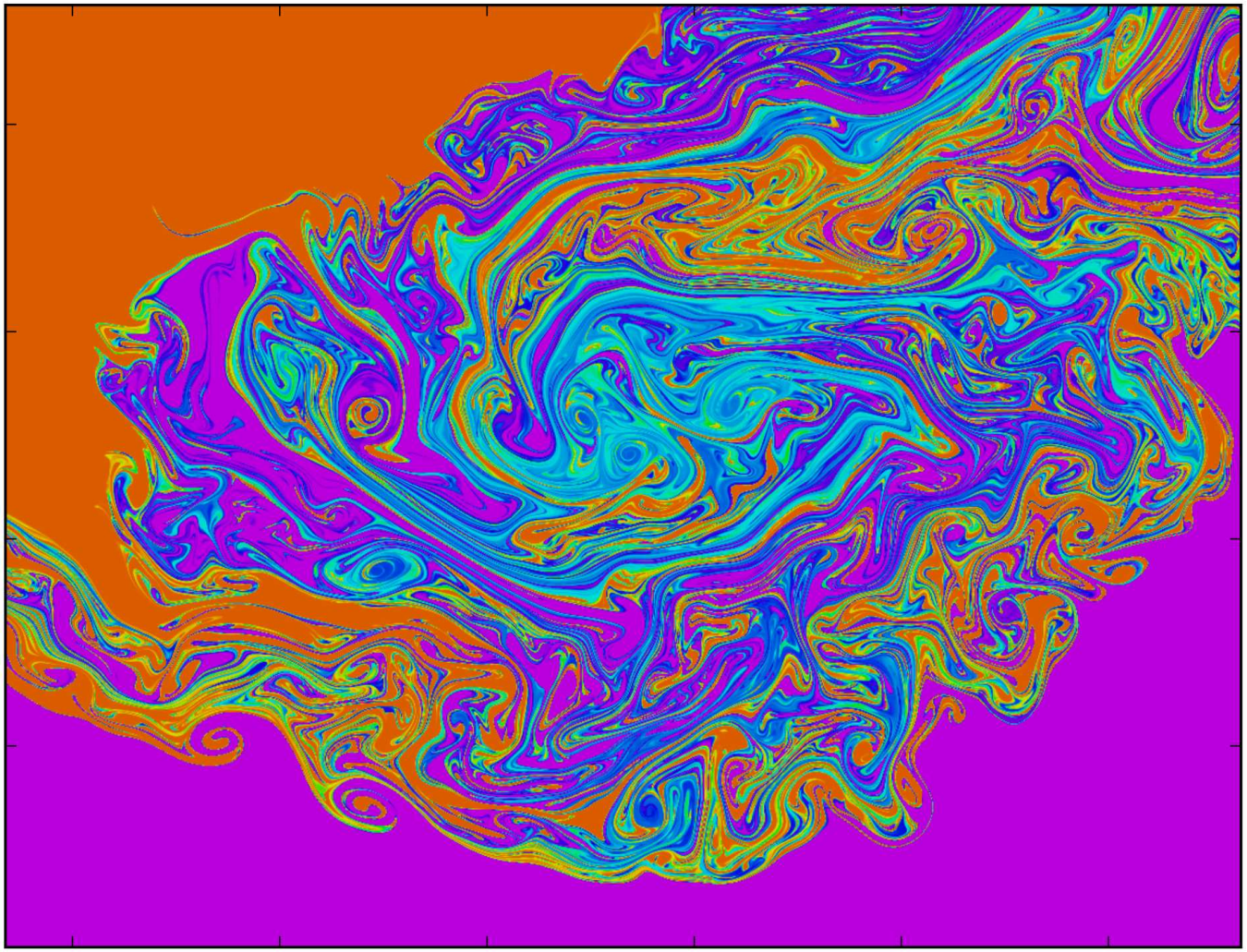
The contour nodes $\boldsymbol{x}_i$ are evolved simply by solving

$$\frac{\mathrm{d}\boldsymbol{x}_i}{\mathrm{d}t} = \boldsymbol{u}(\boldsymbol{x}_i, t)$$

just ODEs, where $\boldsymbol{u}$ is bi-linearly interpolated to $\boldsymbol{x}_i$ from the grid.

Contour Advection

Things can get a bit messy in flows with many contour levels, as here in the materially-conserved density field in a Kelvin–Helmholtz billow:



Contour Advection

Surgery and node-redistribution can result in crossing contour levels.
Such errors can be virtually-eliminated by periodically re-contouring.

- The materially-conserved field q is first converted to a grid as fine as the surgical scale δ, a grid 16 times finer than the inversion grid — in each direction. This uses the same fast-fill algorithm.
- Contours are then efficiently re-built and used until the next re-contouring — after every 20 applications of surgery.

Note: re-contouring acts like surgery (however is much more expensive).

CLAM

The most advanced numerical method based on contour advection is the "Combined Lagrangian Advection Method" (CLAM).

This greatly extends CA by allowing non-conservative forcing, i.e.

$$\frac{\mathrm{D}q}{\mathrm{D}t} = F$$

for *general* forcing F. An example is 2D magneto-hydrodynamics,

$$\frac{\mathrm{D}\omega}{\mathrm{D}t} = B_0\frac{\partial j}{\partial x} - J(A,j)\,, \qquad \frac{\mathrm{D}A}{\mathrm{D}t} = B_0\frac{\partial \psi}{\partial x} - \eta j$$

$$j = -\nabla^2 A\,, \qquad \omega = \nabla^2\psi\,, \qquad \boldsymbol{u} = \boldsymbol{\nabla}^\perp\psi = \left(-\frac{\partial\psi}{\partial y}, \frac{\partial\psi}{\partial x}\right)$$

where j is the *current density* (the curl of the magnetic field $\boldsymbol{B}$) and B_0 is the mean x component of $\boldsymbol{B}$. Here, η is the *magnetic diffusivity*.
$J(f,g) = \partial f/\partial x\,\partial g/\partial y - \partial f/\partial y\,\partial g/\partial x$ is the Jacobian operator.

CLAM: comparison with the pseudo-spectral method

From Dritschel & Tobias, *J. Fluid Mech.* **703** (2012)

PSM

CLAM

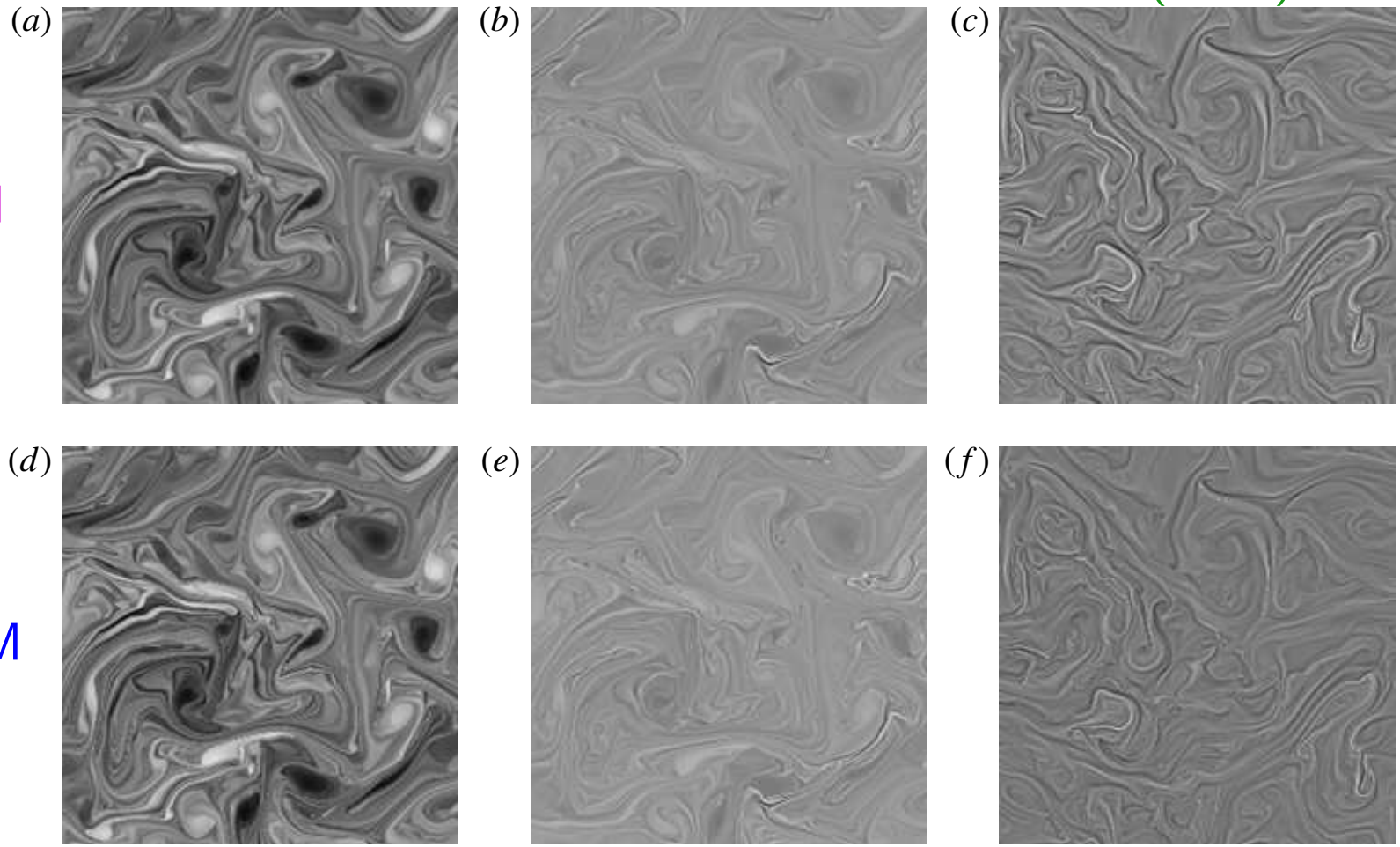


FIGURE 8. Final states of vorticity ω in the PSM simulations (a–c) and in the CLAM simulations (d–f) for $Pm = 1/64$ and $\gamma = 0.2$ (a, d), 1.0 (b, e) and 10.0 (c, f). Note that we have zoomed in to show a quarter of the computational domain.

CLAM: comparison with the pseudo-spectral method

- Here, in CLAM, we used an 'inversion' grid resolution of 1024^2 (in doubly-periodic geometry).

- In the pseudo-spectral method (PSM), we used a resolution of 8192^2 in order to obtain comparably weak dissipation of vorticity, ω. Such dissipation is *essential* for numerical stability.

Cost? The CLAM simulation required 1.6 hours on a single 3.2 GHz processor. The PSM simulation required 63 hours on 128 processors!

This huge gain is due to two factors.

- CLAM uses contours for ω, which while not conserved, is dominantly advected. Only contour surgery dissipates ω at scales 16 times finer than the inversion grid.
- The advecting velocity field $\boldsymbol{u}$ is very well approximated on the inversion grid. Subgrid scales contribute little.

CLAM: overview of the method

CLAM is built on the PSM (it can be built on any grid-based method). The PSM uses Fast Fourier Transforms (FFTs) to carry out all linear operations, e.g. differentiation, inversion of ∇^2, *etc*. These can be done *exactly* by wavenumber multiplication in spectral space.

All nonlinear products are carried out in physical space, i.e. on the grid. De-aliasing is typically required to avoid spurious results at high wavenumbers — this uses the "2/3 rule" (retaining only 2/3 of the wavenumbers in each direction).

Time evolution requires numerical diffusion for stability. Commonly hyper-diffusion is used, e.g.

$$\frac{\mathrm{D}q}{\mathrm{D}t} = F - \nu(-\nabla^2)^m q$$

for $m > 1$. Molecular diffusion corresponds to $m = 1$, but is considered much too diffusive in many applications. $m = 3$ is a common choice.

CLAM: overview of the method

CLAM largely avoids this diffusion by using contours, but *additionally uses two* gridded representations of q, to enable non-conservative forcing F.

The field q is decomposed as

$$q = (1 - \mathcal{F})q_c + \mathcal{F}q_s + q_d$$

where

$$\frac{\mathrm{D}q_c}{\mathrm{D}t} = 0\,, \qquad \frac{\mathrm{D}q_s}{\mathrm{D}t} = 0\,, \qquad \frac{\mathrm{D}q_d}{\mathrm{D}t} = F - \nu(-\nabla^2)^m q_d$$

and where q_c is evolved by contour advection while q_s & q_d are two gridded fields evolved by the PSM.

Above $\mathcal{F}$ is a low-pass filter. Thus q blends the small-scale part of q_c with the large-scale part of q_s, and adds changes generated by the forcing F in q_d.

CLAM: overview of the method

To ensure minimal diffusion, the fields q_s and q_d are re-initialised *at the beginning of every time step*: the entire q field is used to initialise q_s, while q_d is assigned the residual:

$$q_s = q\,, \qquad q_d = (1-\mathcal{F})(q - q_c)\,.$$

In this way, $q = (1-\mathcal{F})q_c + \mathcal{F}q_s + q_d$ *is unchanged*.

This procedure keeps q_d as small as possible, thereby minimising the impact of (numerical) hyper-diffusion on the evolution of q.

Moreover, q_s can be evolved *without diffusion* in the PSM, since it is re-initialised with all of q each time step to give the best possible starting conditions.

CLAM: overview of the method

To incorporate the forcing F into the contours q_c, and keep q_d relatively small, occasionally the *entire* field q is re-contoured on an ultra-fine grid, 16 times smaller than the inversion grid in each direction. The contours q_c are converted to grid values directly, while the remainder of q is interpolated.

This is done when the total "twist" τ, a measure of the net straining, exceeds a prescribed value:

$$\tau \equiv \int_{t_0}^{t} |\omega_{\rm max}| \, {\rm d}t > 2.5 \, ,$$

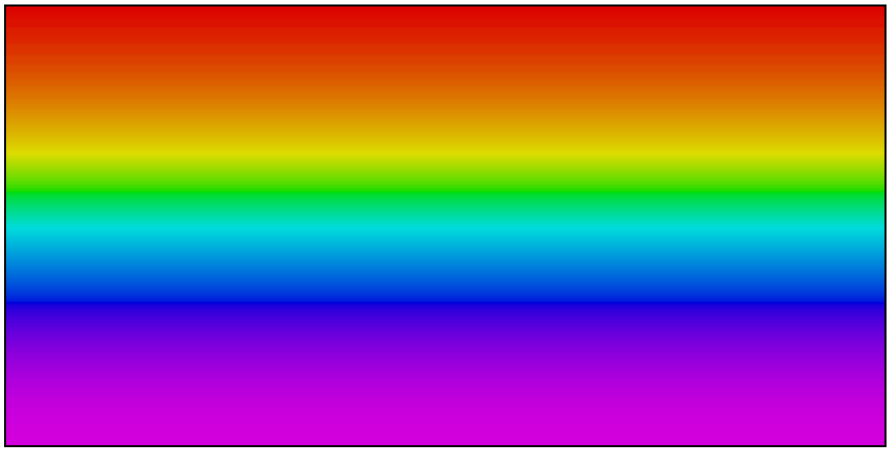
where t_0 is the last time re-contouring occurred (or the initial time).

In this way, the contours receive the forcing *indirectly*, while accurately representing a much wider range of scales than either q_s or q_d.

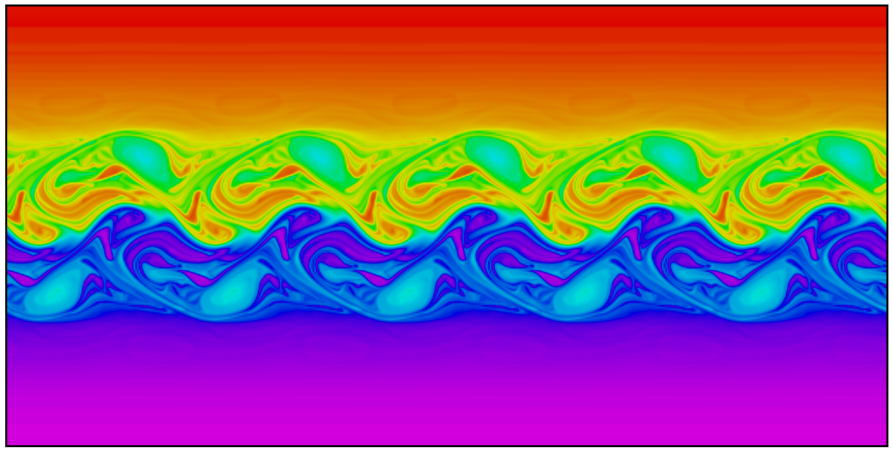
CLAM: application to spherical shallow-water flows

PV field q in a thermally-forced flow; time units in days.

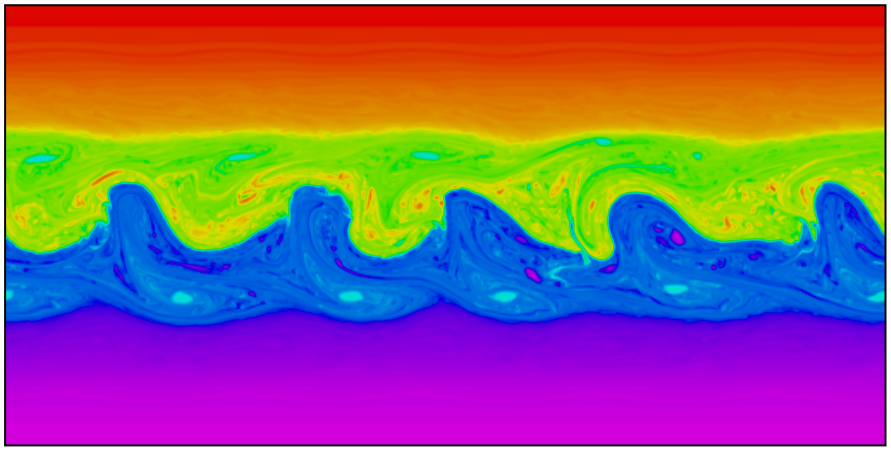
$t = 0$



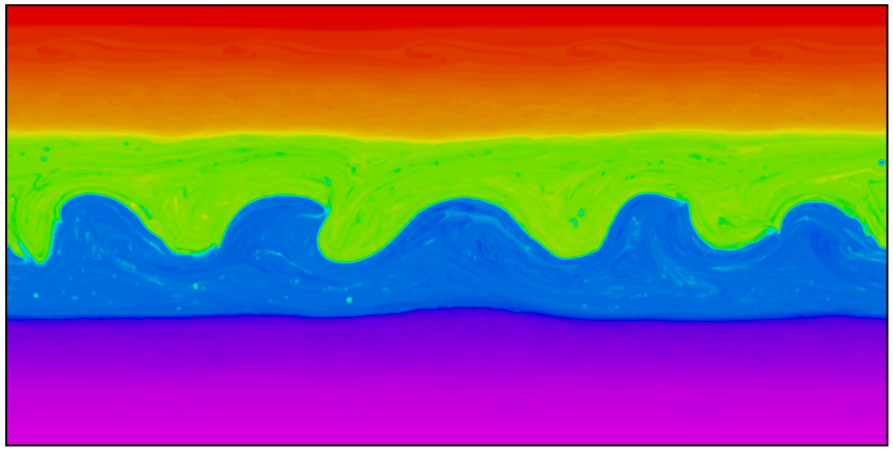
$t = 20$



$t = 50$



$t = 100$

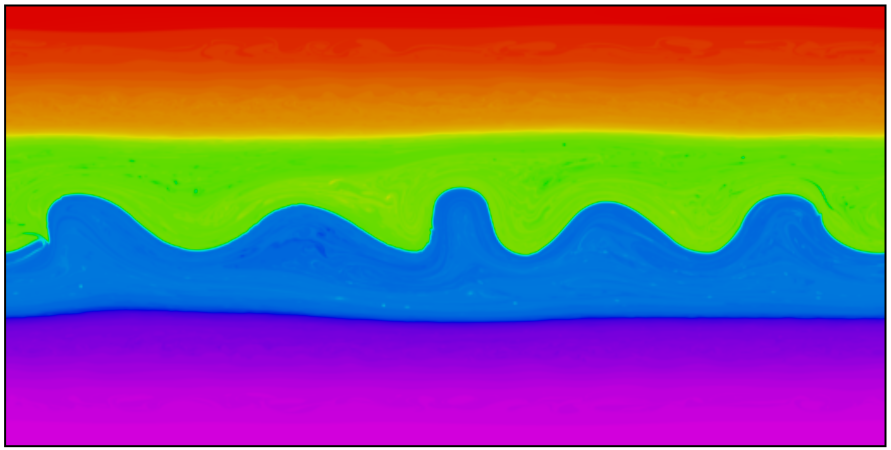


The whole sphere is shown, in a longitude-latitude (λ-ϕ) perspective.

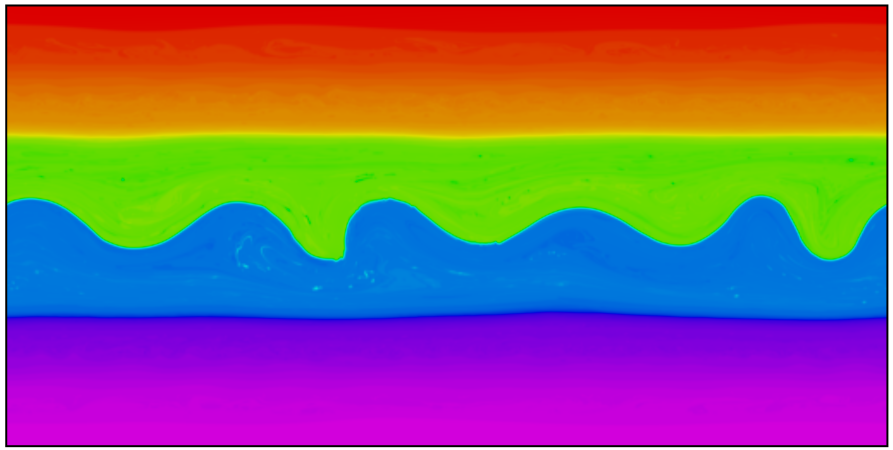
CLAM: application to spherical shallow-water flows

The flow starts from a state of rest, with $q = f = 2\Omega \sin\phi$.

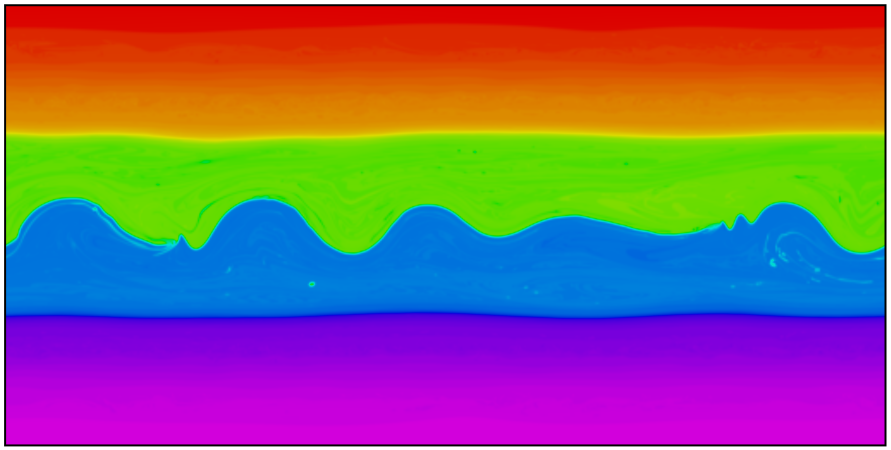
$t = 200$



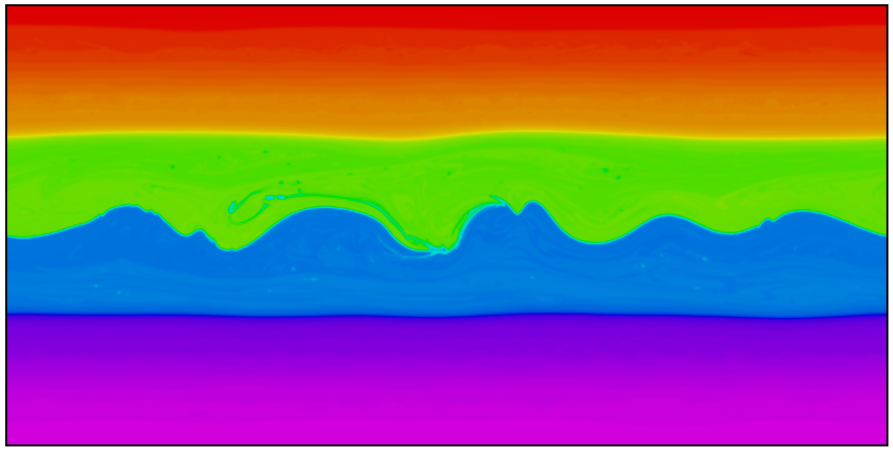
$t = 300$



$t = 400$



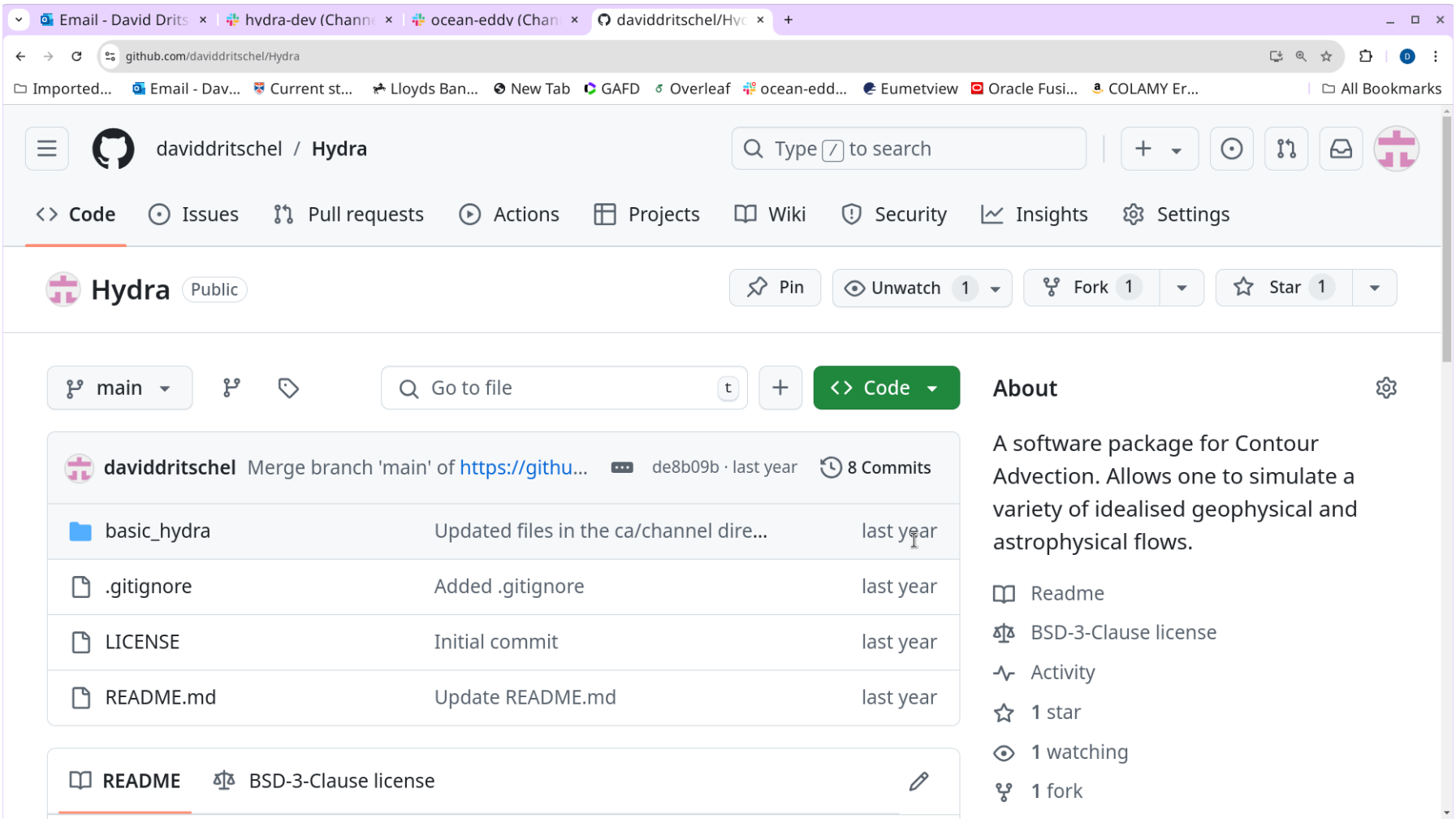
$t = 500$



Persistent, wavy jets develop from PV mixing, particularly near the equator

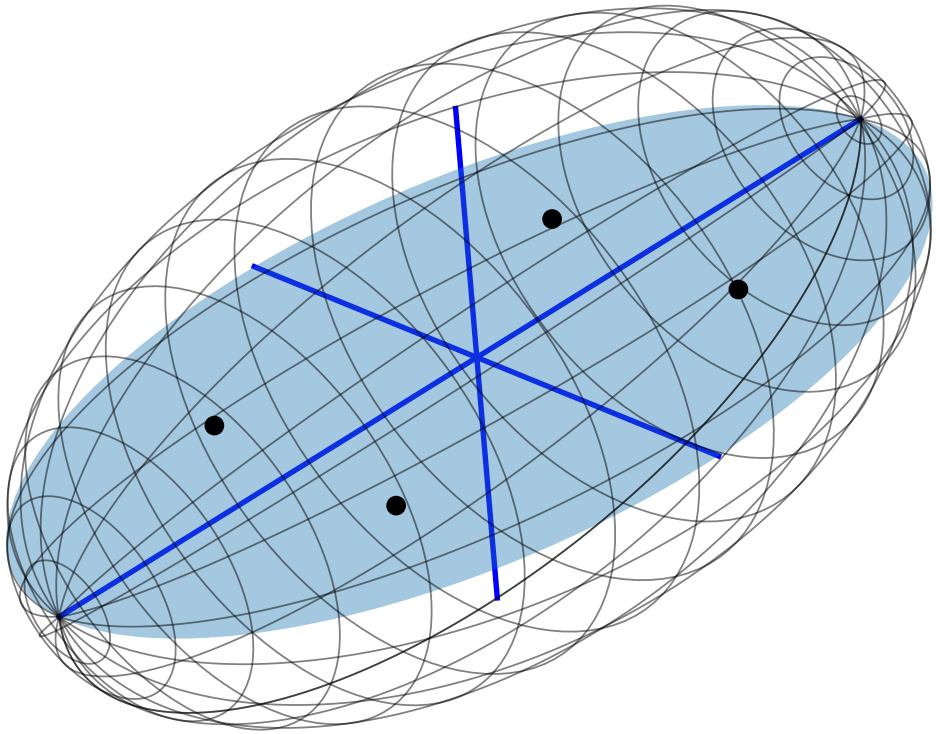
Hydra

A software package for Contour Advection. Allows one to simulate a variety of idealised geophysical and astrophysical flows.



EPIC: the Elliptical Parcel-In-Cell method

A *generalisation* of the Particle/Parcel-In-Cell (PIC) method uses space-filling, *deformable* elliptical parcels:



See Frey, Dritschel & Böing, *J. Comput. Phys. X* **17** (2023).

EPIC: parcel representation

Each ellipsoid, centred at $\boldsymbol{x} = \boldsymbol{x}_i$, moves as a material volume:

$$\frac{\mathrm{d}\boldsymbol{x}_i}{\mathrm{d}t} = \boldsymbol{u}(\boldsymbol{x}_i, t) \quad \text{and} \quad \frac{\mathrm{d}\mathcal{B}_i}{\mathrm{d}t} = \mathcal{S}(\boldsymbol{x}_i, t)\mathcal{B}_i + \mathcal{B}_i\mathcal{S}^T(\boldsymbol{x}_i, t)$$

where $\mathcal{S} = \nabla\boldsymbol{u}$ is the velocity gradient matrix, and $\mathcal{B}_i$ is a symmetric 3×3 matrix representing the i^{th} ellipsoid,

$$(\boldsymbol{x} - \boldsymbol{x}_i)^T \mathcal{B}_i^{-1} (\boldsymbol{x} - \boldsymbol{x}_i) = 1\,, \qquad \forall \boldsymbol{x} \text{ on the ellipsoid}$$

(Dritschel, Reinaud & McKiver, *J. Fluid Mech.* **555**, 2004.)

The eigenvalues of $\mathcal{B}_i$ are the squared axes lengths (a^2, b^2 and c^2), while the eigenvectors of $\mathcal{B}_i$ give the orientation of the principal axes.

EPIC: parcel attributes

- The parcels represent the flow *entirely*.
- Each parcel, i, can be thought of as a vessel/container/cell containing any number of flow attributes q_i (uniform across each parcel).

In applications to rotating convection, the attributes are buoyancy b and vector vorticity, $\boldsymbol{\omega}$. A relevant flow model is governed by the Oberbeck–Boussinesq equations:

$$\frac{\mathrm{D}\boldsymbol{u}}{\mathrm{D}t} + 2\boldsymbol{\Omega}\times\boldsymbol{u} = -\frac{\nabla p}{\rho} + b\hat{\boldsymbol{e}}_z\,, \qquad \frac{\mathrm{D}b}{\mathrm{D}t} = 0\,, \qquad \boldsymbol{\nabla}\cdot\boldsymbol{u} = 0\,.$$

Pressure p can be eliminated by taking the curl of the momentum equations:

$$\frac{\mathrm{D}\boldsymbol{\omega}}{\mathrm{D}t} = (\boldsymbol{\omega}+2\boldsymbol{\Omega})\cdot\nabla\boldsymbol{u} + \nabla b\times\boldsymbol{e}_z\,.$$

EPIC: interpolation and dynamics

- Parcel properties are interpolated tri-linearly (with volume weighting) to an underlying grid, where $\boldsymbol{u}$ is found by inverting the relation $\boldsymbol{\nabla} \times \boldsymbol{u} = \boldsymbol{\omega}$, and where both $\mathcal{S} = \nabla \boldsymbol{u}$ and the vorticity tendency $\boldsymbol{F}$ are computed.

- Conversely, $\boldsymbol{u}$, $\mathcal{S}$ and $\boldsymbol{F}$ are interpolated from the grid to the parcels to evolve the system forward:

$$\frac{\mathrm{d}\boldsymbol{x}_i}{\mathrm{d}t} = \boldsymbol{u}(\boldsymbol{x}_i, t)\,, \quad \frac{\mathrm{d}\mathcal{B}_i}{\mathrm{d}t} = \mathcal{S}(\boldsymbol{x}_i, t)\mathcal{B}_i + \mathcal{B}_i\mathcal{S}^T(\boldsymbol{x}_i, t)\,, \quad \frac{\mathrm{d}\boldsymbol{\omega}_i}{\mathrm{d}t} = \boldsymbol{F}(\boldsymbol{x}_i, t)\,.$$

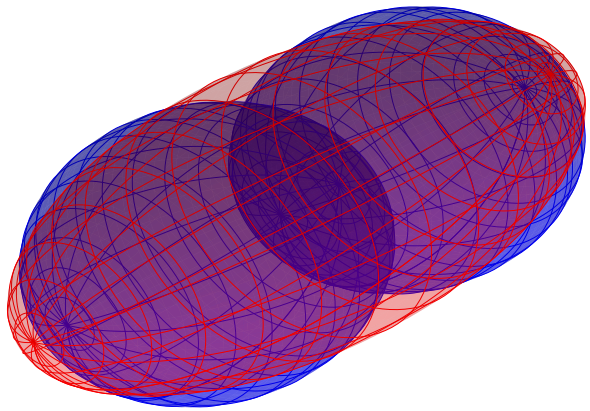
- We ensure interpolated parcel volumes V_i *closely match* grid-cell volumes $\Delta x \Delta y \Delta z$ by nudging parcel centres $\boldsymbol{x}_i$ each time step.

The parcels remain space-filling without ever needing to re-grid.
This maintains incompressibility.

EPIC: mixing I

- Mixing — essential in 3D turbulent flows — is achieved in two ways.

First, *excessively elongated parcels* are split into two identical parcels:

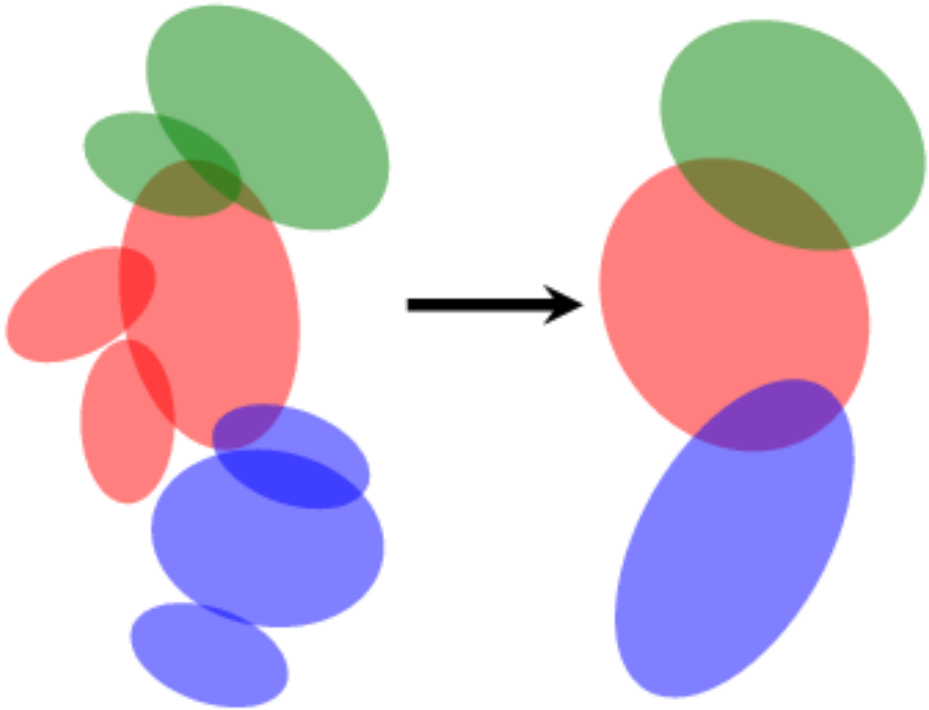


This conserves total volume, centroid and second-order spatial moments.

Splitting occurs when the major-minor axis ratio $a/c > \lambda_{\max} = 4$.

EPIC: mixing II

Second, *very small parcels* (having $V_i < \Delta x \Delta y \Delta z/20$) are merged into their closest neighbour:



This conserves total volume, centroid and, *approximately*, second-order spatial moments.

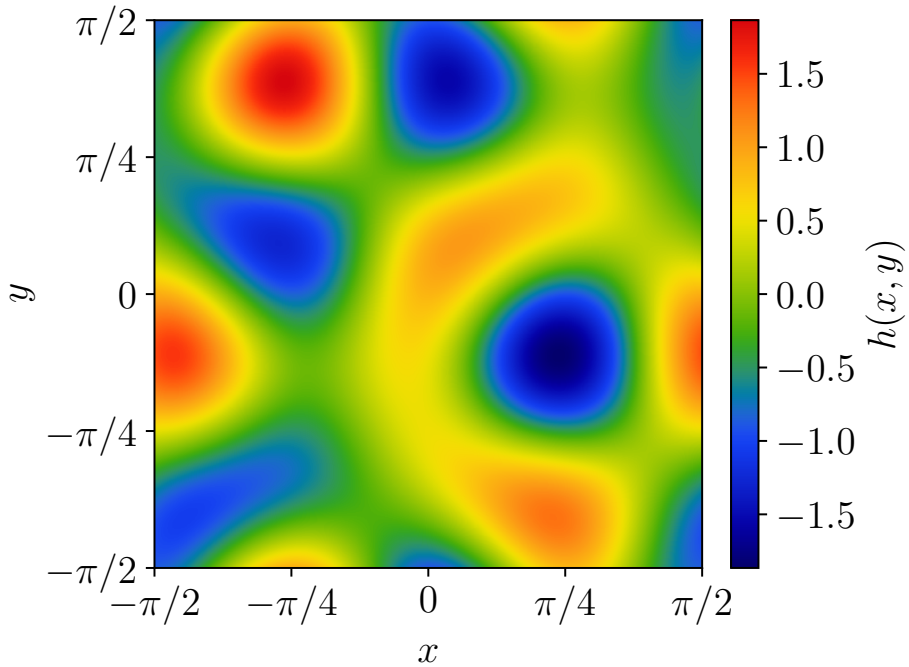
EPIC: application to the Rayleigh-Taylor instability

Consider a flow in the box $[-\pi/2, \pi/2]^3$, horizontally-periodic and bounded vertically between free-slip boundaries — appropriate to inviscid flow.

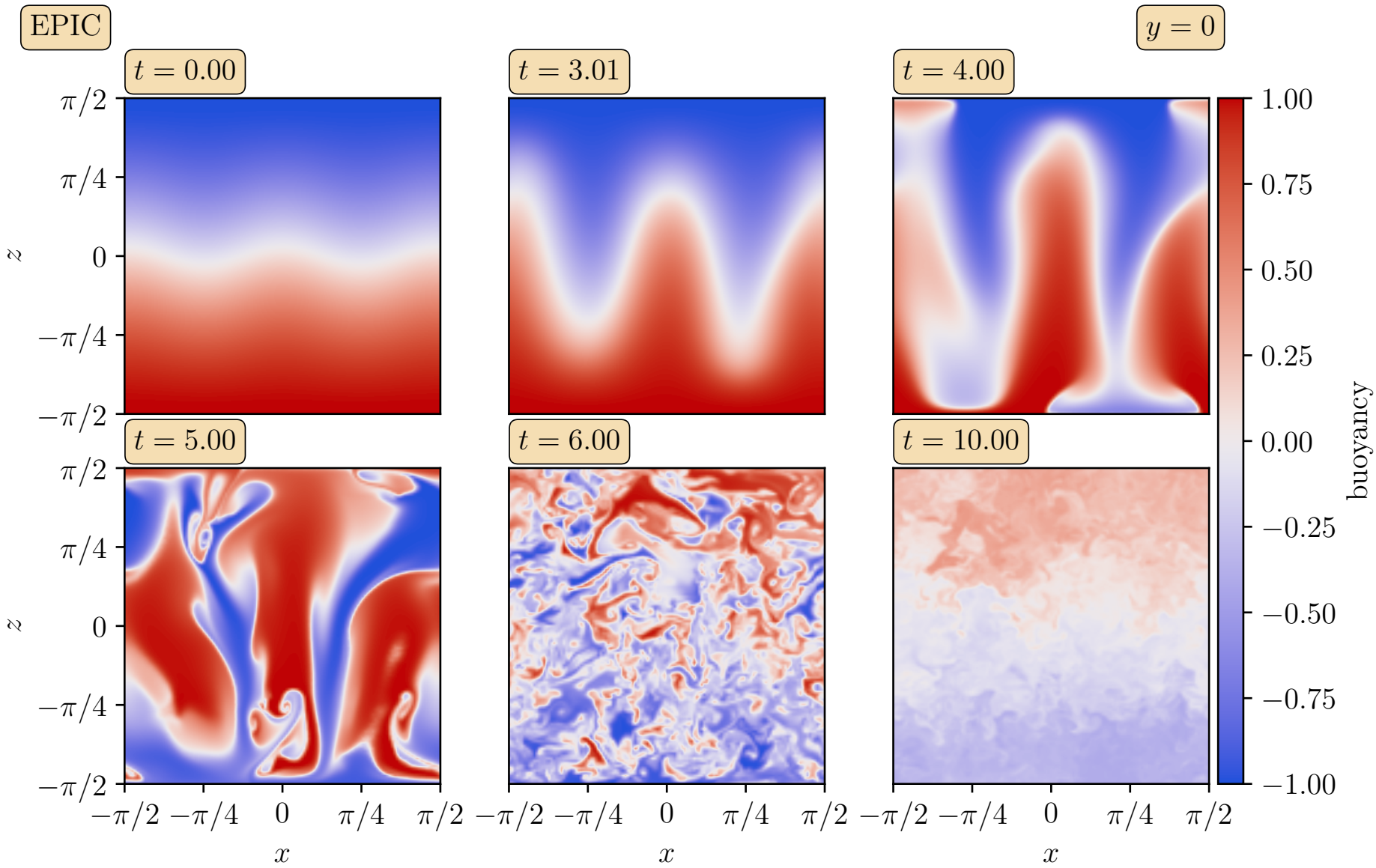
Initially, we start with $\boldsymbol{u} = \boldsymbol{0}$ and an unstable buoyancy distribution:

$$b = -\sin z + \epsilon h(x, y) \cos^2 z$$

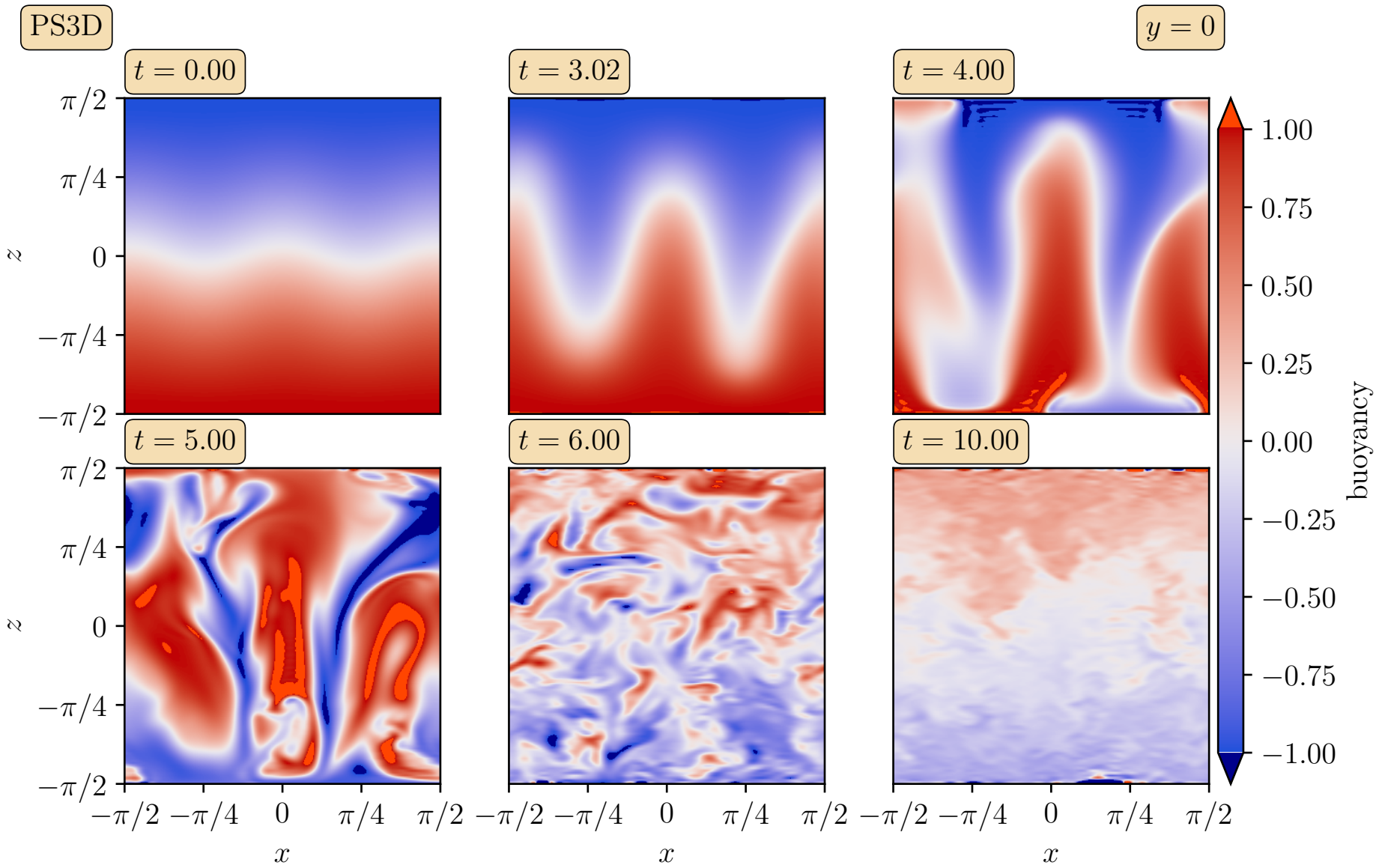
where $\epsilon = 0.1$ and $h(x, y)$ is shown below. Note, $\boldsymbol{\Omega} = \frac{1}{2}\boldsymbol{e}_z$.



EPIC: buoyancy evolution (at $y = 0$) — 128^3 grid

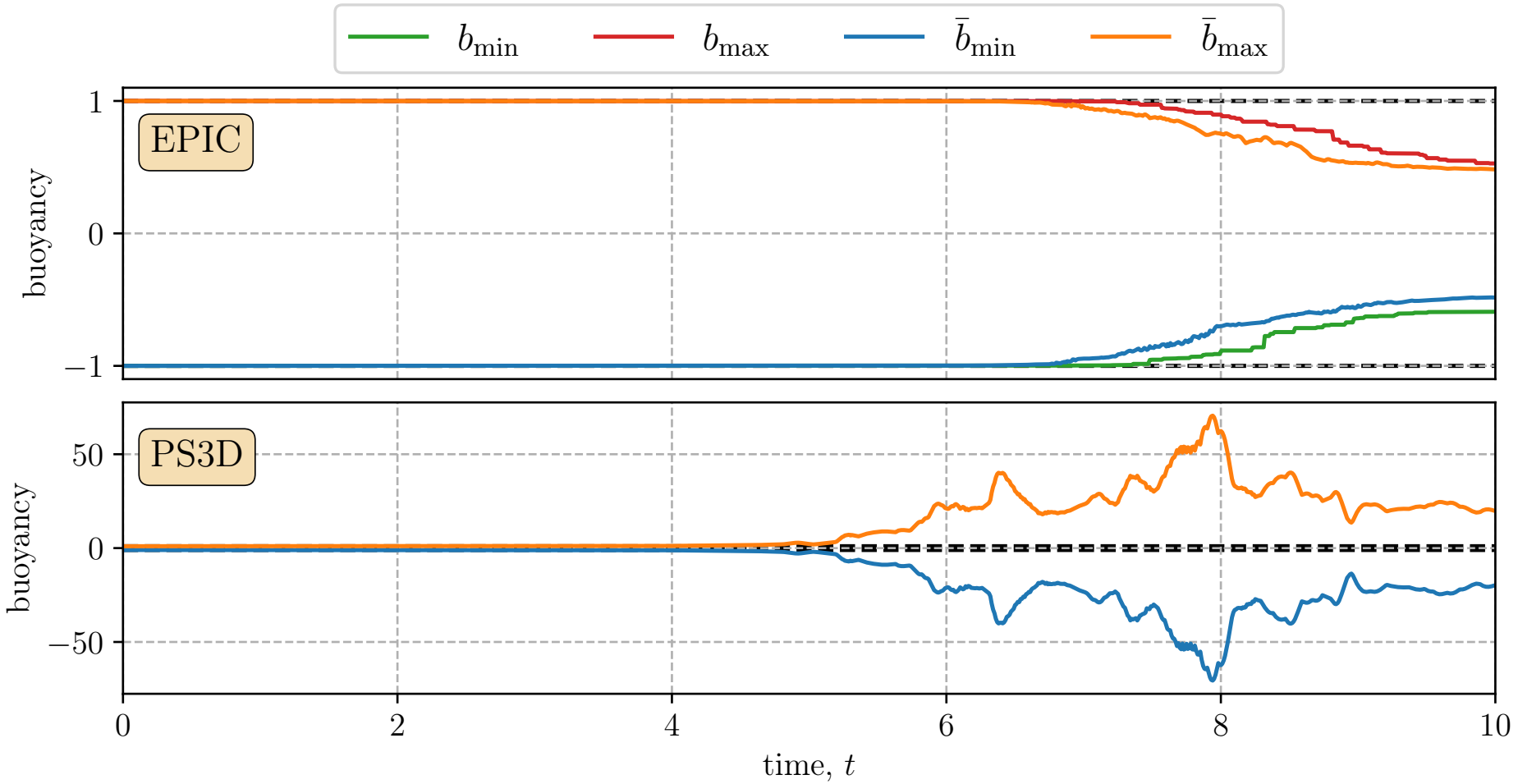


Pseudo-Spectral method (PS3D) with ∇^6 hyperviscosity



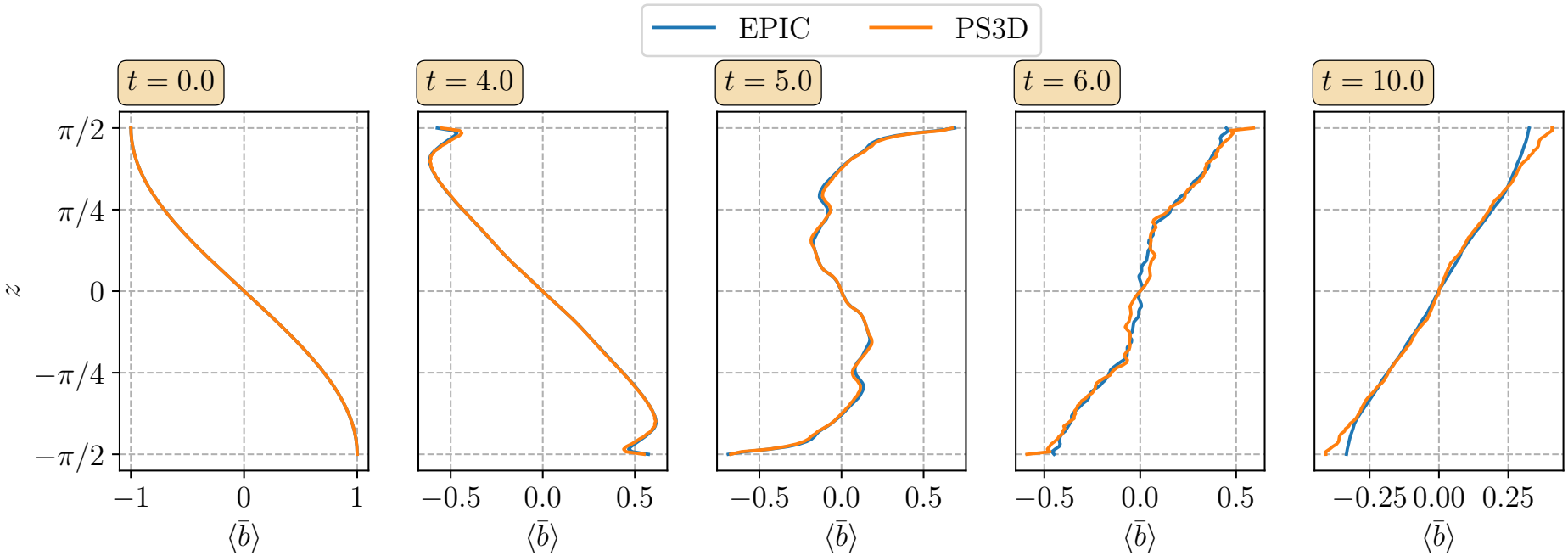
Buoyancy extrema: preservation of monotonicity?

✓ EPIC ensures $b_{\max}(t) \leq b_{\max}(0)$ and $b_{\min}(t) \geq b_{\min}(0)$: monotonicity.



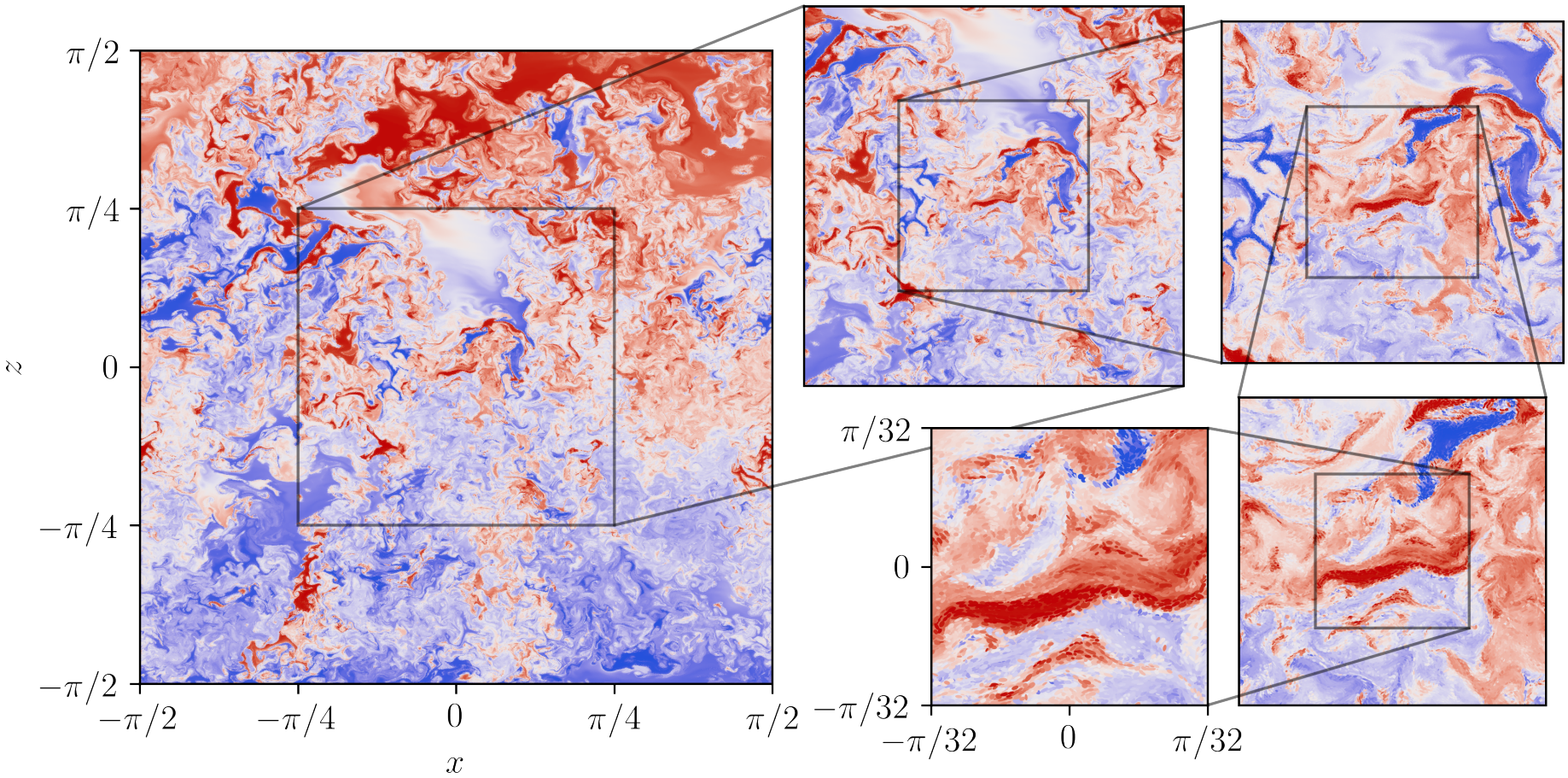
✗ PS3D **strongly violates monotonicity.**

Horizontally-averaged buoyancy: turbulent mixing



EPIC mixes more realistically, especially near boundaries, where PS3D exhibits **strong** overshoots and undershoots.

EPIC with a 384^3 underlying grid resolution



Summary

- A wide range of numerical approaches exists for simulating fluid flow.
- The appropriate (efficient and accurate) method depends strongly on the system under study ... and the question being asked.
- We have reviewed methods for the simplest of models, point vortices, to relatively complex ones, such as the quasi-geostrophic model, 2D density-stratified flows, 2D magneto-hydrodynamics and spherical shallow-water flows.
- Many questions remain, and much still remains to be learned. Having the right tool to do the job is important.
- To make progress, we rely upon reduced models and we continually need to create new ones.
- Finally, we should never stop searching for improvements. Modelling is a *dynamic* process.