Journal of Computational Physics: X 14 (2022) 100109

Contents lists available at ScienceDirect

Journal of Computational Physics: X

www.elsevier.com/locate/jcpx

EPIC: The Elliptical Parcel-In-Cell method

Matthias Frey [a,*], David Dritschel [a,*], Steven Böing [b]

[a] Mathematical Institute, University of St Andrews, KY16 9SS, UK
[b] University of Leeds/Met Office Strategic Research Group, School of Earth and Environment, University of Leeds, LS2 9JT, UK

ARTICLE INFO

Article history:
Received 5 November 2021
Received in revised form 21 May 2022
Accepted 3 June 2022
Available online 6 June 2022

Keywords:
PIC
Elliptical parcels
Lagrangian methods
Density stratified flows
Turbulent flows

ABSTRACT

We present a novel approach to simulating general two-dimensional flows, which could also be applied to other areas of continuum mechanics. The approach generalises the Particle-In-Cell (PIC) method, originally used to model two-dimensional hydrodynamics, by representing fluid elements by elliptical parcels. The rotation and deformation of these parcels are calculated, and parcels split beyond a critical aspect ratio. Conversely, small parcels are eliminated by merging them with larger ones. The elliptical parcels well represent the flow deformation and have excellent conservation properties. In contrast to earlier work that combined PIC with elliptical parcels that split and merge, a vorticity-based framework is used, and accurate integration over ellipses is performed efficiently by two-point Gaussian quadrature. The small-scale mixing associated with parcel splitting and merging is shown to be strongly convergent with grid resolution. The robustness, versatility, accuracy and efficiency of the new Elliptical Parcel-In-Cell (EPIC) method is demonstrated for a variety of standard test cases, and compared with a standard pseudo-spectral method. The results indicate that EPIC is a promising, Lagrangian-based alternative to grid-based methods.

1. Introduction

The numerical representation of fluid flows and other continuum mechanical systems by Lagrangian particles or parcels (the latter objects having a finite volume) communicating with an underlying grid has a long history dating back to [1,2] (for a historical review please see [3–5]). There have been diverse applications, ranging from two-dimensional vortex interactions, to galaxy dynamics and even to gaming. There have also been diverse approaches, such as in the way the particles/parcels communicate with the underlying grid (e.g. interpolation). Yet, despite this, fully grid-based methods are overwhelmingly preferred by the scientific community, especially in fluid dynamics (hydrodynamics) and magneto-hydrodynamics. For example, some of the most intensive uses of computer resources are in weather forecasting and in climate modelling, and those models almost without exception use a fixed computational grid. Largely this is driven by developments in parallelism and the need to parametrise processes such as unresolved convection, radiation and gravity-wave drag, which are often designed around grid-based approaches. Having a fixed number of computational elements allows certain efficiencies and appears simplest.

Fluid flows, in particular the ultra-high Reynolds number turbulent fluid flows found in atmosphere/ocean dynamics, exhibit energetic fine-scale structures like fronts, storms, etc which play an important role in the evolution of the system. These are not well captured using a fixed computational grid, and invariably significant errors accrue when under-resolving

* Corresponding authors.
E-mail addresses: mf248@st-andrews.ac.uk (M. Frey), david.dritschel@st-andrews.ac.uk (D. Dritschel).

https://doi.org/10.1016/j.jcpx.2022.100109

(or not resolving) such structures. While adaptive mesh refinement (AMR, e.g. [6–9]) helps to provide additional resolution in the most turbulent regions of a flow field and can go part way in following complex structures if a moving mesh is used, this method can be challenging to implement in an efficient way. Arguably, a more natural approach is to use Lagrangian computational elements (particles/parcels). They offer great flexibility and may be relatively easily focused in regions requiring greater resolution. They also offer great adaptability, allowing one to increase or decrease the number of parcels in response to changing conditions.

Other advantages include a fully explicit representation of turbulent mixing (especially in the Elliptical Parcel-In-Cell method we introduce below, this is in contrast to Eulerian methods where there is numerical diffusion in all prognostic fields), and an ideal framework for the Lagrangian advection of tracers (both active and passive). Each parcel can carry an arbitrary list of attributes besides its position: shape, volume, mass, vorticity, as well as thermodynamical, chemical and even biological attributes. A separate partial differential equation for each attribute is not required as in fully grid-based methods. Any materially-conserved quantity remains exactly conserved (e.g. a passive tracer, and density or mass in an incompressible fluid), at least when parcel splitting/merging is done consistently (see below and [4]). This means that conservative tracer dynamics can be computed almost for free; only a space in memory is required. Other attributes (e.g. vorticity) are updated by ordinary differential equations (ODEs). Parcels also sample subgrid scales, and thereby offer an alternative to grid-based subgrid closure schemes.

Parcel-based approaches, however, do have potential shortcomings. Besides the greater challenges of parallelisation mentioned above, two serious issues are (1) the difficulty in maintaining an adequate number of parcels in each grid cell, and (2) an underestimation of turbulent mixing. The first issue can also lead to problems in representing incompressible flows, where one would ideally like to ensure that the gridded volume (or area in two dimensions) — obtained by interpolation from the parcels — remains constant. The Lagrangian advection of a finite set of parcels does not automatically preserve gridded volume. The second issue regarding mixing is discussed in more detail in [10]. Although this previous study included a description of parcel stretching and mixing at small scales, convergence analysis suggested that too little mixing occurred at small scales. This is important for example in properly modelling (in a statistical sense) thermodynamical processes and chemical reactions on small scales. A lack of mixing may be the result of either the formulation of parcel stretching, or due to the lack of explicit subgrid scale parcel-parcel interactions [e.g. 11,12] or processes at scales smaller than individual parcels.

The current paper makes major progress on both of these issues. We develop an approach which ensures adequate parcel representation in each grid cell and the preservation of the parcel-interpolated gridded volume, *without* the need to occasionally reset the parcel distribution. We also show that deformable elliptical parcels significantly improve the representation of subgrid scale turbulent mixing.

The article is structured as follows. In the next section, we detail the key elements of the EPIC method after first reviewing the basic aspects of the Particle/Parcel-In-Cell (PIC) method. In Section 3, the structure of the numerical algorithm is presented, including a flowchart. In Section 4, we examine the performance of the EPIC method in three diverse test cases. The results allow us to determine numerical parameter values by balancing cost and accuracy. The objective is to produce the most efficient method for a chosen underlying grid resolution, and to provide a simple user interface wherein only initial data on the grid is required, as in conventional fully grid-based approaches. Our conclusions are offered in Section 5. There we note that the present two-dimensional method is easily extended to three dimensions, with all of the same advantages. We will report on this extension in a subsequent paper.

2. Methodology

2.1. Parcel-In-Cell (PIC): a brief overview

The PIC method was developed to accurately treat advection by following Lagrangian parcels (or particles) directly, while making use of an underlying grid structure to accelerate the calculation of interactions between parcels (the parcels communicate via the grid). For a large number of parcels, the direct calculation of all such interactions is computationally expensive. Instead, the PIC method interpolates parcel properties to the underlying grid where well-established efficient methods exist e.g. for inverting Poisson's equation and for calculating spatial derivatives. Quantities computed on the grid are then interpolated to the parcel positions, e.g. the velocity field required to move each parcel. Then, simple ODEs are used to update parcel positions and possibly other attributes.

The version of the PIC method developed by [2], for example, followed clouds of point vortices to study the behaviour of an unstable jet (two side-by-side bands of approximately uniform vorticity). Point vortices are particles of infinite vorticity, zero area, but finite circulation (the product of vorticity and area). Their direct interaction can be computed in theory, but this is much more expensive than first building a gridded vorticity field from the point vortices, inverting Poisson's equation and computing the velocity field on a regular grid, and finally interpolating this velocity field to each point position. Each particle requires two ODEs to update its x and y coordinates. The approach is simple and efficient, and when it was first developed, it led to a breakthrough in the understanding of nonlinear, inviscid two-dimensional flows.

The PIC method approximates the particle-particle or parcel-parcel interaction to gain efficiency (for a large number of parcels). The error depends on the resolution of the underlying grid and the number of parcels used (as well as details of

the interpolation, etc). This error is unlike that made by fully grid-based methods: it is not diffusive (no numerical diffusion is required for stability).

Parcels, unlike particles, have finite area (or volume in three dimensions). Parcels are usually overlapped to represent properties continuously on a grid. Using a small number of parcels per grid cell will invariably lead to errors in this representation, but computational cost usually dictates the grid resolution and the approximate number of parcels per cell.

The PIC method has been extended in many ways and we will not attempt to review the entire literature on this subject (some discussion can be found in [3–5,10]). Here we instead focus on a relatively simple form of PIC recently developed to model three-dimensional convective clouds in the atmosphere [4], and shown to provide some advantages over fully grid-based Large-Eddy-Simulations (LES) [10]. The PIC method in [4] represents an incompressible density-stratified three-dimensional flow containing water vapour and liquid water by a space-filling distribution of spherical parcels of variable size. Each parcel carries a series of attributes (volume, buoyancy, specific humidity and vorticity) some of which can change in time either by condensation/evaporation or natural advection (e.g. stretching of vorticity). Gridded properties are obtained from the parcels using simple tri-linear interpolation, and these are used to compute the velocity field and the vorticity tendency on the grid. These gridded quantities are then interpolated back to the parcels, again using (reverse) tri-linear interpolation. Then ODEs are solved to move the parcels forward and to update their vorticity. To model the natural scale cascade in a turbulent three-dimensional flow, an estimate of parcel stretch is used to approximate each parcel's total deformation; however, the parcels are kept spherical at all times. When this stretch exceeds a threshold, the parcel is split into two equal parts along the direction of the local vorticity vector. This splitting will create progressively smaller parcels, and a growing population of them. To limit this process, small parcels below a threshold volume ($1/6^3$ of the grid cell volume) are eliminated. Their properties are given back to other parcels in their vicinity by interpolation to and from the grid. In this way, conservation of total volume, mass, etc is preserved.

The present paper describes a further extension of the PIC method which allows for deformable, elliptical parcels. Here we develop the two-dimensional algorithm for a dry density-stratified flow and defer the three-dimensional algorithm to a subsequent paper. A similar extension (the Deformable Particle-In-Cell method, which was the first method to combine a PIC method with elliptical parcels that split and merge) was recently developed in [5], who showed that elliptical parcels significantly improve the representation of fluid flows in various test cases examined. In particular, elliptical parcels much better preserve the gridded areas obtained from parcel interpolation in an incompressible flow compared to non-deformable (circular) parcels. Elliptical and ellipsoidal parcels have also been used in other approaches, such as Smoothed Particle Hydrodynamics (a meshfree approach) and the moving particle semi-implicit (MPS) method [e.g. 13–15].

The EPIC method developed here could be considered to fall into a wider class of Deformable Parcel-In-Cell methods: we use the name EPIC as it makes it explicit that ellipses are used, and makes it easier to refer to the entire framework presented here. The main difference between our work and [5] is in the way parcel attributes are transferred to the grid, and *vice-versa*. Here we use a much simpler approach involving two support points placed between the foci of the ellipse, rather than a much more expensive numerical area integration. These two points correspond to the Gaussian quadrature points for an ellipse, as shown in [16], and there is a simple generalisation to 3D ellipsoids requiring four support points [17]. Another difference is that our parcels are space filling (sum to the domain area), whereas they are not in [5] (they are largely kept from overlapping). In contrast to DPIC, the parcels in the EPIC framework carry vorticity, which evolves only due to horizontal buoyancy gradients. We document the procedure for obtaining the velocity field from vorticity below. Furthermore, the parcel splitting and mixing procedures differ. During splitting, we additionally preserve the second-order spatial moments (besides volume and centroid). We identify mergers using the distance between centres and the parcel areas, whereas [5] does this by locating two or more parcels in a grid box on a finer grid. The construction of the resulting merged parcel also differs as illustrated in Fig. 1 (for details, see below). Our criterion aims to preserve the second-order moments as well as possible. We also preserve parcel gridded area with much smaller errors by applying an area-based correction to the parcel positions, avoiding any need for parcel re-gridding. Finally, there are differences in interpolation: here we use bi-linear interpolation whereas [5] uses a step function (nearest grid cell).

2.2. The flow model

We here develop the EPIC method for simulating two-dimensional (2D) inviscid, incompressible density-stratified flows. However, we stress that the method is much more widely applicable, even beyond fluid dynamics (see discussion). The flow model here is still sufficiently complex to stringently test the new method and relates to a problem of significant importance in environmental modelling. Moreover, it is readily extended in multiple ways, e.g. to include moisture, a magnetic field, etc.

We use x for the horizontal coordinate and y for the vertical one (gravity points *downwards* in y). We assume the density ρ varies only slightly from a constant background value ρ_0. Then, making the Boussinesq approximation, we arrive at the following system of equations for the scalar vorticity $\zeta = v_x - u_y$ (spatial subscripts indicate partial differentiation while $\boldsymbol{u} = (u, v)$ is the velocity field) and buoyancy $b = -g(\rho - \rho_0)/\rho_0$:

$$\frac{\mathrm{D}\zeta}{\mathrm{D}t} = b_x \tag{1}$$

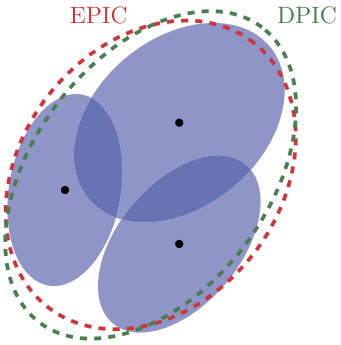

Fig. 1. Merger of three elliptical parcels in DPIC [5] and EPIC. While the centroid is located at the same position, the rotation angle, semi-major and semi-minor axes are not identical.

$$\frac{\mathrm{D}b}{\mathrm{D}t} = 0\,, \tag{2}$$

where g is the acceleration due to gravity and $\mathrm{D}/\mathrm{D}t$ denotes the material or Lagrangian derivative (see e.g. [18]). Since the flow is incompressible ($u_x + v_y = 0$), we can introduce a streamfunction ψ in terms of which $u = -\psi_y$ and $v = \psi_x$. Then the definition of vorticity leads to a Poisson equation for ψ:

$$\nabla^2\psi = \zeta\,. \tag{3}$$

To solve this, we need to consider boundary conditions. Here we assume the domain is periodic in x and is confined between impermeable boundaries at $y = y_{\min}$ and $y = y_{\max}$. As the flow is inviscid, only the normal component of the velocity must be zero on the y boundaries, i.e. $v = 0$ there. But since $v = \psi_x$, then ψ must be constant on each y boundary.

Due to the periodic x boundaries, and the fact that the governing equations Eq. (1) and Eq. (2) have no explicit x dependence, the solution of Eq. (3) for ψ is most easily accomplished in spectral space after a Fast Fourier Transform (FFT) in x. However, this does not generally provide the entire velocity field since ζ may have a component which is independent of x, say $\hat{\zeta}_0(y)$ in spectral space. But using again the definition of vorticity, we have $\mathrm{d}\hat{u}_0/\mathrm{d}y = -\hat{\zeta}_0$. Integration in y then provides the x-independent part of u, apart from a constant. The latter is determined by fixing the global average value of u, which is constant by momentum conservation. Details can be found in Appendix A.

2.3. Elliptical parcels: basic representation and dynamics

Given an Eulerian velocity field $\boldsymbol{u}(\boldsymbol{x}, t)$, a Lagrangian fluid parcel i at position $\boldsymbol{x}_i(t)$ evolves in time t according to

$$\frac{\mathrm{d}\boldsymbol{x}_i}{\mathrm{d}t} = \boldsymbol{u}_i(t) \equiv \boldsymbol{u}(\boldsymbol{x}_i, t) \tag{4}$$

where $\boldsymbol{u}_i(t)$ denotes the parcel velocity interpolated from the grid. We represent parcels in coordinates relative to their centres by the ellipse equation [17]

$$\boldsymbol{x}^{\mathsf{T}}(t)\boldsymbol{B}^{-1}(t)\boldsymbol{x}(t) = 1 \tag{5}$$

where $^{\mathsf{T}}$ denotes a transpose, and $\boldsymbol{B}$ is a symmetric positive-definite matrix with constant determinant $B_{11}(t)B_{22}(t) - B_{12}^2(t) = a^2b^2$ for an incompressible flow. Here a and b are the semi-major and semi-minor axis lengths, and their product multiplied by π gives the ellipse area (here subscripts i are suppressed for simplicity).

Unlike point parcels, elliptical parcels can deform due to local gradients in the velocity field

$$\boldsymbol{S} = \nabla\boldsymbol{u} = \begin{pmatrix} u_x & u_y \\ v_x & v_y \end{pmatrix} \tag{6}$$

as described in [19]. Assuming a locally linear background flow

$$\boldsymbol{u}(\boldsymbol{x}, t) = \boldsymbol{S}(t)\boldsymbol{x}(t)\,, \tag{7}$$

in a frame of reference moving with the parcel, the time derivative of Eq. (5) yields

$$\frac{\mathrm{d}\boldsymbol{B}}{\mathrm{d}t} = \boldsymbol{B}\boldsymbol{S}^{\mathsf{T}} + \boldsymbol{S}\boldsymbol{B} \tag{8}$$

after making use of Eq. (4) and Eq. (7) – see [17,19] or Appendix B. Although we only apply this to two-dimensional (2D) flows in this paper, Eq. (8) is also valid in 3D, and indeed in any dimension. For a 2D incompressible flow, i.e. $\nabla\cdot\boldsymbol{u} = u_x + v_y = 0$, Eq. (8) simplifies to

$$\frac{\mathrm{d}B_{11}}{\mathrm{d}t} = 2\left(u_x B_{11} + u_y B_{12}\right) \tag{9}$$

$$\frac{\mathrm{d}B_{12}}{\mathrm{d}t} = v_x B_{11} + u_y B_{22}\,, \tag{10}$$

where we have exploited the fact that the determinant of $\boldsymbol{B}$ is preserved in time. This property further allows us to store only two matrix components and compute B_{22} when needed. This makes EPIC as memory efficient as possible.

In [5], the elliptical parcel deformation was computed instead with the simpler equation $\mathrm{d}\boldsymbol{M}/\mathrm{d}t = \boldsymbol{S}\boldsymbol{M}$ where $\boldsymbol{M}$ is related to $\boldsymbol{B}$ above by $\boldsymbol{B} = \boldsymbol{M}\boldsymbol{M}^\mathsf{T}$. This leads to our equation for $\boldsymbol{B}$ in Eq. (8). Nevertheless, here we evolve $\boldsymbol{B}$ since this is needed to find the ellipse semi-axis lengths a and b from the eigenvalues of $\boldsymbol{M}\boldsymbol{M}^\mathsf{T} = \boldsymbol{B}$ (see below and section 4.3 in [5]). Furthermore $\boldsymbol{B}$ is more useful than $\boldsymbol{M}$ for parcel splitting and merging.

2.4. Interpolation

As mentioned in the introduction, we only use two support points to interpolate from the parcels to the grid and vice-versa. These points lie between the foci of the ellipse at $\boldsymbol{x}^\pm = \boldsymbol{x} \pm \frac{1}{2}c\hat{\boldsymbol{a}}$ (these are the Gaussian points, see [16]), with semi-focal length

$$c = \sqrt{a^2 - b^2} = \sqrt{2a^2 - B_{11} - B_{22}} \tag{11}$$

and principle eigenvector (lying along the major axis)

$$\hat{\boldsymbol{a}} = \frac{1}{\sqrt{(a^2 - B_{22})^2 + B_{12}^2}} \begin{pmatrix} a^2 - B_{22} \\ B_{12} \end{pmatrix} \tag{12}$$

where

$$a^2 = \frac{1}{2}(B_{11} + B_{22}) + \sqrt{\frac{1}{4}(B_{11} - B_{22})^2 + B_{12}^2} \tag{13}$$

is the largest eigenvalue of $\boldsymbol{B}$.

Following the notation of [4], the value at a grid point $\bar{\boldsymbol{x}} = (\bar{x}, \bar{y})$ of some gridded field $\bar{q}(\bar{\boldsymbol{x}}, t)$ is found by interpolating the attribute q_i of all parcel support points $\boldsymbol{x}_i^\pm$ lying within the four surrounding grid cells, $\mathcal{P}(\bar{\boldsymbol{x}})$. As we are using two-point Gaussian quadrature, there is an additional weight of $1/2$ to apply.

$$\bar{q}(\bar{\boldsymbol{x}}, t) = \frac{1}{2\bar{V}} \left(\sum_{\boldsymbol{x}_i^- \in \mathcal{P}(\bar{\boldsymbol{x}})} \phi(\boldsymbol{x}_i^- - \bar{\boldsymbol{x}}) q_i V_i + \sum_{\boldsymbol{x}_i^+ \in \mathcal{P}(\bar{\boldsymbol{x}})} \phi(\boldsymbol{x}_i^+ - \bar{\boldsymbol{x}}) q_i V_i \right) \tag{14}$$

where analogously

$$\bar{V}(\bar{\boldsymbol{x}}, t) = \frac{1}{2} \left(\sum_{\boldsymbol{x}_i^- \in \mathcal{P}(\bar{\boldsymbol{x}})} \phi(\boldsymbol{x}_i^- - \bar{\boldsymbol{x}}) V_i + \sum_{\boldsymbol{x}_i^+ \in \mathcal{P}(\bar{\boldsymbol{x}})} \phi(\boldsymbol{x}_i^+ - \bar{\boldsymbol{x}}) V_i \right) \tag{15}$$

is the gridded area and

$$\phi(\boldsymbol{x}_i^\pm - \bar{\boldsymbol{x}}) = \left(1 - \frac{|x_i^\pm - \bar{x}|}{\Delta x}\right) \left(1 - \frac{|y_i^\pm - \bar{y}|}{\Delta y}\right) \tag{16}$$

are the bi-linear interpolation weights.

Similarly, the interpolation from the grid to the parcels is given by

$$q_i(t) \equiv q(\boldsymbol{x}_i, t) = \frac{1}{2} \left(\sum_{\bar{\boldsymbol{x}} \in \mathcal{G}_i^-} \phi(\boldsymbol{x}_i^- - \bar{\boldsymbol{x}}) \bar{q}(\bar{\boldsymbol{x}}, t) + \sum_{\bar{\boldsymbol{x}} \in \mathcal{G}_i^+} \phi(\boldsymbol{x}_i^+ - \bar{\boldsymbol{x}}) \bar{q}(\bar{\boldsymbol{x}}, t) \right) \tag{17}$$

where $\mathcal{G}_i^\pm$ stands for the four grid points of the cell in which the support point $\boldsymbol{x}_i^\pm$ is located. The two support points do not necessarily lie in the same cell. The sums in Eq. (14) to Eq. (17) are efficiently computed by summing over all parcel support points.

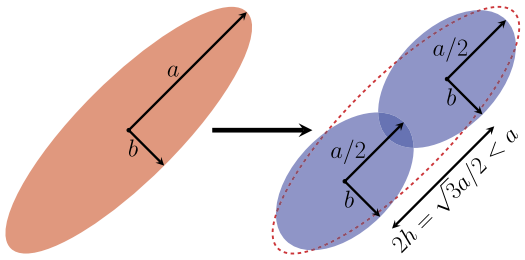

Fig. 2. An elliptical parcel splitting into two parcels of half the area and equal shape, with the same total centroid and total second order moments as the original parcel.

2.5. Parcel splitting

When the elliptical parcels evolve in an incompressible velocity field with non-zero strain, they deform according to Eq. (8) while preserving their area. This can result in extremely elongated parcels with aspect ratios $\lambda = a/b \gg 1$. By contrast, a material area would also bend as it stretches: the elliptical deformation only captures the local strain (or average strain over the area). As the elliptical parcels cannot bend, we instead split them when $\lambda > \lambda_{\max}$ (a user-defined threshold > 2) or when their area is bigger than a maximum prescribed parcel area $V_{\max} = \tilde{V}_{\max} V_{\text{cell}}$ ($\tilde{V}_{\max}$ is also user-defined; see the recommended parameter settings below). Although the individual parcel area is not preserved when splitting or merging occurs, the total area (i.e. the sum over all parcels) is conserved at all times. Parcel merging, as discussed in Section 2.6, reduces the number of parcels per grid box, while the maximum area criterion ensures that we have at least a certain minimum number of parcels per grid cell.

A parcel is split into two equal-shaped ellipses ('child parcels') having half the area of the 'parent' parcel, as illustrated in Fig. 2. Their centres are separated by a distance $h = \sqrt{3}a/4$ from the centre of the parent parcel $\boldsymbol{x}$, and along the direction of the eigenvector $\hat{\boldsymbol{a}}$ corresponding to the largest eigenvalue, i.e. the new centres are $\boldsymbol{x} \pm h\hat{\boldsymbol{a}}$. This construction guarantees that the total area, centroid and second-order spatial moments are conserved. This also results in a simple equation for the shape matrices of the child parcels:

$$\boldsymbol{B}_1 = \boldsymbol{B}_2 = \boldsymbol{B} - \frac{3}{4} a^2 \hat{\boldsymbol{a}}\hat{\boldsymbol{a}}^{\mathsf{T}}. \tag{18}$$

Other parcel properties such as vorticity and buoyancy remain unchanged. A complete derivation of Eq. (18) is given in Appendix C. Note that the parcel splitting used by [5] does not conserve the second-order moments; instead the parcels are prevented from overlapping. A small overlap is required to conserve these moments.

2.6. Parcel merging

Splitting leads to an ever-increasing number of small parcels. In order to limit computational cost, and to model the eventual mixing that takes place due to this scale cascade, we remove parcels of area less than a prescribed minimum area $V_{\min}$ by merging them with neighbouring ones. In practice, $V_{\min}$ is chosen to be a prescribed fraction $\tilde{V}_{\min}$ of the cell area $V_{\text{cell}} = \Delta x \Delta y$. A parcel, indexed say by i, is first marked ready to be merged if its area is smaller than $V_{\min}$ (i.e. $V_i < V_{\min}$). Then nearby grid cells are searched for the closest other parcel, say $i_\star$ (for an efficient algorithm, see Section 3.4 and Appendix D). Every small parcel i always has a closest other parcel $i_\star$: we say 'i points to $i_\star$'. Note, several parcels can point to the same parcel, and two small parcels can point to each other (they are 'double-linked'). Small parcels can point to other small parcels, because in some cases a large number of parcels in a region can become small in a single time step. In such cases, allowing mergers between small parcels ensures that merging takes place over small distances. If $i_\star$ is a large parcel with $V_{i_\star} \geq V_{\min}$, then $i_\star$ does not point to another parcel, but small parcels with $V_i < V_{\min}$ can point to $i_\star$ (all links are 'single links').

Details of the numerical algorithm which enforces these rules can be found in Appendix D, where directed graphs are used to represent different configurations of parcel links. The algorithm prevents chains of parcels from merging, as in parcel a pointing to b pointing to c and so on. This is done by breaking some single links between small parcels using an iterative approach. Otherwise, mergers could result in a highly distorted parcel and may be more difficult to parallelise. Nevertheless, the algorithm ensures that every small parcel is merged in each time step. This is important in particular for the Taylor-Green flow studied in Section 4.1, where we found that the number of small parcels increased without bound when using a simpler approach that did not ensure this. The merging algorithm results in a unique solution, except if the neighbouring parcels are equidistant from the small parcel to be merged with.

Similar to parcel splitting, parcel merging aims to preserve the total area, centroid, and second-order moments. The merged parcel has area $V = \pi ab = \sum_i V_i$ (where $V_i = \pi a_i b_i$ and the sum is over all merging parcels), centroid $\boldsymbol{x} = V^{-1} \sum_i V_i \boldsymbol{x}_i$, and (tentative) second-order moments

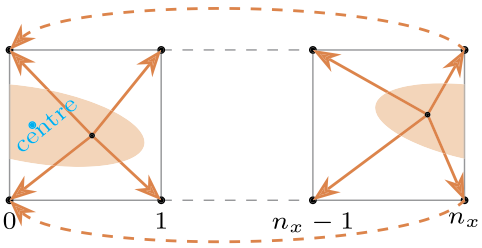

Fig. 3. Parcel to grid interpolation across periodic boundaries (x grid points are labelled). Here, the elliptical parcel whose centre lies in the first horizontal grid cell has a support point in the periodic extension lying in the grid cell at the opposite end of the domain.

$$\begin{aligned} B^\star_{11} &= \frac{1}{V}\sum_i V_i\left(4\check{x}_i^2 + B_{11,i}\right) \\ B^\star_{12} &= \frac{1}{V}\sum_i V_i\left(4\check{x}_i\check{y}_i + B_{12,i}\right) \\ B^\star_{22} &= \frac{1}{V}\sum_i V_i\left(4\check{y}_i^2 + B_{22,i}\right) \end{aligned} \tag{19}$$

where $\check{\boldsymbol{x}}_i \equiv \boldsymbol{x}_i - \boldsymbol{x}$ (see Appendix D for details). They are tentative because these matrix components as given do not generally preserve the total area, i.e. $B^\star_{11}B^\star_{22} - (B^\star_{12})^2 \neq a^2b^2$. In order to resolve this issue, we simply scale each entry in Eq. (19) so that the new matrix preserves area, i.e.

$$\boldsymbol{B} = \frac{ab}{\sqrt{B^\star_{11}B^\star_{22} - (B^\star_{12})^2}}\boldsymbol{B}^\star . \tag{20}$$

Note that merger is the inverse of splitting, in that two split parcels, if allowed to re-merge, would give back the original parent parcel. In general, however, the merging parcels are not identical nor aligned, so it is impossible to exactly conserve the second-order moments after merging, though area and centroid are conserved.

2.7. Enforcing boundary conditions on parcel operations

We use periodic boundaries in the horizontal (x) direction and free-slip boundaries in the vertical (y) direction. The implementation of periodic boundary conditions is the same as in PIC with point-like parcels. Parcels and support points that leave the domain on one side reenter on the opposite side. A simulation having n_x grid cells horizontally consists of the same number of grid points, i.e. the rightmost grid point of Fig. 3 is a periodic copy of grid point 0 on the left. In the example shown, one of the ellipse support points lies in grid cell 1 along with the ellipse centre, while the other lies in grid cell n_x across the periodic edge. The orange lines show how the support points contribute to the gridded values at the cell corners. Note that both support points contribute to the two grid points on the periodic edge ($i_x = 0$).

To ensure the x coordinate of a point lies within the periodic domain, a simple construction is used. Let $x_c = (x_{\min} + x_{\max})/2$ denote the domain centre in x, and $L_x = x_{\max} - x_{\min}$ denote the domain width. Then we use

$$x \leftarrow x - L_x\,[2(x - x_c)/L_x] \tag{21}$$

where $[s]$ denotes the 'integer part' of s. This integer part is -1 if $x < x_{\min}$; in this case x is increased by L_x. The integer part is $+1$ if $x > x_{\max}$; in this case x is decreased by L_x. Otherwise x is unchanged. It is assumed that x is within $L_x/2$ of a periodic edge, but in practice x is always within a few grid spacings Δx.

For the free-slip impermeable boundaries at $y = y_{\min}$ and $y_{\max}$, a different interpolation in the parcel-to-grid algorithm is required. We introduce a row of halo cells next to each y boundary, effectively extending the y grid points from $i_y = -1$ to $n_y + 1$. Then if an ellipse support point lies within a halo cell, see e.g. Fig. 4, each parcel attribute q_i and area V_i is interpolated to the four corners of that cell in the normal way, using Eq. (14) to obtain the product $\bar{q}\bar{V}$ and Eq. (15) to obtain $\bar{V}$. Next, *after all parcel points have been interpolated*, the gridded values of $\bar{q}\bar{V}$ and $\bar{V}$ at $i_y = -1$ are added to those at $i_y = 1$; likewise the gridded values at $i_y = n_y + 1$ are added to those at $i_y = n_y - 1$. Furthermore, the gridded values at $i_y = 0$ and n_y are doubled. Finally, $\bar{q}\bar{V}$ is divided by $\bar{V}$ to determine $\bar{q}$. This construction ensures that the total parcel area $\sum_i V_i$ equals the total gridded area $\sum_{\boldsymbol{x}} \bar{V}(\boldsymbol{x}) = L_xL_y$, and moreover ensures $\sum_i q_iV_i = \sum_{\boldsymbol{x}} \bar{q}(\boldsymbol{x})\bar{V}(\boldsymbol{x})$ for each attribute.

Care must also be taken when interpolating from the grid to the ellipse support points, see Eq. (17). If one of these support points lies in a halo cell, interpolation requires the gridded values at the outer edge of that cell (either at $i_y = -1$ or $n_y + 1$). We only need this interpolation for the velocity field to advect the parcels, for the strain matrix to update the shape matrix $\boldsymbol{B}$, and for the vorticity tendency to update $\mathrm{d}\zeta_i/\mathrm{d}t$ on each parcel i.

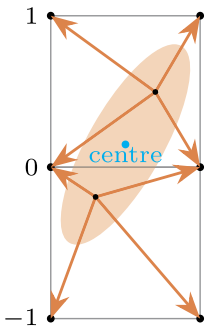

Fig. 4. Parcel to grid interpolation across free slip boundaries (y grid points are labelled).

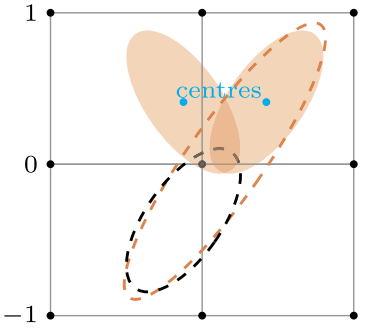

Fig. 5. Parcel mirroring at vertical boundaries. The child parcel in the halo region (black, dashed ellipse) is mirrored such that its centre is inside the domain. The orange, dashed ellipse denotes the parent parcel that was split.

For the velocity, we assume the x component $\bar{u}$ is symmetric across the y boundaries for simplicity, so we copy the values at $i_y = 1$ to $i_y = -1$, and likewise copy the values at $i_y = n_y - 1$ to $i_y = n_y + 1$. We assume the y component $\bar{v}$ is anti-symmetric, so we do the same copy but also flip the sign of $\bar{v}$. This is compatible with the no normal flow boundary conditions, $\bar{v} = 0$ at $i_y = 0$ and n_y. We have tested a more complicated interpolation which exploits the definition of vorticity, $\bar{u}_y = \bar{v}_x - \bar{\zeta}$, but numerical tests indicate this produces only very minor differences because ellipse support points are never more than a small fraction of the grid cell width Δy from a boundary. The current procedure is efficient.

For consistency, for the strain matrix $\boldsymbol{S}$ components, we assume symmetry for $\bar{u}_x$ and $\bar{v}_y$ $(= -\bar{u}_x)$, and anti-symmetry for $\bar{v}_x$. For $\bar{u}_y$, we use the definition of vorticity, but this requires $\bar{\zeta}$ in the halo regions. As there are no symmetry requirements to guide us here, we use extrapolation,

$$\begin{aligned} \bar{\zeta}_{i,-1} &= 2\bar{\zeta}_{i,0} - \bar{\zeta}_{i,1} \\ \bar{\zeta}_{i,n_y+1} &= 2\bar{\zeta}_{i,n_y} - \bar{\zeta}_{i,n_y-1} \end{aligned} \qquad \forall i = 0,\, 1,\, \ldots,\, n_x - 1\,. \tag{22}$$

This extrapolation assumes the second y derivative of $\bar{\zeta}$ is zero at each domain edge. For consistency, the same extrapolation is used to set the gridded vorticity tendency $\bar{b}_x$ in the halo regions.

Notably, it is rare that parcel support points lie in the halo, and when they do, they lie close to the inner edge of the halo region near one of the y boundaries (this is ensured by the time step choice as explained in Section 3.2). Thus, such support points receive their dominant contribution from grid points along the boundaries. To ensure this, parcel centres are forced to lie within the physical domain after each full time step. The vanishing of $\bar{v}$ on the y boundaries implies that the parcel centres remain within the domain, assuming accurate advection. On the other hand, a parcel near a boundary could split into two, and one part could have a centre lying in a halo. When this occurs, the parcel centre is mirrored across the domain edge and the B_{12} component of its shape matrix $\boldsymbol{B}$ is flipped in sign. This has the effect of mirroring the entire parcel across the domain edge (cf. Fig. 5). In this way, all parcel centres remain inside the physical domain, though support points may still lie in the halos.

2.8. Incompressibility

In general, parcel methods struggle to maintain incompressibility, as a result of using a finite number of parcels to represent the flow (see e.g. [20] and section 4.8 in [4]). One common solution is to occasionally re-distribute the parcels to ensure the gridded area found by particle interpolation is the same as the actual grid box area (see e.g. [5] and references therein). For short-time integrations, the error in the interpolated gridded area can be kept small when using a sufficient number of parcels [4,5,10]. However, for long-time integrations, a remedy is required. Re-distributing parcels is excessively

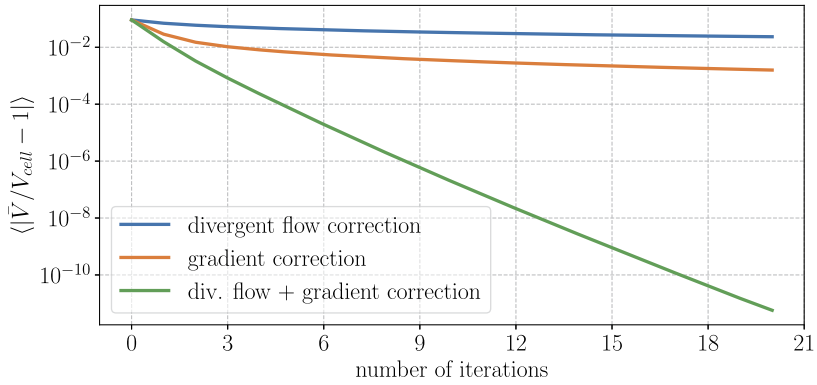

Fig. 6. Reduction of the average error of the gridded area $\bar{V}$ due to the divergent flow and gradient correction. The divergent flow and gradient corrections alone do not correct the gridded area as well as when they are combined.

diffusive [5]; in particular, it destroys fine-scale structure which might be needed to resolve important features like fronts. Below, we discuss an effective alternative which preserves this fine-scale structure. We perform two operations: a divergent flow correction followed by a parcel density gradient correction. This is done at the end of every time step (it can be done less frequently but it is efficient), and usually repeated twice (tests show that this is adequate, see below). The corrections provide a systematic way to ensure the gridded area remains nearly uniform.

2.8.1. Divergent flow correction

The basic idea here is to displace parcel centres slightly to correct for errors in the gridded area $\bar{V}$ in Eq. (15). In theory, we would like to maintain $\bar{V} = V_{\text{cell}} = \Delta x \Delta y$ at all times, but invariably $\bar{V} \neq V_{\text{cell}}$ due to the advection of a finite number of parcels. To correct for this, we add a purely divergent displacement field $\boldsymbol{d} = \nabla\phi$ where $\phi(\boldsymbol{x})$ is the solution to the Poisson equation

$$\nabla^2\phi = \bar{V}/V_{\text{cell}} - 1\,. \tag{23}$$

Tests with random area errors show that this correction rapidly and monotonically reduces those errors, with the biggest error reduction occurring in the first iteration. However, the reduction in error is not as great as found when coupling this correction to the gradient correction discussed next.

2.8.2. Gradient correction

In order to further improve the homogeneous distribution of parcels over the grid, parcels are moved down gradients of gridded area. This additional correction is applied inside each individual grid box, and we ensure a parcel cannot leave a grid box due to this correction, so that the flow field remains continuous. Below, we use $\tilde{x}$ to denote the x coordinate relative to the grid cell (i.e. it has a value between 0 and 1). For a one-dimensional problem, we can specify a gradient correction in position by a fractional grid-box distance $\mathfrak{d}\tilde{x}$ at the end of the time-step of the form

$$\mathfrak{d}\tilde{x} = C\tilde{x}(1-\tilde{x}) \tag{24}$$

where C is a pre-factor which depends on the gradient in gridded area: $|C|$ determines the maximum degree to which the locations of parcel centres are compressed at the edges of the cell. We limit this such that $|C| \leq C_{\max}$. Otherwise, C between grid points i and $i+1$ is given by

$$C = -\beta\frac{\bar{V}_{i+1} - \bar{V}_i}{V_{\text{cell}}}\,. \tag{25}$$

The default value of β, 1.8, was chosen on the basis of one-dimensional tests with parcels initially located in random locations, where the gradient correction and the divergent flow correction were combined.

In the two-dimensional case, a similar correction is applied independently in the y-direction. The gradients with respect to x which determine C_x are determined by linear interpolation between the bottom and the top of the grid box (depending on the vertical position of the parcel), while those in y are determined by linear interpolation between the left and right sides of the grid box.

In Fig. 6 we show the convergence of the gridded area $\bar{V}$ towards the cell area V_{cell} when applying both correction algorithms. To generate this figure we first placed four circular parcels per grid cell (as described below in Section 3.1) in the domain $[0,1]^2$ discretized by 32×32 cells, and we assigned an area of $V_{\text{cell}}/4$ to each parcel. We then perturbed their horizontal and vertical positions randomly by up to $\pm\Delta x/4$ from this optimal setup and applied the parcel correction algorithms iteratively as would be done in a simulation (cf. Fig. 7). Both the divergent flow and the gradient correction reduce the area error, but the combination significantly increases the convergence rate.

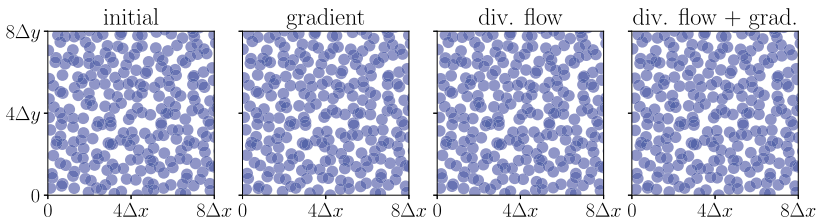

Fig. 7. Parcel configurations before (leftmost) and after applying the different parcel correction methods 2 times in the domain $[0, 1]^2$, discretised into 32×32 cells. Each subplot shows only the lower left sixteenth of the domain. The divergent flow correction tends to cluster the parcels compared to the gradient correction. The parcels are more homogeneously distributed after applying the divergent flow and gradient corrections sequentially.

2.9. Energy conservation

The total energy, here kinetic plus potential, is conserved in the density-stratified flow model considered (see Section 2.2). The degree to which the numerical model conserves energy is a useful measure of accuracy, and is discussed in relation to the Straka and Robert test cases in Section 4.2 and Section 4.3 below. Here, we discuss the calculation of kinetic and potential energy using elliptical parcels.

The kinetic energy (per unit mass) is the integral

$$K = \frac{1}{2} \iint (u^2 + v^2)\, \mathrm{d}x\mathrm{d}y \tag{26}$$

where the integral is over the entire domain. The actual kinetic energy is K multiplied by the constant background density ρ_0 (a consequence of the Boussinesq approximation). Numerically, we calculate K by a sum over parcels:

$$K = \frac{1}{2} \sum_i (u_i^2 + v_i^2)\, V_i \tag{27}$$

where (u_i, v_i) is the velocity at the parcel centre, and i ranges over all parcels.

The potential energy (per unit mass) is the integral

$$P = -\iint b\, y\, \mathrm{d}x\mathrm{d}y\,. \tag{28}$$

Here, recall $b = -g(\rho - \rho_0)/\rho_0$ where g is the acceleration due to gravity. Numerically, we calculate P by the sum

$$P = -\sum_i b_i\, y_i\, V_i \tag{29}$$

where y_i is the height of the parcel centre. However, this potential energy cannot all be converted to kinetic energy. The part which cannot be converted is the potential energy of the hydrostatic equilibrium, P_{ref}, found by re-arranging b as a monotonically increasing (or non-decreasing) function of y. Using parcels, this is done by sorting b_i, then obtaining a reference height y^{ref} by summing the parcel areas V_i divided by the domain width L_x. Specifically, we compute

$$y_1^{\mathrm{ref}} = y_{\min} + \frac{V_1}{2L_x} \tag{30}$$

$$y_i^{\mathrm{ref}} = y_{i-1}^{\mathrm{ref}} + \frac{V_{i-1} + V_i}{2L_x}, \quad i = 2,\, 3,\, \ldots,\, n_{parcels} \tag{31}$$

then find P_{ref} from

$$P_{\mathrm{ref}} = -\sum_i b_i\, y_i^{\mathrm{ref}}\, V_i\,. \tag{32}$$

Since b is materially conserved, this only needs to be computed at $t = 0$; it remains constant. The available potential energy is then just the difference $P - P_{\mathrm{ref}}$. In the results below, we simply use P to denote the available potential energy.

The flow model in Section 2.2 exactly conserves any functional of b, namely

$$\iint F(b)\, \mathrm{d}x\mathrm{d}y = \sum_i F(b_i)\, V_i \tag{33}$$

where $F(b)$ is an arbitrary function of b; this is a consequence of material conservation of b. The numerical model only conserves the linear functional $F(b) \propto b$ due to mixing associated with parcel merging.

3. Structure of the algorithm

3.1. Initialisation: parcel locations and attributes

To prepare the initial conditions for an EPIC simulation, the first step is to place the parcels on the grid. Here we use $m \times m$ parcels per grid cell ($m = 3$ is sufficient as the tests below demonstrate). In each cell, their centres $\boldsymbol{x}_i$ are placed in a regular array, at the locations $(\Delta x(i - 1/2)/m, \Delta y(j - 1/2)/m)$ relative to the lower left corner of the grid cell, for $i = 1, ..., m$ and $j = 1, ..., m$. For square grid cells, $\Delta x = \Delta y$, each parcel is a circle with area $\Delta x \Delta y/m^2$. Otherwise, we initialise with ellipses of the same area; if $\Delta x > \Delta y$, they are aligned with the x axis and have aspect ratio $\lambda = \Delta x/\Delta y$; if $\Delta x < \Delta y$, they are aligned with the y axis and have aspect ratio $\lambda = \Delta y/\Delta x$. The grid length ratio is assumed to lie in the range $\lambda_{\max}^{-1}$ to $\lambda_{\max}$, but a unit value is recommended. In principle, higher grid anisotropy can be handled using a different arrangement of parcel centres (not $m \times m$). In practice, we currently split parcels if $\lambda > \lambda_{\max}$.

Once the parcels have been placed, the next step is to prepare the initialisation of their attributes. The user supplies a gridded field for each attribute, say $\bar{q}(\bar{\boldsymbol{x}}, 0)$ for all grid points $\bar{\boldsymbol{x}}$, and the algorithm below converts these data into parcel attributes $q_i(0)$, for $i = 1, ..., n$ where n is the total number of parcels (this algorithm is adapted from section 2.2 of [21]). The q_i are chosen so that the parcel-interpolated gridded field in Eq. (14) matches the given gridded field (to high precision, see below).

To this end, first the parcel weights ϕ are computed from Eq. (16) but using the parcel centres $\boldsymbol{x}_i$ in place of the support points $\boldsymbol{x}_i^{\pm}$. Then the parcel density $\bar{\rho}$ at each grid point $\bar{\boldsymbol{x}}$ is determined from

$$\bar{\rho}(\bar{\boldsymbol{x}}) = \sum_{i \in \mathcal{P}(\bar{\boldsymbol{x}})} \phi(\boldsymbol{x}_i - \bar{\boldsymbol{x}}) \,. \tag{34}$$

For the parcel placement described above, this is equal to m^2 for all grid points (the approach taken here is analogous to the interpolation algorithm for gridded area in Section 2.4). From this, the inverse parcel density α_i is determined from

$$\alpha_i = 1/\sum_{\bar{\boldsymbol{x}} \in \mathcal{G}_i} \phi(\boldsymbol{x}_i - \bar{\boldsymbol{x}}) \bar{\rho}(\bar{\boldsymbol{x}}) \,. \tag{35}$$

Again, this works out to $1/m^2$ for all parcels when placing them as described above. The next steps can be repeated for each attribute q. First, a guess is made for the parcel attribute:

$$q_i = \alpha_i \sum_{\bar{\boldsymbol{x}} \in \mathcal{G}_i} \phi(\boldsymbol{x}_i - \bar{\boldsymbol{x}}) \bar{q}(\bar{\boldsymbol{x}}, 0) \,. \tag{36}$$

Next, a residual $\bar{R}(\bar{\boldsymbol{x}})$ is computed from

$$\bar{R}(\bar{\boldsymbol{x}}) = \bar{q}(\bar{\boldsymbol{x}}, 0) - \sum_{i \in \mathcal{P}(\bar{\boldsymbol{x}})} \phi(\boldsymbol{x}_i - \bar{\boldsymbol{x}}) q_i \,. \tag{37}$$

Then the parcel attributes are updated using

$$q_i \leftarrow q_i + \alpha_i \sum_{\bar{\boldsymbol{x}} \in \mathcal{G}_i} \phi(\boldsymbol{x}_i - \bar{\boldsymbol{x}}) \bar{R}(\bar{\boldsymbol{x}}) \,. \tag{38}$$

These values are accepted if the maximum value of $|\bar{R}| < \varepsilon_q$ (a prescribed tolerance), otherwise we repeat the last two operations (updating $\bar{R}$ and q_i). The tolerance used here is $\varepsilon_q = 10^{-9} \bar{q}'_{\text{rms}}$ where $\bar{q}'$ is the departure of $\bar{q}$ from its domain mean. In practice, the algorithm converges rapidly, requiring only a few tens of iterations for convergence. If the user supplies a constant field $\bar{q}$, the algorithm is bypassed and we simply assign $q_i = \bar{q}$ (constant).

Note that this algorithm still works even for an anisotropic grid. Only the parcel centres are required because the interpolation of the ellipse support points is equivalent to the interpolation of the ellipse centres for elliptical parcels aligned with the x or y axes. This property is convenient for initialisation.

3.2. Time stepping: low-storage Runge-Kutta and adaptation

In EPIC, all time-dependent quantities ($\boldsymbol{x}_i$, $\boldsymbol{B}_i$ and ζ_i) are propagated in time using the five-stage fourth-order 2N-storage Runge-Kutta scheme (ls-RK4) [22,23] to reduce the memory footprint. Each such quantity, say q_i, then requires only one additional array δq_i. Given a time step Δt, the ls-RK4 scheme loops over the following two steps for $j = 1, 2, ..., 5$:

$$\delta q_i(t_j) = \mathfrak{a}_j \delta q_i(t_{j-1}) + \dot{q}_i(t_j) \tag{39}$$

$$q_i(t_{j+1}) = q_i(t_j) + \mathfrak{b}_j \delta q_i(t_j) \Delta t \tag{40}$$

Table 1
Coefficients of the five-stage fourth-order 2N-storage Runge-Kutta scheme.

j	$\mathfrak{a}_j$	$\mathfrak{b}_j$	$\mathfrak{c}_j$
1	0	$\frac{1432997174477}{9575080441755}$	0
2	$-\frac{567301805773}{1357537059087}$	$\frac{5161836677717}{13612068292357}$	$\frac{1432997174477}{9575080441755}$
3	$-\frac{2404267990393}{2016746695238}$	$\frac{1720146321549}{2090206949498}$	$\frac{2526269341429}{6820363962896}$
4	$-\frac{3550918686646}{2091501179385}$	$\frac{3134564353537}{4481467310338}$	$\frac{2006345519317}{3224310063776}$
5	$-\frac{1275806237668}{842570457699}$	$\frac{2277821191437}{14882151754819}$	$\frac{2802321613138}{2924317926251}$

where $\dot{q}_i = \mathrm{d}q_i/\mathrm{d}t$ is the tendency, $t_j = t_0 + \mathfrak{c}_j \Delta t$ and $\mathfrak{a}_j$, $\mathfrak{b}_j$ and $\mathfrak{c}_j$ are numerical coefficients listed in Table 1 (note $\mathfrak{a}_1 = \mathfrak{c}_1 = 0$, hence $\delta q_i(t_0)$ is not required and $t_1 = t_0$). δq_i stores a weighted combination of the actual tendency and those from earlier stages for memory efficiency (note that the weights do not add up to one at each stage). $\dot{q}_i$ must be updated at each stage j of the scheme, since the tendency may depend on other evolving quantities as well. This also means that all quantities must be iterated in the same loop, not sequentially.

Before entering the above loop, we adapt the time step using

$$\Delta t = \min\left(\frac{\alpha_s}{\gamma_{\max}}, \frac{\alpha_b}{N_{\max}}\right) \tag{41}$$

where α_s and α_b are dimensionless constants,

$$\gamma_{\max} = \frac{1}{2}\sqrt{\max_{\vec{x}}\{(\bar{u}_x - \bar{v}_y)^2 + (\bar{u}_y + \bar{v}_x)^2\}} \tag{42}$$

is the maximum gridded strain rate (see Appendix E), and

$$N_{\max} = \sqrt{\max_{\vec{x}} \|\nabla \bar{b}\|} \tag{43}$$

is the maximum (generalised) buoyancy frequency [18].

The constant α_s controls how much stretch a parcel can experience in a time step Δt. The maximum parcel stretch occurs when the major axis of the parcel is aligned with a local straining flow in which $\gamma = \gamma_{\max}$. Taking this axis to be the x axis without loss of generality, and a straining flow with $u = \gamma_{\max} x$ and $v = -\gamma_{\max} y$, one can use Eq. (9) to show that $\dot{\lambda} = 2\gamma_{\max}\lambda$. Thus, over one time step, the maximum parcel stretch is

$$\frac{\lambda(t + \Delta t)}{\lambda(t)} = e^{2\alpha_s} \tag{44}$$

assuming perfect time integration and that $\Delta t = \alpha_s/\gamma_{\max}$ in Eq. (41). It is less than this if Δt is limited by $N_{\max}$.

In EPIC, α_s is chosen small enough to limit the distance between the ellipse support points, c in Eq. (11), to the *minimum* grid spacing, $\Delta_g = \min(\Delta x, \Delta y)$. In this way, parcels may deform and split at subgrid scales, thereby providing a good representation of the subgrid flow. Moreover, we must ensure that parcel support points never extend beyond the halo regions next to each y boundary (see Section 2.7) during a time step; the condition $c < \Delta_g \leq \Delta y$ strongly satisfies this constraint (indeed a support point should never reach further than midway into the halo, but given that the time stepping is not perfect, this is done for safety).

Using $a^2 = ab\lambda$ and $b^2 = ab/\lambda$, the restriction $c^2 = a^2 - b^2 < \Delta_g^2$ implies

$$\lambda - \lambda^{-1} < \frac{\Delta_g^2}{ab}. \tag{45}$$

If we consider the worst-case scenario in which $\lambda(t + \Delta t) = \lambda_{\max} e^{2\alpha_s}$, then the above inequality implies

$$e^{2\alpha_s} < \frac{\Delta_g^2}{\lambda_{\max} ab}, \tag{46}$$

neglecting λ^{-1} compared to λ.

Now, the most restrictive condition is obtained when the right-hand side takes its minimum value, i.e. for a parcel of maximum size $ab = V_{\max}/\pi$:

$$e^{2\alpha_s} < \frac{\pi \Delta_g^2}{\lambda_{\max} V_{\max}}. \tag{47}$$

This is a restriction on α_s, but may equally be viewed as a restriction on either $\lambda_{\max}$ or $V_{\max}/\Delta_g^2$, i.e.

$$\lambda_{\max}\frac{V_{\max}}{\Delta_g^2} < \pi e^{-2\alpha_s}. \tag{48}$$

Written this way, it is convenient to choose a small value of α_s (for time-stepping accuracy) and a moderate value of $\lambda_{\max}$ (to control parcel splitting), then ensure $V_{\max}/\Delta_g^2$ satisfies Eq. (48). For an isotropic grid (recommended), note that $V_{\max}/\Delta_g^2 = \tilde{V}_{\max}$, the dimensionless maximum parcel area fraction. Otherwise, $V_{\max}/\Delta_g^2 = \tilde{V}_{\max}A_g$ where $A_g = \max(\Delta x/\Delta y, \Delta y/\Delta x)$ is the grid aspect ratio.

The constraint in Eq. (48) not only determines the maximum distance between ellipse support points, but also the maximum separation distance when parcels are split ($2h$). By neglecting λ^{-1} compared to λ in Eq. (45) above, we get $a^2 < \Delta_g^2$. Substituting $h = \sqrt{3}a/4$ as derived in Appendix C, we find $2h < \sqrt{3}\Delta_g/2$.

We require the additional constant α_b not just to resolve buoyancy oscillations (frequencies of $\mathcal{O}(N_{\max})$) but also to ensure a finite time step when starting from a flow at rest (which would have $\gamma_{\max} = 0$). In practice, we take $\alpha_b = \alpha_s = \alpha$, a single user-defined parameter, based on numerical tests presented in Appendix G.

3.3. Tendency calculation

The time stepping above requires the tendencies $\dot{q}_i$ for each time-dependent quantity q_i (and for each parcel i). These are calculated as follows. First, the parcel attributes V_i, b_i and ζ_i are interpolated to obtain the corresponding gridded fields $\bar{V}$, $\bar{b}$ and $\bar{\zeta}$ (in EPIC, this is `par2grid`). Next, the gridded vorticity $\bar{\zeta}$ is inverted to find the gridded velocity field $\bar{\boldsymbol{u}}$ (this is `vor2vel`, see Appendix A). These components are differentiated in x to form the strain matrix $\boldsymbol{S}$ (only $\bar{u}_x$ and $\bar{v}_x$ need be computed, since $\bar{v}_y = -\bar{u}_x$ by incompressibility, and $\bar{u}_y = \bar{v}_x - \bar{\zeta}$ by the definition of vorticity). Next, the gridded vorticity tendency $\bar{b}_x$ is accurately computed using FFTs and wavenumber multiplication (as are $\bar{u}_x$ and $\bar{v}_x$). Then these gridded tendencies are interpolated to the parcel support points to provide the parcel tendencies $\dot{\boldsymbol{x}}_i$, $\dot{\boldsymbol{B}}_i$ and $\dot{\zeta}_i$ (this is `grid2par`).

3.4. Parcel merging and splitting

After the parcels have been advanced one time step, they are checked for merging and for splitting. For merging, any small parcels with $V_i < V_{\min}$ are identified and added to a list of parcels available for merger, of length n_{merge}. To minimise search costs, separate lists of parcels occupying each grid cell are created (this can be done in $\mathcal{O}(n)$ operations for n parcels). The list of small parcels i is then processed to locate the nearest other parcel $i_\star$. This is done by locating the grid point (i_x, i_y) closest to parcel i (using nearest integer arithmetic), then by searching only the 4 grid boxes sharing this grid point (or 2 when adjacent to a y boundary). After the n_{merge} small parcels have been processed in this way, the merger rules described in Section 2.6 are enforced (details of the algorithm may be found in Appendix D).

Note that merging changes all parcel attributes. The new attribute of the merged parcel, say q_m, is the area-weighted mean of the attributes of all merging parcels, i.e.

$$q_m = V_m^{-1}\sum_i q_i V_i \quad \text{where} \quad V_m = \sum_i V_i \tag{49}$$

where the sum extends over the merging parcels only. The same formula is used for position $\boldsymbol{x}_m$, vorticity ζ_m and buoyancy b_m (and other attributes if included). In this way the total mass, proportional to $\sum_{i=1}^n b_i V_i$, and the total circulation $\sum_{i=1}^n \zeta_i V_i$ remains conserved for all time, consistent with Eq. (1) and Eq. (2). The shape matrix $\boldsymbol{B}_m$ is handled slightly differently in that, after the area weighting, $\boldsymbol{B}_m$ needs to be scaled to preserve the parcel area (see Section 2.6).

Next, any parcel satisfying the splitting criteria (see Section 2.5) is split into two parts, creating a new parcel with identical attributes. Splitting is also conservative, and moreover preserves the second-order spatial moments, besides area and centroid (the zeroth and first-order moments).

3.5. Parcel adjustment to correct gridded area errors

At the end of every time step, and after all parcel splitting and merging, the parcels are slightly shifted to correct for gridded area errors. Ideally, we would like to preserve $\bar{V} = V_{\text{cell}} = \Delta x \Delta y$, but the advection of a finite number of parcels, splitting and merging inevitably lead to a discrepancy. The algorithm described in Section 2.8 is used to keep this discrepancy small, typically around 0.01% of V_{cell} (see tests in Appendix G). It can be reduced further, if desired (and at cost), by increasing the number of times the divergent flow and parcel gradient corrections are applied, as shown in Fig. 6.

3.6. Flowchart

The structure of the EPIC algorithm is shown in Fig. 8. In order to run a simulation, the user only needs to provide gridded field data from which the parcels are initialised (cf. Section 3.1). Before performing the first low-storage RK4 step, the time step is calculated and the initial configuration of field and/or parcel data is written to HDF5 [24] files. In a ls-RK4 sub-step, we first interpolate parcel attributes to the grid (`par2grid`), then compute the gridded velocity field using inversion (cf. Appendix A), and the vorticity tendency from the gridded buoyancy by differentiation in semi-spectral space.

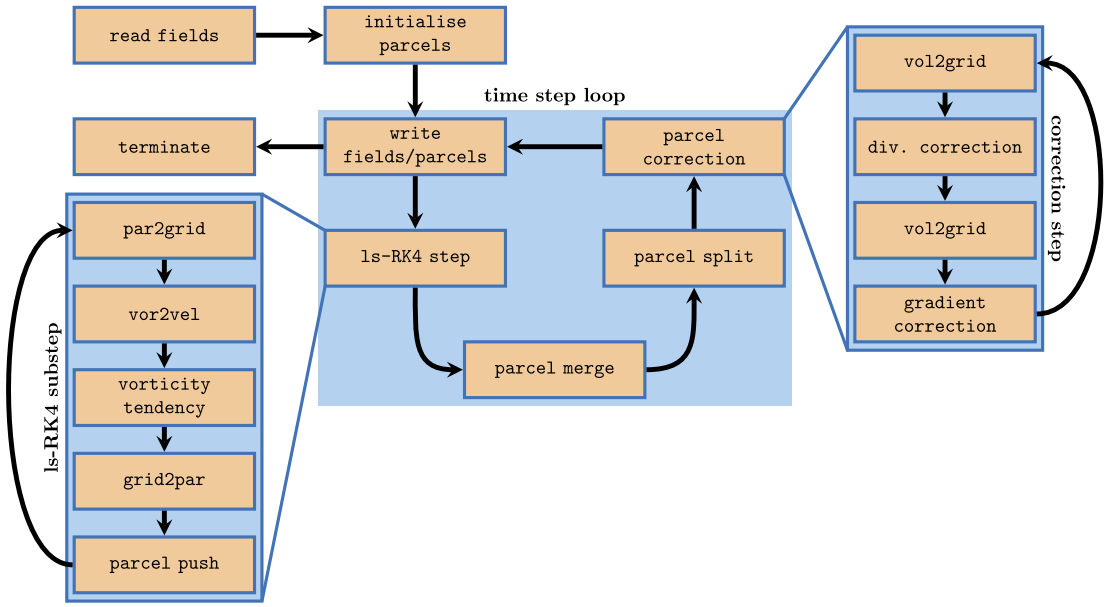

Fig. 8. Flow chart of the EPIC program. The user provides gridded data fields which are used to initialise the parcels. Occasionally, parcel and field data are written at specified time intervals (following a `par2grid` and before the first stage of the ls-RK4 algorithm). The parcels are then evolved in time using the ls-RK4 algorithm (each update of field tendencies requires a call to `par2grid`). In the first stage of this algorithm, the time step is adapted after the call to `vorticity tendency`. After each ls-RK4 step, the parcels are merged, split and then corrected for incompressibility. The program terminates once the time limit is reached.

Table 2
Recommended numerical settings in EPIC, independent of the grid resolution used.

Numerical parameter	Value
Initial number of parcels per cell	9
Maximum aspect ratio, $\lambda_{\max}$	4
Minimum parcel area fraction, $\tilde{V}_{\min}$	1/40
Maximum parcel area fraction, $\tilde{V}_{\max}$	1/2.89
Number of iterations to correct for incompressibility	2
Gradient correction pre-factor, β	1.8
Gradient correction max. compression, $C_{\max}$	0.5
Time step pre-factor, α	0.2

The strain matrix $\boldsymbol{S}$ needed for evolving the shape matrix $\boldsymbol{B}$ is similarly found by differentiation of the gridded velocity. Afterwards, these gridded quantities (the tendencies) are interpolated back to the parcels (`grid2par`) in order to propagate them.

After each ls-RK4 step, any very small parcels are merged with others and highly-elongated parcels are split. Then the parcel positions are corrected to enforce incompressibility, see Section 2.8. The parcel correction is an iterative procedure where the parcel area is interpolated to the grid (`vol2grid`) to apply the divergent flow correction, followed by a `vol2grid` to do the gradient correction. The number of correction steps can be controlled by the user, but normally two steps are sufficient to keep gridded area errors small (cf. Fig. 6). Before the updated configuration is written to the HDF5 files, a new time step is calculated as explained in Section 3.2. Once the time limit is reached, EPIC terminates.

4. Results

In the following subsections we present results for three standard fluid dynamical test cases: (1) the Taylor-Green flow, (2) a density current simulation developed by Straka [25], and (3) a test case by Robert [26] which collides a large warm bubble with a much smaller cold bubble. We use the Straka test case to examine the model sensitivity to its numerical input parameters (see also Appendix G), and establish a recommended model setup in Table 2. These default values are used in all simulations unless otherwise stated.

All simulations are performed on a laptop running an Intel(R) Core(TM) i7-10750H CPU @ 2.60 GHz. The timing comparison between the numerical models is obtained running EPIC single-threaded, otherwise we used 4 OpenMP threads.

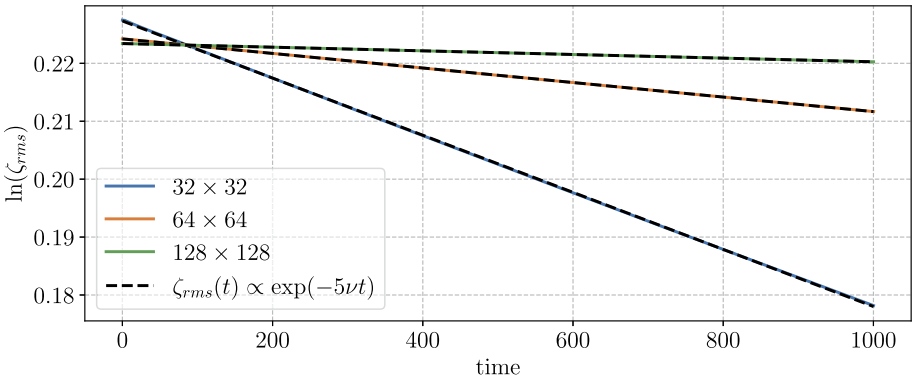

Fig. 9. Evolution of the r.m.s. vorticity $\zeta_{\rm rms}$ for the Taylor-Green test case with initially 9 parcels per cell. The decay reduces with increasing grid resolution. The slopes of the decay are estimates of the effective viscosity ν as given in Table 3.

Table 3
Viscosity as a function of grid resolution.

Number of grid cells	Viscosity, ν
32×32	9.86×10^{-6}
64×64	2.51×10^{-6}
128×128	6.30×10^{-7}

4.1. Taylor-Green test case

In our first test case we use, as the initial condition, the steady incompressible Taylor-Green flow [27] that exhibits two counter-rotating vortices:

$$u(x, y) = -\tfrac{1}{2} \sin 2x \sin y, \qquad v(x, y) = -\cos 2x \cos y \tag{50}$$

in the domain $[-\pi/2, \pi/2]^2$. The corresponding vorticity is $\zeta = \frac{5}{2} \sin 2x \cos y$. Note, this flow is dimensionless. We employ the same flow model outlined in Section 2.2, except that here $b = 0$. In a viscous fluid, the flow preserves its spatial form but decays as $e^{-5\nu t}$, where ν is the viscosity coefficient (the factor of 5 comes from the sum of the squared wavenumbers, $2^2 + 1^2$). The Taylor-Green flow thus provides an analytical solution to the Navier-Stokes equations. (Note: a simpler version of this test was used in [5] to check code accuracy; results for EPIC may be found in Appendix F.)

In EPIC, we do not include viscosity, so we expect the flow to remain approximately steady. The maximum vorticity in the flow is 2.5, corresponding to a rotation period of $4\pi/2.5$ or approximately 5 time units. While the flow is (approximately) steady, fluid particles still circulate, so in EPIC the parcels will move and deform. Moreover, they will split and merge, resulting in vorticity mixing. This will cause the flow to become (slightly) unsteady.

Here, we measure the impact of this by examining the evolution of the r.m.s. vorticity $\zeta_{\rm rms}$ in a series of simulations on grids of dimensions 32^2, 64^2 and 128^2. As shown in Fig. 9, the r.m.s. vorticity decays as if the flow were slightly viscous: the fits to an exponential decay are excellent (and continue well beyond the times shown), enabling us to estimate the effective viscosity ν by fitting $\ln(\zeta_{\rm rms})$ to a straight line with slope -5ν, as done in Table 3. Moreover, ν is seen to be inversely proportional to the total number of grid points. Dimensionally, we expect $\nu \approx C_\nu \zeta_{\max}(0) \Delta x \Delta y$ for some dimensionless constant C_ν which should be independent of grid resolution. The data in Table 3 support this relationship and indicate $C_\nu \approx 4.15 \times 10^{-4}$. Hence, the merger and splitting occurring in EPIC result in a weak diffusion. By comparison with conventional numerical methods like pseudo-spectral and semi-Lagrangian, the diffusion in EPIC appears to be much less for comparable grid resolutions [28]. While the latter study examined shallow-water flows, the numerical diffusion effects are expected to be broadly similar, and this is confirmed for the more realistic Straka and Robert test cases examined in the following subsections. From the viscosity diagnosed for the 32^2 grid, we can derive a scaling Reynolds number $\mathfrak{Re} = \zeta_{\max}(0)\ell^2/\nu \approx 6.26 \times 10^5$, with ℓ the being the shorter extent of one of the vortices (i.e. $\pi/2$). This compares to a prescribed value of 2.48×10^4 used in the convergence tests of the Taylor-Green flow in a previous study [23] (although the model used there can be used as an implicit LES, i.e. without a prescribed or parametrised diffusion).

A similar series of simulations having a smaller time step pre-factor of $\alpha = 0.05$ results in viscosity estimates that are only 3% to 4% smaller compared to Table 3. In fact, by also considering $\alpha = 0.1$, we find that C_ν linearly increases with α, closely obeying the relationship $C_\nu \approx 3.96 \times 10^{-4}(1 + 0.263\alpha)$. Hence, a larger time step results in (slightly) more diffusive behaviour, which suggests that parcel splitting and merging is more vigorous, likely because parcels tend to be more deformed on average.

We next examine the degree to which the flow remains incompressible. This is measured by the (relative) r.m.s. area error $\sqrt{\langle(\bar{V} - V_{\rm cell})^2\rangle}/V_{\rm cell}$. We consider three different initialisations, with 4, 9 and 16 (circular) parcels per grid cell. The

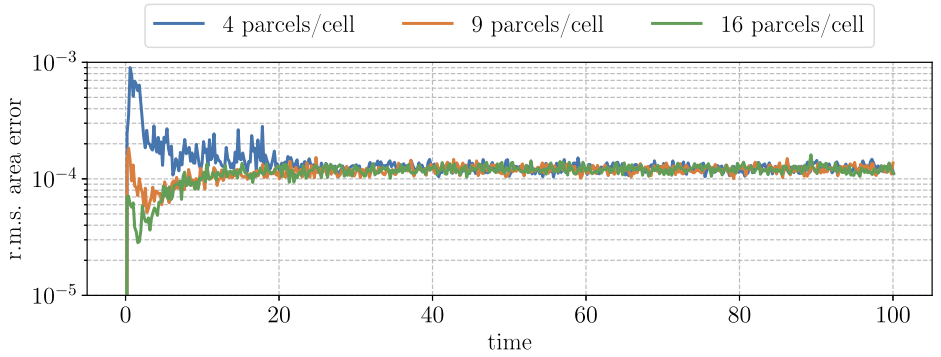

Fig. 10. Gridded relative r.m.s. area error for different initial numbers of parcels per cell. These results are obtained using a grid of 32×32 cells.

evolution of the relative r.m.s. area error for the three simulations is shown in Fig. 10, where we see that the eventual error is approximately the same (and small, around 0.01%) in all cases. This is because parcels can split and merge, and in time the number of parcels becomes closely similar despite using different numbers initially. At early times, however, the r.m.s. area error exhibits a significant overshoot when only 4 parcels per cell are used initially. A good choice appears to be 9 parcels per cell initially, as this keeps the relative r.m.s. area error approximately constant during the early stages of the simulation. The average number of parcels per grid cell saturates at around 21 with a standard deviation between grid cells of approximately 4 at late times, independent of the number of parcels used initially (not shown).

Finally, we examine the degree to which symmetry is maintained in EPIC. To this end, Fig. 11 shows the parcel configuration at times 0, 5, 20 and 80 (for the simulation using all the default parameters in Table 2). The parcels are coloured with respect to their aspect ratio λ (initially all parcels have $\lambda = 1$). The mean aspect ratio, time averaged, is approximately 2.38 with a standard deviation of 0.69 when using either 4, 9 or 16 parcels per cell to initialise. One may observe that, in time, symmetry is not preserved (compare for example the green-yellow spots at the top left and right of Fig. 11 at $t = 20$). Symmetry is broken by parcel mergers on or near the symmetry axis ($x = 0$) and across the periodic boundary ($x = \pi/2$), another symmetry axis of the Taylor-Green flow. It appears extremely difficult to design a parcel merger algorithm that preserves symmetry, without first imposing symmetry e.g. by simulating only one cell of the flow in a domain entirely enclosed by free-slip boundaries. Nonetheless, the vorticity field remains closely anti-symmetric in x, as shown in Fig. 12 which plots the r.m.s. value of $\zeta(x, y, t) + \zeta(-x, y, t)$ (for $x \geq 0$) scaled by $\zeta_{\max}(0)$ as a function of time. Symmetry is first broken between $t = 2$ and 3, but the loss of symmetry is only slight (around 0.15%) over the entire duration of the simulation, and saturates by around $t = 50$. The largest errors occur where vorticity gradients are largest, as expected (not shown).

4.2. Straka density current test case

In the second test case we initialise a cold bubble following the setup of Straka [25], where an ellipse with semi axes $(a_\circ, b_\circ) = (4, 2)$ km is centred at $(x_\circ, y_\circ) = (0, 3)$ km in a domain with extent $[-25.6, 25.6] \times [0, 6.4]$ km. Within the ellipse, i.e. where

$$r(x, y) = \sqrt{(x - x_\circ)^2/a_\circ^2 + (y - y_\circ)^2/b_\circ^2} \leq 1 \tag{51}$$

a temperature perturbation (a cold anomaly) is introduced having the form

$$\Delta T(r) = -\Delta T_{\max}[\cos(\pi r) + 1]/2 \tag{52}$$

with $\Delta T_{\max} = 15$ K. The total buoyancy [4] is therefore given by $b(r) = g\Delta T(r)/T_0$ with reference temperature $T_0 = 300$ K and gravitational acceleration $g = 9.81$ m/s^2 (there is no moisture in this setup). Unlike [25], we consider an incompressible flow and therefore do not need to solve a pressure or energy equation. Moreover, we consider an inviscid flow, whereas [25] used a moderately strong prescribed viscosity. Other results for the Straka density current test case without viscosity can be found in [29], who used radial basis function generated finite differences (RBF-FD) to solve the governing equations.

In Fig. 13 we show the parcel configuration in the right half of the domain at time 900 s. The simulation is initialised with 9 parcels per cell (the default) and 4096×512 grid cells. In the upper panel of the figure, the parcels are coloured according to their aspect ratio. At the interface between the passive and active regions, the parcels undergo significant stretching and have a relatively high aspect ratio. The corresponding parcel buoyancy is shown in the lower panel. Many small-scale vortices are evident, arising from early Kelvin-Helmholtz instabilities developing on the spreading density current.

The evolution of the energy, including the kinetic and potential components, is shown in Fig. 14. The EPIC model conserves the total energy very well considering the great complexity of the flow, and this is true even at lower grid resolution.

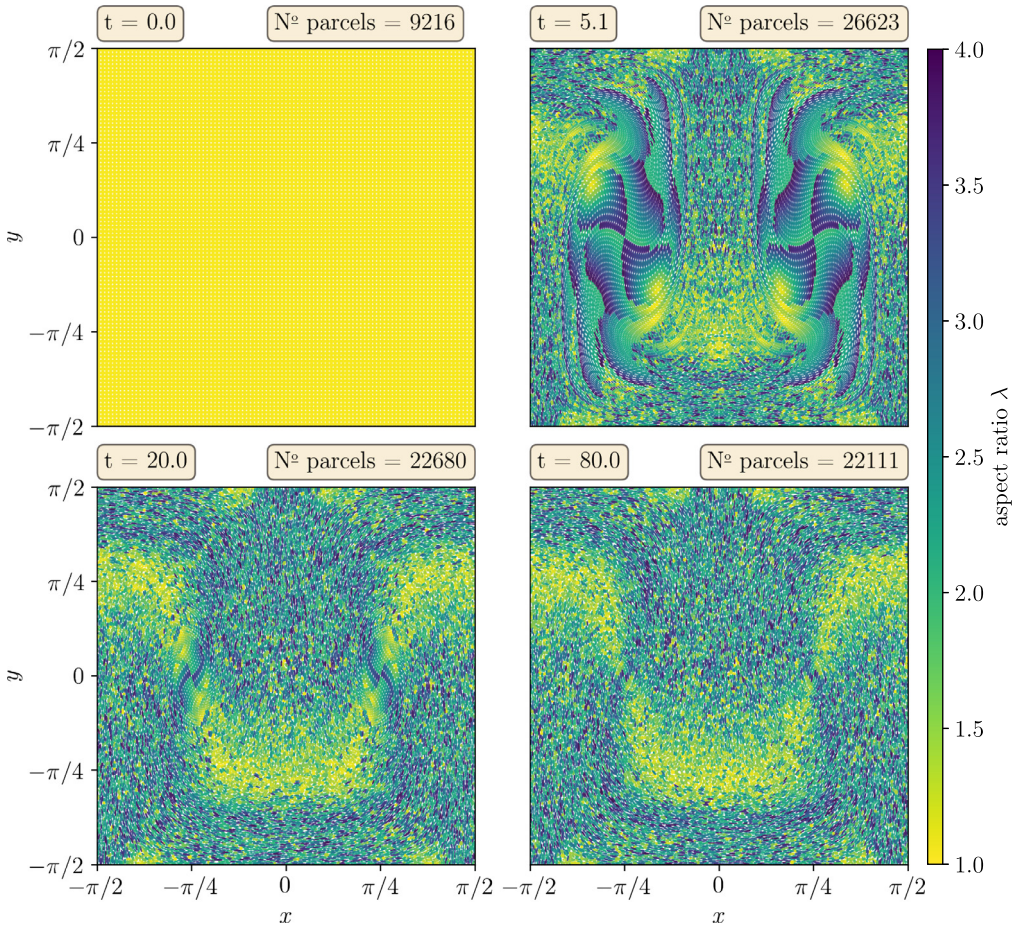

Fig. 11. Parcel configuration at times 0, 5, 20 and 80 (from left to right and top to bottom). The parcels are coloured according to their aspect ratio λ. This simulation starts with 9 parcels per cell on a 32×32 grid, giving a total of 9216 parcels.

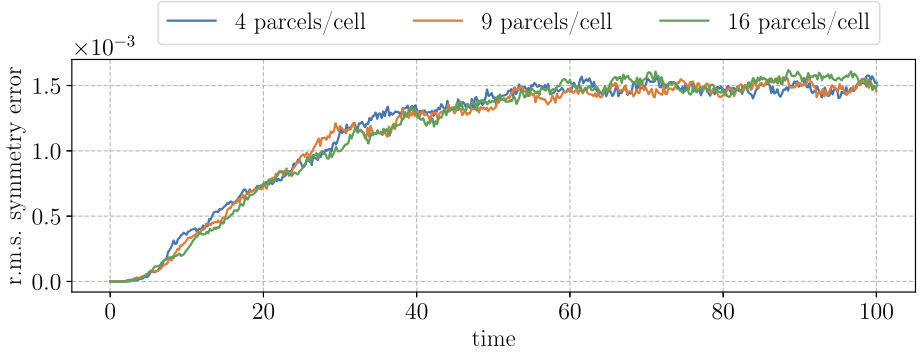

Fig. 12. The symmetry error in the Taylor-Green simulation as measured by the r.m.s. value of the vorticity folded into $x \geq 0$, normalised by the initial maximum vorticity.

The net energy loss over time as a function of grid resolution is listed in Table 4, including results for pseudo-spectral simulations using hyperviscosity and molecular viscosity, PS-h and PS-m (see below). The data indicate that doubling resolution reduces the loss in energy by less than a factor of 2. This is less than the expected factor of 4 for two-dimensional flows without buoyancy [see sec. 3.4 in 30]; however, buoyancy-driven flows exhibit fine-scale vorticity production, and thus greater dissipation is to be expected [31].

We next compare EPIC with a standard pseudo-spectral (PS) method [32]. The latter is a grid-based method that has been extensively used in simulations of fluid flows, in particular turbulence. It is simple and accurate, and is ideally suited to the flow model considered here (for details, see Appendix I). It uses many of the same components as EPIC, in particu-

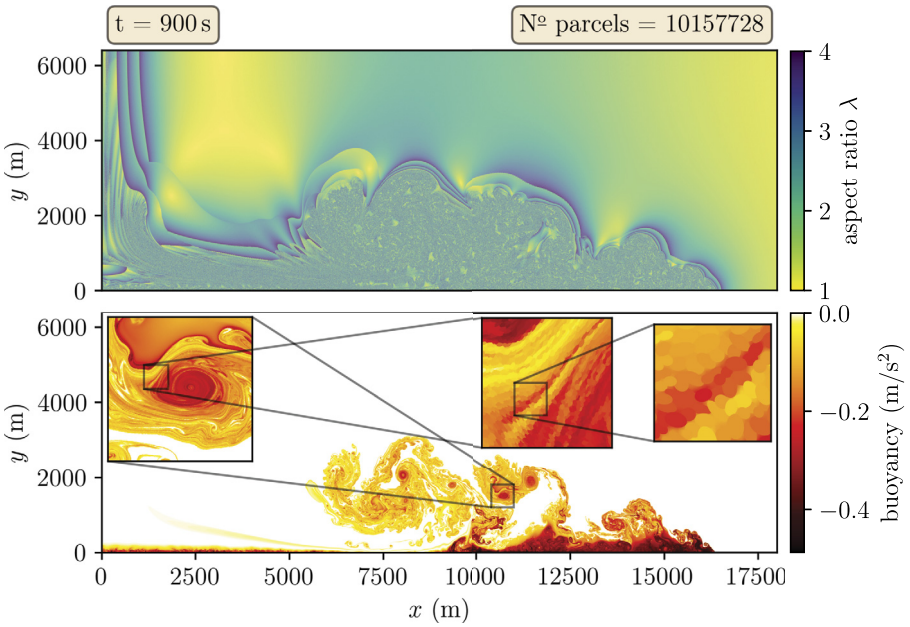

Fig. 13. Parcel configuration of the Straka density current test case (4096 × 512 grid cells) at 900 s in the subdomain $[0, 18] \times [0, 6.4]$ km. This simulation uses the default parameter values listed in Table 2. Each parcel is coloured according to its aspect ratio (above) or buoyancy (below). The square zoom windows in the lower panel are 600, 100 and 25 m wide, the last corresponding to two grid cell widths. Individual parcels can be seen in the finest zoom; they are overplotted depending on their order and thus do not give the actual buoyancy field. To make the images in the lower panel, the buoyancy *field* is first obtained on a very fine grid using the subgrid scale visualisation method described in Appendix H, then the individual parcels are plotted over this background. This ensures that any voids between the parcels are filled. Subsequent figures use subgrid scale visualisation alone, which produces a smoother image at the finest scales. Note: the parcel number indicated on this figure refers only to the subdomain shown.

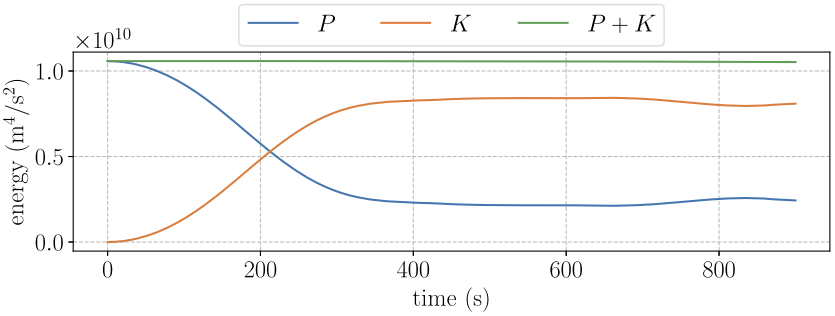

Fig. 14. Evolution of the available potential (P), kinetic (K) and total ($P + K$) energy. The results are obtained using the default parameter values listed in Table 2 and 4096 × 512 grid cells. The relative total energy loss between the initial time ($t = 0$ s) and the final time ($t = 900$ s) is 0.53%.

Table 4
Relative total energy loss between the initial time ($t = 0$ s) and the final time ($t = 900$ s) as a function of grid resolution, and for different numerical methods (see text). Doubling the grid resolution in both directions for EPIC reduces the energy loss by a factor ranging from 1.32 to 1.87.

Number of grid cells	Energy loss (%)		
	EPIC	PS-h	PS-m
256 × 32	4.51	4.88	23.55
512 × 64	2.44	2.28	18.69
1024 × 128	1.31	1.44	12.05
2048 × 256	0.70	0.64	7.07
4096 × 512	0.53	0.38	4.01

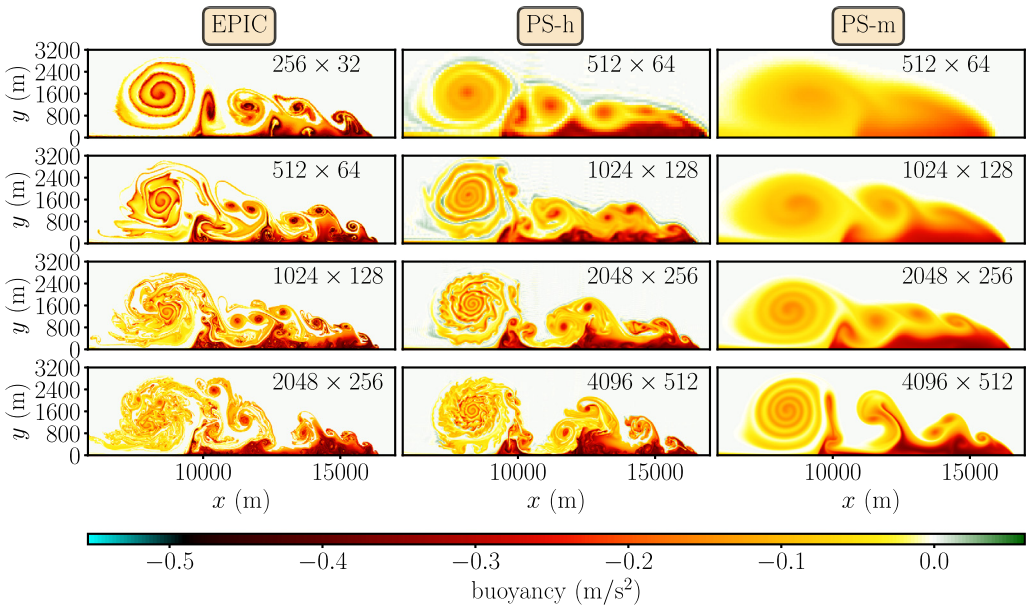

Fig. 15. Zoom of the buoyancy field b at $t = 900$ s, comparing EPIC (left column), PS with hyperviscosity (PS-h, middle column), and PS with molecular viscosity (PS-m, right column) for various grid resolutions (increasing downwards as indicated). Note, in any row, the PS simulations employ a grid which is twice finer than the EPIC one. In this test case, the exact $b_{\min} = -g\Delta T_{\max}/T_0 = -0.4905$ m/s^2, while $b_{\max} = 0$ m/s^2.

lar the same FFTs and the same time-step adaption, to facilitate comparison. For numerical stability, the PS method must obey an additional CFL constraint on the time step, and furthermore requires small-scale damping. As in many previous studies, we have implemented both hyperviscosity, $\propto \nabla^6\zeta$ and $\nabla^6 b$ on the right-hand sides of Eq. (1) and Eq. (2), and ordinary (molecular) viscosity. For both forms of damping, the resolution-dependent viscosity coefficient is tuned to minimise spurious Gibbs fringing and ensure decay in spectra at high wavenumbers (small scales) — full details may be found in Appendix I.

We start with a qualitative comparison of the methods in Fig. 15 (note that some of the details of these plots are best seen using a detailed digital zoom), which shows a portion of the buoyancy field at the final time, at various grid resolutions, and for EPIC and PS simulations using hyperviscosity and molecular viscosity (PS-h and PS-m). The EPIC fields are derived using the fine-scale visualisation method in Appendix H. Across any row, the grid resolution used in EPIC is half that (in each direction) used in PS. The EPIC fields compare well with the PS-h ones at *four* times higher resolution, while the PS-m fields are comparatively diffusive. Indeed, only the highest resolution PS-m field (bottom right) is comparable to, yet still more diffusive than, the coarsest resolution EPIC field (upper left), which uses 16 times lower resolution.

While PS-h is much less diffusive than PS-m, there is a price to pay: the extrema $b_{\min}(t)$ and $b_{\max}(t)$ do not remain within their initial bounds, and do not vary monotonically in time, as in PS-m (almost, here the diffusion is marginally sufficient) and in EPIC — see Fig. 16. Not only does hyperviscosity cause fringing, it leads to overshoots and undershoots where steep gradients develop, as is well known [33,34]. As a result, field extrema may lie well outside their initial range, here up to 50% of $b_{\max}(0) - b_{\min}(0)$. Moreover, the situation worsens for increasing resolution, and is not improved by increasing the damping rate (we have tested a 10-fold increase). This is a particularly undesirable feature in atmospheric models, as it is critical that certain fields remain monotone to prevent spurious negative tracer concentrations, e.g. water vapour.

While other advection schemes could be used to preserve monotonicity, they are typically much more expensive than the PS method used here [25]. EPIC, by construction, preserves monotonicity: buoyancy only changes when parcels merge, and the resulting buoyancy value always lies between the minimum and maximum buoyancy of the merging parcels by Eq. (49).

A *quantitative* comparison of the EPIC and PS methods is provided by examining the buoyancy power spectra, $P(k)$, the spectral variance of b as a function of total wavenumber k (the magnitude of the wavevector). Fig. 17 compares the spectra at the final time in the EPIC, PS-h and PS-m simulations across a wide range of resolutions. The strong diffusion in PS-m is especially evident, and it affects the entire range of scales. By contrast, EPIC and PS-h exhibit power-law spectra with little evidence of diffusion, and the spectra are closely comparable over shared wavenumber ranges, indicating good convergence with resolution. Overall, EPIC captures the widest range of scales for a given grid resolution, about four times wider than PS-h. And EPIC does this while remaining monotone.

Notably, Table 4 indicates that the net energy loss in the EPIC simulations is comparable to that in the PS-h simulations at the *same* grid resolution. As discussed below in Section 4.3, this is due to the fact that EPIC obtains its vorticity tendency b_x from gridded values of the buoyancy $\bar{b}$. This limits vorticity growth (though it is still higher than in PS-h as shown

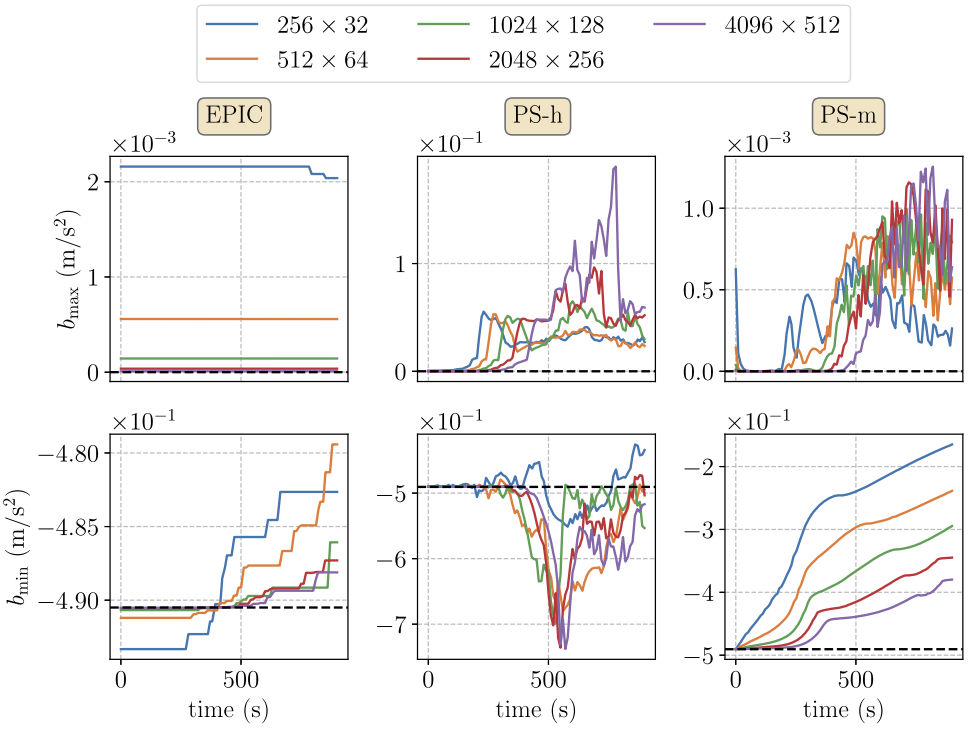

Fig. 16. Comparison of the buoyancy extrema $b_{\max}(t)$ (top row) and $b_{\min}(t)$ (bottom row) in the Straka test case, for different grid resolutions, and for different models (EPIC, PS-h and PS-m). The dashed lines indicate the exact initial values, which are theoretically conserved by Eq. (2). In EPIC, $b_{\max}(0)$ and $b_{\min}(0)$ vary with resolution because parcel attributes are initialised from gridded fields — see Section 3.1.

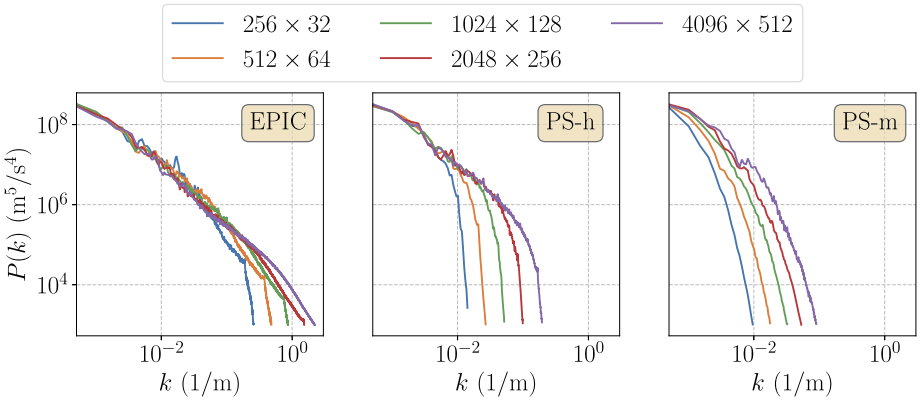

Fig. 17. Comparison of the gridded buoyancy power spectrum $P(k)$ at $t = 900$ s in the Straka test case, for different grid resolutions, and for different models (EPIC, PS-h and PS-m). The power spectrum satisfies Parseval's identity, namely $\int P(k)\,\mathrm{d}k = \iint \tilde{b}^2\,\mathrm{d}x\mathrm{d}y$. Note: the EPIC results were obtained by first interpolating the buoyancy onto a grid which is 12 times finer in each direction, using the method described in Appendix H. Also note: spectra are cut off below $P = 10^3$.

below), and reduces the kinetic energy content of the flow, since a part of the subgrid scale velocity field is unresolved in EPIC. This explains the comparable decay in total energy seen in EPIC and in PS-h. The latter method, however, achieves its level of energy conservation by creating spurious overshoots and undershoots in b and ζ. A monotone method like PS-m is far more dissipative. EPIC achieves its level of conservation *while remaining monotone*.

Regarding the numerical cost, Fig. 18 compares EPIC, PS-h and PS-m over a 16-fold variation in grid resolution. EPIC is seen to be most costly, as expected, given the relative simplicity of the PS method. However, the cost increase per doubling in resolution is less, rising by a factor of approximately 6.3, and at the highest resolution, 4096 × 512, the numerical costs of the three simulations is more closely comparable. EPIC then takes 3.4 times longer than PS-h (at 2048 × 256, EPIC takes 8.4 times longer). The steeper growth in cost in the PS method is due to the CFL stability constraint on the time step, which is not present in EPIC. This constraint limits the time step over an increasing proportion of the simulation with increasing resolution (at early times, the buoyancy frequency limits the time step, see Eq. (41)). It is evident from Fig. 16 and Fig. 17

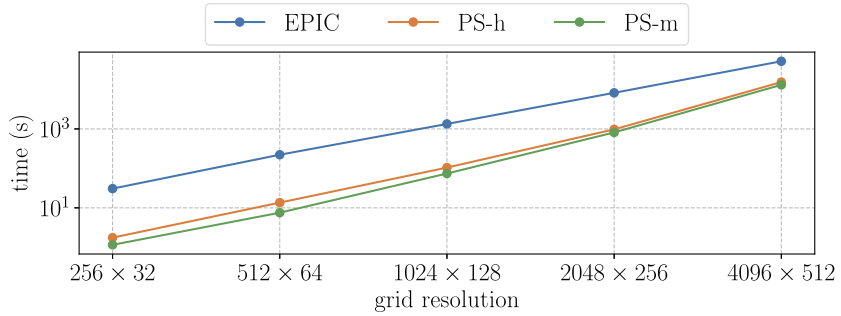

Fig. 18. Cost in CPU seconds versus resolution for EPIC, PS-h and PS-m in simulating the Straka test case.

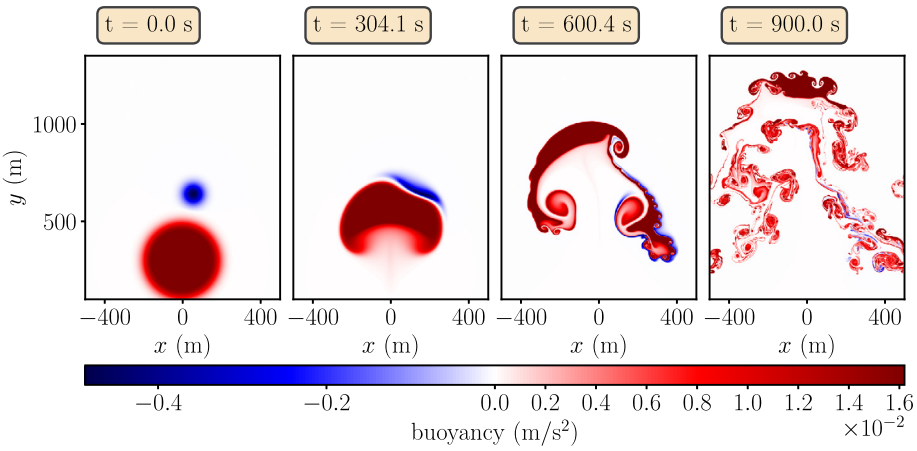

Fig. 19. Development of the buoyancy field in the Robert test case using 256×384 grid cells in EPIC. Only the vertical range $[100, 1350]$ m is shown.

that EPIC is considerably more accurate than PS at the same resolution, comparable to PS-h at 4 times resolution *and* with no fringes or overshoots, and more accurate than PS-m even at 16 times resolution. Taking this into account, EPIC can be seen to be an efficient, cost effective means to model density-stratified flows.

4.3. Robert warm bubble test case

In our final example we demonstrate EPIC with the asymmetric bubble test case introduced by [26]. This case starts with a large warm and a small cold 'bubble'. Each bubble has a potential temperature profile of

$$\Delta\theta(r) = \begin{cases} A & \text{if } r \le R, \\ A \exp\{-(r-R)^2/\sigma^2\} & \text{if } r > R \end{cases} \tag{53}$$

where $r(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x}_c\|$. The warm bubble is centred at $\boldsymbol{x}_c = (0, 300)$ m with $A = 0.5$ K, $R = 150$ m and $\sigma = 50$ m, while the small cold bubble is centred at $\boldsymbol{x}_c = (60, 640)$ m with $A = -0.15$ K, $R = 0$ m and $\sigma = 50$ m. The bubbles are contained in a box of width 1000 m and height 1500 m, whose lower left corner lies at $(-500, 0)$ m. The corresponding buoyancy used in EPIC is $b(r) = g\Delta\theta(r)/\theta_0$ with reference potential temperature $\theta_0 = 303.15$ K and gravitational acceleration $g = 9.81$ m/s^2.

The evolution of the flow to the final time of 900 s is shown in Fig. 19, which displays the buoyancy field b in an EPIC simulation using 256×384 grid cells. The two bubbles collide around $t = 300$ s, and the larger bubble draws in environmental air from below forming a characteristic mushroom shape. Thereafter the flow grows rapidly in complexity. By $t = 600$ s, the stretched and sharpened buoyancy interfaces roll up into a multitude of vortices, with many incipient ones developing. This is especially visible on the right, along the thin filament connecting the main head of the bubble with the complex structure to the lower right. By the final time ($t = 900$ s), the flow has become exceedingly complex, but the wake region also shows significant mixing with the environment and is thus losing its buoyancy.

This test case was run at various grid resolutions with initially 9 parcels per cell and the default parameter settings in Table 2. The net energy loss reduces by a factor of around 2 in EPIC when doubling the resolution as summarised in Table 5. The evolution of the kinetic and potential energy for 1024×1536 grid cells is illustrated in Fig. 20. By the end of the simulation, nearly half of the available potential energy has been converted into kinetic energy. The kinetic energy likely grows further before saturating and oscillating, as in the Straka test case (cf. Fig. 14).

Table 5
Relative total energy loss between the initial time ($t = 0$ s) and the final time ($t = 900$ s) as a function of grid cell resolution. Doubling the grid resolution in both directions for EPIC reduces the energy loss by a factor ranging from 1.88 to 2.27. The energy loss in the EPIC simulations is comparable to that in the PS-h simulations at the same grid resolution.

Number of grid cells	Energy loss (%)		
	EPIC	PS-h	PS-m
32×48	3.54	3.50	10.13
64×96	1.88	2.16	9.36
128×192	0.84	1.21	6.63
256×384	0.37	0.53	4.38
512×768	0.17	0.22	2.64
1024×1536	0.09	0.10	1.52

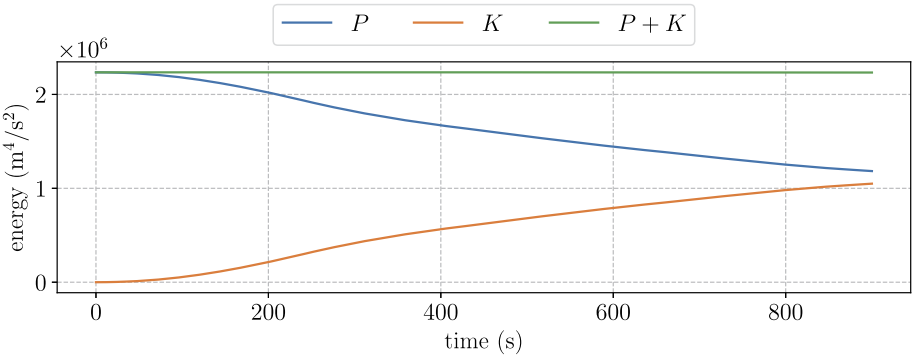

Fig. 20. Evolution of the available potential (P), kinetic (K) and total ($P + K$) energy for the Robert warm bubble case. The results are obtained using the default values listed in Table 2 and a grid of 1024×1536 cells. The relative total energy loss between the initial time ($t = 0$ s) and the final time ($t = 900$ s) is 0.09%.

Regarding accuracy and efficiency, Fig. 21 (top panel) shows that the r.m.s. area error overshoots at early times, reaching approximately 1.5×10^{-4}, before decreasing and stabilising at later times to around 10^{-4}, independent of grid resolution. This decrease occurs because the number of parcels per cell (middle panel) increases, enabling the code to better preserve incompressibility. Despite the increase in parcels, the percentage of small parcels (with $V_i < V_{\min}$, see bottom panel) in the pool of all parcels before merging remains around 5% or less.

The convergence of the spatial fields of both buoyancy and vorticity at the final time $t = 900$ s is presented in Fig. 22 and Fig. 23, comparing EPIC and pseudo-spectral simulations using hyperviscosity and ordinary viscosity, PS-h and PS-m. The height of the leading edge of the rising bubble is closely similar at all resolutions in EPIC, but this edge becomes increasingly convoluted with resolution. As expected in a flow where the production of vorticity is proportional to the buoyancy gradient, and therefore inversely proportional to the grid cell width, each increase in grid resolution will activate more intense and smaller-scale vortices. As soon as buoyancy gradients are limited by the grid resolution, results (on small scales) will differ from those generated at higher resolution. At the time shown in these figures, buoyancy gradients are limited even at the highest resolution, so results at yet higher resolution will differ, but primarily at small scales. The only way to achieve pointwise convergence is to compare an early time at which buoyancy gradients are not limited by resolution, like at $t = 304.1$ s in Fig. 19. However, this hardly challenges the numerical method.

As previously found in the Straka test, qualitatively EPIC resembles a PS-h simulation run at 4 times finer resolution (in each direction). Moreover, EPIC has none of the numerical artefacts of overshoots and undershoots seen in PS-h (see Fig. 24). PS-m is more diffusive than EPIC even when the latter uses 16 times *coarser* resolution in each direction. Notably however, vorticity extrema in EPIC and PS-h are comparable at the same grid resolution. This is due to the fact that EPIC computes the vorticity tendency, b_x, from the gridded buoyancy field $\bar{b}$. Thus, the growth in vorticity is limited ultimately by the inverse of the grid spacing. Yet, what matters to the velocity field is the circulation deposition, effectively the area-integrated vorticity tendency, and this only requires an accurate gridded buoyancy field. Note, the streamfunction $\psi(\boldsymbol{x}, t)$ is essentially the (spatial) sum of the local circulation $\zeta(\boldsymbol{x}', t)\mathrm{d}x\mathrm{d}y$ multiplied by the appropriate Green function $G(\boldsymbol{x}, \boldsymbol{x}')$ for the domain. In EPIC, the main limitation is that the subgrid scale velocity field is crudely represented — it varies only linearly in each coordinate, holding the other fixed, by Eq. (17). Nonetheless, the missing part of the subgrid scale velocity field is much weaker than the resolved field, and so parcel advection is accurate, indeed significantly more accurate than the finite grid resolution would suggest. The dominance of the resolved velocity field in advection is key to the accuracy of Lagrangian-based methods, see e.g. [18,28,35] and references therein.

As in the Straka test case, it is instructive to study how the numerical results converge across scale by examining the buoyancy power spectrum $P(k)$, and again comparing with PS simulations using hyperviscosity and molecular viscosity,

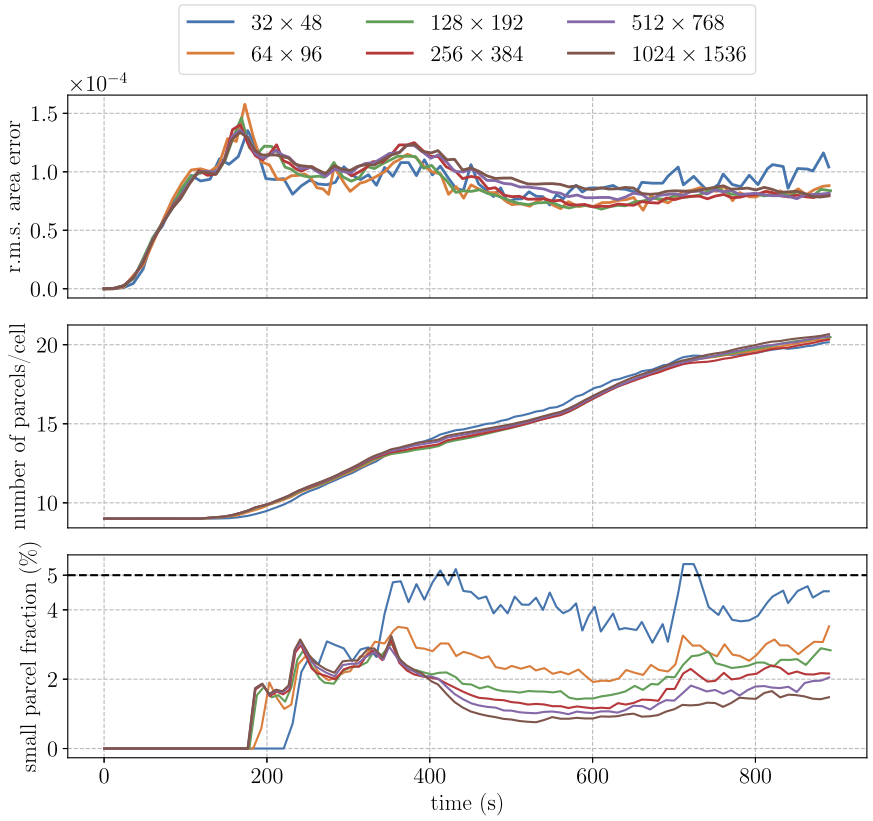

Fig. 21. Evolution of the gridded relative r.m.s. area error (top), average number of parcels per grid cell (middle), and the percentage of small parcels (bottom) for the asymmetric Robert test case at different grid resolutions.

PS-h and PS-m. This is done in Fig. 25 for the final time, and across a wide range of resolutions. Again, PS-m exhibits strong diffusive effects and poor convergence. PS-h and EPIC are much less diffusive and show stronger convergence. Again, EPIC captures the widest scale range without noticeable effects of diffusion, about 4 times wider than PS-h.

The cost of the simulations over a wide range of resolutions is shown in Fig. 26, comparing EPIC with PS-h and PS-m. Again, EPIC costs more at any given resolution, but considering it is also most accurate, perhaps 4 times more accurate than PS-h (and without the associated numerical artefacts), EPIC is seen to be highly cost effective. The cost increase in EPIC per doubling in resolution is approximately 6.5, while it is more than 8 in PS-h and PS-m. Closely similar results were found in the Straka test case, see Fig. 18.

We finish by examining the mixing properties of EPIC for this case. For this, we form a histogram of parcel buoyancy to see how the initial buoyancy values mix as a result of parcel splitting and merging (the only source of mixing in EPIC). The histogram is created by summing parcel volumes V_i into buoyancy bins (like when forming a PDF), then normalising so that the area under the histogram is 1 (without loss of generality). The results are shown in Fig. 27, for the initial conditions (top panel), the final configuration (middle panel), and the difference between the final and the initial configurations (bottom panel) — for all six grid resolutions considered. The large peak at $b = 0$ corresponds to the neutrally buoyant background (here most of the domain), while the secondary peak near $b = 0.0163$ corresponds to the uniform buoyancy inside the large bubble. The broader distribution of the histogram seen at low resolution is a consequence of our initialisation of parcel buoyancy values from a given gridded field. The differences shown in the bottom panel indicate the net mixing that has taken place from the initial to the final time. We see that the middle range of positive buoyancy values has filled in, while the lower negative range of buoyancy has been depleted. The latter is a consequence of the intense stretching and folding of the cold bubble, seen e.g. in Fig. 19 (right panel). Similarly, high buoyancy values mix with lower values or the environment, especially around the sharp interfaces which develop as a result of material stretching.

An especially notable feature is the strong convergence displayed by even the difference histograms with increasing grid resolution. This is promising, as it shows that the mixing of buoyancy, here a material tracer, is well modelled by parcel splitting and merging in EPIC. Such strong convergence was not obtained previously using point parcels in MPIC [10], which used an *ad hoc*, implicit model of stretching (since point parcels cannot explicitly deform). It appears, therefore, that the ability to *explicitly* deform parcels in EPIC is key to properly modelling small-scale mixing. Stretching is accurately modelled by following the shape of each parcel, and a single criterion is used for parcel splitting (a high aspect ratio). Parcel merging is also handled differently in EPIC compared to MPIC. The latter method removes small parcels by spreading their

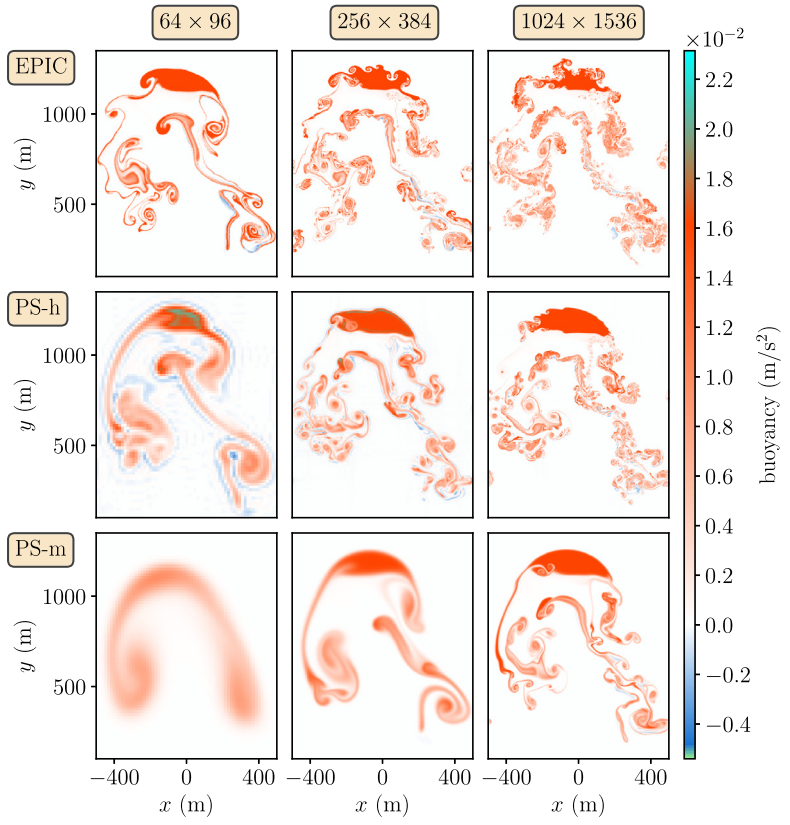

Fig. 22. Buoyancy at 900 s for various resolutions, and for EPIC, PS-h and PS-m, as indicated. Only the vertical range [100, 1350] m is shown. In this test case, the exact $b_{\min} = -0.4854033 \times 10^{-2}$ m/s^2, while $b_{\max} = 1.6180109 \times 10^{-2}$ m/s^2.

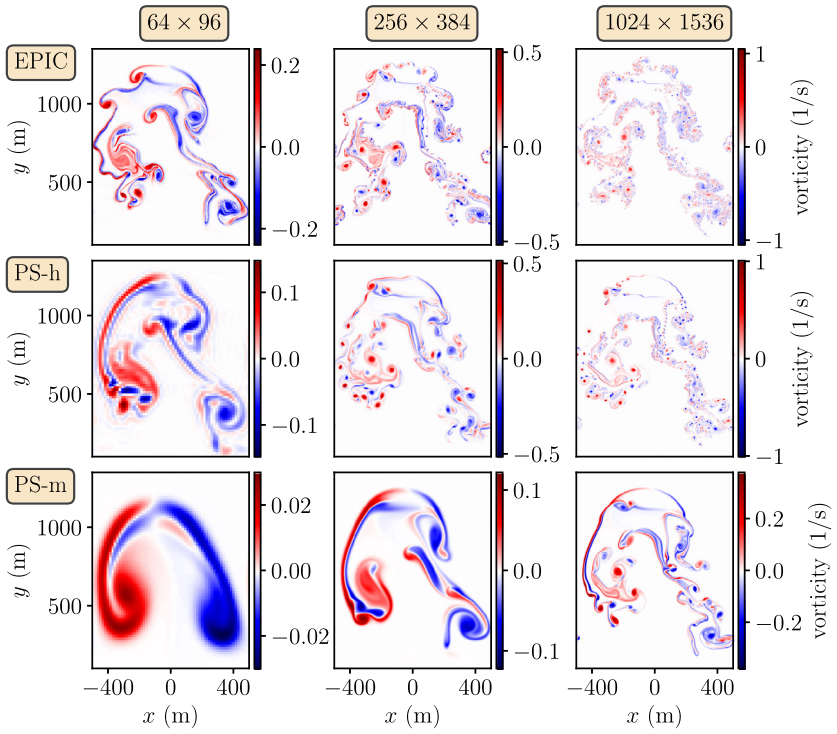

Fig. 23. Vorticity at 900 s for various resolutions, and for EPIC, PS-h and PS-m, as indicated. Only the vertical range [100, 1350] m is shown.

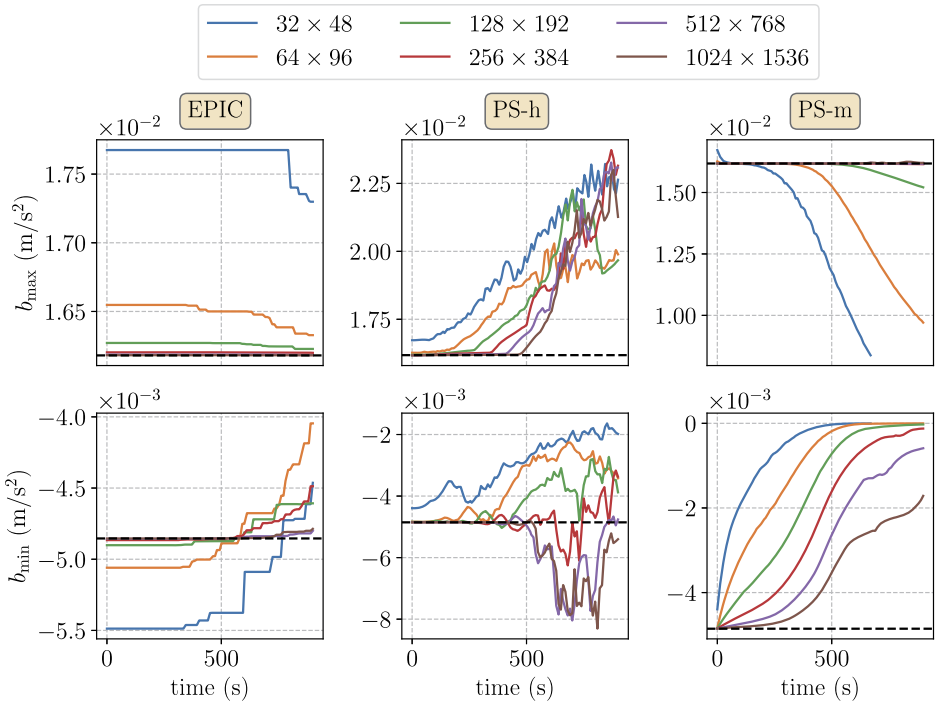

Fig. 24. Comparison of the buoyancy extrema $b_{\max}(t)$ (top row) and $b_{\min}(t)$ (bottom row) in the Robert test case, for different grid resolutions, and for different models (EPIC, PS-h and PS-m). The dashed lines indicate the exact initial values, which are theoretically conserved by Eq. (2). In EPIC, $b_{\max}(0)$ and $b_{\min}(0)$ vary with resolution because parcel attributes are initialised from gridded fields — see Section 3.1.

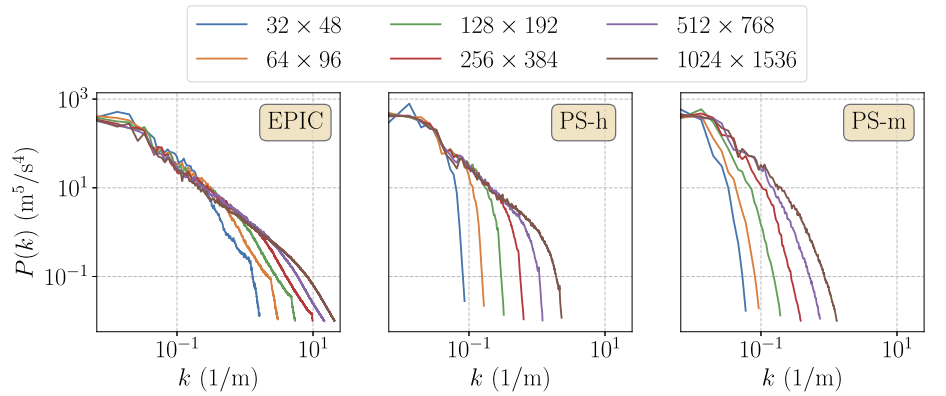

Fig. 25. Comparison of the gridded buoyancy power spectrum $P(k)$ at $t = 900$ s in the Robert test case, for different grid resolutions, and for different models (EPIC, PS-h and PS-m). Note: spectra are cut off below $P = 0.01$. For comparison with the Straka test case, see Fig. 17.

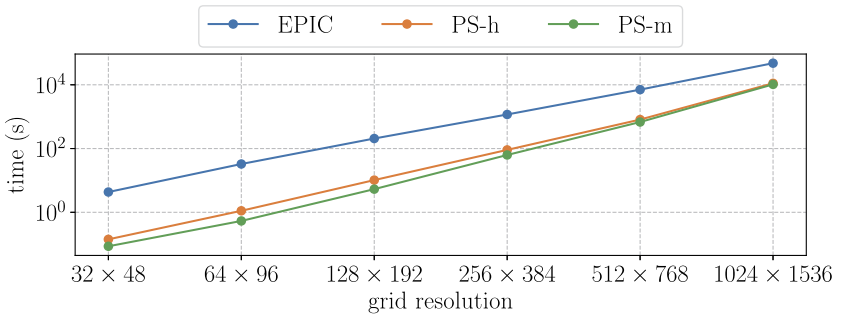

Fig. 26. Cost in CPU seconds versus resolution for EPIC, PS-h and PS-m in simulating the Robert test case.

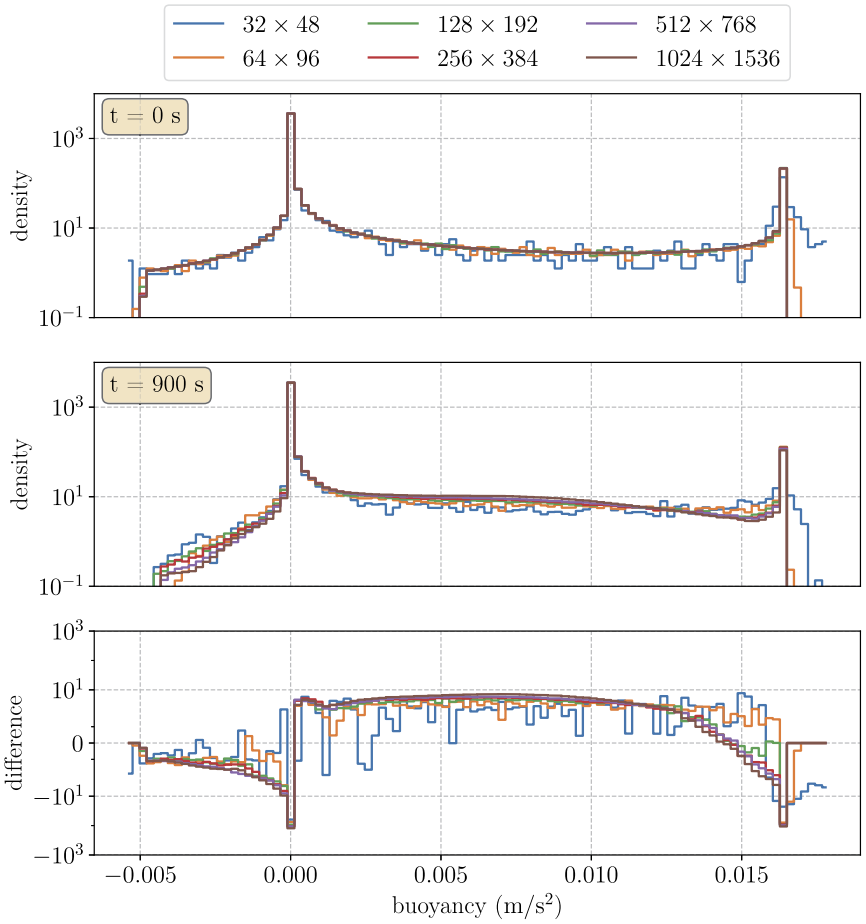

Fig. 27. Volume-weighted buoyancy histograms of the Robert test case at times $t = 0$ s (top panel) and $t = 900$ s (middle panel), together with the difference (final minus initial, bottom panel) for various grid resolutions. The bin width is 2.32×10^{-4} m/s^2. Each histogram is normalised such that its area sums to 1.

properties to the corners of the grid cell they occupy; these properties are then shared among all remaining parcels. In EPIC, by contrast, parcels are merged with the closest other parcel, an operation which is much more local and independent of the grid. Arguably, it is also a more natural approach, and leads to the excellent mixing properties observed in Fig. 27.

5. Discussion

In this paper, we have formulated and tested a new, robust and efficient parcel-in-cell based approach for the simulation of two-dimensional flows and related physical systems. The approach, called EPIC after 'Elliptical Parcel-In-Cell', makes use of deformable elliptical parcels as in [5], but differs in several fundamental ways. These include (1) ensuring that the parcels are space-filling (generally overlapping), (2) simplifying their interpolation using a pair of representative 'support points', (3) splitting highly-deformed parcels in a way that conserves all spatial moments up to second order, (4) merging very small parcels with larger ones in a more geometric way, and (5) automatically correcting for errors in incompressibility without the need to ever re-grid the parcels. Other details also differ.

Three different test flows were used to quantify accuracy, measure convergence and investigate the dependence on numerical parameters. These tests demonstrate that complex flows exhibiting sharp variations and extreme values can be handled efficiently and robustly with the new EPIC method. Moreover, we have compared results obtained with a standard pseudo-spectral method, widely used in fluid dynamical modelling, and have shown that EPIC offers a highly cost-effective alternative. During the development stage, we have exhaustively tested each component individually in a wide variety of 'unit tests'. We have ensured that steady flows like Taylor-Green remain very close to steady for long times. We have ensured that incompressible flows remain area preserving, a known challenge for parcel-based methods [4,5,10,20]. Finally, we have also ensured that the EPIC code database is easy to use and is readily adaptable to other problems (see 'code availability' below for access).

The many tests conducted helped to pin down choices for the numerical parameters in Table 2. The choice of each parameter has an impact on both accuracy and efficiency. We have sought a balance. It makes no sense to use a tiny

minimum area fraction $\tilde{V}_{\min}$ because this does not improve the subgrid representation without an associated subgrid-scale velocity field; moreover it is computationally expensive and takes much memory. Likewise, the time step pre-factor α need not be very small since the greater accuracy in time stepping is then swamped by other errors — like simply using a finite spatial resolution. The recommended parameters in Table 2 do not appear to be dependent on the grid resolution (not listed in the table). At higher resolution, we simply use more parcels. The time step will be shorter as the velocity strain and buoyancy gradients will naturally increase.

Note, the recommended settings need not be strictly adhered to: there is some liberty in the values. We advise against changing β and $C_{\max}$ in particular. Making $\lambda_{\max}$ smaller may be more accurate but will lead to many more parcels, so the code will be less efficient. Likewise, α could be reduced if higher temporal accuracy is needed for some reason. The values of $\tilde{V}_{\min}$ and $\tilde{V}_{\max}$ can be adjusted, but extensive tests indicate that the recommended values enable an adequate representation of subgrid-scale variations.

An improvement we hope to make in the near future concerns small, inactive parcels left over following a period of strong stretching. In the density-stratified flows examined here, a descending patch of denser fluid or a rising patch of lighter fluid leaves a turbulent wake which decays in time through entrainment and mixing. This may leave many small parcels in nearly stagnant regions with low strain. Numerically, these small parcels over-represent the flow, and moreover increase the computational burden and memory footprint. A possible remedy is to make the minimum parcel area fraction $\tilde{V}_{\min}$ depend on certain flow properties. For example, $\tilde{V}_{\min}$ could increase as the stretching rate γ, buoyancy gradient $|\nabla b|$ and/or vorticity gradient $|\nabla\omega|$ decrease.

The EPIC method, developed here for incompressible density-stratified flows under the Boussinesq approximation, could be extended to much more diverse physical systems. The core aspect of the method, namely representing a continuum by a space-filling set of elliptical parcels, does not depend crucially on the physical system modelled so long as it is advection dominated, with diffusion playing a minor role except perhaps in subgrid-scale turbulent mixing. Even if diffusion is important for certain quantities but not for all quantities, as in magneto-hydrodynamic turbulence at small magnetic Prandtl number [35], a hybrid EPIC method could be used. This hybrid method would evolve diffused quantities on a regular grid using conventional numerical methods, and use parcels to evolve the remaining advection-dominated quantities (this would be analogous to the 'Contour Advection' method used in [35]).

A similar strategy may be applied to systems involving a mix of predominantly large-scale waves and advection-dominated quantities, as in rotating shallow-water turbulence [36]. The latter is an example of a compressible (2D) flow. In EPIC, parcel areas would generally change in time, meaning that all three distinct components of the shape matrix (B_{11}, B_{12} and B_{22}) would have to be evolved independently. This is however straightforward. We would still require that the parcel interpolated area $\tilde{V}$ at each grid point is as close as possible to the grid cell area V_{cell}. The algorithm to do this is exactly the same as we have developed for incompressible EPIC.

The original motivation for developing EPIC was to improve the modelling of cloud growth and decay in the atmosphere, following on from the moist parcel-in-cell (MPIC) method [4,10]. In fact, the 2D model developed here optionally includes moisture to allow for condensation and evaporation, and the associated latent heat exchanges. Our next step is to generalise EPIC to three dimensions, to properly account for turbulent processes like vortex stretching. Like in MPIC, each parcel will now carry a *vector* vorticity which evolves not only because of buoyancy gradients (baroclinic processes), but also because of stretching. Otherwise, the attributes carried on parcels are the same as in the 2D model. Moreover, many of the numerical methods carry over, largely unchanged, to the 3D model. The exception is the shape matrix $\boldsymbol{B}$, which now has 5 independent components (in an incompressible flow) rather than 2. The evolution of $\boldsymbol{B}$ is however formally identical. The interpolation in 3D will use 4 support points lying in the major-middle axis plane of the ellipsoid (rather than 2). These support points are the analogues of Gaussian quadrature points for volume integration [17]. Their location depends on obtaining the eigenvalues and eigenvectors of $\boldsymbol{B}$, which is less straightforward than in 2D but is nonetheless efficient. A 3D version will also incorporate distributed memory parallelism. Most of the functionality required for parallelism has been previously implemented for the Moist Parcel-in-Cell method [37], but some adjustments will be needed because the 4 parcel support points can be in different grid cells. The new merging algorithm (see Appendix D) will also require local parallel communication. We hope to report on the 3D version of EPIC in the near future.

CRediT authorship contribution statement

Matthias Frey: Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **David Dritschel:** Conceptualization, Formal analysis, Funding acquisition, Investigation, Methodology, Software, Supervision, Writing – original draft, Writing – review & editing. **Steven Böing:** Conceptualization, Formal analysis, Funding acquisition, Investigation, Methodology, Software, Visualization, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Code availability

The source code of EPIC is publicly available and version 0.10.8 can be downloaded from [38]. The pseudo-spectral source code version 0.0.4 can be downloaded from [39]. The static Taylor-Green vortex test case of Appendix F was performed with a special EPIC code version available from [40].

Third party libraries

In order to post-process the data and generate the figures in this publication we heavily used the following Python libraries: Matplotlib (version 3.5.1) [41], colorcet (version 3.0.0) [42,43], h5py (version 3.6.0) [44], NumPy (version 1.21.2) [45], pandas (version 1.3.5) [46,47] and SciPy (version 1.7.3) [48].

Acknowledgements

The authors thank Doug Parker for comments that substantially improved the manuscript. This research is supported by the UK Engineering and Physical Sciences Research Council (grant numbers EP/T025301/1 and EP/T025409/1).

Appendix A. Inverting vorticity for the velocity field: numerics

Here we discuss the numerical procedure used for determining the velocity field $\boldsymbol{u} = (u, v)$ from the vorticity field ζ in an x-periodic domain bounded between free-slip walls at $y = y_{\min}$ and $y = y_{\max}$. As discussed in Section 2.2, the procedure is facilitated by working in (semi-)spectral space, after a Fourier transform in x. The finite domain length in x gives rise to a set of discrete wavenumbers $k \in \{0,\ \Delta k,\ 2\Delta k,\ \ldots,\ n_x \Delta k\}$ where $\Delta k = 2\pi/L_x$, $L_x = x_{\max} - x_{\min}$ is the domain width and n_x is the number of x grid points (counting the periodic edge only once). The Fourier transform of $\psi(x, y)$ (here time t is suppressed) yields $\hat{\psi}^k(y)$ for each discrete k.

The first step is to Fourier transform the gridded vorticity field (obtained by `par2grid`). The x-independent mode $\hat{\zeta}^0(y)$ is then used to compute the x-independent horizontal velocity component $\hat{u}^0(y)$, making use of the relation $\hat{\zeta}^0(y) = -\mathrm{d}\hat{u}^0/\mathrm{d}y$. Numerically, we integrate this using the trapezoidal rule to obtain $\hat{u}^0(y)$ on a set of $n_y + 1$ grid points in y which are equally spaced between $y_{\min}$ and $y_{\max}$. Indexing the y grid points from 0 to n_y, we start with $\hat{u}^0_0 = 0$ and determine the remaining values from

$$\hat{u}^0_j = \hat{u}^0_{j-1} - \frac{\Delta y}{2}\left(\hat{\zeta}^0_j + \hat{\zeta}^0_{j-1}\right), \qquad j = 1,\ \ldots,\ n_y\,. \tag{A.1}$$

We next adjust all values to a prescribed domain mean value $\langle u \rangle$ by adding a constant C. The domain mean value is constant in time due to momentum conservation. The latter follows from the fact that the x momentum equation can be written in flux form as

$$u_t = -(u^2 + p/\rho_0)_x - (uv)_y \tag{A.2}$$

due to incompressibility $u_x + v_y = 0$ (here p is the pressure). Integration over the domain and using $v = 0$ on the y boundaries implies that $\langle u_t \rangle = 0$, or $\langle u \rangle$ is a constant. Numerically, the constant C required to enforce momentum conservation is found by summing the tentative values of $\hat{u}^0_j$ (again using the trapezoidal rule), and subtracting this from the prescribed value of $\langle u \rangle$.

The remaining Fourier modes with $k > 0$ are handled differently. Instead, we solve a Poisson equation, $\nabla^2 \psi = \zeta$, for the streamfunction ψ. In spectral space, this becomes

$$\frac{\mathrm{d}^2 \hat{\psi}^k}{\mathrm{d}y^2} - k^2 \hat{\psi}^k = \hat{\zeta}^k \qquad (k > 0), \tag{A.3}$$

and the boundary conditions $v = \psi_x = 0$ imply $\hat{\psi}^k = 0$ at $y = y_{\min}$ and $y_{\max}$. Numerically, this is solved by fourth-order compact differences, following [49] and [50]. We discretise Eq. (A.3) as follows

$$\frac{\hat{\psi}^k_{j+1} - 2\hat{\psi}^k_j + \hat{\psi}^k_{j-1}}{(\Delta y)^2} - \frac{k^2}{12}\hat{\psi}^k_{j+1} - \frac{5k^2}{6}\hat{\psi}^k_j - \frac{k^2}{12}\hat{\psi}^k_{j-1} = \frac{1}{12}\hat{\zeta}^k_{j+1} + \frac{5}{6}\hat{\zeta}^k_j + \frac{1}{12}\hat{\zeta}^k_{j-1} \tag{A.4}$$

for $j = 1,\ \ldots,\ n_y - 1$. Using the boundary conditions $\hat{\psi}^k_0 = \hat{\psi}^k_{n_y} = 0$ results in a simple tri-diagonal problem which is solved in $\mathcal{O}(n_y)$ operations. This provides $\hat{\psi}^k_j$ at all y grid points. Differentiation in spectral space (by simple wavenumber multiplication) then gives $\hat{v}^k_j$. On the other hand, the horizontal velocity component $\hat{u}^k = -\mathrm{d}\hat{\psi}^k/\mathrm{d}y$ must be found by numerical differentiation. Again we use fourth-order compact differences and solve the tri-diagonal problem

$$\frac{1}{6}\hat{u}^k_{j+1} + \frac{2}{3}\hat{u}^k_j + \frac{1}{6}\hat{u}^k_{j-1} = \frac{\hat{\psi}^k_{j-1} - \hat{\psi}^k_{j+1}}{2\Delta y} \tag{A.5}$$

for $j = 1, \ldots, n_y - 1$, supplemented by the one-sided compact difference relations at the y boundaries [50]:

$$\frac{1}{3}\hat{u}^k_1 + \frac{2}{3}\hat{u}^k_0 = \frac{\Delta y}{6}\hat{\zeta}^k_0 - \frac{\hat{\psi}^k_1}{\Delta y} \tag{A.6}$$

$$\frac{2}{3}\hat{u}^k_{n_y} + \frac{1}{3}\hat{u}^k_{n_y-1} = \frac{\hat{\psi}^k_{n_y-1}}{\Delta y} - \frac{\Delta y}{6}\hat{\zeta}^k_{n_y}\,. \tag{A.7}$$

Finally, inverse FFTs are performed to recover u and v in physical space (on the grid). This completes the inversion problem.

Tests starting from a non-trivial exact solution confirm that errors in the vertical velocity field v scale like n_x^{-3} (taking $n_y \propto n_x$). This scaling is also found for the horizontal velocity component u when the x-independent component of the vorticity $\hat{\zeta}^0(y) = 0$; otherwise the scaling is n_x^{-2} due to limitations imposed by the trapezoidal rule (generally, one cannot assume any symmetry properties for $\hat{\zeta}^0(y)$ across the y boundaries). The n_x^{-3} scaling for v and for u when $\hat{\zeta}^0(y) = 0$ is the best one can do for free-slip boundaries when fourth-order compact differencing is used [50].

Appendix B. The shape matrix $\boldsymbol{B}$: evolution and eigenvalues

In this section we derive equations Eqs. (8) to (13) given in Section 2.3. We first derive Eq. (8). Taking the time derivative of the general definition of an ellipse, Eq. (5), and inserting Eq. (7) yields

$$\frac{\mathrm{d}}{\mathrm{d}t}(\boldsymbol{x}^\mathsf{T}\boldsymbol{B}^{-1}\boldsymbol{x}) = 0 \tag{B.1}$$

$$\Rightarrow \quad \frac{\mathrm{d}\boldsymbol{x}^\mathsf{T}}{\mathrm{d}t}\boldsymbol{B}^{-1}\boldsymbol{x} + \boldsymbol{x}^\mathsf{T}\frac{\mathrm{d}\boldsymbol{B}^{-1}}{\mathrm{d}t}\boldsymbol{x} + \boldsymbol{x}^\mathsf{T}\boldsymbol{B}^{-1}\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = 0 \tag{B.2}$$

$$\Rightarrow \quad \boldsymbol{x}^\mathsf{T}\boldsymbol{S}^\mathsf{T}\boldsymbol{B}^{-1}\boldsymbol{x} + \boldsymbol{x}^\mathsf{T}\frac{\mathrm{d}\boldsymbol{B}^{-1}}{\mathrm{d}t}\boldsymbol{x} + \boldsymbol{x}^\mathsf{T}\boldsymbol{B}^{-1}\boldsymbol{S}\boldsymbol{x} = 0 \tag{B.3}$$

$$\Rightarrow \quad \boldsymbol{S}^\mathsf{T}\boldsymbol{B}^{-1} + \frac{\mathrm{d}\boldsymbol{B}^{-1}}{\mathrm{d}t} + \boldsymbol{B}^{-1}\boldsymbol{S} = 0\,. \tag{B.4}$$

The evolution of $\boldsymbol{B}^{-1}$ is therefore given by

$$\frac{\mathrm{d}\boldsymbol{B}^{-1}}{\mathrm{d}t} = -(\boldsymbol{S}^\mathsf{T}\boldsymbol{B}^{-1} + \boldsymbol{B}^{-1}\boldsymbol{S})\,. \tag{B.5}$$

Multiplying with $\boldsymbol{B}$ from the left and right side and making use of the relation

$$\frac{\mathrm{d}(\boldsymbol{B}^{-1}\boldsymbol{B})}{\mathrm{d}t} = \frac{\mathrm{d}\boldsymbol{B}^{-1}}{\mathrm{d}t}\boldsymbol{B} + \boldsymbol{B}^{-1}\frac{\mathrm{d}\boldsymbol{B}}{\mathrm{d}t} = 0 \Longleftrightarrow \frac{\mathrm{d}\boldsymbol{B}}{\mathrm{d}t} = -\boldsymbol{B}\frac{\mathrm{d}\boldsymbol{B}^{-1}}{\mathrm{d}t}\boldsymbol{B} \tag{B.6}$$

results in

$$\boldsymbol{B}\frac{\mathrm{d}\boldsymbol{B}^{-1}}{\mathrm{d}t}\boldsymbol{B} = -(\boldsymbol{B}\boldsymbol{S}^\mathsf{T} + \boldsymbol{S}\boldsymbol{B}) \tag{B.7}$$

$$\frac{\mathrm{d}\boldsymbol{B}}{\mathrm{d}t} = \boldsymbol{B}\boldsymbol{S}^\mathsf{T} + \boldsymbol{S}\boldsymbol{B} \tag{B.8}$$

which is exactly Eq. (8).

The eigenvalues υ of $\boldsymbol{B}$ determine the squared semi-major and semi-minor axes a^2 and b^2. These are obtained by solving

$$(B_{11} - \upsilon)(B_{22} - \upsilon) - B_{12}^2 = 0 \tag{B.9}$$

$$\text{or} \quad \upsilon^2 - (B_{11} + B_{22})\upsilon + B_{11}B_{22} - B_{12}^2 = 0 \tag{B.10}$$

giving the two roots

$$\upsilon_\pm = \frac{1}{2}(B_{11} + B_{22}) \pm \sqrt{\frac{1}{4}(B_{11} + B_{22})^2 - B_{11}B_{22} + B_{12}^2}\,. \tag{B.11}$$

Since

$$\frac{1}{4}(B_{11} + B_{22})^2 - B_{11}B_{22} + B_{12}^2 = \frac{1}{4}(B_{11} - B_{22})^2 + B_{12}^2, \tag{B.12}$$

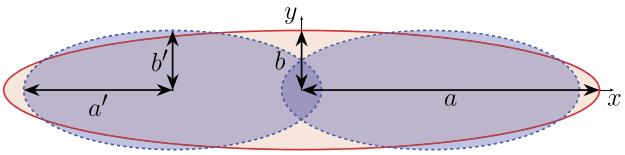

Fig. C.28. An elliptical parcel centred at the origin with major axis along the x axis before splitting. The dashed ellipses represent the child parcels obtained after splitting.

we may simplify the eigenvalues to

$$\upsilon_\pm = \frac{1}{2}(B_{11}+B_{22}) \pm \sqrt{\frac{1}{4}(B_{11}-B_{22})^2 + B_{12}^2} \tag{B.13}$$

where $a^2 = \upsilon_+$ and $b^2 = \upsilon_-$ (as we require $a^2 \geq b^2$).

The eigenvector along the major axis of the ellipse is given by $\hat{\boldsymbol{a}} = (\hat{a}_x, \hat{a}_y) \equiv (\cos\varphi, \sin\varphi)$, the (normalised) solution to

$$\begin{pmatrix} B_{11}-a^2 & B_{12} \\ B_{12} & B_{22}-a^2 \end{pmatrix} \begin{pmatrix} \hat{a}_x \\ \hat{a}_y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}. \tag{B.14}$$

Both rows are equivalent; using the second gives Eq. (12). Under the assumption that $a \geq b$, the semi-focal length c as given in Eq. (11) is obtained by inserting Eq. (B.13) into $c^2 = a^2 - b^2$, yielding

$$c^2 = \upsilon_+ - \upsilon_- = 2\sqrt{\frac{1}{4}(B_{11}-B_{22})^2 + B_{12}^2} = 2a^2 - B_{11} - B_{22} \tag{B.15}$$

$$\Rightarrow \quad c = \sqrt{2a^2 - B_{11} - B_{22}}. \tag{B.16}$$

Appendix C. Parcel splitting

To derive the positions and centres of the two identical 'child' parcels that result from splitting, we orientate without loss of generality the major axis of the 'parent' parcel along the x axis, so that its minor axis is along the y axis (cf. Fig. C.28). After splitting, the child parcels must each have half of the area, the same total centroid and the same total second-order spatial moments as the parent parcel. The child parcels are therefore centred on the x axis at $\boldsymbol{x}' = \boldsymbol{x} \pm (h, 0)$, with h to be determined by equating the second-order moments. The moments for the parent parcel are given by

$$\int_\Omega x^2 \mathrm{d}x\mathrm{d}y = a^3 b \int_0^{2\pi}\int_0^1 R^3 \cos^2\theta \mathrm{d}R\mathrm{d}\theta = \frac{1}{4}\pi a^3 b, \tag{C.1}$$

$$\int_\Omega y^2 \mathrm{d}x\mathrm{d}y = a^3 b \int_0^{2\pi}\int_0^1 R^3 \sin^2\theta \mathrm{d}R\mathrm{d}\theta = \frac{1}{4}\pi a b^3, \tag{C.2}$$

$$\int_\Omega xy \mathrm{d}x\mathrm{d}y = 0 \tag{C.3}$$

where Ω denotes the area of integration (here we have used elliptical polar coordinates (R, θ) in which $x = aR\cos\theta$ and $y = bR\sin\theta$). The last integral is zero due to symmetry. Assuming the child parcels have the same shape, we can evaluate the moments for just one of them and double the result:

$$\int_{\Omega'} x^2 \mathrm{d}x\mathrm{d}y = 2a'b' \int_0^{2\pi}\int_0^1 (h + a'R\cos\theta)^2 R\mathrm{d}R\mathrm{d}\theta = 2a'b'\left(\pi h^2 + \frac{\pi}{4}a'^2\right), \tag{C.4}$$

$$\int_{\Omega'} y^2 \mathrm{d}x\mathrm{d}y = 2a'b'^3 \int_0^{2\pi}\int_0^1 R^3 \sin^2\theta \mathrm{d}R\mathrm{d}\theta = \frac{1}{2}\pi a' b'^3, \tag{C.5}$$

$$\int_{\Omega'} xy \mathrm{d}x\mathrm{d}y = 0 \tag{C.6}$$

where Ω' denotes the area of the child parcels and $a'b' = ab/2$. Equating Eq. (C.2) and Eq. (C.5) gives $b' = b$. Hence, equal area requires $a' = a/2$. The distance h is obtained by equating Eq. (C.1) and Eq. (C.4); this yields $h = \sqrt{3}a/4$.

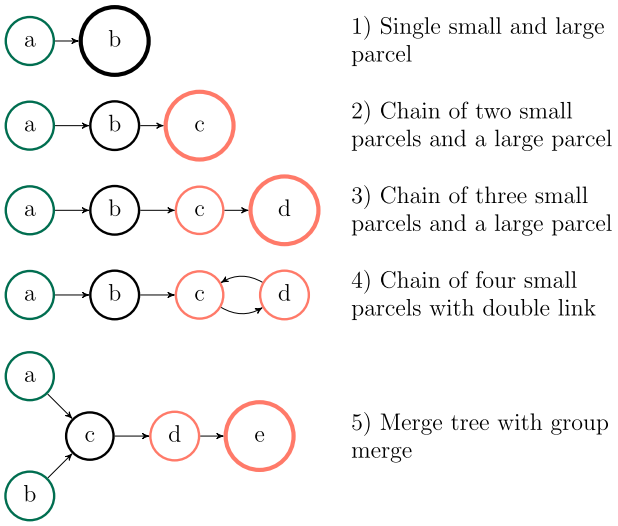

Fig. D.29. Examples of parcel merging chains and a tree.

Appendix D. Parcel merging

D.1. Enforcing the merger rules

In this section, the algorithm for merging parcels is discussed in detail. We start by showing a number of parcel configurations of increasing complexity and introducing the concepts of *leaf parcels*, *unavailable parcels* and *double links*. Subsequently, a full example of the merging algorithm with several iterations is given. The merging algorithm ensures all small parcels are merged, and consists of an iterative stage, and a second stage that deals with small parcels that point to each other (i.e. they are each other's nearest neighbour).

Fig. D.29 shows five different parcel configurations: the arrows in the diagrams refer to parcels pointing to other parcels.

1. The first example is a small parcel pointing to a large parcel, with no other parcels involved. In this example, the algorithm will merge the two parcels. The small parcel is labelled in green to indicate that it is a so-called *leaf parcel*. This terminology comes from graph theory, and more specifically the study of trees. A tree is a connected graph without cycles. In our case, a leaf parcel is defined as a small parcel that does not have any parcels pointing to it. In the iterative approach, we ignore merged parcels after each iteration, which means a parcel can become a leaf parcel in a later iteration (after a parcel that was pointing to it is merged).
2. The second example is a chain, where an additional intermediate small parcel is added. In this case, we merge small parcels a and b, and ignore parcel c as it is a large parcel. In our algorithm, every parcel that has an (unmerged) non-leaf parcel pointing at it, such as parcel c which has parcel b pointing at it, will be marked as *unavailable* for merging in the current iteration. Unavailable parcels are identified only after all leaf parcels have been determined.
3. In the third example, another intermediate parcel is introduced. In this case, we merge parcels a and b in the first iteration, where parcels c and d are marked as unavailable. Parcels c and d are merged in a second iteration. An example of the merging procedure with several iterations is given later in this section.
4. The fourth example is similar, but involves a pair of double-linked parcels instead of a small and large parcel. Again, a and b are merged in the first iteration. A special procedure that deals with double-linked parcels is introduced later in this section. Double-linked parcels cannot become leaf parcels during the iterative stage, as they will always continue to have the incoming link from the other double-linked parcel. For the same reason, they will remain marked as unavailable.
5. The fifth example shows a group merge, where two parcels (a and b) point to the same parcel (c). Any parcel configuration without a double link can be described as a tree structure, where the edges are all directed towards the large parcel. In graph theory, this is known as an *in-tree*. Here, the parcels a and b merge with c. The parcel d merges with parcel e.

A more complex example of a configuration having a double link is given in Fig. D.30. A double link introduces a cycle into the graph, which therefore no longer is a tree in the strict sense. However, the graph can be described as a combination of two trees which share two nodes at the double link (a dual tree). In Fig. D.30, the different trees are indicated by the shading of parcels and links (the purple nodes indicate parcels that are shared between the blue and red tree). Any graph can contain at most one double link, as each parcel can only point to one other parcel. Larger cycles are also excluded: given

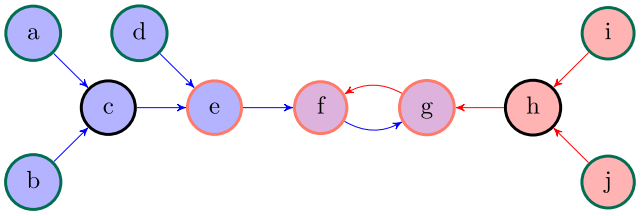

Fig. D.30. A parcel merging graph that is a dual tree.

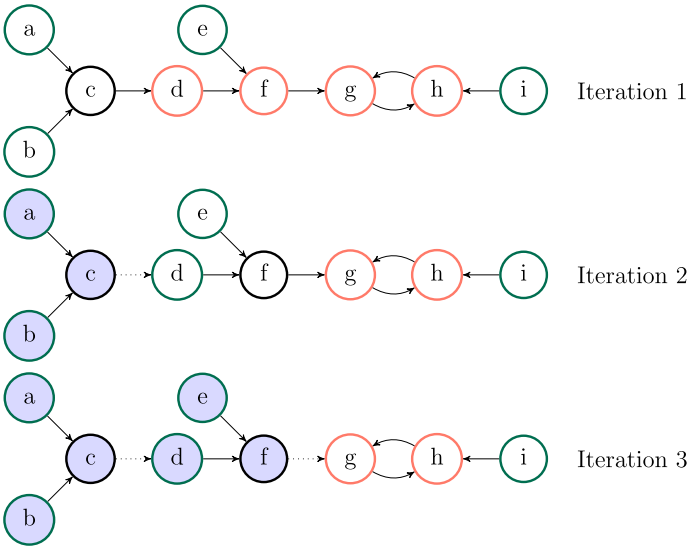

Fig. D.31. Merging procedure example: iterative stage.

a set of small parcels, there will be one link which corresponds to the shortest distance (the double link). The corresponding parcel pair will not have another outward link that would be needed to form a larger cycle.

The first (iterative) stage of the algorithm deals both with trees having a large parcel, and with any links in a dual tree that do not involve a double-linked parcel. As discussed above, the double-linked parcels remain unavailable for merging during the iterative stage of the algorithm and require a separate (though simple) procedure.

An example of the iterative algorithm is given in Fig. D.31. The algorithm gradually merges the (dual) tree, starting from the initial leaf parcels. In each iteration, we first establish the leaf parcels, followed by the unavailable parcels. Then we determine valid mergers, which involve a single or multiple leaf parcels merging with an available parcel.

In the first iteration shown in Fig. D.31, parcel c is the only available parcel, and we merge parcels a and b with it. Subsequently, these parcels are marked as merged, indicated by light blue shading here. The link from the merged parcel c to the unmerged parcel d is disregarded in the second iteration, and d now becomes a leaf parcel. This subsequently means parcel f becomes available, and parcels d and e can be merged with it. Once this has been done, no further mergers can be established: note parcel i cannot merge with the unavailable parcel h.

Once no further mergers have been found during an iteration, the algorithm proceeds to resolving the double links and parcels that point to double-linked parcels (see Fig. D.32). We now mark any double-linked parcel that is connected to at least one leaf parcel as available for merging. Here, this means h becomes available for merging and both g and i are merged with it. In principle, we could mark parcel g as a leaf parcel as well at this point, but there is no need to do this in our implementation.

The outgoing links from an available parcel are removed in this stage. In some other parcel configurations, both double-linked parcels have leaf parcels pointing to them and become available: in such cases the double link is broken in both directions (see Fig. D.33).

An isolated pair of double-linked parcels results in both parcels being marked as unavailable. If a link between two unavailable parcels is found while resolving double links (e.g. a pointing to b and vice versa, as in Fig. D.34, panel 1), we mark the parcel with an outgoing link (e.g. parcel a in panel 2) as available, so that parcel b merges with it when the link pointing from b to a is found (panel 3). Note that because of the conservative properties of the merging process, the direction of merging is arbitrary: parcel a merging into b yields the same result as b merging into a.

Although the current implementation of EPIC only has shared-memory parallelism, the algorithm has been designed with distributed parallelism in mind. The number of iterations involved in tests with about 300,000 parcels is typically at most

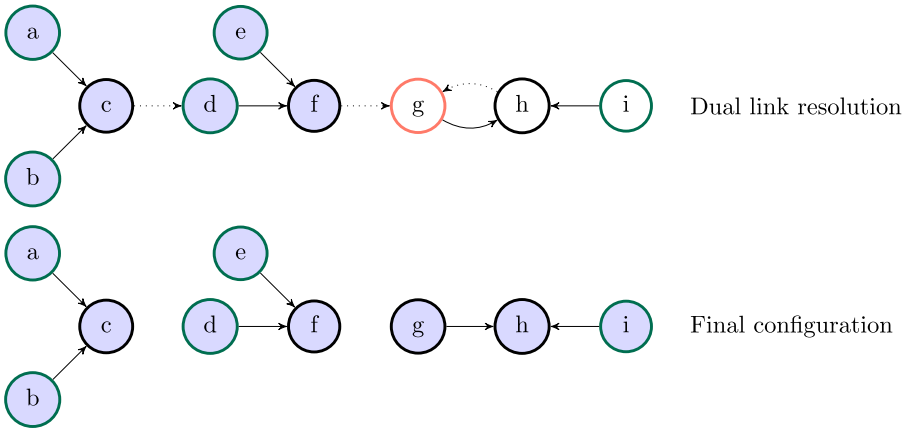

Fig. D.32. Merging procedure example: resolution of dual link and final configuration.

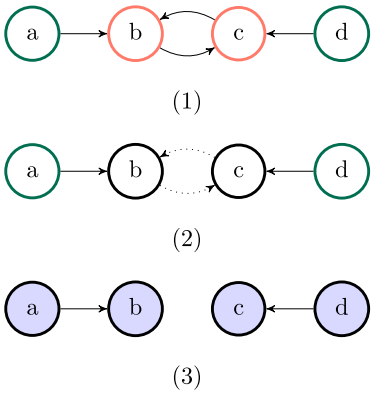

Fig. D.33. Merging procedure example: resolution of a dual link with leaf parcels on both sides. 1) Configuration after first (iterative) stage. 2) Both dual-linked parcels are made available, and outgoing links are removed. 3) Final configuration.

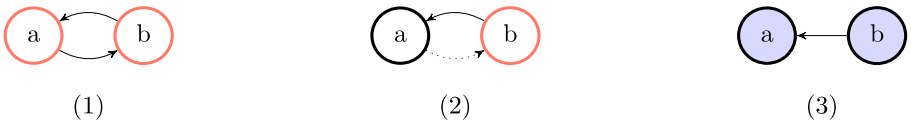

Fig. D.34. Procedure for resolving isolated dual links. 1) Configuration after first (iterative) stage. 2) The first parcel encountered is made available, and the corresponding outgoing link is removed. 3) Final configuration.

5, although this number goes up to about 10 during the initial stage of the Taylor-Green simulation, where chains of small parcels tend to occur near the symmetry axis. The number of parcels that need to be merged in any time step is small, and none of the operations involved are global.

D.2. Merging parcel groups

The merger rules above identify groups of parcels to be merged in the current time step. The actual merger is carried out as follows. Each merger group consists of a set of elliptical parcels centred at $\boldsymbol{x}_k$, with areas $V_k = \pi a_k b_k$ and shape matrices $\boldsymbol{B}_k$, i.e.

$$B_{11,k} = \frac{4}{V_k} \int_{\Omega_k} x'^2 \, \mathrm{d}x' \mathrm{d}y' \tag{D.1}$$

$$B_{12,k} = \frac{4}{V_k} \int_{\Omega_k} x' y' \, \mathrm{d}x' \mathrm{d}y' \tag{D.2}$$

$$B_{22,k} = \frac{4}{V_k} \int_{\Omega_k} y'^2 \, \mathrm{d}x' \mathrm{d}y' \tag{D.3}$$

where Ω_k denotes the area of integration, while $x' \equiv x - x_k$ and $y' \equiv y - y_k$ are coordinates relative to the parcel centres. Here the index k identifies the parcels belonging to the specific merger group under consideration.

We then create a merged parcel having total area

$$V = \sum_k V_k \tag{D.4}$$

and centre

$$\boldsymbol{x} = \frac{1}{V} \sum_k V_k \boldsymbol{x}_k \tag{D.5}$$

(modulo periodicity in x). The corresponding $\boldsymbol{B}$ matrix components are found from

$$B_{11} = \frac{4}{V} \sum_k \int_{\Omega_k} (\check{x}_k + x')^2 \, \mathrm{d}x' \mathrm{d}y' \tag{D.6}$$

$$B_{12} = \frac{4}{V} \sum_k \int_{\Omega_k} (\check{x}_k + x')(\check{y}_k + y') \, \mathrm{d}x' \mathrm{d}y' \tag{D.7}$$

$$B_{22} = \frac{4}{V} \sum_k \int_{\Omega_k} (\check{y}_k + y')^2 \, \mathrm{d}x' \mathrm{d}y' \tag{D.8}$$

where $\check{\boldsymbol{x}}_i \equiv \boldsymbol{x}_i - \boldsymbol{x}$. Performing the integrals yields Eq. (19). These are scaled by a constant factor as described in Section 2.6 to ensure that the determinant $\boldsymbol{B}$ is $a^2 b^2 = (V/\pi)^2$, required for consistency.

Appendix E. Time step adaptation

In EPIC, the time step is adapted to ensure that the parcel stretch between successive time steps is kept small, and that buoyancy oscillations (internal waves) are well resolved in time. The parcel stretch is defined by

$$\gamma = \frac{1}{a} \frac{\mathrm{d}a}{\mathrm{d}t} = \frac{1}{2a^2} \frac{\mathrm{d}a^2}{\mathrm{d}t}. \tag{E.1}$$

Inserting Eq. (13) to replace a^2 by $\frac{1}{2}(B_{11} + B_{22}) + Q$ where

$$Q \equiv \sqrt{\tfrac{1}{4}(B_{11} - B_{22})^2 + B_{12}^2}\,, \tag{E.2}$$

and using the shorthand notation $\dot{B}_{ij} \equiv \mathrm{d}B_{ij}/\mathrm{d}t$, we have

$$\frac{\mathrm{d}a^2}{\mathrm{d}t} = \frac{1}{2}(\dot{B}_{11} + \dot{B}_{22}) + \frac{\frac{1}{2}(B_{11} - B_{22})(\dot{B}_{11} - \dot{B}_{22}) + 2B_{12}\dot{B}_{12}}{2Q} \tag{E.3}$$

Next, using Eq. (8) to replace the time derivatives of B_{ij}, we find

$$\frac{\mathrm{d}a^2}{\mathrm{d}t} = S_{11}B_{11} + (S_{12} + S_{21})B_{12} + S_{22}B_{22} + \frac{(B_{11} - B_{22})(S_{11}B_{11} - S_{22}B_{22}) + (B_{11} + B_{22})(S_{12} + S_{21})B_{12}}{2Q} \tag{E.4}$$

$$= S_{11}(B_{11} - B_{22}) + (S_{12} + S_{21})B_{12} + \frac{(B_{11} + B_{22})[S_{11}(B_{11} - B_{22}) + (S_{12} + S_{21})B_{12}]}{2Q} \tag{E.5}$$

$$= [S_{11}(B_{11} - B_{22}) + (S_{12} + S_{21})B_{12}]\left[1 + \frac{(B_{11} + B_{22})}{2Q}\right] \tag{E.6}$$

where we have used incompressibility, $S_{22} = -S_{11}$, in the penultimate step. The term in square brackets on the right is just a^2/Q. Hence, it follows that the parcel stretch γ is equal to

$$\gamma = \frac{1}{2a^2} \frac{\mathrm{d}a^2}{\mathrm{d}t} = \frac{S_{11}(B_{11} - B_{22}) + (S_{12} + S_{21})B_{12}}{\sqrt{(B_{11} - B_{22})^2 + 4B_{12}^2}}\,. \tag{E.7}$$

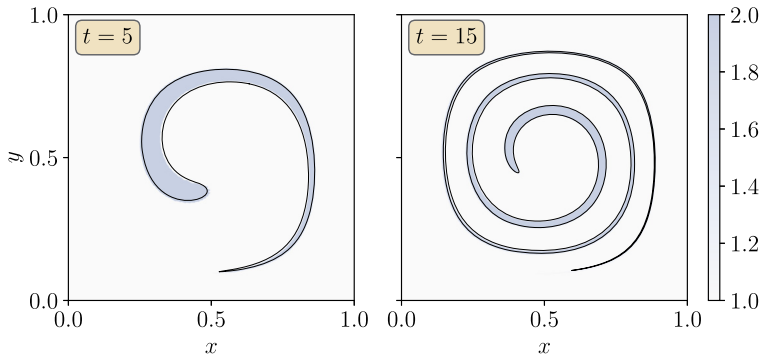

Fig. F.35. Evolution of the tracer scalar field in the static Taylor-Green vortex test case (as defined in [5]) using 100×100 grid cells. We use the default parameter settings as given in Table 2, except for the initial number of parcels per cell, here 4 as in [5]. The black solid line defines the reference solution obtained from the contour advection (CA) algorithm on a 400×400 grid.

Next, we seek to find the parcel shape which maximises γ. To this end, let $x \equiv B_{11} - B_{22}$, $y \equiv 2B_{12}$, $\sigma_1 = S_{11}$ and $\sigma_2 = (S_{12} + S_{21})/2$. Then γ may be written more simply as

$$\gamma = \frac{\sigma_1 x + \sigma_2 y}{\sqrt{x^2 + y^2}} = \sigma_1 \cos\theta + \sigma_2 \sin\theta \tag{E.8}$$

where $\tan\theta = y/x$. Maximising γ over θ, i.e. setting $\mathrm{d}\gamma/\mathrm{d}\theta = 0$, implies

$$\cos\theta = \frac{\sigma_1}{\sqrt{\sigma_1^2 + \sigma_2^2}} \quad \text{and} \quad \sin\theta = \frac{\sigma_2}{\sqrt{\sigma_1^2 + \sigma_2^2}}\,. \tag{E.9}$$

Hence, the maximum stretch $|\gamma|$ is simply

$$\sqrt{\sigma_1^2 + \sigma_2^2} = \sqrt{S_{11}^2 + \tfrac{1}{2}(S_{12} + S_{21})^2} = \frac{1}{2}\sqrt{(S_{11} - S_{22})^2 + (S_{12} + S_{21})^2}. \tag{E.10}$$

Equation (42) takes the maximum value of $\sqrt{\sigma_1^2 + \sigma_2^2}$ over all grid points.

Appendix F. Static Taylor-Green vortex test case

In this section we report results for EPIC running the static Taylor-Green vortex test case discussed by [5]. This case uses the velocity field

$$u(x, y) = \sin\pi x \cos\pi y, \qquad v(x, y) = -\cos\pi x \sin\pi y \tag{F.1}$$

in the domain $[0, 1]^2$. Each parcel i is given a tracer scalar attribute C_i with value

$$C_i = \begin{cases} 2 & r_i \le 0.15 \\ 1 & r_i > 0.15 \end{cases} \tag{F.2}$$

where $r_i^2 = (x_i - 0.5)^2 + (y_i - 0.75)^2$. Here, we initialise the parcels directly since the algorithm in Section 3.1 is not able to handle discontinuous fields (parcels close to the interface with $r_i \approx 0.15$ would have smoothed values of C_i). Note, we use a 100×100 grid, the coarsest used by [5]. In contrast to all other test cases in this paper, we use the exact gridded velocity field and its derivatives to advect the parcels in time, as done by [5]; hence there is no need to solve the inversion problem.

In Fig. F.35 we show the tracer scalar field at times $t = 5$ and $t = 15$ (reported by [5]), and we compare with a reference solution using 'Contour Advection' (CA) [51]. CA represents the tracer contour by a set of connected nodes, and advects them in the analytical velocity field. The number of nodes varies in time in response to both contour curvature and length. Here, we have used enough nodes to ensure an accurate solution (364 initially, rising to 1359 at $t = 5$ and to 3905 at $t = 15$). The time step is fixed at $\Delta t = 0.01$, and results vary insignificantly when using smaller time steps. The results in Fig. F.35 show excellent agreement between the two numerical methods, despite the use of a coarse grid in EPIC.

Appendix G. Code performance and parameter choices

Here we discuss some of the tests we have performed in order to determine, or narrow down, appropriate numerical parameter settings for EPIC. We have varied these settings as indicated in Table G.6. The tests below, and many others like in

Table G.6
The ranges of the different input parameters used to investigate code performance. Numbers in bold denote the default values that are used when another parameter is scanned.

Numerical parameter	Value
Maximum aspect ratio, $\lambda_{\max}$	3, **4**, 5
Minimum parcel area fraction, $\tilde{V}_{\min}$	1/20, **1/40**, 1/60
Maximum parcel area fraction, $\tilde{V}_{\max}$	1/2.25, **1/2.89**, 1/3.61
Gradient correction pre-factor, β	1.2, **1.8**, 2.4
Gradient correction max. compression, $C_{\max}$	0.25, **0.5**, 0.75
Time step pre-factor, α	0.1, **0.2**, 0.3

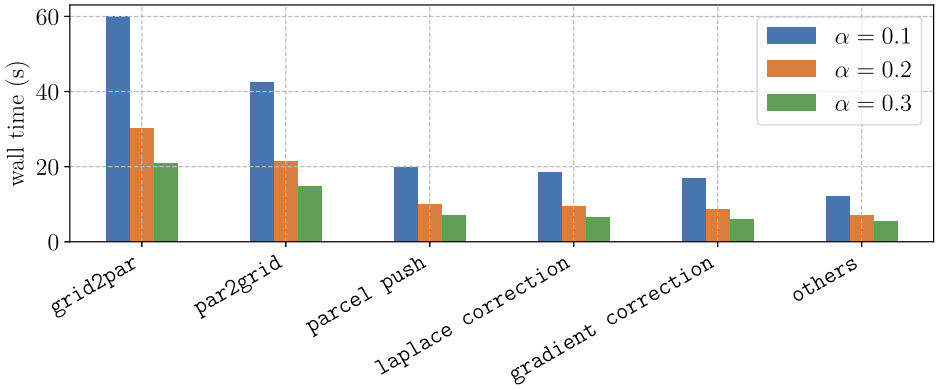

Fig. G.36. Timings of the Straka case (for 512×64 grid cells) with various time step pre-factors α. All timings below 4% of the total time are accumulated in `others`.

Appendix F, form the basis for the recommended parameter settings in Table 2. These tests, for example, included using an analytical velocity field in the Taylor-Green flow to verify that the deforming elliptical parcels well represent the analytical vorticity field out to arbitrarily long integration times (there is no degradation and parcel numbers remain approximately constant despite splitting and merging; note that not all of the cases in this article are run until the parcel number has equilibrated). Furthermore, we have extensively tested each part of the method by constructing a large number of 'unit tests'. Here we focus on the more challenging tests we have performed.

G.1. Time step

A major feature of EPIC which is critical to both accuracy and efficiency is time step adaption (see Section 3.2). An advantage of EPIC is that the time step Δt is chosen purely for accuracy, not for numerical stability (the constraint we impose on aspect ratio growth here is much weaker than the CFL constraint commonly used in grid-based approaches). An accurate time step for the fluid model investigated requires one to resolve both the strain and the buoyancy frequency time scales, respectively $\gamma_{\max}^{-1}$ and $N_{\max}^{-1}$, see Eq. (41). The dimensionless pre-factors α_s and α_b limit the time step Δt to $\alpha_s/\gamma_{\max}$ or $\alpha_b/N_{\max}$, whichever is smaller. Choosing small pre-factors is accurate but it is expensive; the recommended choice $\alpha_s = \alpha_b = \alpha = 0.2$ is a compromise between accuracy and efficiency.

This is summarised in Fig. G.36, where we have broken down the cost for each part of the code in the Straka test case. The final time of 900 s is reached after 672 time steps when $\alpha = 0.1$, 337 time steps when $\alpha = 0.2$ and 233 time steps when $\alpha = 0.3$. Increasing α from 0.1 to 0.2 halves the total execution time from 170.046 s to 86.45 s, but $\alpha = 0.3$ reduces the execution time further to only 60.78 s. Values of $\alpha \geq 0.4$ can cause excessive parcel stretching, violating the consistency condition, Eq. (48).

Choosing both pre-factors α_s and α_b to be equal is based on the performance of EPIC in both the Straka and Robert test cases. For example, in the Straka test, Fig. G.37 shows how the time step is first controlled by the buoyancy frequency when the flow is less active, then switches to being controlled by the velocity strain at later times. The two time step estimates are of similar magnitude and tend to co-vary except for the earliest times. This is why we have chosen a single parameter α for both time-step estimates. After the crossing point, the time step stabilises to about $1.34\ \text{s} \pm 0.30\ \text{s}$.

Similar results are found in the Robert case, see Fig. G.38. Again the time step at early times (before $t = 203.5$ s) is limited by the buoyancy frequency. Thereafter, as the flow becomes more active, the time step is limited by the velocity strain. The two time step estimates differ by a mean value of 1.71 s after the crossing point.

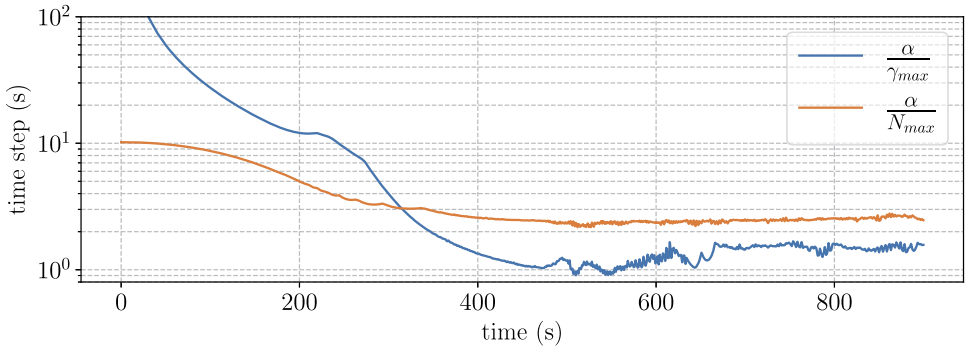

Fig. G.37. Evolution of the velocity strain and buoyancy frequency time step estimates Eq. (41) in the Straka test case (for 1024×128 grid cells). The simulation uses the default values listed in Table 2. The initial time step estimate using the velocity strain is undefined (since the flow starts from rest), but drops substantially after a few time steps. We only show a portion of the y axis range.

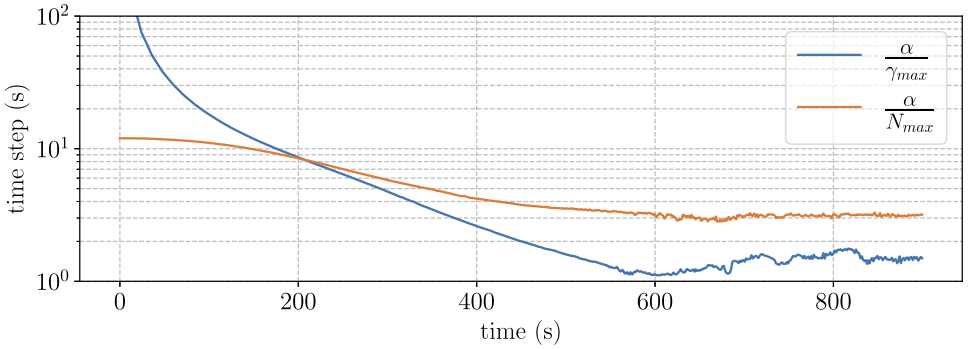

Fig. G.38. As in Fig. G.37 but for the Robert test case (for 256×384 grid cells).

G.2. Variations in the numerical parameter settings

We next use the Straka test case to find suitable values for all user-defined input parameters. To this end, we fix the grid resolution at 512×64 cells but vary individual input parameters over the ranges indicated in Table G.6, holding all other parameters fixed to the values indicated in bold. We also use 9 parcels per cell initially, as this appears to be a good choice for keeping the r.m.s. area error uniformly small in time (see Fig. 10).

The default parameters in Table 2 were chosen for efficiency and accuracy, but the results prove not to be sensitive to these choices. This is quantified next by comparing the location of the buoyancy-weighted centre of mass $(\langle x\rangle_b, \langle y\rangle_b)$ at time $t = 900$ s. We evaluate this measure over the right half of the domain ($x \geq 0$) using

$$\langle x\rangle_b = \frac{\sum_{x_i \geq 0} x_i b_i V_i}{\sum_{x_i \geq 0} b_i V_i} \quad \text{and} \quad \langle y\rangle_b = \frac{\sum_{x_i \geq 0} y_i b_i V_i}{\sum_{x_i \geq 0} b_i V_i}\,. \tag{G.1}$$

We then compare these values with their reference values $\langle x\rangle_b^{\text{ref}}$ and $\langle y\rangle_b^{\text{ref}}$, obtained using all the default parameters in bold, by computing the maximum relative deviation

$$|\langle x\rangle_b - \langle x\rangle_b^{\text{ref}}|/r_\circ \quad \text{and} \quad |\langle y\rangle_b - \langle y\rangle_b^{\text{ref}}|/r_\circ \tag{G.2}$$

where

$$r_\circ = \frac{1}{2\pi}\int_0^{2\pi} \frac{a_\circ b_\circ \mathrm{d}\theta}{b_\circ^2 \cos^2\theta + a_\circ^2 \sin^2\theta} \approx 2.746 \text{ km}. \tag{G.3}$$

is the mean radius of the initial elliptical buoyancy distribution.

The percentage maximum relative deviation is listed in Table G.7 for each varied parameter (holding all others fixed). We see that the deviations are exceedingly small for both $\tilde{V}_{\max}$ and $C_{\max}$, likely because these parameters are rarely needed over the course of the simulation. On the other hand, and as expected, the maximum aspect ratio $\lambda_{\max}$ and the time step pre-factor α have the greatest influence on the flow. Nevertheless, the deviations in e.g. $\langle y\rangle_b$ are still small, as shown in Fig. G.39 for variations in α, despite the complexity of the flow evolution.

Table G.7
The maximum relative deviations of the mass-weighted centre of mass at 900 s. The reference values are obtained using the default parameter settings.

Varied numerical parameter	Max. relative deviation (%) in	
	$\langle x \rangle_b$	$\langle y \rangle_b$
Maximum aspect ratio, $\lambda_{\max}$	1.3	1.9
Minimum parcel area fraction, $\tilde{V}_{\min}$	1.3	1.6
Maximum parcel area fraction, $\tilde{V}_{\max}$	6.9×10^{-7}	1.1×10^{-8}
Gradient correction pre-factor, β	9.1×10^{-1}	4.5×10^{-1}
Gradient cor. max. compression, $C_{\max}$	9.4×10^{-7}	2.5×10^{-8}
Time step pre-factor, α	1.7	2.0

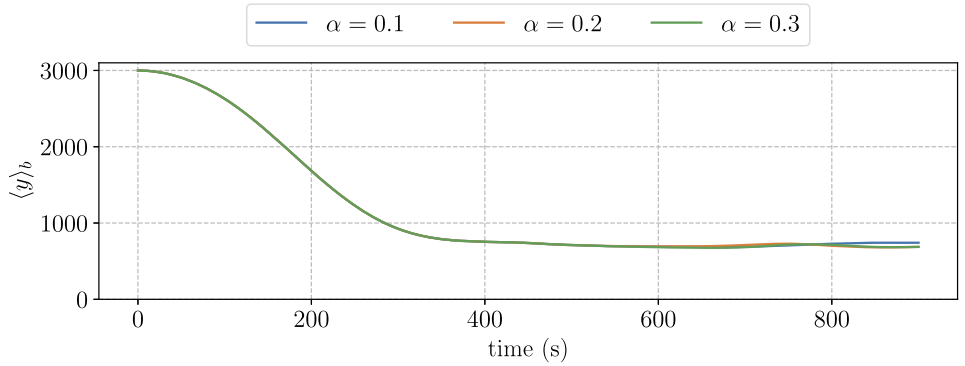

Fig. G.39. Buoyancy weighted centre of mass in the vertical direction for various time step pre-factors α. Results are obtained with 512×64 grid cells.

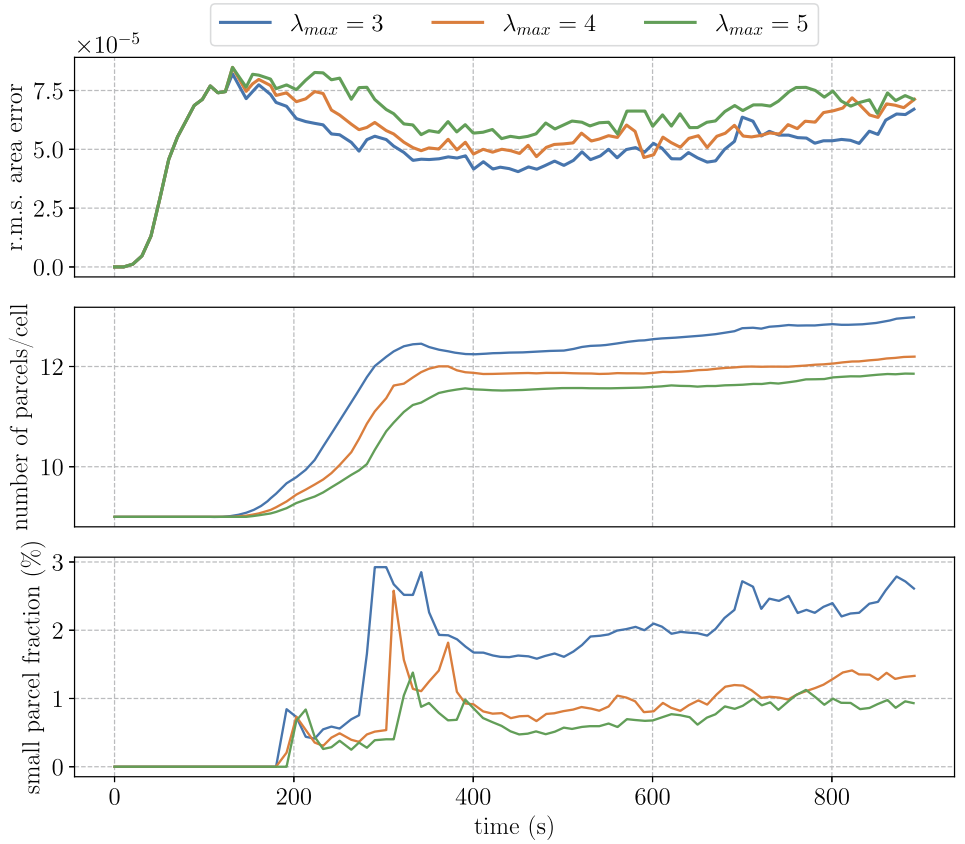

Fig. G.40. Effect of changing the maximum aspect ratio $\lambda_{\max}$ on (top) the relative r.m.s. area error, (middle) the average number of parcels per grid cell, and (bottom) the percentage of small parcels with $V_i \leq V_{\min}$ following splitting.

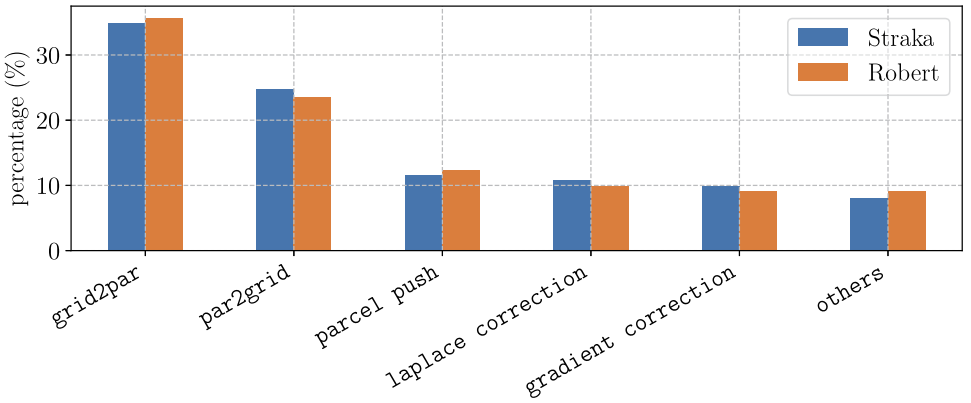

Fig. G.41. Timing percentages of the Straka and Robert cases using the default parameter values in Table 2 and 512×64 and 256×384 grid cells, respectively. As expected the operations `par2grid` and `grid2par` take most of the time. Together they account for about 60% of the total time. All timings below 5% of the total time are accumulated in `others`.

We next consider the impact of $\lambda_{\max}$ on other measures of accuracy. Fig. G.40 (top panel) shows that the gridded relative r.m.s. area error reduces when using smaller maximum aspect ratios $\lambda_{\max}$. This improvement comes, however, at a higher cost since the number of parcels per cell increases (see middle panel) and parcels split more frequently (see bottom panel). As the loss in efficiency going from $\lambda_{\max} = 5$ to $\lambda_{\max} = 4$ is relatively small, while the gain in accuracy is more significant, we have adopted $\lambda_{\max} = 4$ as the default, recommended value.

G.3. Timings

Detailed timing percentages for the Straka and Robert test case are presented in Fig. G.41. The figure shows that the routines interpolating to and from the grid, i.e. `par2grid` and `grid2par`, are responsible for the highest costs. Together, they take approximately 60% of the total time. The advection of the parcels (`parcel push`) by the low-storage RK4 algorithm uses about 12%. The newly developed parcel correction methods (`gradient` and `laplace correction`) account for about 20%. All other operations taking less than 5% are accumulated in `others`, accounting for 8% to 9%.

Appendix H. Subgrid visualisation

For subgrid visualisation and detailed spectral analysis, we follow the approach described in [10], extended to elliptical parcels with some modifications. Each parcel contributes to the fine-scale gridded field at a number of grid points in its surroundings. The values of a field q on the fine-scale grid $\breve{q}(\bar{\boldsymbol{x}})$ (distinguished from an overbar to prevent confusion with the coarser grid used for advection) are given by a sum of contributions from nearby parcels (indexed by i):

$$\breve{q}(\bar{\boldsymbol{x}}) = \frac{\sum_i q_i f(r_{\text{eff}}(\bar{\boldsymbol{x}}, \boldsymbol{x}_i))}{\sum_i f(r_{\text{eff}}(\bar{\boldsymbol{x}}, \boldsymbol{x}_i))} \tag{H.1}$$

where $r_{\text{eff}}(\bar{\boldsymbol{x}}, \boldsymbol{x}_i)$ is the effective radial distance in elliptical coordinates, i.e. the distance between $\boldsymbol{x}_i$ and $\bar{\boldsymbol{x}}$ normalised by the distance between the ellipse centre point and its edge on the line connecting $\boldsymbol{x}_i$ and $\bar{\boldsymbol{x}}$:

$$r_{\text{eff}}(\bar{\boldsymbol{x}}, \boldsymbol{x}_i) = \sqrt{(\bar{\boldsymbol{x}} - \boldsymbol{x}_i)^T \boldsymbol{B}_i^{-1} (\bar{\boldsymbol{x}} - \boldsymbol{x}_i)} \tag{H.2}$$

For the plots in this article, we use a generalised form of equation A7 in [10]:

$$\begin{aligned} f(r_{\text{eff}}) &= \exp\left(-(r_{\text{eff}}/r_v)^{p_v}\right) && \text{for } r_{\text{eff}} \le r_{\max} \\ f(r_{\text{eff}}) &= 0 && \text{for } r_{\text{eff}} > r_{\max}\,. \end{aligned} \tag{H.3}$$

Here, r_v and p_v are two coefficients, for which we recommend the choices $r_v = 1$ and $p_v = 3$. A low value of r_v (e.g. 1) and a high value of p_v (e.g. 3) result in the shapes of individual ellipses being visible with sharp gradients between individual ellipses, and give larger spectral contributions at high subgrid wavenumbers, whereas the combination of a relatively high value of r_v (e.g. 2) and a low value of p_v (e.g. 2) smooths out filaments in the flow. A low value of p_v also means that parcel contributions fall off less rapidly at larger distances from a parcel.

The area integral $\iint f(r_{\text{eff}})\,\mathrm{d}A$ is proportional to the volume of the parcel. We choose a relatively large value of $r_{\max} = 4$ here so that the discontinuity in f at $r_{\text{eff}} = r_{\max}$ is small. In order to calculate the contributions of each parcel efficiently, we first determine the major semi-axis length of the ellipse a_i from $\boldsymbol{B}_i$. We then consider the fine-scale grid points in a square with sides $2a_i r_{\max}$ around the ellipse centre (see Fig. H.42). For each such point, we first check if it falls within a circle

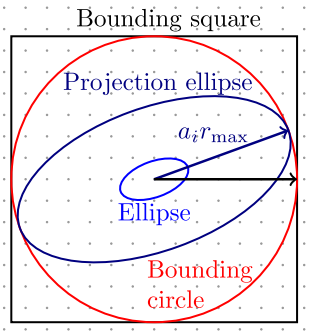

Fig. H.42. Determination of the region contributing to the subgrid visualisation of a parcel, with the fine-scale visualisation grid shown by grey points.

of radius $a_i r_{\max}$ (this is relatively efficient). If it does, we further check if it falls within the projection ellipse $r_{\text{eff}} < r_{\max}$ defined in Eq. (H.3). Only then do we add its contribution to $\breve{q}(\bar{\boldsymbol{x}})$.

Appendix I. Pseudo-spectral approach

In this appendix we provide details of the pseudo-spectral code which we have used to compare with the EPIC simulations presented in this paper. The essence of the pseudo-spectral approach is to carry out all numerical operations apart from nonlinear products in spectral space, after a (fast) Fourier transform [32]. That is, the time stepping of the fields and the calculation of the velocity field from the vorticity field are all done in spectral space. The only operation done in physical space is the calculation of the terms $\boldsymbol{u}\cdot\nabla\zeta$ and $\boldsymbol{u}\cdot\nabla b$ appearing in the material derivatives $\mathrm{D}\zeta/\mathrm{D}t$ and $\mathrm{D}b/\mathrm{D}t$, respectively, in Eq. (1) and Eq. (2). The standard '2/3 rule' is used to truncate all spectral fields prior to use in nonlinear products to avoid aliasing errors [32].

In the pseudo-spectral code developed, Fourier series are used in both x and y to represent all fields. In the periodic direction, x, a full sine and cosine series (or a series in e^{ikx}) is used in wavenumbers k, as in EPIC (see Appendix A). In the y direction, we use a cosine series for ζ, b and u, with terms proportional to $\cos l(y-y_{\min})$ where l is the discrete y wavenumber taking the values 0, π/L_y, $2\pi/L_y$, ..., $n_y\pi/L_y$, and where L_y is the domain width in y. We use a sine series for v with terms proportional to $\sin l(y-y_{\min})$ to ensure $v=0$ on the y boundaries.

The evolution equations are rewritten as

$$\frac{\partial\zeta}{\partial t} = -\boldsymbol{u}\cdot\nabla\zeta + b_x - \mathcal{D}\zeta \tag{I.1}$$

$$\frac{\partial b}{\partial t} = -\boldsymbol{u}\cdot\nabla b - \mathcal{D}b\,, \tag{I.2}$$

to include either a molecular diffusion operator $\mathcal{D}=-\nu\nabla^2$, or a hyperdiffusion operator $\mathcal{D}=C\zeta_{\text{char}}(-\nabla^2)^3$, for numerical stability.

When using molecular diffusion, we choose a constant viscosity given by

$$\nu = C\sqrt{(b_{\max}(0)-b_{\min}(0))/k_{\max}^3}\,, \tag{I.3}$$

where C is a dimensionless constant and $k_{\max}$ is the maximum horizontal wavenumber ($k_{\max}=n_x/2L_x$). The rationale behind this choice is that ν should have units of length-squared over time, and that the length scale should be proportional to the diffusion length, i.e. $\propto 1/k_{\max}$, while the time scale should be inversely proportional to the highest possible buoyancy frequency, i.e. $\propto 1/\sqrt{k_{\max}(b_{\max}(0)-b_{\min}(0))}$. By trial and error, and at many different resolutions, we have found that $C=2$ well preserves monotonicity: $b_{\max}(t)$ does not increase by more than approximately $10^{-3}(b_{\max}(0)-b_{\min}(0))$, and $b_{\min}(t)$ does not decrease by more than the same amount (theoretically, diffusion can only reduce $b_{\max}$ and increase $b_{\min}$). Notably, $\nu\approx 89$ m^2/s in the Straka test case using a 100 m grid spacing (corresponding to 512×64 grid cells), which compares favourably with $\nu=75$ m^2/s used in the original test case [25].

When using hyperdiffusion, C is a dimensionless constant chosen to ensure that field spectra decay at large wavenumbers ($C=10$ was used here for all resolutions — lower values produce visible spurious fringes in physical space and upturns in field spectra), while ζ_{char} is a characteristic vorticity value computed as follows. We first compute the r.m.s. vorticity ζ_{rms}. Then, for all grid points having $|\zeta|>\zeta_{\text{rms}}$, we accumulate the sums of $|\zeta|$ and ζ^2, respectively ζ_{L1} and ζ_{L2}. From these, we define $\zeta_{\text{char}}=\zeta_{\text{L2}}/\zeta_{\text{L1}}$. The rationale behind this is to be able to determine a sensible damping rate even for flows in which vorticity is confined to a small portion of the domain, as in the examples considered in this paper. Using the maximum (absolute) vorticity $\zeta_{\max}$ is an alternative, but this is much more variable in time than ζ_{char}. The latter by contrast slowly varies (see below). Note, for a patch of uniform vorticity ζ_0 of any area, we have $\zeta_{\text{char}}=|\zeta_0|$, which is a sensible choice for the characteristic vorticity.

The time stepping of Eq. (I.1) and Eq. (I.2) is done in spectral space, using the (implicit) trapezoidal rule with iteration. Specifically, if we denote the (k,l) spectral coefficient of any quantity q by $\hat{q}$, then we solve the equations

$$\frac{\hat{\zeta}^{n+1}-\hat{\zeta}^n}{\Delta t} = \frac{\hat{S}_\zeta^{n+1}+\hat{S}_\zeta^n}{2} - \hat{\mathcal{D}}\frac{\hat{\zeta}^{n+1}+\hat{\zeta}^n}{2} \tag{I.4}$$

$$\frac{\hat{b}^{n+1}-\hat{b}^n}{\Delta t} = \frac{\hat{S}_b^{n+1}+\hat{S}_b^n}{2} - \hat{\mathcal{D}}\frac{\hat{b}^{n+1}+\hat{b}^n}{2}, \tag{I.5}$$

where n denotes the time level, $\hat{S}_\zeta$ is the Fourier coefficient of the vorticity source term $-\boldsymbol{u}\cdot\nabla\zeta + b_x$, and similarly $\hat{S}_b$ is the Fourier coefficient of the buoyancy source term $-\boldsymbol{u}\cdot\nabla b$ (all spatial derivatives are computed spectrally by wavenumber multiplication). In the above, quantities at time level n correspond to time t and are known, while we seek $\hat{\zeta}^{n+1}$ and $\hat{b}^{n+1}$ at time $t+\Delta t$. In spectral space, the dissipation operator $\hat{\mathcal{D}} = \nu(k^2+l^2)$ for molecular diffusion, and $\hat{\mathcal{D}} = C\zeta_{\text{char}}^n(k^2+l^2)^3$ for hyperdiffusion.

We solve the above system iteratively by first using $\hat{S}_\zeta^{n+1} = \hat{S}_\zeta^n$ and $\hat{S}_b^{n+1} = \hat{S}_b^n$, making a first estimate of $\hat{\zeta}^{n+1}$ and $\hat{b}^{n+1}$, then iterating the above equations two more times with improved estimates of the source terms $\hat{S}_\zeta^{n+1}$ and $\hat{S}_b^{n+1}$. Algebraically, it is simplest to first store the fixed quantities $\hat{\zeta}_m \equiv \hat{\zeta}^n + \frac{1}{4}\Delta t\hat{S}_\zeta^n$ and $\hat{b}_m \equiv \hat{b}^n + \frac{1}{4}\Delta t\hat{S}_b^n$, then, at each iteration, update

$$\hat{\zeta}^{n+1} = \hat{\mathcal{L}}\left(\hat{\zeta}_m + \tfrac{1}{4}\Delta t\hat{S}_\zeta^{n+1}\right) - \hat{\zeta}^n \qquad \text{and} \qquad \hat{b}^{n+1} = \hat{\mathcal{L}}\left(\hat{b}_m + \tfrac{1}{4}\Delta t\hat{S}_b^{n+1}\right) - \hat{b}^n \tag{I.6}$$

where $\hat{\mathcal{L}} \equiv 2/(1+\frac{1}{2}\Delta t\hat{\mathcal{D}})$ is fixed. The above update is performed three times, each time with an improved estimate of the source terms.

The time step Δt is adapted before the above iteration is carried out using the same restrictions on the maximum strain rate and buoyancy frequency enforced in EPIC, see Eq. (41), except here we take $\alpha_s = \alpha_b = 0.1$ by default. This is half the default used in EPIC, but the time-stepping used in the pseudo-spectral code is only second order, compared to fourth order in EPIC. Additionally, to maintain numerical stability, it is necessary to ensure the time step is below the CFL limit $\Delta t_{\text{CFL}} = \min(\Delta x, \Delta y)/|\boldsymbol{u}|_{\max}$. Here we require both Eq. (41) and $\Delta t < 0.8\Delta t_{\text{CFL}}$; the factor of 0.8 is necessary due to nonlinearity, which is not accounted for when deriving the CFL limit.

The source terms in Eq. (I.6) depend on the velocity field $\boldsymbol{u}$, which must also be updated using the most recent estimate of $\hat{\zeta}^{n+1}$. Since the flow is incompressible, we do this by expressing $\boldsymbol{u} = (-\partial\psi/\partial y, \partial\psi/\partial x)$ and finding the streamfunction ψ from the solution to Poisson's equation $\nabla^2\psi = \zeta$ as in Appendix A. Here, however, since ζ is represented as a double Fourier series, it makes sense to represent ψ similarly. Then, the (k,l) Fourier coefficient $\hat{\psi}$ is found simply from $\hat{\psi} = -\hat{\zeta}/(k^2+l^2)$ for k and l not both zero. The problem with this solution is that it does not satisfy the boundary condition $v = \partial\psi/\partial x = 0$ on $y = y_{\min}$ and $y_{\max}$. This is because ζ is represented as a cosine series in y to allow the vorticity to take arbitrary boundary values, and $v = 0$ there requires $\psi = 0$, except for the x independent Fourier modes with $k = 0$. It is thus impossible to express ψ entirely in terms of a cosine series in y. If we further require that the mean horizontal velocity, i.e. the part of $u = -\partial\psi/\partial y$ for $k = 0$, be zero (without loss of generality), we can then require $\psi = 0$ on the boundaries, even for the $k = 0$ modes.

The boundary conditions on ψ can be enforced by adding solutions to Laplace's equation, $\nabla^2\psi = 0$. Let the streamfunction ψ be separated into three parts so that $\psi = \psi_a + \psi_p + \psi_h$, where $\psi_a = \frac{1}{2}\zeta_a(y-y_{\min})(y-y_{\max})$ is the part due to the domain-average vorticity ζ_a (associated with $k = l = 0$), ψ_p is the particular solution arising from $\hat{\psi}_p = -\hat{\zeta}/(k^2+l^2)$ in spectral space (for k and l not both zero), and ψ_h consists of solutions to Laplace's equation, i.e. linear combinations of $e^{\mathrm{i}kx}e^{\pm ky}$ for each discrete k. We want to choose $\psi_h = -\psi_p$ on each boundary, and there are just enough solutions of Laplace's equation to do this. If we represent the boundary values of ψ_p by

$$\psi_p(x, y_{\min}) = \sum_k a_k^- e^{\mathrm{i}kx} \qquad \text{and} \qquad \psi_p(x, y_{\max}) = \sum_k a_k^+ e^{\mathrm{i}kx} \tag{I.7}$$

then the solution for ψ_h is simply

$$\psi_h(x,y) = -\sum_k \frac{a_k^- \sinh k(y_{\max}-y) + a_k^+ \sinh k(y-y_{\min})}{\sinh kL_y} e^{\mathrm{i}kx}. \tag{I.8}$$

The first term in the sum, for $k = 0$, is found by taking the limit $k \to 0$, e.g. $\sinh k(y_{\max}-y)/\sinh kL_y$ reduces to $(y_{\max}-y)/L_y$.

Once the streamfunction ψ is determined, it is re-expressed as a sine series in y. This allows the velocity field $\boldsymbol{u}$ to be computed efficiently in spectral space by simple wavenumber multiplication. An inverse fast Fourier transform is then performed to recover $\boldsymbol{u}$ in physical space.

To justify the choice of hyperdiffusion, in particular the choice of the characteristic vorticity ζ_{char} defined above as a pre-multiplier, Fig. I.43 shows vorticity diagnostics from a 4096×512 pseudo-spectral simulation of the Straka test case. Here,

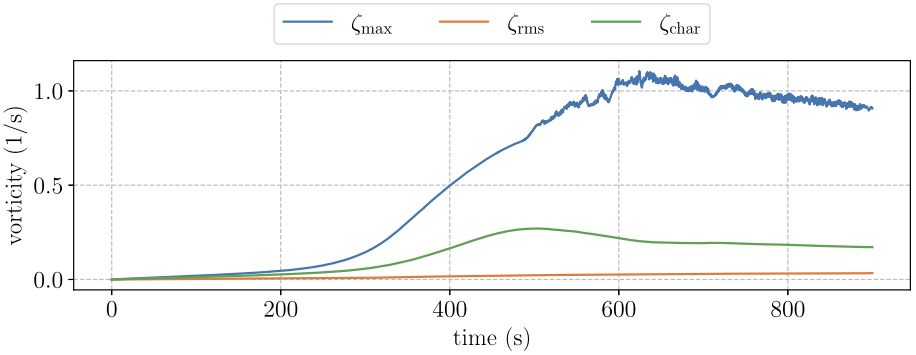

Fig. I.43. Time evolution of the rms, maximum and characteristic vorticity values in the Straka test case with hyperdiffusion, as found in a pseudo-spectral simulation with grid resolution 4096×512.

we see that the r.m.s. vorticity ζ_{rms} is smallest, as it is dominated by vast areas of the domain with little or no vorticity. On the other hand, the maximum vorticity ζ_{max} is largest, and is furthermore highly variable in time. The characteristic vorticity ζ_{char} lies in between ζ_{rms} and ζ_{max}, and varies slowly in time like ζ_{rms}. Notably, ζ_{char} is about 4 times smaller than ζ_{max}.

References

[1] F.H. Harlow, The particle-in-cell computing method for fluid dynamics, Methods Comput. Phys. 3 (1964) 319–343.
[2] J. Christiansen, N.J. Zabusky, Instability, coalescence and fission of finite-area vortex structures, J. Fluid Mech. 61 (1973) 219–243.
[3] G.H. Cottet, P. Koumoutsakos, Vortex Methods: Theory and Practice, Cambridge University Press, Cambridge, 2000.
[4] D.G. Dritschel, S.J. Böing, D.J. Parker, A.M. Blyth, The moist parcel-in-cell method for modelling moist convection, Q. J. R. Meteorol. Soc. 144 (715) (2018) 1695–1718, https://doi.org/10.1002/qj.3319.
[5] H. Samuel, A deformable particle-in-cell method for advective transport in geodynamic modelling, Geophys. J. Int. 214 (3) (2018) 1744–1773, https://doi.org/10.1093/gji/ggy231.
[6] M.J. Berger, P. Colella, Local adaptive mesh refinement for shock hydrodynamics, J. Comput. Phys. 82 (1) (1989) 64–84.
[7] S. Popinet, Gerris: a tree-based adaptive solver for the incompressible Euler equations in complex geometries, J. Comput. Phys. 190 (2) (2003) 572–600.
[8] H. Weller, T. Ringler, M. Piggott, N. Wood, Challenges facing adaptive mesh modeling of the atmosphere and ocean, Bull. Am. Meteorol. Soc. 91 (1) (2010) 105–108.
[9] J.A. Van Hooft, S. Popinet, C.C. Van Heerwaarden, S.J. Van der Linden, S.R. de Roode, B.J. Van de Wiel, Towards adaptive grids for atmospheric boundary-layer simulations, Bound.-Layer Meteorol. 167 (3) (2018) 421–443.
[10] S.J. Böing, D.G. Dritschel, D.J. Parker, A.M. Blyth, Comparison of the Moist Parcel-in-Cell (MPIC) model with large-eddy simulation for an idealized cloud, Q. J. R. Meteorol. Soc. 145 (722) (2019) 1865–1881.
[11] J. Walther, G. Morgenthal, An immersed interface method for the vortex-in-cell algorithm, J. Turbul. 3 (1) (2002) 039.
[12] X. Zhang, R. Bridson, A PPPM fast summation method for fluids and beyond, ACM Trans. Graph. 33 (6) (2014) 1–11.
[13] J.M. Owen, J.V. Villumsen, P.R. Shapiro, H. Martel, Adaptive smoothed particle hydrodynamics: methodology. II, Astrophys. J. Suppl. Ser. 116 (2) (1998) 155.
[14] E. Jo, D. Kim, O.-Y. Song, A new SPH fluid simulation method using ellipsoidal kernels, J. Vis. 14 (4) (2011) 371–379.
[15] K. Shibata, S. Koshizuka, I. Masaie, Cost reduction of particle simulations by an ellipsoidal particle model, Comput. Methods Appl. Mech. Eng. 307 (2016) 411–450.
[16] B. Legras, D. Dritschel, The elliptical model of two-dimensional vortex dynamics. I: The basic state, Phys. Fluids A, Fluid Dyn. 3 (5) (1991) 845–854, https://doi.org/10.1063/1.858015.
[17] D.G. Dritschel, J.N. Reinaud, W.J. McKiver, The quasi-geostrophic ellipsoidal vortex model, J. Fluid Mech. 505 (2004) 201–223, https://doi.org/10.1017/S0022112004008377.
[18] M. Carr, J. Franklin, S.E. King, P.A. Davies, J. Grue, D.G. Dritschel, The characteristics of billows generated by internal solitary waves, J. Fluid Mech. 812 (2017) 541–577, https://doi.org/10.1017/jfm.2016.823.
[19] W.J. McKiver, D.G. Dritschel, The motion of a fluid ellipsoid in a general linear background flow, J. Fluid Mech. 474 (2003) 147–173, https://doi.org/10.1017/S0022112002002859.
[20] D. Meyer, P. Jenny, Conservative velocity interpolation for PDF methods, in: PAMM: Proceedings in Applied Mathematics and Mechanics, vol. 4, Wiley Online Library, 2004, pp. 466–467.
[21] J. Fontane, D.G. Dritschel, The HyperCASL algorithm: a new approach to the numerical simulation of geophysical flows, J. Comput. Phys. 228 (2009) 6411–6425, https://doi.org/10.1016/j.jcp.2009.05.025.
[22] M.H. Carpenter, C.A. Kennedy, Fourth-order 2N-storage Runge-Kutta schemes, Technical Memorandum NASA-TM-109112 NASA Langley Research Center, 1994, https://ntrs.nasa.gov/citations/19940028444.
[23] C.C. van Heerwaarden, B.J.H. van Stratum, T. Heus, J.A. Gibbs, E. Fedorovich, J.P. Mellado, MicroHH 1.0: a computational fluid dynamics code for direct numerical simulation and large-eddy simulation of atmospheric boundary layer flows, Geosci. Model Dev. 10 (8) (2017) 3145–3165, https://doi.org/10.5194/gmd-10-3145-2017.
[24] The HDF Group, Hierarchical Data Format, version 5, https://www.hdfgroup.org/HDF5/, 1997-2021.
[25] J.M. Straka, R.B. Wilhelmson, L.J. Wicker, J.R. Anderson, K.K. Droegemeier, Numerical solutions of a non-linear density current: a benchmark solution and comparisons, Int. J. Numer. Methods Fluids 17 (1) (1993) 1–22, https://doi.org/10.1002/fld.1650170103.
[26] A. Robert, Bubble convection experiments with a semi-implicit formulation of the Euler equations, J. Atmos. Sci. 50 (13) (1993) 1865–1873, https://doi.org/10.1175/1520-0469(1993)050<1865:BCEWAS>2.0.CO;2.
[27] G.I. Taylor, A.E. Green, Mechanism of the production of small eddies from large ones, Proc. R. Soc. Lond. Ser. A, Math. Phys. Sci. 158 (895) (1937) 499–521, https://doi.org/10.1098/rspa.1937.0036.

[28] D. Dritschel, R. Polvani, A. Mohebalhojeh, The contour-advective semi-Lagrangian algorithm for the shallow water equations, Mon. Weather Rev. 127 (7) (1999) 1551–1565.
[29] B. Fornberg, N. Flyer, Solving PDEs with radial basis functions, Acta Numer. 24 (2015) 215–258.
[30] D.G. Dritschel, M.R. Jalali, On the regularity of the Green-Naghdi equations for a rotating shallow-water fluid layer, J. Fluid Mech. 865 (2019) 100–136, https://doi.org/10.1017/jfm.2019.47.
[31] M.K. Verma, A. Kumar, A. Pandey, Phenomenology of buoyancy-driven turbulence: recent results, New J. Phys. 19 (2017) 025012, https://doi.org/10.1088/1367-2630/aa5d63.
[32] D.G. Fox, S.A. Orszag, Pseudospectral approximation to two-dimensional turbulence, J. Comput. Phys. 11 (1973) 612–619.
[33] A. Mariotti, B. Legras, D. Dritschel, Vortex stripping and the erosion of coherent structures in two-dimensional flows, Phys. Fluids 6 (1993) 3954–3962.
[34] H. Yao, D. Dritschel, N. Zabusky, High-gradient phenomena in two-dimensional vortex interactions, Phys. Fluids 7 (1994) 539–548.
[35] D.G. Dritschel, S.M. Tobias, Two-dimensional magnetohydrodynamic turbulence in the small magnetic Prandtl number limit, J. Fluid Mech. 703 (2012) 85–98, https://doi.org/10.1017/jfm.2012.195.
[36] M.R. Jalali, D.G. Dritschel, Balance in non-hydrostatic rotating shallow-water flows, Phys. Fluids 33 (2021) 086601, https://doi.org/10.1063/5.0057707.
[37] G. Gibb, S. Boeing, D. Dritschel, A fully lagrangian dynamical core for the Met office NERC cloud model, Tech. rep, University of Edinburgh, 2018, https://www.archer.ac.uk/community/eCSE/eCSE12-10/eCSE12-10_Technical_Report.pdf.
[38] M. Frey, S.J. Böing, D. Dritschel, EPIC – elliptical parcel-in-cell, https://doi.org/10.5281/zenodo.6012883, Feb. 2022.
[39] D. Dritschel, Pseudo-spectral code for 2d stratified flows, https://doi.org/10.5281/zenodo.5940146, Feb. 2022.
[40] M. Frey, S.J. Böing, D. Dritschel, EPIC version 0.10.8 with static TG, https://doi.org/10.5281/zenodo.6560922, May 2022.
[41] T.A. Caswell, M. Droettboom, A. Lee, E.S. de Andrade, T. Hoffmann, J. Hunter, J. Klymak, E. Firing, D. Stansby, N. Varoquaux, J.H. Nielsen, B. Root, R. May, P. Elson, J.K. Seppänen, D. Dale, J.-J. Lee, D. McDougall, A. Straw, P. Hobson, H. Aizenman, C. Gohlke, A.F. Vincent, T.S. Yu, E. Ma, S. Silvester, C. Moad, N. Kniazev, E. Ernest, P. Ivanov, matplotlib/matplotlib: Rel: v3.5.1, https://doi.org/10.5281/zenodo.5773480, Dec. 2021.
[42] J.A. Bednar, J. Signell, K. Bowen, M. Liquet, C. Ball, J.-L. Stevens, R. Pittman, D.J. Sutherland, K. Pevey, P. Kats, holoviz/colorcet: Version 3.0.0, https://doi.org/10.5281/zenodo.5732885, Nov. 2021.
[43] P. Kovesi, Good colour maps: how to design them, arXiv:1509.03700, Sep. 2015.
[44] A. Collette, T. Kluyver, T.A. Caswell, J. Tocknell, J. Kieffer, A. Jelenak, A. Scopatz, D. Dale Chen, T. Vincent, H. Payno, J.G. Ferrer, P. Sciarelli, V. Valls, S. Ghosh, U.K. Pedersen, J. Kirkham, M. Raspaud, C. Danilevski, H. Abbasi, J. Readey, A. Paramonov, L. Chan, V.A. Solé, L. Jialin, Y. Feng, G.A. Vaillant, M. Teichmann, M. Brucher, S.R. Johnson, h5py/h5py: 3.5.0, https://doi.org/10.5281/zenodo.594310, Oct. 2021.
[45] C.R. Harris, K.J. Millman, S.J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N.J. Smith, R. Kern, M. Picus, S. Hoyer, M.H. van Kerkwijk, M. Brett, A. Haldane, J.F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, T.E. Oliphant, Array programming with NumPy, Nature 585 (7825) (2020) 357–362, https://doi.org/10.1038/s41586-020-2649-2.
[46] The pandas development team, pandas-dev/pandas: Pandas 1.3.5, https://doi.org/10.5281/zenodo.5774815, Dec. 2021.
[47] W. McKinney, Data structures for statistical computing in Python, in: Stéfan van der Walt, Jarrod Millman (Eds.), Proceedings of the 9th Python in Science Conference, 2010, pp. 56–61.
[48] P. Virtanen, R. Gommers, T.E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S.J. van der Walt, M. Brett, J. Wilson, K.J. Millman, N. Mayorov, A.R.J. Nelson, E. Jones, R. Kern, E. Larson, C.J. Carey, İ. Polat, Y. Feng, E.W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E.A. Quintero, C.R. Harris, A.M. Archibald, A.H. Ribeiro, F. Pedregosa, P. van Mulbregt, SciPy 1.0 contributors, SciPy 1.0: fundamental algorithms for scientific computing in Python, Nat. Methods 17 (2020) 261–272, https://doi.org/10.1038/s41592-019-0686-2.
[49] V. Esfahanian, S. Ghader, A.R. Mohebalhojeh, On the use of the super compact scheme for spatial differencing in numerical models of the atmosphere, Q. J. R. Meteorol. Soc. 131 (2005) 2109–2129, https://doi.org/10.1256/qj.04.73.
[50] S. Ghader, A. Ghasemi, M.R. Banazadeh, D. Mansoury, High-order compact scheme for Boussinesq equations: implementation and numerical boundary condition issue, Int. J. Numer. Methods Fluids 69 (2012) 590–605, https://doi.org/10.1002/fld.2576.
[51] D. Dritschel, M. Ambaum, A contour-advective semi-Lagrangian numerical algorithm for simulating fine-scale conservative dynamical fields, Q. J. R. Meteorol. Soc. 123 (1997) 1097–1130.