A Fast Monotone Discretization of the Rotating Shallow Water Equations

Guillaume Roullet[1], **Tugdual Gaillard**[1]

[1]Univ. Brest, CNRS, IRD, Ifremer, Laboratoire d'Océanographie Physique et Spatiale (LOPS), IUEM, Brest, France

Key Points:

- Use WENO reconstructions on the mass flux and on the nonlinear Coriolis term
- Reach low level of energy dissipation and high accuracy on material conservation of potential vorticity
- Express the continuous equations with index coordinates, finite volume quantities, covariant and contravariant components of the velocity
- Minimize the number of operations and maximize the arithmetic intensity
- Achieve 2 GFlop per second per core with a pure Python code

Corresponding author: Guillaume Roullet, `roullet@univ-brest.fr`

Abstract

This paper presents a new discretization of the rotating shallow water equations and a set of decisions, ranging from a simplification of the continuous equations down to the implementation level, yielding a code that is fast and accurate. Accuracy is reached by using WENO reconstructions on the mass flux and on the nonlinear Coriolis term. The results show that the build-in mixing and dissipation, provided by the discretization, allow a very good material conservation of potential vorticity and a minimal energy dissipation. Numerical experiments are presented to assess the accuracy, which include a resolution convergence, a sensitivity on the the free-slip vs. no-slip boundary conditions, a study on the separation of waves from vortical motions. Speed is achieved by a series of choices rather than a single recipe. The main choice is to discretize the covariant form written in index coordinates. This form, rooted in the discrete differential geometry, removes most of the grid scale terms from the equations, and keep them only where they should be. The model objects appearing in resulting continuous equations have a natural correspondence with the grid cell features. The other choices are guided by the maximization of the arithmetic intensity. Finally this paper also proves that a pure Python implementation is not only possible but also very fast, thanks to the possibility of having compiled Python. As a result, the code performs 2 TFlop per second using thousand cores.

Plain Language Summary

Using a simplified model of the ocean and atmosphere dynamics, this paper presents a set of numerical and programming decisions that yields a code that is both fast and accurate. The accuracy is assessed in terms of capacity of the code to maintain dynamic structures over long periods of time while avoiding the emergence of numerical noise in the solution. Accuracy is achieved by using a very accurate discretization on two decisive terms of the model equations. Speed is achieved by a series of choices ranging from a simplification of the continuous equations down to the implementation level. This paper also proves that a pure Python code is a viable alternative to perform simulations on high performance computing centers with as much as 2 TFlop per second using thousand cores.

1 Introduction

The rotating shallow water (RSW) equations are the perfect framework to test concepts, methods and ideas for later applications to more sophisticated atmospheric or oceanic models. When it comes to numerical modeling, two goals are particularly important: speed and accuracy. They are rather antagonistic for accuracy comes with higher order schemes, which are computationally more expensive than low order ones, therefore penalizing speed. However things are more subtle because a higher order discretization in time may allow a larger time step, whereas a higher order discretization in space may increase the effective resolution, allowing to use a coarser grid. The price of having high-order discretizations may thus be largely compensated. In this paper we show how the WENO reconstruction (Jiang & Shu, 1996), a highly computationally demanding scheme, can be used in a RSW model on both the continuity and the momentum equations to provide high accuracy, while still allowing a very fast code. The merits are such that this numerical method opens the way for a new class of sub-grid-scale closure.

Having a code running fast is a very valuable quality. For a given amount of computational resources, it allows for a longer time integration or a greater spatial resolution. Achieving speed involves many design choices, rather than one, that include the programming language, the algorithms implementation and the code design in general. When measured in terms of floating point operations (Flop) per second, the speed issue rapidly touches to the hardware architecture. The question should be, for a given computer, how close is the code speed to the maximum speed achievable on this computer. The maximum speed is given by the clock frequency but, if the code involves too much data transfer between the memory and the CPU, the effective speed can be far from this maximum. Indeed, according to the roof-line model (Williams et al., 2009), the speed might be memory-bound or compute-bound, and that depends on the arithmetic intensity, which is the ratio of the number of Flop per float exchanged between the memory and the core. To achieve the optimal speed, a code should be in the compute-bound region, namely it should have a large enough arithmetic intensity, which means to perform as many Flop on the data, once the data have been transfered to the core. This issue is often overlook in atmosphere and ocean models.

Increasing the arithmetic intensity is not so easy. We are aware of at least three techniques. First, this can be done by blending many operations into a few large loops, as opposed to having many loops each doing one operation. A typical example of code following this approach is ROMS (Shchepetkin & McWilliams, 2005). This technique comes at the expend of code readability and modularity, which makes code evolutions harder, e.g. changing the time stepping. The second possibility is to use numerical discretizations that require more Flop per grid point. A very good example of such demanding computation is a a high order WENO reconstruction (Shu, 1999), which loops back on the question of accuracy. Indeed, replacing linear schemes with high-order non-linear schemes not only increases the arithmetic intensity, but it also increases the model accuracy. This is the main point of this paper. We elaborate on the benefits of WENO reconstruction below.

The third technique is to simply reduce the number of Flop, and the associated data transfer. This might sound odd but there is actually an obvious way, though neglected: strip down the RSW equations to a minimal covariant form. The discretized RSW equations, when written either in curvilinear coordinates or on non rectangular grids, are usually cluttered with a lot of grid scale factors multiplications (lengths, inverse of lengths and areas). In this paper we show how these scale factors can be removed almost everywhere in the vector invariant form of the RSW equations. The price is to slightly change the objects the code manipulates. Without further explanations yet, the changes are the following: use the array indices (i, j) as spatial coordinates, use finite volume quantities carrying their area, replace the velocity components with the pairs of covariant and contravariant components. These changes arise naturally from the discrete differential geometry (Desbrun et al., 2006; Cotter & Thuburn, 2014), which identifies the basic objects such as scalars, vectors, vorticity, as differential forms and which connects them with the grid features, respectively cells, edges and vertices, while emphasizing the crucial difference between the primal and the dual mesh. To avoid burying the ideas into an overwhelming formalism, we will start from known grounds and make the concepts emerge naturally. For the reader tempted to know more we may suggest this very tutorial paper (Perot & Zusi, 2014). The obtained simplified form of the RSW equations has many advantages. It is light, in terms of operations involved ; it is fully adapted to a discretization on a logically rectangular C-grid ; and, last but not least, it is covariant, in the sense that the form is invariant under a change of coordinates. Thanks to the covariance the space is really seen as an array of cells, even on the continuous equations.

As already mentioned, the programming language is central. Until recently the climate-atmospheric-ocean community mostly relied on Fortran and MPI. Fortran has long been considered as the ultimate language for HPC. Things are changing. New codes in Cython or Julia (Ramadhan et al., 2020) are now popping up quite regularly. But pure Python codes remain rare, mostly because Python is an interpreted language. This can now be overcome thanks to the numba module (Lam et al., 2015) that allows to compile Python. This paper proves that all the ideas presented so far can be implemented in a pure Python code, while reaching 2.0 GFlop per second on a 2.5 GHz core, and 2.0 TFlop per second on the same architecture with a thousand cores.

Finally another possibility to increase the speed is to trade it with accuracy by using single precision floats, or even a blend of a single precision and BFloats (two bytes floats), which de facto reduces the memory traffic and the time of each Flop. This approach has been recently tested quite thoroughly (Klöwer et al., 2020).

Let us now turn on the accuracy aspect. Accuracy encompasses several properties. In this paper we are particularly interested in the ability: i) to have minimal energy dissipation, ii) to materially conserve the potential vorticity (PV), iii) to maintain noise-free PV, iv) to separate vortical motions from wave motions and v) to enforce clean lateral boundary conditions, either free or no-slip. We achieve these properties with essentially one key idea: use WENO reconstructions on the mass flux and the nonlinear Coriolis term, namely the two decisive terms that control these properties.

Using a WENO reconstruction on the nonlinear Coriolis term may seem odd because the upwinding breaks the invariance under the time reversal symmetry, which unavoidably introduces dissipation. The opposite strategy for accuracy is to seek a symplectic integrator (Brecht et al., 2019). There are in fact several good reasons for using WENO. First, a close inspection of the RSW equations written in vector-invariant form reveals the equal importance in the material conservation of PV of the mass flux and the nonlinear Coriolis term, which is a vorticity flux. So if one applies a WENO reconstruction on the mass flux, to provide mixing, it is appealing to proceed similarly on the nonlinear Coriolis term to have a consistent discretization of the PV and to ensure maximum symmetry between the two fluxes. We will show that this technique brings the aforementioned properties on the PV dynamics. Second, from the energy point of view, the nonlinear Coriolis term should have a vanishing work, but if we consider the filtered version of the RSW equations in vector-invariant form, following the LES filter technique (Sagaut, 2006), then, once again, the nonlinear Coriolis term turns out to be the key player. Indeed, the vector-invariant form transforms the momentum flux into the nonlinear Coriolis term and the gradient of kinetic energy term. These two terms play a very different role on the energy equation. The gradient term becomes the divergence of the kinetic energy flux whereas the nonlinear Coriolis term turns out to be the term responsible for the exchange of energy between the resolved grid scales and the sub-grid scales. Therefore advocating for using the WENO reconstruction to compute this term. The third, and last reason was originally formulated by Mullen et al. (2011). If we let the differential geometry guides our numerical choices, then the transport of the momentum should be discretized in such way that it obeys the properties of the Lie derivative. This pleads for upwinding the vorticity in the nonlinear Coriolis term. If one also demands high order discretization and monotonicity, then a WENO reconstruction is a natural solution. Note that WENO reconstructions have already been tested for shallow water models (Xing & Shu, 2005; Noelle et al., 2007; Gallerano & Cannata, 2011) but it was on the flux form of the momentum equation. Applying it on the nonlinear Coriolis term is completely new to our knowledge.

From the more general perspective of large eddy simulations (LES) models, the idea stems from the MILES approach (Boris et al., 1992). MILES was designed for three dimensional models as an alternative to physically based explicit closures, typically the Smagorinsky closure or one of its variant. In ILES the closure takes the form of a monotonic discretization of the mass and momentum fluxes. Such closure is coined *implicit* (Margolin et al., 2006) or *numerical* (Pope, 2004). A numerical closure is opposed to a physical or purely physical closure for which there is a physical model supporting the closure. The use a monotonic discretization on the nonlinear Coriolis term rather than on the momentum flux, can be seen as a variant of the MILES approach. This paper adds up to the list of closures for LES models solving the RSW equations (Graham & Ringler, 2013).

This paper is organized as follows. In Section 2, we show how the continuous RSW equations can be strip down to a very simple form while still handling general curvilinear coordinates and being fully covariant. We discuss the material conservation of PV to motivate the discretization, which is presented in Section 3. In Section 4, implementation choices are described and the code speed is assessed. In Section 5, the accuracy of the code is tested with three experiments, each assessing one aspect. A summary is given in Section 6.

2 A fresh look at the RSW equations

The goal of this section is to present the RSW equations in the form that it is well suited for having a fast and accurate numerical model, namely

$$\frac{\partial \mathbf{u}}{\partial t} = -(\zeta^\star + f^\star)\,\mathbf{U}^\perp - \boldsymbol{\nabla}\left(g(h+b)+k\right) \tag{1}$$

$$\frac{\partial h^\star}{\partial t} = -\boldsymbol{\nabla}\cdot\left(h^\star\,\mathbf{U}\right) \tag{2}$$

$$\zeta^\star = \boldsymbol{\nabla}\times\mathbf{u} \tag{3}$$

$$k = \frac{1}{2}\mathbf{u}\cdot\mathbf{U} \tag{4}$$

which is the vector invariant form slightly in disguise. Indeed, at this stage only four terms have their classical definition: h, the layer depth, g is the acceleration due to gravity, b the bottom topography and k the kinetic energy density. The other terms require more context before being fully defined. In particular the meaning of the $\star$ decorator and the use of two different terms $\mathbf{u}$ and $\mathbf{U}$ for the velocity will be explained. The $^\perp$ symbol on $\mathbf{U}^\perp$ has its usual meaning, it designates the quarter turn counterclockwise rotated $\mathbf{U}$.

2.1 Index coordinates

We start by endowing the space with a mapping system. The most general way is to use curvilinear coordinates (η_1, η_2). They might be Cartesian (x, y), spherical (ϕ, θ), cylindrical (r, θ), or any other. Among the other possibilities are the index coordinates (i, j), associated with a logically rectangular grid. These coordinates, which are grid resolution dependent, are natural to locate grid cell features such as centers, edges, and vertices because all the variables are mapped with only integers or half integers indices, depending on the variable staggering. But their most interesting property is that two adjacent points of the same feature in the direction either i or j are separated by either $di = 1$ or $dj = 1$. Thus, the partial derivative $\partial\phi/\partial i$ of a field $\phi(i, j)$ is naturally discretized as

$$\frac{\partial \phi}{\partial i} \to \phi[i+1, j] - \phi[i, j]\,, \tag{5}$$

with no division, because $di = 1$. By using index coordinates, a spatial derivative boils down to one subtraction. This is the first optimization and simplification of this paper.

For the rest of this paper, we will use the index coordinates, therefore using (i, j) instead of (η_1, η_2). The consequence is that the ∇ operator reads

$$\boldsymbol{\nabla} = \left(\frac{\partial}{\partial i}, \frac{\partial}{\partial j}\right), \tag{6}$$

and its discretized version only involves the two points differences (5).

Once the coordinates system is defined, the space must be equipped with a metric to measure the distance between two nearby points, say P_1 at (i, j) and P_2 at $(i+di, j+dj)$. This is achieved with the first fundamental form

$$ds^2 = e_1^2\, di^2 + e_2^2\, dj^2 \tag{7}$$

where $e_1(i, j)$ and $e_2(i, j)$ describe the metric of the space. For the index coordinates system, (e_1, e_2) are the elementary distances between two points separated either by $(1, 0)$ in the direction i, or by $(0, 1)$ in the direction j. In other words, (e_1, e_2) are the grid cell lengths and they carry the length dimension. For other coordinates systems, e_1 and e_2 may not have the dimensions of a length, e.g. in the Cartesian coordinate case $(e_1, e_2) = (1, 1)$, or not have the same dimension, e.g. in the cylindrical coordinate case $(e_1, e_2) = (1, r)$.

2.2 Finite volumes and contravariant components

To present the second optimization and simplification, let us recall how the equations in curvilinear coordinates are usually written. In particular, the continuity equation $\partial h/\partial t = -\boldsymbol{\nabla} \cdot (h\tilde{\mathbf{u}})$, where $\tilde{\mathbf{u}} = (\tilde{u}, \tilde{v})$ is the velocity, reads

$$\frac{\partial h}{\partial t} = -\frac{1}{e_1 e_2}\left(\frac{\partial}{\partial i}(h\,\tilde{u}\,e_2) + \frac{\partial}{\partial j}(h\,\tilde{v}\,e_1)\right). \tag{8}$$

This equation, though absolutely correct, is unnecessarily cluttered. The drawbacks are many. Beyond the code readability, it harms the code speed because it requires unnecessary multiplications and unnecessary data transfer from the memory to the CPU, as e_1 and e_2 are also bi-dimensional arrays in the general case. It also makes the interpolation of model variables more involved. (8) can be simplified into (2), viz.

$$\frac{\partial h^*}{\partial t} = -\frac{\partial}{\partial i}(h^*\,U) - \frac{\partial}{\partial j}(h^*\,V) \tag{9}$$

with no sacrifice, by simply defining

$$h^* = h\,e_1 e_2\,, \quad \text{and} \quad \mathbf{U} = (U, V) = (\tilde{u}/e_1,\ \tilde{v}/e_2)\,. \tag{10}$$

(9) now involves only two multiplications, that correspond to a genuine non-linearity of the RSW equations, and three additions/subtractions. The grid scale factors are gone. The price to pay is to accept working with the less intuitive variables $(h^\star, \mathbf{U})$ rather than the usual "physical" $(h, \tilde{\mathbf{u}})$. The benefits are considerable: computationally, implementation wise and even conceptually. The simplification neither comes by chance or is a mathematical trick. (9) exposes the geometric nature of the objects we should manipulate. Let us comment on these two variables.

The first realization is that the velocity which fluxes the mass is $\mathbf{U}$, whose dimensions are T^{-1}. $\mathbf{U}$ turns out to be the contravariant form of the velocity in the index coordinates system. The second realization is the use of $h^\star$. As the product of h with the area $A = e_1 e_2$, $h^\star$ is naturally the *amount* of h, i.e. the finite volume version of h. The discretized version of $h^\star$ should be natural for every numerical modeler but its continuous version might be a bit more mysterious. It is worth an explanation. In the continuous equations, A is an infinitesimal surface area. In Cartesian coordinates, A would be $dx\,dy$ and $h^\star$ would be $h\,dx\,dy$. This might look awkward, but it is not, for there is a solid underlying mathematical theory: the differential geometry. In this paper we have decided to not use the artillery of differential geometry because it would overwhelm the discussion with too many concepts. However, it is with these concepts in mind that this work has been carried out. The reader interested in the connection with the differential geometry may look at these papers. The present paper is really aimed at numerical modelers. A consequence of $h^\star$ carrying its infinitesimal area is that it can be used as is in a domain integration. For instance, the total volume is $V = \int h^\star$.

2.3 Covariant components

Similarly the momentum equations in curvilinear coordinates vector-invariant form usually read

$$\frac{\partial \tilde{u}}{\partial t} = (\zeta + f)\tilde{v} - \frac{1}{e_1}\frac{\partial}{\partial i}\left(g(h+b) + \frac{1}{2}|\tilde{\mathbf{u}}|^2\right) \tag{11}$$

$$\frac{\partial \tilde{v}}{\partial t} = -(\zeta + f)\tilde{u} - \frac{1}{e_2}\frac{\partial}{\partial j}\left(g(h+b) + \frac{1}{2}|\tilde{\mathbf{u}}|^2\right), \tag{12}$$

where f is the Coriolis parameter and ζ is the vorticity

$$\zeta = \frac{1}{e_1 e_2}\left(\frac{\partial}{\partial i}(e_2 v) - \frac{\partial}{\partial j}(e_1 u)\right). \tag{13}$$

(12) can be transformed into (1), viz.

$$\frac{\partial u}{\partial t} = (\zeta^\star + f^\star)V - \frac{\partial}{\partial i}\left(g(h+b)+k\right) \tag{14}$$

$$\frac{\partial v}{\partial t} = -(\zeta^\star + f^\star)U - \frac{\partial}{\partial j}\left(g(h+b)+k\right) \tag{15}$$

by defining

$$\mathbf{u} = (u, v) = (\tilde{u}\, e_1, \tilde{v}\, e_2)\,, \tag{16}$$

and

$$f^\star = f\, e_1 e_2\,, \quad \zeta^\star = \frac{\partial v}{\partial i} - \frac{\partial u}{\partial j} \quad \text{and} \quad k = \frac{1}{2}\mathbf{u}\cdot\mathbf{U}\,. \tag{17}$$

As in the continuity equation, no grid lengths are involved in either the gradient or the curl. The vector $\mathbf{u}$ has two interpretations: it is both a circulation element, and the covariant form of the velocity in the index coordinates system. By combining the definitions of $\mathbf{u}$ and $\mathbf{U}$ we have $(u, v) = (U\, e_1^2, V\, e_2^2)$. This relation can be written in tensor notation $\mathbf{u} = \mathbf{g}\,\mathbf{U}$, with

$$\mathbf{g} = \begin{pmatrix} e_1^2 & 0 \\ 0 & e_2^2 \end{pmatrix} \tag{18}$$

the metric tensor. The dimensions of the covariant components are $L^2\,T^{-1}$. Therefore neither $\mathbf{u}$ or $\mathbf{U}$ have the dimensions $L\,T^{-1}$ of a speed. The distinction between $\mathbf{u}$ and $\mathbf{U}$ may seem quite artificial and formal at first. It turns out that they correspond to two very different substances: $\mathbf{u}$ is the momentum, the dynamical quantity that is transported and that obeys a conservation law, whereas $\mathbf{U}$ is the flux, the kinematic quantity that transports things. $\zeta^\star$ has the same dimensions as $\mathbf{u}$ and satisfies $\zeta^\star = \zeta\, e_1 e_2$. Consequently $\zeta^\star$ can be seen either as an elementary circulation along a closed loop, or as the usual vorticity times the area element, i.e. the finite volume version of ζ. Likewise, $f^\star$ is the finite volume version of the planetary vorticity f. At this stage, (9-10, 14-17) are in the form we use for the discretization.

2.4 Potential vorticity

A central diagnostic quantity of the RSW equations is $q = (\zeta^\star + f^\star)/h^\star$, the potential vorticity, abbreviated PV throughout this paper. PV plays a central role in rotating flows for it allows to split the dynamics into a balanced part, captured by the PV evolution, and the unbalanced, the gravity waves, that propagate with vanishing net PV transport. Being a ratio of two finite volume quantities, q is a density, as opposed to a finite volume quantity. It obeys $\partial q/\partial t+\mathbf{U}\cdot\boldsymbol{\nabla}q=0$, which expresses the material conservation on fluid parcels. This conservation law is highly desirable at the numerical level. It should be emphasized that the material conservation is much more demanding numerically than a global conservation. In practice, it means that the probability density function of q remains stationary in time. Ensuring exact material conservation of this derived quantity is possible on steady flows, e.g. the cases 2 and 3 of (Williamson et al., 1992), but it is impossible on arbitrary flows, for a fundamental reason. Indeed, the material conservation holds as long as there is no dissipation nor mixing, viz. for inviscid flows but, sooner or later, mixing of PV kicks in. This is because of the tendency for the PV to develop filaments that, under the flow deformation, elongate and get thinner with time, a process known as the direct cascade of enstrophy. For RSW equations the enstrophy density is $q^2\,h$ and for inviscid flows, the total enstrophy, integrated over the domain, $Z = \int q^2\,h^\star$ should be conserved. In a numerical model the direct cascade of enstrophy should proceed as inviscidly as possible across the resolved scales until it reaches the grid scale, at which point the numeric should be helped to parameterize the unresolved cascade continuation. This parameterization usually boils down to dissipate the enstrophy at the grid scale. In this paper we adopt the MILES approach consisting in using monotonic upwinded reconstructions to provide the required dissipation of enstrophy. But the tricky point is that q is essentially a by-product of the equations, there is no direct handle on the PV evolution. The PV dynamics is controlled only through the dynamics of $h^\star$ and $\omega^\star=\zeta^\star+f^\star$, the finite volume absolute vorticity. To complicate even more, $\omega^\star$ is also a derived quantity, but fortunately, the vector invariant form exposes the $\omega^\star$ dynamics in plain sight offering a way to consistently handle $h^\star$ and $\omega^\star$.

The numerical discretization we propose aims at having a PV material conservation as good as possible. The material conservation is not a mere coincidence, it corresponds to a hidden symmetry of the equations: the invariance of the equations under a relabeling of the parcels. Enforcing material conservation discretely is thus a way to satisfy this hidden symmetry of the equations. For that we adopt a slight change of perspective on the role of q in the numerical integration. Instead of focusing on q, we focus on $\omega^\star$. Indeed, in practice, the material conservation of PV derives from a subtle cancellation in the momentum and the continuity equation between the vorticity flux $\omega^\star\mathbf{U}$ and the mass flux $h^\star\mathbf{U}$. Let us carefully examine how this cancellation works for this will suggest a new way to discretize the RSW equations.

To derive the material conservation of PV we apply the chain rule on

$$\frac{\partial q}{\partial t} = \frac{1}{h^{\star 2}}\left(h^\star \frac{\partial \omega^\star}{\partial t} - \omega^\star \frac{\partial h^\star}{\partial t}\right) \tag{19}$$

that reveals the very symmetrical role between the continuity equation and the the equation for the absolute vorticity. The latter is derived by taking the curl of (1), namely

$$\frac{\partial \omega^\star}{\partial t} = -\boldsymbol{\nabla}\cdot(\omega^\star\,\mathbf{U}), \tag{20}$$

which expresses that the vorticity obeys a conservation law in flux form, exactly like $h^\star$. Substituting (20) in (19) yields

$$\begin{aligned}
\frac{\partial q}{\partial t} &= \frac{1}{h^{\star 2}}\left(-h^\star\boldsymbol{\nabla}\cdot(\omega^\star\,\mathbf{U}) + \omega^\star\boldsymbol{\nabla}\cdot(h^\star\mathbf{U})\right) && (21)\\
&= -\frac{1}{h^{\star 2}}\left[h^\star\mathbf{U}\cdot\boldsymbol{\nabla}\omega^\star - \omega^\star\mathbf{U}\cdot\boldsymbol{\nabla}h^\star + \underbrace{h^\star\omega^\star\boldsymbol{\nabla}\cdot\mathbf{U} - h^\star\omega^\star\boldsymbol{\nabla}\cdot\mathbf{U}}_{=0}\right] && (22)\\
&= -\mathbf{U}\cdot\boldsymbol{\nabla}q\,. && (23)
\end{aligned}$$

We see that the material conservation arises because of the cancellation of the two terms in (22), which follows from the identity

$$\boldsymbol{\nabla}\cdot(\phi^\star\,\mathbf{U}) = \mathbf{U}\cdot\boldsymbol{\nabla}\phi^\star + \phi^\star\boldsymbol{\nabla}\cdot\mathbf{U}\,, \tag{24}$$

where $\phi^\star$ is either $h^\star$ or $\omega^\star$. On a C-grid, the discrete version of this identity can be made exact provided the quantity $\phi^\star$, in the flux $\phi^\star\,\mathbf{U}$, is interpolated at velocity point.

The discretization we propose is now clear: use monotonic high-order biased reconstruction of $\omega^\star$ and $h^\star$ to estimate the terms $\omega^\star\mathbf{U}^\perp$, in the momentum equation, and $h^\star\mathbf{U}$, in the continuity equation. For the mass flux, this is the usual upwind interpolation. For the nonlinear Coriolis term the upwinding should be done in the direction of the flux, namely $\mathbf{U}^\perp$, not in the direction of the momentum on which it applies. This is the main originality of this paper.

3 Discretization

We now present the model discretization by going through three aspects: the space and time discretizations ; and the handling of the boundary conditions.

3.1 Space discretization

The model equations are discretized on a logically rectangular C-grid. In the C-grid there is a natural distinction between the *primal* grid and the *dual* grid (Figure 1a).

In this paper we chose to map the primal grid centers with integer indices and the dual grid centers with half integer indices. The velocity components, both covariant and contravariant, are defined on the edges of the dual grid, $h^\star$ is defined at cell centers of the primal grid and the vorticity terms $\zeta^\star$ and $f^\star$ are defined at cell centers of the dual grid, which are also the vertices of the primal grid. The rotated $\mathbf{U}^\perp$ is defined on the edges of the primal grid, which implies that its components are staggered compared to the components of $\mathbf{U}$ (Figure 1a). Following the C-grid terminology, we denote "u-point" and "v-point" the place where u and v are discretized.

Because we use the index coordinates (i, j), the model equations are completely oblivious to e_1 and e_2, the grid scale factors, which means that for u, v and $h^\star$ the space is seen as an array of regular indices, regardless of the underlying metric. Consequently the grid cells are truly squares, of size 1×1 in the index units. This also means that spatial interpolations, involved in the evaluation at non native locations, should be done on the regular grid of indices, not on the irregular grid of spatial locations.

Before giving the discretized equations we define three spatial operators. The first one is the finite difference operator

$$\delta_{i+1/2}[\phi] = \phi_{i+1} - \phi_i \tag{25}$$

estimating the along i partial derivative of ϕ at location $i+1/2$ assuming ϕ is discretized at integer locations along i. The converse is also needed $\delta_i[\phi] = \phi_{i+1/2} - \phi_{i-1/2}$ to estimate a partial derivative at location i using a quantity discretized at half integer locations. To designate an along j partial derivative we should use either $\delta_j[\phi]$ or $\delta_{j+1/2}[\phi]$.

The two others are interpolation operators, interpolating along direction i (or along the j direction, with the j index). The first one is the linear second order, or two points averaging

$$\overline{\phi}^{i+1/2} = \frac{1}{2}\left(\phi_i + \phi_{i+1}\right) , \tag{26}$$

and the second is the WENO reconstruction

$$I_{i+1/2}[\phi^\star, U] = U \sum_{s \in S} c_s \, \phi^\star_{i+s} \, , \tag{27}$$

where S is the stencil of the reconstruction and c_s are the weights. In this paper we use n-order WENO reconstructions (Jiang & Shu, 1996; Shu, 1999), $n \in \{1, 3, 5\}$, whose stencils have n elements. The $n = 1$ case is the first order upwind interpolation and we have either $c_0 = 1$ if $U > 0$, or $c_1 = 1$ if $U < 0$. In the $n = 3, 5$ cases the reconstruction is nonlinear because the weights c_s depend on $\{\phi_{i+s},\, s \in S\}$. In any case $\sum_{s\in S} c_s = 1$. Having an even number of points, the stencils are shifted in the upwind direction, which depends on the sign of U. In the 5th order case, $S = \{-2, -1, 0, 1, 2\}$ if $U > 0$, and $S = \{-1, 0, 1, 2, 3\}$ if $U < 0$. As defined, the interpolation operators use quantities discretized at integer locations (ϕ_{i+s}) to estimate it at $i+1/2$. The reverse is also needed: use quantities discretized at half integer locations to estimate it at i. In that case we would write either $\overline{\phi}^i$ or $I_i[\phi^\star; U]$. Note that (27) assumes that U is discretized at the location where $\phi^\star$ is reconstructed. This will always be the case. For sake of completeness the WENO reconstruction is detailed in the appendix. Note that (27) is referred as a *reconstruction* rather than an *interpolation*. Reconstruction is the word used when the quantity to be interpolated is a *finite volume* quantity, and interpolation is usually reserved when the interpolated quantity is the density (e.g. h or ω), or equivalently the finite difference quantity. In this paper, the WENO scheme is applied to $h^\star$ and $\omega^\star$, the finite volume quantities. We therefore exclusively use the WENO reconstruction.

With these notations defined, we can now give the discretized model equations. They read

$$U = u/e_1^2, \qquad V = v/e_2^2, \qquad B = g(b + h^\star/A) + k\,, \tag{28}$$

$$\begin{aligned}
\boldsymbol{\nabla} B &\rightarrow (\delta_{i+1/2}[B],\, \delta_{j+1/2}[B]) &(29)\\
\boldsymbol{\nabla}\cdot(h^\star \mathbf{U}) &\rightarrow \delta_i[I_{i+1/2}[h^\star, U]] + \delta_j[I_{j+1/2}[h^\star, V]] &(30)\\
\omega^\star = f^\star + \boldsymbol{\nabla}\times\mathbf{u} &\rightarrow f^\star + \delta_{i+1/2}[v] - \delta_{j+1/2}[u] &(31)\\
k = \frac{1}{2}\mathbf{u}\cdot\mathbf{U} &\rightarrow \frac{1}{2}\left(\overline{uU}^i + \overline{vV}^j\right) &(32)\\
\omega^\star\,\mathbf{U}^\perp &\rightarrow (I_j[\omega^\star, V_m]\,, -I_i[\omega^\star, U_m]) &(33)
\end{aligned}$$

and

$$U_m = \overline{\overline{U}^i}^{j+1/2}, \qquad V_m = \overline{\overline{V}^j}^{i+1/2}. \tag{34}$$

The only place where the metric terms are used is in (28). The required metric terms are (e_1, e_2), the edge lengths of the dual, at respectively u-points and v-points, and A the primal cell area. In addition, and only during the initialization, A_v, the dual cell area, is needed to define $f^\star = A_v\, f$.

Because of the rotation, the components of $\mathbf{U}^\perp = (-V, U)$ must be deduced from $\mathbf{U}$ through some averaging because U and V are discretized at $(i + 1/2, j)$ and $(i, j +$

1/2) whereas $\mathbf{U}^\perp$ requires flipped locations. This is done with the four points averaging in (34). This averaging is one of the decisive ingredient of the TRISK discretization (Thuburn et al., 2009; Ringler et al., 2010).

The discretization (33) is the main originality of this paper. The WENO reconstruction is usually applied on conservation laws written in flux form. In the case of the momentum equation this is on the flux of momentum. Here, because of the vector invariant form, there is no momentum flux. But, as discussed earlier the nonlinear Coriolis term is the vorticity flux and, as such, it can be computed with a WENO reconstruction.

The idea of putting some kind of upwinding and monotonicity on the nonlinear Coriolis term is not new. In some aspect, the anticipated PV method (APVM) (Sadourny & Basdevant, 1985) implements it, although in a quite different fashion. APVM has been compared to other sub-grid closures (Graham & Ringler, 2013) in the context of RSW models. The APVM consists in expliciting the PV in the vorticity term ($\omega = qh$) and in using a first order upwind interpolation of the PV in the local direction of the flow. In the APVM there is no directional splitting. The APVM can be seen as a semi-lagrangian method where the PV is estimated at the place where it was a time step earlier. As being a first order interpolation, the APVM induces more enstrophy dissipation than the method presented in this paper, while being energy-conserving. Also, contrary to our method that is parameter free, the original APVM introduces a numerical parameter that must be tuned with respect to the grid size and time step. A parameter-free extension of the APVM has been proposed (Chen et al., 2011) for the small h deviations case, whose assumption our method does not require. Finally, the use of the finite volume $\omega^\star$ makes the APVM completely unnatural within the framework we propose. Indeed, it would require to write $\omega^\star\mathbf{U}^\perp$ as $q\,(h^\star\mathbf{U}^\perp)$. We would then use a centered discretization for the mass flux $h^\star\mathbf{U}^\perp$ and have the upwinding on the q term. This would completely break the symmetry of treatment between the continuity and the momentum equation. With this in mind, the discretization we propose appears oppositely quite natural, almost as self emerging from the equations, without ad-hoc choice and parameter-free.

For the kinetic energy term we use (32), which is a classical discretization. However, it is worth noting that the kinetic energy term could also be discretized with a WENO reconstruction, as follows

$$\overline{uU}^i + \overline{vV}^j \to s_u\, I_i[uU, s_u] + s_v\, I_j[vV, s_v] \qquad (35)$$

with $s_u = \mathrm{sign}(\overline{U}^i)$ and $s_v = \mathrm{sign}(\overline{V}^j)$ inside the operator, to keep track of the upwinding directions at the grid center, and as prefactors, to ensure positivity even if $s_u < 0$ or $s_v < 0$. Doing so would seriously increase the number of FLOP per time iteration (see Section 4). Given the lack of obvious immediate benefit, we did not pursue this idea further.

The use of WENO for the kinetic energy term may look surprising, at least for the reader not familiar with the differential geometry. The differential geometry identifies the spatial derivative in the flow direction of a quantity as the Lie derivative of its associated differential form (Frankel, 2011). The Cartan identity splits the Lie derivative in two terms, each one participating to the transport in a very specific way. For the momentum, the associated differential form is $\mathbf{u}$, and these two terms are the nonlinear Coriolis term and the gradient of kinetic energy of the vector-invariant form. Each term can be seen as a composition of two basic operations of the differential geometry: the exterior derivative and the interior product, respectively a generalization of the $\boldsymbol{\nabla}$ operator, and of the inner product. In the discretized equations we presented, the exterior derivative shows up as the finite difference operators $\delta_i[\cdot]$ and $\delta_j[\cdot]$, whereas the interior product shows up as the WENO reconstruction operators $I_i[\cdot]$ and $I_j[\cdot]$. The idea of using Cartan identity to discretize the transport of a vector field was pioneered by Mullen et al. (2011), who also showed that a WENO reconstruction improves the accuracy, compared to the upwind first order reconstruction. Here we somehow generalize these results to the full RSW equations.

3.2 Time stepping

The code clearly separates the time scheme in one generic module. The implementation of a time scheme is very close to a textbook presentation. This is made possible because space and time discretizations are independent. The model variables are split into two groups: the prognostic variables $\phi = (u, v, h^\star)$, obeying an explicit time evolution equation, and the diagnostic variables $\Phi = (\zeta^\star, U, V, p)$. Formally, the code handles $s = (\phi, \Phi)$ the model state, as a collection of all variables, that obeys $\partial s/\partial t = \mathsf{L}(s)$, with L the model operator. To better expose the prognostic and the diagnostic parts L

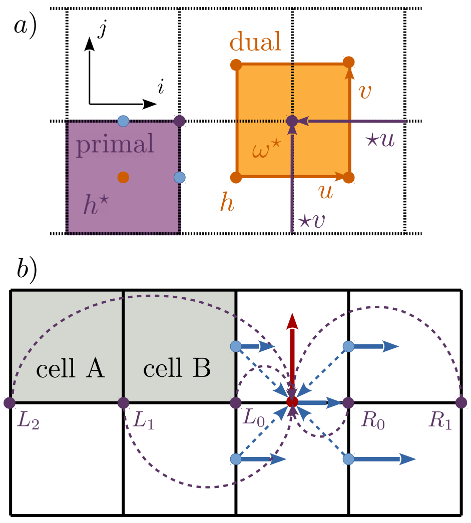

Figure 1. a) The classical C-staggering with locations for scalars (orange circles), vector components (blue circles) and vorticity (purple circles). The features (vertices, edges, faces) are colored in purple for the primal grid and orange for the dual grid. b) Illustration of the vorticity upwinding. Away from the boundaries, the vorticity flux (red arrow) at the v-point (red circle) is computed as the product of U (blue arrow), interpolated with a four points averaging (dotted blue arrows), and $\omega^\star$ reconstructed along i using the five points stencil $(L_2, L_1, L_0, R_0, R_1)$. If cell A is masked, the stencil is shortened (L_1, L_0, R_0) ; if both cell A and cell B are masked, the stencil is (L_0). In that latter case, if U were to the left, the vorticity would be reconstructed with the stencil (R_1, R_0, L_0).

can be expanded into

$$\frac{\partial \phi}{\partial t} = \mathsf{R}[\phi, \Phi] \tag{36}$$

$$\Phi = \mathsf{D}[\phi]\,, \tag{37}$$

with R the right hand side for the prognostic variables and D the diagnostic relations for Φ. Currently the code proposes two time schemes: the Leap-Frog Adams Moulton scheme (LFAM3) (Shchepetkin & McWilliams, 2005) and the 3rd order stably strongly preserving Runge Kutta scheme (Gottlieb et al., 2001) (RK3)

$$s^{(1)} = s^n + \Delta t\,\mathsf{L}[s^n] \tag{38}$$

$$s^{(2)} = s^n + \frac{1}{4}\Delta t\,(\mathsf{L}[s^n] + \mathsf{L}[s^{(1)}]) \tag{39}$$

$$s^{n+1} = s^n + \frac{1}{6}\Delta t\,(\mathsf{L}[s^n] + \mathsf{L}[s^{(1)}] + 4\,\mathsf{L}[s^{(2)}])\,, \tag{40}$$

where Δt is the time step and s^k the model state at time step k. Both are third order in time. The LFAM3 is a predictor corrector scheme with only two calls to the right-hand side L per time iteration, whereas RK3 requires three calls to the right-hand side. RK3 is the model default choice.

3.3 Boundary conditions at lateral boundaries

RSW models are quite often tested either in bi-periodic domains or on the whole sphere, more rarely in domains with lateral boundaries. For oceanic applications, handling the lateral boundaries is a necessity. The other reason to present the lateral boundary conditions is that they fit particularly well with the choice of upwinding the vorticity in the nonlinear Coriolis term. The no-flow is enforced at no cost thanks to the C-grid but interestingly, the free-slip and the no-slip boundary conditions appears very naturally as conditions on the vorticity, which directly impact the normal component of flux of vorticity at the boundary.

Solid boundaries can be either at the domain boundary or inside the domain. For the latter case, we use a mask system $m_{i,j}$. A cell (of the primal grid) is solid if $m_{i,j} = 0$, fluid if $m_{i,j} = 1$. The no-flow boundary condition is imposed at each edge of the primal grid where one adjacent cell is solid. It simply consists in setting $u = 0$ or $v = 0$ at this edge. This is the standard technique. The real point of attention is on defining $\zeta^\star$ at points sitting along the boundary. The curl expression (31) cannot be immediately used because the dual cell is not fully fluid. However, $\zeta^\star$ conserves its physical meaning of being both the amount of vorticity in this partial cell, and the circulation along the boundary of this partial cell. The latter offers the natural way to define $\zeta^\star$, which completely depends on the slip condition. In the free-slip case, $\zeta^* = 0$ at points along the boundary. In the no-slip case, we keep compute ζ^* with (31) but we set $u = 0$ and $v = 0$ for all edges, not fully in the fluid. For a straight boundary, say along i at $j = 0$ and the fluid being for $j > 0$, this definition yields $\zeta^\star_{i,0} = -u_{i,1/2}$, which expresses that a right-going flow generates a negative vorticity. The use of the differential forms remove, once again, all the metric terms from the relation. The no-slip boundary condition behaves as a source of vorticity localized at the boundary. Interestingly, once this vorticity is generated, it might be transported into the fluid by the nonlinear Coriolis term. Let us see how.

For cell edges close to the boundary, the five points stencil of the the WENO 5th does not fit in. To overcome this issue, the code implements a varying stencil width with the following policy: use the widest biased stencil, i.e. either one, three or five points, fitting within the fluid cells. The one point stencil is the upwind first order interpolation. For the three points stencil, we use the third order WENO reconstruction (Shu, 1999) (explicited in Appendix A). The consequence is that the outward vorticity flux at the edge next to the boundary is computed with a first upwind scheme, that adds a little bit more of dissipation. In the free-slip case, since $\zeta^\star = 0$ at the boundary there is no outward flux.

Instead of imposing the velocity at the boundary, and therefore the vorticity, we may want to impose the normal stress. In that case, it requires to introduce a viscosity to relate the stress to the velocity. We did not pursue this idea further as it is beyond the scope of the paper.

4 Speed

4.1 Implementation choices

Performances on both the quality of the solutions and the speed were the top priorities in the code design. To achieve speed, a compiled language is required. Until recently this imposed the use of Fortran or C or a blend of Python and C (Pressel et al., 2015). The Julia language is currently used on several projects (Ramadhan et al., 2020; Klöwer et al., 2020), whose chief advantage is to be a compiled language. Here we chose another route. The code is entirely written in Python, without sacrificing speed. This is possible thanks to the numba module. Numba (Lam et al., 2015) uses the LLVM compiler (Lattner & Adve, 2004) to compile Python code. The bulk of the code is interpreted Python, but all the computational functions are compiled. In practice, to compile a function amounts in specifying its signature, namely the types of its inputs and outputs. Inside a function, and contrary to the pythonic policy, the loops can be explicitly developed ; the compiler takes care of them. Note that since Julia's compiler is also LLVM, it makes the use of Julia less decisive for HPC.

The second element of speed is to systematically duplicate all arrays. Arrays are thus stored in `[k,j,i]` and in `[k,i,j]` conventions, the `k` index being for the layer index. The motivation is to always do finite differences with the convention where the data are contiguous in memory. Thus, the computation of the spatial derivative $\partial/\partial j$ or of the along j-interpolation is done with the `[k,i,j]` convention. Data contiguity allows a better usage of the L1-cache, which is the fastest. The price of duplicating is to perform transpose operations to exchange the data from one convention to the other. Fortunately, the transpose operation is very fast as it is highly optimized. In practice the transpose operation is done 20 times per time stage, which represents 16% of the total time. Another advantage of this approach is to easily guarantee the numerical isotropy. The two directions i and j are treated absolutely equivalently because there is only one function for both operations. This is particularly convenient for the WENO reconstruction. The WENO reconstruction, even though requiring many more operations than the linear interpolation, is not much slower.

A third element of speed is of course the use of the covariant equations with the index coordinates that turn spatial differentiations into subtractions. The number of multiplications is minimal. The only computation involving many multiplications is the WENO reconstruction.

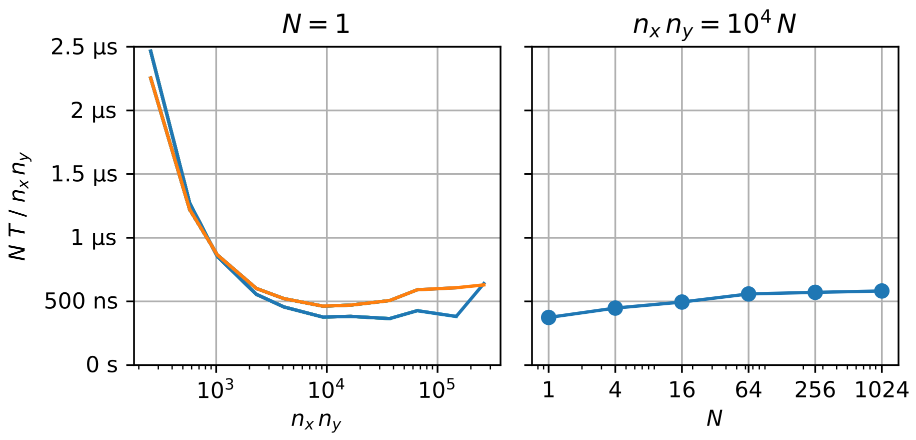

Figure 2. Walltime per time iteration T rescaled with $N/(n_x n_y)$. On the left for the monocore case as a function of the domain size $n_x n_y$ and on the right the weak scaling, where the number of cores N is increased from N=1 up to N=1024, while the domain size per core $n_x n_y$=10^4 is kept constant. In blue are the performances for the supercomputer and in orange for a notebook (see text for the CPU specs).

4.2 Speed assessment

With these choices, the code (Roullet, 2021) is very fast. With the default 5th order WENO, and the SSP-RK3 time scheme, the code speed is about $0.3\,\mu s$ per iteration per grid point (Figure 2). The performances have been measured on a notebook (Intel® Core™ i7-6600U at 2.6GHz) and Rome (AMD® EPYC 7502 at 2.5GHz) a supercomputer hosted at TGCC (Saclay, France). Timing has been averaged over thousand time iterations and excludes the I/O. The code has an almost perfect weak scaling (Figure 2b). We may wonder how close is the code speed to the peak CPU performances. To answer it we need to determine how many floating operations are done per grid cell and per time iteration.

To estimate how far this speed is from the maximum peak performance of the CPU, we count all the floating point operations (Table 1). The current implementation uses 840 Flop per time step and per grid point. With the minimum time $T = 400$ ns on the Rome supercomputer, this gives 2.1 GFlop s^{-1}, which corresponds to 84% of the CPU clock frequency. It is not overstated to say that this implementation is close to the maximum hardware limits. Interestingly, with 708 Flop, the WENO reconstruction is the major contributor, representing 84% of the total Flop. By using a linear interpolation (upwind 5th), the reconstruction involves only 108 Flop[1] and therefore only 240 Flop per time step. Naively we could expect the code to be 840:240=3.5 time faster. This is not the case. In practice the linear interpolation gives $T \approx 350$ ns corresponding to 0.7 GFlop s^{-1}. The reason is clear. In this case the code speed is limited by the memory access, which makes the CPU waiting for data. By increasing the arithmetic intensity, the use of WENO puts the code into the compute-bound region, which maximizes the Flop per second.

5 Accuracy

The merits of the numerical choices are tested with three experiments, each testing one aspect: the merging of two vortices, the interaction of a dipole in an elliptical domain with free and no-slip condition, a dam break experiment in an annulus. The experiments are set in quite intense nonlinear regimes, although not going up to either shock wave formation or dry bed emergence.

[1] Corresponding to 3 stages, 2 functions, 2 directions, 5 multiplications and 4 additions.

Table 1. Number of floating operations[a] breakdown

Function	Term	#M	#A
Continuity	WENO 5th	64	54
	$\mathrm{d}(h^\star \mathbf{U})$	2	4
Vorticity flux	WENO 5th	64	54
	$\mathbf{U}^\perp$	2	4
	Coriolis	2	4
	$\omega^\star \mathbf{U}^\perp$	2	2
Bernoulli	grad	0	4
Diagnostics	$\mathbf{U}$	2	0
	$\zeta^\star$	0	3
	$\mathbf{u} \cdot \mathbf{U}/2$	3	3
	$g(h^\star + h_b^\star)/A$	2	1
Total per stage		143	133
SSP RK3	stage 1	1	1
	stage 2	2	2
	stage 3	3	3
Total per time step		435	405

[a]Floating points operations involve multiplications (#M) and additions/subtractions (#A). The numbers are given per grid point and per call to the function. For the RK3 time stepping, which is the default, the total number per time step is the three times the total per stage plus the operations in the time scheme itself.

5.1 Merging of two vortices

This first set of experiments tests the sensitivity of the conservation laws on the model resolution. The experiments consist in the time evolution of two Gaussian vortices initially in geostrophic balance. The two vortices, of radius $\sigma = 0.07$, are separated by a distance $d = 1.4\,\sigma$; specifically

$$h(x, y) = H + h_0\ (G(x - d/2, y; \sigma) + G(x + d/2, y; \sigma))\ ,$$

with $h_0 = 0.2$, $H = 1$ and $G(x, y; \sigma) = \exp[-(x^2 + y^2)/(2\sigma^2)]$. The domain is square, with an edge length $L = 1$. We use the free-slip boundary condition. The two physical parameters are $g = 1$ and $f = 5$ which yield a Rossby deformation radius $R = \sqrt{gH}/f = 0.2$, that sets the vortices in the submesoscale range. The speed scale is $gh_0/(f\sigma)$, which yields a Rossby number of $Ro = gh_0/(f\sigma)^2$, namely $Ro \approx 1.6$, again typical of the submesoscale regime. The vortices are anticyclones because $h_0 > 0$. The flow is integrated up to time $t = 10$. The domain is meshed with N^2 grid cells of uniform size. N is varied from $N = 100$ to $N = 3{,}200$, by a succession of doubling.

The two vortices are close enough to merge, as revealed by the presence of single core of negative PV in the center at $t = 10$ (Figure 3), instead of two initially. The details of the merging sequence depend on the resolution, among which the amount of filaments and the balancing time. But quite clearly, and fortunately, the solution converges with increasing N. The cases $N = 1{,}800$ and $N = 3{,}200$ are almost indistinguishable by eye. A striking property is the absence of noise on the PV fields, for all resolutions. This is a consequence of the implicit dissipation and mixing provided by the MILES approach. A second striking feature is the capability for the code to produce and maintain very thin filaments. Of course the case $N = 3{,}200$ is quite extreme for such a trivial flow but nevertheless it is worth emphasizing. Not only are the filaments thin, they can also be intense in terms of PV difference with the background state. This results in the shear instability of a few filaments, as seen on the $N = 1{,}800$ case.

To better assess the convergence with the resolution we diagnosed the cumulative global dissipation (Fig. 4a-b) for both the energy $\epsilon_E = (E_0 - E)/E_0$ and the enstrophy $\epsilon_Z = (Z_0 - Z)/Z_0$, where the superscript 0 denotes the value at $t = 0$. The global energy E is defined as $E = \int eh^\star - E_b$, with the energy density

$$e = \frac{1}{2}\mathbf{u} \cdot \mathbf{U} + \frac{1}{2}gh\,, \tag{41}$$

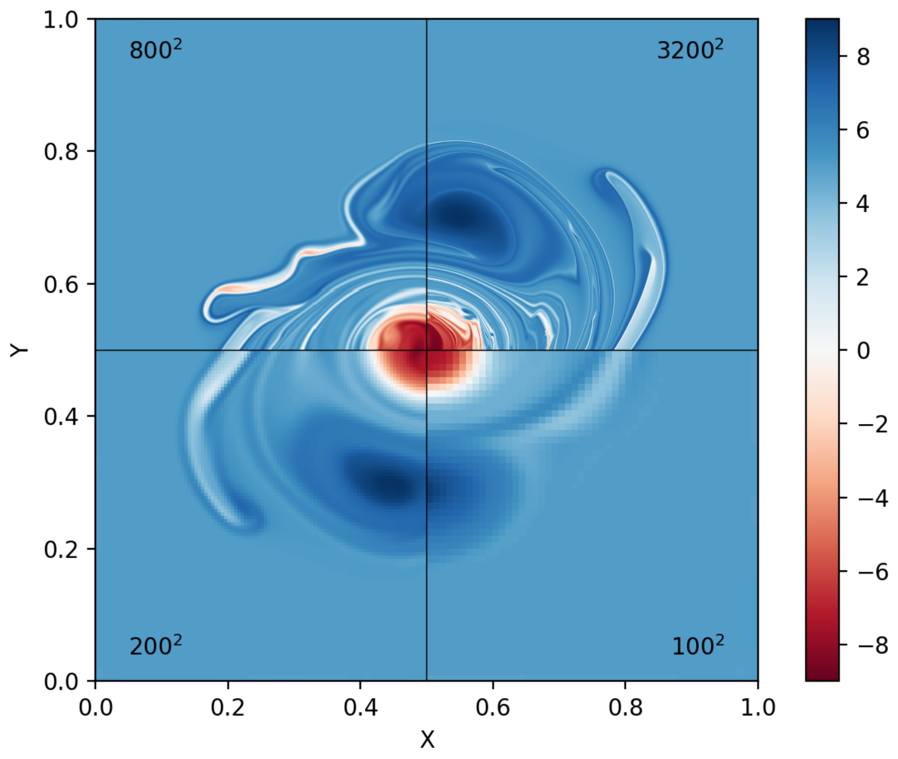

Figure 3. Snapshots of potential vorticity at t=10, after the vortices merged, for four resolutions $n_x n_y$= 100^2, 200^2, 800^2, $3{,}200^2$. Only a quarter of each domain is displayed. The parameters are g=1, H=1, f=5. The anticyclones were initially Gaussian, in geostrophic balance, with a layer depth h=1.3 at their center.

and the background potential energy $E_b = \int (gH^2/2)\, dxdy$. E_b is removed from $\int eh^\star$ because it is the part of the energy that cannot be dissipated. This is the energy of the rest state. Still, the total amount of energy dissipated is fairly small: $\epsilon_E \sim 2\%$ for $N = 100$ and $\epsilon_E \sim 4.10^{-5}$ for $N = 3{,}200$, which is approaching perfect conservation. The code actually reaches the point where the question becomes theoritical: during a vortex merging event should the energy dissipation go to zero in the limit of infinite resolution? The present experiments suggest that not but this would deserve a more thorough study, beyond the scope of this paper. The case of the enstrophy dissipation (Fig. 4b) is very different. In all cases there is a finite amount of dissipation but the increase of resolution delays the time at which the dissipation really starts, as well as it increases the equilibration time. In the $N = 100$ case, the merging process is almost completed as indicated by the plateau, at the largest resolution, there is still a lot of enstrophy to

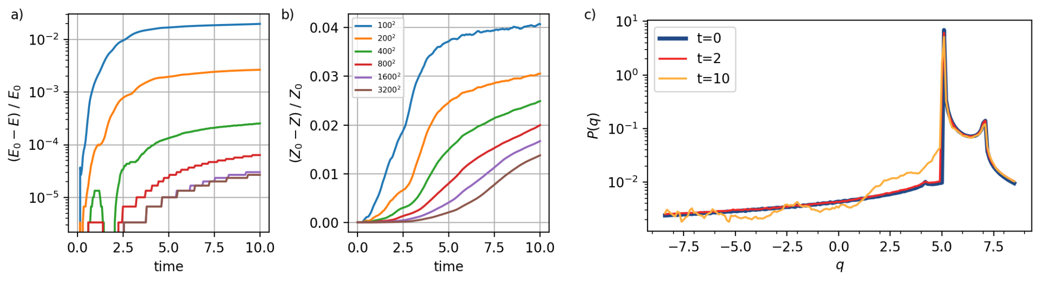

Figure 4. Energy (a) and enstrophy (b) dissipated as a function of time and resolution (color) for the vortex merging experiment. The energy of the rest state $gH^2/2$ has been removed from E and E_0. (c) probability density function of potential vorticity at $t = 0$ (blue), $t = 2$ (red) and $t = 10$ (orange) for the merging experiment at the 3,200^2 resolution.

be dissipated. It is not clear whether all resolutions yield the same amount of enstrophy dissipation. Again this requires a more thorough study that we postpone for a later paper.

Finally to assess the material conservation of PV we plot the probability density function of PV in the $N = 3{,}200$ case for $t = 0$, $t = 2$ and $t = 10$ (Fig. 4c). At $t = 2$ the enstrophy dissipation has not yet started (Fig. 4b) meaning the flow is still inviscid, even though the vortices are already producing filaments (not shown). The pdf of PV is remarkably close to its $t = 0$ value. Material conservation is very well ensured. At $t = 10$ the pdf departs from its initial value. This is due to the mixing at the grid-scale. Interestingly the PV on the cyclonic part (the initial vortices are slightly shielded with a ring of cyclonic PV) remains quite well conserved. This confirms the visual impression of the snapshot (Fig. 3), the cyclonic PV does not filament, therefore it does not mix, therefore its pdf should remain constant in time, as it does.

5.2 Vortex-wall interaction

In this set of experiments we test how the code performs on handling boundary conditions. The experiments consist in the time evolution of a vortex dipole with either the free-slip or the no-slip boundary condition (Figure 5). In order to have a variety of boundary shapes at the grid scale, the domain is elliptical, whose major and minor axis lengths are respectively 2 and 1. The grid is 1,600×800 with square grid cells. The experiments are started at $t = 0$ with two Gaussian vortices initially in geostrophic balance at the center of the domain. The two vortices, of radius $\sigma = 0.1$, are separated by a distance $d = 1.1\,\sigma$; specifically

$$h(x, y) = H + h_0\ (G(x - d/2, y; \sigma) - G(x + d/2, y; \sigma))\ ,$$

with $h_0 = 0.15$ and $H = 1$. The two physical parameters are $g = 1$ and $f = 5$.

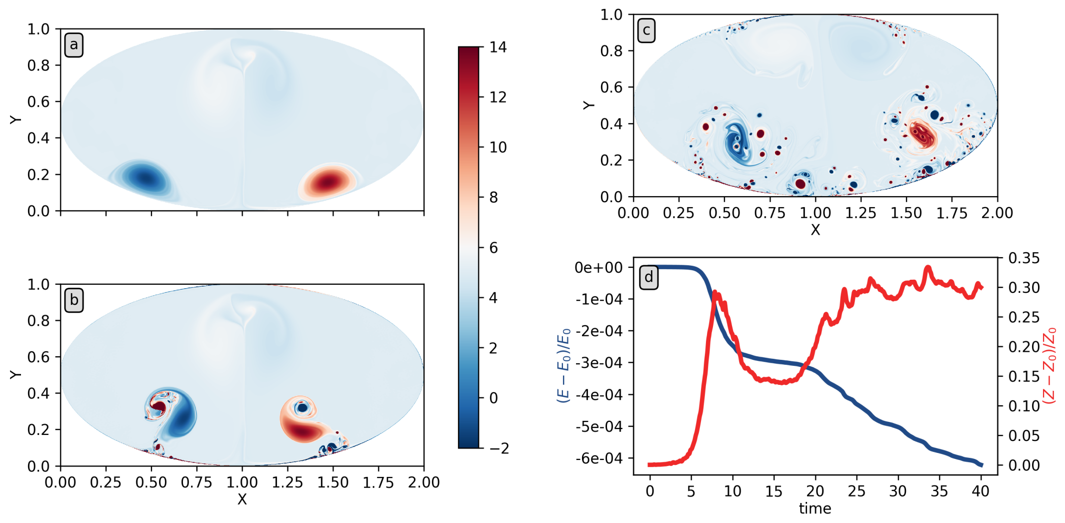

Figure 5. Potential vorticity snapshots of the dipole-wall interaction in the free-slip case at $t = 12$ (a) and no-slip case at $t = 12$ (b) and $t = 40$ (c). (d) Evolution of the energy and the enstrophy in the no-slip case. The resolution is 1,600×800. The initial position of the dipole center is at $(1, 0.5)$.

In either cases, the dipole starts to move along the minor axis Southward, while a weak trail of opposite PV, due to the vortex shield, moves Northward. As the dipole approaches the wall, the dynamics starts to differ between the free-slip and the no-slip boundary conditions. In the free-slip case, the dipole splits and each vortex continues its journey, following the wall, in an inviscid manner, according to the mirror rule (Fig. 5a). The PV remains materially well conserved, even close to the boundary. In particular there is no spurious source or sink of PV near the wall. The no-slip case differs dramatically (Fig. 5b). The phenomenology is well documented even though it is usually studied in the context of the two dimensional Euler equations (Keetels et al., 2007; Farge et al., 2011). The dipole generates a thin ribbon of opposite PV along the wall. As the dipole splits, this ribbon detaches from the wall and gets entrained in the domain where it wraps around each vortex. This halts the vortex drift along the wall. Instead, the vortices describe a loop and hit the wall again, generating another ribbon of PV that is later detached. The rebonds continue causing the initial vortices to remain trapped near the collision point. The series of layer detachments seed the flow with PV ribbons. The ribbons width and the magnitude of their vorticity depend on the numerical resolution. For this experiment, the ribbons are strong enough to experience shear instability causing them to roll up into small vortices. The domain is thus progressively filled with a swarm of small scales vortices (Fig. 5c). The dipole-wall interaction is fundamentally dissipative. It dissipates energy but it creates enstrophy (Fig. 5d). After the first collision ($t = 8$), the dissipated energy $(E_0 - E)/E_0 \approx 3\,10^{-4}$, whereas the created enstrophy $(Z - Z_0)/Z_0 \approx 30\%$. During the following collisions, the dissipated energy increases steadily up to $6\,10^{-4}$ at $t = 40$. The enstrophy behaves differently: it globally increases with time but with oscillations. As the PV distribution becomes more and more random, the amount of created enstrophy plateaus at roughly 30%. In comparison, in the free slip case and at $t = 40$, $(E_0 - E)/E_0 \approx 3\,10^{-6}$, and $(Z - Z_0)/Z_0 \approx -2\,10^{-3}$, which again shows the code ability to preserve global invariants, even though the numerics has a build-in mechanism for dissipation.

The solution at $t = 40$ has become quite turbulent (Fig. 5c), suggesting a fairly large Reynolds number. Determining the Reynolds number is a challenging task because there is no explicit viscosity in the model. The dissipation is solely handled by the WENO reconstructions, in a highly implicit manner. This is a classical issue with the implicit approach (Zhou et al., 2014). A possibility is to diagnose an effective numerical viscosity $\nu = Z^{-1}\,dE/dt$, based on the fact that for a true viscous operator the energy dissipation rate is related to Z by $dE/dt = -\nu\,Z$. From this numerical viscosity we can form an equivalent Reynolds number $Re = E^{1/2}/(H\nu)$. With this metric, the Reynolds number at $t = 40$ is $Re \sim 3.10^9$.

5.3 Dam-break problem

In this last experiment we focus on the gravity waves dynamics, and its relation with the PV, on a dam-break experiment. To illustrate the code ability to handle curved coordinates we use an annulus domain with inner radius $r_0 = 1$ and outer radius $r_1 = 2$. The coordinates (i, j) represent respectively the radial and the orthoradial directions. The metric tensor reads $\mathbf{g} = \mathrm{diag}(dr^2\, r^2\, d\theta^2)$, where dr and $r\,d\theta$ are the grid lengths in the i and j direction. The discretization is uniform in dr and $d\theta$, with respectively and 200 grid points in i and 1,600 in j. The initial state is $h = H + h_0 \tanh(y/\sigma)$ and $\mathbf{u} = 0$, with $y = r\sin\theta$, $h_0 = 0.15$, $H = 1$ and $\sigma = 0.05$. The two physical parameters are $g = 1$ and $f = 5$. The imbalance at $t = 0$ generates inertia-gravity waves and

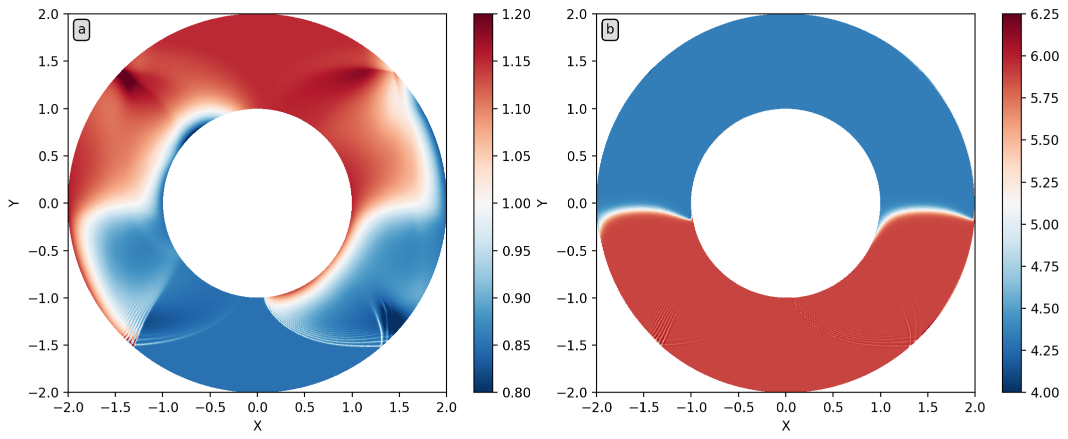

Figure 6. Snapshot of layer depth (a) and potential vorticity (b) at t=1.5 resulting from a dam-break located along y=0, with amplitude Δh=0.3. The other parameters are g=H=1 and f=5. Cylindrical coordinates are used to define the model metric. The resolution is 200×1,600.

four Kelvin waves, two along each boundary. The Kelvin waves have a clear signature on h (Fig. 6a), propagating along the boundaries with the boundary on their right (because $f > 0$), with a trapping width consistent with $R_d = 0.1$. Their propagation speed is close to $c = \sqrt{gH} = 1$ as a visual estimate tells: at $t = 1.5$ the Kelvin waves propagating along the inner boundary have moved of roughly a quarter turn. The agreement is not perfect because the regime is nonlinear enough, introducing nonlinear corrections in the wave speed. The structure of the inertia-gravity waves is more complicated. There is a net asymmetry between the waves propagating on the shallower part $H - h_0$ and the deeper part $H+h_0$. On the shallower part, the waves have clear nonlinear effects, as revealed by the series of small scales ripples and suggestive of shock wave dynamics. As there is no particular numerical treatment to handle the correct dissipation at shocks, there is no warranty that these ripples should be there, although they might be solitons. Having such small scales patterns on h is really due to the 5th order WENO reconstruction on the mass flux. Switching to a first order interpolation removes all these signals and makes h very smooth.

In contrast, the PV field has a very simple structure (Fig. 6b). At $t = 1.5$, the geostrophic currents, resulting from the geostrophic adjustment and localized along the initial discontinuity, have started to transport PV. This is the reason for the PV jump to be deformed near the boundaries. The PV field is remarkably free of any wave signal, except at the shock waves places where the PV exhibits the same ripples structure than the wave. These ripples are indicative of dissipation in action, breaking the inviscid assumption and the material conservation. Interestingly these PV ripples propagate with the waves so that their rectifying effect on the PV is much smaller. With this colorscale the net effect is invisible but a magnified colorscale reveals thin striations at few places. These small amplitude striations are the clear evidence that dissipation occurred which yielded local creation and destruction of PV. We will not go into more details as the study of wave-PV coupling is far beyond the scope of this paper. However we believe the numeric we propose is very promising to study these questions.

6 Conclusions

In this paper we have presented a fast and accurate discretization for the RSW equations. Accuracy, measured in terms of potential vorticity dynamics and conservation laws, is achieved by adapting the MILES approach (Boris et al., 1992) to the vector-invariant form of the RSW equations. The decisive step is to use a 5^{th} order WENO reconstruction on both the mass flux and the nonlinear Coriolis term. Currently the method requires a logically rectangular C-grid. The generalization to the cubed sphere is possible, the difficulty lays in handling the vorticity interpolation at the grid cells next the cube edges. The generalization to hexagonal grids is more challenging because the vorticity points are not immediately aligned with $\mathbf{U}^{\perp}$, but the recent developments on WENO reconstructions for unstructured grids (Tsoutsanis et al., 2011) pave the way to a clean solution. Speed is achieved with a series of choices rather than a single recipe, yet with a pure Python code. Though not the main point of this paper, we clearly proved that Python has become a serious option for HPC, rivaling with Fortran. In the perspective of using trained neural networks as parameterization for models, having a kernel in Python is an advantage. The code reaches typically 2 GFlop per second per core on a classical CPU architecture, which is above half the theoretical peak performance. The choices are: a reformulation of the continuous equations, the use of the Numba module to compile the most demanding functions, and the duplication of all arrays in two memory layouts to increase the arithmetic intensity by ensuring data contiguity in all functions. The reformulation is based on the introduction of $h^\star$ and $\omega^\star$, the finite volume version of h and the vorticity ω ; the use of index coordinates (i, j) ; and the introduction of $\mathbf{u}$ and $\mathbf{U}$, respectively, the covariant and the contravariant velocity. The grid lengths are used at only two places, to compute $\mathbf{U}$ from $\mathbf{u}$ with the metric tensor $\mathbf{g}$, and to relate h to $h^\star$. Everywhere else grid lengths are gone. Finite differences boil down to subtractions with no multiplication or division, which reduces the number of Flop and the amount of data transfered between the CPU and the memory.

With these choices, the floating points operations associated with the WENO reconstructions represent 85% of the total number of operations and, thanks to data contiguity, these operations are done at the CPU clock frequency, without being penalized by memory access. This particular combination of a large fraction of the total Flop with the data available in the fastest L1 cache is responsible for the overall code speed.

From the physical point of view, the numerical solutions show remarkable properties: the PV field does not exhibit any noise at the grid scale, the material conservation is excellent as far as the flow does not require enstrophy dissipation. The energy dissipation is vanishingly small with increasing resolution, even in the case where a finite amount of enstrophy is dissipated. The code handles arbitrary shaped domains with both free-slip and no-slip condition. The boundary condition on momentum is done quite naturally through the definition of the vorticity along the boundary, which is used to estimate the nonlinear Coriolis term. The no-slip boundary condition generates enstrophy, as expected, whereas it dissipates energy. In that case, by interacting with the boundary, an initially smooth PV field continuously develops fine scale structures, causing the flow to become turbulent. Finally we have shown on a dam-break experiment that the PV field remains very smooth even when small scale waves propagate. The build-in numerical dissipation allows the code to handle shock waves without blow-up even though it remains to be proven that this implicit dissipation satisfies the proper entropy condition on shock waves.

This paper has shown a new way of implementing the MILES approach in a RSW model. Several generalizations can be contemplated, some of them already mentioned earlier, but the real generalization is to adapt this idea to the full three dimensional equations, in the non-hydrostatic regime. The extension is simple: use the WENO reconstruction to each sub-term of the vortex-force term. The hope is that it provides enough build-in dissipation to handle the direct cascades of both enstrophy and energy, and it acts as a substitute for an explicit subgrid-scale closure. This idea has already been turned in a real LES code, that shows comparable performances to the code presented in this paper.

Appendix A WENO reconstruction

We now specify the $I_{i+1/2}[\phi, U]$ operator, that computes the flux $U\phi$ at location $i+1/2$. Because the operator is applied to finite volume quantities exclusively, it is strictly speaking a *reconstruction*, rather than an *interpolation*. We use the original WENO reconstruction (Jiang & Shu, 1996; Shu, 1999), also denoted WENO-JS. We express it in terms of Legendre polynomial (Balsara et al., 2016). We assume without loss of generality $U > 0$ and we start with the fifth order case, which is the general case.

A1 5th order case

The fifth order reconstruction is based on the three stencils $S_1 = \{i-2, i-1, i\}$, $S_2 = \{i-1, i, i+1\}$ and $S_3 = \{i, i+1, i+2\}$ relative to cell index i. The reconstruction reads

$$I_{i+1/2}[\phi, U] = U\left(w_1\tilde{\phi}_1 + w_2\tilde{\phi}_2 + w_3\tilde{\phi}_3\right) \tag{A1}$$

with

$$\tilde{\phi}_k = \phi_i + \phi_k^{(1)} P_1(1/2) + \phi_k^{(2)} P_2(1/2)\,, \tag{A2}$$

$$w_k = \frac{\alpha_k}{\alpha_1 + \alpha_2 + \alpha_3} \qquad \text{and} \qquad \alpha_k = \frac{\gamma_k}{(\beta_k + \epsilon)^2} \tag{A3}$$

where $P_1(x) = x$, $P_2(x) = x^2/2 - 1/24$ are the Legendre polynomials on the $[-1/2, 1/2]$ interval, and w_k are the nonlinear weights associated with the stencils S_k. The discretization is completed with the definitions of the smoothness indicator

$$\beta_k = \left(\phi_k^{(1)}\right)^2 + \frac{13}{12}\left(\phi_k^{(2)}\right)^2\,, \tag{A4}$$

the value of first ($\phi_k^{(1)}$) and second ($\phi_k^{(2)}$) moments associated with the stencil S_k

$$\phi_1^{(1)} = (\phi_{i-2} - 4\phi_{i-1} + 3\phi_i)/2 \qquad \text{and} \qquad \phi_1^{(2)} = (\phi_{i-2} - 2\phi_{i-1} + \phi_i)\,, \tag{A5}$$

$$\phi_2^{(1)} = (-\phi_{i-1} + \phi_{i+1})/2 \qquad \text{and} \qquad \phi_2^{(2)} = (\phi_{i-1} - 2\phi_i + \phi_{i+1})\,, \tag{A6}$$

$$\phi_3^{(1)} = (-3\phi_i + 4\phi_{i+1} - \phi_{i+2})/2 \qquad \text{and} \qquad \phi_3^{(2)} = (\phi_i - 2\phi_{i+1} + \phi_{i+2})\,, \tag{A7}$$

and the linear weights

$$\gamma_1 = 1/10\,, \quad \gamma_2 = 3/5\,, \quad \gamma_2 = 3/10\,. \tag{A8}$$

The regularization factor is set to $\epsilon = 10^{-8}$. These linear weights are the original ones proposed by Shu. They are the ones that makes the whole reconstruction fifth order at locations where ϕ is smooth.

The adaptation of this reconstruction to the case of the vorticity, namely $I_i[\omega, V]$, is straightforward. Because of the vorticity being discretized at half integers indices, the only change is to replace the ϕ_i terms with $\omega_{i-1/2}$ in the above formulas.

Close to boundary we use a 3rd order WENO reconstruction (Shu, 1999), if either $\{i-2\}$ or $\{i+2\}$ is outside of the domain but the $\{i-1, i, i+1\}$ cells are inside the domain. We downgrade to the 1st order reconstruction if either $\{i-1\}$ or $\{i+1\}$ is outside the domain. For sake of completeness we explicit the formula in these two cases.

A2 3rd and 1st order cases

The third order case (Shu, 1999) reads

$$I_{i+1/2}[\phi, U] = U\,(w_1\tilde{\phi}_1 + w_2\tilde{\phi}_2) \tag{A9}$$

with $\tilde{\phi}_k = \phi_i + \phi_k^{(1)} P_1(1/2)$,

$$w_k = \frac{\alpha_k}{\alpha_1 + \alpha_2}, \qquad \alpha_k = \frac{\gamma_k}{(\beta_k + \epsilon)^2}, \qquad \beta_k = \left(\phi_k^{(1)}\right)^2, \tag{A10}$$

$\gamma_1 = 1/3$, $\gamma_2 = 2/3$, $\phi_1^{(1)} = \phi_i - \phi_{i-1}$ and $\phi_2^{(1)} = \phi_{i+1} - \phi_i$.

The first order case is simply

$$I_{i+1/2}[\phi, U] = U\phi_i\,. \tag{A11}$$

Acknowledgments

This work was supported by the French National program LEFE (Les Enveloppes Fluides et l'Environnement) and also by the recurrent annual funding provided by the LOPS parent agencies UBO, CNRS, IRD and Ifremer. This recurrent funding is an important tool of the French research system that helps fundamental research. The work used HPC resources from GENCI-TGCC (Grant 2020-A0090112051). The code is available from `https://doi.org/10.5281/zenodo.4968737`.

References

Balsara, D. S., Garain, S., & Shu, C.-W. (2016). An efficient class of weno schemes with adaptive order. *Journal of Computational Physics*, *326*, 780–804.

Boris, J., Grinstein, F., Oran, E., & Kolbe, R. (1992). New insights into large eddy simulation. *Fluid dynamics research*, *10*(4-6), 199.

Brecht, R., Bauer, W., Bihlo, A., Gay-Balmaz, F., & MacLachlan, S. (2019). Variational integrator for the rotating shallow-water equations on the sphere. *Quarterly Journal of the Royal Meteorological Society*, *145*(720), 1070–1088.

Chen, Q., Gunzburger, M., & Ringler, T. (2011). A scale-invariant formulation of the anticipated potential vorticity method. *Monthly Weather Review*, *139*(8), 2614–2629.

Cotter, C. J., & Thuburn, J. (2014). A finite element exterior calculus framework for the rotating shallow-water equations. *Journal of Computational Physics*, *257*, 1506–1526.

Desbrun, M., Kanso, E., & Tong, Y. (2006). Discrete differential forms for computational modeling. *ACM SIGGRAPH 2006 Courses on - SIGGRAPH '06*. doi: 10.1145/1185657.1185665

Farge, M., Schneider, K., et al. (2011). Energy dissipating structures produced by walls in two-dimensional flows at vanishing viscosity. *Physical review letters*, *106*(18), 184502.

Frankel, T. (2011). *The geometry of physics: an introduction.* Cambridge university press.

Gallerano, F., & Cannata, G. (2011). Central weno scheme for the integral form of contravariant shallow-water equations. *International Journal for Numerical Methods in Fluids*, *67*(8), 939–959.

Gottlieb, S., Shu, C.-W., & Tadmor, E. (2001). Strong stability-preserving high-order time discretization methods. *SIAM review*, *43*(1), 89–112.

Graham, J. P., & Ringler, T. (2013). A framework for the evaluation of turbulence closures used in mesoscale ocean large-eddy simulations. *Ocean Modelling*, *65*, 25–39.

Jiang, G.-S., & Shu, C.-W. (1996). Efficient implementation of weighted eno schemes. *Journal of computational physics*, *126*(1), 202–228.

Keetels, G. H., D'Ortona, U., Kramer, W., Clercx, H. J., Schneider, K., & van Heijst, G. J. (2007). Fourier spectral and wavelet solvers for the incompressible navier–stokes equations with volume-penalization: Convergence of a dipole–wall collision. *Journal of Computational Physics*, *227*(2), 919–945.

Klöwer, M., Düben, P., & Palmer, T. (2020). Number formats, error mitigation, and scope for 16-bit arithmetics in weather and climate modeling analyzed with a shallow water model. *Journal of Advances in Modeling Earth Systems*, *12*(10), e2020MS002246.

Lam, S. K., Pitrou, A., & Seibert, S. (2015). Numba: A llvm-based python jit compiler. In *Proceedings of the second workshop on the llvm compiler infrastructure in hpc* (pp. 1–6).

Lattner, C., & Adve, V. (2004). Llvm: A compilation framework for lifelong program analysis & transformation. In *International symposium on code generation and optimization, 2004. cgo 2004.* (pp. 75–86).

Margolin, L. G., Rider, W. J., & Grinstein, F. F. (2006). Modeling turbulent flow with implicit les. *Journal of Turbulence*(7), N15.

Mullen, P., McKenzie, A., Pavlov, D., Durant, L., Tong, Y., Kanso, E., ... Desbrun, M. (2011). Discrete lie advection of differential forms. *Foundations of Computational Mathematics*, *11*(2), 131–149.

Noelle, S., Xing, Y., & Shu, C.-W. (2007). High-order well-balanced finite volume weno schemes for shallow water equation with moving water. *Journal of Computational Physics*, *226*(1), 29–58.

Perot, J. B., & Zusi, C. J. (2014). Differential forms for scientists and engineers. *Journal of Computational Physics*, *257*, 1373–1393.

Pope, S. B. (2004). Ten questions concerning the large-eddy simulation of turbulent flows. *New journal of Physics*, *6*(1), 35.

Pressel, K. G., Kaul, C. M., Schneider, T., Tan, Z., & Mishra, S. (2015). Large-eddy simulation in an anelastic framework with closed water and entropy balances. *Journal of Advances in Modeling Earth Systems*, *7*(3), 1425–1456.

Ramadhan, A., Wagner, G. L., Hill, C., Campin, J.-M., Churavy, V., Besard, T., ... Marshall, J. (2020). Oceananigans.jl: Fast and friendly geophysical fluid dynamics on gpus. *Journal of Open Source Software*, *5*(53), 2018. doi: 10.21105/joss.02018

Ringler, T. D., Thuburn, J., Klemp, J. B., & Skamarock, W. C. (2010). A unified approach to energy conservation and potential vorticity dynamics for arbitrarily-structured c-grids. *Journal of Computational Physics*, *229*(9), 3065–3090.

Roullet, G. (2021, June). *pyRSW: A Fast Monotone Rotating Shallow Water model.* Zenodo. Retrieved from `https://doi.org/10.5281/zenodo.4968737` (The code is maintained on https://github.com/pvthinker/pyRSW) doi: 10.5281/zenodo.4968737

Sadourny, R., & Basdevant, C. (1985). Parameterization of subgrid scale barotropic and baroclinic eddies in quasi-geostrophic models: Anticipated potential vorticity method. *Journal of Atmospheric Sciences*, *42*(13), 1353–1363.

Sagaut, P. (2006). *Large eddy simulation for incompressible flows: an introduction.* Springer.

Shchepetkin, A. F., & McWilliams, J. C. (2005). The regional oceanic modeling system (roms): a split-explicit, free-surface, topography-following-coordinate oceanic model. *Ocean modelling*, *9*(4), 347–404.

Shu, C.-W. (1999). High order eno and weno schemes for computational fluid dynamics. In *High-order methods for computational physics* (pp. 439–582). Springer.

Thuburn, J., Ringler, T. D., Skamarock, W. C., & Klemp, J. B. (2009). Numerical representation of geostrophic modes on arbitrarily structured c-grids. *Journal of Computational Physics*, *228*(22), 8321–8335.

Tsoutsanis, P., Titarev, V. A., & Drikakis, D. (2011). Weno schemes on arbitrary mixed-element unstructured meshes in three space dimensions. *Journal of Computational Physics*, *230*(4), 1585–1601.

Williams, S., Waterman, A., & Patterson, D. (2009). Roofline: an insightful visual performance model for multicore architectures. *Communications of the ACM*, *52*(4), 65–76.

Williamson, D. L., Drake, J. B., Hack, J. J., Jakob, R., & Swarztrauber, P. N. (1992). A standard test set for numerical approximations to the shallow water equations in spherical geometry. *Journal of Computational Physics*, *102*(1), 211–224.

Xing, Y., & Shu, C.-W. (2005). High order finite difference weno schemes with the exact conservation property for the shallow water equations. *Journal of Com-*

putational Physics, *208*(1), 206–227.

Zhou, Y., Grinstein, F. F., Wachtor, A. J., & Haines, B. M. (2014). Estimating the effective reynolds number in implicit large-eddy simulation. *Physical Review E*, *89*(1), 013303.