UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Escola Superior d'Enginyeries Industrial,
Aeroespacial i Audiovisual de Terrassa

STUDY: SIMULATION OF PLANETARY ATMOSPHERES

Marc Andrés Carcasona

A thesis submitted in partial fulfillment of the requirements for the degree of
BSc. in Aerospace Engineering Technologies

Director Enrique García Melendo
Co-director Manel Soria Guerrero

June 30, 2020

Study: Simulation of Planetary Atmospheres

Abstract

by Marc Andrés Carcasona
Universitat Politècnica de Catalunya, BarcelonaTech (UPC)
June 30, 2020

English

This thesis analyzes different ways of simulating planetary atmospheres to reproduce the observed phenomena and further understand its dynamics. To do so, a two dimensional Shallow Water model in ellipsoidal coordinates will be used in order to start capturing part of the dynamics that occur at large scales in the atmospheres of gas giant planets. Then, some additions will be done in order to control the instabilities that appear in the model, the possibility of simulating the poles will be presented and a new way of perturbing the system to achieve accurate steady-state vortices will be assessed. Then, a three dimensional Boussinesq model will be implemented to capture the convective movement of atmospheres in the radial direction. The results prove that the Shallow Water with the additions here presented is a very good option to reproduce with good accuracy the horizontal dynamics of planetary atmospheres, while the three dimensional model, at a smaller scale, is capable of reproducing the upwards movement of fluid that the Shallow Water lacks. Both codes are capable of running in parallel computers improving the resolutions of the simulations carried out. Then, these codes might be used to further understand the dynamics of planetary atmospheres and complement the observational data available.

Català

En aquest treball s'analitzaran diferents maneres de simular atmosferes planetàries per poder reproduir la fenomenologia observada i poder així entendre millor les seves dinàmiques. Per fer això, un model bidimensional anomenat Shallow Water, el qual treballa en coordenades el·lipsoidals, serà emprat i permetrà reproduir la dinàmica a gran escala de les atmosferes dels planetes gasosos gegants. A aquest codi se li han afegit algunes millores com són el control d'una inestabilitat qua apareix al model, la possibilitat de simular a les regions polars i una nova manera d'introduir pertorbacions per tal de poder obtenir un vòrtex que en equilibri tingui els paràmetres desitjats. Llavors, es programarà també un model tridimensional basat en l'aproximació de Boussinesq per tal de poder captar el moviment convectiu en la direcció radial característic de les atmosferes. Els resultats indiquen que el model Shallow Water amb les millores que es descriuen en aquest treball reprodueixen amb bona exactitud els grans moviments horitzontals de les atmosferes planetàries, mentre que el model tridimensional és capaç de reproduir els moviments verticals a petita escala que no té en compte el Shallow Water. Ambdós codis poden funcionar en ordinadors paral·lels, de manera que es poden fer simulacions a alta resolució. Per tant, aquests codis poden ser utilitzats en un futur per tal d'ajudar a entendre la dinàmica de les atmosferes planetàries i complementar les dades experimentals existents.

Castellano

En este trabajo se analizan distintas formas de simular atmósferas planetarias para poder reproducir la fenomenología observada y poder así entender mejor sus dinámicas. Para poder conseguir este objetivo, un modelo bidimensional llamado Shallow Water, el cuál trabaja en coordenadas elipsoidales, será usado y permitirá reproducir la dinámica a gran escala de las atmósferas de los planetas gaseosos gigantes. A este código se le han añadido algunas mejoras como son el control de una inestabilidad que aparece en el modelo, la posibilidad de simular las regiones polares y una nueva manera de introducir perturbaciones para poder obtener un vórtice que en equilibrio tenga unos parámetros deseados. Entonces, se programará también un modelo tridimensional basado en la aproximación de Boussinesq para poder captar el movimiento convectivo en la dirección radial característico de las atmósferas. Los resultados indican que el modelo Shallow Water con las mejoras aquí propuestas reproducen con buena exactitud los grandes movimientos horizontales de las atmósferas planetarias, mientras que el modelo tridimensional es capaz de reproducir los movimientos verticales a pequeña escala que no tiene en cuenta el Shallow Water. Ambos códigos están paralelizados, lo que permite que se puedan realizar simulaciones a alta resolución. Por lo tanto, dichos códigos pueden ser usados en un futuro para ayudar a entender la dinámica de las atmósferas planetarias y complementar así los datos experimentales disponibles.

I declare that,

- the work in this Degree Thesis is completely my own work,
- no part of this Degree Thesis is taken from other people's work without giving them credit,
- all references have been clearly cited,
- I'm authorised to make use of the research group related information I'm providing in this document.

I understand that an infringement of this declaration leaves me subject to the foreseen disciplinary action by *The Universitat Politècnica de Catalunya - BarcelonaTECH*

Marc Andrés Carcasona June 30, 2020

Study: Simulation of planetary atmospheres

Contents

List of Figures

List of Algorithms

Perhaps some day in the dim future
it will be possible to advance the computations
faster than the weather advances and at a cost less
than the saving to mankind due to the information gained.
But that is a dream.

LEWIS FRY RICHARDSON ([1])

Introduction

The atmospheres of gas giant planets are still a great source of scientific knowledge, yet not many information can be gathered from observational data. This is due to the lack of space missions dedicated to track the phenomena that occurs in the outer layers of their atmospheres. Therefore, many computational techniques have to be applied in order to unvail its secrets and to deeply understand the large scale dynamics and that complement the available experimental data. In this thesis two different ways will be presented, analyzed and even improved.

The first of those is the so called Shallow Water model. This model assumes very thin layer of fluid and some simplifications to the original Navier-Stokes equations can be applied in order to reduce it to a 2D problem. This method has been applied with great results before [2] and in this work it will be implemented in *MATLAB*® and then some additions will be added in order to increase the capabilities of an already programmed parallel code called Shallow Worlds [3].

The first of those additions should be the control of the Hollingsworth instability that has appeared in some high resolution simulations. The two ways in which this will be done is through the addition of a hyperviscosity and changing the flux limiter of the numerical integration of the advection. Secondly, a new way of introducing perturbations into the system will be assessed. This will be based on the geostrophic equilibrium, the mechanism responsible of the appearance of large scale vortices. Finally, the capability of simulating polar regions will be added. This problem arises from the discretization of the ellipsoid that is singular at the poles and a solution to overcome this problem is proposed in order to reuse the core of the code.

The second will be a 3D model, that will give the opportunity to simulate the vertical movement of fluid that is of great interest for the case of the study of storms in gas giant planets. Also, this non-hydrostatic model should include the thermodynamic effects that induce this type of movements. Note, though, that for this case the code will have to be written from scratch, using the parallel library developed by the co-director of this thesis, Prof. Dr. Manel Soria as described in [3].

The main reason to do so is that this first version of the code will serve as a milestone for the atmospheric model as well as for other future applications. Since the development of such code can be really difficult (attending also the problems that can arise from parallelization) at this stage some benchmark cases can be carried out in order to ensure the correct good functioning of it. The first of those is the so-called Taylor Green Vortices (TGV). The domain is full-periodic and thus, any boundary condition problems will be avoided and its solution has been thoroughly studied. Once this benchmark is completed, the next milestone will be to reproduce the Rayleigh-Bénard case. At this point an extra transport equation will be solved and there will be top and bottom boundary conditions. Once this test is passed, the following point will be to implement the Boussinesq approximation equations and perform a benchmark in an atmospheric level. More precisely, the case that will be solved is the so-called Robert Rising Bubble, which is a case used as a benchmark for atmospheric models with vertical convection.

As a requirements for the project we can state that both codes shall be scalable to at least 10^3

processors to ensure that can run in supercomputers. The Shallow Worlds code already has this capability but it is important that any addition performed to it does not affect it. The programming language for the parallel codes must be C + MPI. The final requirement is that the simulations should try to reproduce as accurately as possible the experimental observations and hence, when needed, changes will have to be introduced (e.g. the numerical scheme, boundary conditions, ...).

Part I

2D Shallow Water Model

1 — Shallow Water model

In a planet, the surface that an atmosphere occupies is normally significantly bigger than its thickness. Due to this fact, it is possible to perform an order of magnitude analysis so as to simplify the Navier-Stokes equations and obtain a more simplified ones, that still capture the dynamics of the atmosphere. The resulting set of equations are the so-called Shallow Water (SW) equations. In this section a proper study of them will be presented.

1.1 Derivation of the SW equations in Cartesian coordinates

We can start with the Navier-Stokes equations for a non-viscous fluid in a rotating frame of reference with gravity as an external force and the continuity equation, since all thermodynamic effects will be neglected. This set of equations are the following

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{u}) = 0 \tag{1.1a}$$

$$\frac{\partial \vec{u}}{\partial t} + (\vec{u} \cdot \nabla)\, \vec{u} - f\left(\vec{u} \times \vec{k}\right) = -\frac{1}{\rho}\nabla p - \vec{g} \tag{1.1b}$$

where f is the Coriolis parameter and takes the form of

$$f = 2\Omega \sin(\varphi) \tag{1.2}$$

being Ω the angular speed of the planet and φ the planetographic latitude. For more details of ellipsoidal coordinates, see Section 1.2. The inclusion of the rotation effects has been done attending to [4].

The derivation of these equations will be the one presented in [5]. The first assumption will be that of neglecting the variations in the z axis. This can be done because the fluid layer is very thin, and only the horizontal components are taken into account. By the same reason it can be assumed that $\vec{u} = (u, v, w) \approx (u, v, 0)$. The next thing to consider is the geometry of a generic problem. The diagram found in Figure 1.1 illustrates it.

From the scheme it can be deduced that $h = D - h_B + \eta$, where h_B would denote the topography of the bottom layer, D the mean fluid depth, η the surface elevation with respect to D and h the fluid height. It is necessary to introduce these new variables into the equations and one way to do so is by integrating the continuity equation for a fluid column.

$$\int_{h_B}^{h_B+h} \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{u}) dz = 0$$

$$\int_{h_B}^{h_B+h} \frac{\partial \rho}{\partial t} dz + \int_{h_B}^{h_B+h} \nabla \cdot (\rho \vec{u}) dz = 0$$

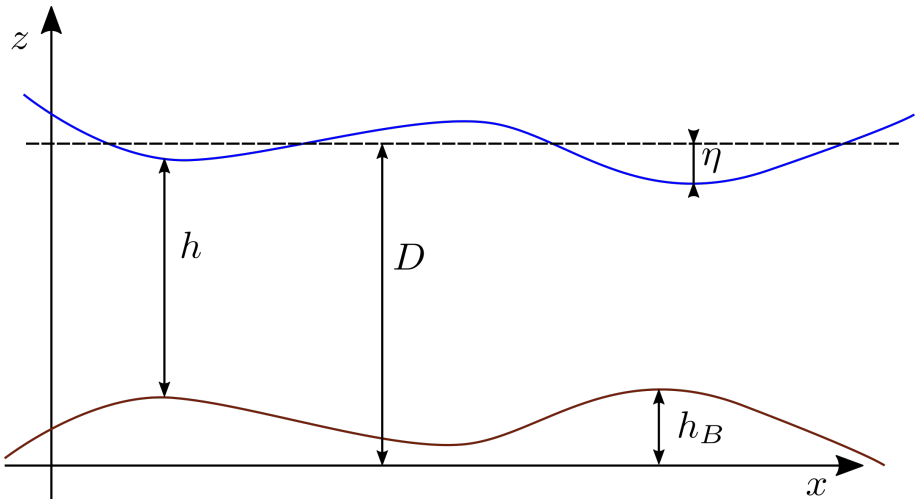


Figure 1.1: Diagram of the shallow water model.

$$\frac{\partial}{\partial t}\left(\rho\int_{h_B}^{h_B+h} dz\right) + \nabla\cdot\left(\rho\vec{u}\int_{h_B}^{h_B+h} dz\right) = 0$$

$$\frac{\partial \rho h}{\partial t} + \nabla\cdot(\rho\vec{u}h) = 0$$

Assuming incompressible flow and taking into account that $\frac{\partial h}{\partial t} = \frac{\partial \eta}{\partial t}$ the new continuity equation is found and equals

$$\frac{\partial \eta}{\partial t} + \nabla\cdot(h\vec{u}) = 0 \tag{1.3}$$

On the other hand, the three momentum equations are reduced to

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} - fv = -\frac{1}{\rho}\frac{\partial p}{\partial x}$$
$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + fu = -\frac{1}{\rho}\frac{\partial p}{\partial y}$$
$$0 = -\frac{1}{\rho}\frac{\partial p}{\partial z} - g$$

The third of these equations is only telling that an hydrostatic assumption has been done and it can be integrated to find the function of the pressure with the height knowing that at the free surface the pressure is zero. Hence,

$$\frac{\partial p}{\partial z} = -\rho g$$

$$\int_p^0 dp = \int_0^\eta -\rho g dz$$

$$p = \rho g \eta \tag{1.4}$$

Replacing the result of Eq. (1.4) into the other momentum components, the whole set of equations for the Shallow Water model are obtained. The system of resulting PDEs is

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} - fv = -g\frac{\partial \eta}{\partial x} \tag{1.5a}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + fu = -g\frac{\partial \eta}{\partial y} \tag{1.5b}$$

$$\frac{\partial \eta}{\partial t} + \nabla \cdot (h\vec{u}) = 0 \tag{1.5c}$$

$$h - D - \eta + h_b = 0 \tag{1.5d}$$

Or, in a compact notation,

$$\frac{\partial \vec{u}}{\partial t} + (\vec{u} \cdot \nabla)\,\vec{u} - f\left(\vec{u} \times \vec{k}\right) = -g\nabla\eta \tag{1.6a}$$

$$\frac{\partial \eta}{\partial t} + \nabla \cdot (h\vec{u}) = 0 \tag{1.6b}$$

$$h - D - \eta + h_b = 0 \tag{1.6c}$$

The form presented in Eq. (1.6) show the equations in Cartesian coordinates, yet they are not suitable for atmospheric applications since the planets will be approximated with an ellipsoid. Therefore, this equations will have to be transformed into the new coordinate system.

1.2 SW equations in ellipsoidal coordinates

The main idea of this section is to derive the actual equations that will have to be solved for ellipsoidal geometries and to do so, tensor calculus will be needed. The derivation of the vector operators are presented in Appendix A. The ellipsoidal coordinate system can be described through the scheme presented in Figure 1.2, yet this general situation can be simplified by assuming an oblate spheroid as shown in Figure 1.3. This is an ellipsoid of revolution which will simplify dramatically the mathematics and will only leave two variables to describe the position: the planetographic latitude φ and the longitude θ.

Then, for the continuity equation, the divergence operator can be substituted by the one in ellipsoidal coordinates (given in Eq. (A.11)) to obtain the new form of the equation:

$$\frac{\partial \eta}{\partial t} + \frac{1}{r_Z(\varphi)}\frac{\partial (hu)}{\partial \theta} + \frac{1}{r_M(\varphi)}\frac{\partial (hv)}{\partial \varphi} - \frac{\sin(\varphi)}{r_Z(\varphi)}hv = 0 \tag{1.7}$$

The same procedure can be followed for the momentum equations but this time using the fact that the convection operator is not as straightforward as it might seem. For further details see Appendix A and its final expression can be seen in Eq. (A.17). The next term that could generate problems is the one related to the Coriolis force. Its expression is rather simple for Cartesian coordinates but, in fact, this term ($\vec{F}_c$) can be generalized to any curvilinear coordinates as stated in [6] like

$$\vec{F}_c = f\vec{e}_3 \times \vec{u} \tag{1.8}$$

where in this case

$$f = -\frac{2\Omega}{h_2}\frac{dh_1}{d\xi_2} \tag{1.9}$$

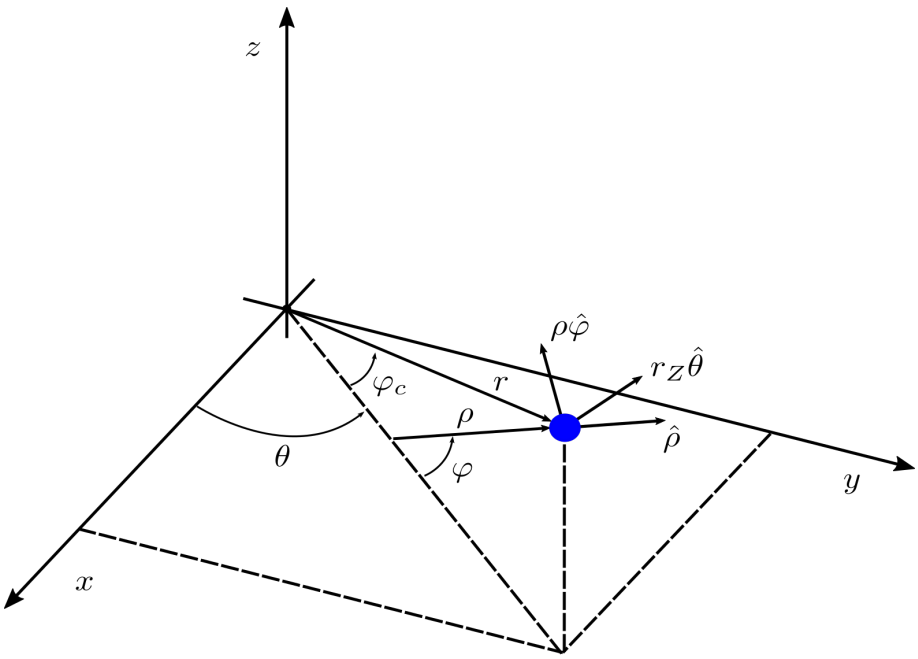


Figure 1.2: Ellipsoidal planetocentric coordinate system.

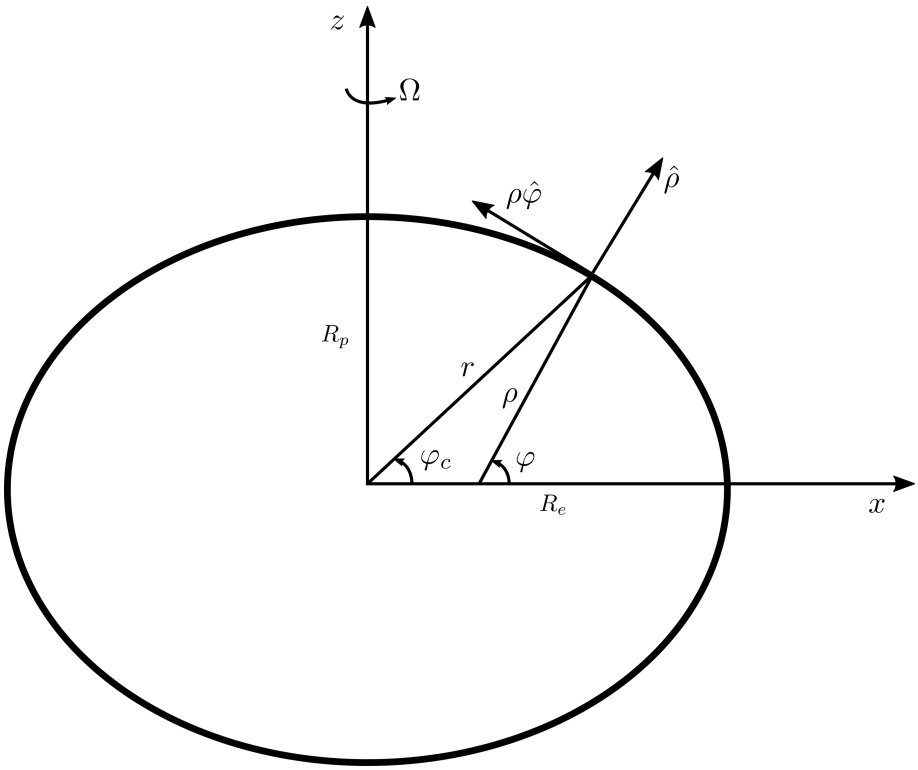


Figure 1.3: Oblate spheroid scheme.

The terms h_i and ξ_i are the scale parameters and the general curvilinear coordinates, respectively. All together they form a metric that can be described as

$$ds^2 = h_1^2 d\xi_1^2 + h_2^2 d\xi_2^2 + h_3^2 d\xi_3^2$$

Taking then into account that the metric that is being used is the oblate spheroid ones, then it is described by

$$ds^2 = r_Z^2 d\theta^2 + r_M^2 d\varphi^2 \tag{1.10}$$

A derivation of the scale parameters can be seen in Appendix A and its final expressions are shown in Eq. (A.10). Performing then the appropriate operations with those scale parameters and computing the new Coriolis parameter we get that

$$f = 2\Omega \sin(\varphi) \tag{1.11}$$

Thus, the change of coordinates has not affected the definition of the Coriolis parameter. Equation (1.8) can be evaluated as shown in [7] yielding a result, for any orthogonal curvilinear coordinates, of

$$\vec{F}_c = \begin{bmatrix} 2\Omega u_3 \cos(\phi) - 2\Omega u_2 \sin(\phi) \\ 2\Omega u_1 \sin(\phi) \\ -2\Omega u_1 \cos(\phi) \end{bmatrix} \tag{1.12}$$

And the definitions of $\cos(\phi)$ and $\sin(\phi)$ are

$$\cos(\phi) = \frac{1}{h_3}\frac{\partial h_1}{\partial \xi_3} \tag{1.13}$$

$$\sin(\phi) = -\frac{1}{h_2}\frac{\partial h_1}{\partial \xi_2} \tag{1.14}$$

Evaluating them for the ellipsoidal coordinate system, the result is

$$\cos(\phi) = 0 \tag{1.15}$$

$$\sin(\phi) = \sin(\varphi) \tag{1.16}$$

Adding also the fact that $u_3 = 0$ the expression for the Coriolis force is

$$\vec{F}_c = \begin{bmatrix} -2\Omega v \sin(\varphi) \\ 2\Omega u \sin(\varphi) \\ 0 \end{bmatrix} = \begin{bmatrix} -fv \\ fu \\ 0 \end{bmatrix} \tag{1.17}$$

which corresponds to the same that appeared on the Cartesian Shallow Water equations. With all this information one is prepared to rewrite the momentum equations in ellipsoidal coordinates. This is shown in Eq. (1.18), where the pressure gradient has been adapted using the gradient operator shown in Eq. (A.13).

$$\frac{\partial u}{\partial t} + \frac{u}{r_Z(\varphi)}\frac{\partial u}{\partial \theta} + \frac{v}{r_M(\varphi)}\frac{\partial u}{\partial \varphi} - \left(f + \frac{\sin(\varphi)}{r_Z(\varphi)}u\right)v = \frac{-g}{r_Z(\varphi)}\frac{\partial \eta}{\partial \theta} \tag{1.18a}$$

$$\frac{\partial v}{\partial t} + \frac{u}{r_Z(\varphi)}\frac{\partial v}{\partial \theta} + \frac{v}{r_M(\varphi)}\frac{\partial v}{\partial \varphi} + \left(f + \frac{\sin(\varphi)}{r_Z(\varphi)}u\right)u = \frac{-g}{r_M(\varphi)}\frac{\partial \eta}{\partial \varphi} \tag{1.18b}$$

Then, gathering all this information and rewriting it as a single system of PDEs, the Shallow Water equations in an ellipsoidal coordinate system are obtained. These are

$$\frac{\partial u}{\partial t} + \frac{u}{r_Z(\varphi)}\frac{\partial u}{\partial \theta} + \frac{v}{r_M(\varphi)}\frac{\partial u}{\partial \varphi} - \left(f + \frac{\sin(\varphi)}{r_Z(\varphi)}u\right)v = \frac{-g}{r_Z(\varphi)}\frac{\partial \eta}{\partial \theta} \tag{1.19a}$$

$$\frac{\partial v}{\partial t} + \frac{u}{r_Z(\varphi)}\frac{\partial v}{\partial \theta} + \frac{v}{r_M(\varphi)}\frac{\partial v}{\partial \varphi} + \left(f + \frac{\sin(\varphi)}{r_Z(\varphi)}u\right)u = \frac{-g}{r_M(\varphi)}\frac{\partial \eta}{\partial \varphi} \tag{1.19b}$$

$$\frac{\partial \eta}{\partial t} + \frac{1}{r_Z(\varphi)}\frac{\partial (hu)}{\partial \theta} + \frac{1}{r_M(\varphi)}\frac{\partial (hv)}{\partial \varphi} - \frac{\sin(\varphi)}{r_Z(\varphi)}hv = 0 \tag{1.19c}$$

$$h - D - \eta + h_b = 0 \tag{1.19d}$$

1.3 Potential vorticity and its conservation

For atmospheric applications is very common to use the so-called potential vorticity as a tracer to see what happens to the fluid, since it is a material invariant. To see what it represents, the derivation presented in [5] will be followed.

First, let's define the shallow water vorticity as

$$\vec{\omega}^* = \nabla \times \vec{u} = \vec{k}\zeta \tag{1.20}$$

Using also the following vector identity to split the advection

$$(\vec{u}\cdot\nabla)\vec{u} = \frac{1}{2}\nabla(\vec{u}\cdot\vec{u}) - \vec{u}\times(\nabla\times\vec{u}) \tag{1.21}$$

we obtain an equivalent formulation of the momentum equation as

$$\frac{\partial \vec{u}}{\partial t} + (\vec{\omega}^* + f\vec{k})\times\vec{u} = -\nabla\left(gh + \frac{1}{2}\vec{u}^{\,2}\right) \tag{1.22}$$

Taking the curl of this expression and having in mind that the divergence of a curl is zero and that $(\vec{\omega}^*\cdot\nabla)\vec{u} = 0$ due to the perpendicularity of the shallow water vorticity to any direction of the velocity vector, one arrives to the following expression

$$\frac{\partial \zeta}{\partial t} + (\vec{u}\cdot\nabla)(\zeta + f) = -(f+\zeta)\nabla\cdot\vec{u} \tag{1.23}$$

where $\zeta = \vec{k}\cdot\vec{\omega}^*$. On the other hand, the mass conservation equation can be rewritten as shown in Eq. (1.24) provided that $\nabla\cdot(h\vec{u}) = (\vec{u}\cdot\nabla)h + h(\nabla\cdot\vec{u})$.

$$\frac{Dh}{Dt} + h(\nabla\cdot\vec{u}) = 0 \tag{1.24}$$

Multiplying this new expression by $(\zeta + f)/h$ the result is

$$\left(\frac{\zeta + f}{h}\right)\frac{Dh}{Dt} + (\zeta + f)\nabla \cdot \vec{u} = 0 \tag{1.25}$$

Introducing this relation into Eq. (1.23) we obtain

$$\frac{D(\zeta + f)}{Dt} = \left(\frac{\zeta + f}{h}\right)\frac{Dh}{Dt} \tag{1.26}$$

Taking into account that

$$\frac{D}{Dt}\left(\frac{\zeta + f}{h}\right) = \frac{1}{h}\frac{D\left(\zeta + f\right)}{Dt} - \left(\frac{\zeta + f}{h^2}\right)\frac{Dh}{Dt} \tag{1.27}$$

the final expression can be obtained by introducing this relation into Eq. (1.26) and this is

$$\frac{D}{Dt}\left(\frac{\zeta + f}{h}\right) = 0 \tag{1.28}$$

Defining for convenience the potential vorticity as

$$q = \left(\frac{\zeta + f}{h}\right) \tag{1.29}$$

we can say that the potential vorticity is conserved provided that

$$\frac{Dq}{Dt} = 0 \tag{1.30}$$

This variable is interesting as a a tracer because it depends on many relevant variables of the atmosphere such as the velocity field (that appears in the vorticity term), the planet's rotation and position (relevant in the Coriolis parameter) and, finally, the surface elevation.

2 — Numerical resolution

To solve the Shallow Water equations, various methods are applied. The way to solve the system of partial differential equations may vary depending on the terms that are taken into account. The Shallow Water equations are Eq. (2.1). The explanation will be performed for the Cartesian coordinate system since they contain the major problems of the numerical resolution. Then, to adapt the scheme to the ellipsoidal coordinates only the addition of the geometric factors are needed and they are immediate to integrate.

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} - fv = -g\frac{\partial \eta}{\partial x} \tag{2.1a}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + fu = -g\frac{\partial \eta}{\partial y} \tag{2.1b}$$

$$\frac{\partial \eta}{\partial t} + \nabla \cdot (h\vec{u}) = 0 \tag{2.1c}$$

$$h - D - \eta + h_b = 0 \tag{2.1d}$$

Now, in an independent section for each term all the methods will be explained and discussed.

2.1 Arakawa C-grid

To solve the Shallow Water equations it is of great importance the way in which the grid is constructed and where the variables are going to be computed. Plenty of options are available but only some of them will be actually useful. This comes from the coupling of the velocity field and the free surface elevation. The main problem is that using a collocated grid (all variables calculated at the center point) lead to nonphysical result unless a special algorithm is applied. Therefore, other options have to be considered.

Arakawa and Lamb contributed a lot in this aspect and described many types of possible combinations [8]. These are the so-called Arakawa grids and are distinguished by a letter from A to E. The one that we are interested in, for its good properties, is the C-grid. This grid has the disposition shown in Figure 2.1. As it can be seen, the velocity field is evaluated at a staggered position that coincides with the faces of the control volume, while the surface elevation is evaluated at the central point.

This idea can be generalized in the following way: all vector fields will be evaluated in a staggered fashion like the velocity, while any scalar field will be centered. This type of disposition turns out to be one of the best well-posed situations since the staggering is somehow natural due to the algorithm that will be used to solve these equations. A similar staggered grid is used for other CFD applications as will be seen in Part II.

In case of an ellipsoidal grid the treatment will be exactly the same with the only difference that the

distances Δx and Δy will not be constant and instead will depend on φ and θ. For further information the reader is referred to [3].

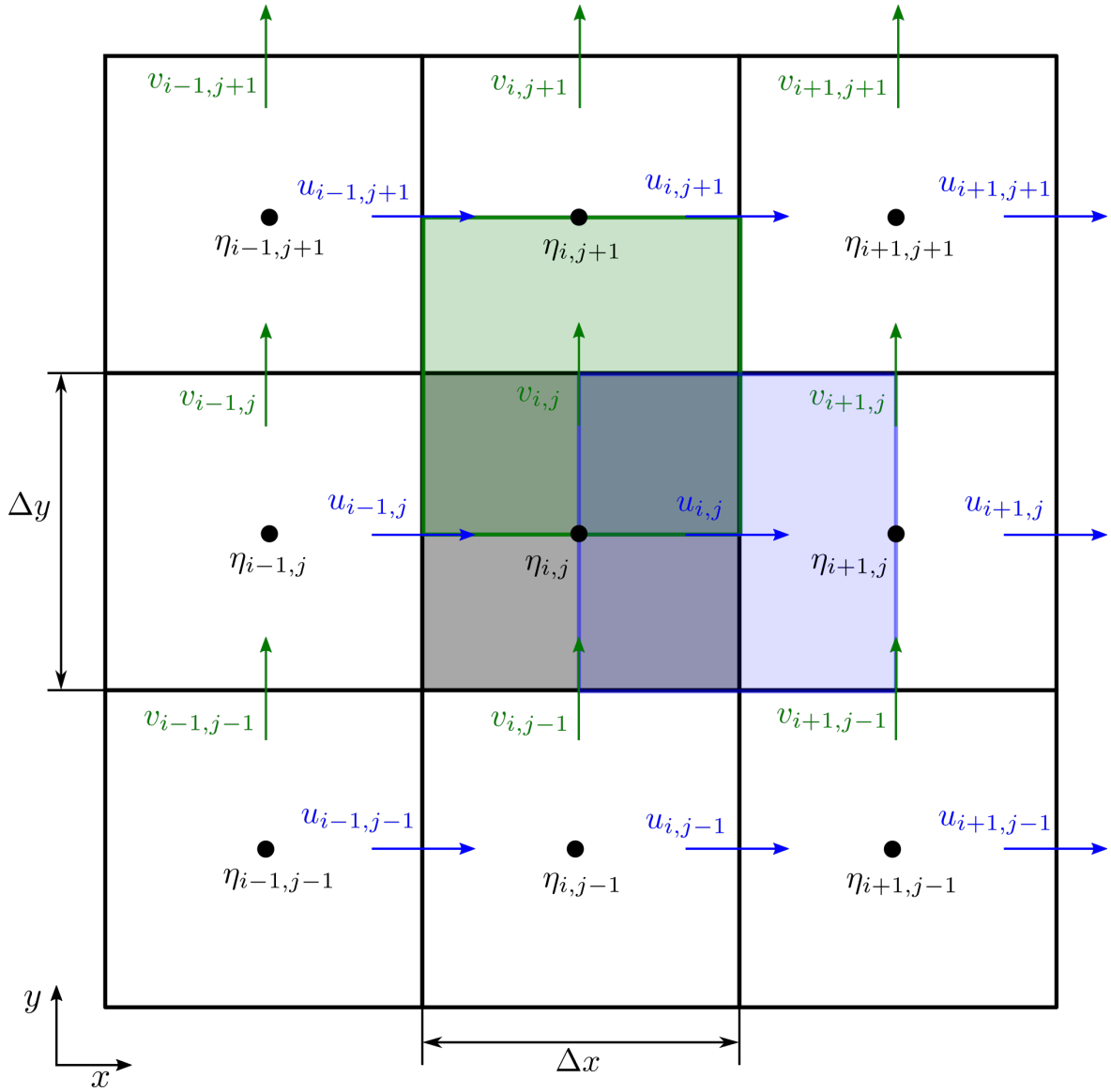


Figure 2.1: Scheme of the Arakawa C-grid.

It is worth noting that the CV shown are centered for the h-grid, which correspond to the gray one, and the staggered for each component of the velocity vector. There is one final comment to be added: if the potential vorticity is computed this will be in the upper right corner of the CV.

2.2 Pressure gradient contribution

The pressure gradient contribution will be calculated using a finite difference method. Due to the fact that this is a well-known method, an accuracy test will not be performed and its discretization is straight-forward. This method is based on the Taylor expansion. The expression of the expansion of a function $f(x)$ around a point $x = x_n$ can be found in Eq. (2.2) and can be used to calculate a first order approximation of the derivative evaluating this resulting function in the next point of the domain ($x_{n+1} = x_n + \Delta x$) as shown in Eq. (2.4)

$$f(x) = \sum_{k=0}^{\infty} \frac{1}{k!} \left. \frac{d^k f(x)}{dx^k} \right|_{x_n} (x - x_n)^n \tag{2.2}$$

$$f(x_{n+1}) = f(x_n) + \left. \frac{df(x)}{dx} \right|_{x_n} \Delta x \tag{2.3}$$

$$\left. \frac{df(x)}{dx} \right|_{x_n} = \frac{f(x_{n+1}) - f(x_n)}{\Delta x} \tag{2.4}$$

The results of the approximation of the derivative in a certain point can be used to numerically solve the term of the pressure contribution by directly applying the result and is found in Eq. (2.5) where the direction in which the derivative is computed has been taken into account.

$$P_{pu} = -g \frac{\partial \eta}{\partial x} = -g \frac{\eta_{i+1,j} - \eta_{i,j}}{\Delta x} \tag{2.5a}$$

$$P_{pv} = -g \frac{\partial \eta}{\partial y} = -g \frac{\eta_{i,j+1} - \eta_{i,j}}{\Delta y} \tag{2.5b}$$

2.3 Adective terms

One way to achieve the numerical solution of the advection terms is by using the chain rule and rewriting the advection in a flux form with some extra terms to leave the equality unchanged. Note that the expression of the continuity equation is equivalent to an advective one but already in a conservative form. The result of this transformation and the denominations of the new terms are shown in Eq. (2.6). In this new form it is possible to apply two different algorithms and solve it.

$$u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = \underbrace{\frac{\partial (uu)}{\partial x} + \frac{\partial (uv)}{\partial y}}_{P_1} - \underbrace{u\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right)}_{P_2} \tag{2.6a}$$

$$u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = \underbrace{\frac{\partial (uv)}{\partial x} + \frac{\partial (vv)}{\partial y}}_{P_1} - \underbrace{v\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right)}_{P_2} \tag{2.6b}$$

$$\nabla \cdot (h\vec{u}) = \underbrace{\frac{\partial (hu)}{\partial x} + \frac{\partial (hv)}{\partial y}}_{P_1} \tag{2.6c}$$

The first of the parts (P_1) will be solved using a Finite Volume Method (FVM) that will ensure a property called Total Variation Diminishing (TVD). This method will also be used to solve a similar term in the continuity equation and, hence, is of great interest. The second of those terms will be solved using a Finite Difference approach as seen in Section 2.2.

Solution of P_1 for the x-axis momentum equation

This term can be interpreted as a transport of a velocity due to other components of it. Hence, a FVM can be easily applied. The component in the x-direction we are interested in numerically evaluating is

$$P_{1u} = \nabla \cdot (u\vec{u})$$

The first thing to do would be to integrate this quantity over a control volume and then apply the Gauss divergence theorem to convert it into fluxes through boundaries. This would be

$$\int_{\Omega} \nabla \cdot (u\vec{u}) d\Omega = \int_{\partial\Omega} u\vec{u} \cdot d\vec{S}$$

$$\int_{\partial\Omega} u\vec{u} \cdot d\vec{S} = u_e F_e - u_w F_w + u_n F_n - u_s F_s \tag{2.7}$$

Then, an approximation of the non integrated quantity can be obtained by assuming that it can be considered to be constant inside a sufficiently small CV. That is

$$\nabla \cdot (u\vec{u}) \approx \frac{1}{\Omega} \int_{\Omega} \nabla \cdot (u\vec{u}) d\Omega = \frac{1}{\Omega} \left(u_e F_e - u_w F_w + u_n F_n - u_s F_s \right) \tag{2.8}$$

where F denotes a flux term that will consist of a velocity multiplied by the cross section of that particular face of the CV. Multiplying the expression by $-\Delta t$ one obtains

$$-\Delta t \nabla \cdot (u\vec{u}) = C_w u_w - C_e u_e + C_s u_s - C_n u_n \tag{2.9}$$

where C denotes the dimensionless Courant number. This notation will help to introduce the advection scheme with the TVD property. To follow this derivation one must consider the direction of the flux, because the schemes will strongly depend on that. To do so, the easiest way is to split all the quantities that appear on Eq. (2.9) in the positive and negative components denoted by a $(+)$ or $(-)$ sign, respectively, that will indicate that a certain quantity moves in the positive direction of the axis or in the negative one. Obviously, one must ensure that if a flux is positive the only non-zero quantities allowed are the positive ones, and the negatives should be zero. This property must be ensured once the expressions for all the variables are chosen. The result of such split is shown in Eq. (2.10).

$$P_{1u} - \Delta t \nabla \cdot (u\vec{u}) = C_w^+ u_w^+ + C_w^- u_w^- - C_e^+ u_e^+ - C_e^- u_e^- + C_s^+ u_s^+ + C_s^- u_s^- - C_n^+ u_n^+ - C_n^- u_n^- \tag{2.10}$$

Now, to choose the values for the variables involved it is necessary to recall the staggered nature of the Arakawa C-grid. For this component of the momentum equation the grid is staggered in the x-axis as it can be seen in Figure 2.1 . Therefore, not all the quantities are positioned where it is desirable and an interpolation will be needed. With this information the values of the Courant numbers can be computed and are displayed in Eq. (2.11). This numbers are in accordance to the ones presented in [9].

$$\begin{cases} C_w^+ = \frac{1}{2}\left(\frac{1}{2}\left(u_{i-1,j} + |u_{i-1,j}|\right) + \frac{1}{2}\left(u_{i,j} + |u_{i,j}|\right)\right)\frac{\Delta t}{\Delta x} \\ C_w^- = \frac{1}{2}\left(\frac{1}{2}\left(u_{i-1,j} - |u_{i-1,j}|\right) + \frac{1}{2}\left(u_{i,j} - |u_{i,j}|\right)\right)\frac{\Delta t}{\Delta x} \\ C_e^+ = \frac{1}{2}\left(\frac{1}{2}\left(u_{i,j} + |u_{i,j}|\right) + \frac{1}{2}\left(u_{i+1,j} + |u_{i+1,j}|\right)\right)\frac{\Delta t}{\Delta x} \\ C_e^- = \frac{1}{2}\left(\frac{1}{2}\left(u_{i,j} - |u_{i,j}|\right) + \frac{1}{2}\left(u_{i+1,j} - |u_{i+1,j}|\right)\right)\frac{\Delta t}{\Delta x} \\ C_s^+ = \frac{1}{2}\left(\frac{1}{2}\left(v_{i,j-1} + |v_{i,j-1}|\right) + \frac{1}{2}\left(v_{i+1,j-1} + |v_{i+1,j-1}|\right)\right)\frac{\Delta t}{\Delta y} \\ C_s^- = \frac{1}{2}\left(\frac{1}{2}\left(v_{i,j-1} - |v_{i,j-1}|\right) + \frac{1}{2}\left(v_{i+1,j-1} - |v_{i+1,j-1}|\right)\right)\frac{\Delta t}{\Delta y} \\ C_n^+ = \frac{1}{2}\left(\frac{1}{2}\left(v_{i,j} + |v_{i,j}|\right) + \frac{1}{2}\left(v_{i+1,j} + |v_{i+1,j}|\right)\right)\frac{\Delta t}{\Delta y} \\ C_n^- = \frac{1}{2}\left(\frac{1}{2}\left(v_{i,j} - |v_{i,j}|\right) + \frac{1}{2}\left(v_{i+1,j} - |v_{i+1,j}|\right)\right)\frac{\Delta t}{\Delta y} \end{cases} \tag{2.11}$$

This result may seem confusing and complicated but looking to it carefully it is not. Take as an example C_w^+ and C_w^-. When the flow at the west face of the CV moves towards the positive direction of the x-axis the velocity interpolated as a simple mean should be obtained for the C_w^+ term while $C_w^- = 0$. Thanks to the the addition or subtraction of the absolute value of the velocities this result can be obtained. In this example the procedure would lead to

$$\begin{aligned}
\frac{1}{2}\left(u_{i-1,j} + |u_{i-1,j}|\right) &= \frac{1}{2}\left(u_{i-1,j} + u_{i-1,j}\right) = u_{i-1,j} \\
\frac{1}{2}\left(u_{i,j} + |u_{i,j}|\right) &= \frac{1}{2}\left(u_{i,j} + u_{i,j}\right) = u_{i,j} \\
\frac{1}{2}\left(u_{i-1,j} - |u_{i-1,j}|\right) &= \frac{1}{2}\left(u_{i-1,j} - u_{i-1,j}\right) = 0 \\
\frac{1}{2}\left(u_{i,j} - |u_{i,j}|\right) &= \frac{1}{2}\left(u_{i,j} - u_{i,j}\right) = 0
\end{aligned}$$

and its corresponding Courant numbers

$$\begin{aligned}
C_w^+ &= \left(\frac{u_{i-1,j} + u_{i,j}}{2}\right)\frac{\Delta t}{\Delta x} \\
C_w^- &= 0
\end{aligned}$$

which are the values expected. This intuitive justification is applicable to all the other terms attending always to the appropriate staggering. Having seen the Courant numbers it is moment to switch to the interpolation of the actual transported component of the velocity in all the faces. This is the most critic part of the the whole computation, since how this interpolation is done will determine the overall behaviour of the simulation. A very common and general notation for the interpolation of a given general variable in the eastern face of the control volume ϕ_e, is the one presented in [10], [11] which can be seen in Eq. (2.12).

$$\phi_e = \phi_{i,j} + \frac{1}{2}\Psi(r)(\phi_{i+1,j} - \phi_{i,j}) \tag{2.12}$$

The other components can be easily deduced. The additional parameters introduced are $\Psi(r)$ which stands for the limiter function and r which is the upwind-downwind gradients quotient and is defined as

$$r = \frac{\phi_{i,j} - \phi_{i-1,j}}{\phi_{i+1,j} - \phi_{i,j}} \tag{2.13}$$

This expression is only valid for a positive flow so there's a final generalization needed and the Courant numbers will be used to evaluate the direction of the flux. For the eastern face this would become

$$\begin{cases}
\phi_e^+ = \phi_{i,j} + \dfrac{1}{2}\Psi(r^+)(1 - C_e^+)(\phi_{i+1,j} - \phi_{i,j}) \\[2ex]
\phi_e^- = \phi_{i+1,j} - \dfrac{1}{2}\Psi(r^-)(1 + C_e^-)(\phi_{i+1,j} - \phi_{i,j})
\end{cases} \tag{2.14}$$

with

$$r^+ = \frac{\phi_{i,j} - \phi_{i-1,j}}{\phi_{i+1,j} - \phi_{i,j}} \tag{2.15}$$

$$r^- = \frac{\phi_{i+2,j} - \phi_{i+1,j}}{\phi_{i+1,j} - \phi_{i,j}} \tag{2.16}$$

Applying this concept to all the different components of the horizontal velocity one gets Eq. (2.17), with the corresponding r terms shown in Eq. (2.18), that are in accordance to the ones of [12]. It is worth noting that in case that the denominator of r happens to be 0, then $r = 0$ for convenience.

$$\begin{cases} u_w^+ &= u_{i-1,j} + \frac{1}{2}\Psi(r_w^+)(1 - C_w^+)(u_{i,j} - u_{i-1,j}) \\ u_w^- &= u_{i,j} - \frac{1}{2}\Psi(r_w^-)(1 + C_w^-)(u_{i,j} - u_{i-1,j}) \\ u_e^+ &= u_{i,j} + \frac{1}{2}\Psi(r_e^+)(1 - C_e^+)(u_{i+1,j} - u_{i,j}) \\ u_e^- &= u_{i+1,j} - \frac{1}{2}\Psi(r_e^-)(1 + C_e^-)(u_{i+1,j} - u_{i,j}) \\ u_s^+ &= u_{i,j-1} + \frac{1}{2}\Psi(r_s^+)(1 - C_s^+)(u_{i,j} - u_{i,j-1}) \\ u_s^- &= u_{i,j} - \frac{1}{2}\Psi(r_s^-)(1 + C_s^-)(u_{i,j} - u_{i,j-1}) \\ u_n^+ &= u_{i,j} + \frac{1}{2}\Psi(r_n^+)(1 - C_n^+)(u_{i,j+1} - u_{i,j}) \\ u_n^- &= u_{i,j+1} - \frac{1}{2}\Psi(r_n^-)(1 + C_n^-)(u_{i,j+1} - u_{i,j}) \end{cases} \tag{2.17}$$

$$\begin{cases} r_w^+ &= \dfrac{u_{i-1,j} - u_{i-2,j}}{u_{i,j} - u_{i-1,j}} \\ r_w^- &= \dfrac{u_{i+1,j} - u_{i,j}}{u_{i,j} - u_{i-1,j}} \\ r_e^+ &= \dfrac{u_{i,j} - u_{i-1,j}}{u_{i+1,j} - u_{i,j}} \\ r_e^- &= \dfrac{u_{i+2,j} - u_{i+1,j}}{u_{i+1,j} - u_{i,j}} \\ r_s^+ &= \dfrac{u_{i,j-1} - u_{i,j-2}}{u_{i,j} - u_{i,j-1}} \\ r_s^- &= \dfrac{u_{i,j+1} - u_{i,j}}{u_{i,j} - u_{i,j-1}} \\ r_n^+ &= \dfrac{u_{i,j} - u_{i,j-1}}{u_{i,j+1} - u_{i,j}} \\ r_n^- &= \dfrac{u_{i,j+2} - u_{i,j+1}}{u_{i,j+1} - u_{i,j}} \end{cases} \tag{2.18}$$

The only aspect left to close this term is to specify the flux limiter $\Psi(r)$. Since this is a very important topic and the results will strongly depend on it, an individual section will be devoted to discuss the different options and its corresponding advantages and drawbacks. Therefore, the reader is referred to Section 2.6 to see the chosen function for the flux limiter and the reasons.

Solution of P_1 for the y-axis momentum equation

In this part, the solution for the first convection term of the y-axis momentum equation will be discussed. Its mathematical expression is

$$P_{1v} = \nabla \cdot (v\vec{u})$$

Then, following a similar procedure, we can integrate this quantity over a control volume and apply the Gauss divergence theorem to obtain an approximate expression for the convection. This procedure leads to

$$\nabla \cdot (v\vec{u}) \approx \frac{1}{\Omega}\int_\Omega \nabla \cdot (v\vec{u}) d\Omega = \frac{1}{\Omega}\left(v_e F_e - v_w F_w + v_n F_n - v_s F_s\right) \tag{2.19}$$

Then, this term will be multiplied by $-\Delta t$ to obtain the Courant numbers and again, the flux will be divided into the positive and negative contributions. This would read as

$$-\Delta t\nabla\cdot(v\vec{u}) = C_w v_w - C_e v_e + C_s v_s - C_n v_n \tag{2.20}$$

$$P_{1v} = -\Delta t\nabla\cdot(v\vec{u}) = C_w^+ v_w^+ + C_w^- v_w^- - C_e^+ v_e^+ - C_e^- v_e^- + C_s^+ v_s^+ + C_s^- v_s^- - C_n^+ v_n^+ - C_n^- v_n^- \tag{2.21}$$

As it was done before, the Courant numbers can be evaluated taking into account the staggering of the grid as seen in Figure 2.1. The results for this Courant numbers are found in Eq. (2.22).

$$\begin{cases} C_w^+ = \frac{1}{2}\left(\frac{1}{2}\left(u_{i-1,j+1} + |u_{i-1,j+1}|\right) + \frac{1}{2}\left(u_{i-1,j} + |u_{i-1,j}|\right)\right)\frac{\Delta t}{\Delta x} \\ C_w^- = \frac{1}{2}\left(\frac{1}{2}\left(u_{i-1,j+1} - |u_{i-1,j+1}|\right) + \frac{1}{2}\left(u_{i-1,j} - |u_{i-1,j}|\right)\right)\frac{\Delta t}{\Delta x} \\ C_e^+ = \frac{1}{2}\left(\frac{1}{2}\left(u_{i,j+1} + |u_{i,j+1}|\right) + \frac{1}{2}\left(u_{i,j} + |u_{i,j}|\right)\right)\frac{\Delta t}{\Delta x} \\ C_e^- = \frac{1}{2}\left(\frac{1}{2}\left(u_{i,j+1} - |u_{i,j+1}|\right) + \frac{1}{2}\left(u_{i,j} - |u_{i,j}|\right)\right)\frac{\Delta t}{\Delta x} \\ C_s^+ = \frac{1}{2}\left(\frac{1}{2}\left(v_{i,j-1} + |v_{i,j-1}|\right) + \frac{1}{2}\left(v_{i,j} + |v_{i,j}|\right)\right)\frac{\Delta t}{\Delta y} \\ C_s^- = \frac{1}{2}\left(\frac{1}{2}\left(v_{i,j-1} - |v_{i,j-1}|\right) + \frac{1}{2}\left(v_{i,j} - |v_{i,j}|\right)\right)\frac{\Delta t}{\Delta y} \\ C_n^+ = \frac{1}{2}\left(\frac{1}{2}\left(v_{i,j} + |v_{i,j}|\right) + \frac{1}{2}\left(v_{i,j+1} + |v_{i,j+1}|\right)\right)\frac{\Delta t}{\Delta y} \\ C_n^- = \frac{1}{2}\left(\frac{1}{2}\left(v_{i,j} - |v_{i,j}|\right) + \frac{1}{2}\left(v_{i,j+1} - |v_{i,j+1}|\right)\right)\frac{\Delta t}{\Delta y} \end{cases} \tag{2.22}$$

Then, the vertical components of the velocity have to be interpolated and, similarly, this will be done using a general formulation that will allow the application of different numerical schemes just by choosing the limiter function. This velocities are expressed in Eq. (2.23) and its corresponding r terms in Eq. (2.24). Again, this limiting function will determine how this solution will be in terms of accuracy and numerical stability, so it is of major interest to choose wisely the one to be used. This topic will be discussed in Section 2.6.

$$\begin{cases} v_w^+ &= v_{i-1,j} + \frac{1}{2}\Psi(r_w^+)(1 - C_w^+)(v_{i,j} - v_{i-1,j}) \\ v_w^- &= v_{i,j} - \frac{1}{2}\Psi(r_w^-)(1 + C_w^-)(v_{i,j} - v_{i-1,j}) \\ v_e^+ &= v_{i,j} + \frac{1}{2}\Psi(r_e^+)(1 - C_e^+)(v_{i+1,j} - v_{i,j}) \\ v_e^- &= v_{i+1,j} - \frac{1}{2}\Psi(r_e^-)(1 + C_e^-)(v_{i+1,j} - v_{i,j}) \\ v_s^+ &= v_{i,j-1} + \frac{1}{2}\Psi(r_s^+)(1 - C_s^+)(v_{i,j} - v_{i,j-1}) \\ v_s^- &= v_{i,j} - \frac{1}{2}\Psi(r_s^-)(1 + C_s^-)(v_{i,j} - v_{i,j-1}) \\ v_n^+ &= v_{i,j} + \frac{1}{2}\Psi(r_n^+)(1 - C_n^+)(v_{i,j+1} - v_{i,j}) \\ v_n^- &= v_{i,j+1} - \frac{1}{2}\Psi(r_n^-)(1 + C_n^-)(v_{i,j+1} - v_{i,j}) \end{cases} \tag{2.23}$$

$$\begin{cases} r_w^+ &= \dfrac{v_{i-1,j} - v_{i-2,j}}{v_{i,j} - v_{i-1,j}} \\ r_w^- &= \dfrac{v_{i+1,j} - v_{i,j}}{v_{i,j} - v_{i-1,j}} \\ r_e^+ &= \dfrac{v_{i,j} - v_{i-1,j}}{v_{i+1,j} - v_{i,j}} \\ r_e^- &= \dfrac{v_{i+2,j} - v_{i+1,j}}{v_{i+1,j} - v_{i,j}} \\ r_s^+ &= \dfrac{v_{i,j-1} - v_{i,j-2}}{v_{i,j} - v_{i,j-1}} \\ r_s^- &= \dfrac{v_{i,j+1} - v_{i,j}}{v_{i,j} - v_{i,j-1}} \\ r_n^+ &= \dfrac{v_{i,j} - v_{i,j-1}}{v_{i,j+1} - v_{i,j}} \\ r_n^- &= \dfrac{v_{i,j+2} - v_{i,j+1}}{v_{i,j+1} - v_{i,j}} \end{cases} \tag{2.24}$$

With all this information the first term of the advection for the y momentum equation can be computed.

Solution of P_1 for the continuity equation

Finally, the last element that will be treated using a high order scheme is the one that appears on the continuity equation since it has a conservative form and its perfect to be applied to. The only different aspect to take into account with respect to the previous components is the staggering of the mesh. Recall that h and η are computed in the centered grid and thus, the final expression will differ. Despite that, this is the easiest term because after all some fluxes will be needed and the velocities will already be on the faces of the CV. This can be seen from the scheme of Figure 2.1.

Starting from the beginning, the Gauss theorem must be applied to the volume integral over the quantity $P_{1h} = \nabla \cdot (h\vec{u})$ leading to

$$\nabla \cdot (h\vec{u}) \approx \frac{1}{\Omega} \int_{\Omega} \nabla \cdot (h\vec{u}) d\Omega = \frac{1}{\Omega} \left(h_e F_e - h_w F_w + h_n F_n - h_s F_s \right) \tag{2.25}$$

Then, we multiply the expression by $-\Delta t$ to be able to apply the formulation here followed. The result is, once having the terms split in the flux direction,

$$P_{1h} = -\Delta t \nabla \cdot (h\vec{u}) = C_w^+ h_w^+ + C_w^- h_w^- - C_e^+ h_e^+ - C_e^- h_e^- + C_s^+ h_s^+ + C_s^- h_s^- - C_n^+ h_n^+ - C_n^- h_n^- \tag{2.26}$$

As it has been explained before, the Courant numbers can be found attending to the particular staggering and the results for this case are displayed in Eq. (2.27).

$$\begin{cases} C_w^+ = \frac{1}{2} \left(u_{i-1,j} + |u_{i-1,j}| \right) \frac{\Delta t}{\Delta x} \\ C_w^- = \frac{1}{2} \left(u_{i-1,j} - |u_{i-1,j}| \right) \frac{\Delta t}{\Delta x} \\ C_e^+ = \frac{1}{2} \left(u_{i,j} + |u_{i,j}| \right) \frac{\Delta t}{\Delta x} \\ C_e^- = \frac{1}{2} \left(u_{i,j} - |u_{i,j}| \right) \frac{\Delta t}{\Delta x} \\ C_s^+ = \frac{1}{2} \left(u_{i,j-1} + |u_{i,j-1}| \right) \frac{\Delta t}{\Delta x} \\ C_s^- = \frac{1}{2} \left(u_{i,j-1} - |u_{i,j-1}| \right) \frac{\Delta t}{\Delta x} \\ C_n^+ = \frac{1}{2} \left(u_{i,j} + |u_{i,j}| \right) \frac{\Delta t}{\Delta x} \\ C_n^- = \frac{1}{2} \left(u_{i,j} - |u_{i,j}| \right) \frac{\Delta t}{\Delta x} \end{cases} \tag{2.27}$$

As expected, the expressions are slightly simpler than for the momentum equations. Then, the height of the fluid column must be interpolated using a certain limiter function $\Psi(r)$. The expressions for the interpolated values as well as the upwind-downwind ratios can be seen in Eq. (2.28) and Eq. (2.29), respectively.

$$\begin{cases} h_w^+ &= h_{i-1,j} + \frac{1}{2} \Psi(r_w^+)(1 - C_w^+)(h_{i,j} - h_{i-1,j}) \\ h_w^- &= h_{i,j} - \frac{1}{2} \Psi(r_w^-)(1 + C_w^-)(h_{i,j} - h_{i-1,j}) \\ h_e^+ &= h_{i,j} + \frac{1}{2} \Psi(r_e^+)(1 - C_e^+)(h_{i+1,j} - h_{i,j}) \\ h_e^- &= h_{i+1,j} - \frac{1}{2} \Psi(r_e^-)(1 + C_e^-)(h_{i+1,j} - h_{i,j}) \\ h_s^+ &= h_{i,j-1} + \frac{1}{2} \Psi(r_s^+)(1 - C_s^+)(h_{i,j} - h_{i,j-1}) \\ h_s^- &= h_{i,j} - \frac{1}{2} \Psi(r_s^-)(1 + C_s^-)(h_{i,j} - h_{i,j-1}) \\ h_n^+ &= h_{i,j} + \frac{1}{2} \Psi(r_n^+)(1 - C_n^+)(h_{i,j+1} - h_{i,j}) \\ h_n^- &= h_{i,j+1} - \frac{1}{2} \Psi(r_n^-)(1 + C_n^-)(h_{i,j+1} - h_{i,j}) \end{cases} \tag{2.28}$$

$$\begin{cases} r_w^+ &= \dfrac{h_{i-1,j} - h_{i-2,j}}{h_{i,j} - h_{i-1,j}} \\ r_w^- &= \dfrac{h_{i+1,j} - h_{i,j}}{h_{i,j} - h_{i-1,j}} \\ r_e^+ &= \dfrac{h_{i,j} - h_{i-1,j}}{h_{i+1,j} - h_{i,j}} \\ r_e^- &= \dfrac{h_{i+2,j} - h_{i+1,j}}{h_{i+1,j} - h_{i,j}} \\ r_s^+ &= \dfrac{h_{i,j-1} - h_{i,j-2}}{h_{i,j} - h_{i,j-1}} \\ r_s^- &= \dfrac{h_{i,j+1} - h_{i,j}}{h_{i,j} - h_{i,j-1}} \\ r_n^+ &= \dfrac{h_{i,j} - h_{i,j-1}}{h_{i,j+1} - h_{i,j}} \\ r_n^- &= \dfrac{h_{i,j+2} - h_{i,j+1}}{h_{i,j+1} - h_{i,j}} \end{cases} \tag{2.29}$$

Solution of P_2 for the x-axis momentum equation

This term might seem also as complicated as the previous one, but it turns out that due the the quasi-linear form of it can be treated in a much easier way. Recalling the finite difference approach explained in Section 2.2, a similar procedure can be used to approximate this P_2 term. But in this case, the staggering is different and hence, the final expressions will differ a little bit. For the case of the x-axis momentum equation the expression to be evaluated is

$$P_{2u} = u\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right)$$

The term u that premultiplies the entire expression is as simple as to be evaluated as $u_{i,j}$. Then, the two derivatives are left. The first of those can be evaluated as a second order approximation of the derivative like

$$\frac{\partial u}{\partial x} = \frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} + \mathcal{O}(\Delta x^2) \tag{2.30}$$

For the v related term it becomes a little bit more complicated since the variable itself is not in the correct position in the x-axis. Therefore, an interpolation is needed before the actual derivative is computed. Calling v_1 and v_2 the interpolated vertical velocity at the x position of u, its expressions are

$$\begin{aligned} v_1 &= \frac{1}{2}\left(v_{i,j} + v_{i+1,j}\right) \\ v_2 &= \frac{1}{2}\left(v_{i,j-1} + v_{i+1,j-1}\right) \end{aligned} \tag{2.31}$$

With this variables on the correct spot, the derivative can be computed, with second order spatial accuracy, as

$$\frac{\partial v}{\partial y} = \frac{v_1 - v_2}{\Delta y} + \mathcal{O}(\Delta y^2) \tag{2.32}$$

Rewriting everything in a single expression, the evaluation of the P_{2u} term will be done as

$$P_{2u} = u_{i,j}\left(\frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} + \frac{(v_{i,j} + v_{i+1,j}) - (v_{i,j-1} + v_{i+1,j-1})}{2\Delta y}\right) \tag{2.33}$$

Solution of P_2 for the y-axis momentum equation

This term can also be approximated in a finite difference approach but attending to its different staggering. The expression to be evaluated for this case is

$$P_{2v} = v\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right)$$

According to Figure 2.1 the evaluation of the derivative in the y-axis is rather natural and can be done as

$$\frac{\partial v}{\partial y} = \frac{v_{i,j+1} - v_{i,j-1}}{2\Delta y} + \mathcal{O}(\Delta y^2) \tag{2.34}$$

whereas the derivative in x must be treated in a special fashion. This means that a previous interpolation is needed. The interpolated terms are

$$\begin{aligned} u_1 &= \frac{1}{2}\left(u_{i,j} + u_{i,j+1}\right) \\ u_2 &= \frac{1}{2}\left(u_{i-1,j+1} + u_{i-1,j}\right) \end{aligned} \tag{2.35}$$

Which leads to an evaluation of the derivative as

$$\frac{\partial u}{\partial x} = \frac{u_1 - u_2}{\Delta x} + \mathcal{O}(\Delta x^2) \tag{2.36}$$

Summing up, the P_2 term regarding the y-axis momentum equation can be approximated using a second order spatial approximation like

$$P_{2v} = v_{i,j}\left(\frac{(u_{i,j} + u_{i,j+1}) - (u_{i-1,j+1} + u_{i-1,j})}{2\Delta x} + \frac{v_{i,j+1} - v_{i,j-1}}{2\Delta y}\right) \tag{2.37}$$

2.4 Time integration

Putting Coriolis aside because this term needs a special treatment, the equations that are being solved for now are

$$\begin{aligned} \frac{\partial u}{\partial t} &= f_1(u, v, \eta, x, y, t) \\ \frac{\partial v}{\partial t} &= f_2(u, v, \eta, x, y, t) \\ \frac{\partial \eta}{\partial t} &= f_3(u, v, \eta, x, y, t) \end{aligned} \tag{2.38}$$

where f_1, f_2 denote minus the advection term and pressure gradients and f_3 stands for minus the advection-like term of the continuity equation.

There are plenty of ways to integrate this type of equations but for atmospheric applications the most used type of schemes are the ones called explicit. This schemes are widely used because are simpler to implement, since they use information from previous time steps to predict the value of the function in the following ones. The main drawback of this type of methods, though, is that there is a maximum permitted time step allowed for convergence of the entire scheme. The most widely known of this criteria is the so-called Courant-Friedrichs-Levy (CFL) condition. This condition states that for the scheme to converge to the actual solution the Courant number denoted by C must be less than 1. The way in which this number is computed is

$$C = \frac{U\Delta t}{\Delta} \tag{2.39}$$

where U stands for the norm of the velocity vector, and Δ stands for a spatial increment in any direction. Since the SW equations are two-dimensional, this term can either be Δx or Δy. For a non-regular mesh and non uniform velocity distribution the Courant number that will be taken into account is the most restrictive one.

The expression that will be used to compute the largest time step possible will be

$$\Delta t = \min_{(x,y)\in\Omega} \frac{C\Delta}{U} \tag{2.40}$$

Remember that C is at most equal to 1 but normally is set to be a smaller number (e.g. $C = 0.5$) to have a little bit of a margin. For usual CFD applications this criterion is very restrictive due to small Δ and relatively large velocities, but for atmospheric applications, distances are very large ($\Delta \sim \mathcal{O}(10^5)$) while velocities do not grow that much ($U \sim \mathcal{O}(10^2)$). Therefore, the resulting Δt is more permissive and will suit perfectly for the application here looked.

Now that the usage of an explicit scheme has been justified, it's time to choose one. The easiest type of an explicit scheme is the so-called Euler method. Considering an integration forward in time, this would read as Eq. (2.41). Notice that this scheme is as simple as to integrate along the time using a first-order approximation like in a finite difference approach.

$$\begin{aligned} u^{n+1} &= u^n + \Delta t f_1^n \\ v^{n+1} &= v^n + \Delta t f_2^n \\ \eta^{n+1} &= \eta^n + \Delta t f_3^n \end{aligned} \tag{2.41}$$

The main problem of this type of scheme is that is only first order accurate in time, and will produce a great amount of energy dissipation. Then, the scheme that will be used is a third order Adams-Bashforth. This family of schemes are of the type Linear Multistep Formulas and a brief description of this technique following reference [13] will be presented. Considering a general ODE of the form

$$y' = f(x, y) \tag{2.42}$$

One can integrate is as

$$y(x_{n+1}) = y(x_n) + \int_{x_n}^{x_{n+1}} f(t, y(t))dt \tag{2.43}$$

The point of this method is to take into account the reconstructed part of the function, which is $f_i = f(x_i, y_i)$ and replace the function $f(t, y(t))$ of Eq. (2.43) by an interpolation polynomial in $\{(x_i, f_i) \mid i \in [n-k+1, n]\}$. Following the notation of [13], this polynomial can be assembled as

$$p(t) = p(x_n + sh) = \sum_{j=0}^{k-1} (-1)^j \binom{-s}{j} \nabla^j f_n \tag{2.44}$$

where

$$\nabla^{j+1} f_n = \nabla^j f_n - \nabla^j f_{n-1} \tag{2.45}$$

and

$$\nabla^0 f_n = f_n \tag{2.46}$$

Introducing this expressions into Eq. (2.43), we obtain

$$y_{n+1} = y_n + h \sum_{j=0}^{k-1} \gamma_j \nabla^j f_n \tag{2.47}$$

where the coefficients γ_j help to compact the notation and can be evaluated as

$$\gamma_j = (-1)^j \int_0^1 \binom{-s}{j} ds \tag{2.48}$$

Finally, instead of evaluating this expression, reference [13] provides a very useful recursive formula to compute the m^{th} coefficient which is

$$\gamma_m + \frac{1}{2}\gamma_{m-1} + \frac{1}{3}\gamma_{m-2} + ... + \frac{1}{m+1}\gamma_0 = 1 \tag{2.49}$$

with $\gamma_0 = 1$

With all this information some Adams-Bashforth schemes can be obtained. Particularly, up to $k = 3$ will be used throughout this project. First of all, setting $k = 1$ in Eq. (2.47) and computing the correspondent coefficient using Eq. (2.49) the resulting expression for this scheme is

$$y_{n+1} = y_n + h\left(\gamma_0 \nabla^0 f_n\right) \tag{2.50}$$

Adding the values of

$$\gamma_0 = 1$$

and

$$\nabla^0 f_n = f_n$$

the expression for the Adams-Bashforth scheme of first order obtained is shown in Eq. (2.51), which corresponds to the Euler scheme already explained.

$$y_{n+1} = y_n + h f_n \tag{2.51}$$

Moving on into the second order scheme, one needs to set $k = 2$, and the resulting expression is

$$y_{n+1} = y_n + h\left(\gamma_0 \nabla^0 f_n + \gamma_1 \nabla^1 f_n\right) \tag{2.52}$$

the new terms involved can be computed as

$$\gamma_1 + \frac{1}{2}\gamma_0 = 1$$

$$\gamma_1 = \frac{1}{2}$$

and

$$\nabla^1 f_n = \nabla^0 f_n - \nabla^0 f_{n-1} = f_n - f_{n-1}$$

which introduced into Eq. (2.52) the result is

$$y_{n+1} = y_n + h\left(f_n + \frac{1}{2}(f_n - f_{n-1})\right) \tag{2.53}$$

Rearranging the terms, the final expression for the second order Adams-Bashforth scheme is obtained. This is displayed in Eq. (2.54).

$$y_{n+1} = y_n + h\left(\frac{3}{2}f_n - \frac{1}{2}f_{n-1}\right) \tag{2.54}$$

Finally, the last scheme that will be presented is the third order, which is the one that will be used to integrate in time the Shallow Water equations. Setting $k = 3$ in Eq. (2.47) we get

$$y_{n+1} = y_n + h\left(\gamma_0 \nabla^0 f_n + \gamma_1 \nabla^1 f_n + \gamma_2 \nabla^2 f_n\right) \tag{2.55}$$

where the new terms are

$$\gamma_2 + \frac{1}{2}\gamma_1 + \frac{1}{3}\gamma_0 = 1$$
$$\gamma_2 = \frac{5}{12}$$

and

$$\nabla^2 f_n = \nabla^1 f_n - \nabla^1 f_{n-1} = f_n - f_{n-1} - f_{n-1} + f_{n-2} = f_n - 2f_{n-1} + f_{n-2}$$

Which lead to

$$y_{n+1} = y_n + h\left(f_n + \frac{1}{2}(f_n - f_{n-1}) + \frac{5}{12}(f_n - 2f_{n-1} + f_{n-2})\right) \tag{2.56}$$

Rearranging the terms, the final expression for the third order scheme can be obtained, and this is

$$y_{n+1} = y_n + h\left(\frac{23}{12}f_n - \frac{4}{3}f_{n-1} + \frac{5}{12}f_{n-2}\right) \tag{2.57}$$

Which applied to the SW equations, leads to

$$\begin{aligned} u^{n+1} &= u^n + \Delta t\left(\frac{23}{12}f_1^n - \frac{4}{3}f_1^{n-1} + \frac{5}{12}f_1^{n-2}\right) \\ v^{n+1} &= v^n + \Delta t\left(\frac{23}{12}f_2^n - \frac{4}{3}f_2^{n-1} + \frac{5}{12}f_2^{n-2}\right) \\ \eta^{n+1} &= \eta^n + \Delta t\left(\frac{23}{12}f_3^n - \frac{4}{3}f_3^{n-1} + \frac{5}{12}f_3^{n-2}\right) \end{aligned} \tag{2.58}$$

For convenience we can introduce a new notation. This is to rewrite the equations as

$$\begin{aligned} u^{n+1} &= u^n + \Delta u^{n+1} \\ v^{n+1} &= v^n + \Delta v^{n+1} \\ \eta^{n+1} &= \eta^n + \Delta \eta^{n+1} \end{aligned} \tag{2.59}$$

with

$$\begin{aligned} \Delta u^{n+1} &= \Delta t\left(\frac{23}{12}f_1^n - \frac{4}{3}f_1^{n-1} + \frac{5}{12}f_1^{n-2}\right) \\ \Delta v^{n+1} &= \Delta t\left(\frac{23}{12}f_2^n - \frac{4}{3}f_2^{n-1} + \frac{5}{12}f_2^{n-2}\right) \\ \Delta \eta^{n+1} &= \Delta t\left(\frac{23}{12}f_3^n - \frac{4}{3}f_3^{n-1} + \frac{5}{12}f_3^{n-2}\right) \end{aligned} \tag{2.60}$$

2.5 Coriolis integration

In this section the Coriolis term will be integrated. As will be shown, this term must be treated in a different fashion. Here, the procedure shown in [4] will be presented. Let's start by assuming a pure

free motion on a rotating plane. These equations are

$$\begin{aligned} \frac{du}{dt} - fv = 0 \\ \frac{dv}{dt} + fu = 0 \end{aligned} \tag{2.61}$$

Assuming a simple Euler method, this set of equations can be integrated as

$$\begin{aligned} \tilde{u}^{n+1} = \tilde{u}^n + f\Delta t\tilde{v}^n \\ \tilde{v}^{n+1} = \tilde{v}^n - f\Delta t\tilde{u}^n \end{aligned} \tag{2.62}$$

where the tilde $(\tilde{u}, \tilde{v})$ has been used to denote the numerical solution in opposition to the exact one. The important fact now is to see whether this algorithm leads to a stable result or not. To do so, the norm of the numerical velocity vector can be obtained leading to

$$||\tilde{\vec{u}}^{\,n+1}||^2 = (\tilde{u}^{n+1})^2 + (\tilde{v}^{n+1})^2 = \left(1 + f^2\Delta t^2\right)\left\{(\tilde{u}^n)^2 + (\tilde{v}^n)^2\right\} \tag{2.63}$$

And the application of this formula recursively leads to

$$||\tilde{\vec{u}}||^2 = (\tilde{u}^n)^2 + (\tilde{v}^n)^2 = \left(1 + f^2\Delta t^2\right)^n\left\{(\tilde{u}^0)^2 + (\tilde{v}^0)^2\right\} \tag{2.64}$$

And here, the problem can be spotted, since the term $\left(1 + f^2\Delta t^2\right)^n$ blows up to infinity even for small values of Δt. Physically, this would mean an uncontrolled increase in kinetic energy of the whole system. Then, an explicit method to integrate this term is completely unstable and not desired. The second obvious choice is to try a fully implicit scheme. This would mean that the update for the velocities should be done taking into account the velocities at the following time step. This would read as

$$\begin{aligned} \tilde{u}^{n+1} = \tilde{u}^n + f\Delta t\tilde{v}^{n+1} \\ \tilde{v}^{n+1} = \tilde{v}^n - f\Delta t\tilde{u}^{n+1} \end{aligned} \tag{2.65}$$

Performing a similar analysis with this type of integration scheme the obtained result for the norm of the velocity field is

$$||\tilde{\vec{u}}||^2 = (\tilde{u}^n)^2 + (\tilde{v}^n)^2 = \left(1 + f^2\Delta t^2\right)^{-n}\left\{(\tilde{u}^0)^2 + (\tilde{v}^0)^2\right\} \tag{2.66}$$

In this case, although it is technically stable, the term $\left(1 + f^2\Delta t^2\right)^{-n}$ indicates a dissipation of energy that is unrealistic. Again, this property is not desired and another one should be chosen. The final attempt is to try a semi-implicit scheme. This means to take into account both the value of the variable at time n and at time $n+1$ and weight each one of the contributions. This can be expressed as

$$\begin{aligned} \tilde{u}^{n+1} = \tilde{u}^n + f\Delta t\left[(1-\alpha)\tilde{v}^n + \alpha\tilde{v}^{n+1}\right] \\ \tilde{v}^{n+1} = \tilde{v}^n - f\Delta t\left[(1-\alpha)\tilde{u}^n + \alpha\tilde{u}^{n+1}\right] \end{aligned} \tag{2.67}$$

One can recover the explicit and the implicit schemes by setting $\alpha = 0$ and $\alpha = 1$, respectively. Performing an analysis of the norm of the numerical velocity vector would lead, according to [4], to

$$||\tilde{\vec{u}}||^2 = (\tilde{u}^n)^2 + (\tilde{v}^n)^2 = \left(\frac{1 + (1-\alpha)^2 f^2\Delta t^2}{1 + \alpha^2 f^2\Delta t^2}\right)^n\left\{(\tilde{u}^0)^2 + (\tilde{v}^0)^2\right\} \tag{2.68}$$

If we set $\alpha = 1/2$, the energy of the system will remain constant, which is the desired result: a stable scheme with no energy dissipation. For our particular case, we will start by computing a predictor velocity adding the contributions of both the advection and pressure gradients and performing the third order Adams-Bashforth time integration. After this predictor velocity, called u^* and v^*, is found, the final velocity vector for the next time step can be computed (attending to the proper staggering of the grid) as

$$\begin{aligned} u_{i,j}^{n+1} &= \frac{u_{i,j}^* - (\alpha f \Delta t)^2 u_{i,j}^n + 0.5\alpha f \Delta t (v_{i,j}^n + v_{i+1,j}^n + v_{i,j-1}^n + v_{i+1,j-1}^n)}{1 + (\alpha f \Delta t)^2} \\ v_{i,j}^{n+1} &= \frac{v_{i,j}^* - (\alpha f \Delta t)^2 v_{i,j}^n - 0.5\alpha f \Delta t (u_{i,j}^n + u_{i+1,j}^n + u_{i,j-1}^n + u_{i+1,j-1}^n)}{1 + (\alpha f \Delta t)^2} \\ \alpha &= 0.5 \end{aligned} \tag{2.69}$$

2.6 Flux limiter function

This section will be devoted to the flux limiter $\Psi(r)$ since it is one of the most important parts regarding the numerical integration of the equations. The problem arises from the fact that some schemes are conservative, but they generate wiggles, while others might not generate those spurious oscillations, yet they are too diffusive and thus, they generate unrealistic results as well.

As it is explained in [10], for a scheme to be stable, non-oscillatory and of high order the property that must be fulfilled is monotonicity preservation. This translates into the following characteristics:

1. It must not create local extrema. This will ensure the avoidance of creation of new wiggles.
2. The value of the already existing local extrema should not increase. This will ensure that given a local extrema, its value will not blow to infinity.

To evaluate the presence of wiggles, a common quantity used is the Total Variation (TV). The TV can be defined for a continuous function $f \in L^1(\Omega)$ being $\Omega \subseteq \mathbb{R}^n$, according to [14], as

$$TV(f, \Omega) = \sup \left\{ \int_\Omega f(x) \nabla \xi dx \ : \ \xi \in \mathcal{C}_c^1(\Omega, \mathbb{R}^n), \ |\xi(x)| \leq 1, \ \forall x \in \Omega \right\} \tag{2.70}$$

which can be discretized and only computed for one variable as

$$TV(f) = \sup_{\mathcal{P}} \sum_{i=0}^{n_p - 1} |f(x_{i+1}) - f(x_i)| \tag{2.71}$$

Applied to a general one-dimensional variable ϕ, like Sweby points out [15], this quantity can be interpreted as

$$TV(\phi) = \sum_{\forall k} |\phi_{k+1} - \phi_k| \tag{2.72}$$

Then, monotonicity-preserving schemes are the ones that assure that the total variation reduces with time, satisfying the property of

$$TV(\phi^{n+1}) \leq TV(\phi^n) \tag{2.73}$$

Attending to this last property, monotonicity-preserving schemes are also said to be Total Variation Diminishing (TVD). The question is then, how can a function Ψ be chosen so as to ensure that the

TVD property is met. The answer to this was found by Sweby in [15]. He gave the necessary and sufficient conditions to consider a scheme to be TVD and he did it taking into account the relation between the actual function Ψ and the upwind-downwind relation as seen in the advection solution. The relations $r - \Psi$ for TVD schemes are:

- $\Psi(r) \leq 2r \quad \forall r \in (0, 1)$
- $\Psi(r) \leq 2 \quad \forall r \in [1, \infty)$

The name of flux limiter comes from the fact that if the upwind-downwind ratio r is greater than 2, in order to keep it TVD, we must constrain it and set it to 2 at most, which can be viewed as a way to limit the flux. Some additional interesting properties of this function are that in order for a scheme to be second order accurate it needs to pass through the point $(1, 1)$ and that the range of possible second-order schemes are:

- $r \leq \Psi(r) \leq 1 \quad \forall r \in (0, 1)$
- $1 \leq \Psi(r) \leq r \quad \forall r \in [1, \infty)$

Finally, another interesting property for the schemes is to be symmetric. This property is given if and only if the relation

$$\frac{\Psi(r)}{r} = \Psi(1/r) \tag{2.74}$$

holds. If so, forward and backward gradients are treated similarly.

To close up the explanation of the properties of the flux limiter function, it must be said that for any $r \leq 0$, the limiter function should be set to $\Psi(r) = 0$. With all this information, a diagram showing the feasible region for the flux limiter function can be drawn. This is shown in Figure 2.2 as the blue shaded region.

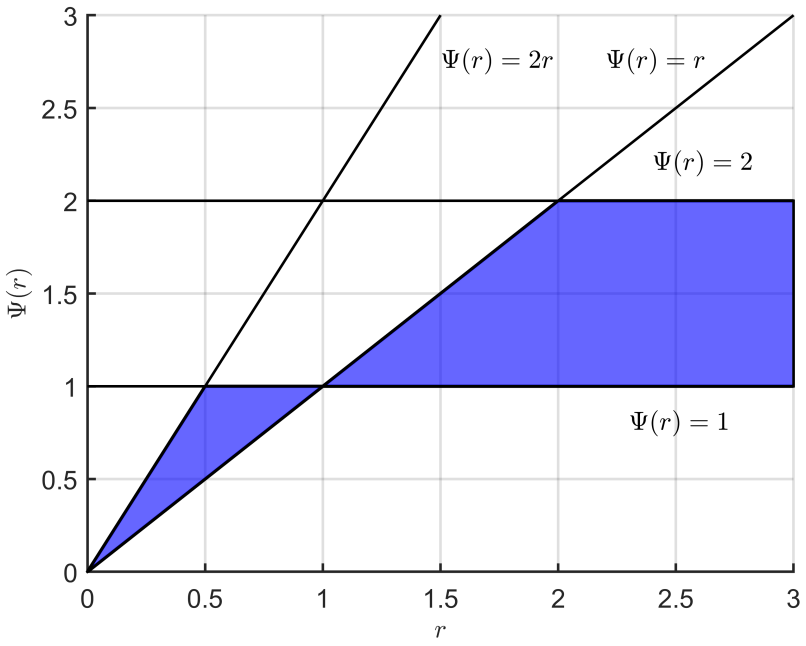


Figure 2.2: $r - \Psi$ plot with the feasible region for a second order TVD scheme flux limiter

Then, different actual flux limiter functions can be defined within the feasible region. There are plenty of those schemes available and here only some of them will be presented. It is also worth noting that with this formulation some well-known non-TVD schemes can be recovered. Starting with this set of non-TVD schemes, extracted from [10], we have:

- **Upwind** (UD): $1st$ order accurate, highly dissipative.

$$\Psi(r) = 0 \tag{2.75}$$

- **Central Difference** (CD): $2nd$ order accurate, generates wiggles.

$$\Psi(r) = 1 \tag{2.76}$$

- **Linear Upwind Differencing** (LUD): $2nd$ order accurate, dissipative.

$$\Psi(r) = r \tag{2.77}$$

- **Quadratic Upstream Interpolation for Convective Kinematics** (QUICK): $3rd$ order accurate, generates wiggles.

$$\Psi(r) = \frac{3+r}{4} \tag{2.78}$$

Some of these schemes might be used to compare the results since they are among the oldest and most studied schemes used in CFD. Then, to avoid wiggles but still have a reasonable order of accuracy and as less dissipative as possible, the TVD schemes must be taken into consideration. The following list contains some TVD schemes extracted from [11], [16], [17]:

- **Van Leer**: $2nd$ order accurate, symmetric.

$$\Psi(r) = \frac{r+|r|}{1+|r|} \tag{2.79}$$

- **Van Albada**: $2nd$ order accurate, symmetric.

$$\Psi(r) = \frac{r+r^2}{1+r^2} \tag{2.80}$$

- **MINMOD**: $2nd$ order accurate, symmetric.

$$\Psi(r) = \max\left[0, \min(r, 1)\right] \tag{2.81}$$

- **Bounded QUICK**: QUICK scheme but bounded to keep it in the TVD zone.

$$\Psi(r) = \max\left[0, \min\left(2r, \frac{3+r}{4}, 2\right)\right] \tag{2.82}$$

- **Superbee**: $2nd$ order accurate, symmetric.

$$\Psi(r) = \max\left[0, \min(2r, 1), \min(r, 2)\right] \tag{2.83}$$

- **UMIST**: $2nd$ order accurate, symmetric.

$$\Psi(r) = \max\left[0, \min\left(2r, \frac{1+3r}{4}, \frac{3+r}{4}\right), 2\right] \tag{2.84}$$

- **Koren**: $3rd$ order accurate (for smooth data), non symmetric.

$$\Psi(r) = \max\left[0, \min\left(2r, \frac{2r+1}{3}, 2\right)\right] \tag{2.85}$$

- **MUSCL**: $2rd$ order accurate, symmetric.

$$\Psi(r) = \max\left[0, \min\left(2, 2r, \frac{1+r}{2}\right)\right] \tag{2.86}$$

Plotting some of them in the Sweby diagram we get Figure 2.3. In it, we see that some of them follow the upper limit (i.e. Superbee scheme) whilst others follow the lower limit (i.e. MINMOD) and the rest move in between. Deciding a scheme is very difficult since each of them are better in different situations. Hence, the flux limiter that will be used for now is the Superbee, following the recommendation of [9] and the experience of the director of this work, Prof. Dr. Enrique García-Melendo, who has applied it, successfully, in planetary atmospheres before [2], [18], [19].

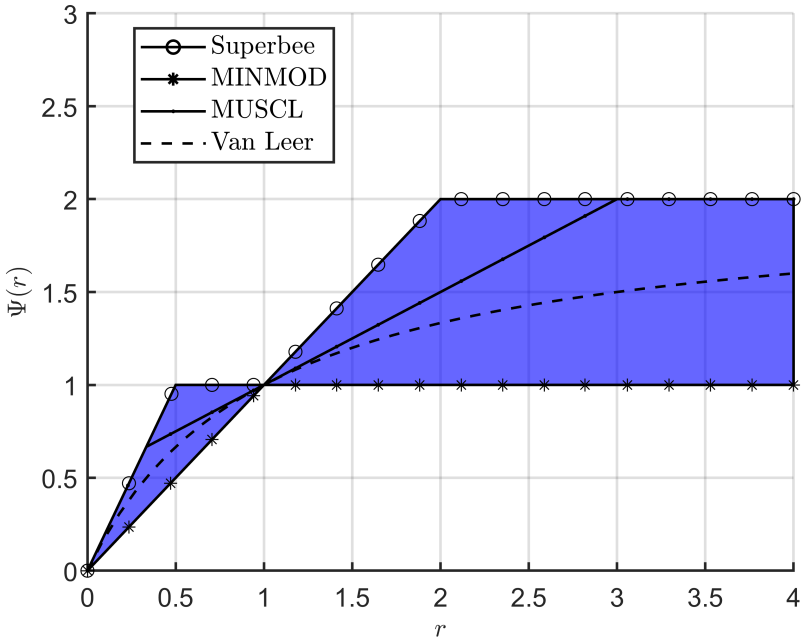


Figure 2.3: Plot of some TVD flux limiters on a Sweby diagram.

2.7 General computation scheme

At this stage, lots of details have been explained in each section, but to close up the problem, the general algorithm will be explained. The procedure to solve the Shallow Water equations will be the following:

1. Solve the mass conservation equation for η^{n+1}. To do so, the following steps have to be done

   (a) Compute the advection-like term of the mass conservation equation using Eq. (2.26) and set

$$\Delta\eta^{n+1} = P_{1h}$$

(b) Update $\Delta\eta^{n+1}$ using the Adams-Bashforth scheme shown in Eq. (2.60) -to start the algorithm use an Euler scheme and after two iterations apply the third order Adams-Bashforth-.

(c) Obtain the new elevation as

$$\eta^{n+1} = \eta^n + \Delta\eta^{n+1}$$

2. Solve the conservation of momentum equations without Coriolis. This is:

(a) Compute the first advection term P_{1u} using Eq. (2.10) and P_{1v} using Eq. (2.21).

(b) Compute the second advection term P_{2u} using Eq. (2.33) and P_{2v} using Eq. (2.37).

(c) Calculate the pressure gradient contribution P_{pu} and P_{pv} via Eq. (2.5) using η^{n+1}.

(d) Add those contributions and multiply the terms that do not have it already by the factor Δt. This is,

$$\Delta u^{n+1} = P_{1u} + \Delta t P_{2u} + \Delta t P_{pu}$$
$$\Delta v^{n+1} = P_{1v} + \Delta t P_{2v} + \Delta t P_{pv}$$

(e) Update Δu^{n+1} and Δv^{n+1} using the Adams-Bashforth scheme shown in Eq. (2.60) -to start the algorithm use an Euler scheme and after two iterations apply the third order Adams-Bashforth-.

(f) Compute the predictor velocities as

$$u^* = u^n + \Delta u^{n+1}$$
$$v^* = v^n + \Delta v^{n+1}$$

3. Finally, add the Coriolis contribution to obtain the final velocity field for the next time step by means of Eq. (2.69).

4. Check that the CFL condition is still satisfied by looking that the Δt in use is smaller than the one that comes out of Eq. (2.40).

5. Perform any other operation desired (e.g. compute the energy, save into a file, ...) and move to the next time-step.

To end this chapter just a comment on three aspects should be done. The first one is how are the zonal winds treated. These are winds that move along parallel latitudes and that are very frequent in gas giant planets. Therefore, they impose a constant speed in the horizontal component U, but if for any case there would be a meridional wind V, both will be considered. The modification of the SW momentum equations is

$$\frac{\partial u}{\partial t} + (u+U)\frac{\partial (u+U)}{\partial x} + (v+V)\frac{\partial (u+U)}{\partial y} - fv = -g\frac{\partial \eta}{\partial x} \tag{2.87a}$$
$$\frac{\partial v}{\partial t} + (u+U)\frac{\partial (v+V)}{\partial x} + (v+V)\frac{\partial (v+V)}{\partial y} + fu = -g\frac{\partial \eta}{\partial y} \tag{2.87b}$$

Numerically, though, the equations are treated in the exact same fashion.

The second aspect to take into account is the geometric effects that arise from the assumption of an ellipsoidal shape. Again, this will not change the global computation algorithm, and after evaluating the contributions of Δu^n, Δv^n, and $\Delta\eta^n$ they will be corrected by multiplying them by the corresponding factors (mainly a combination of r_Z and r_M) and thus adding the effect of a non Cartesian geometry.

Another final comment is regarding the boundary conditions. For atmospheric applications the most common ones will be full periodicity and channel ones. The latter of those will consist of periodicity in the longitudinal direction and full-slip in the northern and southern boundaries. The way in which they can be implemented is through the filling of ghost elements (or halos) in such a way the the interpolation leads to the correct boundary condition. For further details of the implementation see [3].

3 — Introduction of perturbations via geostrophic balance

In this chapter, a new technique to introduce vortices will be studied. This is of great interest because in many cases, the desired use of the simulation will be reproducing the morphology of a vortex and studying *a priori* the physics of it, might lead to a more accurate way of controlling the parameters of the steady state configuration.

The way in which now a perturbation is introduced into the Shallow Worlds code is by setting a surface elevation of the form

$$\eta(x,y) = A\exp\left\{-\left(\frac{x-x_0}{\sigma_x}\right)^2 + \left(\frac{y-y_0}{\sigma_y}\right)^2\right\} \tag{3.1}$$

during a certain amount of time and then letting the system evolve. For example, the simulations shown in Section 6.2 have been done introducing the perturbation in this fashion and the results are good but it takes up to 7 days of the simulation for the vortex to establish and this correct output has been obtained after many simulation trying out different parameters until the desired steady state vortex was outputted. This justifies the effort of looking for a new way of doing it.

Then, let's start by assuming a similar, yet more general shape of the elevation of the free surface and in terms of the latitude and longitude already. This is the one shown in Eq. (3.2).

$$\eta(\theta,\varphi) = A\exp\left\{-\left[\left(\frac{\theta-\theta_0}{a_0}\right)^2 + \left(\frac{\varphi-\varphi_0}{b_0}\right)^2\right]^n\right\} \tag{3.2}$$

The parameters are the maximum elevation A, a_0 is a value related to the major semiaxis (yet not equal) and b_0 is related to the minor semiaxis. Similarly, the terms θ_0 and φ_0 are the central longitude and latitude of the vortex, respectively. The parameter n will control the flatness of the top part of the function. Then, let's consider the SW equation neglecting the advection and considering a steady-state. The equations obtained are

$$\begin{aligned} fv &= g\frac{\partial\eta}{\partial x} \\ fu &= -g\frac{\partial\eta}{\partial y} \end{aligned} \tag{3.3}$$

This equations describe an amazing phenomenon that occurs at large scale in planetary atmospheres which is the geostrophic equilibrium (or geostrophic balance). This is the compensation of the Coriolis force and the pressure gradients. If this happens, the fluid particles, instead of tending to spread out following the pressure gradients, start rotating around and, in absence of any other force, they would reach an stationary state of a vortex. Therefore, it is a good starting point since, theoretically,

we should be able to obtain a velocity field that ensures the above equations and that after a rapid readjustment (contribution of the advection and the mass conservation, mainly) they maintain a good approximation of a vortex. Solving for the velocity field we obtain

$$\begin{aligned} u &= -\frac{g}{f}\frac{\partial \eta}{\partial y} \\ v &= \frac{g}{f}\frac{\partial \eta}{\partial x} \end{aligned} \tag{3.4}$$

Therefore, we only need to evaluate the partial derivatives of the surface elevation shown in Eq. (3.2). Since they are in different coordinate systems a change of coordinates will be needed. In this case, is as simple as

$$\begin{aligned} \frac{\partial \eta}{\partial x} &= \frac{\partial \eta}{\partial \theta}\frac{\partial \theta}{\partial x} = \frac{1}{r_Z}\frac{\partial \eta}{\partial \theta} \\ \frac{\partial \eta}{\partial y} &= \frac{\partial \eta}{\partial \varphi}\frac{\partial \varphi}{\partial y} = \frac{1}{r_M}\frac{\partial \eta}{\partial \varphi} \end{aligned} \tag{3.5}$$

Evaluating these derivatives we obtain

$$\begin{aligned} \frac{\partial \eta}{\partial \theta} &= -\eta(\theta,\varphi)\frac{2n}{a_0^2}(\theta-\theta_0)\left(\left[\frac{\theta-\theta_0}{a_0}\right]^2+\left[\frac{\varphi-\varphi_0}{b_0}\right]^2\right)^{n-1} \\ \frac{\partial \eta}{\partial \varphi} &= -\eta(\theta,\varphi)\frac{2n}{b_0^2}(\varphi-\varphi_0)\left(\left[\frac{\theta-\theta_0}{a_0}\right]^2+\left[\frac{\varphi-\varphi_0}{b_0}\right]^2\right)^{n-1} \end{aligned} \tag{3.6}$$

Defining for simplicity

$$\psi = \left(\left[\frac{\theta-\theta_0}{a_0}\right]^2+\left[\frac{\varphi-\varphi_0}{b_0}\right]^2\right)$$

the resulting velocity field is

$$\begin{aligned} u &= \frac{2ng}{r_M f b_0^2}(\varphi-\varphi_0)\psi^{n-1}\eta(\theta,\varphi) \\ v &= \frac{-2ng}{r_Z f a_0^2}(\theta-\theta_0)\psi^{n-1}\eta(\theta,\varphi) \end{aligned} \tag{3.7}$$

To have control over the vortex that will be generated it might be interesting to find which is the maximum tangential velocity. With that, it is possible to find the amplitude A. The condition of maximum tangential velocity is found at

$$\varphi|_{u_{max}} = \varphi_0 \pm b_0\left[1-\frac{1}{2n}\right]^{\frac{1}{2n}} \tag{3.8}$$

where some assumptions have been made. These are that $r_Z = r_Z(\varphi_0)$, $r_M = r_M(\varphi_0)$ and $f = f(\varphi_0)$. Otherwise, there wouldn't be an analytical solution. At this stage it is very important to note that the term b_0 is not exactly the minor semiaxis, although it tends to be as $n \to \infty$. Therefore, since normally will be easier to determine the position of maximum velocity, b_0 will be computed from Eq. (3.8). A similar effect happens with the major semiaxis which should be computed from

$$\theta|_{v_{max}} = \theta_0 \pm a_0\left[1-\frac{1}{2n}\right]^{\frac{1}{2n}} \tag{3.9}$$

Evaluating the horizontal component of the velocity at the latitude for which it's maximum, we get that

$$u_{max} = \frac{2ng}{r_M f b_0}\left(1-\frac{1}{2n}\right)^{\frac{2n-1}{2n}} A \exp\left\{-\left(1-\frac{1}{2n}\right)\right\} \tag{3.10}$$

Finally, the surface elevation parameter A can be computed as

$$A = \frac{r_M f b_0 u_{max}}{2ng}\left(1-\frac{1}{2n}\right)^{\frac{1-2n}{2n}} \exp\left\{1-\frac{1}{2n}\right\} \tag{3.11}$$

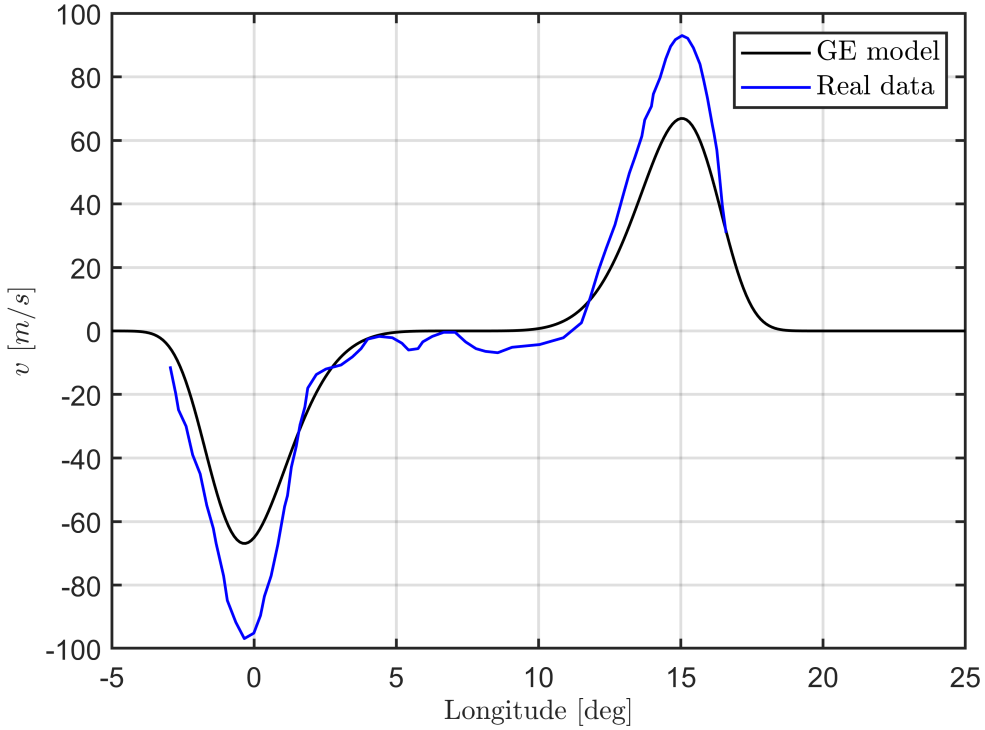
It is clear, though, that imposing this condition forces the system to have a unique solution, and will determine the maximum tangential vertical velocity v which may differ from the real one. The only final thing left to do is to compare this theoretical expressions with real data and then perform a simulation to see the results produced. A very good vortex to compare the results with is Jupiter's Great Red Spot. The main reason is that is one of the most studied vortices at the present moment and consequently plenty of information is available. The parameters will be

- $n = 3$
- $\varphi_0 = -19.62\ deg$
- $\theta_0 = 7.35\ deg$
- $\varphi|_{u_{max}} = -15.30\ deg$
- $\theta|_{v_{max}} = 15.04\ deg$
- $g = 24.79\ m/s^2$
- $\Omega = 1.76 \cdot 10^{-4}\ rad/s$
- $R_e = 71\ 492\ km$
- $R_p = 66\ 854\ km$
- $u_{max} = -126.5\ m/s$

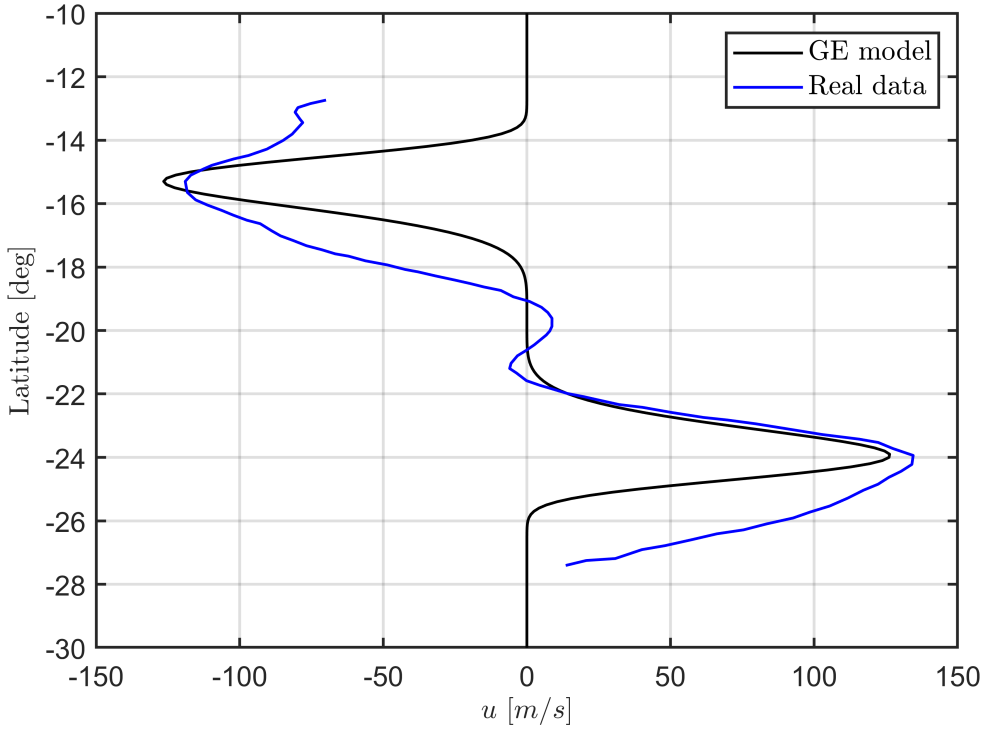
We can construct a plot to compare the results, which can be seen in Figure 3.1 for both components of velocity. It is very important to recall though, that the experimental data extracted from [20] is presented in a slightly different reference system. This is the so-called S_{III} system and the main difference is that the longitude that they use (θ') has the following relation with the longitude used in this thesis:

$$\theta' = -\theta \tag{3.12}$$

This change is important if we want to compare the results and it has been taken into account for the elaboration of Figure 3.1. Taking into account that plenty assumptions have been done for the theoretical model the results are reasonably accurate, but a simulation has to be performed and see the final state obtained.



(a) Meridional component of the velocity.



(b) Zonal component of the velocity.

Figure 3.1: Comparative plot of the velocity profile of the geostrophic balance model here developed and the experimental data extracted from [20].

4 — Hyperviscosity

After running the code with a very fine mesh some instabilities have appeared as can be seen in Section 6.2. Some models that simulate atmospheric dynamics are prone to these types of problems. For a model called *EPIC*, which is similar to the SW but with isentropic coordinates, a solution proposed to overcome them is to add a mathematical artifact to the equations to be able to filter the unstable structures that appear in energy and enstrophy conservative schemes [21]. As proposed by [21], [22] the term to be added is one similar to the diffusion of Navier-Stokes equations but with higher order spatial derivatives. The complete equation of conservation of mass can be seen in Eq. (4.1).

$$\frac{\partial \eta}{\partial t} + \nabla \cdot (h\vec{u}) = \sum_{n=1}^{n_{max}} (-1)^{n-1} \nu_{2n} \nabla^{2n} \eta \tag{4.1}$$

The new momentum equations are

$$\frac{\partial \vec{u}}{\partial t} + (\vec{u} \cdot \nabla)\, \vec{u} - f\left(\vec{u} \times \vec{k}\right) = -g\nabla\eta + \sum_{n=1}^{n_{max}} (-1)^{n-1} \nu_{2n} \nabla^{2n} \vec{u} \tag{4.2}$$

The complication arises from the fact that the laplacian operator of a vector in curvilinear coordinates is not as simple as one might expect. In other words, since in ellipsoidal coordinates $\nabla^2\vec{u} \neq \left(\nabla^2 u,\ \nabla^2 v\right)$, there will be extra terms. Only a way to compute the laplacian is needed since the higher order terms will be found by a repeated application of this operator. Obtaining the expression for the laplacian of a vector in ellipsoidal coordinates is not an easy task and, therefore, the strategy presented in [21] will be used. Then, the expression to compute the laplacian of the velocity is

$$\nabla^2\vec{u} = \left(-\frac{1}{r_M}\frac{\partial \zeta}{\partial \varphi} + \frac{1}{r_Z}\frac{\partial D}{\partial \theta},\ -\frac{1}{r_Z}\frac{\partial \zeta}{\partial \theta} + \frac{1}{r_M}\frac{\partial D}{\partial \varphi}\right) \tag{4.3}$$

where the terms ζ and D are the relative vorticity and the horizontal divergence, respectively, and are given by

$$\zeta = \frac{1}{r_Z r_M}\left(\frac{\partial}{\partial \theta}(r_M v) - \frac{\partial}{\partial \varphi}(r_Z u)\right) \tag{4.4a}$$

$$D = \frac{1}{r_Z r_M}\left(\frac{\partial}{\partial \theta}(r_M u) - \frac{\partial}{\partial \varphi}(r_Z v)\right) \tag{4.4b}$$

The reason to use this notation is because due to the staggered nature of the grid, these quantities can be computed in a natural way. The term ζ will be computed in the q-grid, whereas the divergence

D in a centered field (i.e. in the h-grid). Then, attending to its staggering, the discrete expressions are

$$\zeta_{i,j} = \frac{1}{r^q_{Z_{i,j}} r^q_{M_{i,j}}} \left(\frac{r^v_{M_{i+1,j}} v_{i+1,j} - r^v_{M_{i,j}} v_{i,j}}{\Delta\theta} - \frac{r^u_{Z_{i,j+1}} u_{i,j+1} - r^u_{Z_{i,j}} u_{i,j}}{\Delta\varphi} \right) \tag{4.5a}$$

$$D_{i,j} = \frac{1}{r^h_{Z_{i,j}} r^h_{M_{i,j}}} \left(\frac{r^u_{M_{i,j}} u_{i,j} - r^u_{M_{i-1,j}} u_{i-1,j}}{\Delta\theta} - \frac{r^v_{Z_{i,j}} v_{i,j} - r^v_{Z_{i,j-1}} v_{i,j-1}}{\Delta\varphi} \right) \tag{4.5b}$$

where the superscripts u, v, h, q denote the position in which the meridional and zonal radii should be evaluated. Finally, the discrete expression of the laplacian can be found and it is displayed in Eq. (4.6).

$$\nabla^2 \vec{u}\big|_x = -\frac{1}{r^u_{M_{i,j}}} \left(\frac{\zeta_{i,j} - \zeta_{i,j-1}}{\Delta\varphi} \right) + \frac{1}{r^u_{Z_{i,j}}} \left(\frac{D_{i+1,j} - D_{i,j}}{\Delta\theta} \right) \tag{4.6a}$$

$$\nabla^2 \vec{u}\big|_v = -\frac{1}{r^v_{Z_{i,j}}} \left(\frac{\zeta_{i,j} - \zeta_{i-1,j}}{\Delta\theta} \right) + \frac{1}{r^v_{M_{i,j}}} \left(\frac{D_{i,j+1} - D_{i,j}}{\Delta\varphi} \right) \tag{4.6b}$$

For the continuity equation, the laplacian can be computed using the expression of Eq. (A.15), since its expression is not indicated in [21]. Applying this definition to the h field we obtain

$$\nabla^2 h = \frac{1}{r_Z r_M} \left(\frac{\partial \psi}{\partial \theta} + \frac{\partial \xi}{\partial \varphi} \right) \tag{4.7}$$

with the auxiliary terms

$$\psi = \frac{r_M}{r_Z} \frac{\partial h}{\partial \theta} \tag{4.8a}$$

$$\xi = \frac{r_Z}{r_M} \frac{\partial h}{\partial \varphi} \tag{4.8b}$$

The discrete versions of those terms are

$$\psi_{i,j} = \frac{r^u_{M_{i,j}}}{r^u_{Z_{i,j}}} \left(\frac{h_{i+1,j} - h_{i,j}}{\Delta\theta} \right) \tag{4.9a}$$

$$\xi_{i,j} = \frac{r^v_{Z_{i,j}}}{r^v_{M_{i,j}}} \left(\frac{h_{i,j+1} - h_{i,j}}{\Delta\varphi} \right) \tag{4.9b}$$

which lead to the the discrete laplacian of h shown in Eq. (4.10).

$$\nabla^2 h = \frac{1}{r^h_{M_{i,j}} r^h_{Z_{i,j}}} \left(\frac{\psi_{i,j} - \psi_{i-1,j}}{\Delta\theta} + \frac{\xi_{i,j} - \xi_{i,j-1}}{\Delta\varphi} \right) \tag{4.10}$$

The hyperviscosity acts as a filter since it is a dissipative term, but the interesting property that can be used is that the higher the order of n the more selective the filter is. This means, that for $n = 1$, the resulting term is the common molecular viscosity which dissipates at all scales, but for atmospheric applications it is merely irrelevant for the large domains being solved. But, for $n > 1$ the energy is dissipated a more selective scales. Therefore, the hyperviscosity will be used because it can dissipate high frequency oscillations that may appear in the simulation. Then, if this high frequency oscillations can be filtered, they will not be able to grow and blow up the entire simulation.

Another important concept is the region of stability of the coefficients ν_{2n}, since there will be an additional requirement apart from the CFL condition to ensure the stability. The procedure to be done

is the Von Neumann stability criterion. This analysis is based on ensuring that the error does not grow in time and is different for each PDE and it depends on the integration scheme used. Since this procedure has already been done in [21], [22], their result will be explained yet not derived.

For the third order Adams-Bashforth, the stability analysis leads to the conclusion that the hyperviscosity coefficients should be

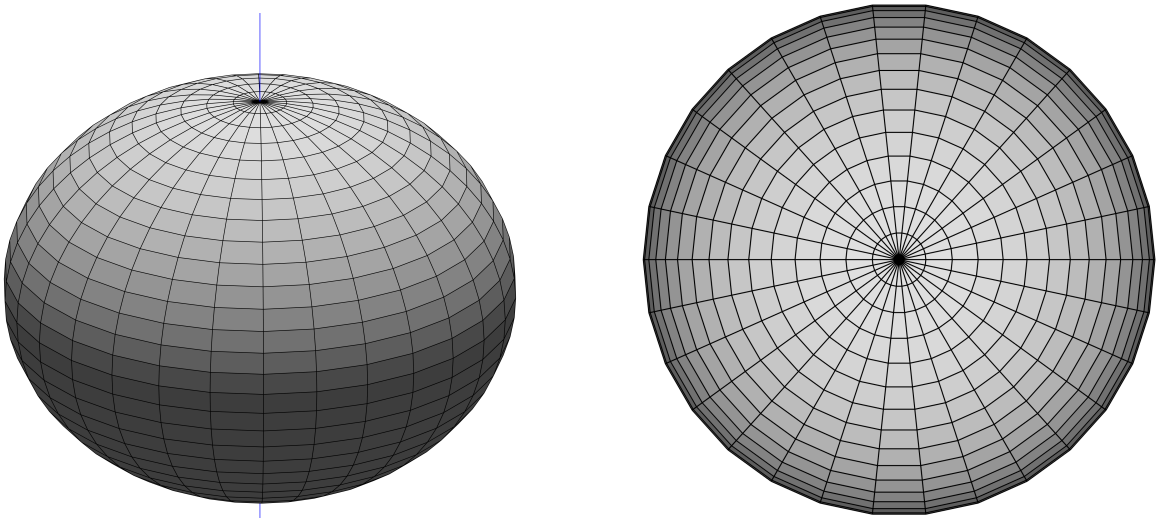
$$0 < \nu_{2n} \leq \frac{3/22}{(3n-2)(1+\delta^2)^n} \frac{(\Delta x)^{2n}}{\Delta t} \tag{4.11}$$

where $\delta = \Delta x/\Delta y$ represents the ratio between the two spatial distances. The authors also comment the addition of an additional safety factor due to the inaccuracy and incorrect assumptions in this derivation (i.e. assuming a finite difference scheme, based on a Cartesian grid, ...) which they set equal to 2. This will be taken into consideration at the time to choose the values for the hyperviscosity coefficients in the simulations.

The main coefficients that are used are for $n = 2$ and $n = 3$, but not higher. Then, only up to the third order hyperviscosity will be implemented. The results of a simulation using this type of filter are displayed in Section 6.2.

5 — Simulation of polar regions

In this chapter the Shallow Worlds code will be adapted to be able to perform simulations at poles. The main reason to do this special treatment is due to the the actual mesh being used. It is based on dividing the angle between the limits of the domain by the number of points, which leads to equal increments of longitude and latitude but not equal distance between the different points. The problem with this splitting at both latitude and longitude is that the resulting mesh is singular at both poles. It is very clear from the schemes shown in Figure 5.1.



(a) General view.

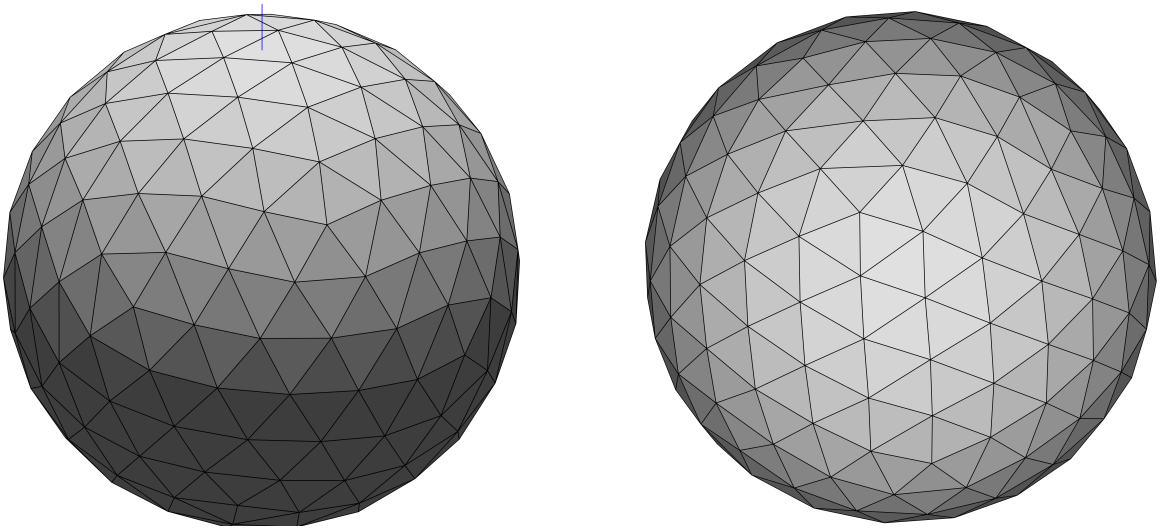
(b) Top view of the north pole.

Figure 5.1: Schemes of the discretization of the ellipsoid. The blue line represents the rotation axis.

This problem justifies the necessity of a special treatment of the poles and many solutions are available. The most direct one would be to do a completely different mesh. One example is the usage of a triangular mesh such as the one shown in Figure 5.2. Another possible option is a cube sphere as seen in Figure 5.3, which uses quadrilateral elements instead -which, again, it is not singular at the poles-.

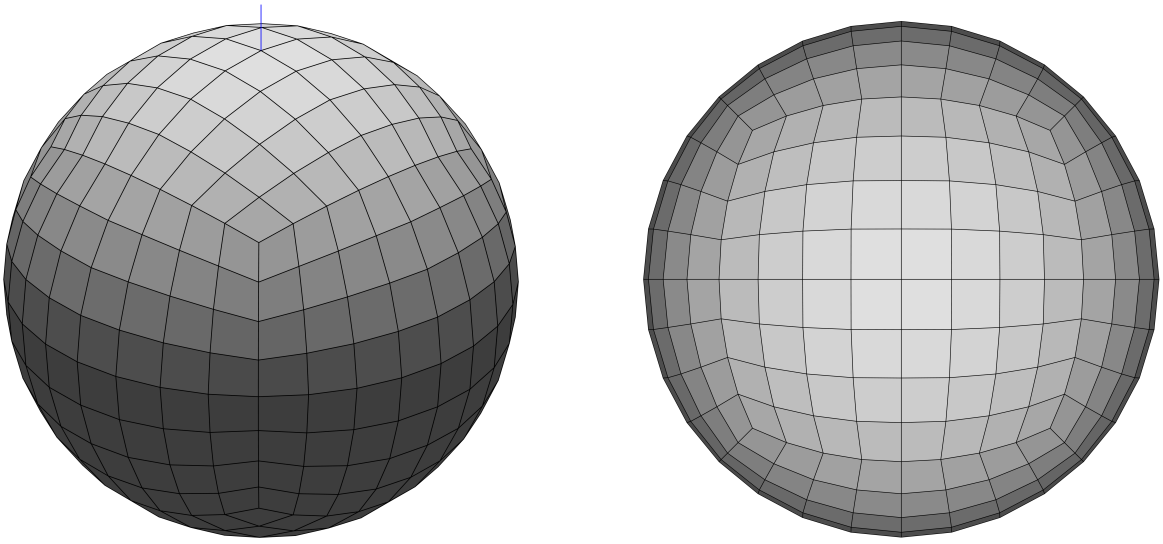
Both of these solutions imply major changes to the code, since a new discretization would be needed, a new computational algorithm -since it is element-dependent-, a new parallelization strategy and probably many other difficulties that could appear during its implementation and validation. Hence, it would be a great option to, maintaining the core of the code unchanged, adding the capability of performing polar simulations. Since the main problem is the mesh singularity, the proposed solution is to transform the rotation axis until its contained within the equatorial plane. Therefore, the mesh at the pole will be that of the equator and the equator will now contain the singularities.

As it is expected, the shape of the planet would be completely changed, and the curvature would strongly vary from that of the original pole. To overcome this problem, the simulation will be carried



(a) General view (b) Top view of the north pole

Figure 5.2: Schemes of the discretization of a sphere using a particle-based sample method. The blue line represents the rotation axis. It has been done using the routines from [23].



(a) General view (b) Top view of the north pole

Figure 5.3: Schemes of the discretization of a sphere as a cube sphere. The blue line represents the rotation axis. It has been done using the routines from [23].

out in a sphere with a radius equal to the radius of curvature of the original ellipsoid at the pole. In this way, we can ensure a good shape of the planet near the polar regions. To know this radius of curvature a simple derivation can be done. Taking into account that there is symmetry of revolution, only the 2D curvature of an ellipse is needed. First of all, consider the following ellipse parameterization:

$$\vec{r}(t) = (R_e \cos(t),\ R_p \sin(t)) \qquad \forall t \in [0, 2\pi] \tag{5.1}$$

Then, the curvature can be computed as

$$\kappa(t) = \frac{||\vec{r}\,'(t) \times \vec{r}\,''(t)||}{||\vec{r}\,'(t)||^3} \tag{5.2}$$

Evaluating this expression, the curvature is found as a function of the parameter t and equals

$$\kappa(t) = \frac{R_e R_p}{\left[R_e^2 \sin^2(t) + R_p \cos^2(t)\right]^{3/2}} \tag{5.3}$$

The radius of curvature that we are interested in is the one of the pole, which means $t = \pi/2$. Then,

$$\rho(\pi/2) = \frac{1}{\kappa(\pi/2)} = \frac{R_e^2}{R_p} \tag{5.4}$$

This is the first change needed, but not the only one, since there will be two latitudes now, the true one which remember that will be close to 90° and the one that will actually be using the code that is close to 0°. Therefore, a transformation of those latitudes should be done to be able to compute the true Coriolis parameter and gravity. This last one, by the way, will be computed on the real ellipsoidal planet, and then used in the modified spherical one. The other problem is that the latitude in this case should vary radially and in this latitude transformation this aspect should also be taken into account.

Calling the true latitude φ_t and the one that uses the code just φ, then the relation between them is

$$\varphi_t = p\left(\frac{\pi}{2} - \Xi(\theta, \varphi, \theta_0, \varphi_0)\right) \tag{5.5}$$

being

$$p = \begin{cases} 1 & \text{at the north pole} \\ -1 & \text{at the south pole} \end{cases} \tag{5.6}$$

and

$$\Xi(\theta, \varphi, \theta_0, \varphi_0) = \arccos\left[\sin(\varphi)\sin(\varphi_0) + \cos(\varphi)\cos(\varphi_0)\cos(\theta - \theta_0)\right] \tag{5.7}$$

This last term is just describing the angle between two points in a sphere as stated in [24] and where θ_0 and φ_0 will denote the longitude and latitude of the new pole. If the pole is centered in the domain (which is a reasonable disposition) these quantities are just $\varphi_0 = 0.5(\varphi_{min} + \varphi_{max})$ and $\theta_0 = 0.5(\theta_{min} + \theta_{max})$. Again, another assumption that is reasonable, is to center the new pole in the origin of both variables. Hence, in all simulations here done, $\varphi_0 = \theta_0 = 0$. Finally, using Eq. (5.5) the values of the Coriolis parameter and the gravity can be computed.

After coding this, it is very important to make sure that both the Coriolis parameter and the gravity have been correctly computed and are the real ones in spite of the transformation. Therefore,

in Figure 5.4 a comparative plot has been drawn, where the latitude of the Shallow Worlds code is the true one (φ_t) just for comparative reasons. There is no appreciable error in this implementation, which indicates a correct physical parameters so far.

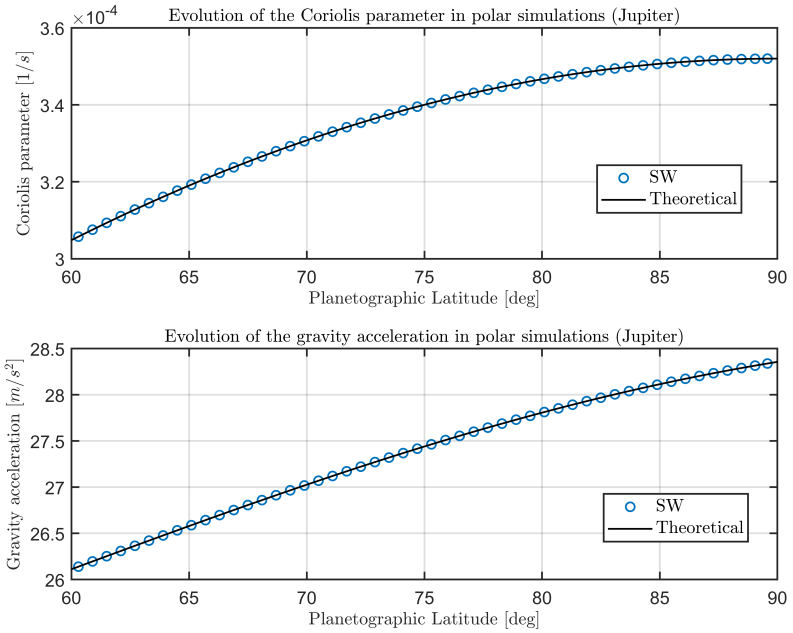


Figure 5.4: Comparative plot of the Coriolis parameter and gravity distributions between the theoretical and the one implemented in Shallow Worlds.

Another change that is needed is the introduction of the zonal winds. The main reason is that in a giant planet like Jupiter or Saturn, the winds move longitudinally (i.e. east to west or west to east, but not from north to south or vice versa) but when the winds near the pole region are introduced, due to the transformation, they have to move around the central point of the simulation which means that is no longer longitudinally, but has two components. Then, the winds must be decomposed in such a way that the signs are correctly introduced into the simulation.

Calling the polar wind W_p^x and W_p^y for the x and y components, respectively, and the original wind profile U, they can be computed as shown in Eq. (5.8) and being p the one defined in Eq. (5.6).

$$\begin{aligned} W_p^x &= -p\cos(\alpha)U(\varphi_t) \\ W_p^y &= p\sin(\alpha)U(\varphi_t) \end{aligned} \tag{5.8}$$

The definition of the angle α is

$$\alpha = \operatorname{sgn}(\theta - \theta_0)\arccos\left(\frac{\varphi - \varphi_0}{\sqrt{(\varphi-\varphi_0)^2 + (\theta-\theta_0)^2}}\right) \tag{5.9}$$

Finally, the boundary conditions must be treated. Normally, the domains are full-periodic or in the form of a channel. In this case, though, both of this options are not feasible since they have no reasonable explanation. The most physically accurate thing would be that the structures that leave the domain disappear. To achieve that, a so-called sponge layer is needed. As defined in [25], the sponge layer is a part of the domain in which an artificial extra term is added to the equations to dampen everything that arrives. With it, the domain will theoretically be full periodic but in practice, anything

that tries to leave it will be dissipated before actually getting to the boundary. The scheme is the one presented in Figure 5.5.

The term to be added for a general transport equation for variable ϕ is

$$\frac{\partial \phi}{\partial t} + F(\phi) = -\sigma(\phi - \phi^{ext}) \tag{5.10}$$

Having added this artifact, everything that approaches the limits of the domain can be dissipated with a controlled intensity that varies with σ. This parameter is important provided that if it's too large it can reflect waves and if it's too small it may not dissipate the outgoing fluid structures. Reference [25] gives an estimation for optimal values of σ but in practice they have turned out to be not the optimal for the simulations performed and hence this value has been chosen by trial and error.

With all this information, we should have a working code to simulate the poles and an example of it is presented in Section 6.4.

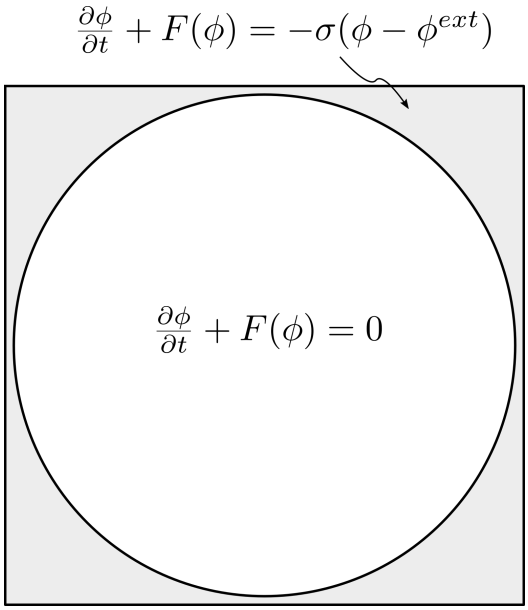


Figure 5.5: Scheme of the sponge used in polar simulations

6 — Simulation results

In this chapter some simulation will be performed and the results presented. Some of these simulations will be performed with the *MATLAB*® code and other with the parallel Shallow Worlds code.

The first of the simulations will serve to check if all the terms that have been implemented work correctly together by watching a water drop followed by gravity waves. The rest will be centered in planetary atmospheres. Mainly, they will be devoted to trying to reproduce the Great Red Spot of Jupiter and the north pole of the same planet.

6.1 Gravity Waves

The first simulation that can be done to test the code is the simulation of gravity waves (GW). This type of waves are the ones driven by the effect of gravity and for this model it can be seen through the surface elevation. This type of situation is reminiscent of the waves produced by rain drops when they impact a surface of water. Therefore, the idea is to introduce a perturbation in the surface elevation and let the system evolve and see if this waves are produced or not.

The perturbation will be introduced via initial conditions in a surface elevation of a Gaussian form as seen in Eq. (6.1). The boundary conditions are full periodic in both axis.

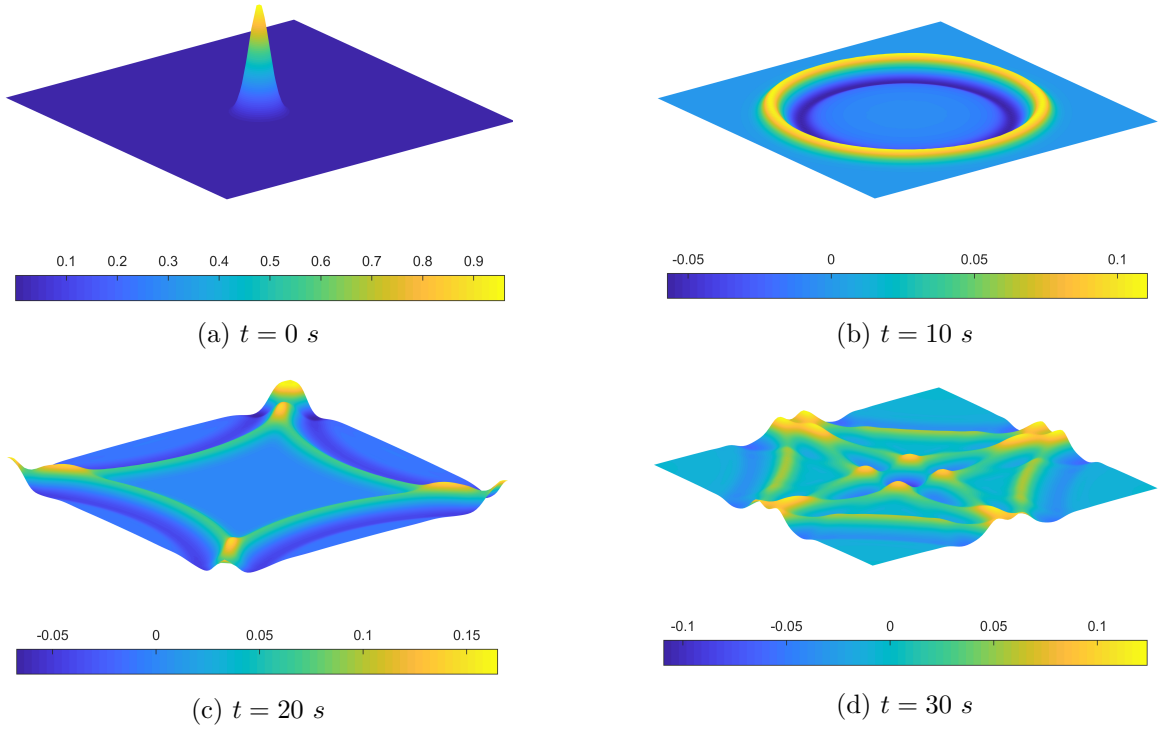
$$\eta(x,y) = A \exp\left\{ -\left(\frac{x-x_0}{\sigma_x}\right)^2 + \left(\frac{y-y_0}{\sigma_y}\right)^2 \right\} \tag{6.1}$$

The exact parameters for this simulation will be:

- General parameters
  - $\Omega = [0, 200] \times [0, 200]$
  - $\Delta t = 0.05\ s$
  - $N_x = N_y = 100$
  - $g = 9.81\ m/s^2$
  - $D = 5\ m$
- Peturbation parameters
  - $A = 1\ m$
  - $x_0 = 100\ m$
  - $y_0 = 100\ m$
  - $\sigma_x = 4\ m^{-1}$

– $\sigma_y = 4\ m^{-1}$

The results for this first simulation are shown in Figure 6.1 at different times.



(a) $t = 0\ s$ (b) $t = 10\ s$

(c) $t = 20\ s$ (d) $t = 30\ s$

Figure 6.1: Surface elevation (η) of the GW simulation at times indicated.

As it can be seen, the results are neat and seem fine, because they imitate correctly the gravity waves propagating through the periodic domain. A final check that can be done it to compute the phase speed of the waves since in the shallow water model, the predicted gravity waves have a phase speed of $c = \sqrt{gD}$. This can be shown from the linearized SW equations which are

$$\frac{\partial u}{\partial t} = -g\frac{\partial \eta}{\partial x} \tag{6.2a}$$

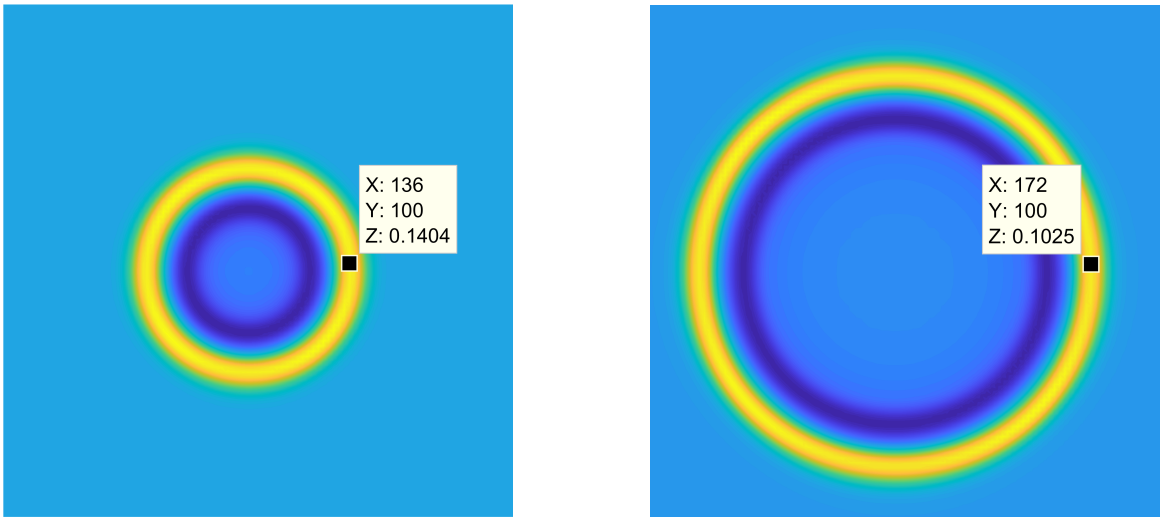
$$\frac{\partial v}{\partial t} = -g\frac{\partial \eta}{\partial y} \tag{6.2b}$$

$$\frac{\partial \eta}{\partial t} + D\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) = 0 \tag{6.2c}$$

and that have been obtained neglecting the Coriolis effect, the advection of the velocity and assuming that $h \approx D$ which comes from $\mathcal{O}(D) \gg \mathcal{O}(\eta)$. Then, taking the partial derivative of Eq. (6.2)a with respect to x, of Eq. (6.2)b with respect to y and introducing the result into Eq. (6.2)c after having taken the derivative with respect to t the result is

$$\frac{\partial^2 \eta}{\partial t^2} - gD\nabla^2\eta = 0 \tag{6.3}$$

The form of this PDE is that of a wave equation with a travel speed of $c^2 = gD$. To compare this predicted phase speed to the one obtained in the simulation we can estimate the distance traveled by the wave front between two times. We can use the frames shown in Figure 6.2, where a data point has been placed in the most accurate position of the wave front possible with this level of resolution.



(a) $t = 5\ s$ (b) $t = 10\ s$

Figure 6.2: Top view of the surface elevation (η) of the GW simulation at times indicated

Then, the phase speed of the gravity wave can be estimated as

$$c_n = \frac{\Delta x}{\Delta t} = \frac{172 - 136}{10 - 5} = 7.2\ m/s$$

while the theoretical value is

$$c_t = \sqrt{gD} = \sqrt{9.81 \cdot 5} = 7.0\ m/s$$

Then, taking into account that the theoretical value comes from a linearized set of equations and that the resolution doesn't permit a more accurate estimation of the numerical wave speed and the result obtained indicates a relative error of $< 3\%$, we can conclude that the code works correctly.

6.2 High resolution simulations of Jupiter's Great Red Spot

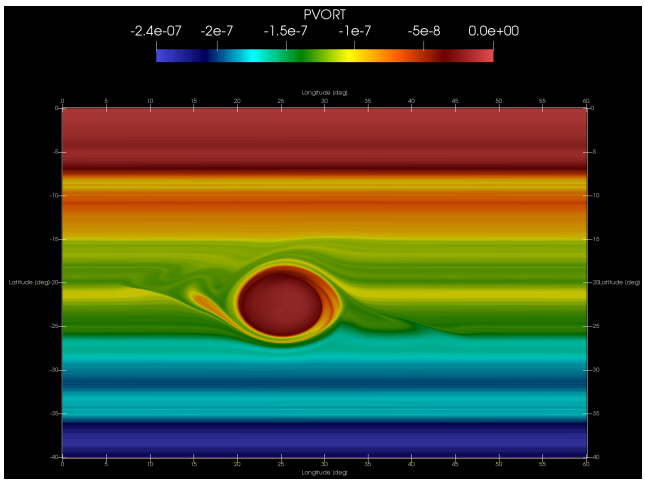
For this type of simulations an other code will be used. For now, all has been done using a personal code written in *MATLAB*®, but to be able to perform simulations at high resolutions to have a good level of detail, a parallel code is needed. In this case, the Shallow Worlds code, developed by the director and co-director of this thesis, as well as the former student Arnau Prat, will be used. For a thorough explanation of how this code works, the reader is referred to [3].

A first simulation that can be performed to test the code is the Great Red Spot (GRS) of Jupiter, since it is a very well-known system. The results obtained for a very fine mesh and a small time step are shown in Figure 6.3. As it can be seen, at first, everything seems fine, but after the day 35 an instability appears and grows rapidly until the whole simulation blows up.

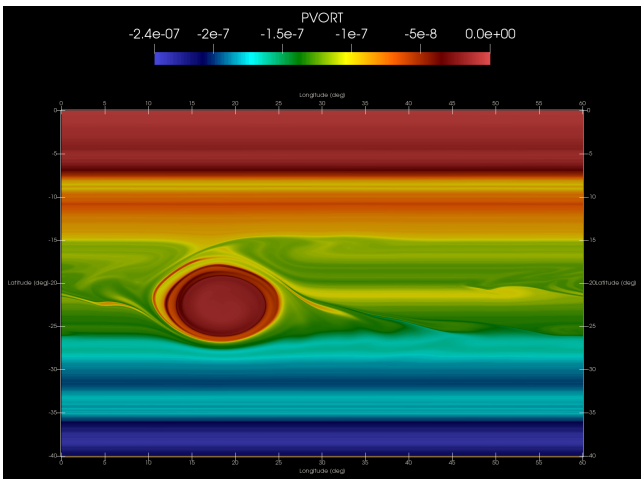
A way to see the effects of this instability is to see what happened to the energy of the system. The main two contributions are the kinetic energy (KE) and the so-called available potential energy (APE) which are defined, according to [26], as

$$KE = \frac{1}{2} \int_\Omega gh(u^2 + v^2) d\Omega \tag{6.4}$$

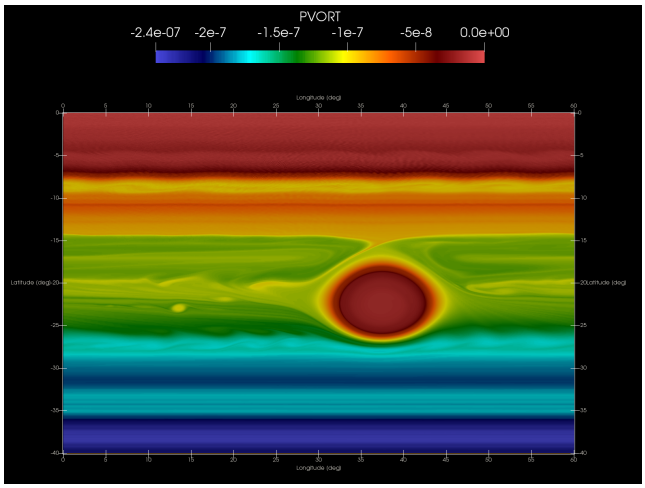
$$APE = \frac{1}{2} \int_\Omega \left[(gh)^2 - \langle gh \rangle^2 \right] d\Omega \tag{6.5}$$



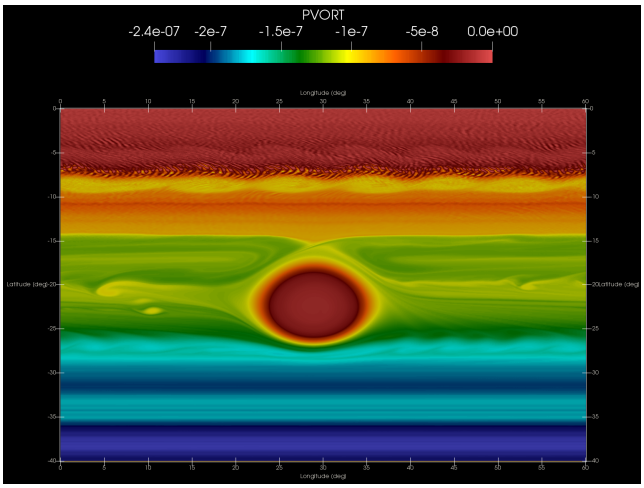
(a) $t = 5\ days$



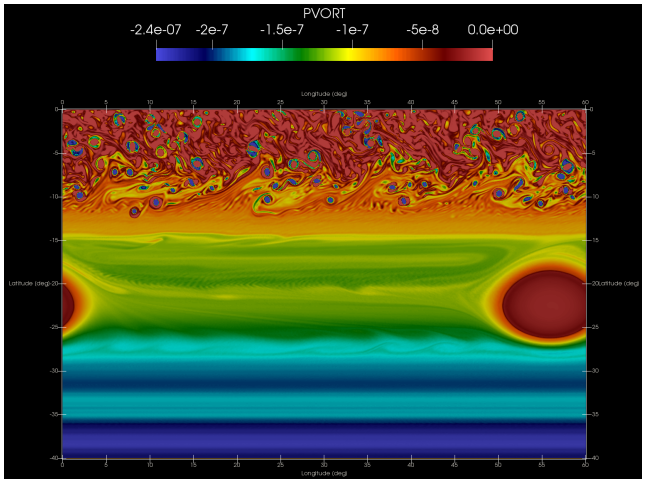
(b) $t = 10\ days$



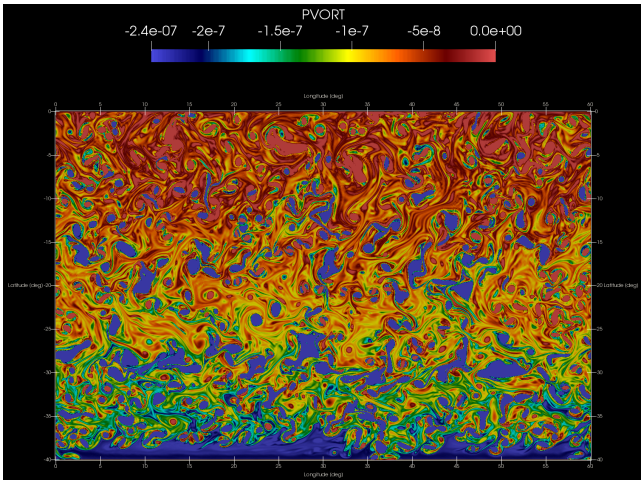
(c) $t = 35\ days$



(d) $t = 40\ s$



(e) $t = 60\ days$



(f) $t = 130\ days$

Figure 6.3: Simulation results of the GRS using Shallow Worlds with a mesh of 1200×800 and $\Delta t = 2\ s$.

where $\langle gh \rangle$ denotes the average specific potential energy of the background. Adding the routines to perform these computations to Shallow Worlds, the results for this particular simulation are shown in Figure 6.4. It can be concluded that after day 47, the energy changes its trend of decaying and starts growing rapidly. This is a counter-intuitive result since in Section 2.6 it was said that the TVD schemes tend to overcome the spurious oscillations by a mechanism of energy dissipation.

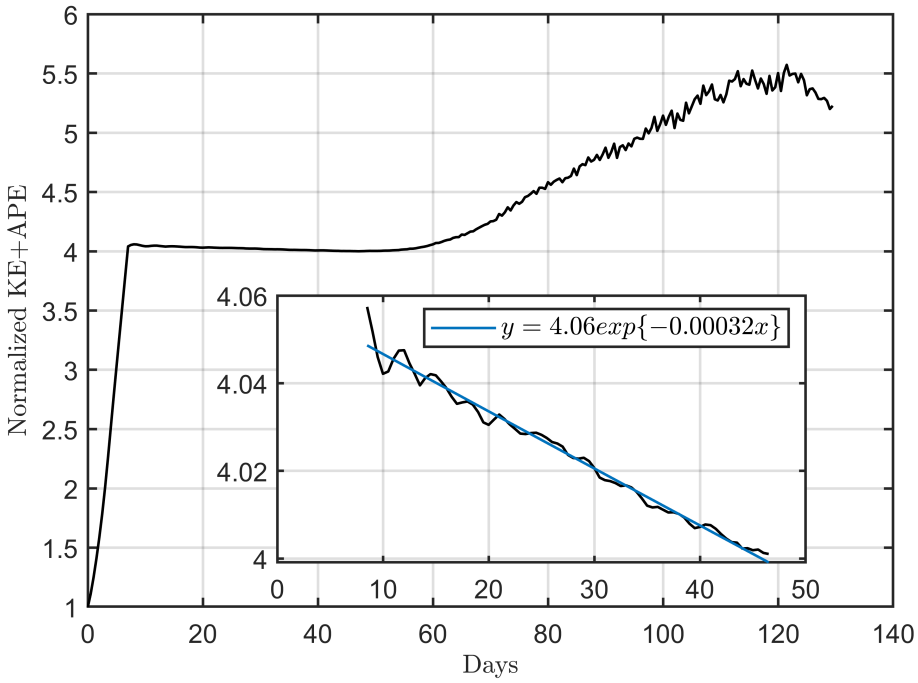


Figure 6.4: Plot of the energy normalized with the initial energy of the system for the GRS simulation. The inner plot represents a zoom in for the energy decay zone (i.e. between days 9 and 47) and an exponential fitting.

After a little bit of research in the literature, this type of behaviour can be associated to the so-called Hollingsworth instability [27]. In that article, Hollingsworth *et al.* analyzed a rapidly growing instability that generated small-scale noise and that was able to produce an increase of the kinetic energy. The only difference, though, is that in that paper they used a finite difference energy and enstrophy conservation scheme, which is different to the one being used in our case, and they related this type of instability to the treatment of the advection which leaded to a non-cancelling term in a vector invariant form of the SW equations. This form is based in splitting up the advective term as

$$(\vec{u} \cdot \nabla)\vec{u} = \nabla K + \zeta \vec{k} \times \vec{u} \tag{6.6}$$

being K the specific kinetic energy and ζ the z component of the vorticity. Therefore, after all, it seems that Shallow Worlds shouldn't be affected by this instability.

On the other hand, though, reference [28] showed that this instability can appear in other schemes, for example based on finite volumes, and even when the equations do not use the vector-invariant form and use the common advection as presented in this thesis. This opens the possibility to being affected by this instability in more situations than those initially observed by Hollingsworth and it is suspected to be the case for what appears in Figure 6.3.

To ensure that the problem is inherent to the equations and the algorithm used and not a problem of the implementation or from the parallel routines, the code of Prof. Dr. García-Melendo [2], [18] was recovered and a similar simulation was performed. The results presented this instability as well indicating that the system is unstable.

At this stage, the first test that needs to be performed in order to overcome this instability is to filter the high frequency waves that appear in the simulation using the hyperviscosity as explained in Chapter 4. Using the hyperviscosity coefficients recommended by [21], which are $\nu_2 = \nu_4 = 0$ and $\nu_6/\nu_6^{max} = 0.7$, the results are displayed in Figure 6.5.

From it, one can see that the instability still appears, although later, and therefore is not a feasible solution if long simulations are to be performed. The energy evolution for this case is displayed in Figure 6.6. The energy decay has the same tendency, which indicates a good behaviour of the hyperviscosity, since it is not destroying too much energy in order to filter the different fields that affects. On the other hand, though, at the end of the simulation the tendency is to increase the energy again, so as seen before, it does not prevent the instability from appearing.

At this stage there is no more information in the literature on how to overcome this problem. Therefore, if a solution is to be encountered it will be on our own. After revising all the steps of the resolution algorithm, one can see that the most problematic term, as expected, is the advective one. At first, in Section 2.6 it was said that the flux limiter used is the Superbee due to the facts already exposed, but there is nothing that prevents us from trying other schemes as well and see if they present the same instability.

This reasoning, though, is not obvious, since the instability has been reported for other algorithms and there is no evidence in the literature of a bad malfunction of the Superbee scheme. Actually, is the other way around, since is one of the less dissipative from the available schemes and the literature exposes lots of good reasons to use it. Despite all that, since it is a small change in terms of coding, it can be a good test to make. One of the flux limiters that shows a very good behaviour apart from the Superbee is the MUSCL scheme [11]. Therefore, the following test will be to repeat the simulation with no hyperviscosity at all, but with the MUSCL scheme.

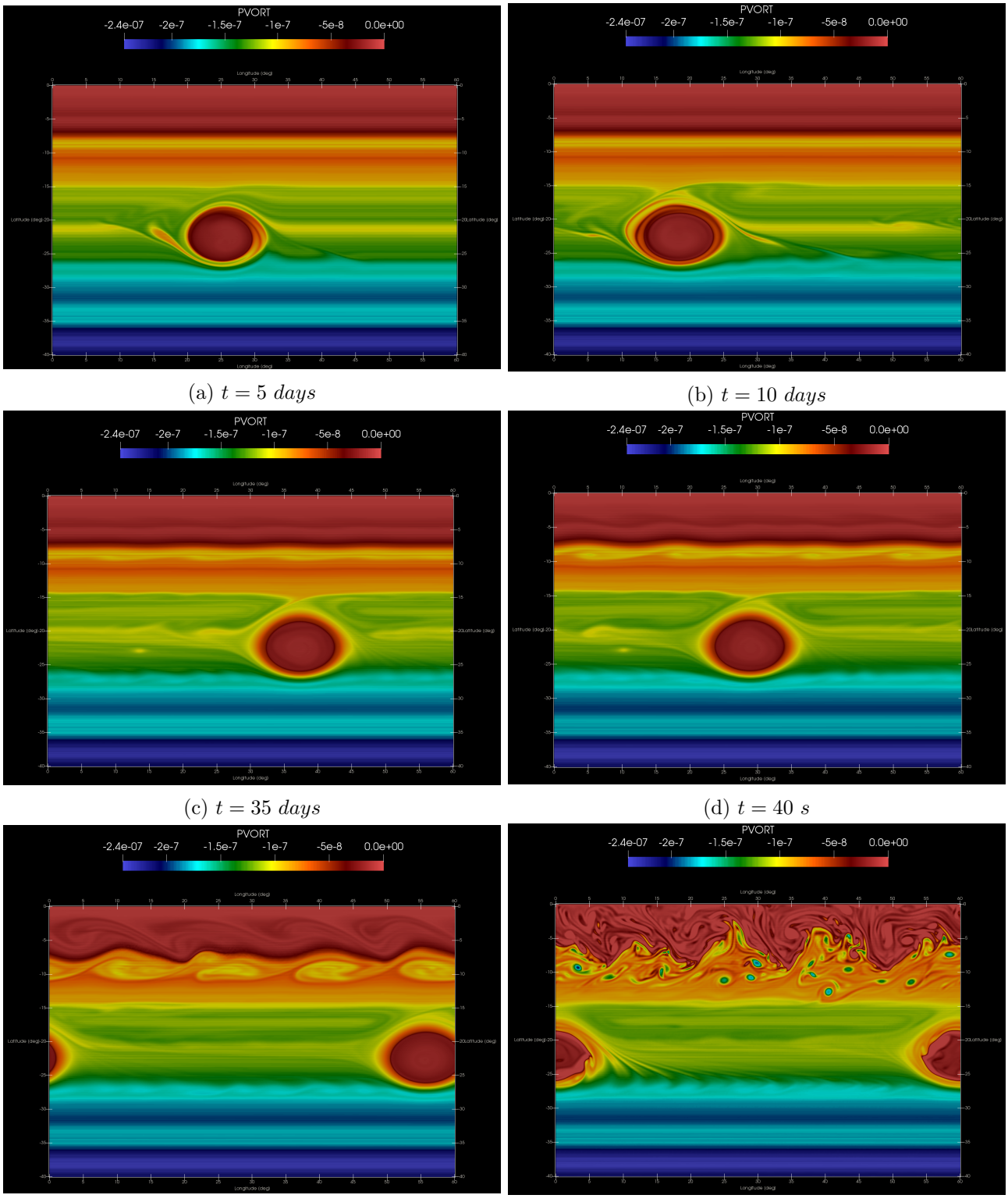
The results are presented in Figure 6.8 and they are incredibly good compared to the previous ones, since there is no trace of any instability. Looking at the energy evolution of the system, which are presented in Figure 6.7, seem completely normal, but with a higher energy decay constant, which was expected to the fact the MUSCL scheme is more dissipative than the Superbee.

Finally, to try to see if the problem comes from the fact that the Superbee is less dissipative another simulation with the MUSCL scheme will be done. It will be achieved with a finer mesh and a smaller Δt. Setting 4 times more control volumes and reducing the time step to $1s$ the results are in Figure 6.9. The simulation is shorter because this one needs a lot of computational resources and it was not feasible to extend the simulation for the same amount of time. Regarding the energy of the system, it can be seen in Figure 6.10. The energy decay is very similar to the one with the Superbee scheme if we take into account that since the simulation is shorter, less points are available to fit an exponential function. With the presented results it can be assumed that using this scheme overcomes the problem of the numerical instability but a thorough study should be done regarding the usage of TVD schemes.

6.3 Simulation of the GRS via geostrophic balance

In this section, the simulation of the GRS will be repeated but with the method and data presented in Chapter 3. The simulation performed is shown in Figure 6.11. Also, a comparative plot on the velocity profile can be done to see if the steady-state reproduces the data or the expected velocity distribution imposed via initial conditions. This plot is shown in Figure 6.12.

From it, we can extract some conclusions. The first and most obvious one is that they are not satisfactory enough. The shape of the GRS and the velocity profiles are wrong and very different from the analytical distribution derived. Another important factor is that it does not shorten the time to establish and reach the steady state and hence, is not an improvement compared to the previous



(a) $t = 5\ days$

(b) $t = 10\ days$

(c) $t = 35\ days$

(d) $t = 40\ s$

(e) $t = 60\ days$

(f) $t = 130\ days$

Figure 6.5: Simulation results of the GRS using Shallow Worlds with a mesh of 1200×800 and $\Delta t = 2\ s$ using hyperviscosity.

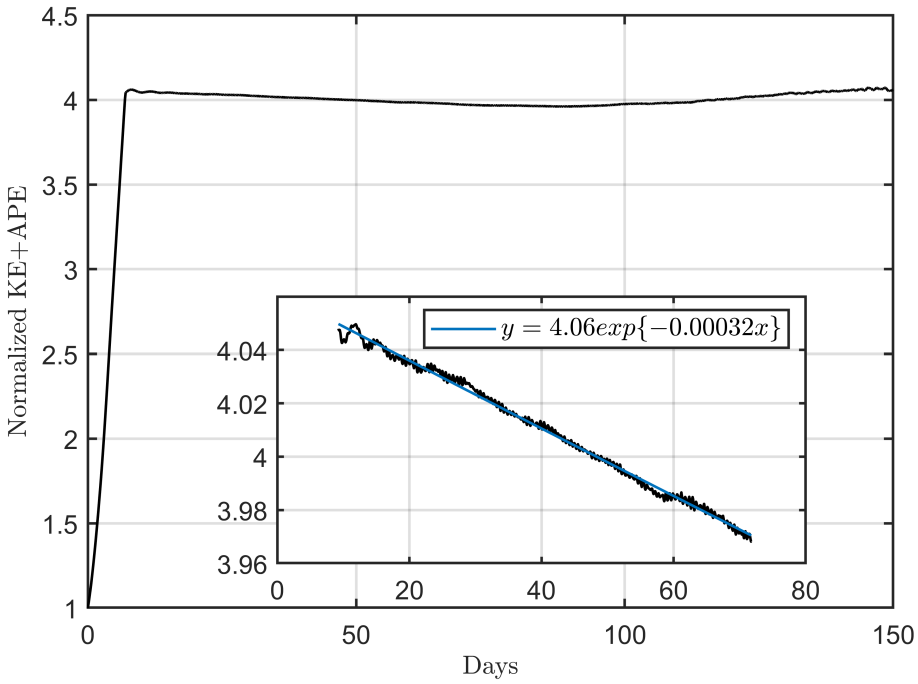


Figure 6.6: Plot of the energy normalized with the initial energy of the system for the GRS simulation with hyperviscosity. The inner plot represents a zoom in for the energy decay zone (i.e. between days 9 and 75) and an exponential fitting.

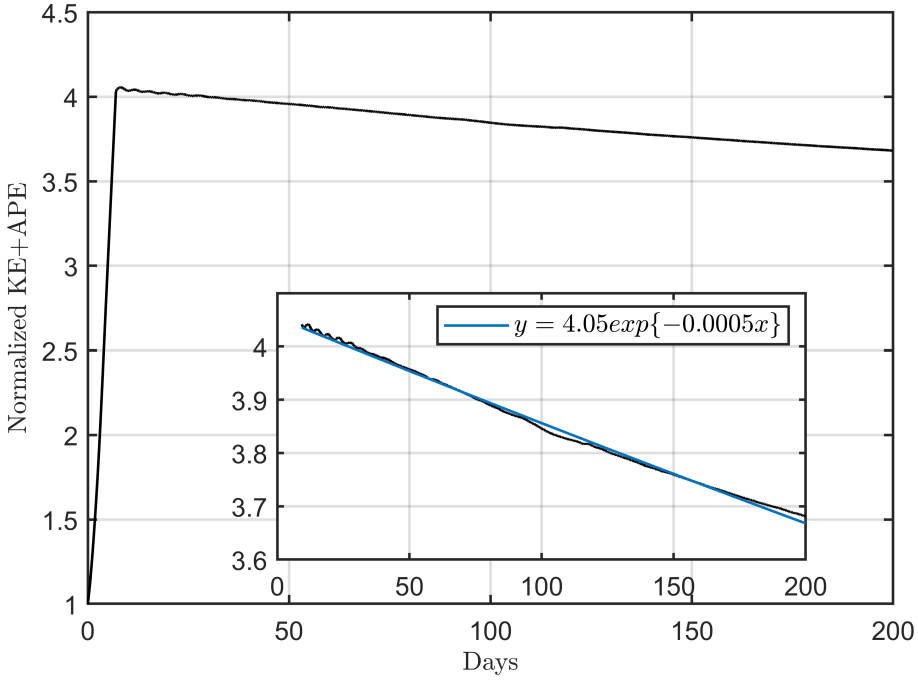
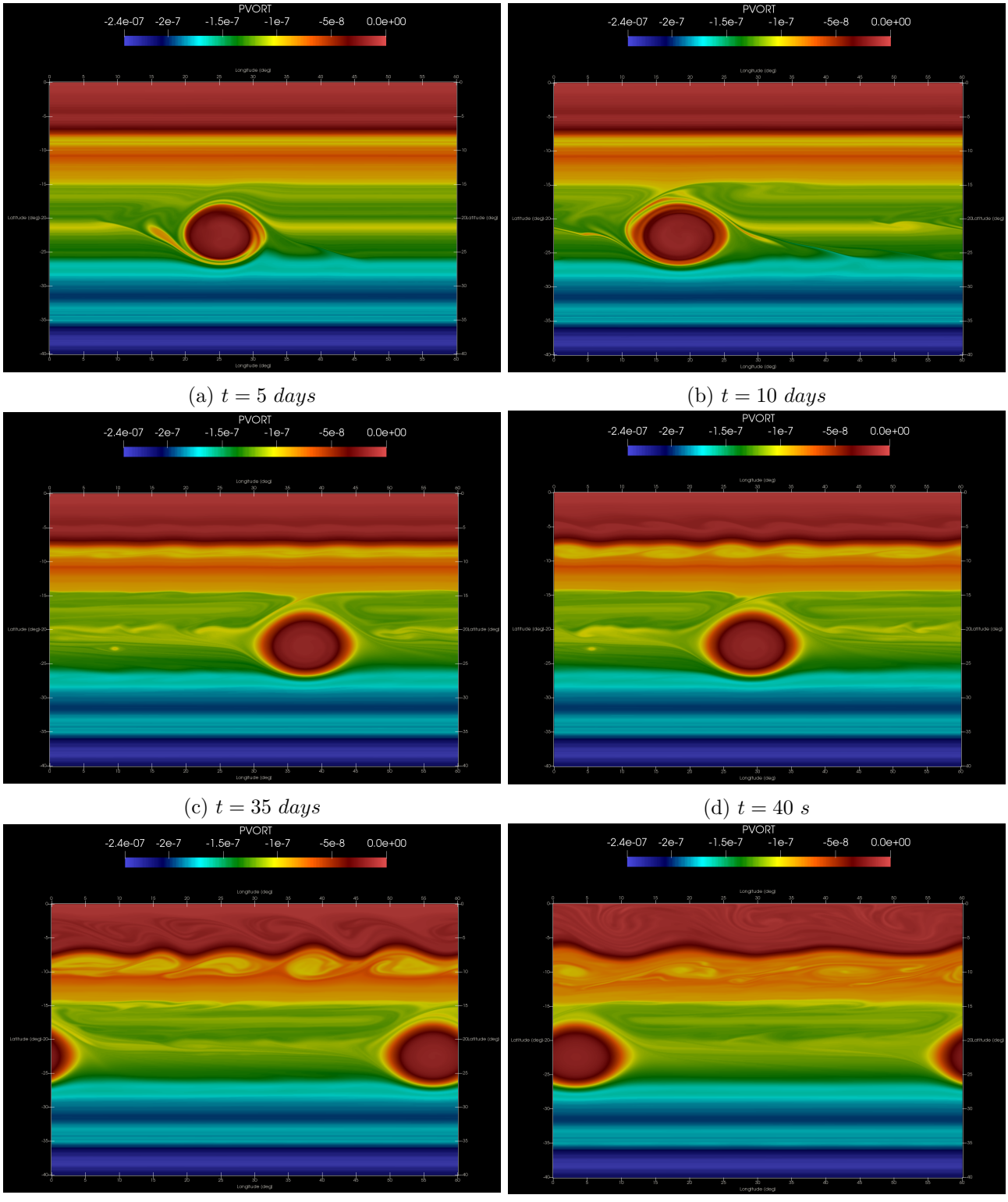


Figure 6.7: Plot of the energy normalized with the initial energy of the system for the GRS simulation with MUSCL scheme. The inner plot represents a zoom in for the energy decay zone (i.e. between days 9 and 130) and an exponential fitting.



(a) $t = 5\ days$

(b) $t = 10\ days$

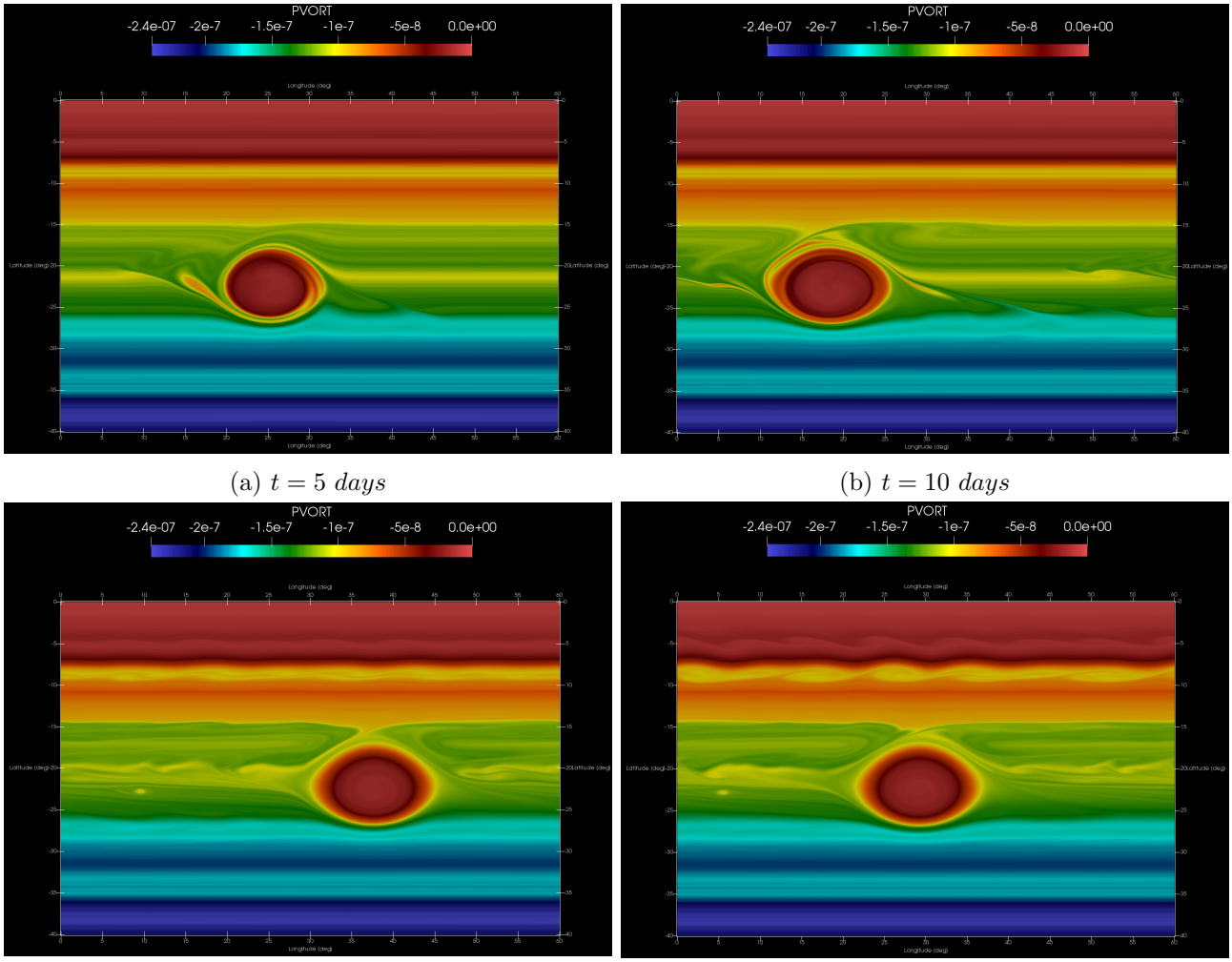
(c) $t = 35\ days$

(d) $t = 40\ s$

(e) $t = 60\ days$

(f) $t = 130\ days$

Figure 6.8: Simulation results of the GRS using Shallow Worlds with a mesh of 1200×800 and $\Delta t = 2\ s$ using the MUSCL scheme.



(a) $t = 5\ days$

(b) $t = 10\ days$

(c) $t = 35\ days$

(d) $t = 40\ s$

Figure 6.9: Simulation results of the GRS using Shallow Worlds with a mesh of 2400×1600 and $\Delta t = 1\ s$ using the MUSCL scheme.

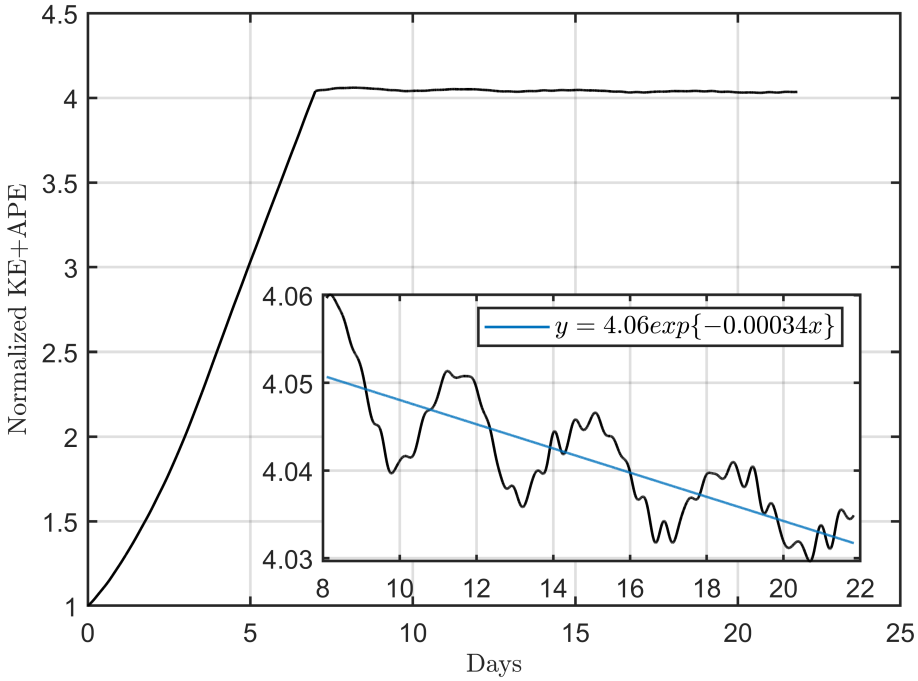


Figure 6.10: Plot of the energy normalized with the initial energy of the system for the GRS simulation with MUSCL scheme for the higher resolution and smaller Δt. The inner plot represents a zoom in for the energy decay zone (i.e. between days 9 and 22) and an exponential fitting.

method. It is worth noting, though that the resolution of this simulation is considerably worse and the time step larger. Therefore, this makes this simulation way much more dissipative which may be the responsible of the flattening of the GRS.

This result indicates, then, that this method can still be used in some occasions because it's still easier to introduce a perturbation by initial conditions than continuously injecting energy (mainly for coding aspects, the original option is harder to code than setting a velocity profile and surface elevation), but overall has not met the requirements that we expected and have turned out to be a small failure to improve the parameters of the resulting vortex.

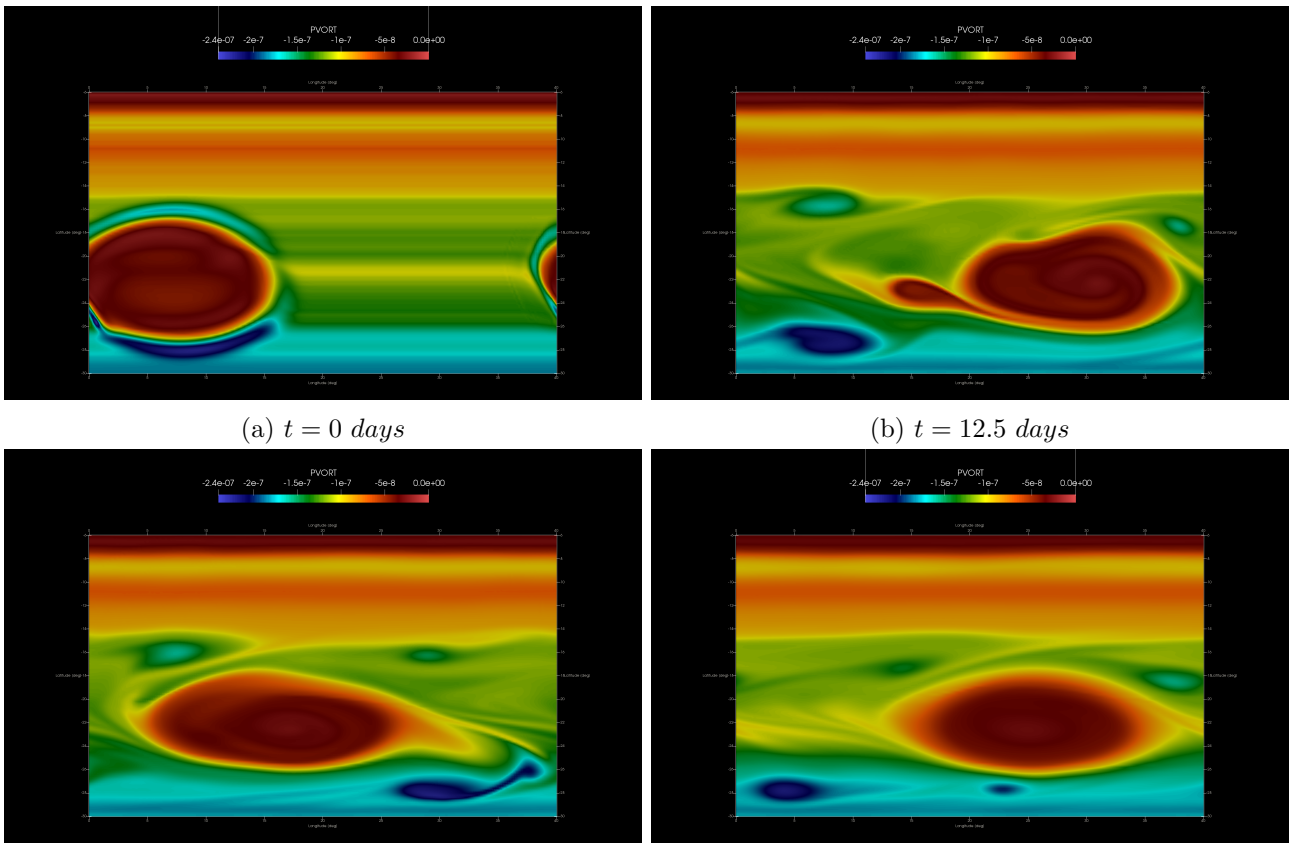
6.4 Simulation of Jupiter's north pole

In this section the results of the simulation of Jupiter's north pole will be presented. For this high resolution simulations only the winds have been introduced and like in the actual planet, they are unstable and start creating vortices and a turbulent motion.

Some images of the chaotic motion of Jupiter's north pole are displayed in Figure 6.13. It is also interesting to note how in the actual pole a system of cyclones is formed. Then, if the physics are correct we should observe a movement of part of the chaotic structures towards the pole. The simulation carried out is shown in Figure 6.14 and it can be seen that the general dynamics are correctly reproduced since the chaotic and turbulent movement is present, with the appearance of cyclones and anticyclones that tend to move towards the center of the domain, which corresponds to the pole.

A comment worth being done is regarding the sponge layer here implemented. The value for the absorbing parameter σ should be changed since it does affect the simulation introducing some shear instabilities that in further simulations should be avoided.

Many more simulations should be carried out in order to see how the parameters influence the



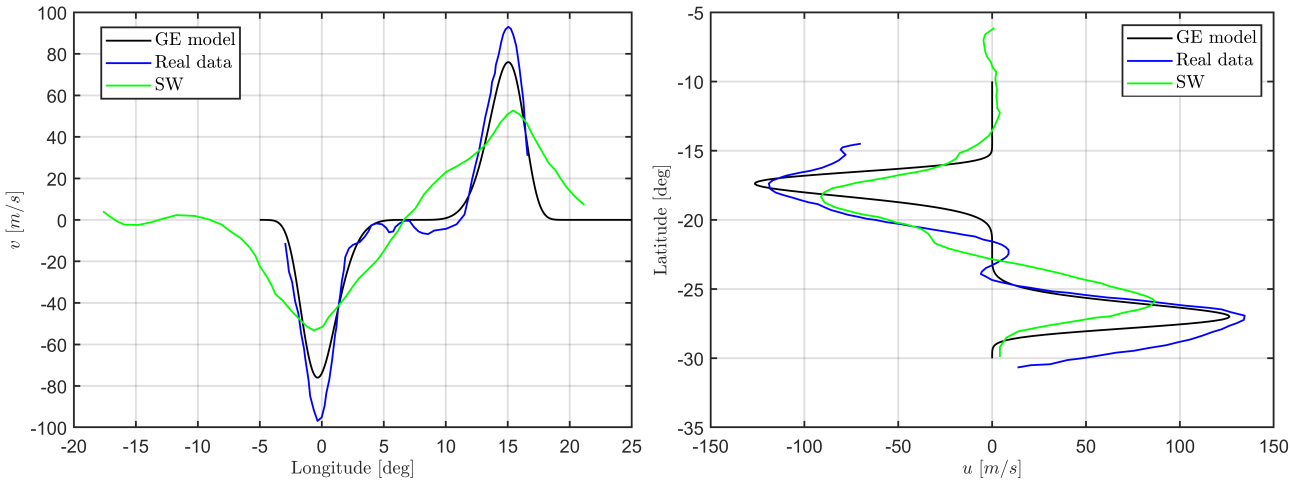
(a) $t = 0$ *days*

(b) $t = 12.5$ *days*

(c) $t = 20$ *days*

(d) $t = 40$ *days*

Figure 6.11: Simulation results of the GRS using Shallow Worlds and the geostrophic equilibrium with a mesh of 200×120 and $\Delta t = 5$ *s*.

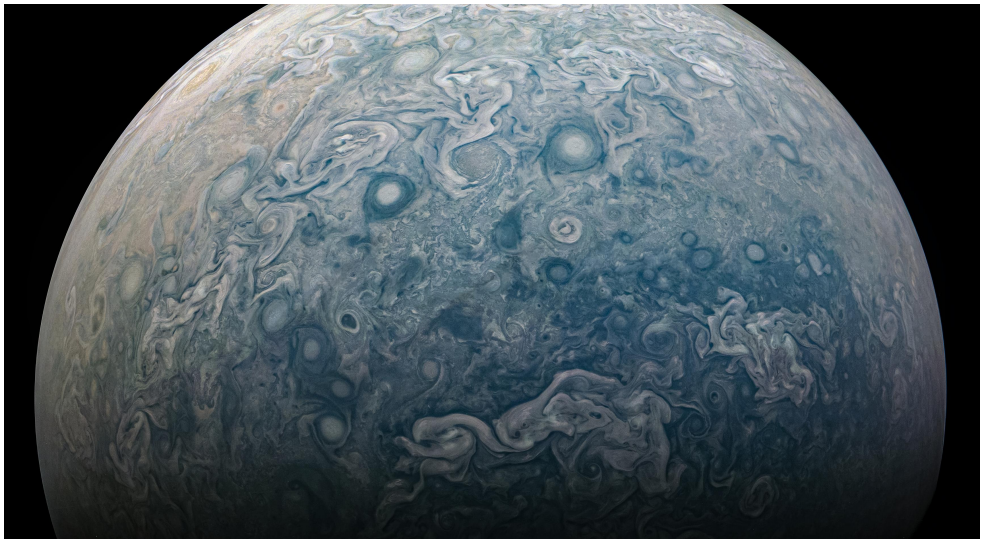


(a) Distribution of the meridional wind

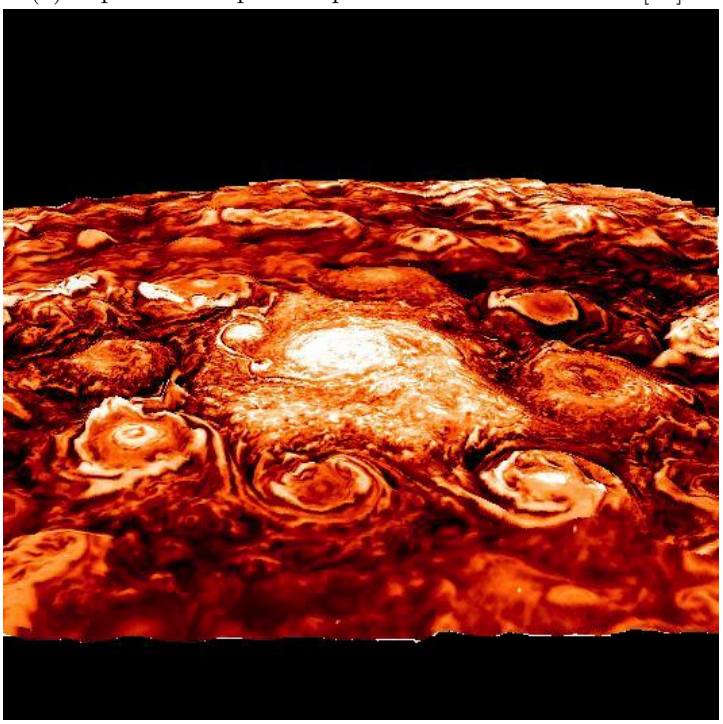
(b) Distribution of the zonal wind

Figure 6.12: Velocity distribution of the GRS using Shallow Worlds and the geostrophic equilibrium with a mesh of 200×120 and $\Delta t = 5$ *s*.

output of the simulation and it would be interesting to try to reproduce more accurately the formation of cyclones and anticyclones as shown in [26]. Reproducing the examples of this reference would be a good start to test the accuracy of this proposed solution, although an improvement introduced with respect to them is the computation on an actual sphere instead of using Cartesian coordinates.

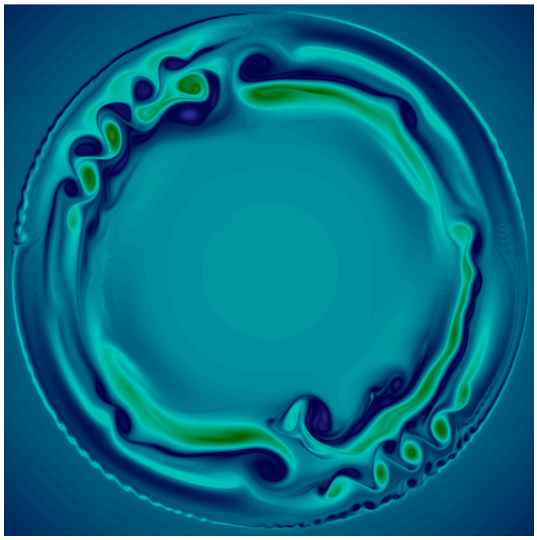


(a) Jupiter north pole deep motion. Extracted from [29].

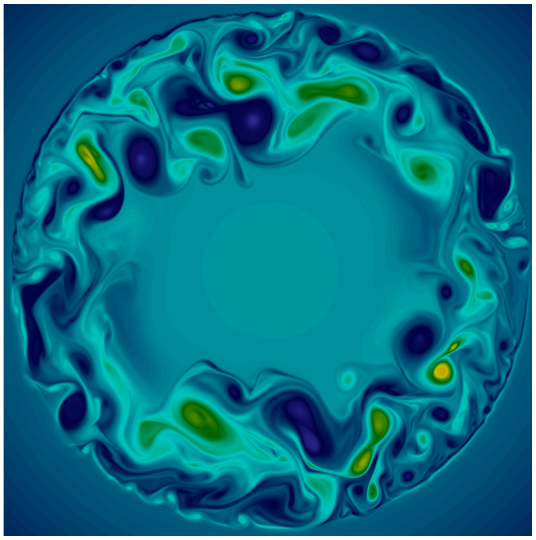


(b) Jupiter north pole cyclones. Extracted from [30].

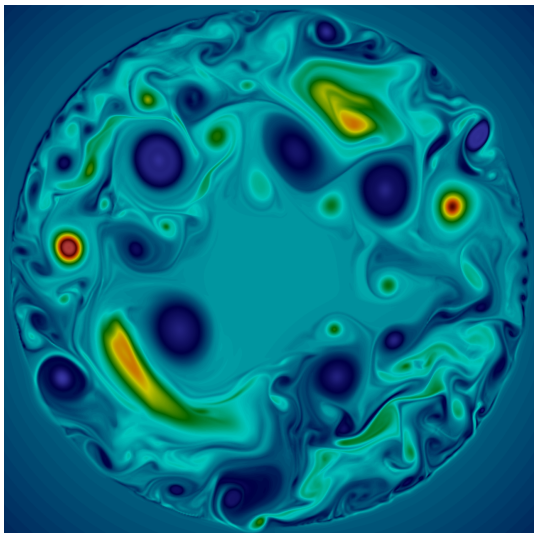
Figure 6.13: Available images from Juno mission of Jupiter's north pole



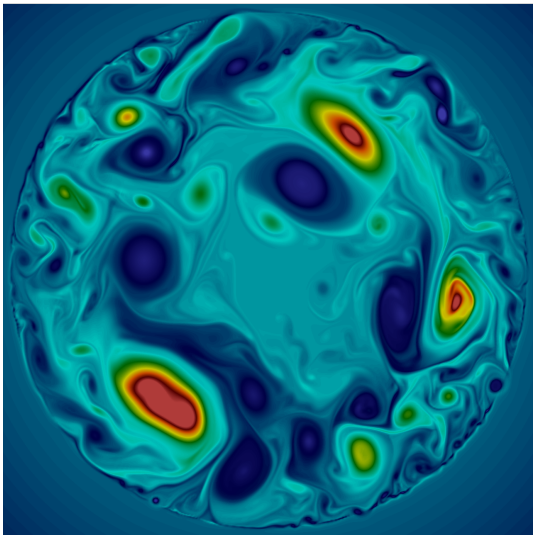
(a) $t = 10$ *days*



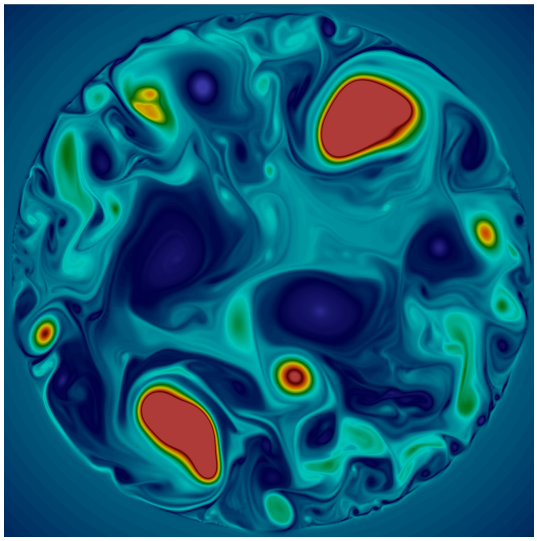
(b) $t = 20$ *days*



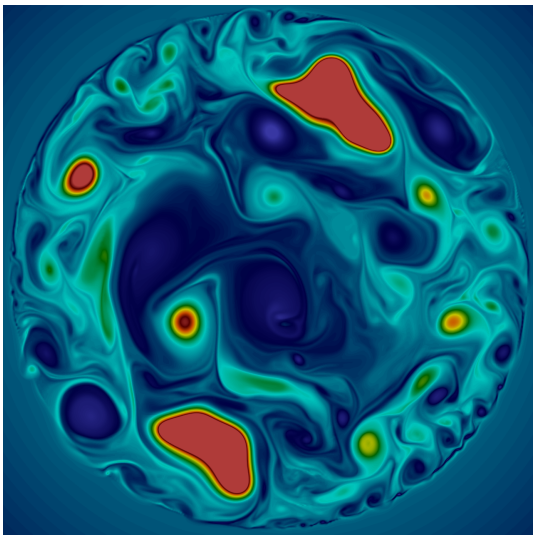
(c) $t = 50$ *days*



(d) $t = 80$ *days*



(e) $t = 160$ *days*



(f) $t = 195$ *days*

Figure 6.14: Jupiter north pole simulation with a mesh resolution of 600×600 and $\Delta t = 2$ s.

Part II

3D Atmospheric Model

7 — Numerical resolution of INS equations

To start with the 3D model, the first code that will be programmed is the one meant to solve the incompressible Navier Stokes (INS) equations with a parallel computer. This will determine the general outline of the code that will be used later in the atmospheric model. The set of equations for this type of fluid are presented in Eq. (7.1).

$$\frac{\partial \vec{u}}{\partial t} + (\vec{u} \cdot \nabla)\, \vec{u} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \vec{u} \tag{7.1a}$$

$$\nabla \cdot \vec{u} = 0 \tag{7.1b}$$

To solve them, the fractional step method will be used as explained in [31]. The first thing that will be done is rewrite the momentum equations from Eq. (7.1) as

$$\frac{\partial \vec{u}}{\partial t} = \vec{R}(\vec{u}) - \frac{1}{\rho}\nabla p \tag{7.2}$$

where

$$\vec{R}(\vec{u}) = -(\vec{u} \cdot \nabla)\, \vec{u} + \nu \nabla^2 \vec{u} \tag{7.3}$$

Now, the way to proceed is to integrate in time Eq. (7.2) in the following fashion:

- The term $\vec{R}(\vec{u})$ will be evaluated in an explicit way using a second order Adams-Bashforth.
- The pressure term ∇p will be integrated implicitly.
- The change in time of the velocity vector, $\partial \vec{u}/\partial t$ will be approximated by a simple first order finite difference scheme.

The result is then

$$\frac{\vec{u}^{\,n+1} - \vec{u}^{\,n}}{\Delta t} = \frac{3}{2}\vec{R}(\vec{u}^{\,n}) - \frac{1}{2}\vec{R}(\vec{u}^{\,n-1}) - \frac{1}{\rho}\nabla p^{n+1} \tag{7.4}$$

At this stage, it is interesting to rearrange the terms and distinguish the known magnitudes and the unknown. This separation would lead to

$$\vec{u}^{\,n+1} = \vec{u}^{\,p} - \nabla \tilde{p}^{\,n+1} \tag{7.5}$$

Where the new terms are defined as

$$\vec{u}^{\,p} := \vec{u}^{\,n} + \Delta t \left(\frac{3}{2}\vec{R}(\vec{u}^{\,n}) - \frac{1}{2}\vec{R}(\vec{u}^{\,n-1}) \right) \tag{7.6}$$

and

$$\tilde{p}^{\,n+1} := \frac{\Delta t}{\rho} p^{\,n+1} \tag{7.7}$$

The former of this quantities will be called the predictor velocity. Take into account, though, that it might not have a physical meaning and that it also can violate the incompressibility condition. The latter will be called pseudo-pressure. The relation displayed in Eq. (7.5) is supported by the Helmholtz-Hodge theorem.

Theorem (Helmholtz-Hodge Decomposition, [32]). *Let Ω be a region in space with smooth boundary $\partial\Omega$. Then, a vector field $\vec{\xi}$ on Ω can be uniquely decomposed in the form*

$$\vec{\xi} = \vec{\gamma} + \nabla D$$

where $\nabla \cdot \vec{\gamma} = 0$ and $\vec{\gamma} \cdot n = 0$ on $\partial\Omega$.

Since the flow is incompressible ($\nabla \cdot \vec{u}^{\,n+1} = 0$) the decomposition of $\vec{u}^{\,p}$ will be unique, and thus, will guarantee a solution. To proceed with the algorithm, the divergence in both sides of Eq. (7.5) must be taken, and thanks to the incompressibility condition, this reduces to

$$\nabla^2 \tilde{p}^{\,p+1} = \nabla \cdot \vec{u}^{\,p} \tag{7.8}$$

Since the right hand side of this equality is known, we can solve it for $\tilde{p}^{\,p+1}$. Then, once the pseudo-pressure is calculated, to recover the velocity field for the next time step will be as easy as to take the predictor velocity field and subtract from it the gradient of the pseudo-pressure. Therefore, the only thing that we need to know is how the convection and diffusion are computed in a discrete way and then how to solve the Poisson equation (Eq. (7.8)).

The mesh tat will be used for this case is the same as the one shown in Figure 2.1. The reason is similar than before: if this staggering is not done, the coupling of the pressure and velocity requires a change to the algorithm here presented. It is worth noting, though, that for this case the centered variable will be the pressure (and any other transported scalar field) and also that the third dimension is not drawn in that scheme but would follow the same reasoning for the staggering of the z component of the velocity.

7.1 Discrete evaluation of the convection

Recall that the vector $\vec{R}(\vec{u}^{\,n})$ contains the contribution of the convection and, hence, this one must be found in a discrete way. To start with, the convection term will be rewritten in a conservative form. This is

$$(\vec{u} \cdot \nabla)\vec{u} = \nabla \cdot (\vec{u} \otimes \vec{u}) - \vec{u}(\nabla \cdot \vec{u}) = \nabla \cdot (\vec{u} \otimes \vec{u}) \tag{7.9}$$

where it has been taken into account that $\nabla \cdot \vec{u} = 0$. Then, we first integrate the convection over a control volume like

$$\int_\Omega \nabla \cdot (\vec{u} \otimes \vec{u}) d\Omega \tag{7.10}$$

According to [33], the Gauss divergence theorem applied to a general m^{th} order tensor T is defined as

$$\int_V \frac{\partial T_{ij\cdots k\cdots m}}{\partial x_k} dV = \oint_{\partial V} T_{ij\cdots k\cdots m} \hat{n}_k dS \tag{7.11}$$

where Einstein's notation convention has been assumed. Then, applying this definition to our second order tensor, that will be defined as $T := \vec{u} \otimes \vec{u}$, we obtain

$$\int_V \frac{\partial T_{ij}}{\partial x_j} dV = \oint_{\partial V} T_{ij} \hat{n}_j dS \tag{7.12}$$

Then, evaluating discretely this expression in terms of fluxes through the faces of the CV we get that the three components of the convection vector $\vec{C}$ are

$$\begin{aligned} C^x &= u_e F_e^x - u_w F_w^x + u_n F_n^x - u_s F_s^x + u_t F_t^x - u_b F_b^x \\ C^y &= v_e F_e^y - v_w F_w^y + v_n F_n^y - v_s F_s^y + v_t F_t^y - v_b F_b^y \\ C^z &= w_e F_e^z - w_w F_w^z + w_n F_n^z - w_s F_s^z + w_t F_t^z - w_b F_b^z \end{aligned} \tag{7.13}$$

Recall, though, that these expressions are integrated over the whole control volume, and in the PDE it appears in the non integrated form. Then, the way to recover the original term the following assumption has to be done

$$\nabla \cdot (\vec{u} \otimes \vec{u}) \approx \frac{1}{\Omega} \int_\Omega \nabla \cdot (\vec{u} \otimes \vec{u}) d\Omega \tag{7.14}$$

The terms of the discrete convection vector can be evaluated taking into account the staggering. The fluxes will denote a velocity multiplied by the surface of the face denoted by the subscript. Finally, another aspect should be taken into account: the grid, although structured, will be nonuniform in the most general case. Then, the way in which the coordinates of the grid might be evaluated is through a rank 5 tensor that will contain the coordinates needed. This will be $\Delta_{ijklm} \quad \forall i,j,k,l,m \in [1,N_x] \times [1,N_y] \times [1,N_z] \times [1,3] \times [0,1]$. The first three indices are the spatial ones, the fourth indicates the axis and the fifth can either be 0 or 1 indicating if the position is the staggered one or the centered, respectively. Then, the flux terms can be computed as displayed in Eq. (7.15) whilst the surfaces involved in Eq. (7.16).

$$\begin{aligned}
F_e^x &= \frac{u_{i,j,k} S_e^x + u_{i+1,j,k} S_e^x}{2}; & F_e^y &= \frac{u_{i,j,k} S_e^{y-} + u_{i,j+1,k} S_e^{y+}}{2}; & F_e^z &= \frac{u_{i,j,k} S_e^{z-} + u_{i,j,k+1} S_e^{z+}}{2} \\
F_w^x &= \frac{u_{i,j,k} S_w^x + u_{i-1,j,k} S_w^x}{2}; & F_w^y &= \frac{u_{i-1,j,k} S_w^{y-} + u_{i-1,j+1,k} S_w^{y+}}{2}; & F_w^z &= \frac{u_{i-1,j,k} S_w^{z-} + u_{i-1,j,k+1} S_w^{z+}}{2} \\
F_n^x &= \frac{v_{i,j,k} S_n^{x-} + v_{i+1,j,k} S_n^{x+}}{2}; & F_n^y &= \frac{v_{i,j,k} S_n^y + v_{i,j+1,k} S_n^y}{2}; & F_n^z &= \frac{v_{i,j,k} S_n^{z-} + v_{i,j,k+1} S_n^{z+}}{2} \\
F_s^x &= \frac{v_{i,j-1,k} S_s^{x-} + v_{i+1,j-1,k} S_s^{x+}}{2}; & F_s^y &= \frac{v_{i,j,k} S_s^y + v_{i,j-1,k} S_s^y}{2}; & F_s^z &= \frac{v_{i,j-1,k} S_s^{z-} + v_{i,j-1,k+1} S_s^{z+}}{2} \\
F_t^x &= \frac{w_{i,j,k} S_t^{x-} + w_{i+1,j,k} S_t^{x+}}{2}; & F_t^y &= \frac{w_{i,j,k} S_t^{y-} + w_{i,j+1,k} S_t^{y+}}{2}; & F_t^z &= \frac{w_{i,j,k} S_t^z + w_{i,j,k+1} S_t^z}{2} \\
F_b^x &= \frac{w_{i,j,k-1} S_b^{x-} + w_{i+1,j,k-1} S_b^{x+}}{2}; & F_b^y &= \frac{w_{i,j,k-1} S_b^{y-} + w_{i,j+1,k-1} S_b^{y+}}{2}; & F_b^z &= \frac{w_{i,j,k} S_b^z + w_{i,j,k-1} S_b^z}{2}
\end{aligned} \tag{7.15}$$

$$
\begin{aligned}
S_e^x &= S_w^x = (\Delta_{i,j,k,2,0} - \Delta_{i,j-1,k,2,0})(\Delta_{i,j,k,3,0} - \Delta_{i,j,k-1,3,0}) \\
S_e^{y-} &= S_w^{y-} = (\Delta_{i,j,k,2,0} - \Delta_{i,j-1,k,2,0})(\Delta_{i,j,k,3,0} - \Delta_{i,j,k-1,3,0}) \\
S_e^{y+} &= S_w^{y+} = (\Delta_{i,j+1,k,2,0} - \Delta_{i,j,k,2,0})(\Delta_{i,j,k,3,0} - \Delta_{i,j,k-1,3,0}) \\
S_e^{z-} &= S_w^{z-} = (\Delta_{i,j,k,2,0} - \Delta_{i,j-1,k,2,0})(\Delta_{i,j,k,3,0} - \Delta_{i,j,k-1,3,0}) \\
S_e^{z+} &= S_w^{z+} = (\Delta_{i,j,k,2,0} - \Delta_{i,j-1,k,2,0})(\Delta_{i,j,k+1,3,0} - \Delta_{i,j,k,3,0}) \\
S_n^y &= S_s^y = (\Delta_{i,j,k,1,0} - \Delta_{i-1,j,k,1,0})(\Delta_{i,j,k,3,0} - \Delta_{i,j,k-1,3,0}) \\
S_n^{x-} &= S_s^{x-} = (\Delta_{i,j,k,1,0} - \Delta_{i-1,j,k,1,0})(\Delta_{i,j,k,3,0} - \Delta_{i,j,k-1,3,0}) \\
S_n^{x+} &= S_s^{x-} = (\Delta_{i+1,j,k,1,0} - \Delta_{i,j,k,1,0})(\Delta_{i,j,k,3,0} - \Delta_{i,j,k-1,3,0}) \\
S_n^{z-} &= S_s^{z-} = (\Delta_{i,j,k,1,0} - \Delta_{i-1,j,k,1,0})(\Delta_{i,j,k,3,0} - \Delta_{i,j,k-1,3,0}) \\
S_n^{z+} &= S_s^{z+} = (\Delta_{i,j,k,1,0} - \Delta_{i-1,j,k,1,0})(\Delta_{i,j,k+1,3,0} - \Delta_{i,j,k,3,0}) \\
S_t^z &= S_b^z = (\Delta_{i,j,k,1,0} - \Delta_{i-1,j,k,1,0})(\Delta_{i,j,k,2,0} - \Delta_{i,j-1,k,2,0}) \\
S_t^{x-} &= S_b^{x-} = (\Delta_{i,j,k,1,0} - \Delta_{i-1,j,k,1,0})(\Delta_{i,j,k,2,0} - \Delta_{i,j-1,k,2,0}) \\
S_t^{x+} &= S_b^{x-} = (\Delta_{i+1,j,k,1,0} - \Delta_{i-1,j,k,1,0})(\Delta_{i,j,k,2,0} - \Delta_{i,j-1,k,2,0}) \\
S_t^{y-} &= S_b^{y-} = (\Delta_{i,j,k,1,0} - \Delta_{i-1,j,k,1,0})(\Delta_{i,j,k,2,0} - \Delta_{i,j-1,k,2,0}) \\
S_t^{y+} &= S_b^{y+} = (\Delta_{i,j,k,1,0} - \Delta_{i-1,j,k,1,0})(\Delta_{i,j+1,k,2,0} - \Delta_{i,j,k,2,0})
\end{aligned}
\tag{7.16}
$$

Secondly, the transported velocities should be interpolated. Instead of using a TVD scheme, an spectroconsistent one will be used. This type of scheme ensures that the energy is conserved and hence, produces more realistic results. To do so, the procedure shown in [34], [35] will be explained and applied. Defining a general convective transport equation of the form

$$
\frac{\partial \phi}{\partial t} + u\frac{\partial \phi}{\partial x} = 0 \tag{7.17}
$$

then its discrete version can be written as

$$
\mathbf{H}\frac{d\boldsymbol{\phi}_h}{dt} + \mathbf{M}\boldsymbol{\phi}_h = 0 \tag{7.18}
$$

being $\mathbf{H}$ a positive-definite diagonal matrix with the volumes of each CV and $\mathbf{M}$ is the contribution of fluxes through control volumes. The discrete solution is a vector $\boldsymbol{\phi}_h$ with the values at the corresponding N nodes in which the domain has been discretized. The norm of the discrete kinetic energy evolution is defined by means of

$$
\frac{d}{dt}||\boldsymbol{\phi}_h||_h^2 = \frac{d}{dt}\left(\boldsymbol{\phi}_h^T\mathbf{H}\boldsymbol{\phi}_h\right) = -\boldsymbol{\phi}_h^T(\mathbf{M} + \mathbf{M}^T)\boldsymbol{\phi}_h \tag{7.19}
$$

If the energy is to be constant, then it is clear that the time derivative should be equal to zero. Hence, the condition that must be fulfilled is that

$$
\boldsymbol{\phi}_h^T(\mathbf{M} + \mathbf{M}^T)\boldsymbol{\phi}_h = 0 \quad \forall \boldsymbol{\phi}_h \in \mathbb{R}^N \tag{7.20}
$$

This condition is only fulfilled for any solution of the discrete problem if and only if, the matrix $\mathbf{M}$ is skew-symmetric. This condition implies that $\mathbf{M}^T = -\mathbf{M}$ and therefore Eq. (7.20) is satisfied. Then, a discretization that produces such a skew-symmetric convective matrix would be spectroconsistent. First of all, let's define the matrix $\mathbf{H}$. In a one dimensional case, this would be

$$
\mathbf{H}_{i,i} = \frac{1}{2}\left(x_{i+1} - x_{i-1}\right) \tag{7.21}
$$

If we define an approximation for the convection as

$$u \left.\frac{\partial \phi}{\partial x}\right|_{x_i} \approx u \frac{\phi_{i+1} - \phi_{i-1}}{x_{i+1} - x_{x-1}} \tag{7.22}$$

notice that can be rewritten as

$$u \left.\frac{\partial \phi}{\partial x}\right|_{x_i} \approx \left(\mathbf{H}^{-1}\mathbf{M}\boldsymbol{\phi}_h\right)_i \tag{7.23}$$

being the components of the $\mathbf{M}$ matrix defined by

$$\mathbf{M}_{i,i-1} = -\frac{1}{2}u \qquad \mathbf{M}_{i,i} = 0 \qquad \mathbf{M}_{i,i+1} = \frac{1}{2}u \tag{7.24}$$

The skew-symmetry property is equivalent to saying that its elements have the form $a_{i,j} = -a_{j,i}$ which is what happens for the matrix $\mathbf{M}$ and, therefore, we can ensure that this scheme is spectroconsistent. Following this strategy, the interpolated values of the velocity at the faces are

$$\begin{aligned}
u_e &= \frac{u_{i,j,k} + u_{i+1,j,k}}{2}; \quad u_n = \frac{u_{i,j,k} + u_{i,j+1,k}}{2}; \quad u_t = \frac{u_{i,j,k} + u_{i,j,k+1}}{2} \\
u_w &= \frac{u_{i,j,k} + u_{i-1,j,k}}{2}; \quad u_s = \frac{u_{i,j,k} + u_{i,j-1,k}}{2}; \quad u_b = \frac{u_{i,j,k} + u_{i,j,k-1}}{2} \\
v_e &= \frac{v_{i,j,k} + v_{i+1,j,k}}{2}; \quad v_n = \frac{v_{i,j,k} + v_{i,j+1,k}}{2}; \quad v_t = \frac{v_{i,j,k} + v_{i,j,k+1}}{2} \\
v_w &= \frac{v_{i,j,k} + v_{i-1,j,k}}{2}; \quad v_s = \frac{v_{i,j,k} + v_{i,j-1,k}}{2}; \quad v_b = \frac{v_{i,j,k} + v_{i,j,k-1}}{2} \\
w_e &= \frac{w_{i,j,k} + w_{i+1,j,k}}{2}; \quad w_n = \frac{w_{i,j,k} + w_{i,j+1,k}}{2}; \quad w_t = \frac{w_{i,j,k} + w_{i,j,k+1}}{2} \\
w_w &= \frac{w_{i,j,k} + w_{i-1,j,k}}{2}; \quad w_s = \frac{w_{i,j,k} + w_{i,j-1,k}}{2}; \quad w_b = \frac{w_{i,j,k} + w_{i,j,k-1}}{2}
\end{aligned} \tag{7.25}$$

With all this information, the convective term can be discretely computed and it ensures the conservation of energy.

7.2 Discrete evaluation of the diffusion

In this section the expression of the discrete diffusion will be derived and hence, the term $\vec{R}(\vec{u})$ will be completed. Recall that the term we are interested in evaluating is $\nabla\cdot(\nabla\vec{u})$ and it can be computed as before by means of the Gauss divergence theorem after integrating this quantity over a CV. Therefore,

$$\nabla \cdot (\nabla \vec{u}) \approx \frac{1}{\Omega} \int_\Omega \nabla \cdot (\nabla \vec{u}) d\Omega \tag{7.26}$$

Applying the divergence theorem for a rank 2 tensor as shown in Eq. (7.12), but being in this case $T := \nabla\vec{u}$, then the three components of the diffusion vector $\vec{D}$ are

$$
\begin{aligned}
D^x &= \left.\frac{\partial u}{\partial x}\right|_e S_e^x - \left.\frac{\partial u}{\partial x}\right|_w S_w^x + \left.\frac{\partial u}{\partial y}\right|_n S_n^x - \left.\frac{\partial u}{\partial y}\right|_s S_s^x + \left.\frac{\partial u}{\partial z}\right|_t S_t^x - \left.\frac{\partial u}{\partial z}\right|_b S_b^x \\
D^y &= \left.\frac{\partial v}{\partial x}\right|_e S_e^y - \left.\frac{\partial v}{\partial x}\right|_w S_w^y + \left.\frac{\partial v}{\partial y}\right|_n S_n^y - \left.\frac{\partial v}{\partial y}\right|_s S_s^y + \left.\frac{\partial v}{\partial z}\right|_t S_t^y - \left.\frac{\partial v}{\partial z}\right|_b S_b^y \\
D^z &= \left.\frac{\partial w}{\partial x}\right|_e S_e^z - \left.\frac{\partial w}{\partial x}\right|_w S_w^z + \left.\frac{\partial w}{\partial y}\right|_n S_n^z - \left.\frac{\partial w}{\partial y}\right|_s S_s^z + \left.\frac{\partial w}{\partial z}\right|_t S_t^z - \left.\frac{\partial w}{\partial z}\right|_b S_b^z
\end{aligned}
\tag{7.27}
$$

As references [34], [35] state, the property of spectroconsistency cannot be ensured for this term due to the fact that this is the one that physically dissipates the energy in Navier Stokes equations, yet a way to evaluate it is presented. This is a simple second order approximation for the derivatives and its expressions can be found in Eq. (7.28) and the surfaces in Eq. (7.29) .

$$
\begin{aligned}
\left.\frac{\partial u}{\partial x}\right|_e &= \frac{u_{i+1,j,k} - u_{i,j,k}}{\Delta_{i+1,j,k,1,0} - \Delta_{i,j,k,1,0}}; & \left.\frac{\partial v}{\partial x}\right|_e &= \frac{v_{i+1,j,k} - v_{i,j,k}}{\Delta_{i,j+1,k,1,1} - \Delta_{i,j,k,1,1}}; & \left.\frac{\partial w}{\partial x}\right|_e &= \frac{w_{i+1,j,k} - w_{i,j,k}}{\Delta_{i,j,k+1,1,1} - \Delta_{i,j,k,1,1}} \\
\left.\frac{\partial u}{\partial x}\right|_w &= \frac{u_{i,j,k} - u_{i-1,j,k}}{\Delta_{i,j,k,1,0} - \Delta_{i-1,j,k,1,0}}; & \left.\frac{\partial v}{\partial x}\right|_w &= \frac{v_{i,j,k} - v_{i-1,j,k}}{\Delta_{i,j,k,1,1} - \Delta_{i-1,j,k,1,1}}; & \left.\frac{\partial w}{\partial x}\right|_w &= \frac{w_{i,j,k} - w_{i-1,j,k}}{\Delta_{i,j,k,1,1} - \Delta_{i-1,j,k,1,1}} \\
\left.\frac{\partial u}{\partial y}\right|_n &= \frac{u_{i,j+1,k} - u_{i,j,k}}{\Delta_{i,j+1,k,2,1} - \Delta_{i,j,k,2,1}}; & \left.\frac{\partial v}{\partial y}\right|_n &= \frac{v_{i,j+1,k} - v_{i,j,k}}{\Delta_{i,j+1,k,2,0} - \Delta_{i,j,k,2,0}}; & \left.\frac{\partial w}{\partial y}\right|_n &= \frac{w_{i,j+1,k} - w_{i,j,k}}{\Delta_{i,j+1,k,2,1} - \Delta_{i,j,k,2,1}} \\
\left.\frac{\partial u}{\partial y}\right|_s &= \frac{u_{i,j,k} - u_{i,j-1,k}}{\Delta_{i,j,k,2,1} - \Delta_{i,j-1,k,2,1}}; & \left.\frac{\partial v}{\partial y}\right|_s &= \frac{v_{i,j,k} - v_{i,j-1,k}}{\Delta_{i,j,k,2,0} - \Delta_{i,j-1,k,2,0}}; & \left.\frac{\partial w}{\partial y}\right|_s &= \frac{w_{i,j,k} - w_{i,j-1,k}}{\Delta_{i,j,k,2,1} - \Delta_{i,j-1,k,2,1}} \\
\left.\frac{\partial u}{\partial z}\right|_t &= \frac{u_{i,j,k+1} - u_{i,j,k}}{\Delta_{i,j,k+1,3,1} - \Delta_{i,j,k,3,1}}; & \left.\frac{\partial v}{\partial z}\right|_t &= \frac{v_{i,j,k+1} - v_{i,j,k}}{\Delta_{i,j,k+1,3,1} - \Delta_{i,j,k,3,1}}; & \left.\frac{\partial w}{\partial z}\right|_t &= \frac{w_{i,j,k+1} - w_{i,j,k}}{\Delta_{i,j,k+1,3,0} - \Delta_{i,j,k,3,0}} \\
\left.\frac{\partial u}{\partial z}\right|_b &= \frac{u_{i,j,k} - u_{i,j,k-1}}{\Delta_{i,j,k,3,1} - \Delta_{i,j,k-1,3,1}}; & \left.\frac{\partial v}{\partial z}\right|_b &= \frac{v_{i,j,k} - v_{i,j,k-1}}{\Delta_{i,j,k,3,1} - \Delta_{i,j,k-1,3,1}}; & \left.\frac{\partial w}{\partial z}\right|_b &= \frac{w_{i,j,k} - w_{i,j,k-1}}{\Delta_{i,j,k,3,0} - \Delta_{i,j,k-1,3,0}}
\end{aligned}
\tag{7.28}
$$

$$
\begin{aligned}
S_e^x = S_w^x &= (\Delta_{i,j,k,2,0} - \Delta_{i,j-1,k,2,0})(\Delta_{i,j,k,3,0} - \Delta_{i,j,k-1,3,0}) \\
S_n^x = S_s^x &= (\Delta_{i+1,j,k,1,1} - \Delta_{i,j,k,1,1})(\Delta_{i,j,k,3,0} - \Delta_{i,j,k-1,3,0}) \\
S_t^x = S_b^x &= (\Delta_{i+1,j,k,1,1} - \Delta_{i,j,k,1,1})(\Delta_{i,j,k,2,0} - \Delta_{i,j-1,k,2,0}) \\
S_e^y = S_w^y &= (\Delta_{i,j+1,k,2,1} - \Delta_{i,j,k,2,1})(\Delta_{i,j,k,3,0} - \Delta_{i,j,k-1,3,0}) \\
S_n^y = S_s^y &= (\Delta_{i,j,k,1,0} - \Delta_{i-1,j,k,1,0})(\Delta_{i,j,k,3,0} - \Delta_{i,j,k-1,3,0}) \\
S_t^y = S_b^y &= (\Delta_{i,j,k,1,0} - \Delta_{i-1,j,k,1,0})(\Delta_{i,j+1,k,2,1} - \Delta_{i,j,k,2,1}) \\
S_e^z = S_w^z &= (\Delta_{i,j,k,2,0} - \Delta_{i,j-1,k,2,0})(\Delta_{i,j,k+1,3,1} - \Delta_{i,j,k,3,1}) \\
S_n^z = S_s^z &= (\Delta_{i,j,k,1,0} - \Delta_{i-1,j,k,1,0})(\Delta_{i,j,k+1,3,1} - \Delta_{i,j,k,3,1}) \\
S_t^z = S_b^z &= (\Delta_{i,j,k,1,0} - \Delta_{i-1,j,k,1,0})(\Delta_{i,j,k,2,0} - \Delta_{i,j-1,k,2,0})
\end{aligned}
\tag{7.29}
$$

With all this information the discrete approximation of the diffusion is found. The only thing left would be to multiply it by the kinematic viscosity coefficient ν and the whole term would be finished. It is worth noting that in case that a turbulent kinetic viscosity is applied, then this multiplication should be performed for each control volume, since it may vary in space and time and cannot be put outside the divergence.

7.3 Poisson's equation solution

At this stage the discrete vector $\vec{R}(\vec{u})$ is completed and thus, the discrete vector of the predictor velocity $\vec{u}^{\,p}$ can be calculated. The next thing that is needed is the divergence of this predictor velocity which can be done following a similar step like in the computation of the discrete diffusion: first apply the Gauss divergence theorem and then compute the fluxes through the faces. Since it is straightforward to do, it will be left to the reader the derivation of such discrete operator.

After that is finished, the Poisson equation of the form $\nabla^2\tilde{p} = \nabla \cdot \vec{u}^{\,p}$ must be solved. The first thing that should be done is to integrate in both sides of the equation in a control volume. This would give

$$\int_\Omega \nabla \cdot (\nabla\tilde{p}) d\Omega = \int_\Omega \nabla \cdot \vec{u}^{\,p} d\Omega \tag{7.30}$$

Applying then the Gauss divergence theorem we get that

$$\int_{\partial\Omega} \nabla\tilde{p}\; d\Omega = \int_{\partial\Omega} \vec{u}^{\,p} d\Omega \tag{7.31}$$

Which evaluated discretely and assuming a uniform grid spacing of h in all direction, leads to

$$\frac{\tilde{p}_E - \tilde{p}_P}{h}h^2 + \frac{\tilde{p}_N - \tilde{p}_P}{h}h^2 + \frac{\tilde{p}_T - \tilde{p}_P}{h}h^2 - \frac{\tilde{p}_P - \tilde{p}_W}{h}h^2 - \frac{\tilde{p}_P - \tilde{p}_S}{h}h^2 - \frac{\tilde{p}_P - \tilde{p}_B}{h}h^2 = f \tag{7.32}$$

Notice that $f := \int_{\partial\Omega} \vec{u}^{\,p} d\Omega$ and can be easily computed since the vector $\vec{u}^{\,p}$ is known. Then, the discrete equation can be rewritten as

$$a_P\tilde{p}_P + a_E\tilde{p}_E + a_W\tilde{p}_W + a_N\tilde{p}_N + a_S\tilde{p}_S + a_T\tilde{p}_T + a_B\tilde{p}_B = \frac{1}{h}f \tag{7.33}$$

with

$$\begin{aligned} a_E &= a_W = a_N = a_s = a_T = a_B = 1 \\ a_P &= -(a_E + a_W + a_N + a_s + a_T + a_B) = -6 \end{aligned} \tag{7.34}$$

Notice that this leads to a system of equations of the form

$$\mathbf{Ax} = \mathbf{b} \tag{7.35}$$

The way in which this equation will be solved is explained in Appendix C. After this is done, the pseudo-pressure -which corresponds to the vector **x**- is obtained and the velocity for the next time step can be found by computing the gradient of $\tilde{p}$ and then applying Eq. (7.5). If the grid is not uniform in z, then the discretization of the Poisson equation has the following parameters:

$$\begin{aligned} a_E &= a_W = a_{EW} = \frac{\Delta y \Delta z}{\Delta x} \\ a_N &= a_S = a_{NS} = \frac{\Delta x \Delta z}{\Delta y} \\ a_T &= \frac{\Delta x \Delta y}{\Delta z^+} \\ a_B &= \frac{\Delta x \Delta y}{\Delta z^-} \\ a_P &= -(a_E + a_W + a_N + a_s + a_T + a_B) \end{aligned} \tag{7.36}$$

The distances involved are

$$\begin{aligned}\Delta x &= \Delta_{i,j,k,1,0} - \Delta_{i-1,j,k,1,0}\\ \Delta y &= \Delta_{i,j,k,2,0} - \Delta_{i,j-1,k,2,0}\\ \Delta z^{+} &= \Delta_{i,j,k+1,3,1} - \Delta_{i,j,k,3,1}\\ \Delta z^{-} &= \Delta_{i,j,k,3,1} - \Delta_{i,j,k-1,3,1}\end{aligned} \tag{7.37}$$

Recall also, that the term $1/h$ on the RHS drops.

7.4 Boussinesq approximation and the energy conservation equation

In this section, a new equation will be introduced to the system. This is the case of the energy conservation equation that will tell us the evolution of the temperature field of the fluid. This will be important for the Boussinesq approximation since the temperature will induce an upwards convective movement.

As stated in [36], the energy conservation equation can be written as

$$\frac{\partial T}{\partial t} + (\vec{u}\cdot\nabla)T = \alpha\nabla^2 T \tag{7.38}$$

being α the thermal diffusivity defined as

$$\alpha = \frac{\lambda}{\rho c_p} \tag{7.39}$$

The rest of the parameters are λ, that stands for the thermal conductivity, ρ for the density and c_p for the specific heat capacity at constant pressure.

This equation can be integrated in a totally explicit way by performing a similar treatment as for the momentum conservation equations. This is by integrating with a second order Adams-Bashforth the convection and diffusion terms. It may be expressed as

$$T^{n+1} = T^n + \Delta t\left(\frac{3}{2}R(T^n) - \frac{1}{2}R(T^{n-1})\right) \tag{7.40}$$

Recall that the term R is defined, for this case, as

$$R(\star) = \alpha\nabla^2(\star) - (\vec{u}\cdot\nabla)(\star) \tag{7.41}$$

Then, to compute the evolution of this variable, only an evaluation of the convection and diffusion must be done. This can be performed using the same techniques shown in Section 7.1 and Section 7.2 but adapting them to the non-staggered mesh which is simpler than the already treated cases. Thus, it will be left to the reader to find those discrete expressions.

To close the Boussinesq approximation for an incompressible flow, the coupling between the energy and momentum equations is needed. This will be done by introducing a source term that is related to the temperature field and that will take into account the buoyancy forces (while keeping the flow incompressible). The derivation will be the one shown in [37].

First of all we will assume that the density is constant $\rho = \rho_0$ except for the buoyancy term. This force will be computed as

$$\vec{F}_g = \rho\vec{g} \tag{7.42}$$

Since in this term a density variation might be relevant we decompose the absolute density into a base one and a variation as

$$\rho = \rho_0 + \Delta\rho \tag{7.43}$$

On the other hand, the gravity vector can be computed from the potential $\Phi = gz$ as

$$\vec{g} = -\nabla\Phi \tag{7.44}$$

Therefore, the gravity force can be rewritten as

$$\vec{F_g} = -\nabla(\rho_0\Phi) + \Delta\rho\vec{g} \tag{7.45}$$

Then, it is possible to define a new pressure as

$$P = p + \rho_0\Phi \tag{7.46}$$

which leads to the new momentum equation of

$$\frac{\partial\vec{u}}{\partial t} + (\vec{u}\cdot\nabla)\vec{u} = -\frac{1}{\rho_0}\nabla P + \nu\nabla^2\vec{u} + \frac{\Delta\rho\vec{g}}{\rho_0} \tag{7.47}$$

One way to approximate the density variation is by linearizing on T. Doing that, we get

$$\Delta\rho = -\beta\rho_0\Delta T \tag{7.48}$$

being β the thermal expansion coefficient of the fluid. Assuming that the gravity acts in the negative direction of the z-axis, then the momentum equation that will be used is

$$\frac{\partial\vec{u}}{\partial t} + (\vec{u}\cdot\nabla)\vec{u} = -\frac{1}{\rho_0}\nabla p + \nu\nabla^2\vec{u} + g\beta(T - T_{ref})\vec{k} \tag{7.49}$$

With this addition alongside the inclusion of the energy conservation equation, we will be able to simulate the natural convective movement of a fluid.

7.5 Time integration

After the Poisson equation is solved, the new velocity can be found. The iteration is almost done and ready to go to the following time step. At this stage, though, the time step must be chosen in order to advance in time. There are some limiting cases that might be taken into account. The first of those is the restriction that arises from the convective movement. This one is the already explained Courant number. Recall that the definition of such number is

$$C = \frac{U\Delta t}{\Delta} \tag{7.50}$$

As stated in [38] its value should be smaller than one to ensure the stability of the solution. Therefore, we will compute the most restrictive time step due to the CFL condition as

$$\Delta t^C \leq \min_{(x,y,z)\in\Omega}\left(\frac{\Delta}{U}\right) \tag{7.51}$$

Where Δ can either be Δx, Δy or Δz and U will equal the corresponding component of velocity (i.e. u, v or w). On the other hand, there is another limiting term that must be taken into account.

That is diffusion, which will also introduce an upper threshold to the time step used. To find it, we can apply the von Neumann stability analysis as shown in [39]. Let's consider a transitory 1D equation with only the diffusive term. That is

$$\frac{\partial u}{\partial t} = \nu \frac{\partial^2 u}{\partial x^2} \tag{7.52}$$

Then, a finite difference explicit integration of this equation would lead to the discrete form of

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \nu \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{(\Delta x)^2} \tag{7.53}$$

This may be rewritten as

$$u_i^{n+1} = u_i^n + r\left(u_{i+1}^n - 2u_i^n + u_{i-1}^n\right) \tag{7.54}$$

being r defined like

$$r = \frac{\nu \Delta t}{(\Delta x)^2} \tag{7.55}$$

Assuming that the error ε follows the same distribution as the discrete equation, we get that

$$\varepsilon_i^{n+1} = \varepsilon_i^n + r\left(\varepsilon_{i+1}^n - 2\varepsilon_i^n + \varepsilon_{i-1}^n\right) \tag{7.56}$$

To ensure stability we should keep this error bounded in time. This means that defining

$$G = \frac{\varepsilon_i^{n+1}}{\varepsilon_i^n} \tag{7.57}$$

then, this growth factor should satisfy the constraint of

$$|G| \leq 1 \tag{7.58}$$

On the other hand, the error can be approximated using the Fourier analysis. This means that the error function will be of the form

$$\varepsilon(x,t) = e^{at} e^{ikx} \qquad \forall a \in \mathbb{R}_+ \wedge \forall k \in [0, \pi/L] \tag{7.59}$$

Introducing this approximation into Eq. (7.56) we obtain

$$e^{a(t+\Delta t)} e^{ikx} = e^{at} e^{ikx} + r\left(e^{at} e^{ik(x+\Delta x)} - 2e^{at} e^{ikx} + e^{at} e^{ik(x-\Delta x)}\right) \tag{7.60}$$

Cancelling out some terms the former expression gets simplified into

$$e^{a\Delta t} = 1 + r\left(e^{ik\Delta x} - 2 + e^{-ik\Delta x}\right) \tag{7.61}$$

Moreover, looking at the definition of G and the approximation for those terms that we have done, it is simple to see that the growth factor is also $G = e^{a\Delta t}$, which means that

$$G = 1 + r\left(e^{ik\Delta x} - 2 + e^{-ik\Delta x}\right) \tag{7.62}$$

Using the Euler equation we might simplify this expression by saying that $e^{ik\Delta x} - 2 + e^{-ik\Delta x} = 2\cos(k\Delta x)$. Therefore,

$$G = 1 + 2r\left(\cos(k\Delta x) - 1\right) \tag{7.63}$$

Imposing then the stability restriction of Eq. (7.58) we get two inequalities that must be satisfied. Those are

$$\begin{aligned} 1 + 2r\left(\cos(k\Delta x) - 1\right) &\leq 1 \\ -1 - 2r\left(\cos(k\Delta x) - 1\right) &\leq 1 \end{aligned} \tag{7.64}$$

After some manipulation, the former constraint leads to Eq. (7.65). Note that by the definition of the cosine function this is always satisfied, which indicates that no special restriction of r is needed to ensure this part of the inequality.

$$\cos(k\Delta x) \leq 1 \tag{7.65}$$

On the other hand, though, the latter expression of Eq. (7.64) gets simplified into Eq. (7.66). Taking the most extreme values of the cosine function, which are -1, 0 and 1, then we will get three limiting values. Those are $r \leq 1/2$ and $r \leq 1$ for the cosine values of -1 and 0, respectively. The value of 1 does not introduce any restriction, since the inequality is fulfilled for any value of r.

$$r[1 - \cos(k\Delta x)] \leq 1 \tag{7.66}$$

Looking at this result, the most restrictive value of r is $1/2$. This leads to the time steps allowed of

$$\Delta t^{\nu} \leq \frac{(\Delta x)^2}{2\nu} \tag{7.67}$$

In case that the energy conservation equation is being solved, then another term will have to be taken into account. This is the diffusion of temperature which has a form of

$$\Delta t^{\alpha} \leq \frac{(\Delta x)^2}{2\alpha} \tag{7.68}$$

To close up, the time step that will be chosen will be

$$\Delta t = f_S \min\left[\Delta t^C, \Delta t^{\nu}, \Delta t^{\alpha}\right] \tag{7.69}$$

where a safety factor f_S has been introduced. This factor is very important since many assumptions have been made in the procedure of showing the stability region that are not exactly true and hence, to overcome this error the final time step chosen will be smaller than the theoretical maximum predicted. For a practical implementation this factor is about $f_S \sim 0.1$.

8 — Atmospheric Boussinesq approximation

The Boussinesq approximation has already been described for an incompressible flow for a general application, but it is not suitable for the explanation of the atmospheric phenomena. Therefore, in this chapter the Boussinesq approximation for atmospheric models will be derived and explained as well as its limitations. At the end, a brief comment on how to numerically solve the equations will be presented.

8.1 Approximation to stratified compressible flow

In this section, the procedure to derive the new set of equation presented in [40] will be followed. The most important aspect of this procedure is the introduction of new variables that are more suitable for atmospheric applications than the primitive ones.

To start this derivation, the momentum equation from the classical Boussinesq approximation will be considered. Recall that this can be written as

$$\rho\left[\frac{\partial\vec{u}}{\partial t}+(\vec{u}\cdot\nabla)\vec{u}\right]=-\nabla p-g\rho\vec{k} \tag{8.1}$$

Taking into account the physics of an atmosphere, the density and pressure can be decomposed into a background contribution (only a function of the height) and a perturbation term. This decomposition may be written like

$$\begin{aligned}\rho(x,y,z,t)&=\rho_0(z)+\rho'(x,y,z,t)\\ p(x,y,z,t)&=p_0(z)+p'(x,y,z,t)\end{aligned} \tag{8.2}$$

Additionally, the hydrostatic approximation for the background quantities may be assumed. Therefore,

$$\frac{dp_0}{dz}=-\rho_0 g \tag{8.3}$$

Consider also the definition of the following terms:

$$P=\frac{p'}{\rho_0};\quad b=-g\frac{\rho'}{\rho_0};\quad N^2=-\frac{g}{\rho_0}\frac{d\rho_0}{dz} \tag{8.4}$$

The first of those is equivalent to a pressure potential, the second is the buoyancy acceleration and N the Brunt-Väisälä frequency. All in all, make the system of the traditional Boussinesq approximation for an incompressible, adiabatic, inviscid and atmospheric flow, which can be read as

$$\begin{aligned} \frac{\partial \vec{u}}{\partial t} + (\vec{u}\cdot\nabla)\vec{u} &= \nabla P + b\vec{k} \\ \frac{\partial b}{\partial t} + (\vec{u}\cdot\nabla)b + N^2 w &= 0 \\ \nabla\cdot\vec{u} &= 0 \end{aligned} \tag{8.5}$$

Now, to extend this approximation to a compressible flow (assuming that the atmosphere behaves like a perfect gas) we first need to introduce two new variables. The first of those is the Exner function. It is computed as

$$\pi = \left(\frac{p}{p_0}\right)^{R/c_p} \tag{8.6}$$

and it is related not only to the pressure, but also to the properties of the gas, since both the specific heat capacity c_p and the specific gas constant R appear on its definition.

The second new variable that will be introduced is the potential temperature. Its definition is

$$\theta = T\left(\frac{p_0}{p}\right)^{R/c_p} \tag{8.7}$$

A way to relate the gradient of the pressure and the gradient of the Exner function is needed. Then,

$$\nabla p = \nabla\left[p_0\pi^{c_p/R}\right] = p_0\frac{c_p}{R}\pi^{c_p/R-1}\nabla\pi = \frac{c_p}{R}\frac{p}{\pi}\nabla\pi \tag{8.8}$$

The perfect gases law gives a relation for the pressure and states, recalling also that $T = \theta\pi$, the following equivalence

$$p = \rho RT = \rho R\theta\pi \tag{8.9}$$

Introducing this expression into Eq. (8.8) leads to

$$\frac{1}{\rho}\nabla p = c_p\theta\nabla\pi \tag{8.10}$$

Moreover, the Brunt-Väisälä frequency can be computed as

$$N^2 = \frac{g}{\theta_0}\frac{d\theta_0}{dz} \tag{8.11}$$

and the buoyancy acceleration

$$b = \frac{\theta'}{\theta_0}g \tag{8.12}$$

Therefore, the final set of equations that will be used, can be obtained from introducing these new variables and the effect of the Coriolis force into Eq. (8.5) and the result is

$$\begin{aligned} \frac{\partial \vec{u}}{\partial t} + (\vec{u}\cdot\nabla)\vec{u} - f\left(\vec{u}\times\vec{k}\right) &= -c_p\theta_0\nabla\pi' + b\vec{k} \\ \frac{\partial b}{\partial t} + (\vec{u}\cdot\nabla)b + N^2 w &= 0 \\ \nabla\cdot\vec{u} &= 0 \\ b &= \frac{\theta'}{\theta_0}g \\ N^2 &= \frac{g}{\theta_0}\frac{d\theta_0}{dz} \end{aligned} \tag{8.13}$$

8.2 Numerical resolution

Looking at the formal expression of the equations of this approximation in Eq. (8.13) and the one used for the incompressible Navier-Stokes explained in Section 7.4 we can see that they are almost equivalent. Therefore, the numerical integration will follow a similar procedure.

The Poisson equation will be in this case

$$\nabla^2 \tilde{\pi} = \nabla \cdot \vec{u}^{\,p} \tag{8.14}$$

where the predictor velocity is computed using

$$\vec{R}(\vec{u}^{\,n}) = -(\vec{u}^{\,n} \cdot \nabla)\vec{u}^{\,n} + b^n \vec{k} \tag{8.15}$$

Notice, that although similar, this model is completely inviscid and, therefore, no diffusion will be evaluated. This is due to the relatively big domains that will be simulated, which make the molecular diffusion small at those scales. The way in which convection is computed is totally analogous to the one presented in Chapter 7 and the buoyancy will be evaluated at the n^{th} time step and will be linearly interpolated at the staggered position of the grid.

The Coriolis term can be integrated in the same fashion as presented in the SW model, but all the runs here performed will not use it and therefore, has not been implemented. For the conservation of energy, the procedure will be completely explicit as used in Section 7.4 but solving for b. This means that the evolution will be given by

$$b^{n+1} = b^n + \Delta t \left(\frac{3}{2} R(b^n) - \frac{1}{2} R(b^{n-1}) \right) \tag{8.16}$$

with

$$R(b^n) = -(\vec{u}^{\,n} \cdot \nabla) b^n - N^2 w^n \tag{8.17}$$

Notice that by the definition of the Brunt-Väisälä frequency in this model, this quantity will be constant in time (but not in space, since it might depend on the height), which means that can be precomputed and, therefore, does not cause any problem in the numerical integration of the set of equations.

Another important aspect is the constraint for the time step. In this case, since there is no temperature nor molecular diffusion, in principle only the CFL condition would apply. This is not exactly true, though, since the Brunt-Väisälä frequency introduces a new constraint to the time step. As explained in [41], this constraint is related to the internal gravity waves that the model may introduce and that must be solved. Therefore,

$$\Delta t^{BV} \leq \frac{1}{N} \tag{8.18}$$

All in all, the time step that will be chosen is

$$\Delta t = f_S \min \left[\Delta t^C, \Delta t^{BV} \right] \tag{8.19}$$

Again the saftey factor f_S appears and it will be set close to $f_S \sim 0.1$.

9 — Simulation results

In this chapter, all the different simulations that have been carried out with the 3D code will be displayed. There will be two benchmark cases that correspond to the Taylor Green Vortices and the Rayleigh-Bénard convection that will serve to check the correct numerical resolution of the incompressible Navier-Stokes equations.

Then, a benchmark simulation regarding the Boussinesq atmospheric model will be carried out. This is the so-called Robert Rising Bubble and will serve to check the correct functioning of the implementation of the model. It will be important to recall that the models used in the literature are not exactly equal and, therefore, the results have to be similar, yet not equal.

9.1 Taylor-Green Vortices

This case was firstly described by Taylor and Green [42] and it consists of setting the initial conditions shown in Eq. (9.1) in a full periodic cubic domain and letting the system evolve. This system has an analytical solution and hence it serves greatly as a benchmark case to test the solver.

$$\begin{aligned} u &= A\cos(ax)\sin(by)\sin(cz) \\ v &= B\sin(ax)\cos(by)\sin(cz) \\ w &= C\sin(ax)\sin(by)\cos(cz) \\ 0 &= Aa + Bb + Cc \end{aligned} \tag{9.1}$$

To compare with the solution of DeBonis [43], the Re must be ensured to be equal to 1600. Then, the following parameters have been set:

- $A = V_0$
- $B = -V_0$
- $C = 0$
- $a = b = c = L^{-1}$
- $\Omega = [0, 2\pi L] \times [0, 2\pi L] \times [0, 2\pi L]$
- $L = 1\ m$
- $\rho = 1\ kg/m^3$
- $\nu = 1/1600\ m^2/s$

DeBonis uses a length of $L = 0.005\ ft$ and then defines a dimensionless time. In the present work, though, the length is unitary and thus, the dimensionless time will be equal to the absolute time. To be able to check the good functioning, the integrated kinetic energy and its dissipation rate will be computed and compared to the reference. The formulae that gives reference [43] are Eq. (9.2) and Eq. (9.3) for the kinetic energy and kinetic energy dissipation, respectively. In the former of these expressions, it has been assumed that the flow is incompressible.

$$E_k = \frac{1}{\Omega}\int_\Omega \frac{u_i u_i}{2} d\Omega \tag{9.2}$$

$$\varepsilon(E_k) = -\frac{dE_k}{dt} \tag{9.3}$$

The plot shown in Figure 9.1 displays a comparison between the results obtained with the code here developed and the reference solution. Additionally, in Figure 9.2 a 3D representation of the structures that appear can be seen. The representation uses the Q-criterion isosurfaces. This new variable Q is computed as the second invariant of the tensor $\nabla\vec{u}$ [44]. Therefore, its formal definition may be written as

$$Q = \frac{1}{2}\left(tr(\nabla\vec{u})^2 - tr([\nabla\vec{u}\ ]^2)\right) \tag{9.4}$$

The results presented in Figure 9.1 demonstrate a good functioning of the code and are in accordance to the ones obtained in [45] who also uses an spectroconsistent scheme for the same meshes.

9.2 Rayleigh-Bénard convection

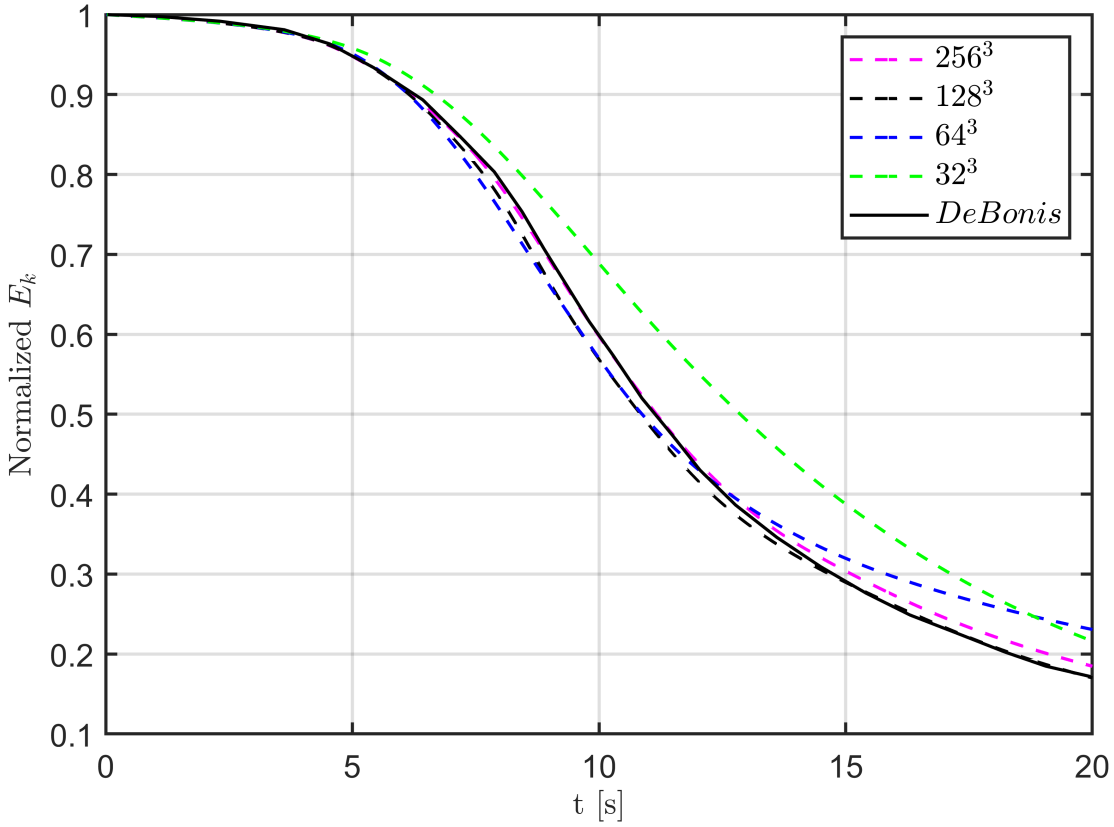
This problem was firstly seen by Bénard when performing an experiment [46] and was then theoretically explained by Lord Rayleigh [47]. It consists of two horizontal isothermal plates, being the top one the colder and the bottom one the hotter with respect to the reference temperature. Then, if the dimensionless Rayleigh number (Ra) is above a certain threshold, then the system becomes unstable and forms turbulent structures within the domain.

It's a case that has been thoroughly studied and many simulations have been performed. Therefore, it is a good benchmark to test the Boussinesq approximation for the incompressible Navier-Stokes equations and the solution of the energy conservation equation. Additionally, it will serve to introduce new boundary conditions (that will be Dirichlet type at the top and bottom boundaries) and a non-uniformity in the grid for the z-axis. Therefore, some changes to the code will be needed. The way in which BC have been implemented are shown in Appendix D and the changes performed to the Poisson solver in Section C.2.

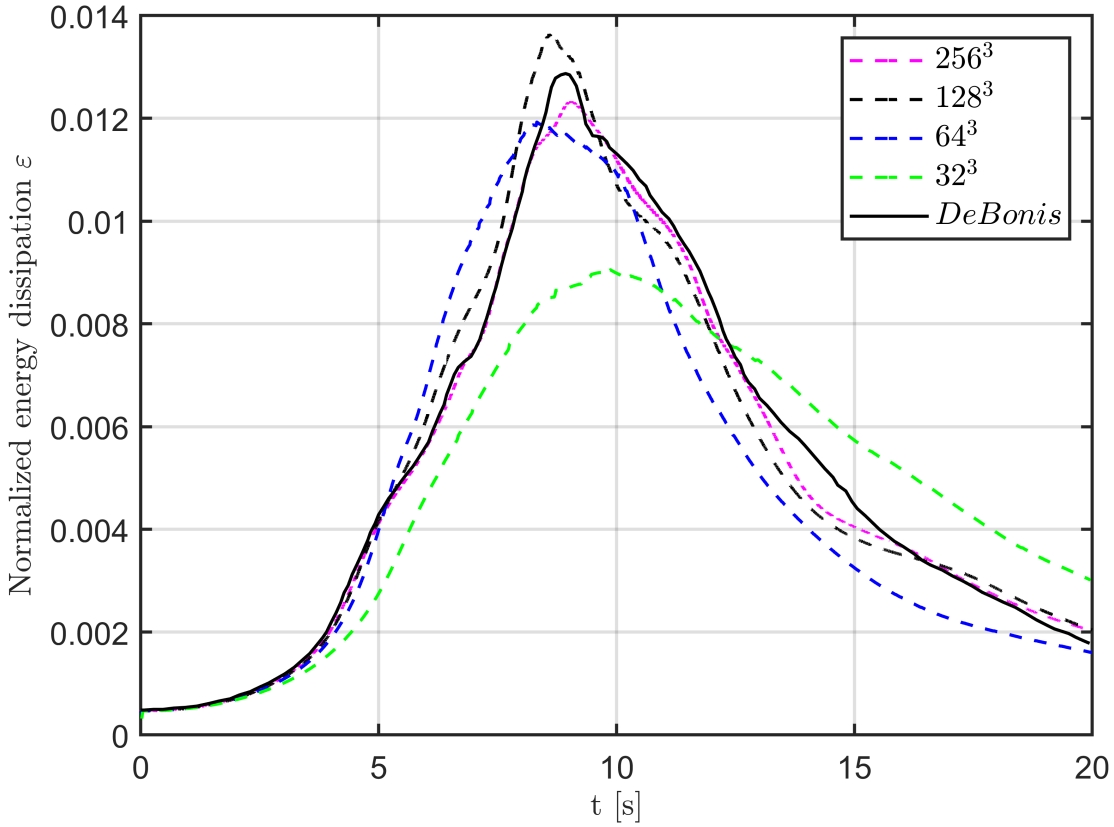
There are many configurations available, but we need a rectangular domain, with periodicity in the x and y axis and the isothermal plates at top and bottom. A reference that reproduces this situation is [48] and thus, it will be taken as a guide for the problem definition. The temperature difference is taken as $\Delta T = 1\ K$ and the aspect ratio will be $\Gamma = L/H = \pi$. In this case, L will stand for the length of the domain for the x and y axis and H for the length in the z-direction. Then, recalling that the Rayleigh number is

$$Ra = \frac{g\beta\Delta T H^3}{\nu\alpha} \tag{9.5}$$

and the Prandtl number

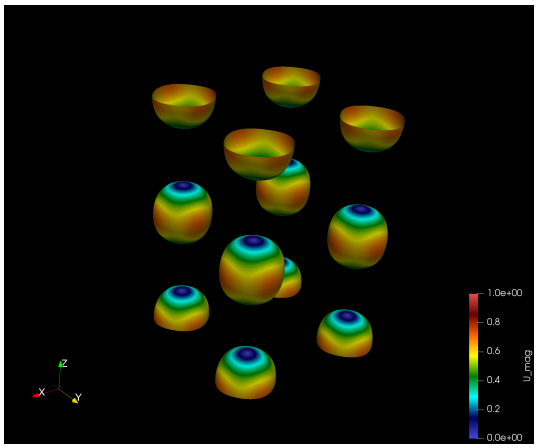


(a) Energy evolution.

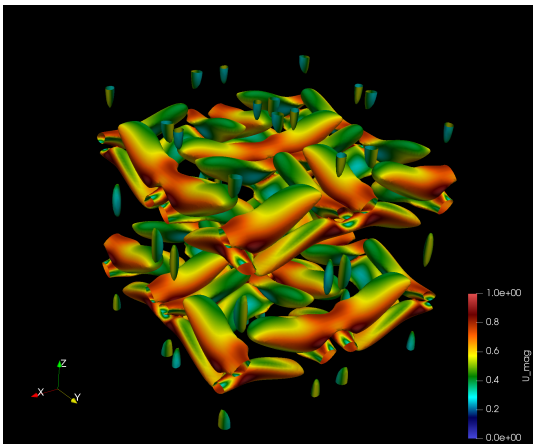


(b) Energy dissipation evolution.

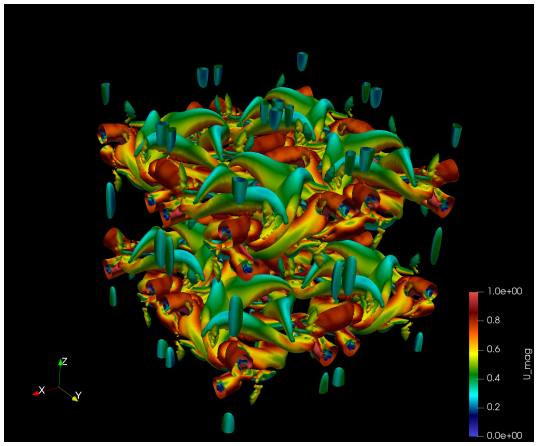
Figure 9.1: TGV energy results.



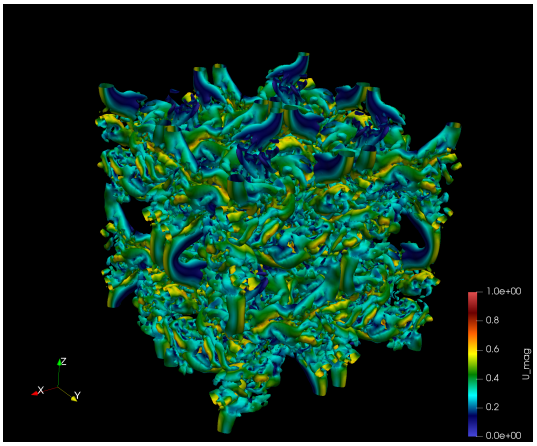
(a) $t = 0\ s$



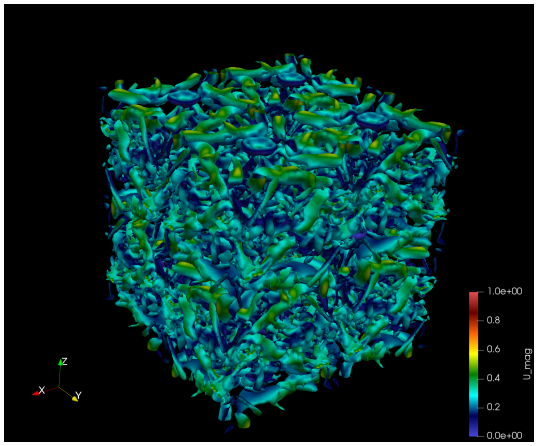
(b) $t = 4\ s$



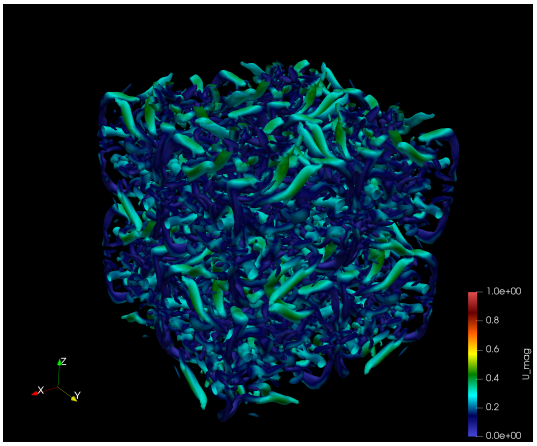
(c) $t = 6\ s$



(d) $t = 12\ s$



(e) $t = 16\ s$



(f) $t = 20\ s$

Figure 9.2: Isosurfaces of $Q = 0.52$ colored by the magnitude of the velocity for the simulation of the Taylor-Green Vortices with $N = 128^3$.

$$Pr = \frac{\nu}{\alpha} \tag{9.6}$$

A way to compare the results will be to compute the average Nusselt number, which according to [48], may be defined as

$$Nu = \sqrt{RaPr}\langle\overline{wT}\rangle - \frac{\partial\langle\overline{T}\rangle}{\partial z} \tag{9.7}$$

where $\bar{\xi}$ denotes the spatial average on the x-y plane and $\langle\xi\rangle$ denotes the time average. Then, the value that will be compared is for a $Pr = 0.7$ and $Ra = 10^8$. The mesh density function that has been used in the z-axis is

$$z'_k = \frac{H}{2}\left(1 + \frac{\tanh\left(p\left[\frac{2z_k}{H} - 1\right]\right)}{\tanh(p)}\right) \tag{9.8}$$

being z_k the original z coordinate of the k^{th} point and z'_k the new one. The parameter p controls how irregular the grid spacing will be. For values of $p \sim 0.01$ the mesh is almost uniform and the larger it gets, the most non-uniform is. For this case, it has been set to $p = 2$, since otherwise the CFL criterion gets too restrictive. The mesh that will be used is $N_x \times N_y \times N_z = 128^3$. The evolution of the Nusselt number can be seen in Figure 9.3. Some snapshots of the volumetric visualization of this simulation are presented in Figure 9.4.

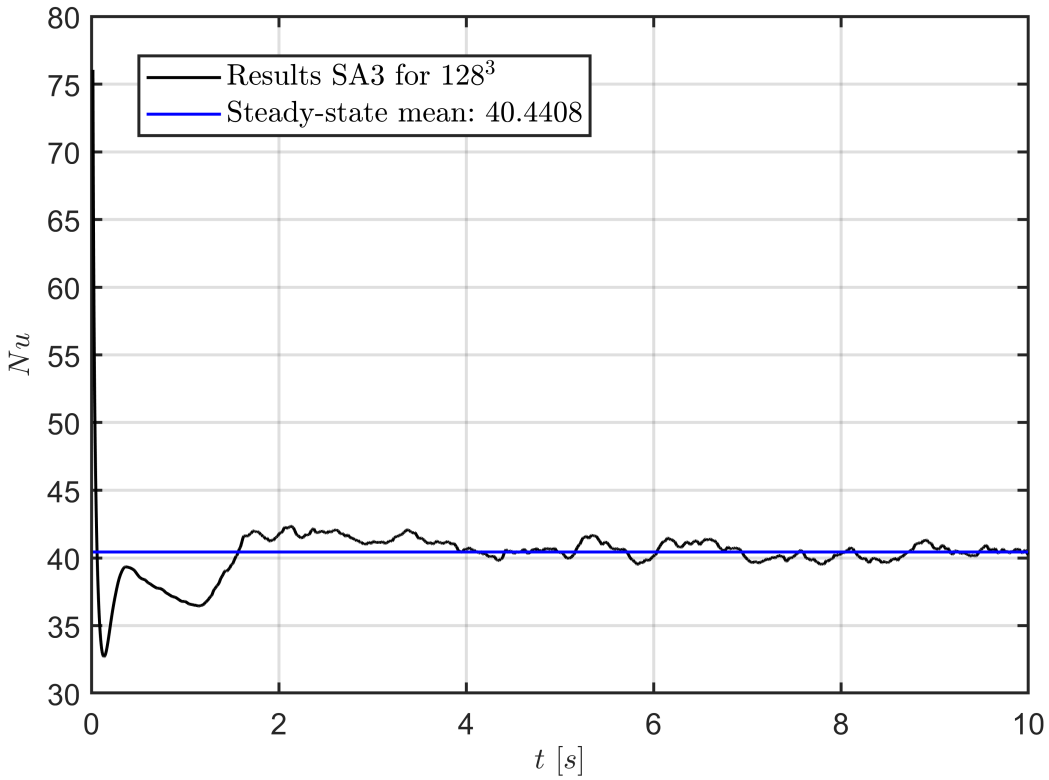
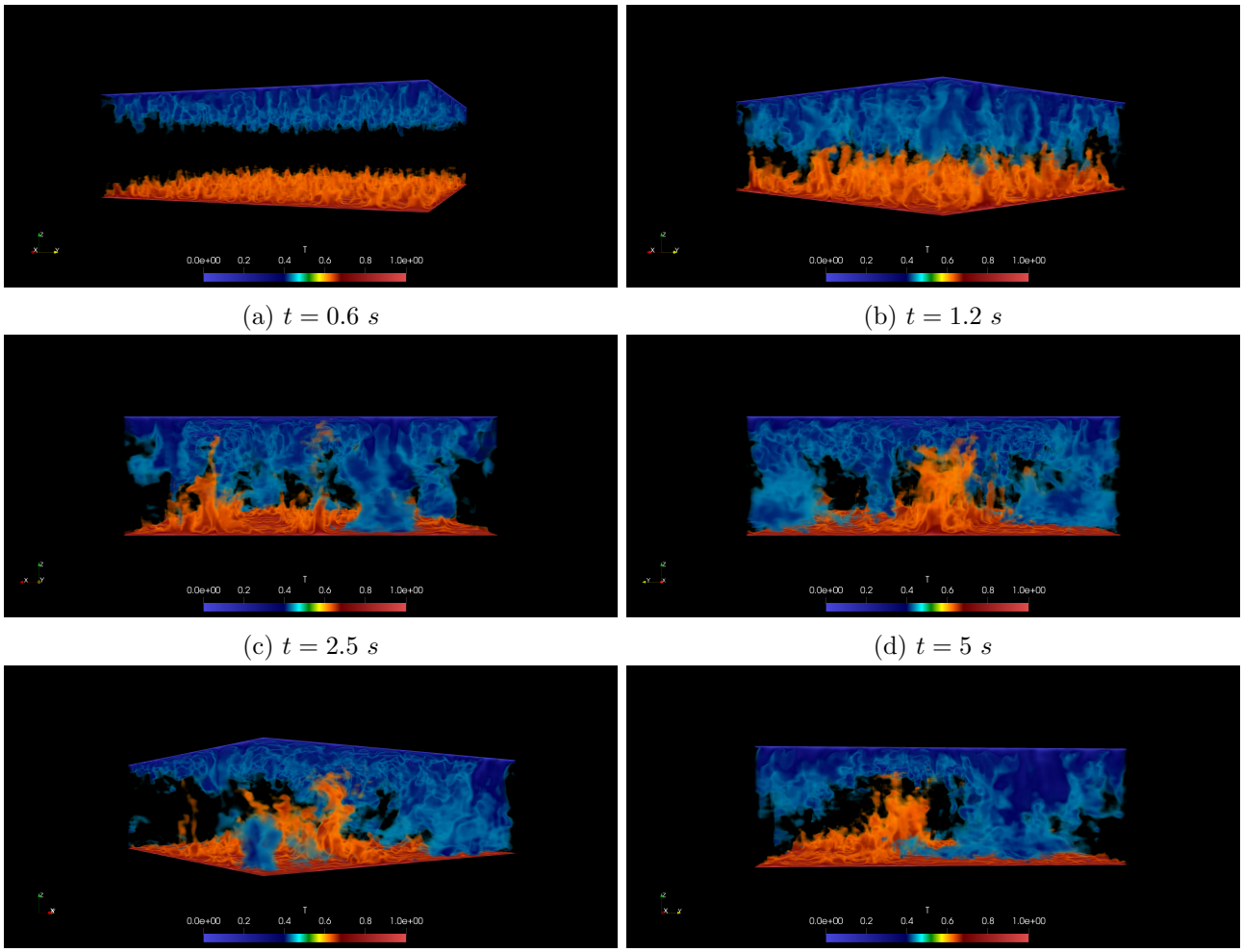


Figure 9.3: Time evolution of Nu for the Rayleigh-Bénard case.

The results must be compared to the ones presented in the literature but plenty of options are available. Since this is a very well-studied case, many different aspect ratios, geometries and dimensionless numbers have been explored and it will be compared to the experimental results presented by Hollands [49]. They predict $Nu = 37.2$ for $Ra = 10^8$. The mean for the steady-state value of the



(a) $t = 0.6\ s$ (b) $t = 1.2\ s$

(c) $t = 2.5\ s$ (d) $t = 5\ s$

(e) $t = 9\ s$ (f) $t = 10\ s$

Figure 9.4: Results of the Rayleigh-Bénard with $N = 128^3$ at the times indicated.

Nusselt number is $Nu_{128^3} = 40.4$. If we plot the result here obtained alongside the experimental results (as seen in Figure 9.5) we can conclude that still is on the acceptable range taking also into account that the meshes used to obtain the published results in most papers contain about 16 million CVs or more while here roughly 2 million where used. This would explain the slight difference between the predicted and obtained results. It is also noticeable the difference between the values given by different authors. For example, reference [48] gives a value of 30.9 for this same case.

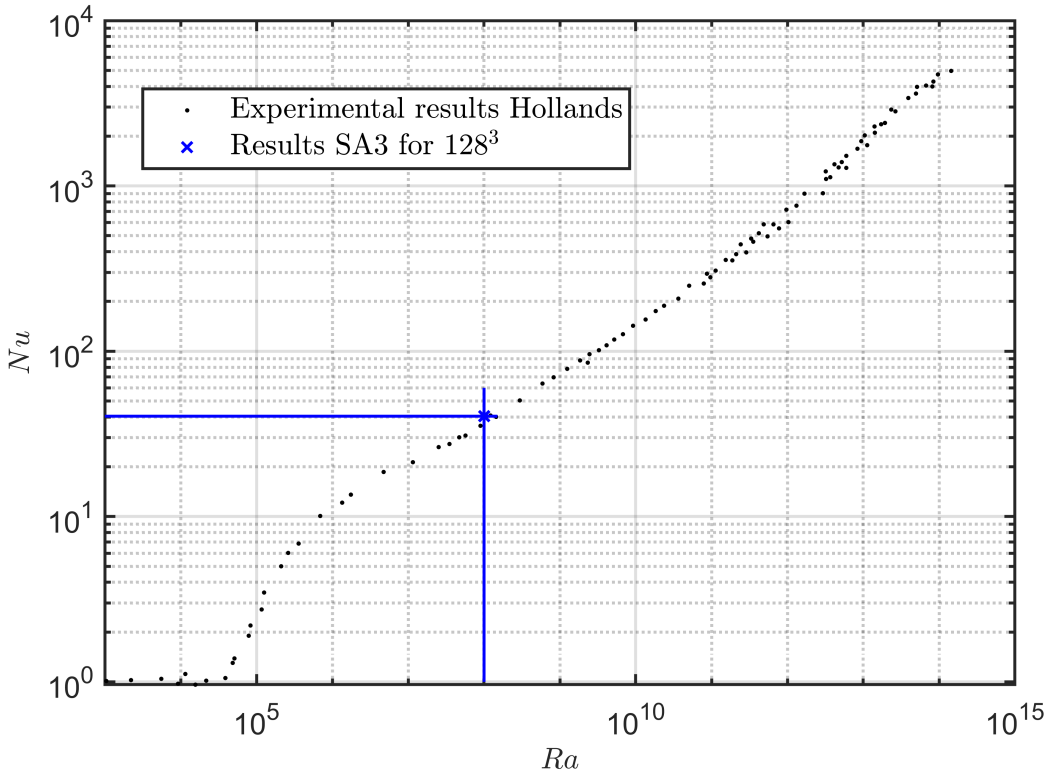


Figure 9.5: Comparison of Nu for the Rayleigh-Bénard simulation alongside the experimental results of [49].

9.3 Robert Rising Bubble

This last section will be devoted to preforming a benchmark for the three dimensional model of the atmosphere. This test will be the so-called Robert Rising Bubble. This numerical experiment was firstly described by Robert [50] and has been reproduced since then by different atmospheric CFD solvers and techniques and thus, will serve so as to check whether our model and implementation is suitable enough to describe the physics of it as well.

This benchmark consists on perturbing the potential temperature field in a rectangular 2D domain of $\Omega = (x, z) \in [0, 1] \times [0, 1.5]\ km$ and letting the system evolve until 20 min. The perturbation will have the following shape [51]:

$$\theta' = \begin{cases} 0 & \text{for } r > r_c \\ \dfrac{\theta_c}{2}\left[1 + \cos\left(\dfrac{\pi r}{r_c}\right)\right] & \text{for } r \leq r_c \end{cases} \tag{9.9}$$

being $r = \sqrt{(x - x_c)^2 + (z - z_c)^2}$. The perturbation will have the parameters of $\theta_c = 0.5\ K$, $r_c = 250\ m$,

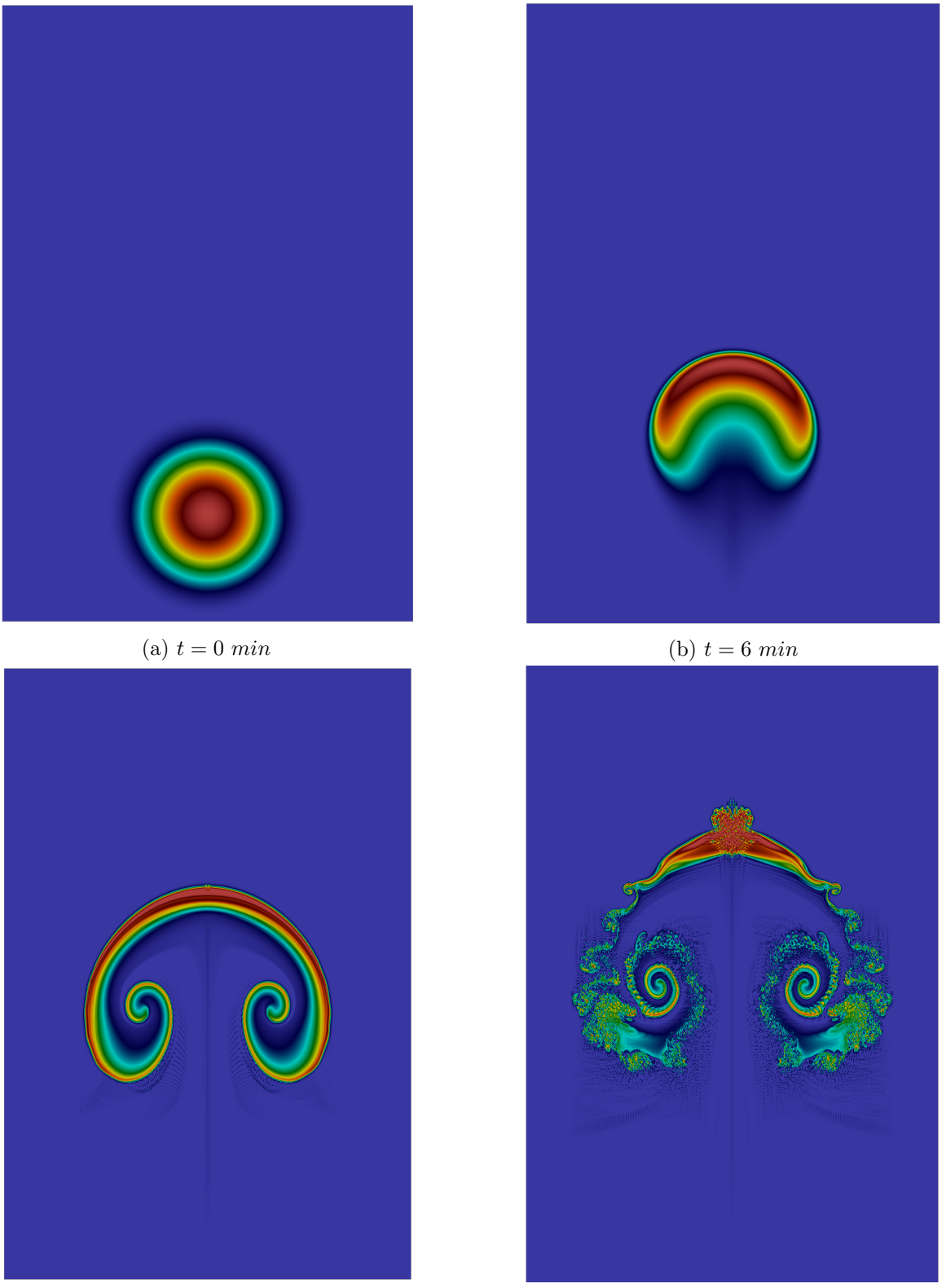
$x_c = 500\ m$ and $z_c = 260\ m$. The background potential temperature will be set, according to [52], to $\theta_0 = 303.5\ K$.

In order to reproduce this experiment and since the code here developed is three dimensional, a very small amount of CV will be set in the y-axis. It is also worth noting that the reference solution is only qualitative since only a plot of the potential temperature perturbation distribution is given but not any statistics to compare to. Therefore, it will be said that the code works only if it exhibits similar structures.

The first simulation carried out led to the results presented in Figure 9.6. As it can be seen, a numerical instability appears and ends up blowing up the simulation. This type of behaviour is expected for spectroconsistent schemes, since no dissipation mechanism is available (like an LES model or like a TVD scheme), then is prone to generating instabilities. It is obvious that we should come up with a solution to overcome this problem.

A possible solution would be to program the TVD schemes explained for the SW model. They are particularly good at dealing with situations in which this type of instabilities appear. Hence, a simulation using the MUSCL scheme is presented in Figure 9.7. Although the results are way much better and the instabilities are completely gone, the shape of the central part of the bubble is completely wrong. As presented in [50]–[52] the central part of the bubble, instead of ripping apart like it happens when the MUSCL scheme is used, it should create vertical filaments that rise up. Therefore, the results are physically incorrect since it dissipates part of the energy in a place where changes completely the final state.

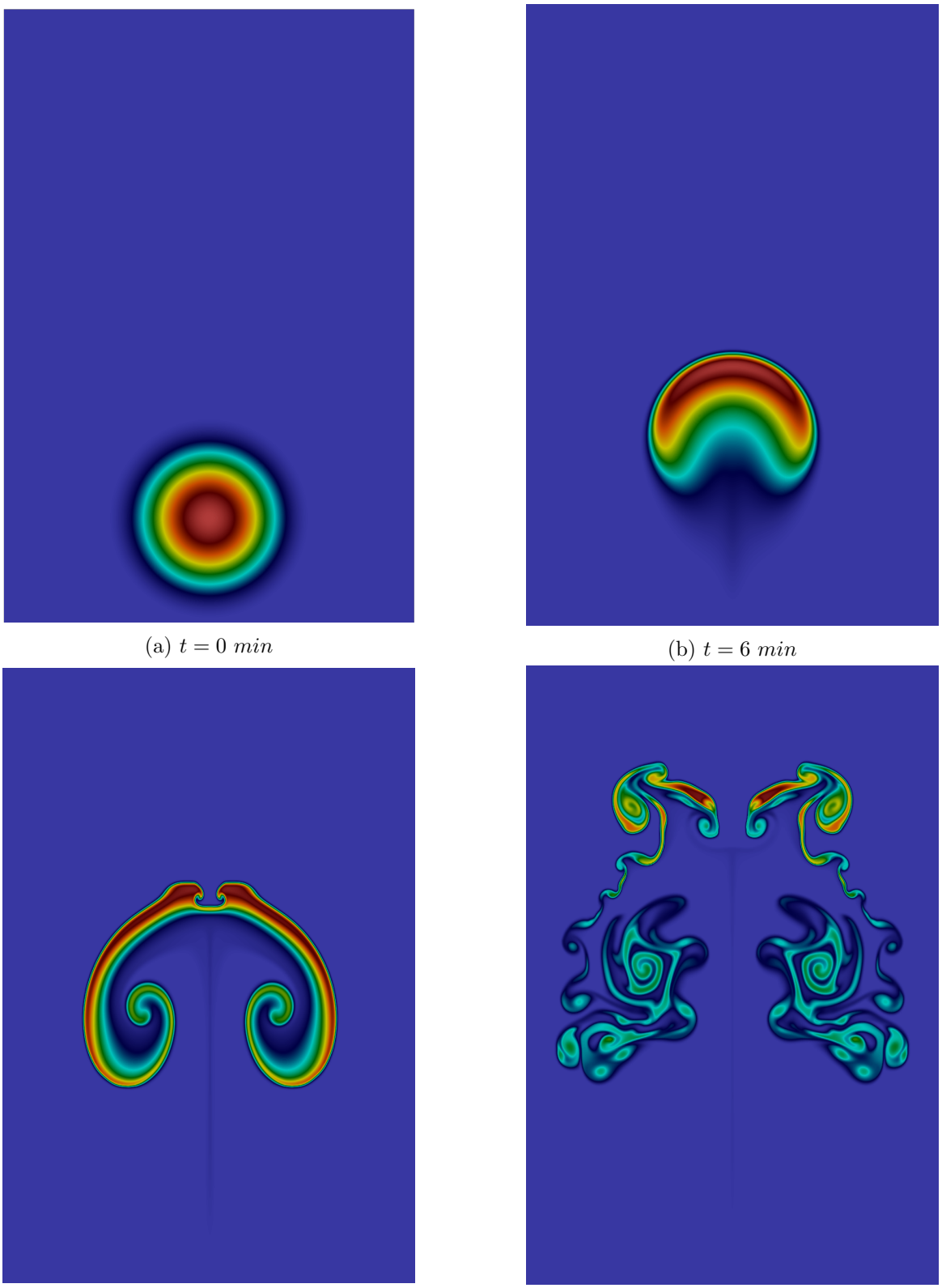
Another solution that can be tested is adding a filter like the hyperviscosity. Since this is a Cartesian domain it is much easier to implement and in [51] need this type of filtering to overcome the spurious oscillations that also have. Therefore, a simulation using a hyperviscosity of second order is displayed in Figure 9.8. This is a really good result since barely no ripple is appreciated and at the same time steps as the literature provide, the shape of the resulting bubble is very similar to theirs.



(a) $t = 0\ min$ (b) $t = 6\ min$

(c) $t = 12\ min$ (d) $t = 15\ min$

Figure 9.6: Results of the Robert Rising Bubble experiment with $N = 1024 \times 1024$ using the spectro-consistent scheme and no additional stabilization technique at the times indicated.



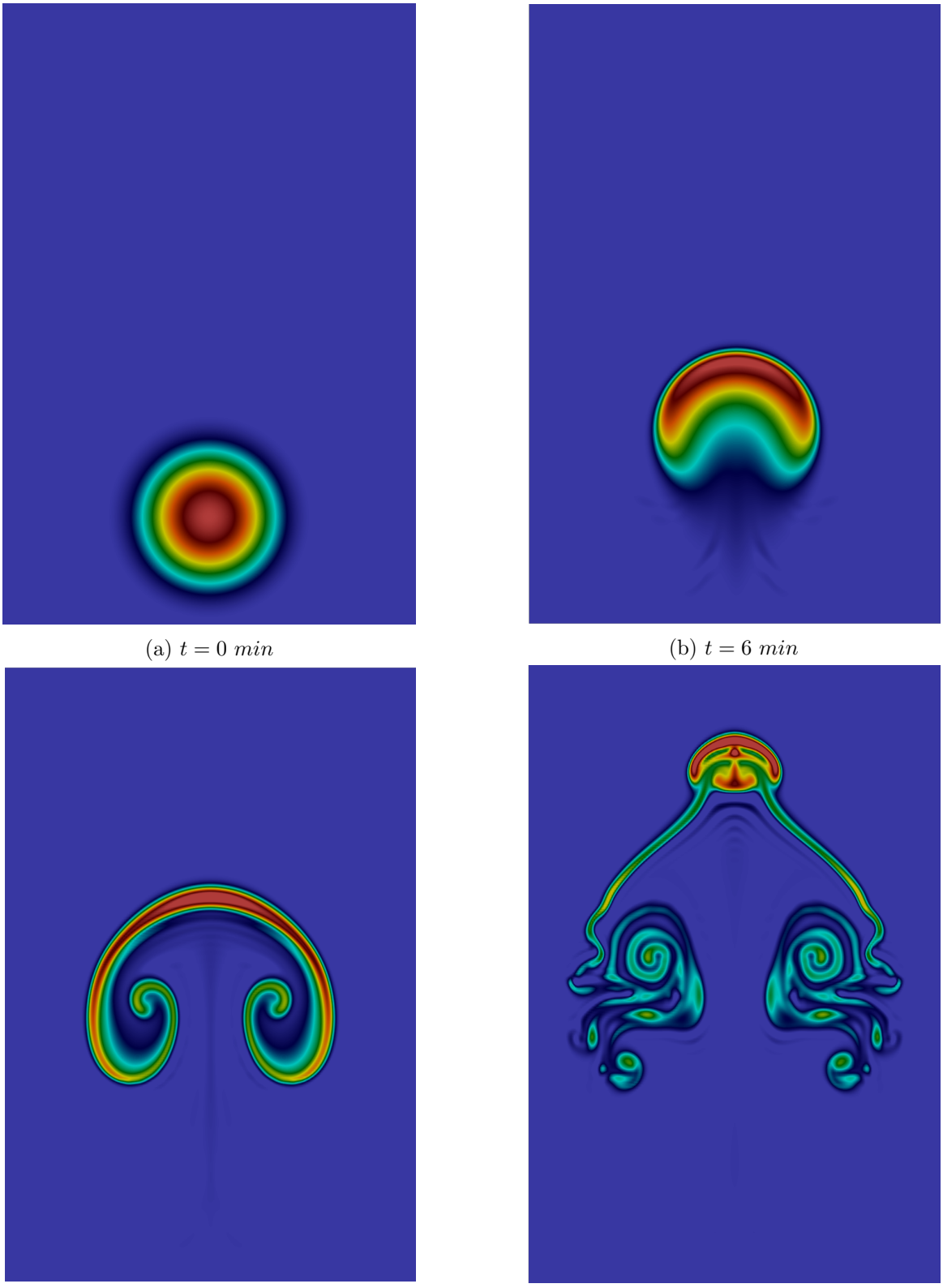
(a) $t = 0\ min$

(b) $t = 6\ min$

(c) $t = 12\ min$

(d) $t = 18\ min$

Figure 9.7: Results of the Robert Rising Bubble experiment with $N = 256 \times 256$ using the MUSCL scheme and no additional stabilization technique at the times indicated.



(a) $t = 0\ min$

(b) $t = 6\ min$

(c) $t = 12\ min$

(d) $t = 18\ min$

Figure 9.8: Results of the Robert Rising Bubble experiment with $N = 256 \times 256$ using the spectroconsistent scheme and hyperviscosity at the times indicated.

Conclusions

At first, we have started by using a two dimensional model to reproduce atmospheric phenomena at large scales. This model, although it can capture many great detail it lacks the vertical convection that is also of great interest. After having understood the hypothesis of a shallow geophysical fluid a solver has been coded and validated a solver in $MATLAB^{®}$. Then, some improvements to a parallel code that uses such model have been done.

The first of such improvements is the introduction of vortices via the geostrophic equilibrium mechanism. The simplifications done were theoretically very accurate, which has been seen by comparing it to the available experimental data for the Great Red Spot of Jupiter, but the simulations have not led to the expected results. Therefore, it might be useful to introduce perturbations in a different fashion, but the steady-state vortex will differ from the expected one.

The second implementation introduced has been the control of the Hollingsworth instability that has appeared in some high resolution simulations. After redoing the simulations with the hyperviscosity filter it has turned out as not a feasible solution, since it retards the apparition of the instability, yet it does not prevent it. A very interesting property that has been seen is that this filter does not increase the energy rate dissipation. To avoid this instability it has been seen that a change of the limiter function overcomes the problem. In this case, the one that has been substituted for is the MUSCL scheme, but as a further development, other schemes should be tested.

To end up with the two-dimensional model, the addition of the capability of simulating the polar regions has been done. In this case, the results are encouraging since the code can reproduce part of the turbulent dynamics that are observed in Jupiter. It is important, though, to perform a thorough analysis on how the different parameters affect the final result and try to tune them to reproduce more accurately the steady vortices observed in gas giant planets.

Regarding the 3D model, a parallel solver for the incompressible Navier-Stokes equations has been done. With it, after validated, some benchmark cases have been solved. These are the Taylor-Green Vortices and the Rayleigh-Bénard cases. The former of those, has led to the expected results and has been compared to a reference that also uses the same spectroconsitent scheme but with a sequential code and both exhibit the same behaviour. The latter, leads to reasonably similar results, yet a finer mesh should be needed to confirm it.

After the incompressible Navier-Stokes equations have been integrated, a Boussinesq model for atmospheres has been coded. They are formally equivalent and thus, its implementation has been straight-forward. After validated, a benchmark case has been reproduced. This case is the Robert Rising Bubble which has been used to check many other atmospheric models. The results obtained show that if the spectroconsistent scheme is used, it needs the presence of a filter (in this case a hyperviscosity has been used) since otherwise, an instability appears. On the other hand, if a TVD scheme is used, the final results differ from the expected ones. This is due to an extreme dissipation of energy in a sensitive part of the bubble. Then, as a future development, this model should be used to try to reproduce some real atmospheric phenomena like a small jovian storm or similar.

References

[1] L. F. Richardson and P. Lynch, *Weather Prediction by Numerical Process*, 2nd ed. New York: Cambridge University Press, 2007.

[2] E. García-Melendo and A. Sánchez-Lavega, "Shallow water simulations of Saturn's giant storms at different latitudes", *Icarus*, vol. 286, pp. 241–260, Apr. 2017. DOI: 10.1016/j.icarus.2016.10.006.

[3] A. Prat, *Shallow Water Model Code for Applications to Atmosphere Dynamics (BSc. Thesis)*. Universitat Politècnica de Catalunya, 2018.

[4] B. Cushman-Roisin and J.-M. Beckers, *Introduction to Geophysical Fluid Dynamics. Physical and Numerical Aspects*. 2009, ISBN: 9780120887590.

[5] G. Vallis, *Atmospheric and Oceanic Fluid Dynamics*, 2nd ed. Cambridge University Press, 2005, ISBN: 9781107588417.

[6] A. Staniforth and A. White, "The shallow-water equations in non-spherical geometry with latitudinal variation of gravity", *Quarterly Journal of the Royal Meteorological Society*, vol. 141, no. 687, pp. 655–662, 2015. DOI: 10.1002/qj.2394.

[7] A. A. White and N. Wood, "Consistent approximate models of the global atmosphere in non-spherical geopotential coordinates", *Quarterly Journal of the Royal Meteorological Society*, vol. 138, pp. 980–988, 2012. DOI: 10.1002/qj.972.

[8] A. Arakawa and V. R. Lamb, "Computational Design of the Basic Dynamical Processes of the UCLA General Circulation Model", *Methods in Computational Physics: Advances in Research and Applications*, vol. 17, pp. 173–265, 1977. DOI: 10.1016/b978-0-12-460817-7.50009-4.

[9] J. Kämpf, *Ocean Modelling for Beginners*, 1st ed. Berlin: Springer Verlag, 2009, ISBN: 9783642008191.

[10] H. Versteeg and W. Malalasekera, *An Introduction to Computational Fluid Dynamics*, 2nd ed., 4. Harlow: Pearson, 2007, vol. 6, pp. 78–78, ISBN: 9780131274983.

[11] O. B. Fringer, S. W. Armfield, and R. L. Street, "Reducing numerical diffusion in interfacial gravity wave simulations", *International Journal for Numerical Methods in Fluids*, vol. 49, no. 3, pp. 301–329, 2005. DOI: 10.1002/fld.993.

[12] A. Sabatés, *Shallow Water Model Code for Applications to Atmosphere Dynamics (BSc. Thesis)*. Universitat Politècnica de Catalunya, 2018.

[13] E. Hairer, S. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations I: Nonstiff Problems*, 2nd ed. Berlin: Springer Verlag, 1993, ISBN: 9783540566700.

[14] A. Chambolle, "An Algorithm for Total Variation Minimization and Applications", *Journal of Mathematical Imaging and Vision*, vol. 20, no. 1-2, pp. 89–97, 2004. DOI: 10.1023/B:JMIV.0000011321.19549.88.

[15] P. Sweby, "High Resolution Schemes using Flux Limiters for Hyperbolic Conservation Laws", *SIAM Journal on Numerical Analysis*, vol. 21, no. 5, pp. 995–1011, 1984.

[16] F. S. Lien and M. A. Leschziner, "Upstream monotonic interpolation for scalar transport with application to complex turbulent flows", *International Journal for Numerical Methods in Fluids*, vol. 19, no. 6, pp. 527–548, 1994. DOI: 10.1002/fld.1650190606.

[17] D. Zhang, C. Jiang, D. Liang, and L. Cheng, "A review on TVD schemes and a refined flux-limiter for steady-state calculations", *Journal of Computational Physics*, vol. 302, pp. 114–154, 2015. DOI: 10.1016/j.jcp.2015.08.042.

[18] T. del Río-Gaztelurrutia, A. Sánchez-Lavega, A. Antuñano, J. Legarreta, E. García-Melendo, K. M. Sayanagi, R. Hueso, M. H. Wong, S. Pérez-Hoyos, J. F. Rojas, A. A. Simon, I. de Pater, J. Blalock, and T. Barry, "A planetary-scale disturbance in a long living three vortex coupled system in Saturn's atmosphere", *Icarus*, vol. 302, pp. 499–513, 2018. DOI: 10.1016/j.icarus.2017.11.029.

[19] A. Sánchez-Lavega, E. García-Melendo, J. Legarreta, R. Hueso, T. Del Río-Gaztelurrutia, J. F. Sanz-Requena, S. Pérez-Hoyos, A. A. Simon, M. H. Wong, M. Soria, J. M. Gómez-Forrellad, T. Barry, M. Delcroix, K. M. Sayanagi, J. Blalock, J. Gunnarson, U. Dyudiana, and S. Ewald, "A complex storm system in Saturn's north polar atmosphere in 2018", *Nature Astronomy*, vol. 4, pp. 180–187, 2020. DOI: 10.1038/s41550-019-0914-9.

[20] T. Liu, B. Wang, and D. S. Choi, "Flow structures of Jupiter's Great Red Spot extracted by using optical flow method", *Physics of Fluids*, vol. 24, no. 9, 2012. DOI: 10.1063/1.4752227.

[21] T. E. Dowling, A. S. Fischer, P. J. Gierasch, J. Harrington, R. P. Lebeau, and C. M. Santori, "The Explicit Planetary Isentropic-Coordinate (EPIC) Atmospheric Model", *Icarus*, vol. 132, no. 2, pp. 221–238, 1998. DOI: 10.1006/icar.1998.5917.

[22] R. LeBeau, "Simulations of Time-Dependent Three-Dimensional Vortices with Application to Neptune's Great Dark Spot", PhD thesis, Massachusetts Institute of Technology, 1997, p. 205.

[23] A. Semechko, *S2-Sampling-Toolbox*, 2018. [Online]. Available: https://github.com/AntonSemechko/S2-Sampling-Toolbox (visited on 05/20/2020).

[24] B. Hofmann-Wellenhof, K. Legat, and M. Wieser, *Navigation: Principles of Positioning and Guidance*, 1st ed. Wien: Springer Verlag, 2003, ISBN: 9783211008287.

[25] A. Modave, É. Deleersnijder, and É. J. Delhez, "On the parameters of absorbing layers for shallow water models", *Ocean Dynamics*, vol. 60, no. 1, pp. 65–79, 2010. DOI: 10.1007/s10236-009-0243-0.

[26] S. R. Brueshaber, K. M. Sayanagi, and T. E. Dowling, "Dynamical regimes of giant planet polar vortices", *Icarus*, vol. 323, no. February, pp. 46–61, 2019. DOI: 10.1016/j.icarus.2019.02.001.

[27] A. Hollingsworth, P. Kållberg, V. Renner, and D. M. Burridge, "An internal symmetric computational instability", *Quarterly Journal of the Royal Meteorological Society*, vol. 109, no. 460, pp. 417–428, 1983. DOI: 10.1002/qj.49710946012.

[28] P. S. Peixoto, J. Thuburn, and M. J. Bell, "Numerical instabilities of spherical shallow-water models considering small equivalent depths", *Quarterly Journal of the Royal Meteorological Society*, vol. 144, no. 710, pp. 156–171, 2018. DOI: 10.1002/qj.3191.

[29] *Space Images — Deep Motion*. [Online]. Available: https://www.jpl.nasa.gov/spaceimages/details.php?id=PIA23444 (visited on 05/25/2020).

[30] *Space Images — A New View on Jupiter's North Pole*. [Online]. Available: https://www.jpl.nasa.gov/spaceimages/details.php?id=PIA22336 (visited on 05/25/2020).

[31] J. Kim and P. Moin, "Application of a fractional-step method to incompressible Navier-Stokes equations", *Journal of Computational Physics*, vol. 59, no. 2, pp. 308–323, Jun. 1985. DOI: 10.1016/0021-9991(85)90148-2.

[32] A. J. Chorin and J. E. Marsden, *A mathematical introduction to fluid mechanics.* 3rd ed. New York: Springer Verlag, 1979, ISBN: 0387904069.

[33] K. F. Riley, M. Hobson, and S. Bence, *Mathematical Methods For Physics and Engineering*, 3rd. New York: Cambridge University Press, 2006, ISBN: 9780521861533.

[34] R. W. Verstappen and A. E. Veldman, "Spectro-consistent discretization of Navier-Stokes: A challenge to RANS and LES", *Journal of Engineering Mathematics*, vol. 34, no. 1-2, pp. 163–179, 1998. DOI: 10.1007/978-94-017-1564-5{_}10.

[35] ——, "Symmetry-preserving discretization of turbulent flow", *Journal of Computational Physics*, vol. 187, no. 1, pp. 343–368, 2003. DOI: 10.1016/S0021-9991(03)00126-8.

[36] M. Soria, C. D. Pérez-Segarra, and A. Oliva, "A Direct Schur-Fourier Decomposition For The Solution Of The Three-Dimensional Poisson Equation Of Incompressible Flow Problems Using Loosely Coupled Parallel Computers", *Numerical Heat Transfer, Part B: Fundamentals*, vol. 43, no. 5, pp. 467–488, 2003. DOI: 10.1080/713836244.

[37] D. J. Tritton, *Physical Fluid Dynamics*, 2nd ed. New York: Oxford University Press, 1977, ISBN: 9780198544937.

[38] R. Courant, K. Friedrichs, and H. Lewy, "Über die partiellen differenzengleichungen der matematischen physik", *Mathematische Annalen*, vol. 43, pp. 32–74, 1928. DOI: 10.1007/BF01448839.

[39] J. G. Charney, R. FjÖrtoft, and J. V. Neumann, "Numerical Integration of the Barotropic Vorticity Equation", *Tellus*, vol. 2, no. 4, pp. 237–254, 1950. DOI: 10.3402/tellusa.v2i4.8607.

[40] D. R. Durran and A. Arakawa, "Generalizing the Boussinesq approximation to stratified compressible flow", *Comptes Rendus - Mecanique*, vol. 335, no. 9-10, pp. 655–664, 2007. DOI: 10.1016/j.crme.2007.08.010.

[41] R. Hueso and A. Sánchez-Lavega, "A Three-Dimensional Model of Moist Convection for the Giant Planets: The Jupiter Case", *Icarus*, vol. 151, no. 2, pp. 257–274, 2001. DOI: 10.1006/icar.2000.6606.

[42] G. Taylor and B. Green, "Mechanism of the production of small eddies from large ones", *Proceedings of the Royal Society of London. Series A - Mathematical and Physical Sciences.*, vol. 158, no. 895, pp. 499–521, 1937. DOI: 10.1098/rspa.1937.0036.

[43] J. R. DeBonis, "Solutions of the Taylor-Green vortex problem using high-resolution explicit finite difference methods", NASA, Tech. Rep. February, 2013.

[44] V. Kolář, "Vortex identification : New requirements and limitations", *International Journal of Heat and Fluid Flow*, vol. 28, pp. 638–652, 2007. DOI: 10.1016/j.ijheatfluidflow.2007.03.004.

[45] D. Duran, *Study of the boundary layer flow control using synthetic jets by means of spectro-consistent discretizations (MSc. Thesis)*. Universitat Politècnica de Catalunya, 2017.

[46] H. Bénard, "Les tourbillons cellulaires dans une nappe liquide. - Méthodes optiques d'observation et d'enregistrement", *Journal de Physique Théorique et Appliquée*, vol. 10, no. 1, pp. 254–266, 1901. DOI: 10.1051/jphystap:0190100100025400.

[47] L. Rayleigh, "On convection currents in a horizontal layer of fluid, when the higher temperature is on the under side", *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 32, no. 192, pp. 529–546, 1916. DOI: 10.1080/14786441608635602.

[48] F. Dabbagh, F. X. Trias, A. Gorobets, and A. Oliva, "On the evolution of flow topology in turbulent Rayleigh-Bénard convection", *Physics of Fluids*, vol. 28, no. 11, 2016. DOI: 10.1063/1.4967495.

[49] K. G. Hollands, G. D. Raithby, and L. Konicek, "Correlation equations for free convection heat transfer in horizontal layers of air and water", *International Journal of Heat and Mass Transfer*, vol. 18, no. 7-8, pp. 879–884, 1975. DOI: 10.1016/0017-9310(75)90179-9.

[50] A. Robert, "Bubble convection experiments with a semi-implicit formulation of the Euler equations", *Journal of the Atmospheric Sciences*, vol. 50, no. 13, pp. 1865–1873, 1993. DOI: 10.1175/1520-0469(1993)050<1865:BCEWAS>2.0.CO;2.

[51] J. E. Guerra and P. A. Ullrich, "A high-order staggered finite-element vertical discretization for non-hydrostatic atmospheric models", *Geoscientific Model Development*, vol. 9, no. 5, pp. 2007–2029, 2016. DOI: 10.5194/gmd-9-2007-2016.

[52] C. Li and X. Chen, "Simulating Nonhydrostatic Atmospheres on Planets (SNAP): Formulation, Validation, and Application to the Jovian Atmosphere", *The Astrophysical Journal Supplement Series*, vol. 240, no. 2, p. 37, 2019. DOI: 10.3847/1538-4365/aafdaa.

[53] P. Grinfeld, *Introduction to Tensor Analysis and the Calculus of Moving Surfaces*, 1st ed. New York: Springer, 2013, ISBN: 9781461478669.

[54] A. A. White, A. Staniforth, and N. Wood, "Spheroidal coordinate systems for modelling global atmospheres", *Quarterly Journal of the Royal Meteorological Society*, vol. 134, pp. 261–270, 2008. DOI: 10.1002/qj.208.

[55] J. M. Vedovoto, A. d. Silveira Neto, A. Mura, and L. F. Figueira da Silva, "Application of the method of manufactured solutions to the verification of a pressure-based finite-volume numerical scheme", *Computers and Fluids*, vol. 51, no. 1, pp. 85–99, 2011. DOI: 10.1016/j.compfluid.2011.07.014.

[56] C. J. Roy, C. C. Nelson, T. M. Smith, and C. C. Ober, "Verification of Euler/Navier-Stokes codes using the method of manufactured solutions", *International Journal for Numerical Methods in Fluids*, vol. 44, no. 6, pp. 599–620, 2004. DOI: 10.1002/fld.660.

[57] N. Bourbaki, *Topological Vector Spaces: Chapters 1-5*. Paris: Springer, 1981, ISBN: 9783540423386.

[58] G. Strang, *Computational Science and Engineering*, 1st ed. Wellesley: Wellesley-Cambridge Press, 2007, ISBN: 0961408812.

[59] P. N. Swarztrauber, "The Methods of Cyclic Reduction, Fourier Analysis and the FACR Algorithm for the Discrete Solution of Poisson's Equation on a Rectangle", *SIAM Review*, vol. 19, no. 3, pp. 490–501, 1977.

[60] J. W. Cooley, P. A. Lewis, and P. D. Welch, "The fast Fourier transform algorithm: Programming considerations in the calculation of sine, cosine and Laplace transforms", *Journal of Sound and Vibration*, vol. 12, no. 3, pp. 315–337, 1970. DOI: 10.1016/0022-460X(70)90075-1.

[61] R. B. Wilhelmson and J. H. Ericksen, "Direct solutions for Poisson's equation in three dimensions", *Journal of Computational Physics*, vol. 25, no. 4, pp. 319–331, 1977. DOI: 10.1016/0021-9991(77)90001-8.

[62] S. Patankar, *Numerical Heat Transfer and Fluid Flow*. Taylor&Francis, 1980, ISBN: 0-89116-522-3.

Part III

Appendices

A — Vector operators in ellipsoidal coordinates

In this part, the different vector operators for ellipsoidal coordinates that are needed throughout the project will be deduced, since it is not a common system. The tools that have to be used are based on tensor calculus and will be extracted from [53]. The first thing to be noticed, though, is that to treat with ellipsoidal coordinates there are two options: planetocentric and planetographic latitudes. The former are more intuitive since the radius vector goes through the center of the ellipsoid, but it is not an orthogonal coordinate system which would complicate the derivation of these operators. Hence, planetogrpahic coordinates will be used and, if needed, the corresponding transformation will be performed to obtain planetocentric values.

In Figure 1.2 an scheme of planetographic ellipsoidal coordinates can be observed alongside the Cartesian system, since some transformations between those two will be needed.

The radius as a function of the planetocentric latitude and the relation between planetocentric and planetographic latitudes can be extracted from [12]. These are Eq. (A.1) and Eq. (A.2).

$$\rho(\varphi_c) = \frac{R_e R_p}{\sqrt{R_e^2 \sin^2(\varphi_c) + R_p^2 \cos^2(\varphi_c)}} \tag{A.1}$$

$$\tan(\varphi) = \frac{R_e^2}{R_p^2} \tan(\varphi_c) \tag{A.2}$$

Then, it is important to obtain the expression of the radius as a function of the planetographic latitude. Looking at Eq. (A.2) one would define $\gamma = \frac{R_p^2}{R_e^2} \tan(\varphi)$ and use some trigonometric identities such as

$$\cos(\arctan(\gamma)) = \frac{1}{\sqrt{\gamma^2 + 1}}$$

and also

$$\sin(\arctan(\gamma)) = \frac{\gamma}{\sqrt{\gamma^2 + 1}}$$

to get a new expression only dependent on the planetogrpahic latitude that reads

$$\rho(\varphi) = \frac{R_e R_p}{\sqrt{R_e^2 \dfrac{\gamma^2}{\gamma^2+1} + R_p^2 \dfrac{1}{\gamma^2+1}}} = \frac{R_e R_p \sqrt{\gamma^2+1}}{\sqrt{R_e^2\gamma^2 + R_p^2}} = \frac{R_e R_p \sqrt{\left(\dfrac{R_p^2}{R_e^2}\right)^2 \tan^2(\varphi) + 1}}{\sqrt{R_e^2 \left(\dfrac{R_p^2}{R_e^2}\right)^2 \tan^2(\varphi) + R_p^2}}$$

After some manipulation the expression can be reduced to Eq. (A.3).

$$\rho(\varphi) = \sqrt{\frac{R_p^4 \tan^2(\varphi) + R_e^4}{R_p^2 \tan^2(\varphi) + R_e^2}} \tag{A.3}$$

From the scheme presented in Figure 1.3 it's clear that

$$\begin{aligned} x &= \rho \cos(\varphi_c) \cos(\theta) \\ y &= \rho \cos(\varphi_c) \sin(\theta) \\ z &= \rho \sin(\varphi_c) \end{aligned}$$

Performing these products and simplifying the terms the result is Eq. (A.4).

$$x = \xi(\varphi) \cos(\theta) \tag{A.4a}$$
$$y = \xi(\varphi) \sin(\theta) \tag{A.4b}$$
$$z = \zeta(\varphi) \tag{A.4c}$$

where

$$\xi(\varphi) = \frac{R_e^2}{\sqrt{R_p^2 \tan^2(\varphi) + R_e^2}} \tag{A.5a}$$
$$\zeta(\varphi) = \frac{R_p^2 \tan(\varphi)}{\sqrt{R_p^2 \tan^2(\varphi) + R_e^2}} \tag{A.5b}$$

With all this information regarding the change between both coordinate systems it is time to actually compute the vector operators for a revolution ellipsoid. The first expression that must be employed is the divergence for any general coordinate system that can be done by means of Voss-Weyl's formula as written in Eq. (A.6).

$$\nabla_i F^i = \frac{1}{\sqrt{|g|}} \partial_i \left(\sqrt{\frac{|g|}{g_{ii}}} \hat{F}^i \right) \tag{A.6}$$

In this case, g stands for the metric tensor, ∂_i for the partial derivative with respect to the *i-th* variable and F^i for the *i-th* component of the vector. It is then obvious that the metric tensor must be computed and the easiest way to do so will be using the jacobian matrix because

$$g_{ij} = J^T J \tag{A.7}$$

Recall that the Jacobian matrix is defined as

$$J = \begin{bmatrix} \dfrac{\partial x}{\partial \theta} & \dfrac{\partial x}{\partial \varphi} \\ \dfrac{\partial y}{\partial \theta} & \dfrac{\partial y}{\partial \varphi} \\ \dfrac{\partial z}{\partial \theta} & \dfrac{\partial z}{\partial \varphi} \end{bmatrix} \tag{A.8}$$

The different terms are listed below and are easily deduced from Eq. (A.4).

$$\frac{\partial x}{\partial \theta} = -\xi(\varphi)\sin(\theta) \qquad \frac{\partial y}{\partial \theta} = \xi(\varphi)\cos(\theta) \qquad \frac{\partial z}{\partial \theta} = 0$$

$$\frac{\partial x}{\partial \varphi} = \partial_\varphi \xi(\varphi)\cos(\theta) \quad \frac{\partial y}{\partial \varphi} = \partial_\varphi \xi(\varphi)\sin(\theta) \quad \frac{\partial z}{\partial \varphi} = \partial_\varphi \zeta(\varphi)$$

Then, performing the matrix multiplication, the covariant metric tensor is obtained

$$g_{ij} = \begin{bmatrix} \xi^2(\varphi) & 0 \\ 0 & (\partial_\varphi \xi(\varphi))^2 + (\partial_\varphi \zeta(\varphi))^2 \end{bmatrix} = \begin{bmatrix} r_Z^2(\varphi) & 0 \\ 0 & r_M^2(\varphi) \end{bmatrix} \tag{A.9}$$

It is worth pointing out that since these are orthogonal coordinates, the covariant matrix tensor is diagonal which makes it simpler to work with. Also, a new notation has been introduced: $r_Z(\varphi)$ stands for the zonal radius and $r_M(\varphi)$ for the meridional radius. These two quantities are commonly seen in literature and, hence, they will be used from now on. Its expressions can be obtained after performing the corresponding operations, and are

$$r_Z(\varphi) = \frac{R_e^2}{\sqrt{R_p^2 \tan^2(\varphi) + R_e^2}} \tag{A.10a}$$

$$r_M(\varphi) = \frac{r_Z(\varphi)/\cos(\varphi)}{\sin^2(\varphi) + \left(\frac{R_e}{R_p}\right)^2 \cos^2(\varphi)} \tag{A.10b}$$

Then, applying the definition of the divergence from Eq. (A.6) with the covariant metric tensor

$$\nabla \cdot \vec{F} = \frac{1}{r_Z(\varphi) r_M(\varphi)} \left[\frac{\partial}{\partial \theta} \left(r_M(\varphi) F_\theta \right) + \frac{\partial}{\partial \varphi} \left(r_Z(\varphi) F_\varphi \right) \right]$$

$$\nabla \cdot \vec{F} = \frac{1}{r_Z(\varphi)} \frac{\partial F_\theta}{\partial \theta} + \frac{1}{r_M(\varphi)} \frac{\partial F_\varphi}{\partial \varphi} + \frac{\partial_\varphi r_Z(\varphi)}{r_Z(\varphi) r_M(\varphi)} F_\varphi$$

And computing $\partial_\varphi r_Z(\varphi)$, the divergence in this oblate spheroidal coordinates is obtained and equals

$$\nabla \cdot \vec{F} = \frac{1}{r_Z(\varphi)} \frac{\partial F_\theta}{\partial \theta} + \frac{1}{r_M(\varphi)} \frac{\partial F_\varphi}{\partial \varphi} - \frac{\sin(\varphi)}{r_Z(\varphi)} F_\varphi \tag{A.11}$$

To compute the gradient operator, Eq. (A.12), extracted from [6], will be used. This equation shows the gradient operator for any general orthogonal curvilinear coordinates as a function of the scale parameters h_i of the metric.

$$\nabla F = \frac{1}{h_i} \partial_i F \tag{A.12}$$

Since for curvilinear orthogonal coordinates the relationship between the metric tensor and the scale factors is defined as $g_{ij} = h_i^2 \delta_{ij}$, the gradient is straightforward to compute and is shown in Eq. (A.13).

$$\nabla F = \frac{1}{r_Z(\varphi)} \frac{\partial F}{\partial \theta} + \frac{1}{r_M(\varphi)} \frac{\partial F}{\partial \varphi} \tag{A.13}$$

Finally, the last operator to be computed is the laplacian. In [53] it is specified a very easy way to compute it. This is the so-called Laplace-Beltrami operator and is defined as

$$\Delta F = \frac{1}{\sqrt{|g|}}\partial_i\left(\sqrt{|g|}g^{ij}\partial_j F\right) \tag{A.14}$$

which applied to the ellipsoidal coordinates one gets

$$\Delta F = \frac{1}{r_Z(\varphi)r_M(\varphi)}\frac{\partial}{\partial\theta}\left(\frac{r_M(\varphi)}{r_Z(\varphi)}\frac{\partial F}{\partial\theta}\right) + \frac{1}{r_Z(\varphi)r_M(\varphi)}\frac{\partial}{\partial\varphi}\left(\frac{r_Z(\varphi)}{r_M(\varphi)}\frac{\partial F}{\partial\varphi}\right) \tag{A.15}$$

$$\Delta F = \frac{1}{r_Z^2(\varphi)}\frac{\partial^2 F}{\partial\theta^2} + \frac{1}{r_Z(\varphi)r_M(\varphi)}\frac{\partial}{\partial\varphi}\left(\frac{r_Z(\varphi)}{r_M(\varphi)}\frac{\partial F}{\partial\varphi}\right) \tag{A.16}$$

Moreover, a final result must be presented. This is the advection in ellipsoidal coordinates. The way in which this operator can be found is by means of the identity $(\vec{u}\cdot\nabla)\vec{u} = \nabla(\vec{u}\cdot\vec{u}/2) - \vec{u}\times(\nabla\times\vec{u})$. This procedure, though, is very tedious and, therefore, the result will be obtained adapting the general case of curvilinear coordinates given in [7], [54] for the ellipsoidal ones. Then, the convection operator is

$$(\vec{u}\cdot\nabla)\vec{F} = \begin{bmatrix} \frac{u_1}{r_Z(\varphi)}\frac{\partial F_\theta}{\partial\theta} + \frac{u_2}{r_M(\varphi)}\frac{\partial F_\theta}{\partial\varphi} - \frac{\sin(\varphi)}{r_Z(\varphi)}u_1u_2 \\ \frac{u_1}{r_Z(\varphi)}\frac{\partial F_\varphi}{\partial\theta} + \frac{u_2}{r_M(\varphi)}\frac{\partial F_\varphi}{\partial\varphi} + \frac{\sin(\varphi)}{r_Z(\varphi)}u_1u_1 \end{bmatrix} \tag{A.17}$$

B — Validation of the SW code

In this appendix a proper validation of the code will be done. This process is of major importance, because there are plenty of details that must be taken into account. The code will be written in *MATLAB*® to test the algorithm and to perform the convergence analysis of each term individually. In this case, the Cartesian set of equations will be solved.

The validation for the ellipsoidal coordinates will not be carried out since the equations are formally equivalent, and the algorithm is what we are interested in assuring its correctly functioning.

B.1 Method of manufactured solutions

To validate the code one of the best options is to use the Method of Manufactured Solutions (MMS). This test consists, as its name indicates, on making any function to be a solution of a particular PDE. The way in which this is achieved is with the addition of a source term that will help maintain the equality. Consider the PDE shown in Eq. (B.1), where $\mathcal{D}$ stands for any general differential operator.

$$\mathcal{D}f = 0 \tag{B.1}$$

Since we have no analytical solution for function f, we create one that satisfies, at least, the boundary conditions. Let's call this function f_M where the subscript M denotes that is a manufactured function. Once this is done, we apply the differential operator $\mathcal{D}$, and since f_M is not a solution of Eq. (B.1) the result will not be equal to 0 but to some function that will be called F_M.

$$\mathcal{D}f_M = F_M \tag{B.2}$$

Then, to actually test if the code is able to solve for f, the PDE that will end up solving is

$$\mathcal{D}f - F_M = 0 \tag{B.3}$$

This way, the solution that must output the code has to be, without any kind of doubt, the manufactured one, f_M. This technique of adding a source term to keep the equality unchanged and knowing *a priori* the solution of the problem is extensively used to validate codes for CFD applications [55], [56].

To perform the convergence plot, it must be ensured that the norm of the error tends to zero as the mesh is refined. Any scheme should satisfy this condition to be considered as consistent, but depending on how fast it does so, the order of convergence will be obtained. To be more precise, the discrete solution f_n is said to converge to an actual solution S with order p if a constant K can be found so as to ensure that

$$|f_n - S| < Kh^p$$

being h a measure of the space between grid points. Then, the value of p can easily be found in a *log-log* plot because it will be the slope of the fitted line. Finally, the norm must be chosen. Recall that the definition of the norm is a mapping $x \mapsto ||x||$ that fulfills the following conditions [57]:

- $||x|| = 0 \iff x = 0$
- $||\lambda x|| = |\lambda| \, ||x|| \quad \forall \lambda \in \mathbb{R}$
- $||x + y|| \leq ||x|| + ||y||$

Then, there are plenty of mappings that can be considered to be a norm and some of the most important ones are the so-called *p-norm*. They are denoted as $|| \cdot ||_p$ and for the present work, also $L^p(\cdot)$ -not to be confused with a Lebesgue space-, and its definition is

$$||x||_p = \left(\sum_{\forall i} |x_i|^p \right)^{1/p} \qquad p \in [1, \infty), \; x \in \mathbb{R}^n \tag{B.4}$$

The most used norms are

$$L^1(x) = ||x||_1 = |x_1| + |x_2| + ... + |x_n| \tag{B.5}$$
$$L^2(x) = ||x||_2 = \sqrt{|x_1|^2 + |x_2|^2 + ... + |x_n|^2} \tag{B.6}$$
$$L^\infty(x) = ||x||_\infty = \max_{\forall i \in [1,n]} |x_i| \tag{B.7}$$

The main problem with L^1 and L^2 is that the greater the sample is (i.e. as n grows) the bigger the norm and that's just because there are more points. This behaviour is not a desired property if what we are trying to see is that the solution tends to zeros as n grows. If we wanted to use this type of norms we should add a normalization to take into account the number of points involved, but this task has to be done in such a way that the convergence rate is not affected by the addition of the normalization. To avoid this problem, the best solution is to take the L^∞ norm to perform the convergence analysis.

To apply the MMS to the SW equations a manufactured solution must be created. Considering a full periodic case, an easy set of functions that will be the solution of the system can be, according to [12],

$$\begin{aligned} u_M &= \sin(2\pi x)\cos(2\pi y)t \\ v_M &= \cos(2\pi x)\sin(2\pi y)t \\ h_M &= 0.01\cos(2\pi x)\cos(2\pi y)\cos(t) + D \end{aligned} \tag{B.8}$$

where t will be set to a non zero constant for the convergence analysis of the advection and D will be a constant sufficiently small to keep the basic hypothesis of the model of $D \ll L$.

B.2 Validation of the advection terms

To validate the advective terms they will be split into the P_1 and P_2 and validate them separately. The first term of the advection for the x-axis momentum equation is

$$P_{1u} = \nabla \cdot (u\vec{u})$$

Then, the source term associated to it must be computed. Expanding the expression, we get

$$\nabla \cdot (u\vec{u}) = \frac{\partial(uu)}{\partial x} + \frac{\partial(uv)}{\partial y}$$

Which evaluated according to Eq. (B.8) leads to the source term for P_{1u} of

$$F_M^{1u} = t^2 \pi \sin(4\pi x)(2\cos(4\pi y) + 1) \tag{B.9}$$

Therefore, to correctly validate this term the numerical solution of the P_{1u} will be computed according to Eq. (2.10) and after dividing it by $-\Delta t$ and subtracting Eq. (B.9) from it, the vector field obtained should tend to 0 as the number of points in the mesh grows. The result for the convergence analysis of this term leads to the result shown in Figure B.1.

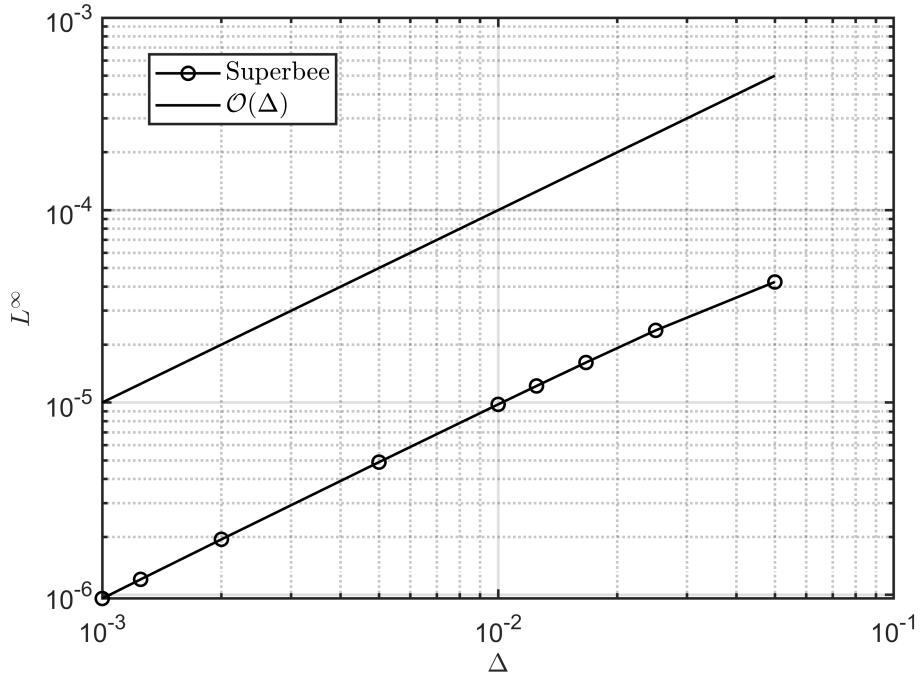


Figure B.1: Convergence plot for the term P_{1u} with $\Delta t = 10^{-3}$.

The results seem wrong since the order of convergence is $\mathcal{O}(\Delta)$ while the theory (i.e. [15]) predicts that it should be of second order. On the other hand, though, the results are in accordance with [3], [12]. It must be recalled that u is staggered in the x-axis and the theory predicts the second order convergence for a centered variable. Hence, it will be assumed that the convergence plot demonstrates a fine working of the algorithm and this second order will be compared for the variable η or h since it is the centered variable being used.

Moving on to the second term of the x-axis momentum equation

$$P_{2u} = u\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right)$$

the source term that will be needed is

$$F_M^{2u} = 4t^2 \pi \cos(2\pi x)\cos(2\pi y)^2 \sin(2\pi x) \tag{B.10}$$

Then, performing the MMS test for this term the result is the one displayed in Figure B.2. As expected, the scheme is second order accurate.

The next term to be analyzed is the P_1 related to the y-axis momentum equation. This term, which recall that is defined as

$$P_{1v} = \nabla \cdot (v\vec{u})$$

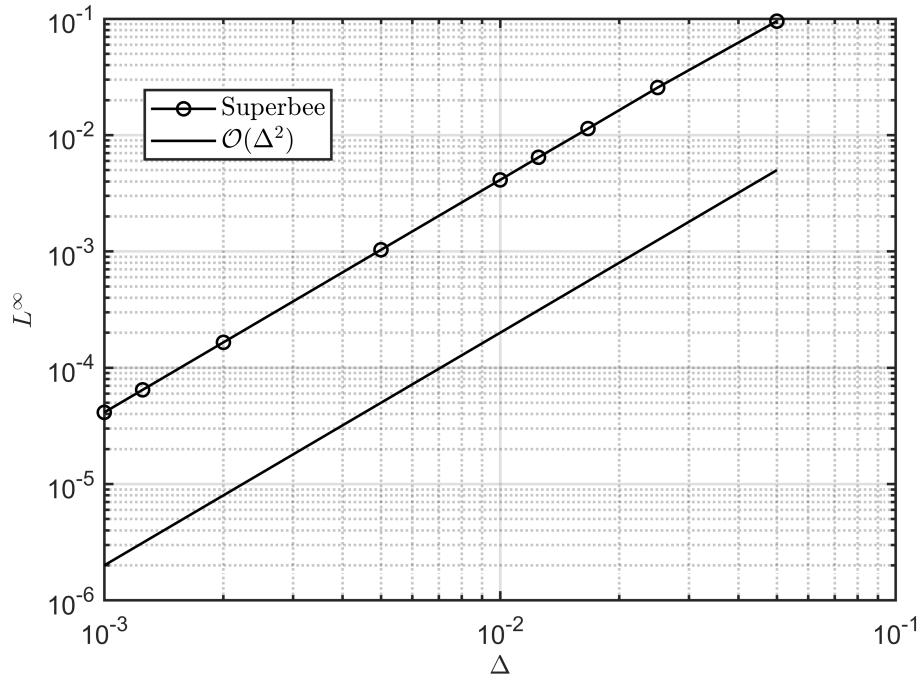


Figure B.2: Convergence plot for the term P_{2u} with $\Delta t = 10^{-3}$.

can be expanded into

$$\nabla \cdot (v\vec{u}) = \frac{\partial (vu)}{\partial x} + \frac{\partial (vv)}{\partial y}$$

Then, the source term can be computed using the manufactured solution of Eq. (B.8) and the result is

$$F_M^{1v} = t^2 \pi \sin(4\pi y)(2\cos(4\pi x) + 1) \tag{B.11}$$

which leads to the outcome of the MMS test shown in Figure B.3. Again, the result is that the order of convergence is just of first order, presumably due to the staggered nature of the variable being computed.

To end with the advection of the velocity, the second term related to the y-momentum equation must be tested. This term takes the form of

$$P_{2v} = v\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right)$$

which leads to a source term of

$$F_M^{2v} = 4t^2 \pi \cos(2\pi x)^2 \cos(2\pi y) \sin(2\pi y) \tag{B.12}$$

Then, the MMS test for this term leads to the result shown in Figure B.4. Again, this result indicates what we expected: a second order convergence.

To end with this section, the last term that has to be tested is the one related to the mass conservation equation. This term has a formulation very similar to the advection for an incompressible flow -and therefore can be written in its conservative form and apply the algorithm already shown-. The term that is being computed is

$$P_{1h} = \nabla \cdot (h\vec{u})$$

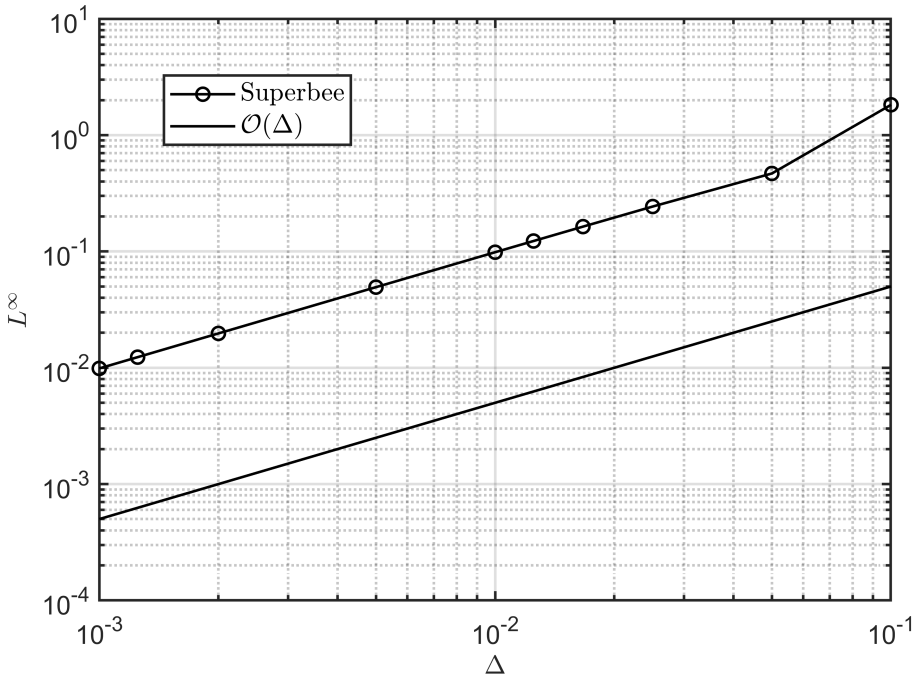


Figure B.3: Convergence plot for the term P_{1v} with $\Delta t = 10^{-3}$.

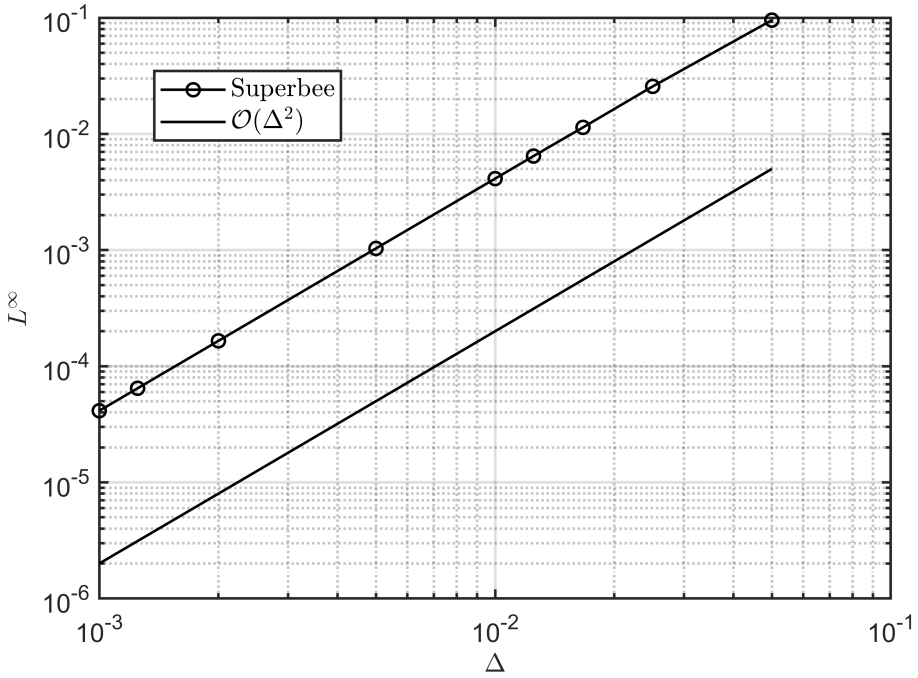


Figure B.4: Convergence plot for the term P_{2v} with $\Delta t = 10^{-3}$.

which means that the source term associated is

$$F_M^{1h} = -\frac{t\pi}{50}[\cos(t)\{\cos(2\pi x)^2 + \cos(2\pi y)^2 - 4\cos(2\pi x)^2\cos(2\pi y)^2\} - 200D\cos(2\pi x)\cos(2\pi y)] \tag{B.13}$$

The result for the MMS test performed for this term is shown in Figure B.5. As it can be stated, this time, the approximation is of second order, as Sweby showed, because this time the variable is centered and not staggered.

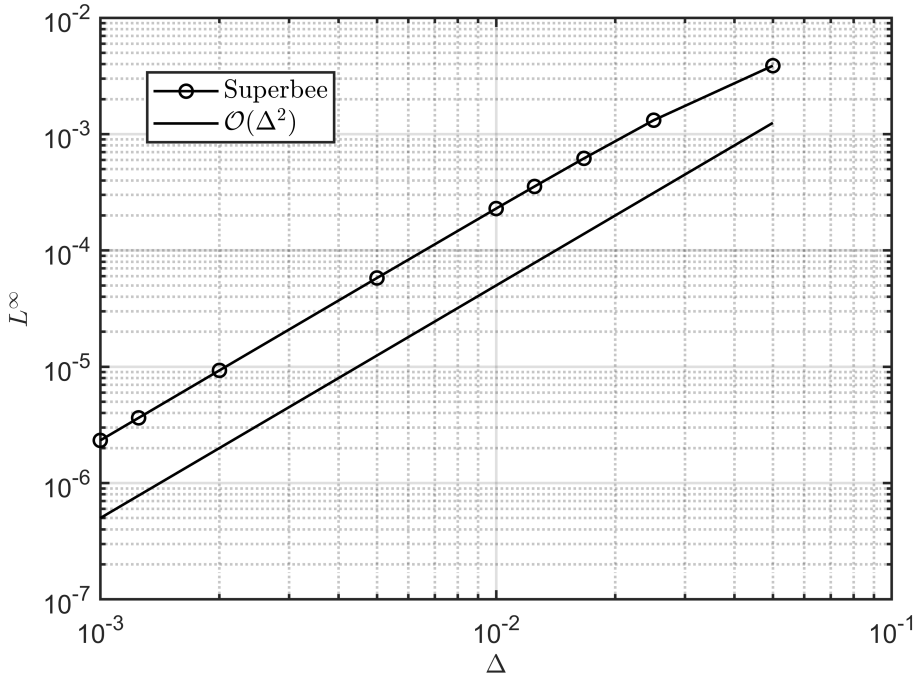


Figure B.5: Convergence plot for the term P_{1h} with $\Delta t = 10^{-3}$.

B.3 Validation of the AB integration scheme

To validate the time integration, the MMS test will be performed again. In this case, though, a small number of time-steps will be computed and the solution of the velocity field of the fluid elevation will be compared to the analytical solution. In this case the equation that will be integrated are only with the advection terms (leaving aside the Coriolis force and the pressure gradient contribution). The source terms that will be added for each of the equations are shown in Eq. (B.14) and Eq. (B.15).

$$\begin{aligned} F_M^{tu} &= \sin(2\pi x)(\cos(2\pi y) - 2t^2\pi\cos(2\pi x) + 4t^2\pi\cos(2\pi x)\cos(2\pi y)^2) \\ F_M^{tv} &= \sin(2\pi y)(\cos(2\pi x) - 2t^2\pi\cos(2\pi y) + 4t^2\pi\cos(2\pi x)^2\cos(2\pi y)) \end{aligned} \tag{B.14}$$

$$\begin{aligned} F_M^{th} =& \frac{1}{50}[t^2\pi\cos(2\pi x)^2\sin(2\pi y)\{100d\cos(2\pi y) - \cos(2\pi x)\cos(t) + 2\cos(2\pi x)\cos(2\pi y)^2\cos(t)\}] \\ &- \frac{1}{100}\cos(2\pi x)\cos(2\pi y)\sin(t) + 2t\pi\cos(2\pi x)\cos(2\pi y)[d + \frac{1}{100}\cos(2\pi x)\cos(2\pi y)\cos(t)] \\ &- \frac{1}{50}t\pi\cos(2\pi y)^2\sin(2\pi x)^2\cos(t) \end{aligned} \tag{B.15}$$

The results are displayed in Figure B.6 for the horizontal component of the velocity, in Figure B.7 for the vertical component and, finally, in Figure B.8 for the fluid column elevation.

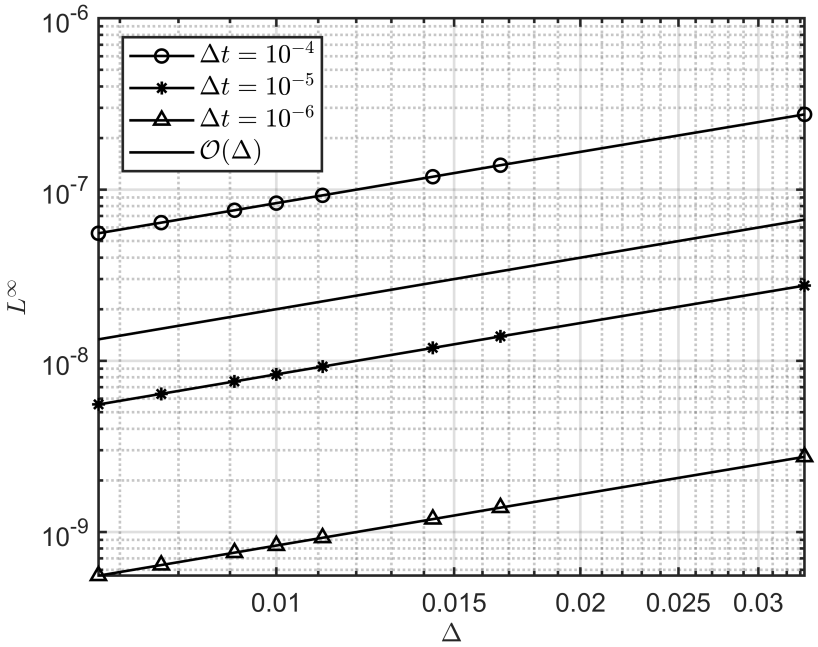


Figure B.6: Convergence plot for the term AB for u.

As it can be seen, the convergence order with respect to the spatial separation is the same as seen in the advection validation, but also it has been tested for different Δt. As it was expected the error decays as the time step goes to zero. Again, it can be noted that the convergence on Δt is not of third order. Again, this result is on agreement with those presented in [3], [12].

This behaviour is still not clear at all, but the presence of Δt in the algorithm to compute the advection might play an important role, as well as the staggering, since in h the convergence order is 2, while in the velocity terms, that are staggered, of 1. Since this results are similar to those carried out before under the same conditions it will be concluded a correct time integration.

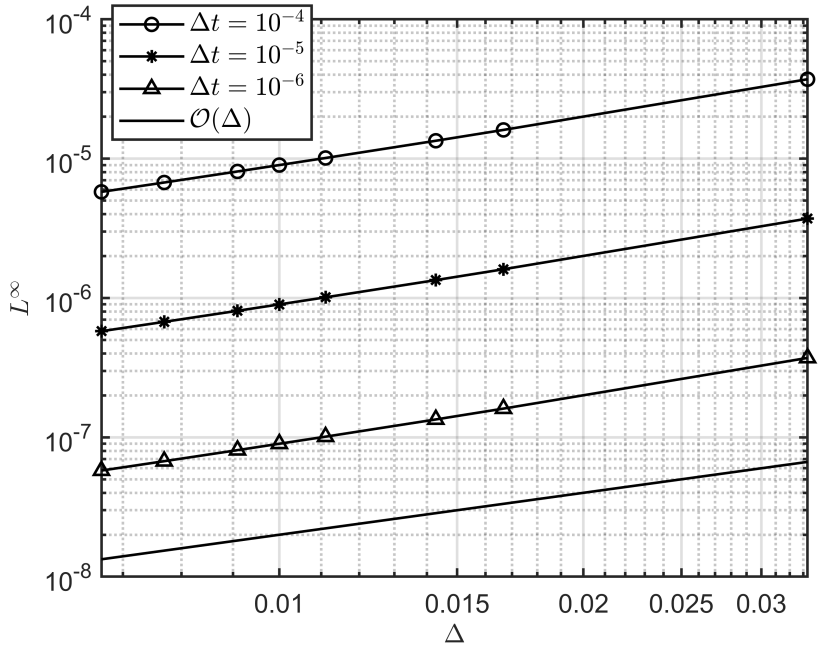


Figure B.7: Convergence plot for the term AB for v.

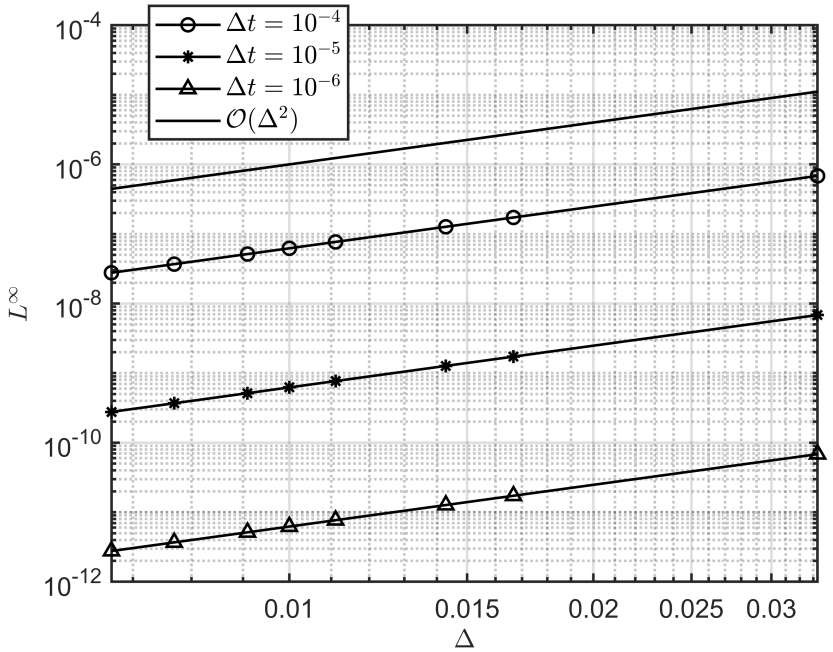


Figure B.8: Convergence plot for the term AB for h.

C — Fourier solver algorithm

In this appendix, the Fourier solver will be explained and applied for two different boundary conditions. Also, some functions will be described in a pseudo-code notation as well as the mathematical foundation of the whole algorithm.

Let's consider a 1D problem, which discretized Poisson equation leads to a problem like

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{C.1}$$

If the BC are of Dirichlet type the matrix $\mathbf{A}$ takes the form

$$\mathbf{A} = \begin{bmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix} \tag{C.2}$$

To solve this system the procedure described in [58] will be followed. First of all, a way to construct the two dimensional matrix from the one dimensional case is just by using the Kronecker product as

$$\mathbf{A}^{2d} = \mathbf{I}_N \otimes \mathbf{A} + \mathbf{A} \otimes \mathbf{I}_N \tag{C.3}$$

where $\mathbf{I}_N$ stands for the identity matrix of rank N. Similarly, the three dimensional one can be obtained performing all possible permutations leading to

$$\mathbf{A}^{3d} = \mathbf{I}_N \otimes \mathbf{I}_N \otimes \mathbf{A} + \mathbf{I}_N \otimes \mathbf{A} \otimes \mathbf{I}_N + \mathbf{A} \otimes \mathbf{I}_N \otimes \mathbf{I}_N \tag{C.4}$$

Taking $N = 5$, a visualization of the non-zero elements of these sparse matrices can be found in Figure C.1. The original $\mathbf{A}$ has an eigenvalue decomposition as

$$\mathbf{A} = \mathbf{S}\Lambda\mathbf{S}^{-1} \tag{C.5}$$

where the eigenvalue matrix Λ is diagonal and contains the eigenvalues of matrix $\mathbf{A}$ while $\mathbf{S}$ contains the associated eigenvectors in each column. The eigenvalues of $\mathbf{A}$ are

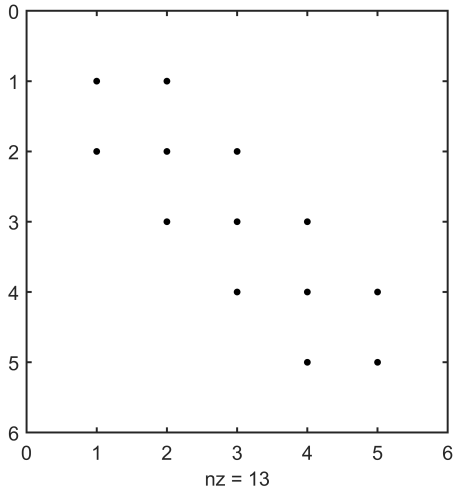
$$\Lambda_{jj} = 2 - 2\cos\left(\frac{\pi j}{N+1}\right) \tag{C.6}$$

and the normalized eigenvector matrix is just

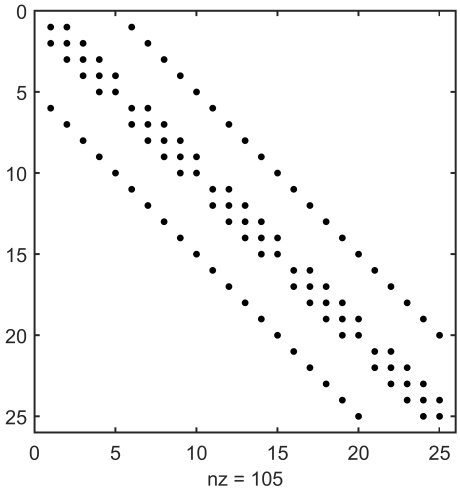
$$\mathbf{S}_{jk} = \sqrt{\frac{2}{N+1}}\sin\left(\frac{\pi jk}{N+1}\right) \tag{C.7}$$

Recall, though, that these expressions only hold for Dirichlet boundary conditions. If the BC are different then these formulae should be slightly modified.

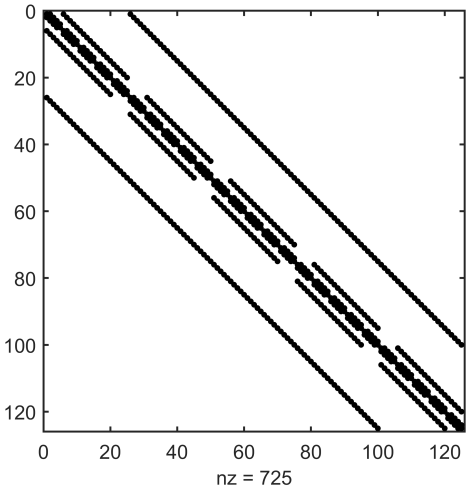
The general resolution algorithm after these decomposition is done will be as follows:



(a) $\mathbf{A}$



(b) $\mathbf{A}^{2d}$



(c) $\mathbf{A}^{3d}$

Figure C.1: Distribution of non-zero elements for the discrete Poisson equation for $N = 5$

1. Compute a new RHS as $\overline{\mathbf{b}} = \mathbf{S}^{-1}\mathbf{b}$. This will correspond to a change of the original space to the eigenspace.

2. Divide each $\bar{b}_j$ by λ_j. This corresponds to solving the system in the eigenspace as $\Lambda\overline{\mathbf{x}} = \overline{\mathbf{b}}$ which is very simple due to the fact that Λ is diagonal.

3. Obtain the final solution by $\mathbf{x} = \mathbf{S}\overline{\mathbf{x}}$. This implies returning the solution to the original space.

The steps that might slow down the process are the first and third, due to the product of a matrix and a vector. This operation is, normally, of a $\mathcal{O}(N^2)$ complexity. The idea is to realize that the Discrete Sine Transform (a particular case of the Discrete Fourier Transform) has a connection to the eigenvectors of $\mathbf{A}$. More precisely, they are related to the imaginary parts of the exponentials of the Fourier matrix. These exponentials are

$$w^k = \exp\left(\frac{i2\pi k}{M}\right) \tag{C.8}$$

where here $i = \sqrt{-1}$ instead of representing an index.

As reference [58] states, take $M = 2N + 2$ and then the sines that compose the eigenvectors of $\mathbf{A}$ will just be

$$\sin\left(\frac{jk\pi}{N+1}\right) = \mathrm{Im}\left[\exp\left(\frac{i\pi jk}{N+1}\right)\right] \tag{C.9}$$

Then, to perform a sine transform of the type $\mathbf{Sb}$ being $\mathbf{b} \in \mathbb{R}^N$, the procedure is to extend this last vector with zeros to another auxiliary vector $\mathbf{b}^\dagger \in \mathbb{R}^{2N+2}$. The first element should be a zero, then the actual vector, and the rest again zeros. In $MATLAB^{\textcircled{R}}$ notation this would be $\mathbf{b}^\dagger = [0; \mathbf{b}; \mathrm{zeros}(N{+}1, 1)]$. After that, another auxiliary vector must be created with the form of $\mathbf{b}^\ddagger = \mathcal{F}(\mathbf{b}^\dagger) \in \mathbb{R}^{2N+2}$. This step can be done using the FFT routines that will speed up the calculation and then recover the result by $\mathbf{Sb} = -\mathrm{Im}\left(\mathbf{b}^\ddagger_j\right) \quad \forall j \in [2, N+1]$. Recall that this expression lacks the factor $\sqrt{\frac{2}{N+1}}$ needed to normalize the eigenvectors.

Now that the idea in 1D is clear, its extension to higher dimensions should be performed. Let's start by the two dimensional case as described in [58]. The matrix $\mathbf{A}^{2d}$ will have N^2 separable eigenvectors which can be denoted as v_{kl}, being each component defined as

$$v_{ijkl} = \sin\left(\frac{ik\pi}{N+1}\right)\sin\left(\frac{jl\pi}{N+1}\right) \tag{C.10}$$

In the former expression, i, j are the components of each eigenvector v_{kl}. The eigenvalues for this case are

$$\lambda_{kl} = \left(2 - 2\cos\left(\frac{k\pi}{N+1}\right)\right) + \left(2 - 2\cos\left(\frac{l\pi}{N+1}\right)\right) \tag{C.11}$$

Notice that the eigenvalues correspond to the sum of each one-dimensional contribution, which are independent of each other. Also, the extension of the eigenspace representation is straightforward. For two dimensions, the vector $\mathbf{b}$ -which now will be useful to keep it in a matrix notation and where each component i, j has its correspondent one-dimensional value to order it as a vector- can be expressed as a linear combination of the eigenvectors as

$$\mathbf{b}_{i,j} = \sum_{\forall k}\sum_{\forall l} \bar{b}_{kl}\sin\left(\frac{ik\pi}{N+1}\right)\sin\left(\frac{jl\pi}{N+1}\right) \tag{C.12}$$

And then the final solution -again with this matrix notation- will be

$$\mathbf{x}_{i,j} = \sum_{\forall k}\sum_{\forall l} \frac{\overline{b}_{kl}}{\lambda_{kl}} \sin\left(\frac{ik\pi}{N+1}\right) \sin\left(\frac{jl\pi}{N+1}\right) \tag{C.13}$$

This procedure can be generalized to any dimensions. Having this base, some particular cases will be treated and its practical implementation presented.

C.1 Full periodic uniform three dimensional Poisson equation

In this section a test case of a full periodic 2D case will be discussed. The main idea in this part is to get a little bit practical and go into details of the implementation (except the FFT routines). Let's start by looking at the fundamental matrix $\mathbf{A}$ for this case, since it will be slightly different than from the one discussed before. As stated in [36], a general matrix for the periodic 1D Poisson equation has the form

$$\mathbf{A} = \begin{bmatrix} \beta & \alpha & & \alpha \\ \alpha & \beta & \ddots & \\ & \ddots & \ddots & \alpha \\ \alpha & & \alpha & \beta \end{bmatrix} \tag{C.14}$$

If we are to obtain the 3D matrix, we can do it by means of Eq. (C.4). Now, for this case the eigenvalues are a little bit different. Reference [36] gives the following values:

$$\begin{aligned} \lambda_1 &= \beta + 2\alpha \\ \lambda_{2\nu} = \lambda_{2\nu+1} &= -4\alpha \sin^2\left(\frac{\nu\pi}{N}\right) + \beta + 2\alpha \quad \forall \nu \in \left[1, \frac{N}{2} - 1\right] \\ \lambda_N &= \beta - 2\alpha \end{aligned} \tag{C.15}$$

The transformation to and from the eigenspace for this case was described by Swarztrauber [59]. The Fourier transform (equivalent to $\overline{\mathbf{b}} = \mathbf{S}^{-1}\mathbf{b}$) is

$$\begin{aligned} \overline{b}_1 &= \frac{2}{N} \sum_{\forall i \in [1,N]} b_i \\ \overline{b}_{2\nu} &= \frac{2}{N} \sum_{\forall i \in [1,N]} b_i \cos\left(\frac{\nu i 2\pi}{N}\right) \quad \forall \nu \in \left[1, \frac{N}{2} - 1\right] \\ \overline{b}_{2\nu+1} &= \frac{2}{N} \sum_{\forall i \in [1,N]} b_i \sin\left(\frac{\nu i 2\pi}{N}\right) \quad \forall \nu \in \left[1, \frac{N}{2} - 1\right] \\ \overline{b}_N &= \frac{2}{N} \sum_{\forall i \in [1,N]} b_i (-1)^i \end{aligned} \tag{C.16}$$

The inverse Fourier transform (equivalent to $\mathbf{x} = \mathbf{S}\overline{\mathbf{x}}$) is given by

$$x_i = \frac{1}{2}\overline{x}_i + \sum_{\forall \nu \in [1, N/2-1]} \left[\overline{x}_{2\nu} \cos\left(\frac{\nu i 2\pi}{N}\right) + \overline{x}_{2\nu+1} \sin\left(\frac{\nu i 2\pi}{N}\right) \right] + \frac{1}{2}\overline{x}_n (-1)^i \tag{C.17}$$

As it was stated before, this operation has a count number of operations that grow with $\mathcal{O}(N^2)$ and hence, the practical implementation goes through the usage of FFT routines. What most of literature lacks, though, is the explanation on how this step has to be performed and thus, it will be of interest summing up how it is done. Swarztrauber [59] gives the pre- and postprocessing necessary to use the FFT routines although not for this particular case and refers to other papers for this information, but in those references the exact procedure is still not clear enough for a practical implementation. Therefore, a mixed approach between the theoretical explanation presented in [60], [61] (which still do not expose a clear way to do so) and the implemented and validated version used in [36] will be explained.

C.1.1 Computation of the Fourier transform using FFT routines

The main idea is that given a vector $\mathbf{b}$, the product $\overline{\mathbf{b}} = \mathbf{S}^{-1}\mathbf{b}$ is performed in a rapid way. Then, let us define an auxiliary vector $\mathbf{b}^\dagger$. The definition of this auxiliary vector will be

$$\mathbf{b}^\dagger = \frac{2}{N}\overline{\mathcal{F}(\mathbf{b})} \tag{C.18}$$

where $\overline{z}$ denotes the complex conjugate of z. Note that in this case the discrete Fourier transform $\mathcal{F}(\cdot)$ will be exactly the result of the FFT algorithm. Then, the resulting vector will be modified as follows

$$b_j^\dagger = b_j^\dagger \left[\cos\left(\frac{2\pi(j-1)}{N}\right) + i \sin\left(\frac{2\pi(j-1)}{N}\right) \right] \qquad \forall j \in [1, N] \tag{C.19}$$

where i denotes the imaginary unit $\sqrt{-1}$.

Finally, the resulting vector, $\overline{\mathbf{b}}$, will be computed as

$$\begin{aligned} \overline{b}_1 &= \mathrm{Re}(b_1^\dagger) \\ \overline{b}_{2\nu} &= \mathrm{Re}(b_{\nu+1}^\dagger) \qquad \forall \nu \in \left[1, \frac{N}{2} - 1\right] \\ \overline{b}_{2\nu+1} &= \mathrm{Im}(b_{\nu+1}^\dagger) \qquad \forall \nu \in \left[1, \frac{N}{2} - 1\right] \\ \overline{b}_N &= \mathrm{Re}(b_{N/2+1}^\dagger) \end{aligned} \tag{C.20}$$

This modification is only a $\mathcal{O}(N)$ operation and then, the most restrictive operation count is the actual FFT routine.

C.1.2 Computation of the inverse Fourier transform using FFT routines

The main idea is that given a vector $\overline{\mathbf{x}}$, the product $\mathbf{x} = \mathbf{S}\overline{\mathbf{x}}$ is performed. First of all we should split the input vector $\overline{\mathbf{x}} \in \mathbb{R}^N$ into two subvectors $\overline{\mathbf{x}}^1, \overline{\mathbf{x}}^2 \in \mathbb{R}^N$. The first one will contain the even elements of $\overline{\mathbf{x}}$ and the second the odd ones. The rest of the terms up to $\mathbb{R}^N$ will be zero. Therefore, the non zero entries are

$$\begin{aligned} \overline{x}_j^1 &= \overline{x}_{2j} \qquad \forall j \in \left[1, \frac{N}{2} - 1\right] \\ \overline{x}_j^2 &= \overline{x}_{2j+1} \qquad \forall j \in \left[1, \frac{N}{2} - 1\right] \end{aligned} \tag{C.21}$$

Let's define two auxiliary vectors as

$$\begin{aligned} \overline{\mathbf{x}}^{\dagger} &= \overline{\mathcal{F}(\overline{\mathbf{x}}^{1})} \\ \overline{\mathbf{x}}^{\ddagger} &= \overline{\mathcal{F}(\overline{\mathbf{x}}^{2})} \end{aligned} \tag{C.22}$$

which have to be transformed additionally as

$$\begin{aligned} \overline{x}_j^{\dagger} &= \overline{x}_j^{\dagger}\left[\cos\left(\frac{2\pi(j-1)}{N}\right) + i\sin\left(\frac{2\pi(j-1)}{N}\right)\right] \qquad \forall j \in [1, N] \\ \overline{x}_j^{\ddagger} &= \overline{x}_j^{\ddagger}\left[\cos\left(\frac{2\pi(j-1)}{N}\right) + i\sin\left(\frac{2\pi(j-1)}{N}\right)\right] \qquad \forall j \in [1, N] \end{aligned} \tag{C.23}$$

Recall that in this context, again, the Fourier transform $\mathcal{F}(\cdot)$ will be exactly the result of the FFT algorithm.

Then, two additional vectors are needed. These are

$$\begin{aligned} \overline{x}_j^{\dagger\dagger} &= \mathrm{Re}\left(\overline{x}_{j+1}^{\dagger}\right) \qquad \forall j \in [2, N-1] \\ \overline{x}_N^{\dagger\dagger} &= \mathrm{Re}\left(\overline{x}_1^{\dagger}\right) \\ \overline{x}_j^{\ddagger\ddagger} &= \mathrm{Im}\left(\overline{x}_{j+1}^{\ddagger}\right) \qquad \forall j \in [2, N-1] \\ \overline{x}_N^{\ddagger\ddagger} &= \mathrm{Im}\left(\overline{x}_1^{\ddagger}\right) \end{aligned} \tag{C.24}$$

To end up, the final vector $\mathbf{x}$ will be computed as

$$x_j = \frac{1}{2}(\overline{x}_1 + \overline{x}_N(-1)^j) + \overline{x}_j^{\dagger\dagger} + \overline{x}_j^{\ddagger\ddagger} \qquad \forall j \in [1, N] \tag{C.25}$$

C.1.3 Global computation algorithm

All the already explained procedures are only valid for a one dimensional case, but the good thing is that it can be easily applied to multidimensional cases in a natural way and using this primitive routines. The case being solved now is three dimensional and uniform. Thus, the matrix $\mathbf{A}^{3d} \in \mathbb{R}^{N^3 \times N^3}$ will be a rearrangement of the fundamental matrix displayed in Eq. (C.14) with $\alpha = 1$ and $\beta = -6$. Then, as an input for this routine only these two coefficients should be needed (not the entire matrix) alongside the independent term that will come in a form of a matrix $\mathbf{B} \in \mathbb{R}^{N \times N \times N}$ where the three indices i, j, k will correspond to a physical grid-point.

Then, the first thing that will be done is to obtain the eigenvalues with the parameters α and β and this will be done using the pseudo-code displayed in Algorithm 1 under the name of *evalEigen3*. Notice that a certain routine *eigenvalues1D* is called. This is just an evaluation of Eq. (C.15) and is left to the reader due to its immediate implementation.

After the eigenvalues have been obtained, the following step is to perform the Fourier transform. Recall that the input has been a matrix with the RHS of the Poisson equation while the algorithm needs a vector. Therefore, an intermediate function must be created that transforms this matrix into a vector and that calls either the Fourier transform or its inverse. A model for such function, that will be called *swafield*, can be seen in Algorithm 2. At this point two functions called *FastSwa* and *FastISwa* are used. This correspond to the operations explained in Subsection C.1.1 and Subsection C.1.2, respectively, and its implementation is immediate and thus, will not be presented.

Algorithm 1: Model for *evalEigen3*

Input : α ; β ;
Output: Λ - eigenvalues matrix
$L_1 \leftarrow eigenvalues1D(\alpha, \beta)$ // Aux vector with eigenvalues in the x direction
for $i \in [1, N]$ **do**
  $L_2(i, :) \leftarrow eigenvalues1D(\alpha, L_1(i))$ // Aux matrix with eigenvalues in the x and y direction
end
for $j \in [1, N]$ **do**
  for $i \in [1, N]$ **do**
    $L_1 \leftarrow eigenvalues1D(\alpha, L_2(i, j))$ // We reuse the aux vector with eigenvalues in the z direction
    for $k \in [1, N]$ **do**
      $\Lambda(i, j, k) \leftarrow L_1(k)$ // Matrix with all eigenvalues
    end
  end
end

Algorithm 2: Model for *swafield*

Input : Matrix **b** ; Integer inv (0: direct, 1: inverse) ; Integer $axis$ ($1 - x$, $2 - y$, $3 - z$)
Output: Matrix **a**
if $axis = 1$ **then**
  for $k \in [1, N]$ **do**
    for $j \in [1, N]$ **do**
      for $i \in [1, N]$ **do**
        $d1 \leftarrow \mathbf{b}(i, j, k)$ // Auxiliary vector that will contain the elements of a row in x direction
      end
      if $inv = 0$ **then**
        $d2 \leftarrow FastSwa(d1)$ // Another auxiliary vector that will contain the elements already transformed of a row in x direction
      else
        $d2 \leftarrow FastISwa(d1)$ // Another auxiliary vector that will contain the elements already inverse transformed of a row in x direction
      end
      for $i \in [1, N]$ **do**
        $\mathbf{a}(i, j, k) \leftarrow d2(i)$ // Auxiliary vector that will contain the elements of a row in x direction
      end
    end
  end
end
if $axis = 2$ **then**
  /* Similarly but the order of indices of the *for* statements will be k, i, j */
end
if $axis = 3$ **then**
  /* Similarly but the order of indices of the *for* statements will be j, i, k */
end

At this stage, all the equations have been decoupled in such a way that there are N^3 independent equations that can be solved in a very straightforward manner. This is due to the fact the the decomposition in the eigenspace in all three space dimensions produces the diagonal matrix of eigenvectors. The solution of this system can be called $\overline{\mathbf{x}}$ and remember that is equivalent to $\overline{\mathbf{x}} = \Lambda^{-1}\overline{\mathbf{b}}$ and can be computed as

$$\overline{x}_{ijk} = \frac{\overline{b}_{ijk}}{\lambda_{ijk}} \tag{C.26}$$

To recover the solution $\mathbf{x}$, which recall that will be expressed in a matrix notation corresponding to the grid-points, the function *swafield* must be applied but with $inv = 1$ to perform the inverse transformation.

If the FFT is not applied, the order of operations that would be needed are, according to [58], of order $\mathcal{O}(N^7)$. This is unacceptable for a CFD code. The FFT algorithm has an order of operations of $\mathcal{O}(N \log_2(N))$ [59]. Hence, for the 3D case, this order (which corresponds to the most restrictive operation count) will grow as $\mathcal{O}(3N^3 \log_2(N))$. This is much faster than many available algorithms to solve a linear system of equations.

Note. *Matrix $\boldsymbol{A}$ contains a singularity, since $||\boldsymbol{A}|| = 0$, and hence, it must be treated. In this algorithm the manifestation of this singularity is in the eigenvalues, because one of them will be zero. Therefore, at some point a division by a zero will be performed and that is unacceptable. This singularity comes from the fact that there are an infinite amount of pseudo-pressures distributions which laplacian equals a certain function and they are off by just a constant. This is the reason why one of those must be selected and a way to do so is by conditioning the matrix. One way, is to ensure that one of the diagonal elements is not exactly equal to the sum of the other coefficients. For example, multiply the element $\boldsymbol{A}_{111}$ by a certain constant different than 1 and 0 (e.g. mulitply by 5).*

Note. *The algorithm here presented, due to the FFT, requires that N is an integer of the form $N = 2^k \quad k \in \mathbb{N}$, otherwise does not work.*

C.2 Three dimensional with periodic-periodic-Dirichlet BC and non-uniform spacing in the z-direction Poisson equation

In this final section a more complicated case will be discussed. This is the case of a mixed boundary conditions as well as a non uniform grid in one axis. This will mess up the nice properties of matrix $\mathbf{A}$ and hence, a different approach must be followed. To start with, let us deduce the new matrix shape to introduce the non uniformity in the z-direction. Additionally, the spacing in x and y directions, although uniform, they will be assumed to be different from each other. A final new aspect that will be introduced is the possibility of having N_x, N_y and N_z points in each axis that might not be equal. In this case, the discrete Poisson equation leaves to a discretization of

$$a_P\tilde{p}_P + a_E\tilde{p}_E + a_W\tilde{p}_W + a_N\tilde{p}_N + a_S\tilde{p}_S + a_T\tilde{p}_T + a_B\tilde{p}_B = f \tag{C.27}$$

but in this case the coefficients are a function of k as seen in Section 7.3. Therefore, in each two dimensional $x-y$ plane (i.e. for each k) will have the good properties that allow us to use the Fourier solver, but not in the z-direction. Therefore, we might use the property of uncoupling the equations in two dimensions to obtain $N_x \times N_y$ tridiagonal systems (of the form shown in Eq. (C.2)). This procedure of applying the Fourier technique in only some directions to break the dependency with some neighboring points is also described and used in [61].

C.2.1 Global computation algorithm

From this basic idea of decoupling the equations, in this section a procedure will be explained of how to perform it. First of all, only the vectors containing the coefficients will be needed. These are $\mathbf{a}_{EW}, \mathbf{a}_{NS}, \mathbf{a}_P, \mathbf{a}_T, \mathbf{a}_B \in \mathbb{R}^{N_z}$. To apply the Dirichlet boundary conditions in the z direction notice that the gradient of pressure should be zero (and thus $\vec{u}^{\,n+1} = \vec{u}^{\,p}$ at the boundary) and this changes the coefficients in the following way:

$$\begin{aligned} \mathbf{a}_P(1) &= \mathbf{a}_P(1) + \mathbf{a}_B(1) \\ \mathbf{a}_P(N_z) &= \mathbf{a}_P(N_z) + \mathbf{a}_T(N_z) \\ \mathbf{a}_B(1) &= 0 \\ \mathbf{a}_T(N_z) &= 0 \end{aligned} \tag{C.28}$$

Now, everything is ready to solve the system of equations. The first step is to diagonalize in the x and y directions to decouple the different equations. The model for such function is found in Algorithm 3. After the diagonalization in those two directions is performed, there are $N_x \times N_y$ systems of tridiagonal equations that must be solved. Good news, though, are that they can be solved using a TDMA algorithm (a very well known solver as explained in [62]). After calling the *swafield* function twice (in x and y directions) to obtain the RHS in the eigenspace, the solution of the uncoupled equations can be done by means of the routine shown in Algorithm 4. The *tdma* subroutine is displayed in Algorithm 5.

Then, the solution must be returned to the original space and that is done by means of *swafield* but performing the inverse transformation.

Algorithm 3: Model for *diagonalizeXY*

Input : Vectors of length N_z with the coefficients a_{ew} ; a_{ns} ; a_p
Output: Λ - eigenvalues matrix
for $k \in [1, N_z]$ **do**
  $L_1(k,:) \leftarrow eigenvalues1D(a_{ew}(k), a_p(k))$ // Aux matrix with eigenvalues in x direction (which are different for each k)
end
for $k \in [1, N_z]$ **do**
  for $i \in [1, N_x]$ **do**
    $L_2 \leftarrow eigenvalues1D(a_{ns}(k), L_1(k,i))$ // Aux vector with eigenvalues in the y direction
    for $j \in [1, N_y]$ **do**
      $\Lambda(i,j,k) \leftarrow L_2(k)$ // Matrix with all eigenvalues in x and y
    end
  end
end

Algorithm 4: Model for *solveUncoupledZ*

Input : Matrix with the eigenalues Λ ; Vectors of length N_z with the coefficients a_t ; a_b ; Matrix with the independent terms of the Poisson equation **b**

Output: **x** - solution matrix

for $j \in [1, N_y]$ **do**
  for $i \in [1, N_x]$ **do**
    for $k \in [1, N_z]$ **do**
      $a_p1d(k) \leftarrow \Lambda(i, j, k)$ `// Auxiliary 1D vector with the eigenvalues`
      $b1d(k) \leftarrow \mathbf{b}(i, j, k)$ `// Auxiliary 1D vector with the independent terms`
    end
    if $i = 1$ *and* $j = 1$ **then**
      $ap1d(1) \leftarrow ap1d(1) \cdot 1.01$ `// Conditioning of the system`
    end
    $x1d \leftarrow tdma(a_b, a_p1d, a_t, b1d)$ `// Vector with the solution of this tridiagonal system`
    for $k \in [1, N_z]$ **do**
      $\mathbf{x}(i, j, k) \leftarrow x1d(k)$ `// Write the solution into the corresponding indexes of the matrix`
    end
  end
end

Algorithm 5: Model for *tdma*

Input : Vectors of length N_z with the coefficients a_t ; a_b ; a_p ; b

Output: x - solution vector

$C(1) \leftarrow a_e(1)/a_p(1)$ `// Auxiliary vector`
$D(1) \leftarrow b(1)/a_p(1)$ `// Auxiliary vector`
for $i \in [2, N_z]$ **do**
  $C(i) \leftarrow a_e(i)/(a_p(1) - C(i-1) \cdot a_w(i))$
  $D(i) \leftarrow (b(i) - D(i-1) \cdot a_w(i))/(a_p(1) - C(i-1) \cdot a_w(i))$
end
$x(N_z) \leftarrow D(N_z)$
for $i \in [N_z - 1, 1]$ **do**
  $x(i) \leftarrow D(i) - C(i) \cdot x(i+1)$
end

D — Boundary conditions in the 3D code

Here, a thorough explanation of how the boundary conditions (BC) have been implemented for the 3D code will be presented. It is of great interest noting not only how the physics are involved but also the numerical techniques. This will be explained since there is not many information regarding the coding details involved for this problem. For now, only one special case apart from the full periodic will be considered: this is the case of imposed velocities and imposed temperatures at the top and bottom boundaries, which will reproduce the effect of a no-slip wall. These are the BC that one encounters in the Rayleigh-Bénard (RB) convection. Therefore, all BC that are non periodic will be Dirichlet-like.

The staggered nature of the grid will imply a very special treatment of these BC and hence, it is very important to recall the disposition of nodes and control volumes (CV) in near wall cells. The situation that is being discussed can be seen in Figure D.1. Note, that both boundaries will have to be treated differently. In those schemes, the last element of the domain are the elements i, k and the others are ghost CV that will help to introduce the BC as it will be explained later.

D.1 Top boundary

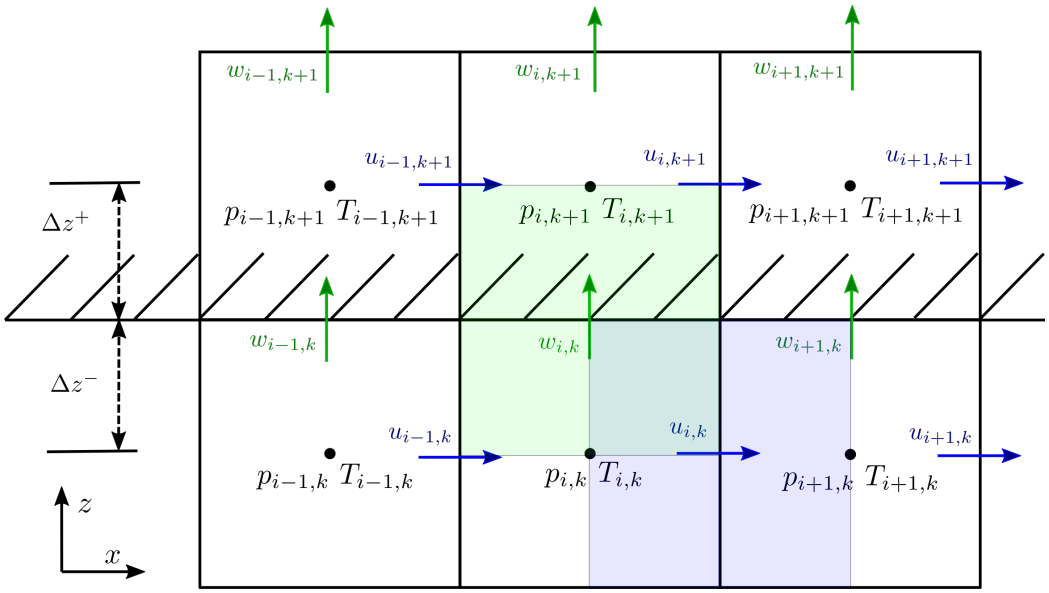
The first boundary that will be discussed is the top one. First of all, recall the expressions that have to be evaluated. This are convection and diffusion, which can be numerically approximated using a finite volume scheme and which expressions can be seen in Eq. (D.1) and Eq. (D.2), respectively, for the x-axis component of the velocity. From here, it will be deduced the new expression for those terms and then the decision on how to compute them will be made.

$$\int_\Omega \nabla \cdot (u\vec{u}) d\Omega = u_e F_e - u_w F_w + u_n F_n - u_s F_s + u_t F_t - u_b F_b \tag{D.1}$$

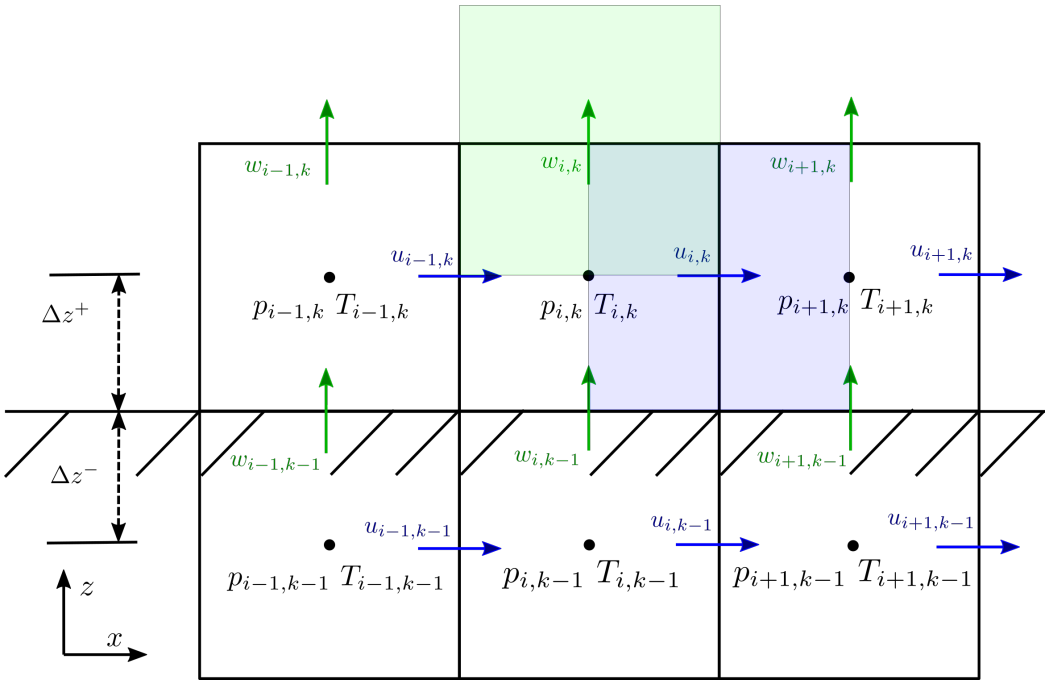
$$\int_\Omega \nabla \cdot (\nabla u) d\Omega = S_{ew} \left(\left.\frac{\partial u}{\partial x}\right|_e - \left.\frac{\partial u}{\partial x}\right|_w \right) + S_{ns} \left(\left.\frac{\partial u}{\partial y}\right|_n - \left.\frac{\partial u}{\partial y}\right|_s \right) + S_{tp} \left(\left.\frac{\partial u}{\partial z}\right|_t - \left.\frac{\partial u}{\partial z}\right|_b \right) \tag{D.2}$$

The first thing worth noting is that the wall boundary condition means null velocity at the actual wall. Then, the last element itself will not be zero, nor it should be the ghost element. The actual zero velocity is the term that in the convective equation is denoted as u_t since it is the horizontal component of the velocity at the top face of the control volume that has been drawn in blue. Therefore, a way to impose this $u_t = 0$, or in a general case for a moving wall, $u_t = u_t^{ref}$ is needed.

A way to implement this in the code would be by imposing it during the loop to compute the convective contribution. Another solution, though, is to fill the ghost element and leave the loop general. Then, recalling the scheme that is now being used, the value to be stored in the ghost CV



(a) Top boundary.



(b) Bottom boundary.

Figure D.1: Schemes showing the disposition of the last elements of the domain to apply the BC.

can be found. The spectro-consistent scheme used computes u_t as shown in Eq. (D.3) where the index $i, k+1$ indicates the position of the ghost term. Then, to impose the BC is as easy as fill that element so as to obtain the desired value of u_t. The expression resulting is shown in Eq. (D.4).

$$u_t = \frac{u_{i,k} + u_{i,k+1}}{2} \tag{D.3}$$

$$u_{i,k+1} = 2u_t^{ref} - u_{i,k} \tag{D.4}$$

It is worth noting that only have been shown the indices relevant for this treatment and j has been omitted because of that. Now it is time to evaluate the derivative term that appears in Eq. (D.2) in the z-direction. Here, an approximation will be needed, since the exact value of that derivative is not known. Then, a first order approximation in the top face of the control volume of u in the z-direction is

$$\left.\frac{\partial u}{\partial z}\right|_t \simeq \frac{u_t - u_{i,k}}{\Delta z^-} \tag{D.5}$$

And the way in which the code is numerically computing it is

$$\left.\frac{\partial u}{\partial z}\right|_t = \frac{u_{i,k} - u_{i,k+1}}{\Delta z^+ + \Delta z^-} \tag{D.6}$$

Therefore, Eq. (D.5) and Eq. (D.6) can be equated and the value for the ghost term found. The result is, in general terms, shown in Eq. (D.7). But note, that if the distances Δz^+ and Δz^- were equal, then the expression would simplify dramatically. If the mesh is done so as to assure this condition, the formula to fill the ghost term will be the same as seen for the convection, Eq. (D.4).

$$u_{i,k+1} = u_{i,k} - \frac{\Delta z^+ + \Delta z^-}{\Delta z^-}\left(u_t^{ref} - u_{i,k}\right) \tag{D.7}$$

The term involving the y-component of the velocity would follow the same structure and derivation leading to a result of

$$v_{j,k+1} = 2v_t^{ref} - v_{j,k} \tag{D.8}$$

$$v_{j,k+1} = v_{j,k} - \frac{\Delta z^+ + \Delta z^-}{\Delta z^-}\left(v_t^{ref} - v_{j,k}\right) \tag{D.9}$$

Then, the only term left to be analyzed is the one involving the z-component of the velocity. The staggered nature of the grid puts half of its CV inside the wall and the other half inside the domain. This might be a problem, and it actually is, but it is important to go back a little and recall why is this calculation being made. In fact, it is only done as part of the computation of $R^n(\vec{u})$ which ultimately is used to find the velocity at the next time step. But looking closer to the scheme in Figure D.1a it can be noticed that $w_{i,k}$ is on the actual boundary, which, by definition, its value is already known. For a no-slip wall this is $w_{i,k} = 0$, and then there is no need of calculating it.

In conclusion, for the top boundary, the ghost elements -also known as halos- will be filled for the u and v fields according to Eq. (D.4) and Eq. (D.8). For w it is not needed to compute anything because it will not be used later in the algorithm, and in a given point of the code this will be filled with the correct value.

D.2 Bottom boundary

Now it's time to focus ourselves in the bottom boundary. It is very similar to the previous but some subtle changes will be needed and, in fact, for coding purposes they will need to be treated differently. The first thing that is worth noting is that in this case the inner element is on the actual physical domain and the one that falls into the exact wall will pertain to the halo. Therefore, it is very easy to see which value must be introduced into this ghost point for the case of the z-component of the velocity because it will be directly $w_{i,k-1} = 0$.

For the ghost CV of the x-component of the velocity, nothing very special should be added, since the situation is similar to the one explained for the top boundary, with the only change that both the indices and distances are slightly different. Therefore, the whole procedure will not be repeated again and instead the result will be presented. This can be seen in Eq. (D.10) and Eq. (D.11).

$$u_{i,k-1} = 2u_b^{ref} - u_{i,k} \tag{D.10}$$

$$v_{j,k-1} = 2v_b^{ref} - v_{j,k} \tag{D.11}$$

D.3 Code implementation

In this last part, the implementation that has been added into the code will be explained. The first point at which this BC must be imposed is at the calculation of R. To do so, before the main loop of this part, a function should be called to correctly fill the halo elements of the velocity field. This function will be called *applyConvectionBC* and *applyDiffusionBC*. As it has been shown, the expression is the same for both terms and for these particular BC they will have an equivalent algorithm, which can be seen in Algorithm 6. The function that fills the halos is *dirichletBC* and its pseudocode is displayed in Algorithm 7.

Algorithm 6: Model for *applyConvectionBC*, *applyDiffusionBC* and *applyValueBC*

```
Data: Information on which BC must be applied on each boundary
Input  : f - field; BCtype[3][2] - matrix with the information of the BC
Output: modified field f
for a ← 1 : 3 do
    for b ← 1 : 2 do
        if BCtype[a][b] = 0 then
            continue
        else
            dirichletBC
        end
    end
end
```

Then, after R has been computed with the correct values at halos, the so-called predictor velocity $\vec{u}_p$ can be obtained. At this point, the halos of this vector are not correctly filled and hence, a function should be called in order to fill them with the values here presented to ensure that its divergence -which correspond to the RHS of the Poisson equation- has the true values at the boundaries. This function is called *applyValueBC* and follows the same structure shown in Algorithm 6. After the new velocity $\vec{u}^{\,n+1}$ is found the halos from $\vec{u}_p$ are just copied.

Algorithm 7: Model for *dirichletBC*

Data: Information regarding the axis in which is being applied and whether the field is staggered in that direction
Input : f - field; a - axis; b - semiaxis; stg - staggering; val - value to be imposed
Output: f after modification

```
if a = 3 then
    if b = 0 then
        if stg = a then
            for i ∈ [1, N_x], j ∈ [1, N_y], k ← 1 do
                /* For all inner elements at the bottom boundary */
                f_{i,j,k-1} ← val
            end
        else
            for i ∈ [1, N_x], j ∈ [1, N_y], k ← 1 do
                /* For all inner elements at the bottom boundary */
                f_{i,j,k-1} ← 2val − f_{i,j,k}
            end
        end
    else
        if stg = a then
            for i ∈ [1, N_x], j ∈ [1, N_y], k ← N_z do
                /* For all inner elements at the top boundary */
                f_{i,j,k} ← val
                f_{i,j,k+1} ← val
            end
        else
            for i ∈ [1, N_x], j ∈ [1, N_y], k ← N_z do
                /* For all inner elements at the top boundary */
                f_{i,j,k+1} ← 2val − f_{i,j,k}
            end
        end
    end
else
    return
end
```

A final question will now be addressed: how can a new BC be implemented? The answer is to change the $BCtype[3][2]$ matrix with a number not used already and then at the main loop of Algorithm 6 call a new function that the user can create to fill the ghost elements as desired.

The implementation of these BC can be validated using the Method of Manufactured Solutions choosing an appropriate set of functions that have the same desired BC. For a no-slip wall in the top and bottom domains a velocity field that would be appropriate to do this test is

$$\vec{u} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} \cos(2\pi x)\sin(2\pi z) \\ 0 \\ \sin(2\pi x)\left(1-\cos(2\pi z)\right) \end{bmatrix} \tag{D.12}$$

It is important to say that this function is only valid for a domain $\Omega = [0,1]\times[0,1]\times[0,1]$. Apart from fulfilling the BC of periodicity in the x-axis and null velocities at z-axis boundaries, it doesn't violate the incompressibility condition since

$$\nabla\cdot\vec{u} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = -2\pi\sin(2\pi x)\sin(2\pi z) + 2\pi\sin(2\pi x)\sin(2\pi z) = 0$$

Performing the correspondent convergence analysis the plot obtained is the one seen in Figure D.2 for the z-component of the velocity. As it was expected, the convergence order is still of $\mathcal{O}(\Delta^2)$ indicating a good implementation of the boundary conditions. A similar analysis has been performed for a centered field and with a staggering in the y-axis.

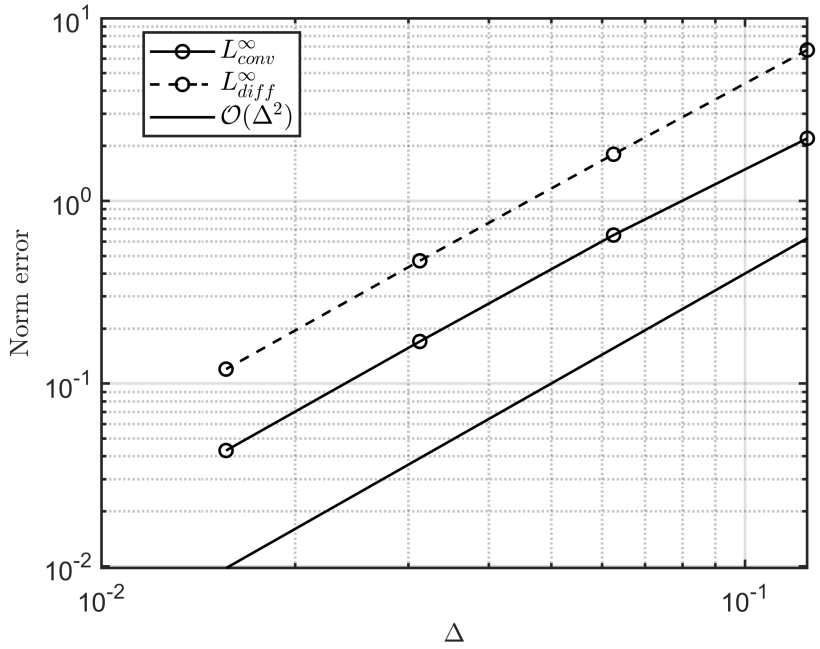


Figure D.2: Convergence plot for the implementation of wall BC at top an bottom for the z-component of the velocity.

E — Validation of the 3D code

In this appendix the validation of the 3D code will be presented. The first of those validations that have been performed is the MMS test as already presented in Figure B.3 for the convection. Then, the entire solver will be tested using a known analytical solution to a particular case.

E.1 MMS validation of the diffusion and convection

To perform this validation the functions that compute the convection and diffusion will be called for the centered and the three staggered possibilities. Then, the maximum L^∞ norm will be taken and the corresponding log-log plot displayed. The functions that must be used have to be in agreement with the boundary conditions of full periodicity and must fulfill the incompressibility condition. The velocity field that has been used in this case is

$$\begin{aligned} u &= \sin(x)\cos(y)\sin(z) \\ v &= \cos(x)\sin(y)\sin(z) \\ w &= 2\cos(x)\cos(y)\cos(z) \end{aligned} \tag{E.1}$$

being the domain $\Omega = \{(x, y, z) \in [0, 2\pi] \times [0, 2\pi] \times [0, 2\pi]\}$. The results obtained for both the diffusion and convection for the spectroconsistent scheme are displayed in Figure E.1. The same test has been performed for the MUSCL and Upwind schemes to see whether the TVD implementation for the 3D code has been correctly implemented. The results can be seen in Figure E.2.

E.2 Comparison with an analytic solution

Navier-Stokes equations are famous for not having a general solution, yet under some some assumptions, we can find an analytic expression for the velocity field such that fulfills the equations and the boundary conditions. That is the case for a full periodic square domain of $\Omega = \{(x, y) \in [0, 1] \times [0, 1]\}$ with the following velocity field

$$\begin{aligned} u &= e^{-8\pi^2\nu t}\cos(2\pi x)\sin(2\pi y) \\ v &= -e^{-8\pi^2\nu t}\cos(2\pi y)\sin(2\pi x) \\ w &= 0 \end{aligned} \tag{E.2}$$

Note that the incompressibility condition is not violated and that is possible to find a pressure field that maintains the equality true. This pressure field has the form of

$$p = -\rho\frac{\cos(4\pi x) + \cos(4\pi y)}{4}e^{-16\pi^2\nu t} + A \tag{E.3}$$

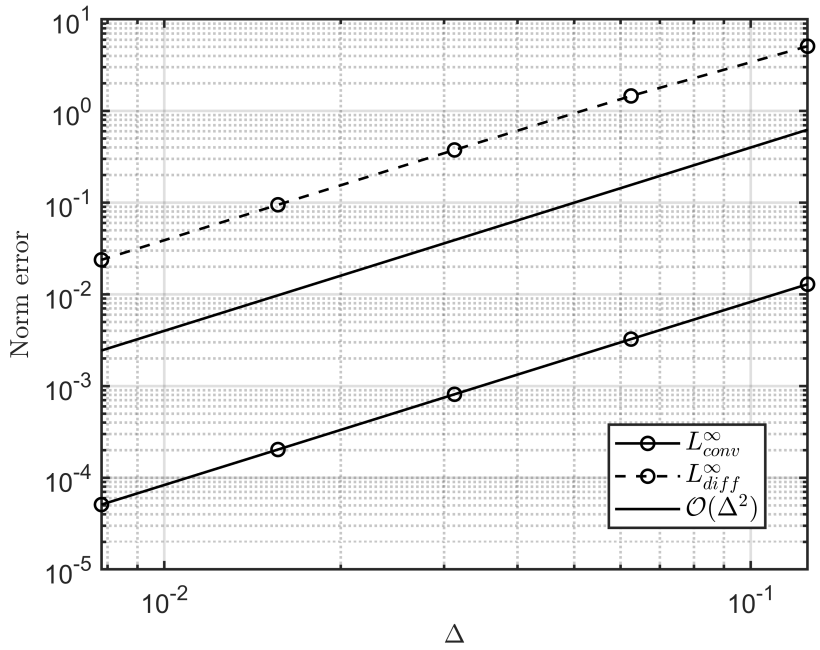


Figure E.1: MMS test for the spectroconsistent scheme.

with $A \in \mathbb{R}$.

Also, it is possible to find the kinetic energy evolution. This will be

$$K_e = \frac{1}{4} e^{-16\pi^2 \nu t} \tag{E.4}$$

Then, we just have to perform a run and compare whether the obtained results coincide with the theoretical ones for a sufficiently smooth mesh. The comparison is displayed in Figure E.3. These results demonstrate a good overall performance of the INS routines.

This two tests only show a good implementation, yet further benchmarks have to be done in order to be sure that it works correctly for scientific research. This is also interesting because since the code runs in parallel more errors can appear than in a sequential code.

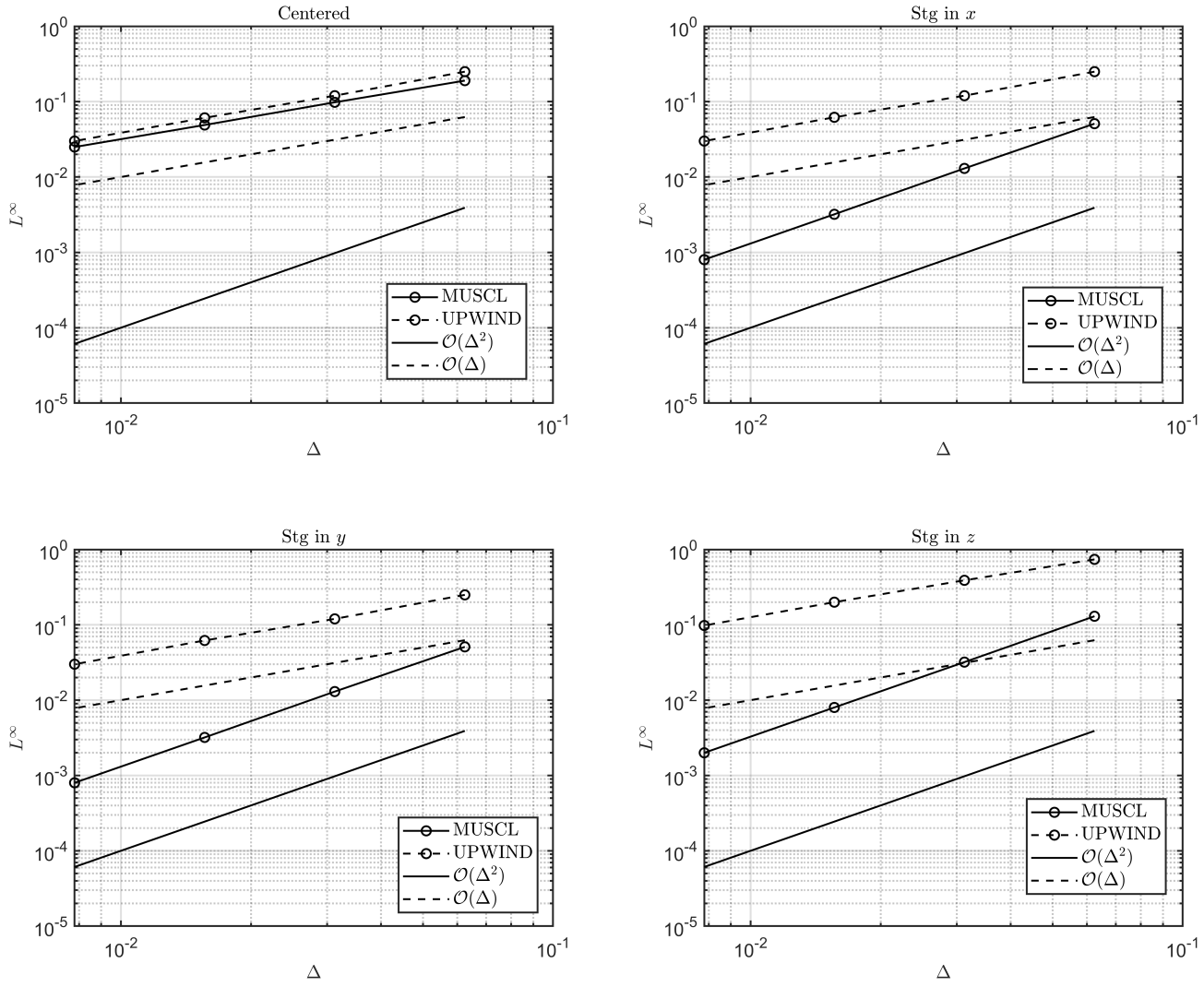
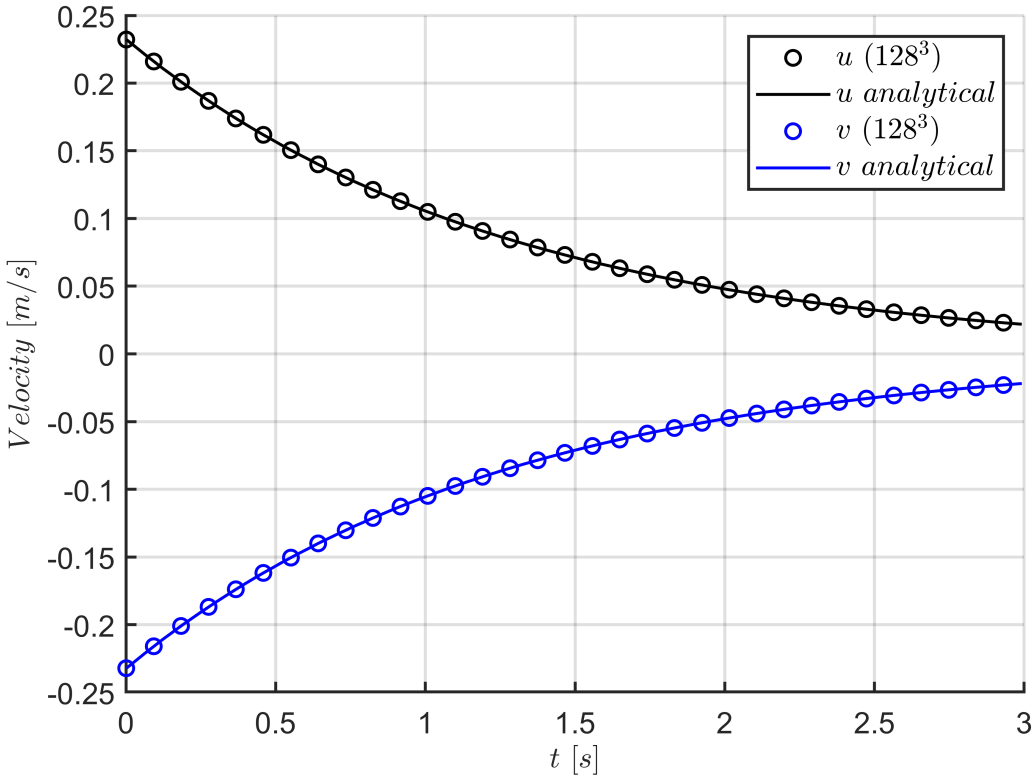
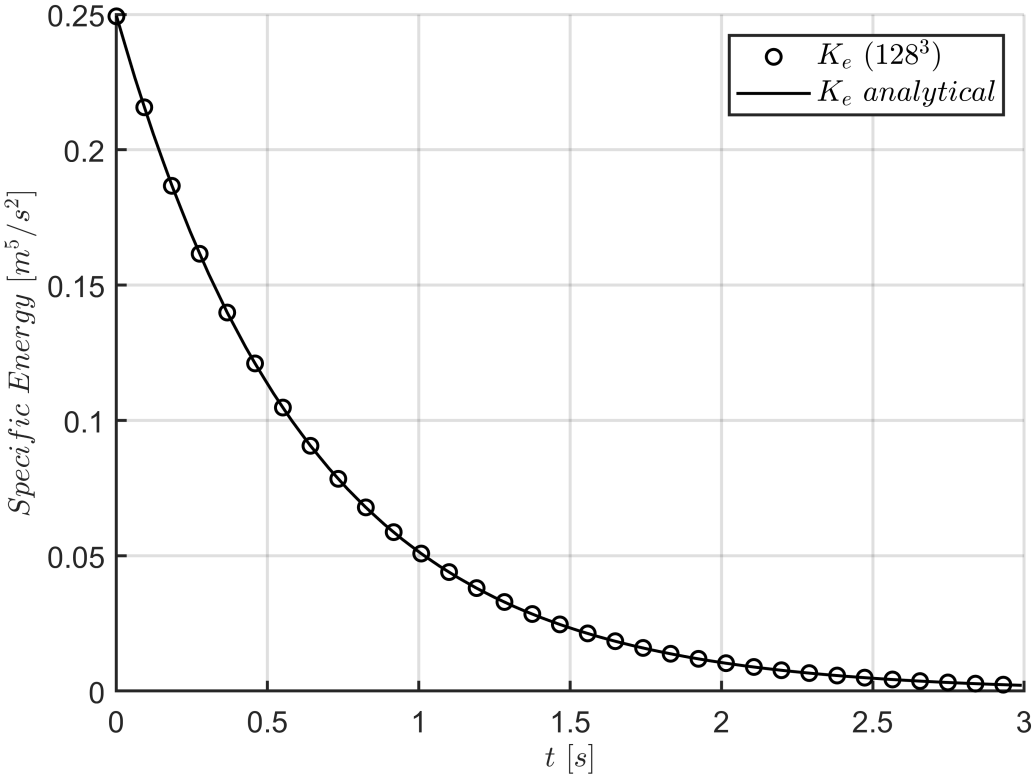


Figure E.2: MMS test for the MUSCL and Upwind schemes for different staggering options.



(a) Comparison of the velocity field.



(b) Comparison of the kinetic energy evolution.

Figure E.3: Comparison for the 3D code between the theoretical and numerical results for a mesh of $N = 128^3$.