Universidad de Cantabria

Facultad
de
Ciencias

SIMULACIÓN NUMÉRICA DE LOS ESCALARES ACTIVOS
(Numerical simulation of active scalars)

Trabajo de Fin de Grado
para acceder al

GRADO EN MATEMÁTICAS

Autor: Carlos Bouso Pérez

Director: Rafael Granero Belinchón

Junio - 2024

Índice general

Resumen

Los escalares activos son un tipo de ecuaciones diferenciales en derivadas parciales con gran importancia en matemáticas y física. En este artículo, se realizan simulaciones numéricas de estas ecuaciones para lo cual se necesitan dos principales herramientas. Los métodos Runge-Kutta sirven para aproximar soluciones de ecuaciones diferenciales ordinarias. Por otro lado, el algoritmo FFT, que se basa en la implementación en la computación de la transformada de Fourier, se utilizará para pasar de derivadas en el espacio real a multiplicadores en el espacio de Fourier con lo que se transforman las ecuaciones diferenciales en derivadas parciales a ecuaciones diferenciales ordinarias.

Palabras clave: Escalar activo, Runge-Kutta, FFT, Euler, SQG, IPM, Stokes, Fourier, vorticidad.

Abstract

Active scalars are a type of equations that have much importance in mathematics and physics. In this article, numerical simulations of this equations are computed. There are two esential tools that are necessary to do these simulations. Runge-Kutta methods calculate an aproximation for the solution of an ordinary differential equation. On the other hand, the FFT algorithm, based on computing Fourier transform, turns partial differential equations into ordinary differential equations.

Key words: Active scalar, Runge-Kutta, FFT, Euler, SQG, IPM, Stokes, Fourier, vorticity.

Capítulo 1

Introducción

En física, un fluido se considera cualquier gas, líquido u otro material que puede moverse y deformarse de forma continua si es expuesto a una fuerza externa y que no pueden resistir ninguna tensión cortante. Modelar el movimiento de los fluidos reales resulta muy complicado debido a la cantidad de variables a tener en cuenta, por eso se introduce el concepto de fluido ideal e incompresible. Un fluido es ideal cuando la viscosidad, que es la resistencia a ser deformado, es cero y cuando no existe ninguna tensión cortante aunque esté en movimiento. De esta forma el movimiento del fluido estará gobernado únicamente por la presión. Un fluido incompresible es aquel en el que el volumen no es afectado por cambios en la presión o la velocidad del fluido.

Para estudiar el movimiento de un fluido ideal incompresible se utiliza un sistema de ecuaciones diferenciales en derivadas parciales (EDP) llamadas Ecuaciones de Euler. Estas ecuaciones aparecen por primera vez en el artículo *Principes généraux du mouvement des fluides* publicado en *Mémoires de l'Académie des Sciences de Berlin* en 1757 y fueron las primeras ecuaciones diferenciales en derivadas parciales no lineales que se formularon.

Estas ecuaciones se derivan de las leyes fundamentales de la física, más concretamente de la conservación de la masa y la conservación del momento lineal, sin embargo veremos que, en determinados casos, la magnitud realmente relevante es la vorticidad. La vorticidad, en un fluido, es la magnitud que describe de forma local el movimiento continuo giratorio cerca de un determinado punto. Las ecuaciones de Euler llevan siendo investigadas durante años y aunque se ha descubierto mucho a cerca de ellas, la naturaleza de las soluciones es todavía un gran misterio. Estudiar las singularidades de las soluciones es algo complicado. Por esto, desde aproximadamente 1990 se empezaron a trabajar con los llamados escalares activos.

Los escalares activos son un tipo de ecuaciones diferenciales en derivadas parciales que modelan fenómenos en los que un campo escalar se transporta con una velocidad que se recupera de conocer el propio escalar. Uno de los campos de estudio más importantes es la mecánica de fluidos donde el campo escalar es una cantidad física que suele ser la densidad, la concentración de una sustancia química o la temperatura y, el campo vectorial que es la velocidad del fluido.

En un primer capítulo se tratará la relación entre las ecuaciones de Euler 3D y los escalares activos. Además, se introducirán los escalares activos más importantes en 2D, así como dos en una dimensión que servirán para comenzar con las simulaciones numéricas. En el tercer y cuarto capítulo se explican las dos herramientas esenciales para poder simular los escalares activos, los métodos Runge-Kutta que aproximan la solución de ecuaciones diferenciales ordinarias (EDOs), y el algoritmo FFT basado en la transformada de Fourier con el que conseguiremos expresar los escalares activos como EDOs pasando las derivadas espaciales a multiplicadores en el espacio de Fourier. Por último, un capítulo donde se muestran las simulaciones numéricas realizadas así como los resultados obtenidos.

Capítulo 2

Las ecuaciones de Euler 3D y los escalares activos

En este capítulo, se verá la relación entre las ecuaciones de Euler 3D y los escalares activos. Con esto, se determina la importancia que tienen estos últimos en las matemáticas y la física de la actualidad. También se introducen aquellos con los que se realizarán simulaciones más adelante.

Como se ha comentado previamente las ecuaciones de Euler 3D se derivan de la conservación de la masa y la conservación del momento lineal, pero usando el Criterio de Beale-Kato-Majda se tiene que la magnitud realmente relevante en determinados casos es la vorticidad.

Criterio de Beale-Kato-Majda

Una de las grandes problemáticas de las Ecuaciones de Euler es la pérdida de regularidad en las soluciones. Desde que se han formulado, dichas pérdidas de regularidad han sido estudiadas mediante computaciones numéricas (ver [2], [3], [10]) . Aunque se obtienen predicciones concordantes entre sí, la verificación es algo muy complicado debido a su gran inestabilidad. Una de las observaciones que se realizó en estas simulaciones, además de en otros estudios cualitativos, fue la relación entre la concentración de vorticidad a nivel local con la formación de singularidades. En 1984 J.T.Beale, T. Kato y A. Majda. probaron el siguiente criterio cuya demostración se puede ver en [6] con objeto de estudiar este suceso.

Criterio. *Sea* $u \in C[0,T];H^s) \cap C^1([0,T];H^{s-1})$*, con* $s \geq 3$*, una solución de las Ecuaciones de Euler y un tiempo* $T_* > 0$ *tal que la solución no puede ser de la clase* $C[0,T];H^s) \cap C^1([0,T];H^{s-1})$ *para* $T = T_*$*. Suponiendo que* T_* *es el primer tiempo con esta condición se tiene:*

$$\int_0^{T_*} |w(t)|_{L^\infty} dt = \infty$$

y en particular

$$\lim_{t \to T_*} sup|w(t)|_{L^\infty} = \infty.$$

De lo anterior se deduce que si la vorticidad está acotada existe solución. Por lo tanto, la cantidad relevante de las ecuaciones de Euler para fluidos incompresibles es la vorticidad.

Con lo anterior, partiendo de la formulación clásica (2.1) de las ecuaciones de Euler (ver [8]) se obtendrá una formulación que introduce la vorticidad.

$$\begin{cases} u_t + (u \cdot \nabla)u = -\nabla p \\ \nabla \cdot u = 0 \end{cases} \tag{2.1}$$

Donde u es la velocidad del fluido y p la presión. Esta ecuación describe la evolución de la velocidad y la presión de un fluido incompresible a través del tiempo y del espacio. Se deriva a continuación la

formulación en función de la vorticidad.

Considerando $u = (u^1, u^2, u^3) \in \mathbb{R}^3$, aplicamos el rotacional a la primera ecuación. La vorticidad se define como el rotacional de la velocidad del fluido por lo que en este caso es $\vec{w} = \nabla \times u = (u^3_y - u^2_z, -u^3_x + u^1_z, u^2_x - u^1_y)$. Se tiene que $\nabla \times u_t = (\nabla \times u)_t = \vec{w}_t$ y $\nabla \times -\nabla p = 0$. Se calcula el siguiente término:

$$\begin{aligned}
\nabla \times (u \cdot \nabla)u = &\begin{vmatrix} i & j & k \\ \partial_x & \partial_y & \partial_z \\ u^1 u^1_x + u^2 u^1_y + u^3 u^1_z & u^1 u^2_x + u^2 u^2_y + u^3 u^2_z & u^1 u^3_x + u^2 u^3_y + u^3 u^3_z \end{vmatrix} \\
&= (u^1_y u^3_x + u^1 u^3_{xy} + u^2_y u^3_y + u^2 u^3_{y^2} + u^3_y u^3_z + u^3 u^3_{yz} - u^1_z u^2_x - u^1 u^2_{xz} - u^2_z u^2_y - u^2 u^2_{yz} - u^3_z u^2_z - u^3 u^2_{z^2}, \\
&u^1_z u^1_x + u^1 u^1_{xz} + u^2_z u^1_y + u^2 u^1_{yz} + u^3_z u^1_z + u^3 u^1_{z^2} - u^1_x u^3_x - u^1 u^3_{x^2} - u^2_x u^3_y - u^2 u^3_{xy} - u^3_x u^3_z - u^3 u^3_{xz}, \\
&u^1_x u^2_x + u^1 u^2_{x^2} + u^2_x u^2_y + u^2 u^2_{xy} + u^3_x u^2_z + u^3 u^2_{xz} - u^1_y u^1_x - u^1 u^1_{xy} - u^2_y u^1_y - u^2 u^1_{y^2} - u^3_y u^1_z - u^3 u^1_{yz}) \\
&= (u \cdot \nabla)\vec{w} + (u^1_y u^3_x + u^3_y (u^2_y + u^3_z) - u^1_z u^2_x - u^2_z (u^2_y + u^3_z), \\
&u^2_x u^1_y + u^1_z (u^1_x + u^3_z) - u^2_x u^3_y - u^3_x (u^1_x + u^3_z), \\
&u^3_x u^2_z + u^2_x (u^1_x + u^2_y) - u^3_y u^1_z - u^1_y (u^1_x + u^2_y))
\end{aligned} \tag{2.2}$$

Donde usando la condición de divergencia se obtiene la expresión $\nabla \times (u \cdot \nabla)u = (u \cdot \nabla)\vec{w} - \vec{w} \cdot \nabla u$. A continuación, se despeja la velocidad en función de la vorticidad porque será útil más adelante. Para ello, vamos a necesitar la siguiente identidad vista en [8]

$$\nabla \times \nabla \times \psi = \nabla(\nabla \cdot \psi) - \triangle \psi \tag{2.3}$$

para cualquier vector de campo $\psi \in \mathbb{R}$. Así, aplicando el rotacional a la vorticidad se tiene que $\nabla \times \vec{w} = \nabla \times \nabla \times u = \nabla(\nabla \cdot u) - \triangle u$. Teniendo en cuanta la condición de divergencia resulta $\nabla \times \vec{w} = -\triangle u$ y despejando se obtiene la expresión para la velocidad $u = (-\triangle)^{-1} \nabla \times \vec{w} = \nabla \times (-\triangle)^{-1} \vec{w}$. Esta última expresión es la ley de Biot-Savart. Recopilando ambas expresiones se tiene

$$\begin{cases} \vec{w}_t + (u \cdot \nabla)\vec{w} = \vec{w} \cdot \nabla u \\ u = \nabla \times (-\triangle)^{-1} \vec{w} \end{cases} \tag{2.4}$$

2.1. Escalares activos

Los escalares activos son ecuaciones diferenciales en derivadas parciales muy utilizadas en mecánica de fluidos donde describen la evolución de una cantidad física del fluido como la temperatura, densidad o la vorticidad a lo largo del tiempo. Además, la velocidad del fluido se puede recuperar a partir de la cantidad escalar mediante un operador. La formulación general de los escalares activos es la que se muestra a continuación.

$$\begin{cases} \theta_t + u \cdot \nabla \theta = 0 \\ u = T(\theta) \end{cases} \tag{2.5}$$

Donde θ es la función escalar (temperatura, presión, densidad...), u es la velocidad del fluido y T es un operador que determina la velocidad a partir de θ.

Para ver la relación que hay entre estas ecuaciones y las de Euler se toma el gradiente en la ecuación (2.5) obteniendo así

$$\nabla \theta_t + (u \cdot \nabla)\nabla \theta = \nabla \theta \cdot \nabla u \tag{2.6}$$

que tiene la misma estructura que (2.4), con $\vec{w} = \nabla \theta$, pero una variable menos. Veamos algunos escalares activos concretos.

2.1.1. Ecuación de Euler incompresible en 2D

Las ecuaciones de Euler modelan el comportamiento de un fluido sin viscosidad y describen la evolución de propiedades del fluido en función del tiempo. En dos dimensiones, las ecuaciones de Euler originales

según [8] son

$$\begin{cases} u_t + (u\cdot\nabla)u = -\nabla p \\ \nabla\cdot u = 0 \end{cases} \tag{2.7}$$

En este caso, p es la presión y consideramos $u = (u^1, u^2)$. Tomando $\vec{w} = \nabla \times u$ la vorticidad podremos expresar estas ecuaciones en su forma de escalar activo y determinar así el operador necesario para recuperar la velocidad a partir de la función θ. Para simplificar la notación, como $\nabla\times u = (0,0,-u^1_y+u^2_x)$, se llama vorticidad al escalar $w = -u^1_y + u^2_x$. Se aplica el rotacional en la primera ecuación de (2.7). Teniendo en cuenta que el rotacional y la derivada conmutan se tiene $\nabla\times(u_t) = (\nabla\times u)_t = (0,0,w_t)$. Además, el rotacional de un gradiente es siempre cero por lo que $\nabla\times(-\nabla p) = 0$. Así, queda por determinar el término

$$\nabla\times(u\cdot\nabla)u = \begin{vmatrix} i & j & k \\ \partial_x & \partial_y & 0 \\ u^1u^1_x + u^2u^1_y & u^1u^2_x + u^2u^2_y & 0 \end{vmatrix} = \tag{2.8}$$

$$= (0,0,u^1_xu^2_x + u^1u^2_{x^2} + u^2_xu^2_y + u^2u^2_{xy} - u^1_yu^1_x - u^1u^1_{xy} - u^2_yu^1_y - u^2u^1_{y^2}) =$$

$$= (0,0,(u\cdot\nabla)w + u^2_x(\nabla\cdot u) - u^1_y(\nabla\cdot u)) =$$

Por lo que, como tenemos $\nabla\cdot u = 0$, se obtiene $(0,0,(u\cdot\nabla)w)$. Falta determinar la velocidad a través de la vorticidad. Para ello, usando de nuevo la condición de divergencia, se tiene que, como $\nabla\cdot u = 0$, existe una función ϕ tal que $u = -\nabla^\perp\phi$. De esta forma, tomando el rotacional en esta ecuación se obtiene

$$\nabla\times u = \nabla\times -\nabla^\perp\phi = \begin{vmatrix} i & j & k \\ \partial_x & \partial_y & 0 \\ \phi_y & -\phi_x & 0 \end{vmatrix} = (0,0,-\phi_{x^2} - \phi_{y^2}) = (0,0,-\triangle\phi). \tag{2.9}$$

De esto se deduce que $w = -\triangle\phi$ y despejando se tiene $u = \nabla^\perp(-\triangle)^{-1}w$. Recogiendo todo lo anterior se tiene el escalar activo correspondiente a Euler 2D

$$\begin{cases} w_t + (u\cdot\nabla)w = 0 \\ u = \nabla^\perp(-\triangle)^{-1}w \end{cases} \tag{2.10}$$

Donde $(-\triangle)^{-1}f(x) = \frac{1}{2\pi}\int_{\mathbb{T}^2} K(x-y)f(y)dy$ con K un núcleo singular o escrito en el espacio de Fourier $\widehat{(-\triangle)^{-1}u} = \frac{1}{|\varepsilon|^2}\hat{u}$ con $\hat{u}$ la serie de Fourier de una función u.

2.1.2. Ecuación cuasigeostrófica de superficie (SQG)

En muchos ámbitos, como la meteorología, es importante el estudio del aire de la atmósfera y el agua de los océanos. El comportamiento de los flujos de agua y de aire caliente o frío, así como la mezcla entre ambos, se estudia para poder predecir sucesos meteorológicos o oceánicos. Considerando uno de estos fluidos en la superficie de la Tierra y teniendo en cuenta que el movimiento predominante sea la rotación terrestre, hay varias ecuaciones que aproximan el comportamiento de estos fluidos. Una de estas ecuaciones son las llamadas cuasi-geostróficas (QG). Además, a partir de las ecuaciones QG se deriva la ecuación SQG que se va a estudiar en este articulo por su importancia no solo en la descripción de flujos sobre la superficie de la Tierra si no como escalar activo.

Las ecuaciones QG se consideran en un sistema de referencia rotativo, como la Tierra, donde las fuerzas de Coriolis y la presión predominan sobre las fuerzas inerciales. El efecto de Coriolis, descrito por Gaspard-Gustave Coriolis en 1835, es una fuerza inercial sobre un objeto en movimiento con respecto a un sistema de referencia en rotación. Es decir, se produce una aceleración relativa del objeto, perpendicular al eje de rotación en el sistema de referencia y al movimiento del objeto, visto también desde el sistema de referencia. Las QG tienen la siguiente formulación (ver [18]):

$$\begin{cases} \theta_t + (u\cdot\nabla)\theta = 0 & \text{en } \mathbb{R}^2\times\{z=0\} = \mathbb{R}^2 \\ (\triangle\psi)_t + (-\psi_y,\psi_x)^\perp\cdot\overline{\nabla}\triangle\psi = 0 & \text{en } \mathbb{R}^2\times\mathbb{R}^+ \\ \theta = \psi_z & \text{en } \mathbb{R}^2\times\{z=0\} \end{cases} \tag{2.11}$$

donde $u = \nabla^{\perp}\psi$ y $\overline{\nabla}$ es el operador gradiente en dos dimensiones. La primera ecuación es la propiedad de conservación de la masa y la tercera es una condición en el borde ($z = 0$). A partir de la ecuación (2.11) se va a obtener la formulación de la SQG. Tomando un valor inicial

$$\triangle\psi(x, y, z, 0) = 0$$

teniendo en cuenta la segunda ecuación de (2.11) se tiene que

$$\triangle\psi(x, y, z, t)$$

permanecerá siendo 0. Por lo que se tiene la ecuación de Laplace:

$$\triangle\psi = \psi_{xx} + \psi_{yy} + \psi_{zz} = 0$$

Aplicando la transformada de Fourier a la ecuación de Laplace se obtiene

$$-\varepsilon_1^2\hat{\psi}(\varepsilon_1, \varepsilon_2, z) - \varepsilon_2^2\hat{\psi}(\varepsilon_1, \varepsilon_2, z) + \hat{\psi}_{zz}(\varepsilon_1, \varepsilon_2, z) = 0.$$

Esta expresión es una ecuación diferencial de segundo orden, homogénea con coeficientes constantes, luego se pueden escribir las soluciones

$$\hat{\psi}(\varepsilon_1, \varepsilon_2, z) = C_1 e^{-\sqrt{\varepsilon_1^2+\varepsilon_2^2}z} + C_2 e^{\sqrt{\varepsilon_1^2+\varepsilon_2^2}z}$$

donde C_1, C_2 son constantes que no dependen de la variable z. Como se quieren considerar soluciones que decaigan cuando z tiende a ∞ y $\sqrt{\varepsilon_1^2 + \varepsilon_2^2}z$ es positivo por estar en $z > 0$, se toma $C_2 = 0$ para evitar la divergencia del término $e^{\sqrt{\varepsilon_1^2+\varepsilon_2^2}z}$. Por tanto, utilizando la condición del borde con series de Fourier en la anterior expresión con $C_2 = 0$:

$$\hat{\psi}_z(\varepsilon_1, \varepsilon_2, 0) = -C_1\sqrt{\varepsilon_1^2 + \varepsilon_2^2} = \hat{\theta} \Longrightarrow \theta = (-\triangle)^{1/2}\psi.$$

Por último, usando la expresión $u = \nabla^{\perp}\psi$ para despejar u y retomando la conservación de la masa se obtiene la ecuación SQG siguiente (ver [12]):

$$\begin{cases} \theta_t + (u \cdot \nabla)\theta = 0 \\ u = \nabla^{\perp}(-\triangle)^{-1/2}\theta \end{cases} \tag{2.12}$$

donde $\widehat{(-\triangle)^{-1/2}u} = \frac{1}{|\varepsilon|}\hat{u}$ para cualquier función u.

2.1.3. Ecuación de un fluido incompresible en un medio poroso en 2D (IPM)

Una de las ecuaciones que tiene más importancia en mecánica de fluidos es la ecuación del medio poroso que estudia el movimiento de un fluido incompresible a lo largo de un medio homogéneo y poroso. Está ecuación se basa en la ley de Darcy.

Henry Philibert Gaspard Darcy fue un ingeniero francés que hizo grandes aportaciones a la mecánica de fluidos. La aportación más importante fue la ley de Darcy que formuló en 1855 basándose en un experimento. Este experimento consistía, como se puede ver en la figura 2.1, en una botella de metal situada en vertical con una entrada para el agua en el punto más alto y una salida al final por las que controlaba la presión mediante una serie de llaves. Realizaba diferentes tests usando distintos tipos de arenas y variando la presión tanto de entrada como de salida del agua.

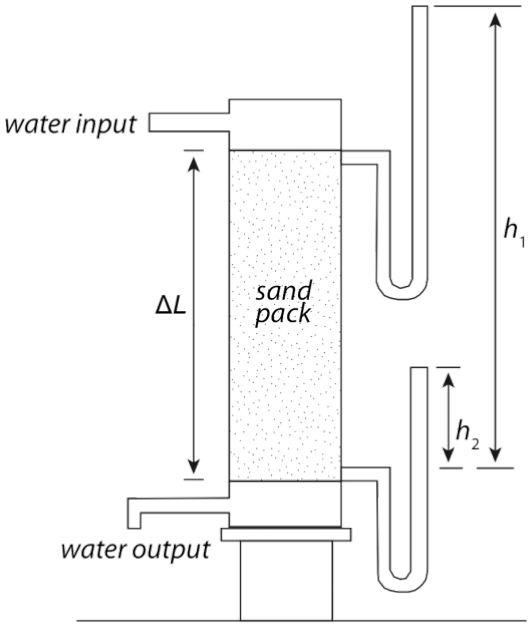

Figura 2.1: Dibujo que simula el experimento realizado por Darcy. Ver en: https://www.e-education.psu.edu/earth111/node/926

Así, pudo describir el comportamiento de un fluido a través de un medio poroso y deducir la ecuación IPM que se conoce en la actualidad. Como se puede ver en [1] la ecuación tiene la siguiente formulación

$$\begin{cases} u = -\nabla p - (0, \theta) \\ \nabla \cdot u = 0 \\ \theta_t + u \cdot \nabla \theta = 0 \end{cases} \tag{2.13}$$

donde θ es la densidad del fluido y p la presión. Se puede ver que la tercera ecuación de (2.13) tiene forma de escalar activo. Falta llegar a una expresión de la que recuperar la velocidad en función de θ utilizando las dos primeras ecuaciones. En 2D, como ya se ha utilizado antes, se tiene la propiedad de que si $\nabla \cdot u = 0$ entonces existe ψ tal que $u = \nabla^\perp \psi$. Sustituyendo esta expresión en la primera ecuación y aplicando el rotacional

$$\nabla \times \nabla^\perp \psi = 0 - \nabla \times (0, \theta) \iff -\triangle \psi = -\theta_x \implies \psi = -(-\triangle)^{-1}\theta_x \implies u = -\nabla^\perp (-\triangle)^{-1}\theta_x$$

Por lo que la ecuación IPM en su forma de escalar activo es

$$\begin{cases} \theta_t + u \cdot \nabla \theta = 0 \\ u = -\nabla^\perp (-\triangle)^{-1}\theta_x \end{cases} \tag{2.14}$$

donde $\widehat{(-\triangle)^{-1}u} = \frac{1}{|\varepsilon|^2}\hat{u}$ para cualquier función u.

2.1.4. Ecuación de Stokes incompresibles en 2D

Otra ecuación que tiene gran importancia en mecánica de fluidos es la ecuación de Stokes en dos dimensiones, llamada así por el irlandés físico y matemático George Gabriel Stokes. Es una gran simplificación de las ecuaciones de Navier-Stokes utilizadas cuando la viscosidad es mucho mayor que las fuerzas inerciales, que se consideran despreciables. Por tanto, serán útiles para describir el comportamiento de un fluido en los casos en los que este es muy viscoso o en los que la velocidad es baja.

Las ecuaciones de Navier-Stokes son ecuaciones que describen el comportamiento de substancias viscosas. Reciben ese nombre por sus desarrolladores, el ingeniero y físico Francés Claude-Louis Navier y por el ya nombrado Grabriel Stokes. Fueron desarrolladas desde 1822 por Navier hasta, entre 1842 y 1850 por Stokes basandose en la conservación del momento y de la masa. Su uso está altamente extendido pues modelan fenómenos tan importantes como el movimiento del agua a través de un tuvo, el del aire al rededor de un ala, el tiempo o las corrientes oceánicas. Pudiendo así ser una de las herramientas más importantes en el diseño de coches, aviones..., así como el estudio del comportamiento de la sangre a través del sistema circulatorio del cuerpo humano o el diseño de centrales eléctricas. La formulación de estas ecuaciones es la siguiente

$$\begin{cases} \theta(u_t + (u\cdot\nabla)u) + \nabla p - \nu\triangle u = -(0, g\theta)^\perp \\ \nabla\cdot u = 0 \\ \theta_t + u\cdot\nabla\theta = 0 \end{cases} \tag{2.15}$$

En este caso p es la presión del fluido, θ es la densidad y g la gravedad.
A partir de (2.17) se realiza una derivación para obtener la ecuación de Stokes en dos dimensiones. Para ello se va a utilizar una técnica muy utilizada en mecánica de fluidos que es la adimensionalización. Para esto se expresan las variables de la primera ecuación (2.17) en función de la longitud característica del sistema (L) y de la velocidad característica (U). De esta forma se tiene

Longitud	$x = L\underline{x}$ y $y = L\underline{y}$
Velocidad	$u = U\underline{u}$
Tiempo	$t = \frac{L}{U}\underline{t}$
Presión	$p = \frac{\nu U}{L}\underline{p}$
Densidad	$\theta = R\underline{\theta}$

Donde las variables subrayadas son las nuevas variables adimensionales. Se calculan los términos de la ecuación

$$u_t = \frac{\partial}{\partial t}u = \frac{\partial}{\partial \underline{t}}\frac{\partial \underline{t}}{\partial t}u = \frac{\partial}{\partial \underline{t}}\frac{\partial t}{\partial t}\frac{U}{L}u = \frac{\partial}{\partial \underline{t}}\frac{U}{L}U\underline{u} = \frac{U^2}{L}\underline{u_t}$$

$$u\cdot\nabla u = ((u^1,u^2)\cdot(\frac{\partial}{\partial x},\frac{\partial}{\partial y})^\perp)(u^1,u^2) = (U\underline{u}\cdot(\frac{\partial}{\partial \underline{x}}\frac{1}{L},\frac{\partial}{\partial \underline{y}}\frac{1}{L})^\perp)U\underline{u} = \frac{U^2}{L}(\underline{u}\cdot\underline{\nabla})\underline{u}$$

$$\nabla p = (\frac{\partial}{\partial x},\frac{\partial}{\partial y})^\perp p = (\frac{\partial}{\partial \underline{x}}\frac{1}{L},\frac{\partial}{\partial \underline{y}}\frac{1}{L})\frac{\nu U}{L}\underline{p} = \frac{\nu U}{L^2}\underline{\nabla p}$$

$$\triangle u = (\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2})u = (\frac{\partial^2}{\partial \underline{x}^2}\frac{1}{L^2}+\frac{\partial^2}{\partial \underline{y}^2}\frac{1}{L^2})U\underline{u} = \frac{U}{L^2}\underline{\triangle u}$$

Por tanto, reescribiendo la ecuación se tiene

$$R\underline{\theta}\frac{U^2}{L}(\underline{u_t}+(\underline{u}\cdot\underline{\nabla})\underline{u}) + \frac{\nu U}{L^2}\underline{\nabla p} - \frac{U}{L^2}\nu\underline{\triangle u} = -(0, gR\underline{\theta})^\perp$$

Multiplicando la ecuación por $\frac{L^2}{\nu U}$ se obtiene:

$$R\underline{\theta}\frac{UL}{\nu}(\underline{u_t}+(\underline{u}\cdot\underline{\nabla})\underline{u}) + \underline{\nabla p} - \underline{\triangle u} = -(0, \frac{gL^2}{\nu U}R\underline{\theta})^\perp$$

A continuación, se denotan

$$\frac{1}{Pr} = R\frac{UL}{\nu} \Longrightarrow \left[\frac{1}{Pr}\right] = \left[R\frac{UL}{\nu}\right] = \frac{ML^2T}{L^2TM}$$

$$Ra = \frac{gL^2}{\nu U}R \Longrightarrow \left[Ra\right] = \left[\frac{gL^2}{\nu U}R\right] = \frac{LML^2TT}{T^2L^2ML}$$

Que son los llamados números de Prandtl y Rayleigh respectivamente y que claramente se ha comprobado que son adimensionales.

$$\frac{1}{Pr}\underline{\theta}(\underline{u_t}+(\underline{u}\cdot\underline{\nabla})\underline{u}) + \underline{\nabla p} - \underline{\triangle u} = -(0, Ra\underline{\theta})^\perp \tag{2.16}$$

Como se ha comentado previamente en las ecuaciones de Stokes están dominadas por la viscosidad lo que supone que se considere un número de Prandtl grande, de esta forma el primer término de la ecuación (2.16). Lo que queda entonces es la ecuación de conservación del momento, que junto a la condición de divergencia y la ecuación de conservación de la masa de (2.17) conforman la ecuación de Stokes que se puede ver en [5]:

$$\begin{cases} -\triangle u = -\nabla p - (0, \theta) \\ \nabla \cdot u = 0 \\ \theta_t + u \cdot \nabla \theta = 0 \end{cases} \tag{2.17}$$

Como en el caso anterior, se expresa la velocidad en función de θ usando las dos primeras ecuaciones de (2.17). Utilizando la misma propiedad que para IPM, existe ψ con $u = \nabla^{\perp}\psi$ y entonces

$$-\nabla \times \triangle \nabla^{\perp}\psi = 0 - \nabla \times (0, \theta) \Longleftrightarrow -\triangle\triangle\psi = -\theta_x \Longrightarrow u = -\nabla^{\perp}(-\triangle\triangle)^{-1}\theta_x$$

De esta forma se puede escribir esta ecuación en su forma de escalar activo

$$\begin{cases} \theta_t + u \cdot \nabla \theta = 0 \\ u = -\nabla^{\perp}(-\triangle\triangle)^{-1}\theta_x \end{cases} \tag{2.18}$$

donde $\widehat{(-\triangle\triangle)^{-1}u} = \frac{1}{-|\varepsilon|^4}\hat{u}$ para cualquier función u.

2.1.5. Escalares activos en una dimensión

Para realizar las simulaciones de los escalares activos el programa a realizar puede resultar complejo. Por ello, se realiza un programa más sencillo que simule escalares activos en una dimensión. En particular, se trabajará con la conocida ecuación de Burgers y la ecuación de Córdoba-Córdoba-Fontelos.

Burgers

La ecuación de Burgers es una ecuación diferencial en derivadas parciales fundamental en varias áreas de las matemáticas. Sus aplicaciones van desde mecánica de fluidos, dinámica de gases, acústica no lineal y el flujo del tráfico. El desarrollo de esta ecuación comienza en 1915, cuando Harry Bateman, un matemático inglés, introdujo una ecuación diferencial parcial que más tarde, en 1948, fue utilizada por Johannes Martinus Burgers para explicar el modelo matemático de la turbulencia. Para honrar las contribuciones de este físico holandés la ecuación recibió su nombre. La ecuación de Burgers que se va a simular en este artículo es la siguiente

$$u_t = -u_x u \tag{2.19}$$

Córdoba-Córdoba-Fontelos (CCF)

Está ecuación no lineal, no local fue propuesta por A. Córdoba, D. Córdoba y M. Fontelos como la análoga en una dimensión a la SQG en dos dimensiones. Se dice que es no local pues en el término H hay una integral. La ecuación es (ver [17])

$$u_t = -Hu \cdot u_x \tag{2.20}$$

con $u = u(x, t)$ y Hu es la transformada de Hilbert que se define como

$$Hu = \frac{1}{\pi} PV \int_{-\infty}^{\infty} \frac{u(\tau)}{x - \tau} d\tau.$$

Capítulo 3

Cálculo numérico de soluciones para EDOs

Como se ha comentado previamente, los escalares activos son ecuaciones diferenciales en derivadas parciales. En esta sección se va a hablar sobre métodos numéricos para la aproximación de soluciones de ecuaciones diferenciales ordinarias. Más adelante, se explicará como pasar de una ecuación diferencial en derivadas parciales a una ecuación diferencial ordinaria y así poder aplicar los métodos que se verán a continuación.

En general, es difícil calcular soluciones para una ecuación diferencial $y'(t) = f(t, y)$. Aunque se conozcan algunas soluciones concretas, este es uno de los problemas más estudiados a lo largo de la historia puesto que las ecuaciones diferenciales son fundamentales en muchos ámbitos como ingeniería, física, economía y biología, donde se usan para modelar situaciones de la vida real. El primero en realizar una aproximación de las soluciones de una ecuación diferencial fue Euler.

3.1. Método de Euler

El método de Euler es el método numérico más simple que aproxima soluciones de ecuaciones diferenciales ordinarias. Por esto, no es uno de los más utilizados en la práctica, pero su simpleza hace que sea el más usado para entender los demás métodos de los que se hablará más adelante. En este estudio se implementará este método en la computación a modo de verificación de que el programa realizado funcione sabiendo que las simulaciones obtenidas no serán muy fiables.

Este método fue propuesto inicialmente por Leonhard Euler en su libro *Institutionum calculi integralis* que publicó entre 1768 y 1770. Se basa en que dado un punto inicial X_0, usando la ecuación diferencial $y(t) = f(t, y)$ se puede saber la pendiente de la curva en ese punto. A continuación, se toma un punto X_1 cercano al X_0 en la recta que determina la pendiente. Este punto estará cerca de la curva real por lo que podemos suponer que está sobre ella y realizar el mismo procedimiento que para X_0. Reiterando este proceso obtendremos una aproximación de la curva que es solución de la ecuación diferencial como se puede ver en la figura 3.1.

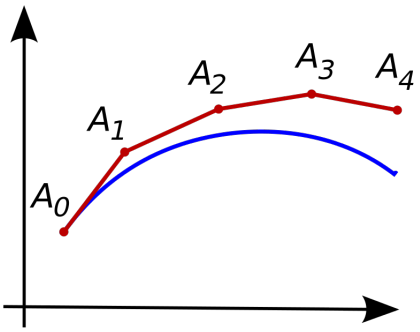

Figura 3.1: Dibujo del método de Euler. Ver en: https://en.wikipedia.org/wiki/Euler_method

Una vez vista la idea geométrica del método se va a dar una derivación del mismo a partir del teorema de Taylor.

Teorema de Taylor

Este teorema, que lleva el nombre del matemático que enunció una versión del mismo en 1715, proporciona una aproximación de una función diferenciable en un entorno de un punto mediante funciones polinómicas. Es uno de los conceptos elementales en el análisis matemático y una de las herramientas más usadas en este ámbito. Como podemos ver en [16] el teorema de Taylor se enuncia de la siguiente forma:

Sea f una función diferenciable n veces en un entorno del punto $a \in \mathbb{R}$ entonces se puede expresar f como

$$f(x) = \sum_{k=0}^{n} \frac{f^(k)(a)}{k!}(x-a)^k + E_n(x).$$

Donde $f^(k)(a)$ indica la derivada k-esima de f en el punto a y $E_n(x)$ es una función que toma como valor el error que se comete al realizar la aproximación por polinomios.

Para la derivación del método se considera la ecuación diferencial $y'(t) = f(t, y)$ y el valor inicial $y(t_0) = y_0$. La función y tiene dos derivadas continuas para t con $t_0 \leq t \leq t_1$. Se van a aproximar en este intervalo N puntos igualmente espaciados donde toma valores la función y. Se selecciona un paso h y se calculan los puntos para $t_0 + hn$ con $n = 0, 1, ..., N-1$. Usando el teorema de Taylor se tiene

$$y(t_{n+1}) = y(t_n) + (t_{n+1} - t_n)y'(t_n) + \frac{(t_{n+1} - t_n)^2}{2}y''(\varepsilon_n)$$

donde $\varepsilon_n \in (t_n, t_{n+1})$. Tomando $h = t_{n+1} - t_n$ se obtiene

$$y(t_{n+1}) = y(t_n) + hy'(t_n) + \frac{h^2}{2}y''(\varepsilon_n).$$

Por último sustituyendo el valor de y'

$$y(t_{n+1}) = y(t_n) + hf(t_n, y(t_n)) + \frac{h^2}{2}y''(\varepsilon_n).$$

Entonces el método de Euler aproxima $y_n = y(t_n)$ y eliminando el término de la segunda derivada se obtiene la formulación del método de Euler (ver [14])

$$y_{n+1} = y_n + hf(t_n, y_n) \qquad n = 0, 1, ..., N-1. \tag{3.1}$$

Como se pude ver, este método no proporciona una solución de manera continua si no que se calcula una aproximación a la función deseada en determinados puntos. En caso de necesitar calcular uno de los

puntos intermedios se utilizan métodos de interpolación.

El error local en el método de Euler, que es el error cometido en un paso, es la diferencia entre la solución calculada numéricamente en el primer paso (y_1) y la solución exacta en el tiempo $t_1 = t_0 + h$. Usando de nuevo el desarrollo de Taylor y siempre y la tercera derivada de y sea acotada, la solución exacta es

$$y(t_1) = y(t_0) + hy'(t_0) + \frac{h^2}{2}y''(t_0) + O(h^3)$$

y haciendo la diferencia con (3.1) para $n = 0$ se obtiene que el error local es

$$EL = y(t_0 + h) - y_1 = \frac{1}{2}h^2 y''(t_0) + O(h^3).$$

Para un h lo suficientemente pequeño se puede ver que el error local es proporcional a h^2.

Por otro lado, el error global es el error cometido en un tiempo determinado t_n. Es el resultado de la acumulación de los errores locales cometidos en cada paso hasta llegar al tiempo t_n. Como el número de pasos es $\frac{t_n - t_0}{h}$ que es proporcional a $\frac{1}{h}$ y el error local es proporcional a h^2 se tiene que el error global es proporcional a h.

A partir de este algoritmo fueron surgiendo otros métodos como los Runge-Kutta que son una generalización del método de Euler.

3.2. Métodos de Runge-Kutta explícito de orden 4 (RK4)

Los métodos Runge-Kutta fueron desarrollados en un principio por C. Runge y M. W. Kutta entorno al año 1900 para mejorar el método de Euler. Son métodos genéricos e iterativos que resuelven ecuaciones diferenciales de forma numérica y pueden ser explícitos e implícitos. Los explícitos son aquellos en los que se calcula el siguiente elemento de la sucesión utilizando únicamente valores previamente calculados. En los implícitos es necesaria la resolución de un sistema de ecuaciones ya que los y_n dependen de ellos mismos. Como se puede ver en [4] la expresión general de un método Runge-Kuta para aproximar una EDO $y(t)' = f(t, y)$ tomando un paso $h > 0$ es:

$$\begin{cases} y_{n+1} = y_n + h\phi(t_n, y_n; h) \\ \phi(t, y; h) = \sum_{r=1}^{R} c_r k_r \\ k_1 = f(t, y) \\ k_r = f(t + ha_r, y + h\sum_{s=1}^{r-1} b_{rs}k_s), \qquad r = 2, 3, ..., R \\ a_r = \sum_{s=1}^{r-1} b_{rs}, \qquad r = 2, 3, ..., R \end{cases} \tag{3.2}$$

Donde y_n son las aproximaciones de la solución en el tiempo t_n. Los coeficientes c_r, b_{rs} y a_r a menudo se recogen en unas tablas llamadas tablero de Butcher, denominadas así por John C. Butcher matemático neozelandés especializado en el cálculo numérico. Estas matrices tienen la forma

$$\begin{array}{c|cccc} a_1 & b_{11} & b_{12} & \dots & b_{1R} \\ a_2 & b_{21} & b_{22} & \dots & b_{2R} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ a_R & b_{R1} & b_{R2} & \dots & b_{RR} \\ \hline & c_1 & c_2 & \dots & c_R \end{array}$$

El vector $(a_1, ..., a_R)$ se conoce como vector de nodos, el vector $(c_1, ..., c_R)$ como vector de pesos y a la matriz $B = (b_{ij})$ $i, j = 1, ...R$ como matriz de coeficientes. En función de estos coeficientes se tendrá un

método Runge-Kutta u otro, de forma que con el mismo algoritmo cambiando la matriz se pueden tener distintos métodos.

En particular, para la simulación numérica de los escalares activos se va a usar el conocido método RK4.

Uno de los métodos Runge-Kutta de uso más extendido es el RK4 puesto que es sencillo de implementar en un software y proporciona un error global de $O(h^4)$, siendo $h > 0$ el paso. Sea la ecuación diferencial $y'(t) = f(t, y)$ y el punto inicial $y(t_0) = y_0$, este método se calcula de la siguiente forma:

$$\begin{cases} y_{n+1} = y_n + h + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\ t_{n+1} = t_n + h \\ k_1 = f(t_n, y_n) \\ k_2 = f(t_n + \frac{h}{2}, y_n + h\frac{k_1}{2}) \\ k_3 = f(t_n + \frac{h}{2}, y_n + h\frac{k_2}{2}) \\ k_4 = f(t_n + h, y_n + hk_3) \end{cases} \tag{3.3}$$

Donde y_{n+1} es el valor aproximado de la función en el punto t_{n+1} y viene dado por la suma de el anterior valor (y_n) más el producto del paso (h) por una pendiente que viene determinada por $k_1,\ k_2,\ k_3$, y k_4. k_1 es la pendiente en el punto anterior al que se va a calcular en la iteración, se corresponde con el método de Euler. k_2 es la pendiente en el punto medio entre t_n y t_{n+1} usando k_1 para determinar el valor de y en ese punto. k_3 es la pendiente en el punto medio pero usando k_2 para determinar el valor de y. Por último, k_4 es la pendiente en t_{n+1} usando k_3 para determinar el valor de y.

El tablero de Butcher del método RK4 es el siguiente

0				
1/2	1/2			
1/2	0	1/2		
1	0	0	1	
	1/6	1/3	1/3	1/6

Este método realiza cuatro evaluaciones por paso por lo que las aproximaciones serán mejores que en el método de Euler habiendo tomado como paso un cuarto del paso de RK4. Por esllo, en este estudio se implementará este método para la simulación de los escalares activos esperando obtener soluciones con un error suficientemente pequeño.

Capítulo 4

Transformada rápida de Fourier (FFT)

Otra de las herramientas necesarias para implementar un programa que simule los escalares activos es la transformada rápida de Fourier. Dado que los métodos Runge-Kutta resuelven ecuaciones diferenciales ordinarias y en el caso de los escalares activos lo que se tiene son ecuaciones diferenciales en derivadas parciales se necesita algún método para pasar de EDP a EDO. Para esto, se van a utilizar las series de Fourier cuya implementación computacional se lleva a cabo a través de la transformada rápida de Fourier. Al aplicarla a los escalares activos, donde antes se tenían derivadas respecto de x e y, en el espacio de Fourier, se tienen multiplicadores por lo que se ha transformado una EDP en una EDO. Por ejemplo, tomando la primera ecuación de (2.5) se tiene que

$$\theta_t + u \cdot \nabla\theta = 0$$

(suponiendo que u es una constante) es una EDP con derivadas en t, x e y. Pero, si aplicamos la transformada de Fourier

$$\hat{\theta}_t + u \cdot (-i\varepsilon_2, i\varepsilon_1) \cdot \hat{\theta} = 0$$

es una EDO ya que solo hay derivadas respecto del tiempo.

4.1. Series de Fourier

Las series de Fourier son una expansión de una función periódica en una suma de funciones trigonométricas. De esta forma se pueden estudiar muchos de los problemas de forma más sencilla. Aunque hemos dicho que las series de Fourier son suma de funciones trigonométricas, para este artículo se van a expresar en su forma de exponencial imaginaria. Por tanto, para una función $u(x)$ $2L$-periódica se utiliza la siguiente formulación (ver [13]):

$$\hat{u}(x) = \sum_{l=-\infty}^{\infty} \frac{1}{2L}\left(\int_{-L}^{L} u(z)e^{-i\frac{\pi}{L}zl}dz\right)e^{i\frac{\pi}{L}xl} \tag{4.1}$$

Como se van a simular escalares activos en dos dimensiones es fácil extender (4.1) de forma que para una función $u(x, y)$ $2L$-periódica en ambas variables se tiene que la serie de Fourier es

$$\hat{u}(x, y) = \sum_{l=-\infty}^{\infty}\sum_{m=-\infty}^{\infty} \frac{1}{2L}\left(\int_{-L}^{L}\int_{-L}^{L} u(z_1, z_2)e^{-i\frac{\pi}{L}(z_1 l + z_2 m)}dz_1 dz_2\right)e^{i\frac{\pi}{L}(xl+ym)} \tag{4.2}$$

Ya hemos dicho que las series de Fourier se utilizarán para calcular derivadas, por lo que es necesario conocer como son las series de Fourier de la derivada de una función. Sea $u(x, y)$ de nuevo una función $2L$-periódica en ambas variables, se tiene

$$\hat{u}(x, y)_{x^k} = \left(i\frac{\pi}{L}l\right)^k \hat{u}(x, y). \tag{4.3}$$

Y de manera análoga para la variable y o para el caso en una dimensión. Es importante observar que hemos pasado de una derivada a un multiplicador.
Demostración: Se sigue de las siguientes igualdades

$$\begin{aligned}
&\hat{u}(x,y)_{x^k} \\
&\quad = \frac{\partial^k}{\partial x^k}\left(\sum_{l-\infty}^{\infty}\sum_{m=-\infty}^{\infty}\frac{1}{2L}\left(\int_{-L}^{L}\int_{-L}^{L}u(z_1,z_2)e^{-i\frac{\pi}{L}(z_1 l+z_2 m)}dz_1 dz_2\right)e^{i\frac{\pi}{L}(xl+ym)}\right) \\
&\quad = \sum_{l=-\infty}^{\infty}\sum_{m=-\infty}^{\infty}\frac{1}{2L}\left(\int_{-L}^{L}\int_{-L}^{L}u(z_1,z_2)e^{-i\frac{\pi}{L}(z_1 l+z_2 m)}dz_1 dz_2\right)e^{i\frac{\pi}{L}(xl+ym)}i\frac{\pi}{L}l \\
&\quad = \left(i\frac{\pi}{L}l\right)^k\hat{u}(x,y).
\end{aligned} \tag{4.4}$$

□

En el caso de los escalares activos en dos dimensiones, para realizar la computación necesitamos trabajar en un espacio finito por lo que se considera el $\mathbb{T}^2$ y periodicidad en $(-\pi,\pi)$ tanto para la variable x como para y, es decir, $2L$-periodicidad con $L=\pi$.

4.2. Transformada de Fourier Discreta (DFT)

La transformada de Fourier discreta es la transformada discreta más importante. Se usa tanto en ámbito puramente matemático sin interpretación física como en, por ejemplo, procesamientos de señales. Cuando la función es discreta y finita en vez de usar la transformada de Fourier continua se usa esta otra que transforma una función matemática con dominio en espacio en otra con dominio en frecuencia.

Como se puede ver en [11] siendo una secuencia $a_n \in \mathbb{C}$ con $n = 0, 1, ..., N-1$ la transformada discreta de Fourier devuelve:

$$A_k = \sum_{n=0}^{N-1} a_n e^{-i2\pi\frac{k}{N}n} \tag{4.5}$$

También se suele escribir de la siguiente forma:

$$A_k = \sum_{n=0}^{N-1} a_n W_N^{kn} \tag{4.6}$$

donde W_N es $e^{\frac{-i2\pi}{N}}$.

4.2.1. Propiedades de la transformada de Fourier discreta

En esta sección las secuencias a_n y b_n tienen como DFT a A_k y B_k, respectivamente. Teniendo en cuenta esto, las propiedades más conocidas son las siguientes:

1. Es lineal, al igual que la transformada de Fourier continua, se cumple que dados α y $\beta \in \mathbb{C}$ entonces la DFT de $\alpha a_n + \beta b_n$ es $\alpha A_k + \beta B_k$.
2. Traslación en tiempo. Si $a_n = b_{n-n_0}$ entonces $A_k = B_k e^{-i2\pi\frac{k}{N}n_0}$.
3. Traslación en frecuencia. Si $a_n = b_n e^{i2\pi\frac{k_0}{N}n_0}$ entonces $A_k = B_{k-k_0}$.

Demostración:

1. Se sigue de las siguientes igualdades.

$$\sum_{n=0}^{N-1}\alpha a_n W_N^{kn} + \sum_{n=0}^{N-1}\beta b_n W_N^{kn} = \alpha\sum_{n=0}^{N-1}a_n W_N^{kn} + \beta\sum_{n=0}^{N-1}b_n W_N^{kn} = \alpha A_k + \beta B_k$$

2. De nuevo se demuestra por igualdades.

$$A_k = \sum_{n=0}^{N-1} a_n e^{-i2\pi \frac{k}{N} n} = \sum_{n=0}^{N-1} b_{n-n_0} e^{-i2\pi \frac{k}{N} n}$$

Haciendo el cambio de variable $m = n - n_0$:

$$\sum_{n=0}^{N-1} b_{n-n_0} e^{-i2\pi \frac{k}{N} n} = \sum_{m=-n_0}^{N-1-n_0} b_m e^{-i2\pi \frac{k}{N}(m+n_0)} = B_k e^{-i2\pi \frac{k}{N} n_0}$$

3. De forma similar al anterior resultado:

$$A_k = \sum_{n=0}^{N-1} a_n e^{-i2\pi \frac{k}{N} n} = \sum_{n=0}^{N-1} b_n e^{i2\pi \frac{k_0}{N} n} e^{-i2\pi \frac{k}{N} n} = \sum_{n=0}^{N-1} b_n e^{-i2\pi \frac{k-k_0}{N} n} = B_{k-k_0}$$

□

4.2.2. Transformada inversa de Fourier discreta

Al igual que en la transformada de Fourier continua se tiene el concepto de transformada inversa de Fourier discreta donde a_n es la secuencia inversa de A_k. Como se puede ver en [11] se define de la siguiente manera

$$a_n = \frac{1}{N} \sum_{k=0}^{N-1} W_N^{-kn} A_k \tag{4.7}$$

4.2.3. DFT en dos dimensiones

Es fácil ver que se puede extender (4.5) a dos dimensiones

$$A_{n,m} = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} a_{n,m} e^{-i2\pi(\frac{k}{N} n + \frac{l}{M} m)} \tag{4.8}$$

Aunque en el caso de los escalares activos siempre se van a realizar las mismas evaluaciones tanto en x como en y, es decir, se tiene $N = M$.

4.3. Algoritmo FFT

Una de las ventajas que tiene la DFT frente a la transformada de Fourier continua es, que al ser necesario su uso con datos finitos permite que su implementación en software sea posible. Esto es muy útil en distintos campos pero a menudo, su implementación directa, usando la definición, es muy lenta para la práctica. Por ello, la forma más extendida de implementarla es usando el algoritmo FFT.

La transformada de Fourier rápida es un algoritmo utilizado en ingeniería, matemáticas, música y otros muchos ámbitos. Fue publicado en 1965 por James W. Cooley y John W. Tukey. Su uso está altamente extendido. Algunas de las aplicaciones son el diseño de circuitos, espectroscopia, tratamiento de imágenes y otros muchos.

Este algoritmo consiste en tomar la fórmula (4.5) en forma matricial y factorizar la matriz en productos de matrices *sparce*, es decir, en matrices cuyos elementos mayoritariamente son cero. Usar directamente la fórmula (4.5) para una secuencia con N números significaría tener que usar $O(N^2)$ multiplicaciones y sumas. Realizando FFT se computa la DFT usando $O(Nlog(N))$ multiplicaciones y sumas lo que supone un gran ahorro de tiempo computacional para N lo suficientemente grande.

A continuación se expone un ejemplo de FFT usando la factorización matricial en la DFT de N=8 puntos visto en [11]. Se va usar la notación $W = W_8 = e^{\frac{-i\pi}{4}}$. Usando la definición (4.5) y reescribiendo los calculos en forma matricial se obtiene:

$$\begin{pmatrix} A_0 \\ A_1 \\ A_2 \\ A_3 \\ A_4 \\ A_5 \\ A_6 \\ A_7 \end{pmatrix} = \begin{pmatrix} W^0 & W^0 & W^0 & W^0 & W^0 & W^0 & W^0 & W^0 \\ W^0 & W^1 & W^2 & W^3 & W^4 & W^5 & W^6 & W^7 \\ W^0 & W^2 & W^4 & W^6 & W^0 & W^2 & W^4 & W^6 \\ W^0 & W^3 & W^6 & W^1 & W^4 & W^7 & W^2 & W^5 \\ W^0 & W^4 & W^0 & W^4 & W^0 & W^4 & W^0 & W^4 \\ W^0 & W^5 & W^2 & W^7 & W^4 & W^1 & W^6 & W^3 \\ W^0 & W^6 & W^4 & W^2 & W^0 & W^6 & W^4 & W^2 \\ W^0 & W^7 & W^6 & W^5 & W^4 & W^3 & W^2 & W^1 \end{pmatrix} \begin{pmatrix} a0 \\ a1 \\ a2 \\ a3 \\ a4 \\ a5 \\ a6 \\ a7 \end{pmatrix}$$

Reordenando las columnas ($2 \longleftrightarrow 5$,$4 \longleftrightarrow 7$):

$$\begin{pmatrix} A_0 \\ A_1 \\ A_2 \\ A_3 \\ A_4 \\ A_5 \\ A_6 \\ A_7 \end{pmatrix} = \begin{pmatrix} W^0 & W^0 & W^0 & W^0 & W^0 & W^0 & W^0 & W^0 \\ W^0 & W^4 & W^2 & W^6 & W^1 & W^5 & W^3 & W^7 \\ W^0 & W^0 & W^4 & W^4 & W^2 & W^2 & W^6 & W^6 \\ W^0 & W^4 & W^6 & W^2 & W^3 & W^7 & W^1 & W^5 \\ W^0 & W^0 & W^0 & W^0 & W^4 & W^4 & W^4 & W^4 \\ W^0 & W^4 & W^2 & W^6 & W^5 & W^1 & W^7 & W^3 \\ W^0 & W^0 & W^4 & W^4 & W^6 & W^6 & W^2 & W^2 \\ W^0 & W^4 & W^6 & W^2 & W^7 & W^3 & W^5 & W^1 \end{pmatrix} \begin{pmatrix} a0 \\ a1 \\ a2 \\ a3 \\ a4 \\ a5 \\ a6 \\ a7 \end{pmatrix}$$

Y factorizando para obtener productos de las siguientes matrices *sparce*:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & W^0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & W^1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & W^2 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & W^3 \\ 1 & 0 & 0 & 0 & W^4 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & W^5 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & W^6 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & W^7 \end{pmatrix}, \begin{pmatrix} 1 & 0 & W^0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & W^2 & 0 & 0 & 0 & 0 \\ 1 & 0 & W^4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & W^6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & W^0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & W^2 \\ 0 & 0 & 0 & 0 & 1 & 0 & W^4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & W^6 \end{pmatrix}$$

$$\begin{pmatrix} 1 & W^0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & W^4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & W^0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & W^4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & W^0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & W^4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & W^0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & W^4 \end{pmatrix}$$

Para $N = 2^r$ se obtendrían r matrices de tamaño $N \times N$ y con dos elementos distintos de cero en cada columna y en cada fila. Este algoritmo se utiliza siempre para $N = 2^r$ con $r \in \mathbb{N}$ ya que el algoritmo divide la secuencia introducida en subsecuencias más pequeñas, así cuando se introduce una secuencia de tamaño una potencia de dos el proceso de división en subsecuencias es mucho fácil de realizar y repetir hasta que las subsecuencias son lo suficientemente pequeñas para computar el algoritmo FFT de manera eficiente.

Otra forma de estudiar el procesamiento del algoritmo es usando un diagrama como el que se muestra en la figura 4.1.

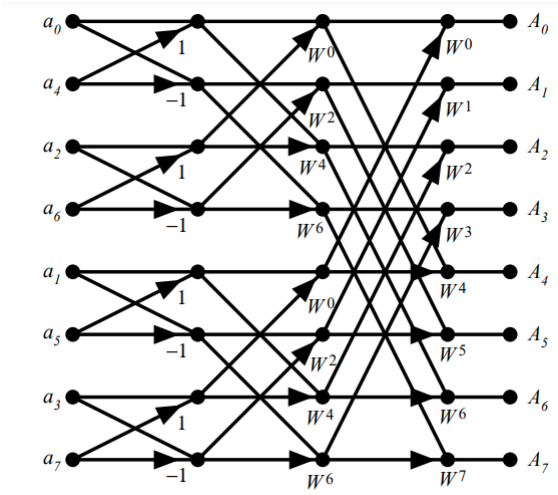

Figura 4.1: Diagrama de las operaciones realizadas en un algoritmo FFT con N=8. Ver en: https://www.cs.cmu.edu/afs/andrew/scs/cs/15-463/2001/pub/www/notes/fourier/fourier.pdf

En este diagrama se observan las conocidas como computaciones *butterfly* que consisten en tomar dos números complejos z, w y calcular otros dos números $w + \alpha z$ y $w - \alpha z$, con $\alpha \in \mathbb{C}$. El FFT descompone la transformada discreta de Fourier en $\log_2(N)$ etapas en las que se realizan $\frac{N}{2}$ computaciones *butterfly*.

Para los escalares activos en una dimensión en Matlab usaremos el algoritmo mediante el comando fft y su inversa ifft. En el caso de dos dimensional necesitaremos hacer tanto la transformada bidimensional de Fourier como la transformada inversa bidimensional. En Matlab, se usa para ello el algoritmo fft2 e ifft2. Calcular la fft2(M) de una matriz M es equivalente a utilizar la fft de la siguiente forma: fft(fft(M).')' .

4.4. Aplicación a los escalares activos

Usando las series de Fourier conseguiremos escribir los escalares activos que son ecuaciones diferenciales parciales como ecuaciones diferenciales ordinarias. Lo que ocurre es que al aplicar la serie de Fourier escribimos todas las derivadas parciales que no son respecto de la variable tiempo en forma de multiplicadores. Aunque es importante tener en cuenta que si se pasa toda la ecuación al espacio de Fourier, las multiplicaciones pasan a ser convoluciones que son muy caras de calcular numéricamente. Por esto, escribiremos las ecuaciones de tal forma que hagamos las derivadas en el espacio de Fourier y las multiplicaciones en el espacio real. Por ejemplo, tomando la ecuación de Euler (2.10) y utilizando la propiedad (4.3) sobre la derivada de la transformada de Fourier ($\partial x = i\varepsilon_1$ y $\partial y = i\varepsilon_2$) se tiene que:

$$\begin{cases} \hat{\theta}_t = (i\varepsilon_1, i\varepsilon_2) \cdot fft2(real(ifft2(\hat{u})) \cdot real(ifft2(\hat{\theta}))) \\ \hat{u} = \frac{(-i\varepsilon_2, i\varepsilon_1)}{|\varepsilon|^2}\hat{\theta} \end{cases} \tag{4.9}$$

Donde fft2 indica la tarnsformada de Fourier e ifft2 la transformada inversa ambas en dos dimensiones. Si se toma la ecuación de superficie cuasigesostrofica (2.12) y se le aplica la serie de Fourier se tiene:

$$\begin{cases} \hat{\theta}_t = (i\varepsilon_1, i\varepsilon_2) \cdot fft2(real(ifft2(\hat{u})) \cdot real(ifft2(\hat{\theta}))) \\ \\ \hat{u} = \frac{(-i\varepsilon_2, i\varepsilon_1)}{|\varepsilon|}\hat{\theta} \end{cases} \tag{4.10}$$

Siguiendo el mismo procedimiento con la ecuación del medio incompresible poroso (2.14) se obtiene:

$$\begin{cases} \hat{\theta}_t = (i\varepsilon_1, i\varepsilon_2) \cdot fft2(real(ifft2(\hat{u})) \cdot real(ifft2(\hat{\theta}))) \\ \hat{u} = \frac{(i\varepsilon_2, -i\varepsilon_1)}{|\varepsilon|^2} i\varepsilon_1 \hat{\theta} \end{cases} \tag{4.11}$$

Para la ecuación de Stokes (2.18):

$$\begin{cases} \hat{\theta}_t = (i\varepsilon_1, i\varepsilon_2) \cdot fft2(real(ifft2(\hat{u})) \cdot real(ifft2(\hat{\theta}))) \\ \hat{u} = \frac{(i\varepsilon_2, -i\varepsilon_1)}{-|\varepsilon|^4} i\varepsilon_1 \hat{\theta} \end{cases} \tag{4.12}$$

Para Burgers se usará

$$u_t = -i\varepsilon \frac{\widehat{(u^2)}}{2}. \tag{4.13}$$

Y por último, para CCF

$$\hat{u}_t = fft(real(ifft(-\widehat{Hu})) \cdot real(ifft(i\varepsilon\hat{u}))) \tag{4.14}$$

donde $-\widehat{Hu} = -i\frac{\varepsilon}{|\varepsilon|}\hat{u}$.

Capítulo 5

Simulaciones

En este capítulo, se verán las simulaciones numéricas de los escalares activos. Se comienza resolviendo un sistema de EDO para luego pasar a los escalares activos en una dimensión y finalmente, en dos dimensiones. También se buscan soluciones estacionarias para algunos de los escalares activos en dos dimensiones de manera que se pueda verificar el correcto funcionamiento de los programas para esos ejemplos.

5.1. Simulación numérica de Lotka-Volterra

Lo primero que se ha hecho es resolver un sistema de EDO para comprobar que los métodos Runge-Kutta funcionan correctamente. El método de Euler sirve para ver que funciona de forma sencilla mientras que con el RK4 se consigue una mejor aproximación de la solución del sistema. Se ha resuelto el sistema conocido como Lotka-Volterra. Este sistema consta de dos ecuaciones diferenciales de primer orden no lineales utilizadas para estudiar la interacción entre dos especies, una especie depredadora y la otra presa. El sistema Lotka-Volterra se puede escribir de forma general como (ver [9])

$$\begin{cases} x_t = \alpha x - \beta xy \\ y_t = \delta xy - \gamma y \\ \alpha > 0, \beta > 0, \delta > 0, \gamma > 0 \end{cases} \tag{5.1}$$

Donde x representa el número individuos de la población de presas e y el número de individuos de la población de depredadores. α es la tasa de crecimiento de las presas en ausencia de depredadores, β es la tasa a la que los depredadores comen a las presas, δ es la tasa de crecimiento de los depredadores en presencia de presas y γ es la tasa de muertes de los depredadores en ausencia de presas.

A continuación se muestra el código utilizado para la simulación tomando como valores iniciales dos individuos presa y un depredador y con tasas de crecimiento iguales a 1 excepto la β que será igual a 0,1. Se proporciona únicamente el código usando RK4 puesto que ambos integradores temporales funcionaban correctamente y proporcionaban resultados similares.

```
a=0;
b=50;
N=1000;
y0=[2,1];

%RK4:
h=(b-a)/N;

t=zeros(N+1,1);
y=zeros(N+1,length(y0));

t(1)=a;
```

```
y(1,:)=y0;

for i=1:N
    ti=t(i);
    yi=y(i,:);

    k1=f(ti,yi);
    k2=f(ti+h/2,yi+h/2*k1);
    k3=f(ti+h/2,yi+h/2*k2);
    k4=f(ti+h,yi+h*k3);

    t(i+1)=t(i)+h;
    y(i+1,:)=y(i,:)+h/6*(k1+2*k2+2*k3+k4);
end

plot(y)

function dydt2=f(t, y)
    dydt2=zeros(1,2);
    dydt2(1)=1*y(1)-0.1*y(1)*y(2);
    dydt2(2)=1*y(1)*y(2)-1*y(2);
end
```

El resultado obtenido es

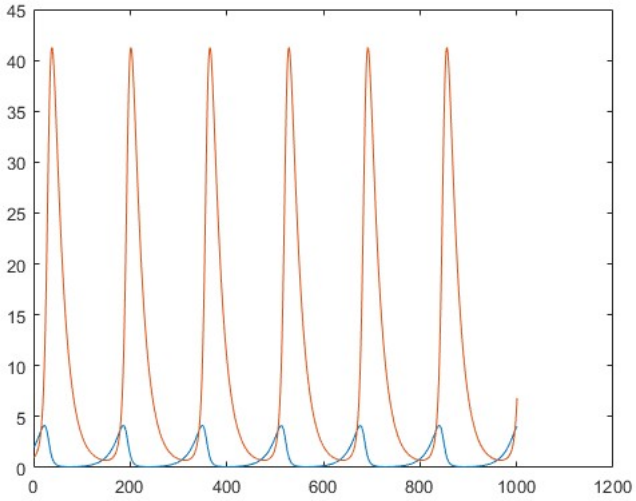

Figura 5.1: Resultado de la simulación del sistema Lotka-Volterra

En la figura 5.1 se puede ver como cuando la población de depredadores (gráfica en rojo) es casi cero empiezan a crecer la población de presas (gráfica en azul). Cuando el número de presas alcanzo un cierto valor vuelven a aparecer los depredadores y por tanto empieza a bajar el número de presas hasta que casi desaparecen. El resultado obtenido concuerda con lo que cabría esperar del modelo de Lotka-Volterra.

5.2. Dimensión 1

Como se ha comentado antes se ha comenzado con un programa que simule los escalares activos en una dimensión. Para ello, se adapta el integrador temporal del programa anterior de manera que tanto el

método de Euler como el RK4 se utilizan para aproximar la transformada de Fourier de la función escalar del escalar activo. La ecuación se escribe de tal forma que las derivadas se realizan en el espacio de Fourier pero las multiplicaciones en el espacio real. Esto es porque, como se ha comentado previamente, se necesita pasar al espacio de Fourier y así tener multiplicadores en vez de derivadas, pero las multiplicaciones entre funciones en el espacio de Fourier son convoluciones que son muy caras de realizar en la computación. La derivada respecto de x en Fourier es una vector con los M coeficientes de la derivada de Fourier. En Matlab este vector no se ordena de la forma que se pensaría naturalmente si no que se empieza con el cero seguido de los coeficientes positivos en orden ascendente, de nuevo el cero y, por último, los coeficientes negativos en orden ascendente. Así, para $M = 8$ se tendría que la derivada respecto de x sería multiplicar por el vector $i \cdot (0, 1, 2, 3, 0, -3, -2, -1)$.

5.2.1. Simulación numérica de Burgers

Se va a simular en primer lugar la ecuación de Burgers. Es importante destacar la manera en la que se ha definido el espacio donde se realizan las simulaciones. Es necesario hacer una identificación de el primer punto (en este caso $-\pi$) con el último (en este caso π). Esto es un error que se cometió al principio en el desarrollo del programa lo que provoco la aparición de un fenómeno de Gibbs. Este fenómeno consiste en el comportamiento oscilante en las series de Fourier ocasionado por un salto de discontinuidad lo que producía imprecisiones a la hora de calcular las derivadas en el espacio de Fourier provocando oscilaciones de gran tamaño en los extremos de la función como se puede ver en la figura 5.2.

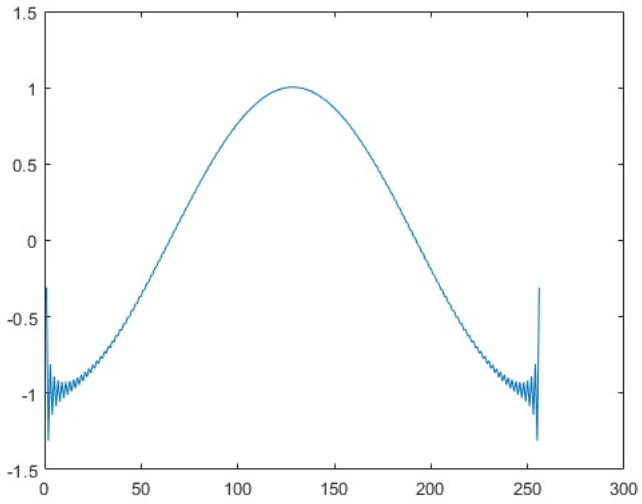

Figura 5.2: Fenómeno de Gibbs observado en la derivada de $sin(x)$ al definir el espacio sin tener en cuenta la periodicidad.

Se empieza usando como integrador temporal el método de Euler para verificar que el programa funciona. Se toma como valor inicial el $-sen(x)$ entre $-\pi$ y π, el número de evaluaciones que realiza el algoritmo FFT es $M = 256$ y el integrador temporal realiza $N = 1000$ evaluaciones desde el tiempo $t = 0$ hasta $t = 5$.

```
a=0;
b=5;
N=1000;
M=256;
x=(2*pi/M)*(-M/2:M/2-1);
u0_hat=fft(-sin(x));
```

```
%Euler forward:
h=(b-a)/N;

t=zeros(N+1,1);
u_hat=zeros(N+1,M);

t(1)=a;
u_hat(1,:)=u0_hat;

for i=1:N
    ti=t(i);
    ui_hat=u_hat(i,:);
    t(i+1)=t(i)+h;
    u_hat(i+1,:)=u_hat(i,:)+h*f(ti,ui_hat);
end
u=real(ifft(u_hat')); %se vuelve a pasar la solucion al real

for i=1:N+1
    plot(u(:,i))
    drawnow
end

function dudt = f(t, u_hat)
    M=256;
    D=1i*[0:M/2-1,0,-M/2+1:-1];
    dudt=-D.*fft(real(ifft(u_hat)).^2)/2;
end
```

En este caso se debe estudiar que ocurre según evoluciona el tiempo.

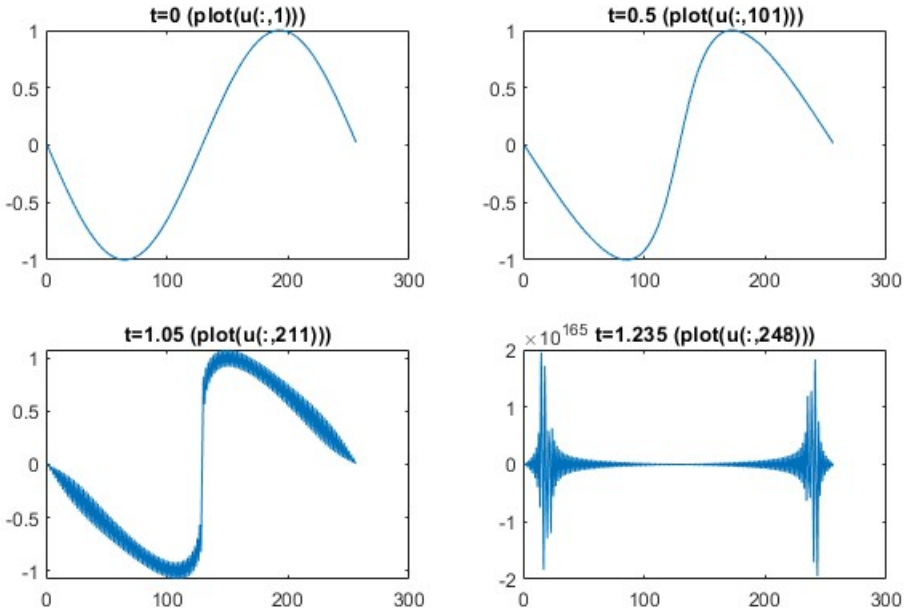

Figura 5.3: Resultado de la simulación de la ecuación de Burgers con Euler.

Como se ha introducido como función inicial la función $f(x) = -sen(x)$ se tiene que el programa devuelve para $t = 0$ lo visto en la figura 5.3. A continuación, se produce un crecimiento de la derivada, por ejemplo, para $t = 0{,}5$ se obtiene la figura de 5.3 donde se observa este crecimiento. Más adelante, empezarán a aparecer más oscilaciones debido a que le derivada alcanza un valor muy grande lo que provoca que el ordenador no procese bien esos números altos y genere esas inestabilidades que se pueden ver para $t = 1{,}05$. A partir del tiempo $t = 1{,}24$ la derivada se va a infinito por lo que el programa devuelve valores Nan. El último tiempo que dibuja es $t = 1{,}235$ Para ver el vídeo completo de la simulación: https://youtu.be/MdVVN9ZsxSY

El siguiente programa simula también la ecuación de Burgers con los mismos datos que el caso anterior pero usando como integrador temporal el método RK4.

```
a=0;
b=5;
N=1000;
M=256;
x=(2*pi/M)*(-M/2:M/2-1);
u0_hat=fft(-sin(x));

h=(b-a)/N;
t=zeros(N+1,1);
u_hat=zeros(N+1,length(u0_hat));
t(1)=a;
u_hat(1,:)=u0_hat;

for i=1:N
    ti=t(i);
    ui_hat=u_hat(i,:);

    k1=f(ti,ui_hat);
    k2=f(ti+h/2,ui_hat+h/2*k1);
    k3=f(ti+h/2,ui_hat+h/2*k2);
    k4=f(ti+h,ui_hat+h*k3);

    t(i+1)=t(i)+h;
    u_hat(i+1,:)=u_hat(i,:)+h/6*(k1+2*k2+2*k3+k4);
end
u=real(ifft(u_hat'));

for i=1:N+1
    plot(u(:,i))
    t(i)
    drawnow
end

function dudt = f(t, u_hat)
    M=256;
    D=1i*[0:M/2-1,0,-M/2+1:-1];
    dudt=-D.*fft(real(ifft(u_hat)).^2)/2;
end
```

La simulación obtenida para los tiempos indicados ha sido la figura 5.4

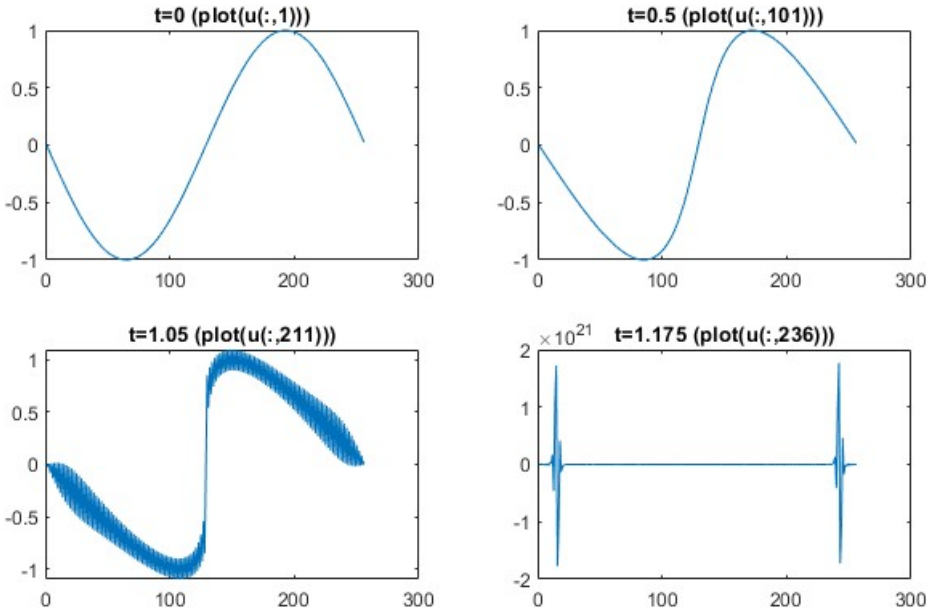

Figura 5.4: Simulación numérica de la ecuación de Burgers usando RK4.

Al igual que antes se ve para $t = 0$ el $-sin(x)$, luego comienza a crecer la derivada ($t = 0{,}5$) y ya para $t = 1{,}05$ aparecen las oscilaciones. En este caso el último tiempo que proporciona visualización es $t = 1{,}175$. Para ver la simulación completa: https://youtu.be/0CL1u-melyA

5.2.2. Simulación numérica de Córdoba-Córdoba-Fontelos

Se estudia ahora la simulación numérica de la ecuación CCF. De nuevo, se empieza utilizando como integrador temporal Euler. El valor inicial vuelve a ser $-sin(x)$ para $x \in [-\pi, \pi]$, $M = 256$ son las evaluaciones de FFT y $N = 1000$ son las evaluaciones del método de Euler. La simulación se realiza para $t \in [0, 5]$.

```
a=0;
b=5;
N=1000;
M=256;
x=(2*pi/M)*(-M/2:M/2-1);
u0_hat=fft(-sin(x));

%Euler:
h = (b - a) / N;

t = zeros(N+1, 1);
u_hat = zeros(N+1, M);

t(1) = a;
u_hat(1, :) = u0_hat;

for i = 1:N
    ti=t(i);
    ui_hat=u_hat(i, :);
```

```
    t(i+1) = t(i) + h;
    u_hat(i+1, :) = u_hat(i, :) + h * f(ti, ui_hat);
end
u=real(ifft(u_hat'));

%Representacion:
for i=1:N+1
    plot(u(:,i))
    drawnow
end

function dudt = f(t, u_hat)
    M=256;
    D=1i*[0:M/2-1,0,-M/2+1:-1];
    D(1)=1;
    D(M/2+1)=1;
    dudt=fft(real(ifft(-D./abs(D).*u_hat)).*(real(ifft(D.*u_hat))));
end
```

Algunas de las visualizaciones que se han obtenido son:

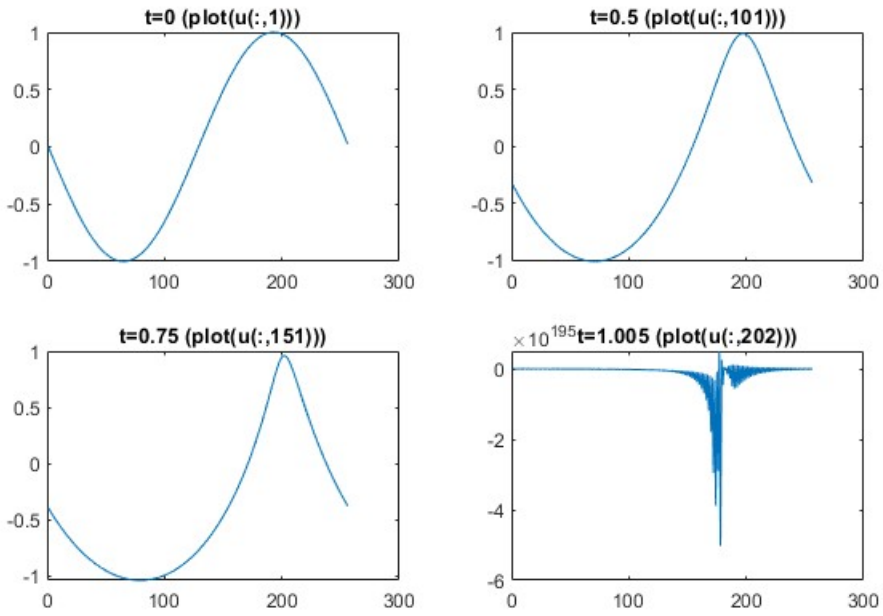

Figura 5.5: Simulación numérica de la ecuación de Córdoba usando Euler.

Se visualiza en primer lugar el $-sin(x)$ (figura 5.5). Como en el caso de Burgers se debe observar un crecimiento de la derivada lo cual se puede ver comparando la figura para $t = 0$, $t = 0{,}5$ y $t = 0{,}75$. A medida que aumenta el tiempo se empiezan a visualizar más oscilaciones hasta que el programa devuelve valores Nan como en los casos anteriores. De nuevo, estas inestabilidades son provocadas por valores muy grandes para la derivada. La última visualización ocurre para $t = 1{,}005$. Para ver la simulación completa: https://youtu.be/hlLqtRk3wwE

Ahora, se muestra el programa para simular la ecuación de Córdoba-Córdoba-Fontelos usando como integrador temporal el método RK4 y los mismos datos que usando Euler.

```
a=0;
b=5;
N=1000;
M=256;
x=(2*pi/M)*(-M/2:M/2-1);
u0_hat=fft(-sin(x));

%RK4:
h=(b-a)/N;

t=zeros(N+1,1);
u_hat=zeros(N+1,length(u0_hat));

t(1)=a;
u_hat(1,:)=u0_hat;

for i=1:N
    ti=t(i);
    ui_hat=u_hat(i, :);

    k1=f(ti,ui_hat);
    k2=f(ti+h/2,ui_hat+h/2*k1);
    k3=f(ti+h/2,ui_hat+h/2*k2);
    k4=f(ti+h,ui_hat+h*k3);

    t(i+1)=t(i)+h;
    u_hat(i+1,:)=u_hat(i,:)+h/6*(k1+2*k2+2*k3+k4);
end
u=real(ifft(u_hat'));

%Representacion:
for i=1:N+1
    plot(u(:,i))
    drawnow
end

function dudt = f(t, u_hat)
    M=256;
    D=1i*[0:M/2-1,0,-M/2+1:-1];
    D(1)=1;
    D(M/2+1)=1;
    dudt=fft(real(ifft(-D./abs(D).*u_hat)).*(real(ifft(D.*u_hat))));
end
```

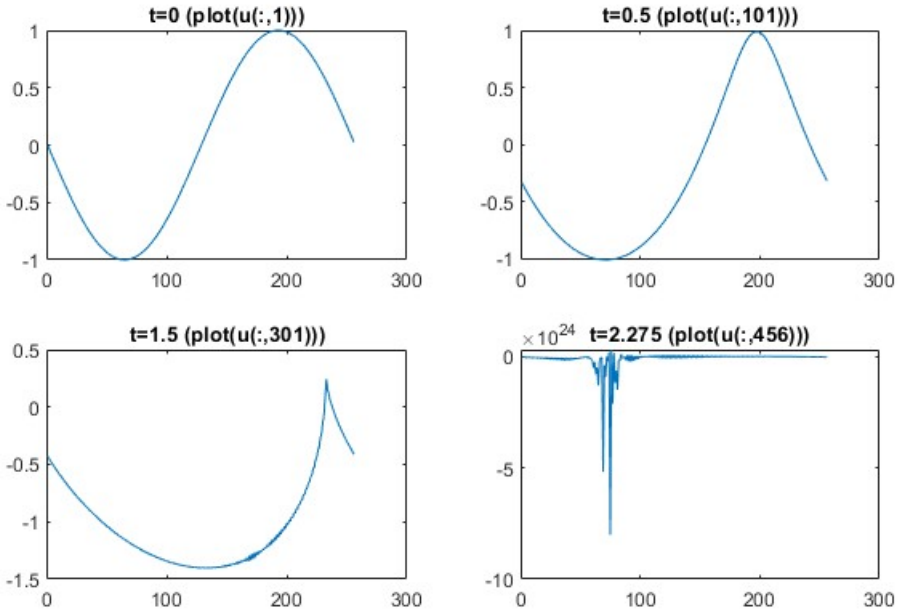

Figura 5.6: Simulación numérica de la ecuación de Córdoba usando RK4.

La diferencia con usar Euler como método de integrador temporal es que se siguen teniendo visualizaciones hasta $t = 2{,}275$. En la figura 5.6 tenemos la evolución de la simulación para diferentes tiempos. Se puede ver, que las dos simulaciones usando diferentes integradores temporales son muy similares hasta el tiempo $t = 0{,}5$. Para el instante $t = 1{,}5$ en cambio, usando Euler no se obtenía ninguna visualización porque el programa ya había dejado de funcionar correctamente pero, en este caso obtenemos lo visto en la figura 5.6. Para ver la simulación completa: https://youtu.be/_vvojPV0iv4

5.3. Dimensión 2

A partir de los programas anteriormente vistos se van a simular los escalares activos Euler 2D e IPM. Para ello, se adaptan el método de Euler y el RK4 de manera que aproximen una función con dos variables espaciales y el tiempo. Esto se consigue definiendo, al igual que en los casos anteriores, la función como una matriz pero, en vez de tener dos coordenadas $(u(t, x))$, tendrá tres $(T(x, y, t))$. Se ha visto que los escalares activos tienen una forma similar, en todos ellos aparecen la derivada en x bidimensional, la derivada en y bidimensional y el Laplaciano de una función en el espacio de Fourier. La derivada en x bidimensional se define como una matriz con los coeficientes de la derivada de Fourier multiplicada por i. El problema es que como antes en 1 dimensión, el orden para definir esta matriz en Matlab no es el natural. Si realizamos M evaluaciones se empieza en orden ascendente desde cero con los números naturales hasta $\frac{M}{2}$, a continuación va el $-\frac{M}{2}+1$ y de nuevo en orden ascendente hasta -1. Por ejemplo, para $M = 8$ se tendría

$$i \cdot \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & -3 & -2 & -1 \\ 0 & 1 & 2 & 3 & 4 & -3 & -2 & -1 \\ 0 & 1 & 2 & 3 & 4 & -3 & -2 & -1 \\ 0 & 1 & 2 & 3 & 4 & -3 & -2 & -1 \\ 0 & 1 & 2 & 3 & 4 & -3 & -2 & -1 \\ 0 & 1 & 2 & 3 & 4 & -3 & -2 & -1 \\ 0 & 1 & 2 & 3 & 4 & -3 & -2 & -1 \\ 0 & 1 & 2 & 3 & 4 & -3 & -2 & -1 \end{pmatrix}$$

La derivada en y es simplemente la traspuesta de la derivada en x. Una vez se tienen las derivadas, el Laplaciano se obtiene mediante:

$$lapla = (i \cdot derfx)^2 + (i \cdot derfy)^2$$

Siendo $derfx$ y $derfy$ la matriz de coeficientes de Fourier correspondiente en cada caso. Por ejemplo para $M = 8$ se tendría

$$\begin{pmatrix} 0 & -1 & -4 & -9 & -16 & -9 & -4 & -1 \\ -1 & -2 & -5 & -10 & -17 & -10 & -5 & -2 \\ -4 & -5 & -8 & -13 & -20 & -13 & -8 & -5 \\ -9 & -10 & -13 & -18 & -25 & -18 & -13 & -10 \\ -16 & -17 & -20 & -25 & -32 & -25 & -20 & -17 \\ -9 & -10 & -13 & -18 & -25 & -18 & -13 & -10 \\ -4 & -5 & -8 & -13 & -20 & -13 & -8 & -5 \\ -1 & -2 & -5 & -10 & -17 & -10 & -5 & -2 \end{pmatrix}$$

Antes de realizar los programa se hicieron una serie de comprobaciones para ver que estas derivadas funcionaban correctamente. Se tuvo un problema con el Laplaciano ya que el ordenador no realizaba correctamente las instrucciones introducidas, aunque se ha solucionado definiéndolo por pasos. En la figura 5.7 se puede ver alguna de las comprobaciones realizadas.

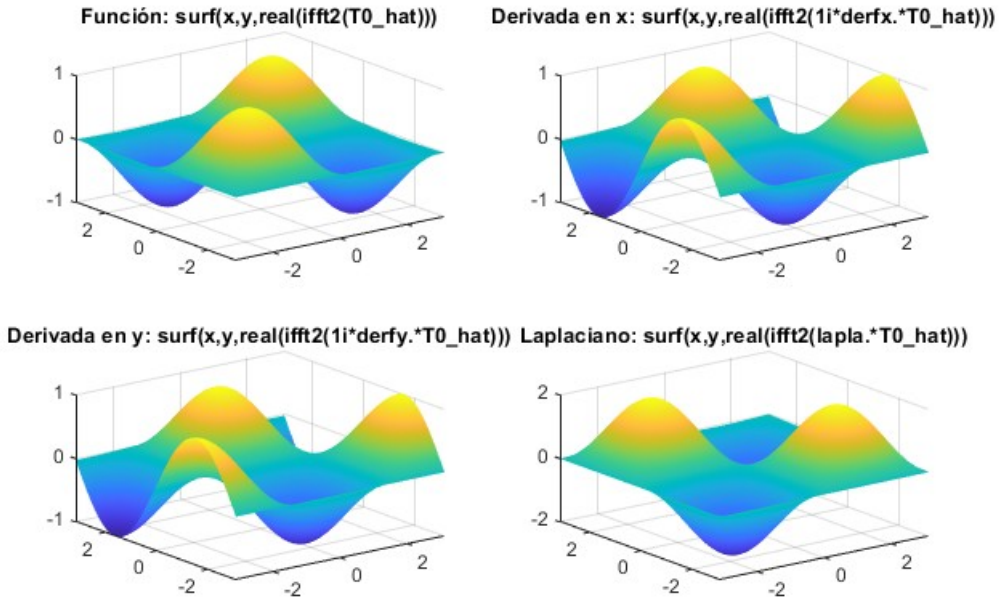

Figura 5.7: Derivadas y Laplaciano de la función $\theta(x, y) = sin(x)sin(y)$.

Otra cosa a tener en cuenta es que el Laplaciano realmente nunca se usa, en cambio se necesita la inversa de este. Esto puede suponer un problema ya que siempre va a tener un cero y por lo tanto, se tendría que dividir por cero. Para solucionarlo, una vez se han definido los multiplicadores donde se usa el inverso del Laplaciano, se sustituye el valor Inf resultante de la división por cero y en su lugar se introduce el valor 1 (En los códigos posteriores $m1(1,1) = 1$ y $m2(1,1) = 1$). Esto se puede hacer siempre y cuando la función que se tome como valor inicial tenga media cero, puesto que de este modo el 1 introducido siempre multiplicará a un cero y por tanto, no afectará a la simulación del escalar activo. Desde el punto de vista matemático, para que dividir entre cero tenga sentido tiene que ser que la división sea cero entre cero y esto sucede siempre y cuando el modo de Fourier cero del dato inicial sea cero. Esto

lo podemos ver en [8] (páginas 34-36) donde se tiene que como

$$\hat{f}(\varepsilon) = \int_{\mathbb{T}^2} e^{-2\pi i x \varepsilon} f(x) dx$$

al sustituir $\varepsilon = 0$ queda

$$\hat{f}(0) = \int_{\mathbb{T}^2} e^{-2\pi i x 0} f(x) dx = \int_{\mathbb{T}^2} f(x) dx$$

que es justamente la media de $f(x)$. Por tanto, si la media de $f(x)$ es cero, lo será también el modo de Fourier cero al igual que el modo cero del Laplaciano y así, se tendrá cero entre cero.

5.3.1. Simulación numérica de Euler 2D

En esta sección se muestra el programa para simular el escalar activo Euler 2D. Se ha tomado como valor inicial $\theta(x, y) = sin(x)^2 sin(y)^3$ con $x, y \in [-\pi, \pi]$, $M = 512$ el número de evaluaciones de Fourier, $N = 1000$ evaluaciones del integrador temporal y el tiempo entre $t = 0$ y $t = 10$. Con esto, y teniendo en cuenta la ecuación (4.9) se tiene el programa siguiente.

```
a=0;
b=10;
N=1000;
M=512;
x=(2*pi/M)*(-M/2:M/2-1);
y=x;
T0=ones(M,M);
for i=1:M
    for j=1:M
        T0(i,j)=sin(x(i)).^2*sin(y(j)).^3;
    end
end
T0_hat=fft2(T0);

%Euler:
h =(b-a)/N;

t=zeros(N+1,1);

t(1)=a;
T_hat=zeros(M,M,N+1);
T_hat(:,:,1)=T0_hat;

for i=1:N
    ti=t(i);
    Ti_hat=T_hat(:,:,i);
    t(i+1)=t(i)+h;
    T_hat(:,:,i+1)=T_hat(:,:,i)+ h*f(ti, Ti_hat);
end

T=ones(M,M,N+1);
for i=1:N+1
    T(:,:,i)=real(ifft2((T_hat(:,:,i))));
end

% Crear mallas para las coordenadas x, y
[X,Y]=meshgrid(x,y);
```

```
figure;

for i = 1:size(T, 3)
    % Graficar la superficie para el tiempo actual
    surf(X,Y,T(:,:,i));
    %contour(X,Y,T(:,:,i));
    drawnow;
end

%Escalar activo:
function dTdt = f(ti, T_hat)
    M=512;
    %Derivada en y, unidimensional
    v=[0:M/2-M/2+1:-1]; v=v';
    derfy = zeros(M);

    %Derivada en y, 2-dimensional
    for j=1:M
        derfy(:,j) = v;
    end


    %Derivada en x, 2-dimensional
    derfx = derfy';

    %Laplaciano
    s=1i*derfx;
    s=1i*derfx.*s;
    r=1i*derfy;
    r=1i*derfy.*r;
    lapla=s+r;

    m1=(-1i*derfy)./(lapla);
    m2=(1i*derfx)./(lapla);

    m1(1,1)=1;
    m2(1,1)=1;
    dTdt=-1i*derfx.*fft2(real(ifft2(m1.*T_hat)).*real(ifft2(T_hat)))+...
    -1i*derfy.*fft2(real(ifft2(m2.*T_hat)).*real(ifft2(T_hat)));
end
```

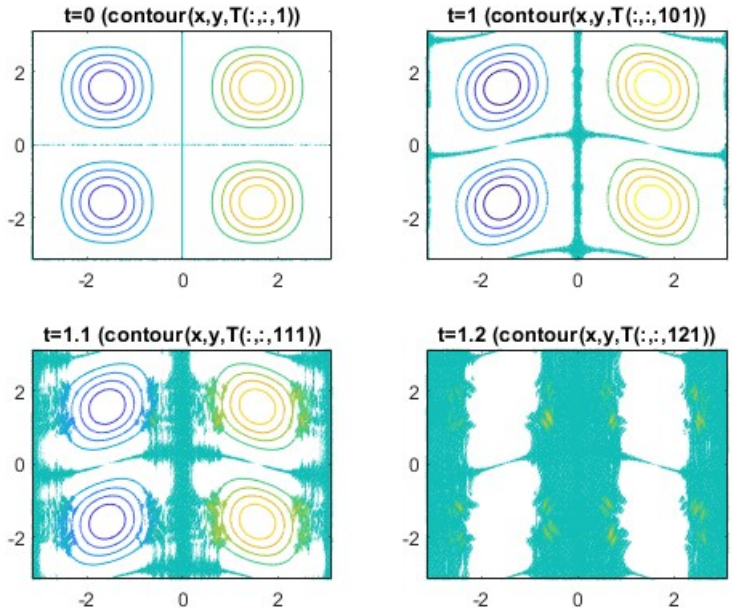

Figura 5.8: Simulación numérica de Euler 2D usando Euler como integrador temporal.

En este caso en vez de dibujar la superficie, visualizaremos el contorno. Se observa como las líneas de contorno se aproximas entre sí, comparando los tiempos $t = 0$, $t = 1$ y $t = 1{,}1$ de 5.8 lo que indica que la derivada está creciendo. Para el tiempo $t = 1{,}2$ se observa que el programa deja de dibujar correctamente el contorno de la solución y más tarde deja de visualizarse. Para ver la simulación completa: https://youtu.be/BDlzQ3RGPT8

Se muestra a continuación el programa con los mismos datos que antes pero para usando RK4.

```
a=0;
b=10;
N=1000;
M=512;
x=(2*pi/M)*(-M/2:M/2-1);
y=x;
T0=ones(M,M);
for i=1:M
    for j=1:M
        T0(i,j)=sin(x(i)).^2*sin(y(j)).^3;
    end
end
T0_hat=fft2(T0);

h=(b-a)/N;
t=zeros(N+1,1);
T_hat=zeros(M,M,N+1);
t(1)=a;
T_hat(:,:,1)=T0_hat;

fori=1:N
    ti=t(i);
    Ti_hat=T_hat(:,:,i);
```

```
    k1=f(ti,Ti_hat);
    k2=f(ti+h/2,Ti_hat+h/2*k1);
    k3=f(ti+h/2,Ti_hat+h/2*k2);
    k4=f(ti+h,Ti_hat+h*k3);

    t(i+1)=t(i)+h;
    T_hat(:,:,i+1)=T_hat(:,:,i)+h/6*(k1+2*k2+2*k3+k4);
end

T=ones(M,M,N+1);
for i=1:N+1
    T(:,:,i)=real(ifft2(T_hat(:,:,i)));
end

[X,Y]=meshgrid(x,y);
figure;
for i=1:size(T, 3)
    surf(X,Y,T(:,:,i));
    %contour(X,Y,T(:,:,i));
    drawnow;
end

function dTdt = f(ti, T_hat)
    M=512;
    v=[0:M/2 -M/2+1:-1]; v=v';
    derfy = zeros(M);

    %Derivada en y, 2-dimensional
    for j=1:M
        derfy(:,j) = v;
    end


    %Derivada en x, 2-dimensional
    derfx = derfy';

    %Laplaciano
    s=1i*derfx;
    s=1i*derfx.*s;
    r=1i*derfy;
    r=1i*derfy.*r;
    lapla=s+r;

    m1=(-1i*derfy)./(lapla);
    m2=(1i*derfx)./(lapla);

    m1(1,1)=1;
    m2(1,1)=1;
    dTdt=-1i*derfx.*fft2(real(ifft2(m1.*T_hat)).*real(ifft2(T_hat)))+...
    -1i*derfy.*fft2(real(ifft2(m2.*T_hat)).*real(ifft2(T_hat)));
end
```

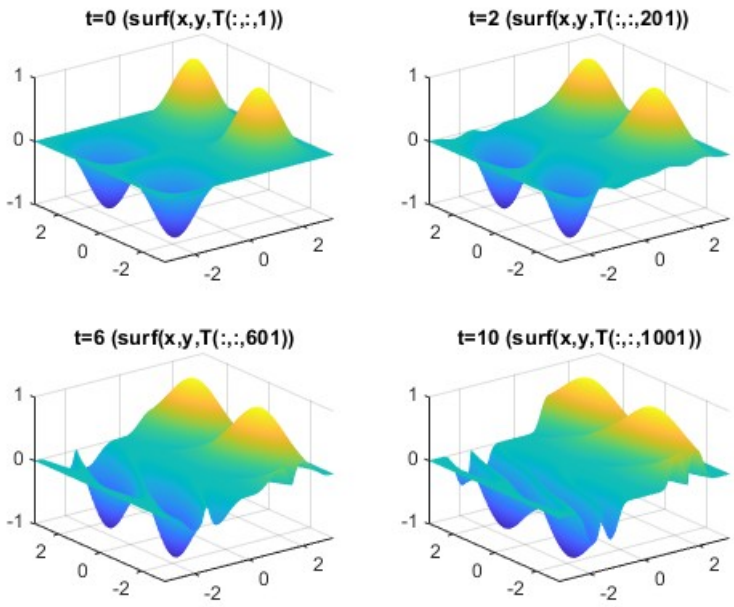

Figura 5.9: Simulación numérica de Euler 2D usando RK4, vista de la superficie.

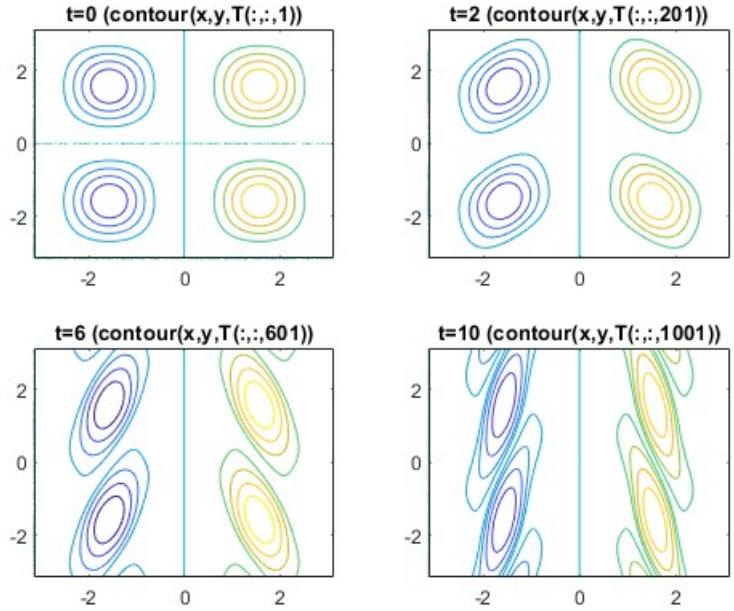

Figura 5.10: Simulación numérica de Euler 2D usando RK4, vista del contorno.

En las figura 5.9 y 5.10 se ve como en este caso conseguimos visualizar la figura para todo t. Se observa como las lineas de contorno se aproximan entre sí lo que indica que la derivada está creciendo. Cabe destacar que esta ecuación tiene existencia global. Para ver la simulación completa: https://youtu.be/b1Clz2IOC-A

Para comprobar que el código funciona correctamente, se introdujo el valor inicial $\psi(x,y) = sin(x)sin(y)$. Para este valor la solución debe permanecer estático a lo largo del tiempo ya que se trata de una solución estacionaria. Usando la proposición 2.2 de [8] (página 46):

Proposición (Construcción de soluciones estacionarias para Euler 2D)

Una función $\psi(x,y)$ definida en el dominio $\Omega \subset \mathbb{R}^2$ define una solución estacionaria para la ecuación de Euler 2D en Ω si y solo si

$$\triangle\psi = F(\psi)$$

para alguna función F.

Para $\psi(x,y) = sin(x)sin(y)$ en $\Omega = [-\pi,\pi]\times[-\pi,\pi] \subset \mathbb{R}^2$ se tiene que $\triangle\psi(x,y) = -2sin(x)sin(y) = F(sin(x)sin(y))$ para $F(r) = -2r$. $\psi(x,y) = sin(x)sin(y)$ es una solución estacionaria. Tomando el último programa pero cambiando el valor inicial por $\theta = sin(x)sin(y)$

```
T0=ones(M,M);
for i=1:M
    for j=1:M
        T0(i,j)=sin(x(i))*sin(y(j));
    end
end
T0_hat=fft2(T0);
```

se obtiene

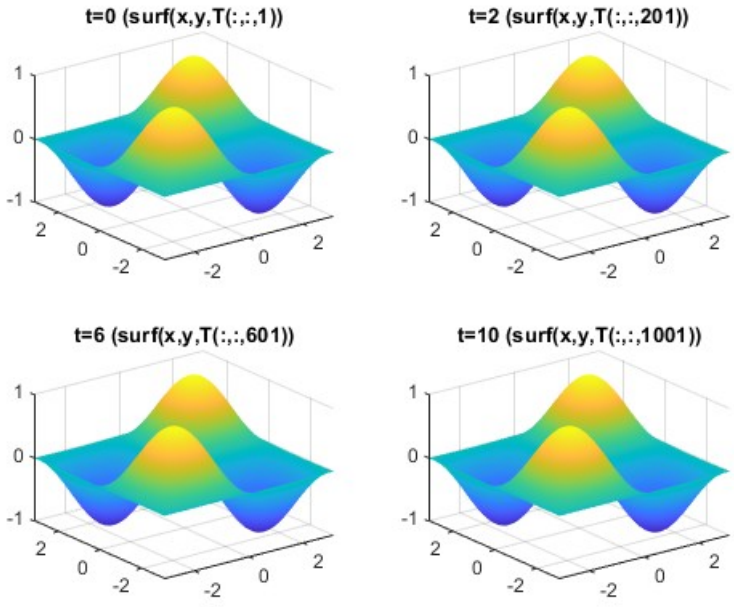

Figura 5.11: Simulación numérica de Euler 2D usando RK4 para $\theta_0 = sin(x)sin(y)$ (solución estacionaria).

En la figura 5.11 se puede ver como la simulación permanece igual a lo largo del tiempo. Se estudia también el contorno.

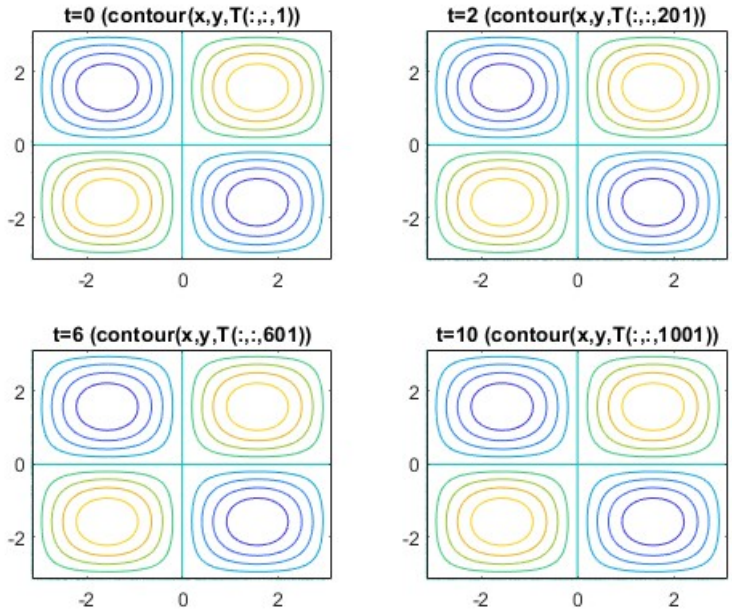

Figura 5.12: Contorno de la simulación numérica de Euler 2D usando Rk4 para T0=sin(x)sin(y) (solución estacionaria).

De nuevo, con el contorno verificamos que la solución es estacionaria. Para ver la simulación completa: https://youtu.be/uA5K12EQW8o

5.3.2. Simulación numérica de SQG

Para la SQG lo único que cambia en los programas es como se definen los multiplicadores $m1$ y $m2$. Teniendo en cuenta la ecuación (4.10) se tiene

```
lapla2=(lapla).^(1/2);
m1=(-1i*derfy)./(-lapla2);
m2=(1i*derfx)./(-lapla2);
m1(1,1)=1;
m2(1,1)=1;
```

Con esto, y con los parámetro usados en Euler 2D pero con $M = 128$ si ejecutamos el programa usando Euler como integrador temporal se podrá visualizar lo siguiente (ver figura 5.13)

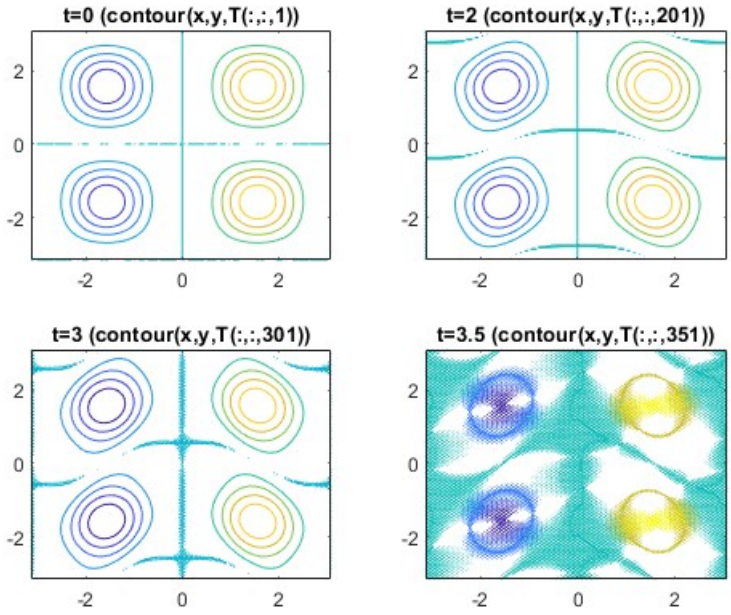

Figura 5.13: Simulación numérica de SQG usando Euler.

Se ha tomado $M = 128$ ya que si se refinaba más el programa dejaba de funcionar para un tiempo muy bajo y a penas se percibía movimiento en las visualizaciones. Esto se puede deber a la existencia de una singularidad lo que provoca que según se aumenta M, se detecte más la singularidad provocando esas inestabilidades. Para ver la simulación completa: https://youtu.be/WwnY1ElrH-Y

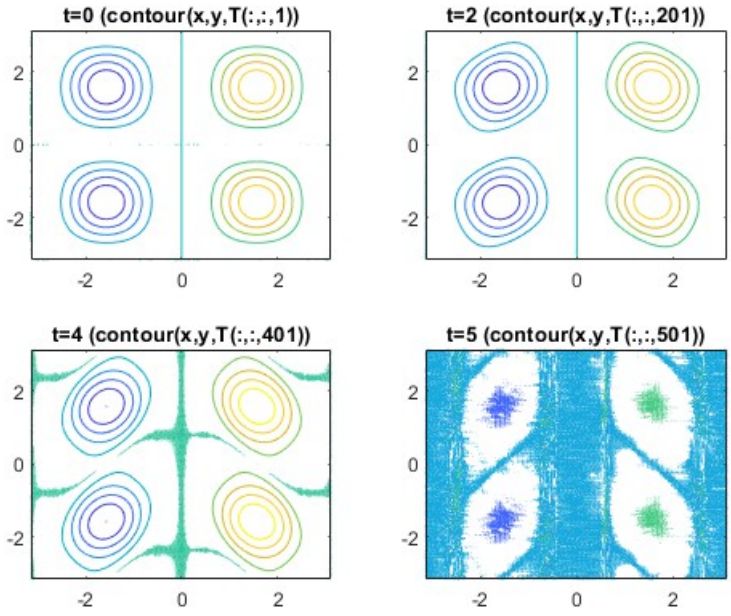

Figura 5.14: Simulación numérica de SQG usando RK4.

En la figura 5.14 se muestran los resultados con los mismos datos que el caso anterior pero $M = 256$ y usando RK4. Se aprecia como las lineas de contorno se aproximan entre si por lo que se concluye que la derivada está creciendo. De nuevo, llega un punto en que la derivada es tan grande que se producen inestabilidades hasta que se va a infinito. Para ver la simulación completa: https://youtu.be/0foFa38ubCU

En el escalar activo Euler 2D se encontró una solución estacionaria con lo que se verificó el buen funcionamiento del programa. En este caso, vamos a ver como la función θ definida como

$$\begin{cases} \theta(x,y) = 1 & si \quad x^2 + y^2 \leq 1 \\ \theta(x,y) = 0 & si \quad x^2 + y^2 > 1 \end{cases}$$

también es estacionaria. Se ejecuta el programa para ese valor inicial y con $M = 256$.

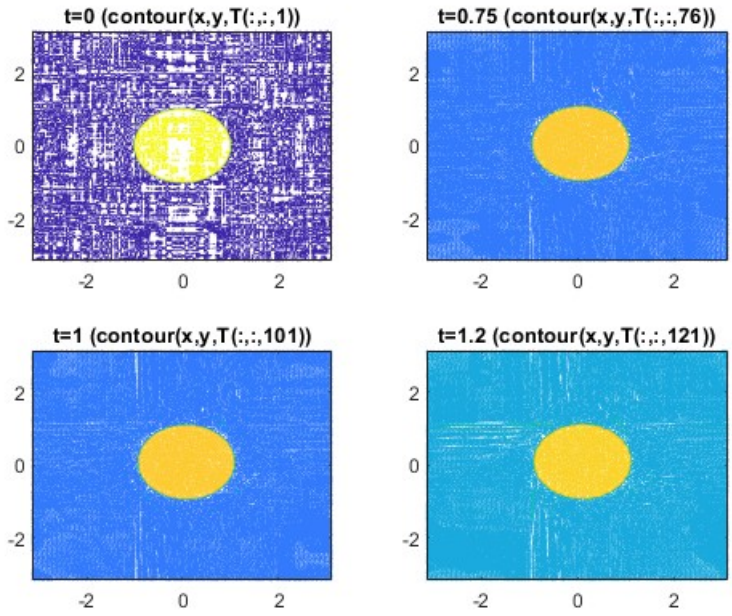

Figura 5.15: Simulación numérica de una solución estacionaria de SQG usando RK4.

Efectivamente, en la figura 5.15 se ve como la solución permanece igual para todo t. En realidad, está girando sobre sí misma lo que hace que permanezca estática. En este caso, lo que ocurre es que cuanto más se refina, la solución se va a infinito con mayor rapidez. Como el dato inicial es discontinuo, conforme aumentamos el número de evaluaciones que realiza la transformada de Fourier, más se hace notable esta singularidad lo que hace que el programa deje de proporcionar visualizaciones para un tiempo más bajo. Para ver simulación completa: https://youtu.be/ypQohMgcI7c

5.3.3. Simulación numérica de IPM

De nuevo, basta con cambiar la forma de definir los multiplicadores $m1$ y $m2$ adaptándolos para describir la ecuación (4.11). De esta forma, en Matlab se escribe:

```
m1=(1i*derfy)./(lapla);
m2=(-1i*derfx)./(lapla);
m1(1,1)=1;
m2(1,1)=1;
```

En primer lugar, con los datos utilizados en los casos anteriores pero con $M = 128$, el valor inicial $\theta(x, y) = sin(x)^2 \cdot sin(y)^3$ y usando Euler como integrador temporal se obtiene la figura 5.16.

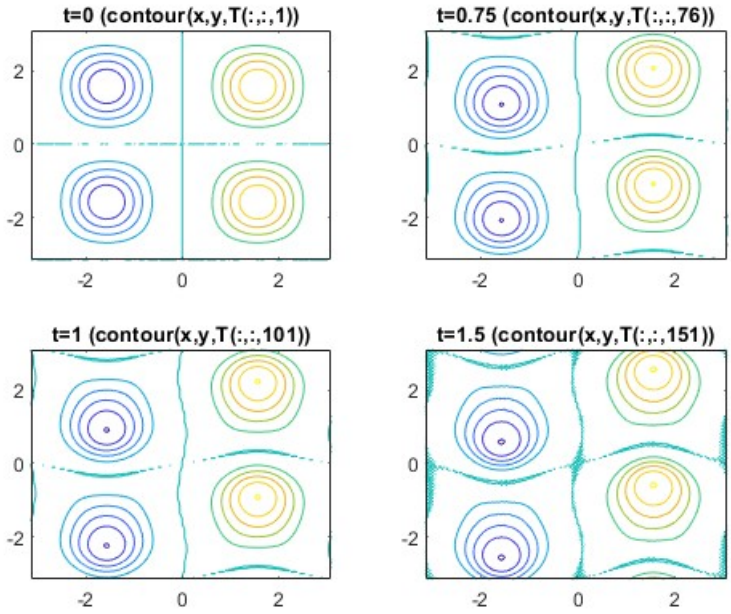

Figura 5.16: Simulación numérica de IPM usando Euler.

Para ver la simulación completa: https://youtu.be/QBgQpcHx6_I

Se ejecuta el programa para $M = 128$ y así ver de forma más clara la evolución con respecto al tiempo.

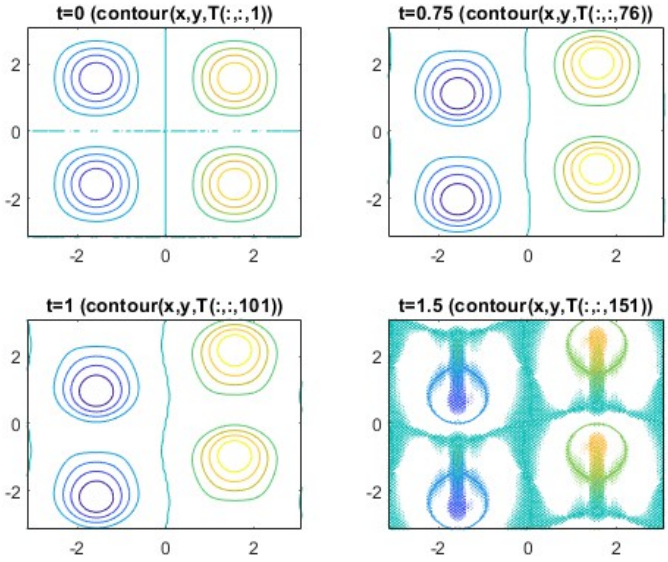

Figura 5.17: Simulación numérica de IPM usando RK4.

Para ver la simulación completa: https://youtu.be/ulUokSYsUt0

Para determinar una solución estacionaria para este escalar activo hemos visto que en [15] (lema 1.1) se ve que las soluciones estacionarias de la ecuación IPM son las de la forma $\theta = f(y)$, es decir funciones que no dependen de x, en las que la velocidad es cero ($u = 0$) y en las que la presión es nula ($p = 0$). Por ejemplo, tomando como valor inicial la función $\theta = sin(y)^3$ se obtiene la siguiente simulación (se ha vuelto a tomar $M = 512$).

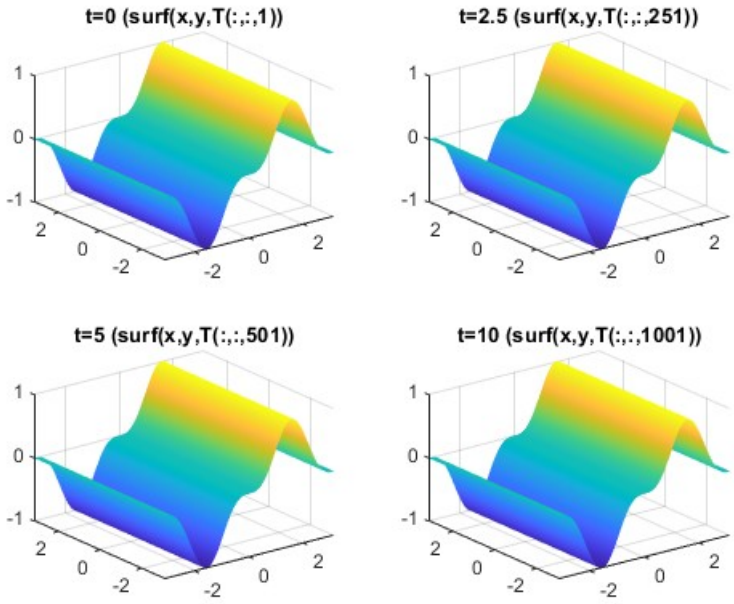

Figura 5.18: Simulación numérica de la solución estacionaria de IPM usando RK4.

En la figura 5.19 se ve como el valor inicial $\theta = sin(y)^3$ permanece estático para todos los tiempos. En este caso, se ha representado la superficie en vez de el contorno ya que al ser una función que depende solo de la variable y en el contorno solo se perciben lineas horizontales. Para ver la simulación completa: https://youtu.be/Ll-ahSBW018

5.3.4. Simulación numérica de Stokes 2D

Recuperando la ecuación (4.12) para redefinir los multiplicadores $m1$ y $m2$ se escribe

```
lapla2=(lapla).^2;
m1=(1i*derfy)./(-lapla2);
m2=(-1i*derfx)./(-lapla2);
m1(1,1)=1;
m2(1,1)=1;
```

Para el valor inicial $f(x, y) = sin(x)^2 sin(y)^3$, los mismos parámetros que en los casos anteriores ($M = 512$) y Euler como integrador temporal se tiene

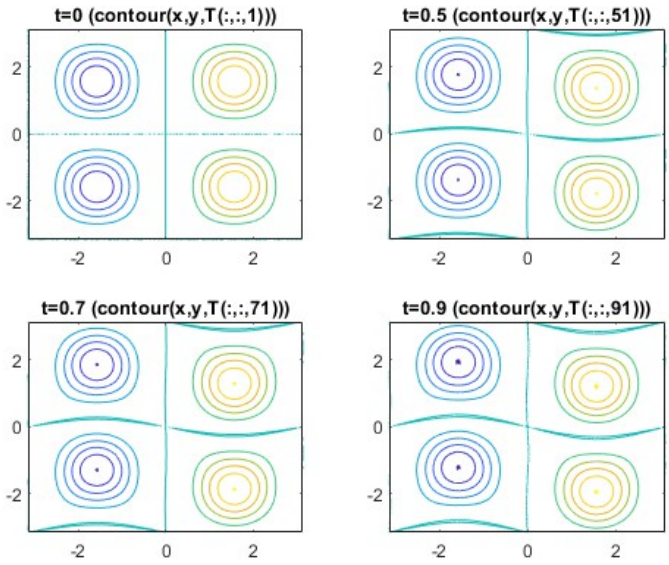

Figura 5.19: Simulación numérica de Stokes usando Euler.

Se observa que el código deja de funcionar correctamemte para un tiempo muy bajo. Para ver la simulación completa: https://youtu.be/oSYr33yuOUw

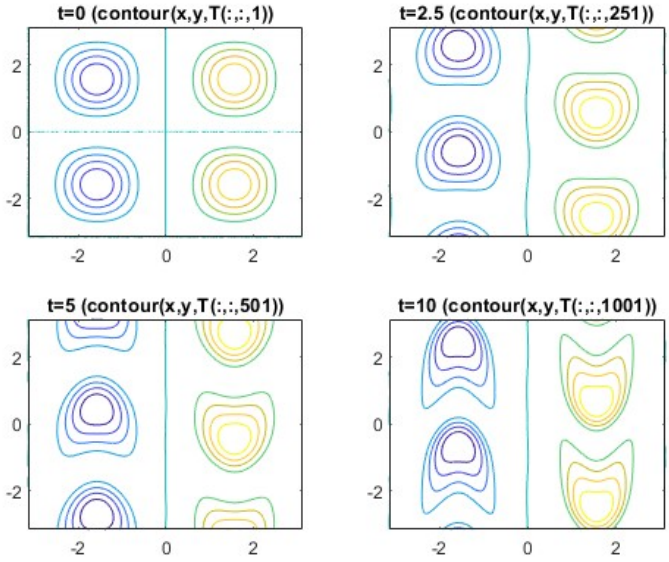

Figura 5.20: Simulación numérica de Stokes usando RK4.

Usando RK4 el código funciona para todo t. Como se puede ver este escalar activo es el más estable. Para ver la simulación completa: https://youtu.be/7DBGjUXDiow

Bibliografía

[1] Ángel Castro, Diego Córdoba, Francisco Gancedo y Rafael Orive. Incompressible flow in porous media with fractional diffusion. Nonlinearity. Vol. 22 no 8. 2009. Páginas 1791-1815.

[2] Chorin A. Estimates of intermittency, spectra, and blow-up in developed turbulance. Commun, Pure Apple. Math. Vol. 34. 1981. Páginas 853-866. https://escholarship.org/content/qt3458b4vs/qt3458b4vs.pdf [Fecha consulta: 11 Junio 2024]

[3] Chorin, A. The evolution of a turbulence vortex. Commun. Math. Phys. Vol. 83. 1982. Páginas 517-535.

[4] Fernando Vadillo. Métodos de un paso. Métodos Runge-Kutta.

[5] Hantaek Bae y Rafael Granero-Belinchón. Global existence for some transport equations with nonlocal velocity. Advances in Mathematics, vol. 269. 2015. Páginas 197-219. https://www.sciencedirect.com/science/article/pii/S0001870814003624/pdfft?md5=719e8fbe6eb8dfb89d971c58fcfdbd3d&pid=1-s2.0-S0001870814003624-main.pdf [Fecha consulta: 13 octubre 2023]

[6] J. T.Beale, T. Kato and A. Majda. Remarks on the Breakdown of Smooth Solutions for the 3-D Euler equations. Communications in Mathematical Physics, 1984. Vol. 94, no 1. 1984. Página 61-66. https://doi.org/10.1007/BF01212349 [Fecha consulta: 22 octubre 2023]

[7] **SOBRA** Luis Alberto Fernandez. Introdución a las Ecuaciones en Derivadas Parciales. Departamento de Matemáticas, Estadística y Computación. Universidad de Cantabria. Febrero, 2022. Páginas 45-50. https://personales.unican.es/lafernandez/Intro-EDP.pdf [Consulta: 17 enero 2024]

[8] Majda, Andrew J and Bertozzi, Andrea L. Vorticity and incompressible flow. Cambridge texts in applied mathematics. 2002.

[9] Mira-Cristiana Anisiu. Lotka, Volterra and their model. Didactica Matemática vol 32, no 01. 2014. https://www.researchgate.net/publication/303683438_Lotka_Volterra_and_their_model [Fecha consulta: 27 mayo 2024]

[10] Morf R., Orszag S., Frisch U. Spontaneus singularity in three-dimensonal incompressible flow. Phys, Rev, Lett. Vol. 44. 1980. Páginas 572-575. https://journals.aps.org/prl/pdf/10.1103/PhysRevLett.44.572 [Fecha consulta: 11 junio 2024]

[11] Paul Heckbert. Fourier Transforms and the Fast Fourier Transform (FFT) Algorithm. Computer Graphics, vol. 2, no 1995. 1995. Páginas 15-463. https://www.cs.cmu.edu/afs/andrew/scs/cs/15-463/2001/pub/www/notes/fourier/fourier.pdf [Fecha consulta: 25 octubre 2023]

[12] Peter Constantin, Andrew J Majda y Esteban Tabak. Formation of strong fronts in the 2-D quasigeostrophic thermal active scalar. Nonlinearity. Vol. 7, no 6. 1994. Página 1495. https://math.nyu.edu/~tabak/publications/bigQG.pdf [Fecha consulta: 13 octubre 2023]

[13] Granero Belinchón, Rafael. Ecuaciones en Derivadas Parciales y series de Fourier.

[14] Richard L. Burden y J. Douglas Faires. Análisis numérico. Thompson Learning. 2002.

ELGINDI, Tarek M. On the asymptotic stability of stationary solutions of the inviscid incompressible porous medium equation. Archive for Rational Mechanics and Analysis, 2017, vol. 225, p. 573-599.

[15] Tarek M. Elgindi. On the Asymptotic Stability of Stacionary Solutions of the Inviscid Incompressible Porous Medium Equation. Arch Rational Mechanics and Analysis. Vol. 225. 2017. Páginas 573-599. https://doi.org/10.1007/s00205-017-1090-7 [Fecha consulta: 3 de junio de 2024]

[16] Tom M. Apostol. Cálculo con funciones de una variable, con una introducción al álgebra lineal. Reverté S.A. 2011.

[17] Vu Hoang y Maria Radosz. Cusp formation for a Nonlocal Evolution Equation. Arch Rational Mechanics and Analysis. Vol 224. 2017. Página 1021-1036. https://doi.org/10.1007/s00205-017-1094-3 [Fecha consulta: 1 junio 2024]

[18] Xin Wei Yu. Hamiltonian Structure for the 2D Surface Quasi-geostrophic Equation. Proyecto final 205 CDS. 2005. Páginas 1-3.