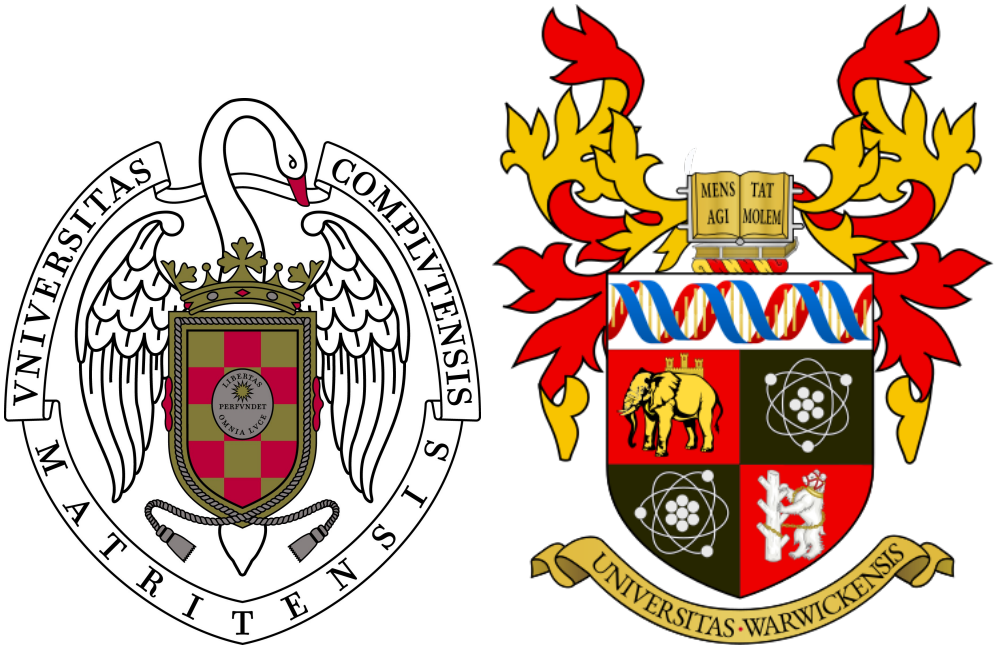
Análisis Numérico de Ecuaciones en Derivadas Parciales

Proyecto Final - Métodos de volúmenes finitos para la ecuación de advección en 1D



Carlos Eduardo Meneses Marín
Ramón Nartallo-Kaluarachchi
Grupo n.º 9

Universidad Complutense de Madrid
University of Warwick

Fecha de entrega: 31 de mayo de 2021

Índice

Resumen

Analizamos varios esquemas de volúmenes finitos para la ecuación de advección en una dimensión haciendo énfasis en la ecuación lineal. Para varios esquemas demostramos estabilidad, mientras que para el método *upwind* demostramos también convergencia a soluciones débiles-* para la ecuación lineal. Finalmente, como un extra, mostramos todos los códigos usados y se presenta una implementación de un mallado esférico refinable.

1. Introducción

Advección es el movimiento o transporte de una sustancia, como masa, o una cantidad, como calor en un líquido. En sistemas cerrados, estas cantidades se conservan. Por lo tanto, esta cantidad puede ser modelizada usando una ley de conservación general llamada una ecuación de continuidad. En este trabajo, estudiamos la ecuación de advección que es un ejemplo de una ley de conservación. En general esta dada por,

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \mathbf{f}(\mathbf{u}) = \mathbf{0} \tag{1}$$

Donde llamamos $\mathbf{u}$ el vector de *estados* y $\mathbf{f}$ el vector de *flujo*.

Para derivar un método de volúmenes finitos, el primer paso es la discretización del espacio en *volúmenes* o *células*. Si consideramos un volumen V_i, entonces la idea es calcular un valor medio de la solución, $U_i(t)$, que sea constante en V_i.

$$U_i(t) = \frac{1}{|V_i|} \int_{V_i} \mathbf{u}(\mathbf{x},t) \ d\mathbf{x} \tag{2}$$

Ahora integrando (1) con respecto a $\mathbf{x}$ sobre V_i obtenemos que,

$$\int_{V_i} \frac{\partial \mathbf{u}}{\partial t} \ d\mathbf{x} + \int_{V_i} \nabla \cdot \mathbf{f}(\mathbf{u}) \ d\mathbf{x} = \mathbf{0} \tag{3}$$

$$|V_i| \frac{dU_i}{dt}(t) + \int_{V_i} \nabla \cdot \mathbf{f}(\mathbf{u}) \ d\mathbf{x} = \mathbf{0} \tag{4}$$

Aplicando el teorema de divergencia,

$$|V_i| \frac{dU_i}{dt}(t) + \oint_{S_i} \mathbf{f}(\mathbf{u}) \cdot \mathbf{n} \ dS = \mathbf{0} \tag{5}$$

$$\frac{dU_i}{dt}(t) + \frac{1}{|V_i|} \oint_{S_i} \mathbf{f}(\mathbf{u}) \cdot \mathbf{n} \ dS = \mathbf{0} \tag{6}$$

Donde S_i es la frontera de V_i y $\mathbf{n}$ es el vector normal hacia afuera del volumen. Usando algún tipo de interpolación, se reconstruye el valor del flujo en el borde S_i. Entonces, el método final dependerá no solo de la malla, sino también de la forma de interpolación elegida.

En una dimensión el problema que nos ocupa es

$$\begin{cases} \dfrac{\partial u}{\partial t} + \dfrac{\partial f(u)}{\partial x} = 0 & x \in \mathbb{R} \quad t \geq 0 \\ u(x,0) = u_0(x) & x \in \mathbb{R} \end{cases} \tag{P}$$

donde $f \in C^1(\mathbb{R}, \mathbb{R})$ y donde el espacio al que pertenece u_0 juega un papel clave en el tipo de soluciones que se pueden encontrar como se explica en la siguiente sección.

Los volúmenes en este caso son intervalos centrados en un punto x_i. Entonces, cada volumen es de la forma $V_i = [x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}]$ con $|V_i| = h_i = x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}}$ y el borde está constituido por dos punto aislados, $S_i = \{x_{i-1/2}\} \cup \{x_{i+1/2}\}$ $\forall i \in \mathbb{Z}$. En este caso la integral del flujo queda,

$$\frac{1}{|V_i|} \oint_{S_i} \mathbf{f}(\mathbf{u}) \cdot \mathbf{n} \; dS = \frac{1}{h_i} \left[f(u(x_{i+\frac{1}{2}}, t)) - f(u(x_{i-\frac{1}{2}}, t)) \right] \tag{7}$$

$$= \frac{1}{h_i} \left[f_{i+\frac{1}{2}} - f_{i-\frac{1}{2}} \right] \tag{8}$$

donde definimos $f_{i\pm\frac{1}{2}} = f(u(x_{i\pm\frac{1}{2}}, t))$. Entonces el método en una dimensión queda,

$$\frac{du_i}{dt} + \frac{1}{h_i} \left[f_{i+\frac{1}{2}} - f_{i-\frac{1}{2}} \right] = 0 \; \forall i \in \mathbb{Z}, t > 0 \tag{9}$$

Igualmente, reconstruimos los flujos usando algún método de interpolación para no tener que evaluarlos en la solución exacta $u(x, t)$.

En la siguiente sección discutimos el tipo de soluciones que podemos esperar encontrar del problema (P) y sus características.

2. Soluciones clásicas, débiles y entrópicas débiles.

Usaremos las definiciones que ofrece el [Eym+19, p.123 - 124]:

Definición 1. *Sean $f \in C^1(\mathbb{R})$ y $u_0 \in C^1(\mathbb{R})$. Una solución clásica al problema (P) es una función $u \in C^1(\mathbb{R} \times \mathbb{R}_+, \mathbb{R})$ tal que*

$$\begin{cases} u_t(x,t) + f'\left(u(x,t)\right) u_x(x,t) = 0, & \forall x \in \mathbb{R}, \forall t \in \mathbb{R}_+ \\ u(x,0) = u_0(0), & \forall x \in \mathbb{R} \end{cases} \tag{10}$$

Durante las clases de ANEP ya trabajamos en el caso lineal ($f(u) = cu$, con $c \in \mathbb{R}$) y sus soluciones clásicas. En efecto, ya vimos que hay existencia y unicidad porque si $u_0 \in C^1(\mathbb{R})$ entonces $u(x,t) = u_0(x - ct)$ es solución clásica.

Definición 2. *Sea $f \in C^1(\mathbb{R}, \mathbb{R})$ y $u_0 \in L^\infty(\mathbb{R})$. Una solución débil al problema (P) es una función $u \in L^\infty(\mathbb{R} \times \mathbb{R}_+^*, \mathbb{R})$ tal que*

$$\int_0^\infty \int_{\mathbb{R}} u(x,t)\partial_t\phi(x,t)\, dxdt + \int_0^\infty \int_{\mathbb{R}} f\left(u(x,t)\right) \partial_x\phi(x,t)\, dxdt + \int_{\mathbb{R}} u_0(x)\phi(x,0)\, dx = 0,\ \forall \phi \in C_c^1(\mathbb{R} \times \mathbb{R}_+, \mathbb{R}) \tag{11}$$

El [Eym+19] refiere la existencia de solución débil a la ecuación de advección (P). Ahora bien, se pierde la unicidad excepto en el caso lineal ($f(u) = cu$, con $c \in \mathbb{R}$). En este caso, si $u_0 \in L^\infty(\mathbb{R})$ entonces $u(x,t) = u_0(x - ct)$ para casi todo $(x,t) \in \mathbb{R} \times \mathbb{R}_+$ es solución débil única. En la sección del análisis del método *upwind*, el caso lineal, se demostrará la convergencia débil.

Finalmente, recuperar la unicidad en el problema general (P) requiere el uso de un tipo más general de solución denominada solución entrópica débil:

Definición 3. *Sea $f \in C^1(\mathbb{R})$ y $u_0 \in L^\infty(\mathbb{R})$; una solución entrópica al problema (P) es una función $u \in L^\infty(\mathbb{R} \times \mathbb{R}_+^*, \mathbb{R})$ tal que*

$$\int_{\mathbb{R}} \int_{\mathbb{R}_+} \eta\left(u(x,t)\right) \phi_t(x,t)\, dtdx + \int_{\mathbb{R}} \int_{\mathbb{R}_+} \varphi\left(u(x,t)\right) \phi_x(x,t)\, dtdx + \int_{\mathbb{R}} \eta\left(u(x,0)\right) \phi(x,0)\, dx \geq 0,$$
$$\forall \phi \in C_c^1(\mathbb{R} \times \mathbb{R}_+, \mathbb{R}_+),\ \forall \eta \in C^1(\mathbb{R}) \text{ función convexa y } \forall \varphi \in C^1(\mathbb{R}) \text{ tal que } \varphi' = \eta' f'. \tag{12}$$

En este sentido, el [Eym+19, p. 124] refiere el siguiente teorema citando la demostración de diversas fuentes.

Teorema 4. *Sea $f \in C^1(\mathbb{R})$, $u_0 \in L^\infty(\mathbb{R})$, entonces existe una única solución entrópica al problema (P).*

3. Análisis de esquemas explícitos en 1D

En todos nuestros esquemas usaremos un mallado uniforme tanto en espacio como en tiempo. Llamaremos mallado a un conjunto $\mathcal{T}_{h,k}$ cuyos valores espaciales son la sucesión $\left(x_{i+1/2}\right)_{i\in\mathbb{Z}}$ creciente, tal que $x_{i+1/2} - x_{i-1/2} = h > 0$, $\forall i \in \mathbb{Z}$. Los valores temporales son $t_n = nk$, $n \in \mathbb{N}$. Los volúmenes son $V_i = (x_{i-1/2}, x_{i+1/2})$, $\forall i \in \mathbb{Z}$.

Consideramos el problema de aproximar $u(x,t)$ donde

$$\begin{cases} \dfrac{\partial u}{\partial t} + \dfrac{\partial f}{\partial x} = 0 & x \in \mathbb{R},\ t{\geq}0 \\ u(x,0) = u_0(x) & x \in \mathbb{R} \end{cases} \tag{13}$$

Como anteriormente, vamos a usar un esquema basado en

$$\frac{du_i}{dt} + \frac{1}{h}\left[f_{i+\frac{1}{2}} - f_{i-\frac{1}{2}}\right] = 0 \tag{14}$$

Discretizamos en tiempo usando la discretización "forward",

$$\frac{U_i^{n+1} + U_i^n}{k} + \frac{1}{h}\left[f_{i+\frac{1}{2}}^n - f_{i-\frac{1}{2}}^n\right] = 0 \tag{15}$$

La forma explícita queda,

$$\begin{cases} U_i^{n+1} = U_i^n - \frac{k}{h}\left[f_{i+\frac{1}{2}}^n - f_{i-\frac{1}{2}}^n\right] & \forall i \in \mathbb{Z},\ \forall n \in \mathbb{N}_0 \\ U_i^0 \quad = \int_{V_i} u_0(x)\,dx & \forall i \in \mathbb{Z} \end{cases} \tag{16}$$

Ahora bien, $f_{i\pm1/2}^n$ evalúa f en valores exactos de la solución $u(x,t)$ que no conocemos. Para obtener un esquema totalmente discreto, hay que aproximar el valor del flujo en los puntos $x_{i\pm\frac{1}{2}}$. Usamos F para referirnos a la aproximación numérica, es decir $F_{i\pm\frac{1}{2}}^n \approx f_{i\pm\frac{1}{2}}^n$.

Hay muchas formas para aproximar f en función de los valores U_i^n. Aquí destacaremos una en particular: asumamos que $F_{i+\frac{1}{2}}^n$ se puede aproximar por los dos puntos a cada lado U_i^n, U_{i+1}^n. Entonces definimos la *función de conservación* $\mathcal{F} : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$

$$F_{i+\frac{1}{2}}^n = \mathcal{F}(U_i^n, U_{i+1}^n) \tag{17}$$

Hay que considerar qué condiciones existen sobre $\mathcal{F}$ para que la aproximación sea "buena". Es claro que $\mathcal{F}$ esta relacionada con f porque debe ser su aproximación en el borde del volumen. Consideramos el caso donde $U_i^n = U_{i+1}^n = c \in \mathbb{R}$ coinciden. No hay ninguna razón para que una aproximación de u en el borde de estos dos volúmenes fuera distinta de c y por tanto $F_{i+\frac{1}{2}}^n = f(c)$ seria la única aproximación sensata del flujo en el borde. Por tanto imponemos la siguiente condición sobre $\mathcal{F}$,

$$\mathcal{F}(c,c) = f(c) \quad \forall c \in \mathbb{R} \tag{18}$$

Esta condición se conoce como *condición de consistencia.*

En definitiva, el esquema numérico de volúmenes finitos para la ecuación de advección (P), dado un mallado $\mathcal{T}_{h,k}$ es

$$\begin{cases} U_i^{n+1} = U_i^n - \dfrac{k}{h}\left[F_{i+\frac{1}{2}}^n - F_{i-\frac{1}{2}}^n\right] & \forall i \in \mathbb{Z},\ \forall n \in \mathbb{N}_0 \\ U_i^0 \quad = \displaystyle\int_{V_i} u_0(x)\,dx & \forall i \in \mathbb{Z} \end{cases} \tag{19}$$

donde, en varios de los esquemas que mostraremos, $F_{i+\frac{1}{2}}^n = \mathcal{F}(U_i^n, U_{i+1}^n)$.

Definición 5. *Llamamos* solución aproximada*, $u_{h,k}$ definida en $\mathbb{R} \times \mathbb{R}_+$, mediante los valores de la discretización correspondiente, U_i^n:*

$$u_{h,k}(x,t) = U_i^n \text{ para } x \in V_i,\ t \in ((n-1)k, nk],\ i \in \mathbb{Z} \text{ y } n \in \mathbb{N} \tag{20}$$

3.1. Esquema *upwind*

Para derivar el esquema *upwind*, recordemos que la solución en un instante *se mueve* con velocidad $f'(u)$ entonces la idea es que

$$F_{i\pm\frac{1}{2}}^n \approx f(u(x_{i\pm\frac{1}{2}}, nk)) \tag{21}$$

Sin embargo, surge un problema. Hay una discontinuidad en el borde del volumen. Como $u_{h,k}(x,t) = U_i(t)$ para $x \in V_i = (x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}})$, entonces si consideramos $0 < \epsilon << h$,

$$\lim_{\epsilon \to 0^-} u_{h,k}(x_{i+\frac{1}{2}} + \epsilon, t) = U_i(t) \tag{22}$$

$$\lim_{\epsilon \to 0^+} u_{h,k}(x_{i+\frac{1}{2}} + \epsilon, t) = U_{i+1}(t) \tag{23}$$

Buscamos una aproximación $U_{i+\frac{1}{2}}^n$ de $u(x_{i+\frac{1}{2}}, t)$ y es claro que hay dos opciones. Recordando otra vez que la solución en un instante se mueve con velocidad $f'(u)$, cogemos la aproximación *upwind* de la solución en cada extremo del volumen.

Por ejemplo, en cada punto $x_{i+\frac{1}{2}}$, consideramos los dos volúmenes a cada lado (donde ya tenemos una aproximación de la solución); si la velocidad en $x_{i+\frac{1}{2}}$ es positiva entonces la dirección a la derecha (hacia V_{i+1}) se llama *downwind* y la dirección a la izquierda (hacia V_i) se llama *upwind*. Si la velocidad en $x_{i+\frac{1}{2}}$ es negativa, las direcciones van al revés. Para la aproximación *upwind* siempre cogemos el valor en la dirección *upwind*. Por tanto,

$$U_{i+\frac{1}{2}}^n := \begin{cases} U_i^n & \text{si } f'(u) \geq 0 \\ U_{i+1}^n & \text{si } f'(u) < 0 \end{cases} \tag{24}$$

Entonces, por las ecuaciones (21) y (24), tenemos la función del flujo numérico,

$$F_{i+\frac{1}{2}}^n = \begin{cases} f(U_i^n) & \text{si } f'(u) \geq 0 \\ f(U_{i+1}^n) & \text{si } f'(u) < 0 \end{cases} \tag{25}$$

Por tanto, el método final queda

$$\begin{cases} U_i^{n+1} & = \begin{cases} U_i^n - \frac{k}{h}[f(U_i^n) - f(U_{i-1}^n)] & \text{si } f'(u) \geq 0 \\ U_i^n - \frac{k}{h}[f(U_{i+1}^n) - f(U_i^n)] & \text{si } f'(u) < 0 \end{cases}, & \forall n \in \mathbb{N}, \forall i \in \mathbb{Z} \\ U_i^0 & = \frac{1}{h}\int_{V_i} u_0(x), & \forall i \in \mathbb{Z} \end{cases} \tag{26}$$

A nivel práctico, necesitamos conocer $f'(u)$ en los extremos de cada volumen. No es claro en general si no se conoce la solución exacta $u(x,t)$. Para ejemplos concretos, como la ecuación de Burgers, ayudados de un principio del máximo y un dato inicial $u_0 \in L^\infty(\mathbb{R})$ y $u_0 \geq 0$, nos aseguramos que siempre estamos en el mismo caso del *upwind*.

3.2. Análisis del esquema *upwind* lineal

Planteamos el problema lineal.

$$\begin{cases} \dfrac{\partial u}{\partial t} + a\dfrac{\partial u}{\partial x} = 0 & x \in \mathbb{R},\ t \in [0,T] \\ u(x,0) = u_0(x) & x \in \mathbb{R} \end{cases} \tag{PLin}$$

con $a \in \mathbb{R}$ constante y $u_0 \in L^\infty(\mathbb{R})$. Ya notamos al introducir las soluciones débiles que $u(x,t) = u_0(x-at)$ casi todo $(x,t) \in \mathbb{R} \times \mathbb{R}_+$ es la única solución. Por tanto, el principio del máximo se cumple en el sentido de que u no supera nunca la cota esencial de u_0.

Ahora demostramos estabilidad, condición de CFL y convergencia para el esquema *upwind* aplicado al problema lineal (PLin).

3.2.1. Estabilidad L^∞ y CFL

Proposición 6. *Sea $u_0 \in L^\infty(\mathbb{R})$. Tomemos el mallado $\mathcal{T}_{h,k}$ verificando la condición $|a| \leq h/k$ (CFL). Sea $u_{h,k}$ la solución aproximada según (20) del problema lineal (PLin). Entonces, $|U_i^0| < \|u_0\|_{L^\infty}$, $\forall i \in \mathbb{Z}$ y además*

$$|u_{h,k}| \leq \|u_0\|_{L^\infty}, \ \forall x \in \mathbb{R} \ y \ \forall t \in \mathbb{R}_+^* \tag{27}$$

Demostración:

Por simplicidad se tomará $a \geq 0$, siendo el otro caso análogo.

Como $u_0 \in L^\infty(\mathbb{R})$ y V_i tiene medida finita, h, entonces $|U_i^0| = |\frac{1}{h}\int_{V_i} u_0(x)| < \infty$ y $|U_i^0| = |\frac{1}{h}\int_{V_i} u_0(x)| \leq \frac{1}{h}\int_{V_i} \|u_0\|_{L^\infty} = \|u_0\|_{L^\infty}$, $\forall i \in \mathbb{Z}$.

Para $n > 0$, veamos que U_i^{n+1} es una combinación lineal convexa de U_i^n y U_{i-1}^n. En efecto, siguiendo el esquema *upwind*,

$$U_i^{n+1} = U_i^n - \frac{k}{h}\left[aU_i^n - aU_{i-1}^n\right] = U_i^n\left(1 - a\frac{k}{h}\right) + a\frac{k}{h}U_{i-1}^n \ \forall i \in \mathbb{Z} \tag{28}$$

Usando la condición CFL, $0 \leq 1 - ak/h \leq 1$, $ak/h \leq 1$ y su suma es 1. Finalmente, una sencilla inducción basta para completar la demostración.

Obsservación: después estudiamos estabilidad para esquemas más generales, los llamados esquemas de flujo monótono.

3.2.2. Convergencia

Pasamos a estudiar la convergencia en el caso lineal (PLin). Para ello, definimos la *convergencia débil-**.

Definición 7. *Sea $(u_n)_{n\in\mathbb{N}} \subset L^\infty(\mathbb{R}\times\mathbb{R}_+^*)$. La sucesión $(u_n)_{n\in\mathbb{N}}$ converge a $u \in L^\infty(\mathbb{R}\times\mathbb{R}_+^*)$ débilmente-* (o converge en la topología débil-*), si*

$$\int_{\mathbb{R}_+}\int_{\mathbb{R}} (u_n(x,t) - u(x,t))\phi(x,t)\,dx\,dt \to 0 \tag{29}$$

cuando $n \to \infty$, $\forall \phi \in L^1(\mathbb{R}\times\mathbb{R}_+^*)$

Usando esta noción de convergencia, veamos que nuestra solución numérica converge a una función y entonces veamos que esta función es solución débil del problema.

Teorema 8. *Sea $u_0 \in L^\infty(\mathbb{R})$. Tomemos el mallado $\mathcal{T}_{h,k}$ verificando la condición CFL $|a| \leq (1-\xi)h/k$ con $\xi \in (0,1)$. Sea $u_{h,k}$ la solución aproximada según (20) del problema lineal (PLin) y sea u su solución débil (única). Entonces, cuando $(h,k) \to 0$, $u_{h,k}$ converge débilmente-* a u.*

Para demostrar el teorema, necesitamos otros resultados.

Según la *Proposición 6*, la solución aproximada esta acotada en L^∞. Usamos entonces el teorema de Banach-Alaoglu-Bourbaki (ver [Bre10]) para asegurar la existencia de una subsucesión convergente debilmente-*. Hemos buscado sin éxito una demostración para el caso particular que nos ocupa. Solo hemos encontrado la siguiente proposición de [Kar06, p. 2] sin demostración.

Proposición 9. *Sea $(u_n)_{n\in\mathbb{N}} \subset L^\infty(\Omega)$ acotada. Entonces, existe una subsucesión de $(u_n)_{n\in\mathbb{N}}$ y una función $u \in L^\infty(\Omega)$ tal que la subsucesión converge a u débilmente-*.*

También necesitamos la siguiente desigualdad.

Proposición 10. (Estimador de variación acotada). *Tomemos el mallado $\mathcal{T}_{h,k}$ verificando la condición CFL $|a| \leq (1-\xi)h/k$ con $\xi \in (0,1)$. Sea $\{U_i^n : i \in \mathbb{Z}; n \in \mathbb{N}_0\}$ solución numérica según (26) del problema lineal (PLin). Sean R, $T \in \mathbb{R}_+^*$ tales que $h < R$, $k < T$. Sea i_0, $i_1 \in \mathbb{Z}$, $N \in \mathbb{N}$ tales que*

$$-R \in [x_{i_0-\frac{1}{2}}, x_{i_0+\frac{1}{2}}] \tag{30}$$

$$R \in [x_{i_1-\frac{1}{2}}, x_{i_1+\frac{1}{2}}] \tag{31}$$

$$T \in (Nk, (N+1)k] \tag{32}$$

Entonces existe $C \in \mathbb{R}_+^*$ *que sólo depende de* R, T, u_0, a *y* ξ, *tal que*

$$\sum_{i=i_0}^{i_1}\sum_{n=0}^{N} k|U_i^n - U_{i-1}^n| \leq Ch^{-\frac{1}{2}} \tag{33}$$

Demostración: supongamos que $a > 0$ (el otro caso es análogo) y partimos del esquema,

$$h\frac{U_i^{n+1} - U_i^n}{ka} + U_i^n - U_{i-1}^n = 0 \tag{34}$$

Entonces, multiplicamos ambos sumandos por kU_i^n y sumamos sobre $i = i_0, \ldots, i_1$ y $n = 0, \ldots, N$. Esto nos da $A + B = 0$, donde

$$A = \sum_{i=i_0}^{i_1}\sum_{n=0}^{N} h(U_i^{n+1} - U_i^n)U_i^n \tag{35}$$

$$B = \sum_{i=i_0}^{i_1}\sum_{n=0}^{N} ka(U_i^n - U_{i-1}^n)U_i^n \tag{36}$$

Ahora reescribimos A.

$$A = -\frac{1}{2}\sum_{i=i_0}^{i_1}\sum_{n=0}^{N} h(U_i^{n+1} - U_i^n)^2 + \frac{1}{2}\sum_{i=i_0}^{i_1} h[(U_i^{N+1})^2 - (U_i^0)^2] \tag{37}$$

Aplicamos la igualdad del esquema,

$$A = -\frac{1}{2}\sum_{i=i_0}^{i_1}\sum_{n=0}^{N} \frac{k^2a^2}{h}(U_i^n - U_{i-1}^n)^2 + \frac{1}{2}\sum_{i=i_0}^{i_1} h[(U_i^{N+1})^2 - (U_i^0)^2] \tag{38}$$

Aplicamos la condición CFL,

$$A \geq -(1-\xi)\frac{1}{2}\sum_{i=i_0}^{i_1}\sum_{n=0}^{N} ka(U_i^n - U_{i-1}^n)^2 - \frac{1}{2}\sum_{i=i_0}^{i_1} h(U_i^0)^2 \tag{39}$$

Ahora, reescribimos B

$$B = \frac{1}{2}\sum_{i=i_0}^{i_1}\sum_{n=0}^{N} ka(U_i^n - U_{i-1}^n)^2 + \frac{1}{2}\sum_{n=0}^{N} ka[(U_{i_1}^n)^2 - (U_{i_0-1}^N)^2] \tag{40}$$

Ahora aplicamos que $\sum_{n=0}^{N} k \leq 2T$ y $|U_i^n| < \|u_0\|_{L^\infty}$, $\forall i \in \mathbb{Z}$, $\forall n \in \mathbb{N}$ (ver *Proposición 6*),

$$B \geq \frac{1}{2}\sum_{i=i_0}^{i_1}\sum_{n=0}^{N} k(U_i^n - U_{i-1}^n)^2 - T\|u_0\|_{L^\infty}^2 \tag{41}$$

Entonces recordando que $A + B = 0$, obtenemos que,

$$0 \geq (1 - 1 + \xi)\frac{1}{2}\sum_{i=i_0}^{i_1}\sum_{n=0}^{N} ka(U_i^n - U_{i-1}^n)^2 - \frac{1}{2}\sum_{i=i_0}^{i_1} h(U_i^0)^2 - T\|u_0\|_{L^\infty}^2 \tag{42}$$

Ahora recordamos que $R \in [x_{i_1-\frac{1}{2}}, x_{i_1+\frac{1}{2}}]$ y $-R \in [x_{i_0-\frac{1}{2}}, x_{i_0+\frac{1}{2}}]$. Por lo tanto, tenemos que

$$R \geq x_{i_1-\frac{1}{2}} > 0 \tag{43}$$
$$-R \leq x_{i_0+\frac{1}{2}} < 0 \tag{44}$$

Entonces,

$$2R \geq x_{i_1-\frac{1}{2}} - x_{i_0+\frac{1}{2}} \tag{45}$$
$$2h + 2R \geq x_{i_1+\frac{1}{2}} - x_{i_0-\frac{1}{2}} \tag{46}$$
$$4R \geq x_{i_1+\frac{1}{2}} - x_{i_0-\frac{1}{2}} \tag{47}$$

Y además $\sum_{i=i_0}^{i_1} h \leq 4R$. Usando la cota $|U_i^0| < \|u_0\|_{L^\infty}$, $\forall i \in \mathbb{Z}$ (ver *Proposición 6*),

$$0 \geq \frac{\xi}{2} \sum_{i=i_0}^{i_1} \sum_{n=0}^{N} ka(U_i^n - U_{i-1}^n)^2 - (2R+T) \|u_0\|_{L^\infty}^2 \tag{48}$$

Reorganizando los miembros tenemos que,

$$\sum_{i=i_0}^{i_1} \sum_{n=0}^{N} ka(U_i^n - U_{i-1}^n)^2 \leq C_1 = \frac{(4R+2T) \|u_0\|_{L^\infty}^2}{\xi} \tag{49}$$

Nótese que C_1 sólo depende de R, T, u_0 y ξ.

Por otro lado, la desigualdad de Jensen para sumas finitas establece que dadas una función convexa $\varphi(x)$ real y convexa, $x_1, \ldots, x_n$ valores de su dominio y $a_1, \ldots, a_n$ pesos positivos,[1]

$$\varphi\left(\frac{\sum_i a_i x_i}{\sum_i a_i}\right) = \frac{\sum_i a_i \varphi(x_i)}{\sum_i a_i} \tag{50}$$

Tomando $\varphi(x) = x^2$ y aplicando la desigualdad de Jensen en la ecuación (49),

$$\left[\sum_{i=i_0}^{i_1} \sum_{n=0}^{N} ka(U_i^n - U_{i-1}^n)\right]^2 \leq \left(\sum_{i=i_0}^{i_1} \sum_{n=0}^{N} ka\right) C_1 \tag{51}$$

Finalmente, teniendo en cuenta que $\sum_{n=0}^{N} k \leq 2T$ y $\sum_{i=i_0}^{i_1} h \leq 4R$ obtenemos

$$\left[\sum_{i=i_0}^{i_1} \sum_{n=0}^{N} ka(U_i^n - U_{i-1}^n)\right]^2 \leq \frac{8aRTC_1}{h} = \frac{C}{h} \tag{52}$$

Siendo C una constante positiva que solo depende de R, T, u_0, a y ξ. □

[1] `https://en.wikipedia.org/wiki/Jensen%27s_inequality#Finite_form`

Demostración del Teorema 8:

Consideramos una sucesión (h_j, k_j) que defina una sucesión de mallas, $\mathcal{T}_j$, y de soluciones aproximadas, u_{h_j,k_j}. Aplicamos el teorema de compacidad de Banach-Alaoglu-Bourbaki (ver [Bre10]) o la *Proposición 9*. Así aseguramos la existencia de una subsucesión (también denotada por u_{h_j,k_j}) que converge débilmente-* a una función u.

La idea es comprobar que este limite u es solución débil del problema. Por la unicidad de la solución débil, toda subsucesión convergente débilmente-* converge a la misma u. Ahora bien, la sucesión al completo debe converger a u, de lo contrario, usando la definición de convergencia y el teorema de compacidad encontraríamos una subsucesión convergente que no tiende a u.

Consideramos el caso $a > 0$, siendo el otro análogo. Partiendo desde la definición de solución débil (ver *Definición 2*), sea $\phi \in C_c^1(\mathbb{R} \times \mathbb{R}^+, \mathbb{R})$. Consideramos el esquema otra vez y para cada malla $\mathcal{T}_j$,

$$h\frac{U_i^{n+1} - U_i^n}{ka} + U_i^n - U_{i-1}^n = 0 \tag{53}$$

Multiplicamos por $\frac{ka}{h}\phi(x, nk)$,

$$(U_i^{n+1} - U_i^n)\phi(x, nk) + \frac{h}{ka}(U_i^n - U_{i-1}^n)\phi(x, nk) = 0 \tag{54}$$

Integramos sobre $x \in V_i$ y sumamos en $i \in \mathbb{Z}$ y $n \in \mathbb{N}_0$. Esto nos da

$$A_j + B_j = 0 \tag{55}$$

donde

$$A_j = \sum_{i\in\mathbb{Z}}\sum_{n\in\mathbb{N}}(U_i^{n+1} - U_i^n)\int_{V_i}\phi(x, nk)\ \ dx, \tag{56}$$

$$B_j = \sum_{i\in\mathbb{Z}}\sum_{n\in\mathbb{N}} ka(U_i^n - U_{i-1}^n)\frac{1}{h}\int_{V_i}\phi(x, nk)\ \ dx, \tag{57}$$

el índice j se refiere a la malla $\mathcal{T}_j$ y $A_j < \infty$, $B_j < \infty$ porque ϕ tiene soporte compacto.

Rescribimos A_m,

$$A_j = \sum_{i\in\mathbb{Z}}\left[\sum_{n\geq 1} U_i^n \int_{V_i}\phi(x, (n-1)k) - \sum_{n\geq 0} U_i^n \int_{V_i}\phi(x, nk)\right] \tag{58}$$

$$= \sum_{i\in\mathbb{Z}}\sum_{n\geq 1} U_i^n \int_{V_i}[\phi(x, (n-1)k) - \phi(x, nk)] - \sum_{i\in\mathbb{Z}} U_i^0 \int_{V_i}\phi(x,0) \tag{59}$$

$$= -\sum_{i\in\mathbb{Z}}\sum_{n\geq 1} U_i^n \int_{V_i}\int_{(n-1)k}^{nk}\phi_t(x,t)\ dt\ dx - \sum_{i\in\mathbb{Z}} U_i^0 \int_{V_i}\phi(x,0) \tag{60}$$

$$= -\int_0^\infty\int_{\mathbb{R}} u_{h_j,k_j}(x,t)\phi_t(x,t)\ \ dx\ dt - \int_{\mathbb{R}} u_0(x)\phi(x,0) \tag{61}$$

Entonces, usando la convergencia débil-*, cuando $j \to \infty$ tenemos que

$$A_j \to -\int_0^\infty \int_{\mathbb{R}} u(x,t)\phi_t(x,t)\ \ dx\ dt - \int_{\mathbb{R}} u_0(x)\phi(x,0)\ dx \tag{62}$$

Ahora estudiamos B_j. Consideramos la expresión B_j^*,

$$B_j^* = -\sum_{n\in\mathbb{N}} \int_{nk}^{(n+1)k} \int_{\mathbb{R}} a u_{h_j,k_j}(x,t)\phi_x(x,nk)\ dx\ dt \tag{63}$$

Cuando $j \to \infty$, sabemos que

$$B_j^* \to -\int_{\mathbb{R}_+} \int_{\mathbb{R}} a u(x,t)\phi_x(x,t)\ \ dx\ dt \tag{64}$$

Siguiendo pasos similares a los anteriores, podemos reescribir B_j^* como

$$B_j^* = \sum_{i\in\mathbb{Z}} \sum_{n\in\mathbb{N}} ka(U_i^n - U_{i-1}^n)\phi(x_{i-\frac{1}{2}},nk) \tag{65}$$

Por otro lado, por ser ϕ de soporte compacto, existen constantes positivas R y T mayores que h y k respectivamente tales que $\phi(x,t) = 0$ si $|x| \geq R$ o $t \geq T$. También tales que existan i_0, $i_1 \in \mathbb{Z}$ y $N \in \mathbb{N}$ y

$$|B_j - B_j^*| = \left| \sum_{i=i_0}^{i_1} \sum_{n=0}^{N} ka(U_i^n - U_{i-1}^n) \left(\frac{1}{h} \int_{V_i} \phi(x,nk)dx - \phi(x_{i-1/2},nk) \right) \right| \tag{66}$$

Desarrollando $\phi(x,kn)$ en serie de Taylor y usando la regularidad de ϕ es fácil ver que existe una constante $C_1 > 0$ dependiente solo de ϕ tal que

$$\left| \frac{1}{h} \int_{V_i} \phi(x,nk)dx - \phi(x_{i-1/2},nk) \right| \leq C_1 h$$

Ahora, aplicando el estimador de variación acotada (ver *Proposición 10)* existe $C > 0$ tal que,

$$|B_j - B_j^*| \leq C_1 h \sum_{i=i_0}^{i_1} \sum_{n=0}^{N} ka|U_i^n - U_{i-1}^n| \leq Ch^{1/2} \tag{67}$$

Pasando al límite, cuando $j \to \infty$ obtenemos que

$$|B_j - B_j^*| \to 0 \tag{68}$$

y

$$B_j \to \int_{\mathbb{R}_+} \int_{\mathbb{R}} u(x,t)\phi_x(x,t) \ dx \ dt \tag{69}$$

En conclusión tenemos que,

$$\int_{\mathbb{R}_+} \int_{\mathbb{R}} u(x,t)\phi_t(x,t) \ dx \ dt + \int_{\mathbb{R}} u_0(x)\phi(x,0) \ dx + \int_{\mathbb{R}_+} \int_{\mathbb{R}} au(x,t)\phi_x(x,t) \ dx \ dt = 0 \tag{70}$$

Por lo tanto, tenemos que el limite débil-*, u, es solución débil del problema. □

3.3. Esquema Lax-Friedrichs

3.3.1. Derivación

Consideramos otra vez la ecuación,

$$\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} = 0 \tag{71}$$

Para derivar el esquema Lax-Friedrichs (LF), usamos las siguientes aproximaciones de las derivadas,

$$\frac{\partial u}{\partial t} \approx \frac{u(x,t+k) - \frac{1}{2}(u(x+h,t) + u(x-h,t))}{k} \tag{72}$$

$$\frac{\partial f(u)}{\partial x} \approx \frac{f(u(x+h,t)) - f(u(x-h,t))}{2h} \tag{73}$$

Aplicando la discretización en espacio y en tiempo, obtenemos la forma,

$$\frac{U_i^{n+1} - \frac{1}{2}(U_{i+1}^n + U_{i-1}^n)}{k} + \frac{f(U_{i+1}^n) - f(U_{i-1}^n)}{2h} = 0 \tag{74}$$

Escrito explícitamente, el esquema queda,

$$U_i^{n+1} = \frac{1}{2}(U_{i+1}^n + U_{i-1}^n) - \frac{k}{2h}(f(U_{i+1}^n) - f(U_{i-1}^n)) \tag{75}$$

Escrito en la forma general derivada anteriormente,

$$U_i^{n+1} = U_i^n - \frac{k}{2h}(F_{i+\frac{1}{2}}^n - F_{i-\frac{1}{2}}^n) \tag{76}$$

Donde el flujo numérico esta dado por,

$$F_{i-\frac{1}{2}}^n = \frac{1}{2}(f(U_{i-1}^n) + f(U_i^n)) - \frac{h}{2k}(U_i^n - U_{i-1}^n) \tag{77}$$

Este esquema se llama *Lax-Friedrichs*. Trivialmente, comprobamos que verifica la condición de consistencia,

$$\mathcal{F}(u,u) = \frac{1}{2}(f(u) + f(u)) - \frac{h}{2k}(u - u) \tag{78}$$
$$= f(u) \tag{79}$$

Observación: Nótese que en la aproximación de la derivada en tiempo aparece el paso en espacio. Esto impone una *condición CFL inversa*. Consideramos las expansiones de Taylor,

$$u(x + h, t) = u(x, t) + h\frac{\partial u}{\partial x}(x, t) + \frac{h^2}{2}\frac{\partial^2 u}{\partial x^2}(\epsilon, t) \tag{80}$$
$$u(x - h, t) = u(x, t) - h\frac{\partial u}{\partial x}(x, t) + \frac{h^2}{2}\frac{\partial^2 u}{\partial x^2}(\nu, t) \tag{81}$$

Entonces sumando y dividiendo por 2 obtenemos que

$$\frac{1}{2}(u(x + h, t) + u(x - h, t)) = u(x, t) + \frac{h^2}{2}\left(\frac{\partial^2 u}{\partial^2 x}(\epsilon, t) + \frac{\partial^2 u}{\partial x^2}(\nu, t)\right) \tag{82}$$

Donde $\epsilon \in [x,x + h]$ y $\nu \in [x - h,x]$. Nuestra aproximación queda,

$$\frac{\partial u}{\partial t} \approx \frac{u(x, t + k) - u(x, t) - \frac{h^2}{2}(\frac{\partial^2 u}{\partial x^2}(\epsilon, t) + \frac{\partial^2 u}{\partial x^2}(\nu, t))}{k} \tag{83}$$

Aplicamos el teorema del valor medio,

$$\frac{\partial u}{\partial t} \approx \frac{u(x, t + k) - u(x, t) - h^2(\frac{\partial^2 u}{\partial x^2}(\mu, t))}{k} \tag{84}$$

Donde $\mu \in [\nu,\epsilon]$. Por lo tanto, esta expresión solo converge a la derivada en tiempo si,

$$\frac{h^2}{k}\left(\frac{\partial^2 u}{\partial x^2}(\mu,t)\right) \underset{h,k\to 0}{\longrightarrow} 0 \tag{85}$$

Entonces, para que este término converja a 0, queremos un paso de tiempo k tal que $k \geq h^2$ cuando $k,h \to 0$ por lo tanto, cogemos un valor de k del mismo orden de h. Esto se introduce como la *condición CFL inversa* en [Eym+19, p. 190]. Ahora bien, a nivel práctico, es preferible usar condiciones del tipo $k \approx h$ así que no surge un gran problema. Sin embargo, es importante observar que si usamos $k \approx h^2$ (del mismo orden), se ha constatado en diversos experimentos numéricos que el esquema converge pero como sufriendo difusión. En este sentido, concluimos que la solución numérica no esta convergiendo a la solución del problema de transporte en un mallado cumpliendo tal condición.

3.3.2. Estabilidad L^∞ y CFL

Estudiamos estabilidad del método con la variante lineal de la ecuación,

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = 0 \tag{86}$$

Si a es constante, entonces la ecuación verifica el principio del máximo.

Proposición 11. *(Estabilidad del esquema LF y CFL para la ecuación lineal) Si* $\frac{k}{h} \leq \frac{1}{|a|}$, *entonces sea* (U_i^n) *una solución numérica obtenida por el método Lax-Friedrichs, tenemos que*

$$\forall n \geq 0, \quad \forall i \geq 0, \quad |U_i^{n+1}| \leq \sup_{j \in \mathbb{Z}^+} U_j^n \tag{87}$$

Más aún, dada la condición inicial $u_0(x) = u(x,0)$, *tenemos que*

$$\sup_{\forall n \geq 0, \forall i \geq 0} |u_i^n| \leq ||u_0||_{L^\infty(\mathbb{R})} \tag{88}$$

Demostración: supongamos que k y h verifican la condición CFL. Entonces, consideramos el esquema,

$$U_i^{n+1} = \frac{1}{2}(U_{i+1}^n + U_{i-1}^n) - \frac{k}{2h}(aU_{i+1}^n - aU_{i-1}^n) \tag{89}$$

$$= U_{i+1}^n \left[\frac{1}{2} - \frac{ak}{2h}\right] + U_{i-1}^n \left[\frac{1}{2} + \frac{ak}{2h}\right] \tag{90}$$

La condición CFL implica que,

$$0 \leq \frac{|a|k}{2h} \leq \frac{1}{2} \tag{91}$$

Entonces tenemos que,

$$0 \leq \frac{1}{2} - \frac{ak}{2h} \tag{92}$$

$$0 \leq \frac{1}{2} + \frac{ak}{2h} \tag{93}$$

$$1 = \frac{1}{2} - \frac{ak}{2h} + \frac{1}{2} + \frac{ak}{2h} \tag{94}$$

Entonces, (90) es una combinación convexa y por tanto tenemos que,

$$\text{mín}\{U_{i-1}^n, U_{i+1}^n\} \leq U_i^{n+1} \leq \text{máx}\{U_{i-1}^n, U_{i+1}^n\} \tag{95}$$

$$\inf_{j \in \mathbb{Z}} U_j^n \leq U_i^{n+1} \leq \sup_{j \in \mathbb{Z}} U_j^n \tag{96}$$

Aplicando esto hasta los bordes, obtenemos que,

$$\sup_{\forall n \geq 0, \forall i \geq 0} |U_i^n| \leq \|u_0\|_{L^\infty(\mathbb{R})} \tag{97}$$

□

3.4. Esquemas de flujo monótono

Los esquemas de flujo monótono forman una subclase importante de métodos para flujos generales pero cumpliendo cierta condición de monotonía aquí desarrollada.

Definición 12. *Un esquema (19) con función de conservación* $\mathcal{F} : \mathbb{R}^2 \to \mathbb{R}$ *se llama* esquema de flujo monótono *para el problema de transporte (P) si dado* $f \in C^1(\mathbb{R})$ *y* $u_0 \in L^\infty(\mathbb{R} \times \mathbb{R})$ *con acotas* $U_m \leq u_0 \leq U_M$ *casi todo punto, se cumple*

1. $\mathcal{F}$ *es localmente Lipschitz de* $\mathbb{R}^2 \to \mathbb{R}$.
2. $\mathcal{F}(s, s) = f(s) \quad \forall s \in [U_m, U_M]$.
3. $(a, b) \mapsto \mathcal{F}(a, b)$ *desde* $[U_m, U_M]^2$ *hasta* $\mathbb{R}$ *es no-decreciente con respeto a la primera variable y no-creciente con respeto a la segunda variable.*

Claramente, por definición, estos métodos son consistentes (condición 2). También veremos su ventaja de ser estables L^∞ bajo cierta condición de CFL del tipo $k/h \leq C$ donde C sólo depende de $\mathcal{F}$ y u_0.

Observación: bajo cierta condición CFL, todo esquema de flujo monótona se puede escribir en la forma,

$$U_i^{n+1} = H(U_{i-1}^n, U_i^n, U_{i+1}^n) \tag{98}$$

donde H es no-decreciente con respeto a los tres argumentos. Esta propiedad es la definición de *esquema monótono*.

3.4.1. Esquemas de flujo separado

Si escribimos f en la forma

$$f(s) = f_1(s) + f_2(s) \tag{99}$$

donde f_1, $f_2 \in C^1(\mathbb{R})$ y $f_1' \geq 0$, $f_2' \leq 0$, entonces la función,

$$\mathcal{F}(a, b) = f_1(a) + f_2(b) \tag{100}$$

nos da un esquema de flujo monótono, porque

1. $\mathcal{F}$ es localmente Lipschitz porque f_1, $f_2 \in C^1(\mathbb{R})$ y por tanto ambos son localmente Lipschitz.
2. $\mathcal{F}(a,a) = f_1(a) + f_2(a) = f(a)$
3. $\dfrac{\partial \mathcal{F}}{\partial a} = f_1'(a) \geq 0 \Rightarrow \mathcal{F}$ es no decreciente en la primera variable.
4. $\dfrac{\partial \mathcal{F}}{\partial b} = f_2'(b) \leq 0 \Rightarrow \mathcal{F}$ es no creciente en la segunda variable.

También observamos que si $f' \geq 0$ entonces obtenemos que,

$$\mathcal{F}(a,b) = f_1(a) = f(a) \tag{101}$$

o si $f' \leq 0$ tenemos que,

$$\mathcal{F}(a,b) = f_2(b) = f(b) \tag{102}$$

que nos da los dos casos del método *upwind*, por tanto el método es de flujo monótono. Similarmente, veamos que siempre es posible encontrar tal descomposición bajo una condición CFL. Imponemos la condición

$$\frac{h}{k} \geq \text{máx}\{|f'(s)| : s \in [U_m,U_M]\} \tag{103}$$

Entonces, descomponemos f tal que

$$f_1(s) = \frac{1}{2}f(s) + \frac{h}{2k}s \tag{104}$$

$$f_2(s) = \frac{1}{2}f(s) - \frac{h}{2k}s \tag{105}$$

Bajo la condición tenemos que,

$$f_1'(s) = \frac{1}{2}f'(s) + \frac{h}{2k} \geq 0 \tag{106}$$

$$f_2'(s) = \frac{1}{2}f'(s) - \frac{h}{2k} \leq 0 \tag{107}$$

Esta descomposición sirve para cualquier función $f \in C^1(\mathbb{R})$ y más aún nos revuelve el método de Lax-Friedrichs que por tanto, es un esquema de flujo monótono.

3.4.2. Estabilidad L^∞ para flujos monótonos

Los esquemas monótonos son estables bajo una condición CFL que solo depende de las constantes de Lipschitz del flujo. Lo demostramos en la siguiente proposición.

Proposición 13. *Consideramos un mallado $\mathcal{T}_{h,k}$ y sea $k > 0$. Sea $u_{h,k}$ la solución aproximada según (20) obtenida para un esquema de flujo monótono. Sean g_1, g_2 las constantes de Lipschitz de $\mathcal{F}$ en $[U_m, U_M]^2$ con respeto a sus dos variables. Bajo la condición CFL*

$$k \leq \frac{h}{g_1 + g_2} \tag{108}$$

la solución aproximada cumple,

$$U_m \leq u_{h,k}(x,t) \leq U_M \qquad \forall x \in \mathbb{R}, t \in \mathbb{R}_+ \tag{109}$$

Demostración: partimos del esquema general,

$$U_i^{n+1} = U_i^n - \frac{k}{h}\left[\mathcal{F}(U_i^n, U_{i+1}^n) - \mathcal{F}(U_{i-1}^n, U_i^n)\right] \tag{110}$$

$$= U_i^n - \frac{k}{h}\left[\mathcal{F}(U_i^n, U_{i+1}^n) - f(U_i^n)\right] + \frac{k}{h}\left[\mathcal{F}(U_{i-1}^n, U_i^n) - f(U_i^n)\right] \tag{111}$$

$$= U_i^n + \frac{k}{h}\left[\frac{\mathcal{F}(U_i^n, U_{i+1}^n) - f(U_i^n)}{U_i^n - U_{i+1}^n}\right](U_{i+1}^n - U_i^n) \tag{112}$$

$$- \frac{k}{h}\left[\frac{\mathcal{F}(U_{i-1}^n, U_i^n) - f(U_i^n)}{U_{i-1}^n - U_i^n}\right](U_i^n - U_{i-1}^n), \text{ si } U_i^n - U_{i+1}^n \neq 0 \tag{113}$$

Entonces definimos los siguientes coeficientes,

$$b^n_{i+\frac{1}{2}} = \begin{cases} \frac{k}{h}\frac{\mathcal{F}(U^n_i,U^n_{i+1})-f(U^n_i)}{U^n_i-U^n_{i+1}} & \text{si } U^n_i - U^n_{i+1} \neq 0 \\ 0 & \text{si } U^n_i - U^n_{i+1} = 0 \end{cases} \tag{114}$$

$$a^n_{i-\frac{1}{2}} = \begin{cases} \frac{k}{h}\frac{\mathcal{F}(U^n_{i-1},U^n_i)-f(U^n_i)}{U^n_{i-1}-U^n_i} & \text{si } U^n_{i-1} - U^n_i \neq 0 \\ 0 & \text{si } U^n_{i-1} - U^n_i = 0 \end{cases} \tag{115}$$

Así,

$$U^{n+1}_i = U^n_i + b^n_{i+\frac{1}{2}}(U^n_{i+1} - U^n_i) - a^n_{i-\frac{1}{2}}(U^n_i - U^n_{i-1}) \tag{116}$$

$$= (1 - b^n_{i+\frac{1}{2}} - a^n_{i-\frac{1}{2}})U^n_i + b^n_{i+\frac{1}{2}}U^n_{i+1} + a^n_{i-\frac{1}{2}}U^n_{i-1} \tag{117}$$

Recordando que $\mathcal{F}(a,a) = f(a)$,

$$b^n_{i+\frac{1}{2}} \leq \left|\frac{k}{h}\frac{\mathcal{F}(U^n_i,U^n_{i+1}) - f(U^n_i)}{U^n_i - U^n_{i+1}}\right| \tag{118}$$

$$= \left|\frac{k}{h}\frac{\mathcal{F}(U^n_i,U^n_{i+1}) - \mathcal{F}(U^n_i,U^n_i)}{U^n_i - U^n_{i+1}}\right| \tag{119}$$

Usando que $\mathcal{F}$ es no creciente en la segunda variable y que es Lipschitz en $[U_m, U_M]^2$,

$$0 \leq b^n_{i+\frac{1}{2}} \leq \frac{k}{h}\frac{g_2|U^n_i - U^n_{i+1}|}{|U^n_i - U^n_{i+1}|} \tag{120}$$

$$\leq \frac{g_2 k}{h} \ \forall i \in \mathbb{Z} \tag{121}$$

Y análogamente obtenemos que,

$$0 \leq a^n_{i-\frac{1}{2}} \leq \frac{g_1 k}{h} \ \forall i \in \mathbb{Z} \tag{122}$$

Ahora, aplicando la condición CFL, obtenemos que,

$$0 \leq b^n_{i+\frac{1}{2}} \leq 1 \tag{123}$$

$$0 \leq a^n_{i-\frac{1}{2}} \leq 1 \tag{124}$$

$$0 \leq a^n_{i-\frac{1}{2}} + b^n_{i+\frac{1}{2}} \leq 1 \tag{125}$$

Por lo tanto, tenemos una combinación convexa de U^n_i, U^n_{i+1}, U^n_{i-1} y obtenemos la cota,

$$\text{mín}\{U^n_i, U^n_{i+1}, U^n_{i-1}\} \leq U^{n+1}_i \leq \text{máx}\{U^n_i, U^n_{i+1}, U^n_{i-1}\} \ \forall i \in \mathbb{Z}, \ \forall n \in \mathbb{N}_0 \tag{126}$$

Una sencilla inducción nos permite obtener el resultado,

$$U_m \leq U^n_i \leq U_M \quad \forall i \in \mathbb{Z}, \ \forall n \in \mathbb{N} \tag{127}$$

□

3.5. Esquema Lax-Wendroff

El esquema Lax-Wendroff tiene la siguente forma,

$$U_i^{n+1} = U_i^n - \frac{k}{2h}\left[f(U_{i+1}^n) - f(U_{i-1}^n)\right] + \frac{k^2}{2h^2}\left[f'\left(\frac{U_i^n + U_{i+1}^n}{2}\right)\cdot\left(f(U_{i+1}^n) - f(U_i^n)\right)\right.$$
$$\left. - f'\left(\frac{U_i^n + U_{i-1}^n}{2}\right)\cdot\left(f(U_i^n) - f(U_{i-1}^n)\right)\right] \quad (128)$$

Para el caso lineal, el método queda,

$$U_i^{n+1} = U_i^n - \frac{ak}{2h}[U_{i+1}^n - U_{i-1}^n] + \frac{a^2k^2}{2h^2}[U_{i+1}^n - 2U_i^n + U_{i-1}^n] \quad (129)$$

Escrito en la forma general, derivada antes, tenemos que el flujo numérico esta dado por,

$$\mathcal{F}(U_i^n, U_{i+1}^n) = F_{i+\frac{1}{2}}^n \quad (130)$$
$$= \frac{1}{2}\left(f(U_i^n) + f(U_{i+1}^n)\right) - \frac{k}{2h}f'\left(\frac{U_i^n + U_{i+1}^n}{2}\right)[f(U_{i+1}^n) - f(U_i^n)] \quad (131)$$

Claramente tenemos que,

$$\mathcal{F}(u,u) = \frac{1}{2}(f(u) + f(u)) - \frac{k}{2h}f'\left(\frac{2u}{2}\right)[f(u) - f(u)] \quad (132)$$
$$= f(u) \quad (133)$$

Por tanto el esquema LW verifica la condición de consistencia.

3.5.1. Estabilidad de von-Neumann para la ecuación lineal

Si seguimos estabilidad del método LW con la misma manera de abordar como los otros métodos, llegamos a una condición $\frac{k}{h} = \frac{1}{|a|}$ que deja el método trivial e inútil. Por lo tanto, consideramos otra filosofía llamada *estabilidad de von-Neumann*. Consideramos que nuestra aproximación en cada punto tiene la forma,

$$U_i^n = \psi^n e^{\mathrm{i}Kx_i} \quad (134)$$

donde $\mathrm{i}^2 = -1$. Entonces, aplicando el esquema para el caso lineal,

$$\psi^{n+1}e^{\mathrm{i}Kx_i} = \psi^n e^{\mathrm{i}Kx_i} - \frac{ak}{2h}[\psi^n e^{\mathrm{i}Kx_{i+1}} - \psi^n e^{\mathrm{i}Kx_{i-1}}] + \frac{a^2k^2}{2h^2}[\psi^n e^{\mathrm{i}Kx_{i+1}} + \psi^n e^{\mathrm{i}Kx_{i-1}} - 2\psi^n e^{\mathrm{i}Kx_i}] \quad (135)$$

Cancelamos $\psi^n e^{\mathrm{i}Kx_i}$ y usamos que $x_{i\pm1} = x_i \pm h$.

$$\psi = 1 - \frac{ak}{2h}[e^{\mathrm{i}Kh} - e^{-\mathrm{i}Kh}] + \frac{a^2k^2}{2h^2}[e^{\mathrm{i}Kh} + e^{-\mathrm{i}Kh} - 2] \quad (136)$$
$$= 1 - \frac{ak}{h}[\mathrm{i}\sin(Kh)] + \frac{a^2k^2}{2h^2}[2\cos(Kh) - 2] \quad (137)$$
$$= [1 + \frac{a^2k^2}{h^2}(\cos(Kh) - 1)] - \mathrm{i}[\frac{ak}{h}\sin(Kh)] \quad (138)$$

Observamos que,

$$U_i^{n+1} = \psi U_i^n \tag{139}$$

Entonces, la magnitud de la solución numérica aumenta por un factor de ψ para cada paso en tiempo. Por lo tanto, si queremos que la solución no crezca en norma en cada paso de tiempo, necesitamos que $|\psi| \leq 1$. Esto nos da una condición (CFL). Por simplicidad, llamamos $c = \frac{ak}{h}$

$$\begin{aligned}
0 < |\psi|^2 &= (1 + c^2(\cos(Kh) - 1))^2 + c^2 \sin^2(Kh) && (140)\\
&= (1 + 2c^2(\cos(Kh) - 1) + c^4(\cos(Kh) - 1)^2) + c^2 \sin^2(Kh) && (141)\\
&= 1 + c^2[2\cos(Kh) - 2 + \sin^2(Kh)] + c^4[\cos^2(Kh) - 2\cos(Kh) + 1] && (142)\\
&= 1+c^2[2(1-\sin^2(Kh/2))-2+4\sin^2(Kh/2)\cos^2(Kh/2)]+c^4[\cos^2(Kh)-2\cos(Kh)] && (143)\\
&= 1+c^2[-4\sin^2(Kh/2)+4\sin^2(Kh/2)(1-\sin^2(Kh/2))]+c^4[\cos^2(Kh)-2\cos(Kh)] && (144)\\
&= 1 - 4c^2 \sin^4(Kh/2) + c^4[(1 - 2\sin^2(Kh/2))^2 - 2(1 - 2\sin^2(Kh/2)) + 1] && (145)\\
&= 1-4c^2 \sin^4(Kh/2)+c^4[1-4\sin^2(Kh/2)+4\sin^4(Kh/2)-2+4\sin^2(Kh/2)+1] && (146)\\
&= 1 - 4c^2 \sin^4(Kh/2) + 4c^4 \sin^4(Kh/2) && (147)\\
&= 1 - c^2(1 - c^2)4\sin^4(Kh/2) && (148)
\end{aligned}$$

Esta expresión es $\leq 1 \iff$

$$\begin{aligned}
&\iff c^2(1 - c^2)4\sin^4(Kh/2) \geq 0 && (149)\\
&\iff (1 - c^2) \geq 0 && (150)\\
&\iff c^2 \leq 1 && (151)\\
&\iff |c| \leq 1 && (152)\\
&\iff \frac{|a|k}{h} \leq 1 && (153)\\
&\iff \frac{k}{h} \leq \frac{1}{|a|} && (154)
\end{aligned}$$

Por lo tanto hemos obtenido la condición (CFL) $\frac{k}{h} \leq \frac{1}{|a|}$.

Sin embargo, como en el caso del *upwind*, necesitamos calcular la derivada de f (o Jacobiano si u es una función vectorial). Esto puede ser lioso o difícil. Por tanto, estudiamos dos variantes del esquema más prácticas que evitan el cálculo del Jacobiano usando un método multipaso muy parecido a los métodos de predicción-corrección.

3.5.2. Método de Richtmyer

El método de Richtmyer se basa en la idea de que podemos evitar el cálculo del Jacobiano si estimamos el valor de la solución en un paso intermedio de tiempo usando el método

LF y entonces usando este valor para aproximar el valor en t_n. Entonces, el primer paso queda,

$$U_{i+\frac{1}{2}}^{n+\frac{1}{2}} = \frac{1}{2}(U_{i+1}^n + U_i^n) - \frac{k}{2h}(f(U_{i+1}^n) - f(U_i^n)) \tag{155}$$

$$U_{i-\frac{1}{2}}^{n+\frac{1}{2}} = \frac{1}{2}(U_i^n + U_{i-1}^n) - \frac{k}{2h}(f(U_i^n) - f(U_{i-1}^n)) \tag{156}$$

Entonces, usando este valor de la solucíon en el tiempo $t_{n+\frac{1}{2}}$ y t_n podemos aproximar el valor en t_{n+1}. El segundo paso queda,

$$U_i^{n+1} = U_i^n - \frac{k}{h}\left[f(U_{i+\frac{1}{2}}^{n+\frac{1}{2}}) - f(U_{i-\frac{1}{2}}^{n+\frac{1}{2}})\right] \tag{157}$$

3.5.3. Método de MacCormack

El método de MacCormack es similar porque también usa dos pasos. La idea aquí es de usar diferencias *forward* en un paso y diferencias *backward* en el segundo paso o al revés. Entonces hay dos formas de esta variante.

Primero está el método *Forward-Backward MacCormack*

Paso 1:

$$U_i^* = U_i^n - \frac{k}{h}(f(U_{i+1}^n) - f(U_i^n)) \tag{158}$$

Paso 2:

$$U_i^{n+1} = \frac{1}{2}(U_i^n + U_i^*) - \frac{k}{2h}(f(U_i^*) - f(U_{i-1}^*)) \tag{159}$$

Y análogamente tenemos el *Backward-Forward MacCormack* que queda,

Paso 1:

$$U_i^* = U_i^n - \frac{k}{h}(f(U_i^n) - f(U_{i-1}^n)) \tag{160}$$

Paso 2:

$$U_i^{n+1} = \frac{1}{2}(U_i^n + U_i^*) - \frac{k}{2h}(f(U_{i+1}^*) - f(U_i^*)) \tag{161}$$

4. Resultados Numéricos y Ejemplos

Consideramos dos ejemplos concretos para calcular y comparar los errores y ordenes de convergencias de los 3 métodos. Para el esquema Lax-Wendroff, usamos la variante de Richtmyer. Al final de esta sección hemos escrito un *link* a videos de las soluciones reproducidas en el tiempo.

Ejemplo 1: Transporte lineal

$$\frac{\partial u}{\partial t} + 7\frac{\partial u}{\partial x} = 0 \quad t \in [0,1] \;\; x \in [0,1] \tag{162}$$

$$u(x,0) = \sin(x) \qquad x \in [0,1] \tag{163}$$

Trivialmente este problema tiene solución exacta,

$$u(x,t) = \sin(x - 7t) \tag{164}$$

Ejemplo 2: Ecuación de Burgers no viscoso

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0 \quad t \in [0,10] \;\; x \in [0,1] \tag{165}$$

$$u(x,0) = 2x + 1 \qquad x \in [0,1] \tag{166}$$

Esta ecuación tiene solución exacta,

$$u(x,t) = \frac{2x+1}{2t+1} \tag{167}$$

La ecuación de Burgers tiene $f(u) = \frac{1}{2}u^2$. Entonces,

$$f'(u) = u \geq 0 \tag{168}$$

Usando que el dato inicial es siempre positivo y aplicando el principio del máximo, sabemos que esta derivada es no negativa. Por tanto, podemos aplicar el esquema *upwind* con solo un caso.

Ejemplo 3: Ecuación de Burgers con 'shocks'

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0 \quad t \in [0,10] \;\; x \in [0,1] \tag{169}$$

$$u(x,0) = e^{-x^2} \qquad x \in [-12,12] \tag{170}$$

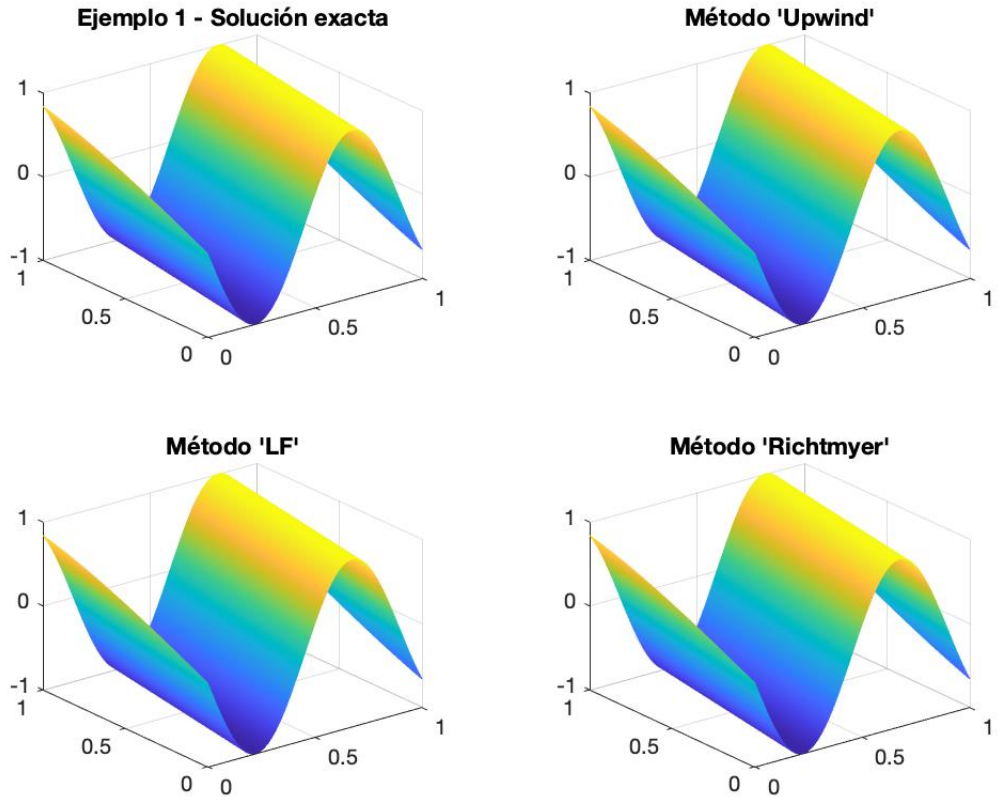
No conocemos la solución exacta de este problema

4.1. Ejemplo 1. Transporte lineal

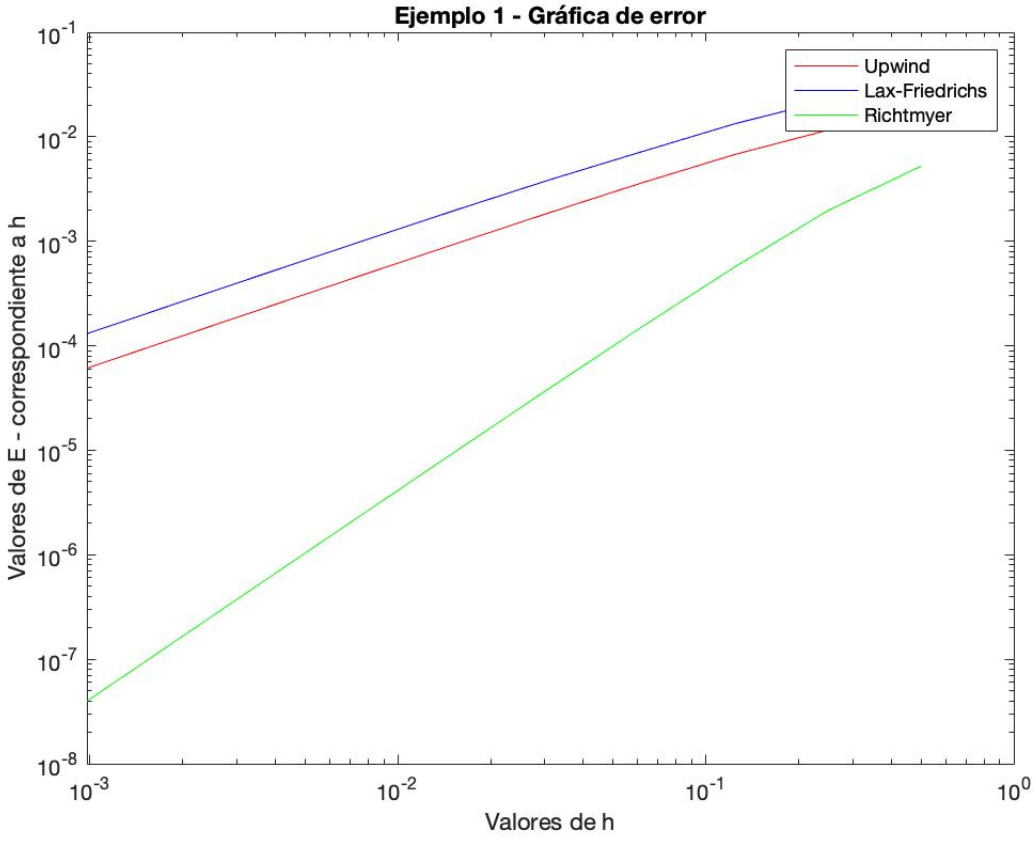
Consideramos una sucesión de $h_j = \frac{1}{2^j}$ con $j = 1, ..., 10$. Fijamos $k = \frac{h}{8}$ para cumplir la condición CFL. Calculamos el error máximo entre la solución exacta y la solución numérica para los tres métodos. Mostramos los grafos de h contra el error máximo en escala

logarítmica y calculamos la pendiente de las rectas para estimar el orden de convergencia experimental.

Primero mostramos la solución exacta con las soluciones numéricas calculadas por los diferentes métodos.



A ojo, los métodos producen buenas aproximaciones de la solución exacta. Ahora mostramos los errores en escala logarítmica,



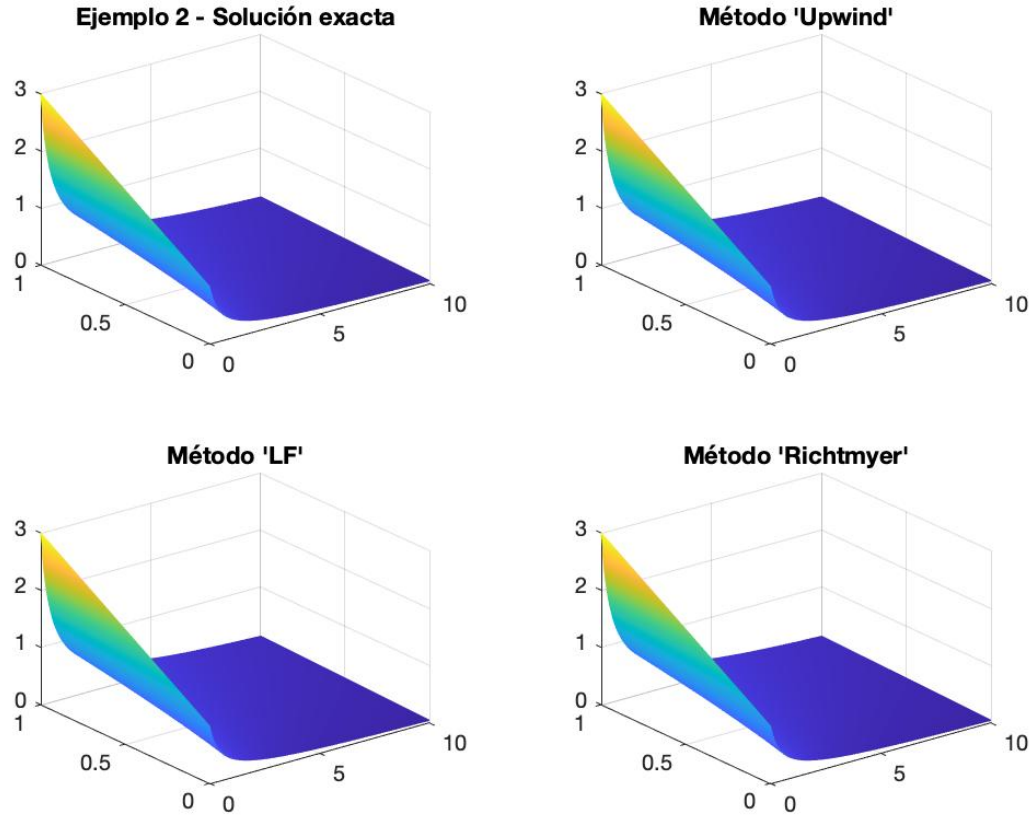
Claramente, tenemos convergencia de las soluciones numéricas hasta la solución exacta porque cuando bajamos h, y por tanto k, el error entre la numérica y la exacta baja también. Las pendientes de las rectas del los métodos *upwind* y Lax-Friedrichs son ambos 1 significando que el orden de convergencia experimental de ambos métodos es 1. La recta del método Richtmyer (Lax-Wendroff) tiene pendiente 2 y por tanto, este método converge doblemente más rápido que los otros dos. También observamos que el método *upwind* produzca errores más pequeños en este caso que el LF.

Ejemplo 1: Errores y valores de α

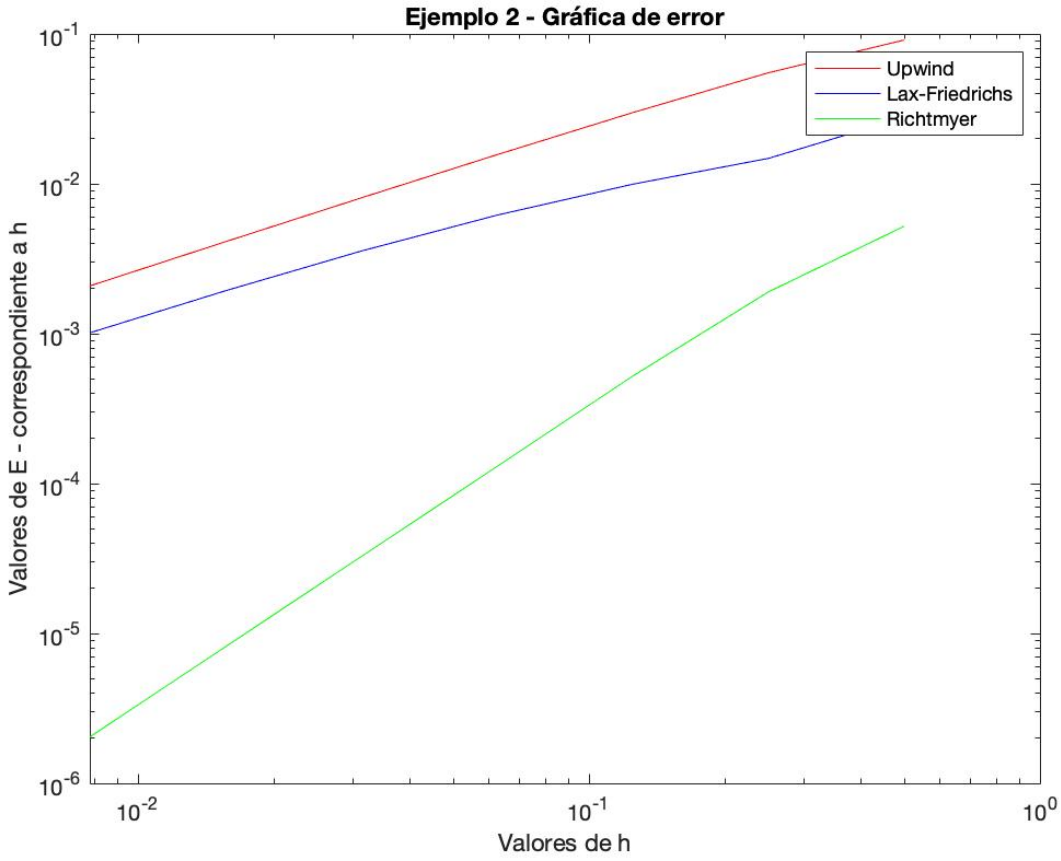
	Upwind		Lax-Friedrichs		Richtmyer	
k	Error máximo	α	Error máximo	α	Error máximo	α
5	0.0018902	0.9636	0.00389617	0.9240	3.97190e-05	1.9686
6	0.00096083	0.9822	0.00201573	0.9634	1.00662e-05	1.9857
7	0.00048435	0.9912	0.00102477	0.9819	2.53224e-06	1.9936
8	0.00024316	0.9956	0.00051674	0.9895	6.34776e-07	1.9972
9	0.00012182	0.9978	0.00025993	0.9935	1.58879e-07	1.9988
10	6.0974e-05	0.9985	0.00013036	0.9957	3.97374e-08	1.9994

4.2. Ejemplo 2. Ecuación de Burgers no viscoso

Consideramos una sucesión de $h_j = \frac{1}{2^j}$ con $j = 1, \ldots, 7$ y mostramos los mismos resultados.



Otra vez, parece que tenemos esquemas convergentes. Mostramos los errores en escala logarítmica,



Similarmente, nos salen rectas con las mismas pendientes que en el Ejemplo 1. Para el esquema *upwind* y LF, nos sale 1 y para el método de Richtmyer, nos sale 2. Sin embargo, observamos que en este ejemplo los errores del método de LF son menores que los del *upwind*.

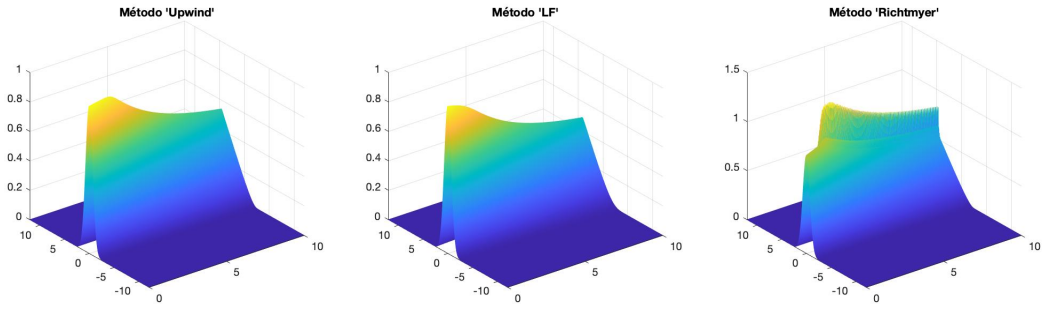
Ejemplo 2: Errores y valores de α

	Upwind		Lax-Friedrichs		Richtmyer	
k	Error máximo	α	Error máximo	α	Error máximo	α
2	0.055075	0.8045	0.014780	0.7413	0.001901	1.6604
3	0.029861	0.9080	0.009939	0.6292	0.000521	1.9349
4	0.015642	0.9450	0.006178	0.7406	0.000130	1.9998
5	0.008057	0.9642	0.003560	0.8376	3.2598e-05	1.9973
6	0.004109	0.9755	0.001935	0.9064	8.1585e-06	1.9988
7	0.002084	0.9797	0.001013	0.9327	2.0406e-06	1.9993

4.3. Ejemplo 3: Ecuación de Burgers con choques

Para este ultimo ejemplo, no conocemos la solución exacta. Por lo tanto no podemos calcular errores ni ordenes de convergencia. Mostramos este ejemplo para demostrar experimentalmente que los métodos monótonos convergen a soluciones no clásicas (lo hemos

comprobado para el método *upwind* para la ecuación lineal). Con los tres métodos dibujamos las soluciones numéricas.



Desdes las soluciones en 3D, podemos ver que el método *upwind* y el método Lax-Friedrichs convergen similarmente sin inestabilidades al contrario que el de Richtmyer.

El comportamiento de las soluciones se observa mucho mejor viendo secciones cruzadas de la soluciones para cada paso en tiempo. Esto los mostramos en forma de video, que se puede encontrar en **esta lista de reproducción de Youtube** o en la URL: `https://www.youtube.com/playlist?list=PLXhFGfjdT6WK6j7njosild4ldxc15tycl`.

Estos vídeos comparan más claramente los métodos. El método *upwind* parece comportarse mejor que los otros dos. El método LF también converge sin inestabilidades pero parece que pierde un poco de masa a lo largo de tiempo. Esto puede ser por el pequeño término difusivo que viene de la discretización. El método de Richtmyer produce inestabilidades pero viendo las secciones cruzadas parece que las inestabilidades se muestran solo en el pico de la solución, y en el resto, la solución es suave.

5. Mallado esférico

El objetivo inicial del proyecto era resolver la ecuación de advección-difusión en la esfera. Con el fin de comprender mejor las matemáticas que subyacen hubo que restringirse a un problema más modesto en una dimensión. No obstante, hemos conseguido implementar en MATLAB un mallado esferico refinable iterativamente. Se expone a continuación.

Seguiremos algunos de los pasos que expone [Gir97, p. 203 - 204], comenzando por definir los doce puntos de un icosaedro incrito en la esfera unidad. Usando $\lambda \in [0, 2\pi]$ como longitud y $\theta \in [-\pi/2, \pi/2]$ como latitud, se definen los puntos del icosaedro: primero el polo norte y sur, luego cinco puntos más en el hemisferio norte y finalmente otros cinco en el sur.

$$[\lambda_1, \theta_1] = \left[0, \frac{\pi}{2}\right] \quad [\lambda_{12}, \theta_{12}] = \left[0, -\frac{\pi}{2}\right], \tag{171}$$

$$[\lambda_i, \theta_i] = \left[\left(i - \frac{3}{2}\right)\frac{2\pi}{5}, 2\arcsin\left(\frac{1}{2\cos\left(\frac{3\pi}{10}\right)}\right) - \frac{\pi}{2}\right], \text{ para } i = 2, \ldots, 6 \tag{172}$$

$$[\lambda_i, \theta_i] = \left[(i-7)\frac{2\pi}{5}, -2\arcsin\left(\frac{1}{2\cos\left(\frac{3\pi}{10}\right)}\right) + \frac{\pi}{2}\right], \text{ para } i = 7, \ldots, 11 \tag{173}$$

Los 12 puntos definen en total 20 triángulos equiláteros que forman las caras del icosaedro y que llamaremos *elementos* (véase la siguiente figura).

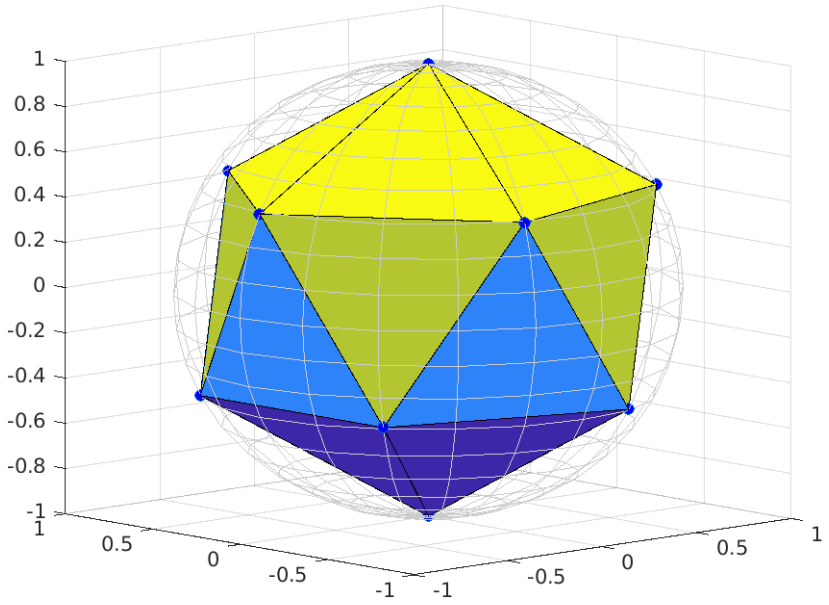


Figura 1: Icosaedro inscrito en la esfera unidad según las ecuaciones 171, 172 y 173.

En la iteración de refinamiento, cada elemento es divido en otros cuatro triángulos equiláteros. Para ello, se obtiene el punto medio de cada borde y se unen entre sí como se muestra a continuación.

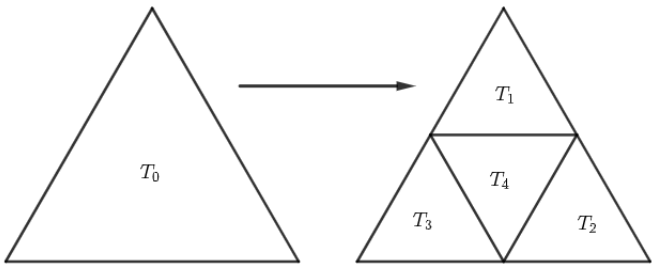


Figura 2: División de un elemento en cuatro triángulos equiláteros.

Sean por ejemplo $\mathbf{x}_1$ y $\mathbf{x}_2$ los dos puntos que definen el borde de un elemento. Entonces, el punto medio es $\mathbf{x}_4 = (\mathbf{x}_1 + \mathbf{x}_2)/2$. Para que $\mathbf{x}_4$ pertenezca a la esfera de radio a simplemente se proyecta,

$$\mathbf{x}_4 = a\frac{\mathbf{x}_4}{|\mathbf{x}_4|}$$

La complejidad del proceso de refinamiento reside en identificar a la perfección cada elemento con cada terna de puntos y no perderla al iterar. En [Gir97] se esgrime la eficiencia de su método, pero por falta de comprensión lo hemos modificado. Primero se crea la matrix **ipoin[1:npoin, 1:6]**, donde **npoin** es el número total de puntos (inicialmente 12). La posición 1 guarda el número identificador del punto, las posiciones 2 - 4 recogen sus coordenadas espaciales y las posiciones 5 - 6 los identificadores de los puntos *padre*.

La matriz **iedge[1:nedge, 1:2]**, donde **nedge** es el número de bordes (inicialmente 30), recoge en cada fila los pares que definen cada borde.

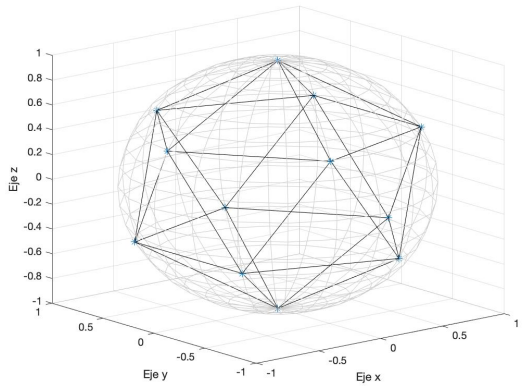
Ahora bien, como cada elemento se divide en otros 4, el refinamiento genera una estructura de datos *quadtree* o árbol cuaternario[2] que se recoge en la matriz **itree[1:ntree, 1:9]**. El valor **ntree** es el número de elementos del árbol. Las posiciones 1 - 3 están reservadas para los números de identificación de cada punto. La posición 4 contiene el número identificativo del elemento *padre* que da lugar al actual. Las posiciones 5 - 8 recogen el número identificativo de los elementos *hijo*. Finalmente, la posición 9 guarda el identificador del elemento actual y es cero si el elemento no está activo.

El algoritmo de refinamiento consiste en los siguientes pasos:

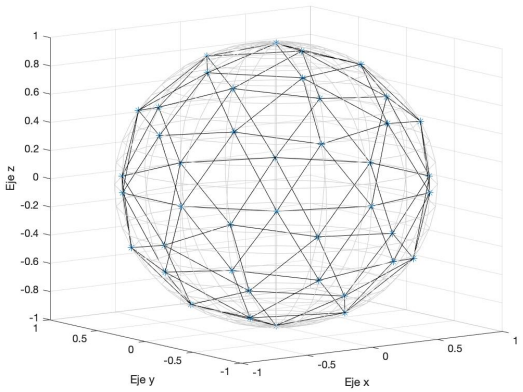
1. Calcular los nuevos puntos y guardarlos en **ipoin**.
2. Crear el nuevo **itree**:
   a. Ubicar los elementos activos e iterar sobre ellos.
   b. Recuperar el identificador de los puntos que generan cada elemento y aquel de cada punto medio que formará un nuevo elemento.
   c. Con todos estos datos, guardar al final de **itree** los cuatro nuevos elementos.
3. Generar el nuevo **iedge** iterando sobre los elementos activos de **itree**. Quitar los bordes repetidos.

[2]ver `https://en.wikipedia.org/wiki/Quadtree`

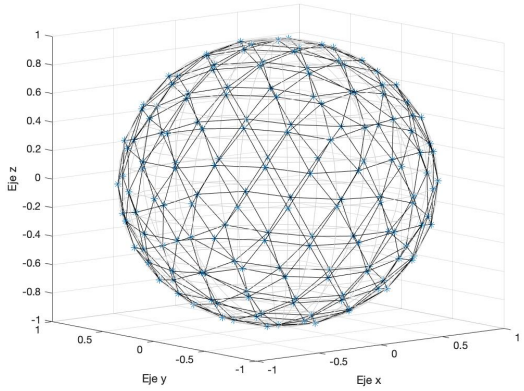
Mostramos el resultado del proceso iterativo:



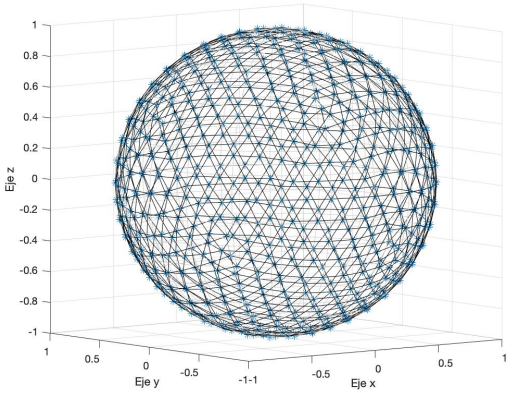
(a) Icosaedro inicial



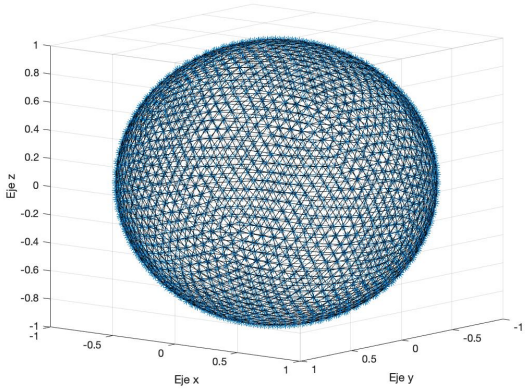
(b) Iteración 1



(c) Iteración 2



(d) Iteración 3



(e) Iteración 4

Figura 3: Mallado de una esfera a partir de un icosaedro.

Para conseguir el mallado, ejecútese el *script* `icos_mesh.m` que ejecuta las funciones `icos_init.m`, `icos_refine.m` e `icos_plot.m`. Véanse los *Códigos 9, 10, 12 y 11* del *Apéndice*.

6. Apéndice

6.1. Implementaciones de Métodos *upwind*

Código 1: UpwindFixedPos.m

```
%Metodo Upwind caso f'(u)>0
function [U] = UpwindFixedPos(F,h,k,T,a,b,g)
x=[a:h:b]; %malla en espacio

t=[0:k:T]; %malla en tiempo
N_x = (b-a)/h; N_t = T/k;
U = zeros(N_x+1,N_t+1);

fun_F = str2func(strcat('@(u)','0*u+',F));
fun_ini = str2func(strcat('@(x,t)','0*x+0*t+',g));
U(:,1)=fun_ini(x,0);
U(1,:)=fun_ini(a,t);
U(N_x+1,:)=fun_ini(b,t);

for n=2:N_t+1
    for i=2:N_x
        f=(fun_F(U(i,n-1))-fun_F(U(i-1,n-1)));
        U(i,n)=U(i,n-1)-(k/h)*(f);
    end
end
end
```

Código 2: UpwindFixedNeg.m

```
%Metodo Upwind caso f'(u)<0
function [U] = UpwindFixedNeg(F,h,k,T,a,b,g)
x=[a:h:b]; %malla en espacio

t=[0:k:T]; %malla en tiempo
N_x = (b-a)/h; N_t = T/k;
U = zeros(N_x+1,N_t+1);

fun_F = str2func(strcat('@(u)','0*u+',F));
fun_ini = str2func(strcat('@(x,t)','0*x+',g));
U(:,1)=fun_ini(x,0);
U(1,:)=fun_ini(a,t);
U(N_x+1,:)=fun_ini(b,t);

for n=2:N_t+1
    for i=2:N_x
        f=(fun_F(U(i+1,n-1))-fun_F(U(i,n-1)));
        U(i,n)=U(i,n-1)-(k/h)*(f);
    end
end
```

6.2. Implementación del Método Lax-Friedrichs

Código 3: LaxFriedrichs.m

```
%Metodo Lax-Friedrichs
function [U] = LF(F,h,k,T,a,b,g)
x=[a:h:b]; %malla en espacio

t=[0:k:T]; %malla en tiempo
N_x = (b-a)/h; N_t = T/k;
U = zeros(N_x+1,N_t+1);

fun_F = str2func(strcat('@(u)','0*u+',F));
fun_ini = str2func(strcat('@(x,t)','0*x+0*t+',g));
U(:,1)=fun_ini(x,0);
U(1,:)=fun_ini(a,t);
U(N_x+1,:)=fun_ini(b,t);

for n=2:N_t+1
    for i=2:N_x
        U(i,n)=(0.5)*(U(i+1,n-1) + U(i-1,n-1)) - ...
            (k/(2*h))*(fun_F(U(i+1,n-1)) - fun_F(U(i-1,n-1)));
    end
end
end
```

6.3. Implementación del Método Richtmyer

Código 4: Richtmyer.m

```
%Metodo Richtmyer
function [U] = Richtmyer(F,h,k,T,a,b,g)
x=[a:h:b]; %malla en espacio

t=[0:k:T]; %malla en tiempo
N_x = (b-a)/h; N_t = T/k;
U = zeros(N_x+1,N_t+1);

fun_F = str2func(strcat('@(u)','0*u+',F));
fun_ini = str2func(strcat('@(x,t)','0*x+',g));
U(:,1)=fun_ini(x,0);
U(1,:)=fun_ini(a,t);
U(N_x+1,:)=fun_ini(b,t);

for n=2:N_t+1
    for i=2:N_x
        mid_upper = (0.5)*(U(i+1,n-1) + U(i,n-1)) - ...
            (k/(2*h))*(fun_F(U(i+1,n-1)) - fun_F(U(i,n-1)));
        mid_lower = (0.5)*(U(i,n-1) + U(i-1,n-1)) - ...
            (k/(2*h))*(fun_F(U(i,n-1)) - fun_F(U(i-1,n-1)));
        U(i,n)=U(i,n-1) - (k/h)*(fun_F(mid_upper) - fun_F(mid_lower));
    end
end
```

```
83 end
```

6.4. Implementaciones de Métodos MacCormack

Código 5: FBMacCormack.m

```
84 %Metodo Forward-Backward MacCormack
85 function [U] = FBMacCormack(F,h,k,T,a,b,g)
86 x=[a:h:b]; %malla en espacio
87 t=[0:k:T]; %malla en tiempo
88 N_x = (b-a)/h; N_t = T/k;
89 U = zeros(N_x+1,N_t+1);
90
91 fun_F = str2func(strcat('@(u)','0*u+',F));
92 fun_ini = str2func(strcat('@(x,t)','0*x+',g));
93 U(:,1)=fun_ini(x,0);
94 U(1,:)=fun_ini(a,t);
95 U(N_x+1,:)=fun_ini(b,t);
96
97 for n=2:N_t+1
98     for i=2:N_x
99         star_for = U(i,n-1) - (k/h)*(fun_F(U(i+1,n-1) - ...
              fun_F(U(i,n-1))));
100        star_back = U(i-1,n-1) - (k/h)*(fun_F(U(i,n-1)) - ...
              fun_F(U(i-1,n-1)));
101        U(i,n)=(0.5)*(U(i,n-1)+star_for) - ...
              (k/(2*h))*(fun_F(star_for) - fun_F(star_back));
102    end
103 end
104 end
```

Código 6: BFMacCormack.m

```
105 %Metodo Backward-Forward MacCormack
106 function [U] = FBMacCormack(F,h,k,T,a,b,g)
107 x=[a:h:b]; %malla en espacio
108 t=[0:k:T]; %malla en tiempo
109 N_x = (b-a)/h; N_t = T/k;
110 U = zeros(N_x+1,N_t+1);
111
112 fun_F = str2func(strcat('@(u)','0*u+',F));
113 fun_ini = str2func(strcat('@(x,t)','0*x+',g));
114 U(:,1)=fun_ini(x,0);
115 U(1,:)=fun_ini(a,t);
116 U(N_x+1,:)=fun_ini(b,t);
117
118 for n=2:N_t+1
119     for i=2:N_x
120         star_for = U(i,n-1) - (k/h)*(fun_F(U(i+1,n-1) - ...
              fun_F(U(i,n-1))));
121         star_back = U(i-1,n-1) - (k/h)*(fun_F(U(i,n-1)) - ...
              fun_F(U(i-1,n-1)));
```

```
122          U(i,n)=(0.5)*(U(i,n-1) + star_back) - ...
                 (k/(2*h))*(fun_F(star_for) - fun_F(star_back));
123      end
124  end
125  end
```

6.5. Ficheros para experimentos y vídeos

Código 7: Experimento.m

```
126  %Convergencia experimental
127  %F= '7*u';T=1;a=0;b=1;g='sin(x-7*t)';sol='sin(x-7*t)'; %linear ...
       transport - Ejemplo 1
128  F='0.5.*u^2'; T=10;a=0;b=1; g='(2*x+1).*((2*t+1).^(-1))'; ...
       sol='(2*x+1).*((2*t+1).^(-1))'; %inviscid burgers equation with ...
       linear initial condition - Ejemplo 2
129  error = zeros(7,4);grads=zeros(7,4);
130  for i=1:7
131      h=1/(2^i); k=h/8; %cfl
132      V1 = UpwindFixedPos(F,h,k,T,a,b,g); %si sabemos que la velocidad ...
           es positivo (e.g. Burger's)
133      V2=LF(F,h,k,T,a,b,g);
134      V3=Richtmyer(F,h,k,T,a,b,g);
135      u = str2func(strcat('@(x,t)','0*x+',sol));
136      x=[a:h:b];
137      t=[0:k:T];
138      [X,Y]=meshgrid(x,t);
139      exacta = (u(X,Y)).';
140      E1=abs(V1-exacta);
141      E2=abs(V2-exacta);
142      E3=abs(V3-exacta);
143      error(i,1)=i;
144      grad(i,1)=i;
145      error(i,2)=max(max(E1));
146      error(i,3)=max(max(E2));
147      error(i,4)=max(max(E3));
148  end
149  tiledlayout(2,2);
150  nexttile
151  mesh(t,x,exacta)
152  title("Ejemplo 2 - Solucion exacta")
153  nexttile
154  mesh(t,x,V1)
155  title("Metodo 'Upwind'")
156  nexttile
157  mesh(t,x,V2)
158  title("Metodo 'LF'")
159  nexttile
160  mesh(t,x,V3)
161  title("Metodo 'Richtmyer'")
162  steps = 2.^(-error(:,1));
163  figure()
164  loglog(steps,error(:,2),'r')
165  hold on
```

```
loglog(steps,error(:,3),'b')
hold on
loglog(steps,error(:,4),'g')
legend('Upwind','Lax-Friedrichs','Richtmyer')
title("Ejemplo 2 - Grafica de error")
xlabel("Valores de h")
ylabel("Valores de E - correspondiente a h")
grad(:,2)=gradient(log(error(:,2)),log(steps)).';
grad(:,3)=gradient(log(error(:,3)),log(steps)).';
grad(:,4)=gradient(log(error(:,4)),log(steps)).';
writematrix(error,'example2error.txt','Delimiter','tab');
writematrix(grads,'example2grads.txt','Delimiter','tab');
```

Código 8: VideoMaker.m

```
%Videos de Burgers con 'shocks'
%burgers with shocks example 1 - movie
F='0.5.*u^2'; T=10;a=-12;b=12;g='exp(-x.^2)';
h=1/32; k=h/2;
V1 = UpwindFixedPos(F,h,k,T,a,b,g);
V2=LF(F,h,k,T,a,b,g);
V3=Richtmyer(F,h,k,T,a,b,g);
x=[a:h:b];
S=size(V3);N=S(2);
for n=1:N
    axis([-10 10 0 1])
    plot(x,V1(:,n))
    hold on
    title("Ejemplo 3 - La ecuacion de Burgers")
    axis([-10 10 0 1])
    plot(x,V2(:,n))
    axis([-10 10 0 1])
    plot(x,V3(:,n))
    axis([-10 10 0 1])
    legend('Upwind','Lax-Friedrichs','Richtmyer')
    hold off
    M(n)=getframe(gcf);
end
movie(M);
burgerstodos2 = VideoWriter('burgerstodos2.mp4','MPEG-4');
open(burgerstodos2)
writeVideo(burgerstodos2,M)
close(burgerstodos2)
```

6.6. Implementación del mallado esférico

Código 9: icos_mesh.m

```
close all
clear all

r = 1;  % radio
refine_num = 1;  % Numero de bucles de refinamiento.
```

```
211
212 % Inicializamos ipoin, iedge e itree
213 [ipoin, iedge, itree] = icos_init(r);
214
215 % Dibujamos datos inicializados.
216 icos_plot(ipoin, iedge)
217
218 if refine_num > 0
219     for i = 1:refine_num
220     [ipoin, iedge, itree] = icos_refine(ipoin, iedge, itree, r);
221     end
222 end
223
224 % Dibujamos datos refinados
225 icos_plot(ipoin, iedge)
```

Código 10: icos_init.m

```
226 function [ipoin, iedge, itree] = icos_init(r)
227 %ICOS_INIT Esta funcion inicializa las matrices ipoin, iedge e
228 %itree con los datos del icosaedro inicial. Para mas informacion
229 %sobre cada una de estas matrices mirar las funciones init_ipoin,
230 %init_iedge e init_itree que ese encuentran en este mismo fichero.
231
232 % Inicializamos ipoin, iedge e itree
233 [ipoin, npoin] = init_ipoin(r);
234 [iedge, nedge] = init_iedge();
235 [itree, ntree] = init_itree();
236
237 end
238
239 function [ipoin, npoin] = init_ipoin(r)
240 %INIT_IPOIN Esta funcion inicializa la matriz ipoin con los 12 ...
      puntos de un
241 %icosaedro.
242 %   La variable de entrada es el radio de la esfera, r.
243 %   Las variable de salida son, npoin, el numero de puntos e
244 %   ipoin, la matriz de vertices emulando la nomenclatura de
245 %   Giraldo:
246 %       - Cada fila representa los datos de un punto, i.
247 %       - La primera columna es el identificador del punto, id.
248 %       - La segunda, tercera y cuarta son las coordenadas xi, yi, zi.
249 %       - La quinta y sexta muestran los parent de dichos puntos.
250
251 % El numero de puntos es doce.
252 npoin = 12;
253
254 % Creamos los vectores longitud (lambda) y latitud (theta).
255 lambda = zeros(1, 12);
256 theta = zeros(1, 12);
257
258 % Tambien el creamos el vector de identificadores.
259 ipoint_n = zeros(1, 12);
260
261
262 % Introducimos el polo norte (ipoint_n 1) y el polo sur (ipoint_n 12)
263 % respectivamente
```

```
lambda(1) = 0;
theta(1) = pi/2;
ipoint_n(1) = 1;

lambda(12) = 0;
theta(12) = -pi/2;
ipoint_n(12) = 12;

% Ahora los cinco puntos del hemisferio norte.
for i = 2:6
    lambda(i) = (i - 3/2)*2*pi/5;
    theta(i) = 2*asin( 1/( 2*cos( 3*pi/10 ) ) ) - pi/2;
    ipoint_n(i) = i;
end

% igual con los puntos del hemisferio sur.
for i = 7:11
    lambda(i) = (i - 7)*2*pi/5;
    theta(i) = - 2*asin( 1/( 2*cos( 3*pi/10 ) ) ) + pi/2;
    ipoint_n(i) = i;
end

% Inicializamos y calculamos las coordendas.
xi = zeros(1, 12);
yi = zeros(1, 12);
zi = zeros(1, 12);

for i = 1:12
    xi(i) = r*cos( theta(i) )*cos( lambda(i) );
    yi(i) = r*cos( theta(i) )*sin( lambda(i) );
    zi(i) = r*sin( theta(i) );
end

% Creamos la matriz
ipoin = [ipoint_n; xi; yi,; zi; zeros(1,12); zeros(1,12)];
ipoin = ipoin';

end

function [iedge, nedge] = init_iedge(¬)
%INIT_IEDGE Esta funcion inicializa la matriz iedge con las 30
%aristas de un icosaedro.
%   La variable de entrada es la matriz ipoin creada segun se explica
%   en init_ipoin.m
%   Las variable de salida son, nedge, el numero de aristas e iedge, la
%   matriz de aristas segun la nomenclatura de Giraldo:
%       - La columnas 1 y 2 indentifican los puntos que generan la ...
    arista.
%       - Las columnas 3 y 4 identifican los dos triangulos que ...
    comparten
%         esta arista.
%   NOTA: para rellenar esta primera matriz se ha adoptado un ...
    convenio que
%   no aparece en el Giraldo. Puede servirnos para encontrar errores ...
    en la
%   inicializacion de init_iedge:
%       - En la columna 1 ponemos el identificador de punto de menor ...
    valor.
```

```
317 %       - Si tomamos el camino del punto de la columna 1 al punto de la
318 %         columna 2, entonces, el identificador de triangulo que ...
       ponemos en
319 %         la columna 3 es el del triangulo a la derecha.
320
321 % El numero de aristas es treinta.
322 nedge = 30;
323
324 % Inicializamos la matriz.
325 iedge = zeros(nedge, 4);
326
327 % Rellenamos la matriz a mano.
328 % Primero la tapa del hemisferio norte.
329 iedge(1,:) = [1, 2, 5, 1];
330 iedge(2,:) = [1, 3, 1, 2];
331 iedge(3,:) = [1, 4, 2, 3];
332 iedge(4,:) = [1, 5, 3, 4];
333 iedge(5,:) = [1, 6, 4, 5];
334
335 % Ahora el paralelo norte (aristas entre los puntos 2, 3, 4, 5 y 6).
336 iedge(6,:) = [2, 3, 6, 1];
337 iedge(7,:) = [3, 4, 7, 2];
338 iedge(8,:) = [4, 5, 8, 3];
339 iedge(9,:) = [5, 6, 9, 4];
340 iedge(10,:) = [2, 6, 5, 10];
341
342 % Ahora aristas entre el paralelo norte y el paralelo sur.
343 iedge(11,:) = [2, 7, 10, 11];
344 iedge(12,:) = [2, 8, 11, 6];
345 iedge(13,:) = [3, 8, 6, 12];
346 iedge(14,:) = [3, 9, 12, 7];
347 iedge(15,:) = [4, 9, 7, 13];
348 iedge(16,:) = [4, 10, 13, 8];
349 iedge(17,:) = [5, 10, 8, 14];
350 iedge(18,:) = [5, 11, 14, 9];
351 iedge(19,:) = [6, 11, 9, 15];
352 iedge(20,:) = [6, 7, 15, 10];
353
354 % Ahora el paralelo sur (aristas entre los puntos 7, 8, 9, 10 y 11).
355 iedge(21,:) = [7, 8, 16, 11];
356 iedge(22,:) = [8, 9, 17, 12];
357 iedge(23,:) = [9, 10, 18, 13];
358 iedge(24,:) = [10, 11, 19, 14];
359 iedge(25,:) = [7, 11, 15, 20];
360
361 % Finalmente la tapa del hemisferio sur.
362 iedge(26,:) = [7, 12, 20, 16];
363 iedge(27,:) = [8, 12, 16, 17];
364 iedge(28,:) = [9, 12, 17, 18];
365 iedge(29,:) = [10, 12, 18, 19];
366 iedge(30,:) = [11, 12, 19, 20];
367
368 end
369
370 function [itree, ntree] = init_itree()
371 %INIT_ITREE Esta funcion inicializa la matriz itree con los 20
372 %elementos/triangulos del icosaedro.
373 %   No tiene variables de entrada.
```

```
374 %   Las variable de salida son, npoin, el numero de puntos e
375 %   ipoint, la matriz de vertices emulando la nomenclatura de
376 %   Giraldo:
377 %       - Cada fila representa los datos de un punto, i.
378 %       - La primera columna es el identificador del punto, id.
379 %       - La segunda, tercera y cuarta son las coordenadas xi, yi, zi.
380
381 % El numero de elementos o caras inicial es 20.
382 ntree = 20;
383
384 % Inicializamos el itree como una cell array.
385 itree = num2cell(zeros(ntree, 9));
386
387 % Rellenamos el id de cada uno de los 20 elementos.
388 itree(:,9) = num2cell((1:20)');
389
390 % Rellenamos manualmente los id de puntos que identifican dichos
391 % elementos.
392 %Primero la tapa del hemisferio norte.
393 itree(1,1:3) = {1, 2, 3};
394 itree(2,1:3) = {1, 3, 4};
395 itree(3,1:3) = {1, 4, 5};
396 itree(4,1:3) = {1, 5, 6};
397 itree(5,1:3) = {1, 2, 6};
398
399 % Ahora el paralelo norte (elementos de la mitad superior).
400 itree(6,1:3) = {2, 3, 8};
401 itree(7,1:3) = {3, 4, 9};
402 itree(8,1:3) = {4, 5, 10};
403 itree(9,1:3) = {5, 6, 11};
404 itree(10,1:3) = {2, 6, 7};
405
406 % Ahora el paralelo sur.
407 itree(11,1:3) = {2, 7, 8};
408 itree(12,1:3) = {3, 8, 9};
409 itree(13,1:3) = {4, 9, 10};
410 itree(14,1:3) = {5, 10, 11};
411 itree(15,1:3) = {6, 7, 11};
412
413 % Ahora la tapa sur
414 itree(16,1:3) = {7, 8, 12};
415 itree(17,1:3) = {8, 9, 12};
416 itree(18,1:3) = {9, 10, 12};
417 itree(19,1:3) = {10, 11, 12};
418 itree(20,1:3) = {11, 7, 12};
419
420 % Rellenamos manualmente los futuros id de los elementos child
421 % (ubicaciones 5 - 8). Tambien el parent id (ubicacion 4), en este
422 % caso sera 0.
423 % Primero la tapa del hemisferio norte.
424 itree(1,4:8) = {0, 1, 2, 3, 4};
425 itree(2,4:8) = {0, 5, 6, 7, 8};
426 itree(3,4:8) = {0, 9, 10, 11, 12};
427 itree(4,4:8) = {0, 13, 14, 15, 16};
428 itree(5,4:8) = {0, 17, 18, 19, 20};
429
430 % Ahora el paralelo norte (elementos de la mitad superior).
431 itree(6,4:8) = {0, 21, 22, 23, 24};
```

```
itree(7,4:8) = {0, 25, 26, 27, 28};
itree(8,4:8) = {0, 29, 30, 31, 32};
itree(9,4:8) = {0, 33, 34, 35, 36};
itree(10,4:8) = {0, 37, 38, 39, 40};

% Ahora el paralelo sur.
itree(11,4:8) = {0, 41, 42, 43, 44};
itree(12,4:8) = {0, 45, 46, 47, 48};
itree(13,4:8) = {0, 49, 50, 51, 52};
itree(14,4:8) = {0, 53, 54, 55, 56};
itree(15,4:8) = {0, 57, 58, 59, 60};

% Ahora la tapa sur
itree(16,4:8) = {0, 61, 62, 63, 64};
itree(17,4:8) = {0, 65, 66, 67, 68};
itree(18,4:8) = {0, 69, 70, 71, 72};
itree(19,4:8) = {0, 73, 74, 75, 76};
itree(20,4:8) = {0, 77, 78, 79, 80};

end
```

Código 11: icos_plot.m

```
function icos_plot(ipoin, iedge)
%UNTITLED3 Summary of this function goes here
%   Detailed explanation goes here

% Dibujamos datos inicializados.
plotipoint(ipoin)
plotiedge(ipoin, iedge)

end

function plotipoint(ipoin)
%PLOTIPOINT Esta funcion representa en tres dimensiones los puntos ...
   de una
%matriz ipoint estructurada segun se explica en la funcion init_ipoint.
%   Variables de entrada:
%       - ipoint: es la matriz de puntos de la esfera.
%       - label: su valor debe ser "true" o "false". Para label = true,
%         sobre cada punto se muestrara su identificador como etiqueta.
figure()
plot3(ipoin(:,2), ipoin(:,3), ipoin(:,4), '*')
xlabel('Eje x')
ylabel('Eje y')
zlabel('Eje z')
grid on

end

function plotiedge(ipoin, iedge)
%PLOTIPOINT Esta funcion representa en tres dimensiones los puntos ...
   de una
%matriz ipoint estructurada segun se explica en la funcion init_ipoint.
%   Variables de entrada:
%       - ipoint: es la matriz de puntos de la esfera.
%       - label: su valor debe ser "true" o "false". Para label = true,
```

```
%         sobre cada punto se muestrara su identificador como etiqueta.

% El numero de aristas.
nedge = max(size(iedge));

% Queremos representar cada arista en el mismo grafico. Usamos hold on
hold on

for i = 1:nedge
    % Recuperamos las coordenadas de la arista i.
    [xyz1, xyz2] = search(i, ipoin, iedge);
    xyz = [xyz1; xyz2];

    % Representamos la arista
    plot3(xyz(:,1), xyz(:,2), xyz(:,3), 'k-')
end


hold off
end

function [xyz1, xyz2] = search(i, ipoin, iedge)
%SEARCH Encuentra las coordenadas de los puntos que forman la arista
%iedge(i,:).

% Recuperamos el identificador de los puntos que forman la arista.
ipoin1 = iedge(i,1);
ipoin2 = iedge(i,2);

% Recuperamos las coordenadas usando el identificador.
xyz1 = ipoin(ipoin1,2:4);
xyz2 = ipoin(ipoin2,2:4);

end
```

Código 12: icos_refine.m

```
function [ipoin2, iedge2, itree2] = icos_refine(ipoin, iedge, itree, r)
%ICOS_REFINE Esta funcion genera un bucle de refinamiento de la malla.

%% Calculamos el nuevo ipoin
% Calculemos el numero de puntos y de aristas.
npoin = max(size(ipoin));
nedge = max(size(iedge));

% Inicializamos ipoin2
ipoin2 = ipoin;

% Inicializamos las coordenadas de los puntos medios de las aristas.
xyz = zeros(nedge, 3);

% Recorremos las aristas. Calculamos las coordenadas.
for i = 1:nedge
    % Recuperamos las coordenadas de la arista i.
    [point1, point2, xyz1, xyz2] = search(i, ipoin, iedge);

    % Calculamos el punto medio
```

```
538      midpoin = 1/2*( xyz1 + xyz2 );
539
540      % Grabamos el punto sobre la esfera
541      xyz(i,:) = r*midpoin./norm( midpoin );
542
543      % Grabamos en ipoin2
544      newpoin = [npoin + i, xyz(i,:), point1, point2];
545      ipoin2(npoin + i, :) = newpoin;
546  end
547
548  %% Calculamos el nuevo itree
549
550  % Calculamos npoint2, ntree y ntree2.
551  npoin2 = max(size(ipoin2));
552  ntree = max(size(itree));
553  ntree2 = ntree + 2*(npoin2 - 2);
554
555  % Inicializamos itree2
556  itree2 = itree;
557
558  % Debemos conocer cual es el parent activo (ubicacion 4): ...
       recuperamos el id
559  % de cada elemento de itree. Calculamos la primera ubicacion no ...
       nula. Por
560  % construccion st_active debe ser 1.
561  elem = cell2mat(itree(:,9));
562  st_active = find(elem);
563  st_active = st_active(1);
564
565  if elem(st_active) ¬= 1
566      error("El primer elemento activo debe tener id 1.")
567  end
568
569  % Recuperamos parent activo.
570  pt_active = itree{st_active, 4};
571
572  % Ahora creamos el nuevo itree con un bucle for.
573  for i = ntree+1:ntree2
574
575      % Anulamos la ubicacion 9 del elemento inicial
576      elem_0 = st_active + floor( (i - ntree - 1)/4);
577      itree2{elem_0, 9} = 0;
578
579      % Recuperamos id de los puntos del elemento.
580      point1 = itree{elem_0,1};
581      point2 = itree{elem_0,2};
582      point3 = itree{elem_0,3};
583
584      % Buscamos id de las combinaciones.
585      id_12 = parent_id(ipoin2, [point1, point2]);
586      id_13 = parent_id(ipoin2, [point1, point3]);
587      id_23 = parent_id(ipoin2, [point2, point3]);
588
589      % Nombramos posiciones de los nuevos elementos.
590      pos5 = itree{elem_0, 5};
591      pos6 = itree{elem_0, 6};
592      pos7 = itree{elem_0, 7};
593      pos8 = itree{elem_0, 8};
```

```
594
595     resto = rem(i,4);
596     if resto == 1
597         % Primer nuevo elemento
598         itree2(i,:) = {point1, id_12, id_13, pt_active + 1,...
599             4*(pos5-1)+1, 4*(pos5-1)+2, 4*(pos5-1)+3,...
600             4*(pos5-1)+4, pos5};
601     elseif resto == 2
602         % Segundo nuevo elemento
603         itree2(i,:) = {id_12, point2, id_23, pt_active + 1,...
604             4*(pos6-1)+1, 4*(pos6-1)+2, 4*(pos6-1)+3,...
605             4*(pos6-1)+4, pos6};
606     elseif resto == 3
607         % Tercer nuevo elemento
608         itree2(i,:) = {id_13, id_23, point3, pt_active + 1,...
609             4*(pos7-1)+1, 4*(pos7-1)+2, 4*(pos7-1)+3,...
610             4*(pos7-1)+4, pos7};
611     else
612         % Cuarto nuevo elemento
613         itree2(i,:) = {id_12, id_13, id_23, pt_active + 1,...
614             4*(pos8-1)+1, 4*(pos8-1)+2, 4*(pos8-1)+3,...
615             4*(pos8-1)+4, pos8};
616     end
617
618
619 end
620
621 %% Creamos iedge2
622 % ATENCION: este iedge tendra aristas repetidas.
623 % Anadimos las nuevas aristas a iedge. ATENCION: no seguimos el ...
       convenio de
624 % las caras. No se como se hace para nombrar las nuevas caras y ...
       saber cual
625 % esta a cada lado.
626 iedge2 = zeros(3*(ntree2 - ntree), 2);
627
628 for i = ntree+1:ntree2
629     % Recuperamos id de los puntos que forman el triangulo nuevo.
630     id_1 = itree2{i,1};
631     id_2 = itree2{i,2};
632     id_3 = itree2{i,3};
633
634     % Calculamos un indice que nos ayude a rellenar iedge2
635     iedge_ix = i - ntree - 1;
636
637     % Rellenamos iedge2
638     iedge2(3*iedge_ix + 1, 1:2) = [id_1, id_2];
639     iedge2(3*iedge_ix + 2, 1:2) = [id_1, id_3];
640     iedge2(3*iedge_ix + 3, 1:2) = [id_2, id_3];
641 end
642
643 % Vamos a quitar los elementos repetidos: ordenamos los pares y ...
       usamos la
644 % funcion unique.
645 nedge2 = max(size(iedge2));
646 for i = 1:nedge2
647     if iedge2(i,1) > iedge2(i,2)
648         iedge2(i,:) = [iedge2(i,2), iedge2(i,1)];
```

```
649     end
650 end
651
652 iedge2 = unique(iedge2, 'rows');
653
654
655 end
656
657
658 function [ipoin1, ipoin2, xyz1, xyz2] = search(i, ipoin, iedge)
659 %SEARCH Encuentra las coordenadas de los puntos que forman la arista
660 %iedge(i,:).
661
662 % Recuperamos el identificador de los puntos que forman la arista.
663 ipoin1 = iedge(i,1);
664 ipoin2 = iedge(i,2);
665
666 % Recuperamos las coordenadas usando el identificador.
667 xyz1 = ipoin(ipoin1,2:4);
668 xyz2 = ipoin(ipoin2,2:4);
669
670 end
671
672 function child_id = parent_id(ipoin, parent)
673 % PARENT_ID encuentra el child_id de los puntos parent = [id1, id2]
674
675 list = ipoin(:, 5:6) == parent;
676 list = list(:,1) & list(:,2);
677
678 % Puede ser que los indices de parent esten dados la vuelta.
679 if any(list) == false
680     parent = [parent(2), parent(1)];
681     list = ipoin(:, 5:6) == parent;
682     list = list(:,1) & list(:,2);
683 end
684
685 child_id = find(list);
686
687 end
```

Referencias

[Gir97] Francis X. Giraldo. «Lagrange–Galerkin Methods on Spherical Geodesic Grids». En: *JOURNAL OF COMPUTATIONAL PHYSICS 136, 197–213 (1997)* (1997), págs. 203-207.

[Kar06] Kenneth H. Karlsen. «Notes on weak convergence». En: Chapter 1 (2006), pág. 2.

[Dro08] Jérôme Droniou. «A beginner's course in finite volume approximation of scalar conservation laws». En: (2008).

[Bre10] Haim Brezis. *Functional Analysis, Sobolev Spaces and Partial Differential Equations*. New York, NY: Springer New York, 2010. ISBN: 978-0-387-70913-0. DOI: 10.1007/978-0-387-70914-7. URL: http://link.springer.com/10.1007/978-0-387-70914-7.

[LeV10] Randall J. LeVeque. «Nonlinear Scalar Conservation Laws - Ch.11». En: *Finite Vol. Methods Hyperbolic Probl.* Chapter 11 (2010), págs. 203-226. DOI: 10.1017/cbo9780511791253.012.

[F C13] M. Ribot F. Coquel T. Goudon. «Num. #2: Hyperbolic PDE equation : transport equation». En: *Summer School on Numerics and Control of PDEs* (2013), págs. 1-3.

[Eym+19] Robert Eymard y col. *Finite Volume Methods To cite this version : HAL Id : hal-02100732*. Vol. M. Part 3. 2019. ISBN: 9780444503503.