CIMAT

Centro de Investigación en Matemáticas

Solución de la ecuación de Burgers con el método de Galerkin discontinuo en un problema de control óptimo

Tesis presentada por
Lourdes Bibiana Merino Solís
para obtener el grado de Maestro en Ciencias

2019

Especialidad de Computación y Matemáticas Industriales

Agradecimientos

Este trabajo de tesis fue realizado en el Centro de Investigación en Matemáticas, A. C. (CIMAT) Unidad Guanajuato, el cual fue completado con la ayuda de distintas personas que directa o indirectamente estuvieron explicándome, enseñándome, corrigiéndome, teniéndome paciencia y apoyándome en todo momento.

En primer lugar quiero agradecer al Consejo Nacional de Ciencia y Tecnología (CONACYT) por la beca otorgada para la realización de los estudios de maestría y del proyecto de tesis en CIMAT. Del mismo modo, agradezco a CIMAT por permitirme la oportunidad de llevar a cabo esta experiencia, además de brindarme el apoyo intelectual y material necesario para terminar este trabajo.

A mi asesor el Dr. Miguel Ángel Moreles Vázquez y mi co-asesor el Dr. Joaquín Peña Acevedo por toda su ayuda y paciencia en el desarrollo de la tesis. Por su confianza puesta en mi para terminar el trabajo.

A mis sinodales, el Dr. Alonso Ramírez Manzanares y el Daniel Alejandro Cervantes Cabrera por su revisiones y comentarios.

A mis amigos y compañeros que siempre estuvieron apoyándome académica y emocionalmente.

Y finalmente pero no menos importante a mi familia, a mi mamá Ma. Luz Solís Zarate, mi papá Cayetano Merino Peréz y mis hermanas María de la Luz Merino Solís y María del Pilar Merino Solís, por que sin el apoyo de ellos no podría haber llegado al lugar y momento en el que estoy ahora.

Índice general

Capítulo 1

Introducción

El presente trabajo de tesis tiene como objetivo principal resolver el problema inverso de la ecuación de Burgers en el caso no viscoso en horizontes de tiempo largo, utilizando como *solver* para la ecuación diferencial el método de Galerkin Discontinuo. En otras palabras, a partir de una función objetivo conocida determinar la condición inicial de esta, la cual es dejada evolucionar hasta un tiempo T mediante la ecuación de Burgers no viscosa para alcanzar la función objetivo. Asimismo se propone una nueva forma de escoger las variables a las que estará sujeta la optimización, las cuales conforman la parametrización de la función inicial.

Los métodos numéricos para problemas dominados por convección, como lo es la ecuación de Burgers, son muy útiles en una amplia variedad de aplicaciones, como meteorología, oceanografía, dinámica de gases, turbomaquinaría, flujos turbulentos, flujos granulares, simulaciones de recuperación de petróleo, modelado de aguas poco profundas, transporte de contaminantes de medios porosos, flujos viscoelásticos, simulación de dispositivos semiconductores, magneto-hidrodinámica y electromagnetismo entre muchas otras. Esta es la razón por la cual el diseño de métodos robustos, precisos y eficientes para resolver numéricamente estos problemas es de gran importancia.

La ecuación de Burgers es una ecuación no lineal perteneciente a las leyes de conservación, la cual presenta discontinuidades en el caso no viscoso. Diversos estudios han mostrado que a pesar de los saltos la ecuación presenta solución única ya que se cumple con tanto con la condición de Rankine-Hugonist como con la de entropía de

Lax [5], por lo que que es necesario un método computacional que permita abordar discontinuidades para resolver numéricamente esta ecuación.

Uno de los métodos diseñados para resolver este problema es el *Runge-Kutta Discontinuos Galerkin*, el cual fue desarrollado como parte de este trabajo de tesis. El hecho de que este método utiliza una aproximación discontinua aunado a que se aplican flujos numéricos en las fronteras de los elementos del mallado permite que este método sea la mejor opción para el manejo de las discontinuidades. Además, este presenta las ventajas de que es altamente paralelizable y que posee un alto orden en tiempo y espacio.

La parte de optimización fue realizada con el método de Nelder-Mead [18] debido a que este es utilizado generalmente en casos no lineales, no requiere del cálculo de derivadas y garantiza el no incremento del valor de la función a optimizar, haciendo que este método sea confiable y fácil de implementar.

Para llevar a cabo este trabajo, se desarrollo un programa en el lenguaje de programación C, implementando los métodos anteriormente mencionados. Primero se realizaron diversas pruebas para verificar el funcionamiento del solver, tanto en el caso lineal como en la ecuación de Burgers, inicialmente se probó el caso lineal ya que este era más sencillo de comparar con la literatura además de facilitar la detección de errores.

Una vez verificado el Runge-Kutta Discontinuos Galerkin, se pasó a trabajar con los problemas inversos, en los cuales también se trabajaron ejemplos lineales como no lineales, las primeras pruebas se realizaron sobre casos sintéticos, mientras que la última es una comparación de nuestro método contra los resultados obtenidos por [1].

El trabajo de tesis está dividido en 8 capítulos contando la introducción, la cual corresponde al capítulo uno, mientras que en el segundo capítulo se da un explicación más detallada del problema a abordar.

En el tercer capítulo corresponde a la explicación del modelo, esto es, de donde surge la ecuación de Burgers y por que es valido trabajar con ella, en otras palabras, por que existe solución en el caso no viscoso a pesar de la presencia de discontinuidades en ésta.

El cuarto capítulo corresponde a la explicación del método solución, por lo que

este se divide en el desarrollo del Runge-Kutta Discontinuos Galerkin y la explicación del funcionamiento del Nelder-Mead. En este capítulo se explica a detalle la obtención del algoritmo para Galerkin Discontinuo, se describen las ventajas del Runge-Kutta diseñado para este método y se muestra el funcional utilizado como función a minimizar.

El capítulo 5 muestra una explicación del algoritmo para resolver el problema inverso, mostrando todos los pasos necesarios para llevarlo a cabo. Además en este capítulo se describe la propuesta a utilizar para parametrizar la función inicial.

Los resultados se encuentran en el capítulo 6, este se encuentra dividido en tres secciones, la primera donde se muestran los resultados obtenidos solamente por del *solver*, la segunda en la que se trabajan los casos sintéticos y para terminar, la tercera sección, la cual contiene el ejemplo ya trabajado anteriormente en la literatura.

Finalmente las conclusiones de la tesis y el trabajo futuro se encuentran en el capítulo 7 y 8 respectivamente.

Capítulo 2

Formulación de Problema

La ecuación de Burges es una ecuación no lineal perteneciente a las leyes de transporte, que se caracteriza por tener soluciones discontinuas para el caso no viscoso. En la literatura existen diversas aplicaciones sobre control óptimo de la ecuación de Burgers, ya que esta se aplica en problemas de física, ingeniería, economía y administración, algunos ejemplos son: modelos hidrológicos [6], identificación de fuentes de calor [15] y la minimización del *sonic boom* (ruptura de la barrera del sonido) generado por naves aéreas supersónicas [1].

Debido a lo anterior, es de gran interés tener un programa capaz de resolver la ecuación de Burgers, y más aún resolver el problema inverso de ésta. Un problema inverso interesante de resolver es la ecuación de Burgers en el caso no viscoso en horizontes de tiempo largo en 1D, esto es la minimización del siguiente funcional:

$$min_{u_0} J(u_0) = min_{u_0} \frac{1}{2} \int_{\mathbb{R}} (u(x,T) - u^*(x))^2 dx. \tag{2.1}$$

Sujeto a

$$\partial_t u(x,t) + (u(x,t)\partial_x(u(x,t)) = 0 \quad x \in \mathbb{R},\ t > 0, \tag{2.2}$$

$$u(x,0) = u_0(x). \tag{2.3}$$

Donde J es la función a minimizar, $u(x,T)$ es la solución de la ecuación de Burgers después de un tiempo T, u^* es la función conocida al tiempo T y u_0 es la condición

inicial por determinar.

Una buen candidato para resolver la ecuación de Burgers es el método de Garlekin Discontiuo Runge-Kutta (RKDG por sus siglas en inglés), ya que este combina una clase especial de Runge-Kutta con una discretización espacial de elementos finitos por aproximaciones discontinuas. Estas dos características otorgan al método un alto orden en tiempo y espacio. Además el método implementa la idea de los flujos numéricos, permitiendole trabajar con discontinuidades en la solución de la ecuación, la cual converge a la solución de entropía, por lo que la solución numérica obtenida es única.

Capítulo 3

Modelo

Los modelos en ciencia usualmente contienen 3 ingredientes: leyes básicas, principios de conservación y relaciones constitutivas.

Las leyes básicas expresan propiedades que están bien establecidas y mediante mediciones, comparan teoría y observación. Una ley básica es aplicable a una gran clase de sistemas, esto contrasta con una relación constitutiva la cual caracteriza un sólo sistema.

Los principios de conservación también son referidos como leyes de balance. Como su nombre lo indica, están basadas en la conservación de alguna propiedad. Estos se obtienen a partir de las observaciones y el análisis del sistema, esta visión conduce a las ecuaciones de equilibrio. Algunos ejemplos de estos principios son la conservación total de masa, momento y energía en algún sistema, esto quiere decir que estas cantidades incrementan en alguna parte de sistema mientras que en el resto decrecen, por lo que en el sistema global estas cantidades se mantienen. Cabe aclarar que estas cantidades no siempre se conservan separadamente.

3.1. Leyes de conservación

Todos las leyes de conservación en medios que fluyen pueden ser derivadas de un principio general llamado teorema de transporte.

Para obtener este teorema se considera que se tiene una propiedad con densidad $f(\vec{x}, t)$, que se transporta a través de un sistema o un medio, cuya densidad de flujo

correspondiente está dada por $Q(\vec{x}, t)$, mientras que la tasa a la que se produce y/o aniquila está dada por la fuente/sumidero de densidad $S(\vec{x}, t)$.

Tomando un volumen arbitrario V en el medio, la cantidad de $f(\vec{x}, t)$ en V está dada por

$$F(t) = \int_V f(\vec{x}, t) dV. \tag{3.1}$$

La conservación de $f(\vec{x}, t)$ implica que el cambio en $F(t)$ es igual al flujo que existe a través de la superficie A más la producción de $f(\vec{x}, t)$ en V.

$$\frac{dF}{dt} = \int_V \frac{\partial f}{\partial t} dV = -\int_A Q \cdot n dA + \int_V S dV. \tag{3.2}$$

El primer término del lado derecho de la igualdad en la ec. 3.2 representa el flujo que sale a través de la superficie A, mientras que el segundo término corresponde a la fuente.

Aplicando el teorema de Gauss a la ecuación 3.2 llegamos a que la conservación de f en V en cualquier tiempo es expresado por la forma integral del teorema de transporte:

$$\int_V (\partial_t f + \nabla \cdot Q - S) dV = 0. \tag{3.3}$$

Si el integrando es continuo, y dado que la ec. 3.3 es válida para cualquier volumen V, se puede escribir su forma diferencial.

$$\partial_t f + \nabla \cdot Q = S. \tag{3.4}$$

Por lo tanto el teorema de transporte relaciona a la propiedad f con el flujo Q y la fuente S.

3.2. Ecuación de Burgers

La ecuación de Burgers es una ecuación no lineal perteneciente a las leyes de transporte, la cual puede presentar soluciones de tipo onda de choque (soluciones discontinuas) que cumplen tanto la condición de Rankine-Hugonist como la condición

de entropía de Lax, donde $f = u$ y $Q = \frac{u^2}{2} - \nu\partial_x u$ con ν siendo un coeficiente de viscosidad, por lo que el primer término de Q corresponde a la advectiva, mientras que el segundo término corresponde a la parte difusiva. La ecuación de Burgers en 1D se expresa como

$$\frac{\partial}{\partial t}u(x,t) + \frac{\partial}{\partial x}(\frac{u^2(x,t)}{2}) = \nu\frac{\partial^2}{\partial x^2}u(x,t). \tag{3.5}$$

El caso viscoso conduce a un problema parabólico no lineal, mientras que el caso no viscoso, $\nu = 0$, constituye una ley hiperbólica de conservación no lineal, en la cual se presentan soluciones discontinuas. La ecuación 3.5 para el caso no viscoso se escribe de la siguiente manera:

$$\frac{\partial}{\partial t}u(x,t) + u(x,t)\frac{\partial}{\partial x}u(x,t) = 0. \tag{3.6}$$

Cabe mencionar que en un tiempo cercano al inicial, el problema viscosos con un bajo coeficiente de viscosidad y el caso no viscoso poseen un comportamiento similar, pero después de cierto tiempo, la viscosidad transforma el problema hiperbólico a uno parabólico.

3.3. Ondas de choque

Tomemos la siguiente ecuación diferencial parcial en 1D.

$$\frac{\partial u}{\partial t} + c(x,t,u(x,t))\ \frac{\partial u}{\partial x} = r(x,t,u(x,t)) \quad x \in \mathbb{R},\ t > 0, \tag{3.7}$$

$$u(x,0) = w(x), \tag{3.8}$$

donde se asume que c y r son clase C^1 y la condición inicial $w : \mathbb{R} \to \mathbb{R}$ también es C^1. El coeficiente c es establecido como la velocidad de la onda, esto puede ser verificado resolviendo la ec. 3.7 mediante el método de las características, el cual reduce el problema a un sistema de ecuaciones diferenciales ordinarias (EDOs) [5].

Este método depende de que la observación $(x(t), t) : t \geq 0$ sea una curva suave, generando que a lo largo de ésta $u(x(t), t)$ tenga la siguiente tasa de cambio

$$\frac{d}{dt}u(x(t),t) = u_t + \frac{dx}{dt}u_x. \tag{3.9}$$

Comparando las ecuaciones 3.7 y 3.9 se tiene que $c = \frac{dx}{dt}$, por lo que ésta se interpreta como una velocidad. Ahora la ecuación diferencial parcial (EDP) puede ser reescrita como el siguiente sistema de EDOs

$$\frac{dx}{dt} = c, \tag{3.10}$$

$$\frac{du}{dt} = r. \tag{3.11}$$

Las condiciones iniciales para el sistema de EDOs son derivadas de la condición inicial $u(x,0) = w(x)$ del problema original.

Note que esta ecuación es autónoma solo si c y r son independientes de t. Este sistema de EDOs se llama la ecuación característica. De la teoría de las EDOs sabemos que el sistema tiene una solución única $(x(t), u(t))$ al menos localmente en el tiempo t para cada x_0. El parámetro x_0 especifica la curva en el plano $x - t$, esto es $C(x_0) : x = \hat{x}(t, x_0)$, la cual recibe el nombre de característica (Fig. 3.1).

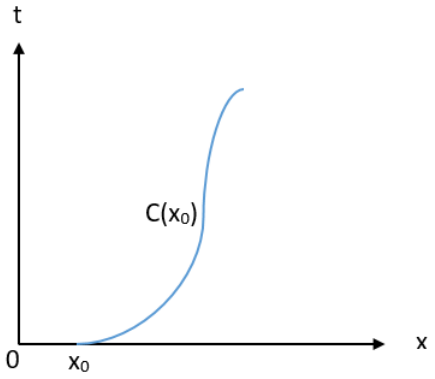

Figura 3.1: Característica $C(x_0)$ en el plano $x - t$.

Mientras que las curvas con diferentes valores de x_0 no se cruzen, el sistema tendra solución única. En cambio, si las curvas se cruzan en un cierto tiempo $t = t'$, el problema tendra multilples soluciones (Fig. 3.2), a esto se le llama que la solución rompe (break down) en $t = t'$ [5].

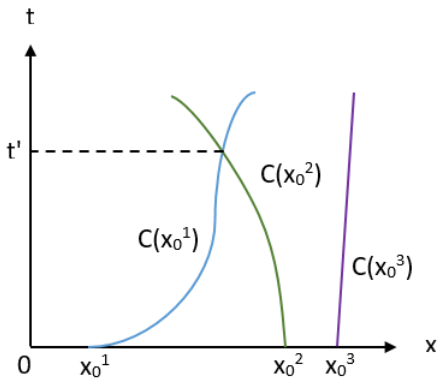

Figura 3.2: Ruptura de la solución en $t = t'$.

La solución de la ecuación de Burgers en el caso no viscoso (ec. 3.6) generada por el método de características típicamente rompe en tiempo finito. La ruptura de la solución señala la formación de una onda de choque, a través de la cual la solución es discontinua. Este comportamiento ocurre en dinámica de gases, combustión y detonación y elasticidad no lineal.

Si todas las ondas de choques fueran permitidas como solución débil, se tendrían múltiples soluciones. Para corregir esta falla, se introduce la noción de admisibilidad de choques.

Considere para la ec. 3.6 una solución $u(x,t)$ con un salto discontinuo en una curva $x = \gamma(y)$ con γ una función C^1. Se asume que $u(x,t)$ es C^1 y satisface la EDP excepto en $x = \gamma(t)$. Desde que u tiene un salto, este tiene límites unilaterales $u_{\pm}(t) = lim_{x \to \gamma_{\pm}(t)} u(x,t) = u(\gamma_{\pm}(t), t)$.

Considere los puntos $x = a$ y $x = b$ con $a < \gamma(t) < b$ para un t en un intervalo, como se muestra en la Fig. 3.3. La ley de balance (conservación) expresa la EDP en la forma

$$\frac{d}{dt}\int_a^b u(x,t)dx = \frac{u^2}{2}(a,t) - \frac{u^2}{2}(b,t). \tag{3.12}$$

Para evaluar el lado izquierdo de 3.12, separamos la integral y desarrollamos

$$\begin{gathered}\tfrac{d}{dt}\int_a^b u(x,t)dx = \tfrac{d}{dt}\int_a^{\gamma(t)} u(x,t)dx + \tfrac{d}{dt}\int_{\gamma(t)}^b u(x,t)dx \\ = u(\gamma(t)_-,t)\tfrac{d\gamma(t)}{dt} + \int_a^{\gamma(t)} \tfrac{\partial u(x,t)}{\partial t}dx - u(\gamma(t)_+,t)\tfrac{d\gamma(t)}{dt} + \int_{\gamma(t)}^b \tfrac{\partial u(x,t)}{\partial t}dx \\ = \tfrac{u^2}{2}(a,t) - \tfrac{u^2}{2}(b,t).\end{gathered}$$

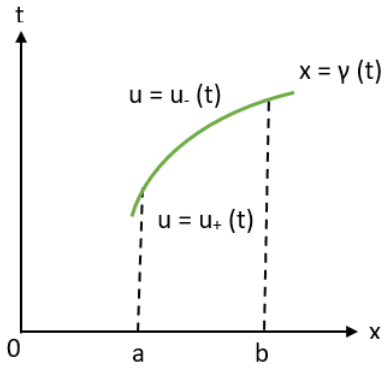

Figura 3.3: Onda de choque en $x = \gamma(t)$.

Usando la EDP en las integrales:

$$\int u_t = -\int \tfrac{u^2}{2} dx = -\tfrac{u^2}{2}.$$

El cálculo ahora se reduce a

$$\frac{d\gamma(t)}{dt}(u_+(t) - u_-(t)) = \frac{u_+^2(t)}{2} - \frac{u_-^2(t)}{2}. \tag{3.13}$$

Dado que $x = \gamma(t)$ es una discontinuidad para u, tenemos que $u_+(t) \neq u_-(t)$ y

$$\frac{d\gamma(t)}{dt} = \frac{u_+(t) + u_-(t)}{2}, \tag{3.14}$$

la cual es la condición de Rankine-Hugonist. El lado izquierdo es la velocidad de la discontinuidad o la onda de choque. La condición expresa que la velocidad de choque $\gamma(t)$ es el promedio de u a través del choque, en otras palabras, $\gamma(t)$ es el promedio de las velocidades características en cada lado del choque.

En muchas circunstancias se selecciona una única solución al imponer una condición adicional, además de la condición de Rankine-Hugonist, de que las características deben incidir en una discontinuidad de ambos lados. Esto es consistente con la causalidad, ya que las características llevan información acerca de la solución en el tiempo a partir de las condiciones iniciales y de contorno. Esta condición es conocida como la condición de entropía de Lax y aplica para soluciones discontinuas de cualquier ley de conservación escalar de la forma $\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} = 0$, y esta se puede generalizar

para discontinuidades en dimensiones más altas. La condición de entropía de Lax se puede escribir de la siguiente manera:

$$u_+(t) < \gamma'(t) < u_-(t) \tag{3.15}$$

La información a la izquierda y a la derecha de la curva $\gamma(t)$ determinan la localización del choque. Esta propiedad de localización del choque basado en la información de ambos lados de la discontinuidad nos dice el flujo de información de la onda. Desde que la información llevada por las características $C(x_0)$ varia continuamente con los datos iniciales, la condición de Lax asegura que la solución cambia continuamente con los datos, en este sentido, los choques que satisfacen la condición de entropía de Lax se les llama estables.

Capítulo 4

Método de solución

Tomando en cuenta las características de la ecuación diferencial a resolver, se seleccionó el método Runge-Kutta Discontinuos Galerkin (RKDG) para elaborar el solver, ya que éste posee un alto orden en espacio y tiempo, converge a la solución de entropía y es altamente paralelizable.

En cuanto a la parte de optimización, se utilizó el método Nelder-Mead, debido a que es un método utilizado para resolver problemas no lineales y no necesita del cálculo del gradiente ni del hessiano para funcionar, como ocurre en el método de Newton.

4.1. RKDG

El método de Runge-Kutta con Garlekin Discontinuo (RKDG) se utiliza para resolver problemas no lineales dominados por convección, como lo son los sistemas hiperbólicos no lineales en forma de divergencia $u_t + \nabla \cdot f(u) = 0$. Debido a que este método es robusto y preciso, se utilizan en dinámica de fluidos computacional y está encontrando rápidamente su uso en una amplia variedad de aplicaciones.

La aproximación de las soluciones exactas de estos sistemas es particularmente difícil debido a la presencia de discontinuidades en éstas, incluso si la condición inicial $u_0(\vec{x})$ es suave, causando que se obtengan soluciones numéricas múltiples para éstos, por lo que una gran ventaja de este método es que se obtiene la solución de entropía, la cual es única. Esta solución se alcanza debido a la aplicación del Galerkin

Discontinuo en la parte espacial.

Además el RKDG combina una clase especial de discretización temporal de Runge-Kutta con una discretización espacial de elementos finitos por aproximaciones discontinuas, que incorpora las ideas de flujos numéricos. El método resultante es estable, preciso de alto orden y con un esquema altamente paralelizable que puede manejar fácilmente geometrías complicadas y condiciones de contorno.

4.1.1. Galerkin Discontinuo

Para obtener el método de Galerkin Discontinuo (GD), se trabaja con el modelo simple del problema de Cauchy para la ley escalar de conservación no lineal en 1D con condiciones de contorno periódicas:

$$u_t + f(u)_x = 0, \tag{4.1}$$

$$u(x,0) = u_0(x). \tag{4.2}$$

Se busca una solución aproximada discontinua u_h de modo que cuando se restrinja al elemento $I_j = (x_{j-\frac{1}{2}}, x_{j+\frac{1}{2}})$ en cualquier tiempo t (Fig. 4.1), ésta sea perteneciente al espacio dimensional finito local $\mathcal{U}(I_j)$, que es típicamente un espacio de polinomios de grado K.

Para determinar la solución aproximada para $t > 0$ primero se multiplica la ecuación 4.1 para la aproximación discontinua por la función de prueba v_h, donde $v_h \in \mathcal{U}(I_j)$, esto es:

$$(u_h(x,t))_t v_h(x) + (f(u_h(x,t)))_x v_h(x) = 0. \tag{4.3}$$

Si integramos la ecuación diferencial 4.3 en un segmento I_j, se obtiene

$$\int_{I_j} (u_h(x,t))_t v_h(x)dx + \int_{I_j} (f(u_h(x,t)))_x v_h(x)dx = 0. \tag{4.4}$$

Resolviendo la segunda integral de la ecuación 4.4 por integración por partes, llegamos a

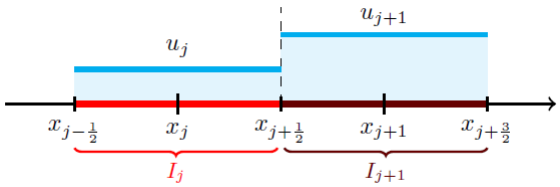

Figura 4.1: Elemento I_j o volumen de control.

$$\int_{I_j}(u_h(x,t))_t v_h(x)dx - \int_{I_j} f(u_h(x,t))(v_h(x))_x dx + \hat{f}(u_h(\cdot,t))v_h|_{x_{j-1/2}}^{x_{j+1/2}} = 0, \quad (4.5)$$

donde $\hat{f}(u_h)$ es el flujo numérico.

El flujo numérico es una aproximación del valor de $f(u_h)$ en el contorno del elemento I_j. Esto es ideado de tal forma que el método resultante sea siempre estable y converja a la solución exacta. Para esto el flujo debe depender solamente de los dos valores de la solución aproximada u_h en las fronteras del elemento,

$$\hat{f}(u_h)(x_{j+1/2}) = \hat{f}(u_h(x_{j+1/2}^-), u_h(x_{j+1/2}^+)), \quad (4.6)$$

siendo $u_h(x_{j+1/2}^-)$ el valor de u_h por el lado izquierdo y $u_h(x_{j+1/2}^+)$ el valor por el lado derecho del contorno del elemento. Los principales ejemplos de flujos numéricos utilizados en este método son las siguientes:

a) Flujo de Gudunov

$$\hat{f}^G(a,b) = \begin{cases} min_{a\le u\le b} & f(u) & si\ a \le b, \\ max_{b\le u\le a} & f(u) & en\ otro\ caso. \end{cases} \quad (4.7)$$

b) Flujo Engquist-Osher

$$\hat{f}^{EO}(a,b) = \int_0^b min(f'(s),0)ds + \int_0^a max(f'(s),0)ds + f(0). \quad (4.8)$$

c) Flujo de Lax-Friedrichs

$$\hat{f}^{LF}(a,b) = \frac{1}{2}[f(a) + f(b) - c(b-a)], \quad (4.9)$$

$$c = max|f'(s)|,$$
$$inf\ u^0(x) \leq s \leq sup\ u^0(x).$$

Para problemas no lineales, una buena elección de flujo numérico es el flujo Gudunov, ya que es bien sabido que este es el flujo numérico que produce la menor cantidad de viscosidad artificial. Sin embargo el flujo Enquist-Osher es el que presenta un mejor comportamiento al aplicarlo en la ecuación de Burgers, ya que captura correctamente la dinámica asintótica [7] y a pesar de que el Lax-Friedrichs presenta más viscosidad en este tipo de problemas, esto puede evitarse elevando el orden del método, ya que la experiencia numérica sugiere que a medida que aumenta el grado K de la solución aproximada, disminuye el impacto de la elección del flujo numérico sobre la calidad de las aproximaciones.

Los flujos numéricos también son necesarios cuando se trabaja con problemas de control óptimo en largos periodos de tiempo, ya que como se señala en [7], los esquemas numéricos que agregan demasiada viscosidad en los sistemas hiperbólico pueden conducir a soluciones con un comportamiento asintótico incorrecto.

En el caso lineal $f(u) = cu$ todos los flujos numéricos anteriores coinciden con el denominado flujo numérico up-wind

$$\hat{f}(a,b) = \begin{cases} ca & si \quad c \geq 0, \\ cb & si \quad c < 0. \end{cases} \tag{4.10}$$

Regresando al desarrollo del GD, para $x \in I_j$ expresamos nuestra solución aproximada u_h como la combinación lineal

$$u_h(x,t) = \sum_{l=0}^{k} u_j^l \Psi_l^j(x), \tag{4.11}$$

donde Ψ_l^j son las funciones base

$$\Psi_l^j(x) = P_l(\frac{2(x - x_j)}{x_{j+1/2} - x_{j-1/2}}), \tag{4.12}$$

y P_l son los polinomios de Legendre de grado l.

$$P_l(s) = \sum_{k=0}^{[l/2]} \frac{(-1)^k (2l-2k)!}{2^l k!(l-k)!(l-2k)!} s^{l-2k}. \tag{4.13}$$

El sistema de ecuaciones diferenciales ordinarias toma una forma particularmente simple, una matriz de masa diagonal, si elegimos a los polinomios $\Psi_l(x)$ tanto para las funciones base como para las funciones prueba, ya que los polinomios de Legendre cuentan con la siguiente propiedad de ortogonalidad:

$$\int_{-1}^{1} P_l(s) P_{l'}(s) ds = (\frac{2}{2l+1}) \delta_{ll'}. \tag{4.14}$$

Por lo tanto, la formulación débil toma la siguiente forma:

$$\frac{d}{dt} u_j^l(t) = \frac{2l+1}{\Delta j} \left(\int_{I_j} f(u_h(x,t))(v_h(x))_x dx - \hat{f}(u_h(\cdot,t)) v_h|_{x_{j-1/2}}^{x_{j+1/2}} \right) \tag{4.15}$$

para $l = 0, .., k$, donde u^m son los grados de libertad, $\Delta j = x_{j+1/2} - x_{j-1/2}$ y j el elemento identificador del elemento sobre el cual se está trabajando.

Por lo que podemos obtener sistemas de ecuaciones diferenciales ordinarias para los grados de libertad por elemento, el cual podemos escribir de la siguiente manera:

$$\frac{d}{dt} \vec{u}_h = L_h(\vec{u}_h). \tag{4.16}$$

Una forma razonable para obtener la condición inicial $u_h(x,0)$ en el elemento I_j es tomar la proyección L^2 de u_0 en el espacio de polinomios $\mathcal{U}(I_j)$, esto es:

$$\int_{I_j} u_h(x,0) v_h(x) dx = \int_{I_j} u_0(x) v_h(x) dx \tag{4.17}$$

Aplicando la propiedad de ortognalidad de los polinmios de Legendre la condición inicial se convierte en:

$$u_j^l(0) = \frac{2l+1}{x_{j+1/2} - x_{j-1/2}} \int_{I_j} u_0(x) \Psi_l^j(x) dx. \tag{4.18}$$

Al igual que todos los métodos de elementos finitos, con el Galerkin Discontinuo es posible manejar geometrías complejas y condiciones de contorno muy fácilmente.

Además, dado que la aproximación es discontinua y que la matriz de masa que se obtiene es diagonal, el problema es fácilmente invertible y de precisión de alto orden.

Las discretizaciones GD podrían usarse para una amplia gama de ecuaciones para las cuales los métodos GD no habían sido diseñados originalmente como, por ejemplo, ecuaciones púramente elípticas [13] [14].

4.1.2. Runge-Kutta

Una vez obtenida la discretización espacial, se pasa a discretizar el sistema de ecuaciones diferenciales ordinarias resultante, ecuación 4.16, utilizando el siguiente método Runge-Kutta (RK) de orden superior

1. Se establece $\vec{u}_h^{(0)} = \vec{u}_h^n$
2. Para $i = 1, ..., \mathcal{K}$ el cálculo de funciones intermedias es

$$\vec{u}_h^i = \sum_{l=0}^{i-1} \alpha_{il} \vec{w}_h^{il} \tag{4.19}$$

$$\vec{w}_h^{il} = \vec{u}_h^{(l)} + \frac{\beta_{il}}{\alpha_{il}} \Delta t^n L_n(\vec{u}_h^{(l)}) \tag{4.20}$$

3. Se establece $\vec{u}_h^{n+1} = \vec{u}_h^{(\mathcal{K})}$

Donde $\vec{u}_h^n$ es el paso de tiempo actual, $\vec{u}_h^{n+1}$ es el siguiente paso de tiempo y las constantes α y β están determinadas por el orden del Runge-Kutta. A continuación en la tabla 4.1 se muestran algunos de estos valores.

orden	α_{il}	β_{il}
2	1	1
	$\frac{1}{2}$ $\frac{1}{2}$	0 $\frac{1}{2}$
3	1	1
	$\frac{3}{4}$ $\frac{1}{4}$	0 $\frac{1}{4}$
	$\frac{1}{3}$ 0 $\frac{2}{3}$	0 0 $\frac{2}{3}$

Cuadro 4.1: Parámetros de las funciones intermedias del Runge-Kutta.

La característica distintiva de este método RK es que su estabilidad viene de la estabilidad del mapeo $u_h^{(l)} \rightarrow w_h^{il}$ definido en los pasos intermedios. Entre otras características se encuentran:

- Captura de choques
- Disminución drástica de la disipación artificial con el aumento del orden de precisión.
- Omisión de los limitadores de pendientes para la estabilidad numérica en problemas lineales, incluso con soluciones discontinuas. Los resultados numéricos obtenidos por el método son oscilatorios pero las oscilaciones se reducen y se localizan cuando aumenta el orden del RK.
- Aproximación de soluciones complejas, ya que el método puede manejar soluciones con una estructura muy complicada [10].
- Posibilidad de utilizar contornos/fronteras curvos [11].
- Alta parelelización [12].

Esta clase elegante y simple de métodos RK fue identificada en 1988 por Shu [16] en el marco de métodos de diferencias finitas para leyes de conservación hiperbólicas no lineales y se denominó discretizaciones de tiempo RK de disminución de la variación total (Total Variation Diminishing RK time discretizations) porque se utilizó la variación total como la semi-norma.

Debido a la estructura de la discretización del espacio y del tiempo en el método RKDG, para actualizar los grados de libertad dentro de un elemento I_j, sólo la información de los elementos que comparten bordes con este es necesaria, originando que el método RKDG sea altamente paralelizable. Esta propiedad no la comparten otros métodos de alta resolución, los cuales en su paso de reconstrucción usan información de elementos alejados.

Cuando se utilizan polinomios de grado alto, se debe usar un método RK de alto orden que coincida con la precisión de la discretización del espacio, lo que hace que el método resultante sea de orden superior.

Obsérvese que el método RKDG está diseñado de tal manera que cuando se utilizan aproximaciones constantes a trozos para la discretización del espacio junto con el método de Euler directo para la discretización del tiempo, se obtiene un esquema de volumen finito estándar. Entonces se puede decir que el RKDG es una generalización de los métodos de volumen finito.

Es esencial para la estabilidad del método cumplir con la condición de Courant-Friedrichs-Lewy (CFL). Para el caso lineal la condición esta dada por

$$|c|\frac{\Delta t}{\Delta x} \leq CFL_{L^2}, \tag{4.21}$$

mientras que para el caso no lineal es la siguiente

$$\frac{\Delta t}{\Delta x} max_{j\in\mathcal{R}}|u_0^j| \leq CFL_{L^2}, \tag{4.22}$$

donde

$$CFL_{L^2} = \frac{1}{2K+1} \tag{4.23}$$

y K es el grado del polinomio utilizado para la función de prueba en Galerkin Discontinuo. La constante CFL_{L^2} también puede tomar los valores 1 ó $\frac{1}{2}$, siendo el caso de $\frac{1}{2}$ el utilizado cuando se trabaja con el flujo EO.

En el caso no lineal basta con escoger el valor máximo de $u_0(x)$ debido a que el valor de u tiende a disminuir conforme avanza el tiempo debido a la viscosidad o a la ruptura de la solución.

4.2. Optimizador

Para resolver el problema del control óptimo de la ecuación de Burgers no viscosa se usa una aproximación de mínimos cuadrados definiendo el siguiente funcional $J : \mathbb{R} \to \mathbb{R}$ como,

$$min_{u_0} J(u_0) = min_{u_0} \frac{1}{2} \int (u(x,T) - u^*(x))^2 dx, \tag{4.24}$$

sujeto a la ecuación de Burgers no viscosa (4.25) con la condición inicial (4.26)

$$\partial_t u(x,t) + \partial_x \left(\frac{u^2(x,t)}{2} \right) = 0 \quad x \in \mathbb{R},\ t > 0, \tag{4.25}$$

$$u(x,0) = u_0(x), \tag{4.26}$$

donde J es la función a minimizar, $u(x,T)$ es la solución de la ecuación de Burgers después de un tiempo T, u^* es la función conocida al tiempo T, también conocida en este caso como función objetivo, y u_0 es la condición inicial por determinar.

Para la discretización del funcional, se considera una regla de cuadratura simple

$$J_\Delta(u_\Delta^0) = \sum_j \frac{\Delta x_j}{2} (u(x_j,T) - u^*(x_j))^2 \tag{4.27}$$

Debido a la no linealidad cuadrática involucrada en la ecuación de Burgers, el funcional J no es convexo o al menos no hay evidencia de su convexidad, sin embargo se demuestra en [9] que el caso no viscoso del problema de control óptimo tiene al menos un mínimo siempre que los datos iniciales estén restringidos apropiadamente, esto es: Sea $\Omega \subset \mathbb{R}$ un intervalo acotado y asumiendo que $u^* \in L^2(\mathbb{R})$

$$u_{ad} = \{u_0 \in L^\infty(\mathbb{R}), supp(u_0) \subset \Omega\} \tag{4.28}$$

El algoritmo utilizado para la optimización es el Nelder-Mead, debido a que este es utilizado en funciones no lineales y no requiere de derivadas, además generalmente éste es robusto para problema con discontinuidades y para funciones con ruido.

4.2.1. Nelder-Mead

Este método mantiene en cada paso un simplejo no degenerado, al cual se le aplica una de las siguientes cuatro operaciones: reflexión, expansión, contracción y encogimiento [19]. Dado un conjunto de vértices, un simplejo es la envoltura convexa de los mismos.

Es necesario especificar cuatro parámetros para definir la diferentes operaciones que se realizan sobre un simplejo, estos son: coeficiente de reflexión (ρ), coeficiente de expansión (χ), el coeficiente de contracción (γ) y el coeficiente de encogimiento (σ). Los valores usados generalmente para estas variables son:

$$\rho = 1, \quad \chi = 2, \quad \gamma = \tfrac{1}{2}, \quad \sigma = \tfrac{1}{2}.$$

Cada iteración comienza con un simplejo, especificado por sus $n+1$ vértices ($\vec{x}_1$, $\vec{x}_2$, ... , $\vec{x}_{n+1}$) y el valor de la función a optimizar asociado a estos ($f(\vec{x}_1)$, $f(\vec{x}_2)$, ... , $f(\vec{x}_{n+1})$), los cuales se ordenan de mejor a peor, siendo el primero el menor valor de f y el último el valor más grande f.

Existen dos tipos diferentes de resultados que se pueden obtener después de una iteración, en el primer caso se tiene un nuevo vértice, el cual se obtuvo aceptando el remplazo de $\vec{x}_{n+1}$, en el segundo caso se tiene un nuevo conjunto de vértices, en el cual sólo se mantiene la mejor solución $\vec{x}_1$.

Los pasos que se realizan en una iteración del algoritmo Nelder-Mead son los siguientes:

1. **Orden**. Se ordena los $n+1$ vértices $f(\vec{x}_1) < f(\vec{x}_2) < \ldots \leq f(\vec{x}_{n+1})$
2. **Reflexión**. Se calcula la reflexión $\vec{x}_r$

$$\vec{x}_r = \bar{\vec{x}} + \rho(\bar{\vec{x}} - \vec{x}_{n+1}) = (1+\rho)\bar{\vec{x}} - \rho\vec{x}_{n+1},$$

donde $\bar{\vec{x}} = \sum_{i=1}^{n} \frac{\vec{x}_i}{n}$ es el centroide de los n mejores puntos. Después se evalúa $f_r = f(\vec{x}_r)$. Si $f_1 \leq f_r < f_{n+1}$ se acepta el cambio de $\vec{x}_r$ por $\vec{x}_{n+1}$.

3. **Expandir**. Si $f_r < f_1$, se calcula el punto de expansión $\vec{x}_e$

$$\vec{x}_e = \bar{\vec{x}} + \chi(\vec{x}_r - \bar{\vec{x}}) = \bar{\vec{x}} + \rho\chi(\bar{\vec{x}} - \vec{x}_{n+1}) = (1+\rho\chi)\bar{\vec{x}} - \rho\chi\vec{x}_{n+1}.$$

se evalúa $f_e = f(\vec{x}_e)$. Si $f_e < f_r$ se acepta el cambio de $\vec{x}_e$ por $\vec{x}_{n+1}$.

4. **Contraer**. Si $f_r \geq f_{n+1}$ se realiza una contracción entre $\vec{\bar{x}}$ y el mejor de $\vec{x}_{n+1}$ y $\vec{x}_r$.

a. **Outside**. Si $f_n \leq f_r < f_{n+1}$. Se calcula un outside contract x_c

$$\vec{x}_c = \vec{\bar{x}} + \gamma(\vec{x}_r - \vec{\bar{x}}) = \vec{\bar{x}} + \gamma\rho(\vec{\bar{x}} - \vec{x}_{n+1}).$$

se evalúa $f_c = f(\vec{x}_c)$. Si $f_c \geq f_r$ se acepta el cambio de $\vec{x}_c$ por $\vec{x}_{n+1}$.

b. **Inside**. Si $f_r \geq f_{n+1}$, se calcula

$$\vec{x}_{cc} = \vec{\bar{x}} + \gamma(\vec{\bar{x}} - \vec{x}_{n+1}) = (1-\gamma)\vec{\bar{x}} - \gamma\vec{x}_{n+1}.$$

se evalúa $f_{cc} = f(\vec{x}_{cc})$. Si $f_{cc} \geq f_{n+1}$ se acepta el cambio de $\vec{x}_{cc}$ por $\vec{x}_{n+1}$.

5. **Reducción** En caso de que no se haya aceptado ningún punto se evalúa f en los n puntos $\vec{v}_i = \vec{x}_i + \sigma(\vec{x}_i - \vec{x}_1)$ con $i = 2, 3, \ldots, n+1$, los cuales serán los vértices del simplejo en la siguiente iteracion, junto con x_1 (se conserva la mejor solución).

El algoritmo Nelder-Mead termina cuando se alcanza una condición de paro, entre las cuales se halla el tamaño del simplejo.

Capítulo 5

Algoritmo

El programa minimiza el funcional J, ec. 5.1, seleccionando la condición inicial u_0 más óptima para generar la función $u^*(x)$, después de un tiempo T. Es decir, en cada iteración del algoritmo de optimización se resuelve la ecuación de Burgers para un tiempo T para una condición inicial dada, después la solución obtenida $u(x,T)$ se comparan mediante mínimos cuadrados contra la función objetivo $u^*(x)$ mediante

$$J(u_0) = \frac{1}{2}\int_R (u(x,T) - u^*(x))^2 dx. \tag{5.1}$$

La implementación consta de tres archivos, dos archivos con extención .c (main.c y funciones.c) y un archivo con extension .h (funciones.h). El archivo funciones.c es el que contiene todas las funciones que se elaboraron, mientras que main.c es el archivo principal. La parte del solver fue elaborada desde cero, mientras que la parte de optimización se trabajó con la función gsl-multimin-fminimizer-nmsimplex2 de la librería gsl.

Las variables que se manejan para la parte del Galerkin Discontinuo son: el mallado x_{mall}, la discretización espacial Δx_{mall}, el número del elementos $elem_{mall}$, el número de nodos $nodos_{mall}$ y el grado de los polinomios de Legendre L a utilizar. Mientras que para la parte del Runge-Kutta se requiere la discretización espacial Δt y el número de pasos de tiempo $pasos$.

En el caso de nuestro algoritmo, debido a que se utilizan cuadraturas Gaussianas para realizar las integrales de 4.15, se requiere especificar el orden de éstas. Se escogió este método debido a que se tienen que realizar pocas evaluaciones para calcular la

integral, además de que al tener una malla fina se puede tomar el valor de $u_j^m(t)$ en 4.11 como constante en el espacio en cada uno de los elementos de la red al momento de analizar un paso de tiempo, por lo que los polinomios de Legendre son los únicos que se evalúan en los puntos de la cuadratura al realizar la integral.

También es necesario ingresar el número de hilos con los que se va a trabajar, ya que en el algoritmo un paso Runge-Kutta se realiza de forma paralela. Esto es debido a que para calcular el siguiente paso de tiempo el método sólo necesita la información del elemento que se desea actualizar y la de las fronteras de sus vecinos Fig. 5.1. Una vez calculados todos los elementos del mallado, se actualiza el paso de tiempo.

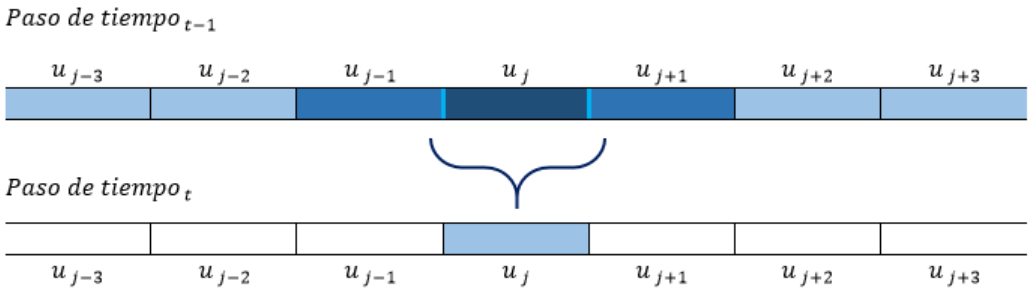

Figura 5.1: Paralelización de un paso de tiempo.

En el caso de la función inicial u_0, esta se parametriza con un conjunto de pares ordenados (u_i, x_i), los cuales luego se interpolan para generar $u_0(x)$ $\forall x$ dentro de la malla. Por lo tanto, las variables utilizadas para generarla son el número de pares ordenados n, las x_i y las u_i. En cuanto a la función objetivo, ésta se toma como un muestreo, por lo que las variables utilizadas son el mallado x_{mues}, el valor de función objetivo u^*_{mues} y el número de nodos $nodos_{mues}$.

Finalmente hay que especificar el número de iteraciones it a realizar, el tamaño de paso del simplejo ss y la tolerancia del tamaño mínimo tol de éste.

Para ejecutar el programa se necesitan cinco archivos ".dat"donde estará contenida toda la información necesaria para el programa, los cuales son; mallado.dat, datos.dat, uT.dat, parametros.dat y ss.dat. En el primer archivo se tiene toda la información del mallado (discretización espacial), en el segundo se tiene los datos que requiere el solver (grado del polinomio de Legendre, orden de la cuadratura y núme-

ro de hilos para paralelizar) y la discretización temporal para el Runge-Kutta, en el tercero se encuentran los datos de la función objetivo, en el siguiente se encuentran los parámetros para generar la condición inicial y en el último archivo se encuentra el tamaño de paso para el simplejo. El código nos da como resultado un archivo ".txt"donde se encuentra el valor de J, la tolerancia del simplejo y la información de todos los parámetros que generan la condición inicial por cada iteración.

Los pasos del algoritmo principal (optimizador) se muestran en el algoritmo 1. Se define $N_{1j} = 1 - \frac{x-x_j}{x_{j+1}-x_j}$, $N_{2j} = \frac{x-x_j}{x_{j+1}-x_j}$. La constante tol_J es un criterio de paro que se aplica sobre el valor de J, generalmente posee los valores de $0.01J_{inicial}$ ó $0.1J_{inicial}$.

Algorithm 1 Optimizador

1: Inicializar de las variables del solver
2: Leer los parámetros de u_0
3: Leer los datos de u^*
4: Inicializar los parámetros a optimizar $param$
5: Inicializar el entorno del solver
6: **while** $it \leq itmax$ **and** $tam > tol$ **and** $J > tol_J$ **do**
7: Actualizar los parámetros a optimizar
8: Ejecutar el solver y calcular J (Algoritmo 2)
9: Calcular el tamaño del simplejo tam
10: $it = it + 1$
11: **end while**
12: Reportar los resultados

Para trabajar con el algoritmo de optimización es necesario crear una función que resuelva la ecuación diferencial 4.15 a partir de la condición inicial y calcule el valor de J. Esta función recibe como variables entrada los parámetros para generar la condición inicial, la información del solver (mallado, orden del polinomio, grado de cuadratura, etc) y los datos de la función objetivo u^* y regresa como salida el valor del funcional J. Los pasos se muestran en el algoritmo 2. La revisión de los parámetros consiste en que verifica que se cumplan las condiciones impuestas en éstos, esto es, que los valores de las x_i vayan en orden creciente Fig. 5.2, además de que la distancia entre estos no sea menor que el valor máximo de la discretización de la malla $Max\ \Delta x_{mall}$.

Cabe mencionar que la aplicación de un paso de tiempo Runge-Kutta en los elementos del mallado se encuentra paralelizado mediante la arquitectura de openMP.

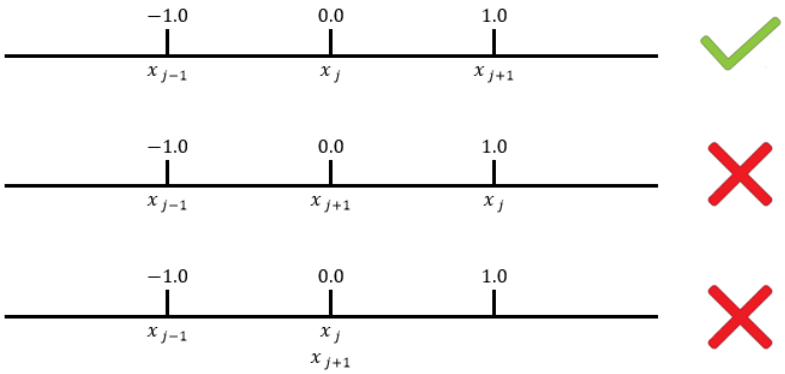

Figura 5.2: Revisión de los parámetros x_i de la condición inicial.

Se utilizó esta librería para paralelizar debido a su fácil implementación y al hecho de que se trabaja con memoria compartida, lo cual es de gran ventaja para el método RKDG, ya que al momento de calcular el nuevo paso de tiempo se utiliza la información actual de los elementos del mallado sin modificar ésta, la actualización se realiza una vez calculados todos los elementos.

Algorithm 2 Solver

```
Revisar y corregir los parámetros
u_0 = ⊕_{j=1}^{n-1} N_{1j} u_j + N_{2j} u_{j+1}
Inicializar el solver
for i=0 to T do
    for j=0 to Num. Elem. do
        Calcular un paso de tiempo por elemento espacial (Algoritmo 3)
    end for
    Actualizar de todo el paso de tiempo
end for
Interpolar u*
Calcular ec. 4.27
```

El algoritmo de un paso Runge-Kutta de orden 2 se muestra en el algoritmo 3. Esta función recibe como parámetros de entrada toda la información del mallado, el flujo numérico a utilizar, el paso de tiempo Δt actual y la información de los coeficientes $u_j^l(t)$, además de un apuntador a una matriz que nos servirá para ir guardando los resultados de cada elemento del mallado. La función no regresa ninguna valor.

Algorithm 3 Runge-Kutta

1: Inicializar las variables $\vec{u}_0$, $\vec{u}_1$, $\vec{u}_2$
2: Calcular $\vec{u}_0 = \vec{u}(x_j, t_p)_j$
3: Calcular $\vec{u}_1 = \vec{u}_0 + \Delta t_p * L_h(u_0)$ (Algoritmo 4)
4: Calcular $\vec{u}_2 = 0{,}5\vec{u}_0 + 0{,}5\vec{u}_1 + 0{,}5\Delta t_p L_h(\vec{u}_1)$ (Algoritmo 4)
5: Calcular $\vec{u}(x_j, t_{p+1})_j = \vec{u}_2$

Por último el algoritmo para la función L_h (ecuación 4.16) se muestra en el algoritmo 4.

Algorithm 4 Galerkin Discontinuo

1: Calcular la integral $\int_{I_j} f(u(x,t))(v(x))_x dx$
2: Calcular el flujo numérico en el límite izquierdo
3: Evaluar P_l en el límite izquierdo
4: Calcular el flujo numérico en el límite derecho
5: Evaluar P_l en el límite derecho
6: Calcular ec. 4.16

También se cuenta con un programa auxiliar donde solamente se ejecuta el solver y se realiza la evaluación del funcional, el cual se utiliza para generar los datos de las gráficas de la condición inicial y la solución de la EDP después de un tiempo T. Este programa recibe 4 archivos de entrada (mallado.dat, datos.dat, uT.dat y parametros.dat) y regresa como resultado el valor de J y 2 archivos de salida con los datos a gráficar (inicial.txt y solucion.txt). El algoritmo implementado se muestra en 5.

Todo el proceso de optimización se divide en dos fases. En la primera se optimiza $J(x_1, x_2, \ldots, x_m, u_1, u_2, \ldots, u_m)$, esto es que la variables que se optimizan son tanto las x_i como las u_i, evitando con esto tomar demasiados puntos para parametrizar u_0, ya que al optimizar las x_i se pueden encontrar los sitios donde están los saltos o los cambios de pendiente más pronunciados.

En la segunda fase se refina la malla duplicando los pares ordenados, esto es: $\{x_1, \ldots, x_m\} \subset \{y_1, \ldots, y_n\}$ y $\{u_1, \ldots, u_m\} \subset \{w_1, \ldots, w_n\}$ con $n = 2m - 1$, y se optimiza sobre $J(u_1, u_2, \ldots, u_n)$, esto se realiza para hacer más suave la función u_0 y disminuir aún mas el valor de J, esta segunda etapa puede repetirse varias veces. Este proceso se puede observar en la Fig. 5.3.

Algorithm 5 Datos a gráficar

1: Inicialización de las variables del solver
2: Muestreo de los datos de u_0
3: Muestreo de los datos de u^*
4: $u_0 = \oplus_{j=1}^{n-1} N_{1j} u_j + N_{2j} u_{j+1}$
5: Impresión de $u_0(x)$
6: Inicialización del solver
7: **for** i=0 **to** T **do**
8: **for** j=0 **to** Num. Elem. **do**
9: Cálculo de un paso Runge-Kutta en un elemento espacial (Algoritmo 3)
10: **end for**
11: Actualización de todo el paso de tiempo
12: **end for**
13: Impresión de $u(x, T)$
14: Interpolación de u^*
15: Cálculo de ec. 4.27
16: Reportar valor de J

El solver utilizado en el presente trabajo, utiliza condiciones de frontera periódicas, por lo que hay que tener en cuenta algunos detalles al momentos de ingresar los parámetros, estos son, que el valor final y el inicial del mallado deben ser iguales para conservar la continuidad (a menos que se desee una condición inicial discontinua en los extremos). Tomando en cuenta la continuidad, el último parámetro no se incluye para la optmización, ya que al ser igual que el primer parámetro, al momento de parametrizar éste, se parametrizan los dos al mismo tiempo. Otros de los parámetros que no se incluyen en la optimización son las fronteras del mallado, ya que se tiene que tener bien delimitado el espacio de trabajo.

Por lo tanto flujo del proceso de optimización es el siguiente:

Se comienza leyendo la información del mallada, la discretización espacial, los datos de la función objetivo y los parámetros de u_0. Luego se toman como variables a optimizar tanto las x_i como las u_i para poder inicializar el entorno del optimizador. Una vez hecho esto, se comienza con las iteraciones del Nelder-Mead, donde en cada iteración se realiza el RKDG y posteriormente se evalúa J por cada vértice del simplejo que se analice. Después se evalúa el criterio de paro, que en este caso se utiliza el número de iteraciones, el tamaño del simplejo y el valor de J, no se utiliza la

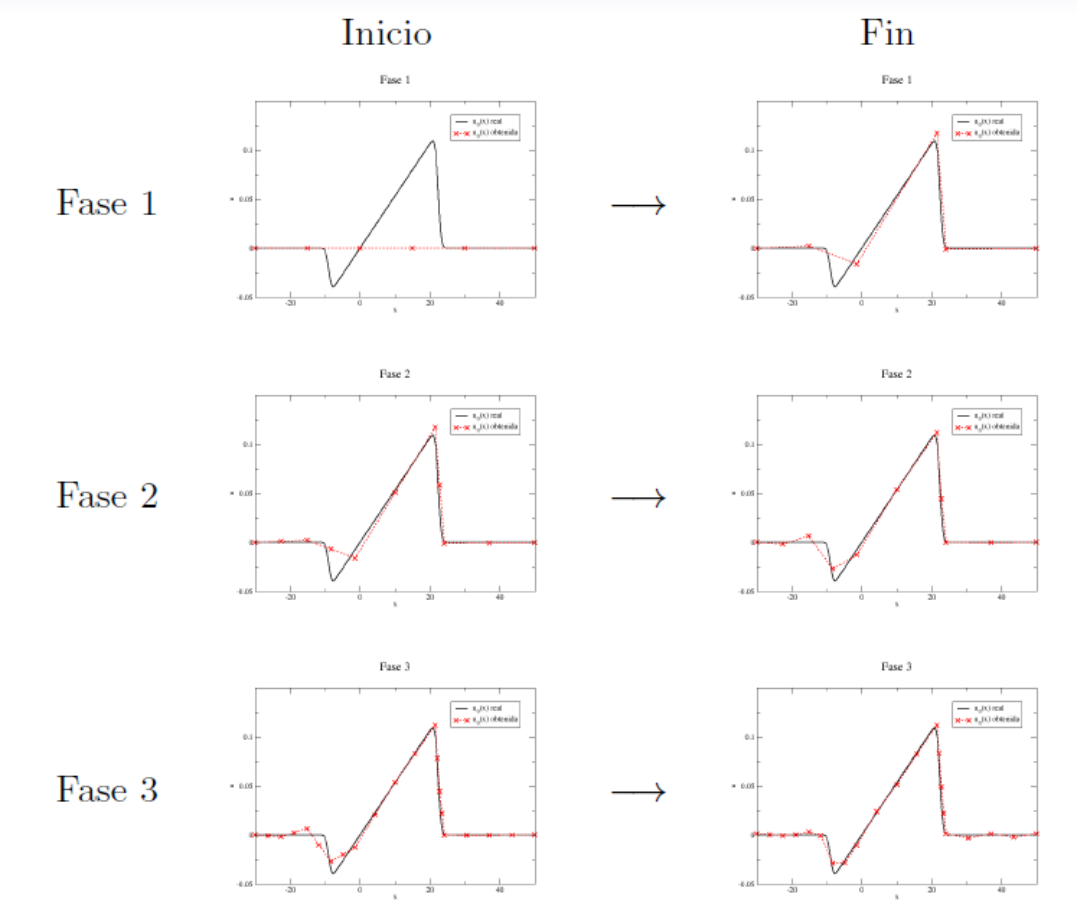

Figura 5.3: Ejemplo de 3 fases del proceso de optimización propuesto.

variación en J como criterio de paro ya que debido a las operaciones que se realizan en Nelder-Mead pueden pasar varias iteraciones donde el valor de J no cambie. Si no se cumple la condición de paro se realiza otra iteración, en el caso de cumplirse se reporta el valor de J y los parámetros correspondientes. Una vez terminada la fase uno, se pasa a duplicar los pares ordenados, y se inicia la fase dos, tomando ésta vez sólo como variables a optimizar los parámetros u_i. Se vuelve a inicializar el entorno del optimizador para poder trabajar con los nuevos parámetros y se vuelve a repetir el procedimiento de optimizacion, si el valor de J no ha descendido todavía lo suficiente, se puede implementar la fase dos tantas veces como sea necesario. El flujo del proceso puede observarse en la figura 5.4.

El programa puede trabajar con una malla no homogénea (diferentes valores para

Δx) y con diferentes pasos de tiempo (Δt), por lo que se puede hacer un refinamiento de algún lugar o tiempo que se quiera analizar con mas detalle.

Se utiliza el flujo numérico EO, debido a que en la literatura es el que reportan con el mejor comportamiento en los choques, y es el que tiene menos problemas con la viscosidad, la cual afecta en tiempos de evolución muy largos.

Para las interpolaciones de los datos de muestreo (u^*), se utilizaron interpoladores lineales debido a que la malla que se utiliza en el muestreo es fina (aunque no tan fina como la del solver), por lo tanto, el interpolador lineal logra recobrar en buena forma el comportamiento de la función objetivo.

También hay que tener en cuenta que existen funciones objetivos que no pueden ser alcanzadas, ya que no existen como soluciones a la ecuación de Burgers en el tiempo T. Para asegurar que las soluciones pueden ser alcanzas se requiere que éstas satisfagan la condición de Oleinki [1].

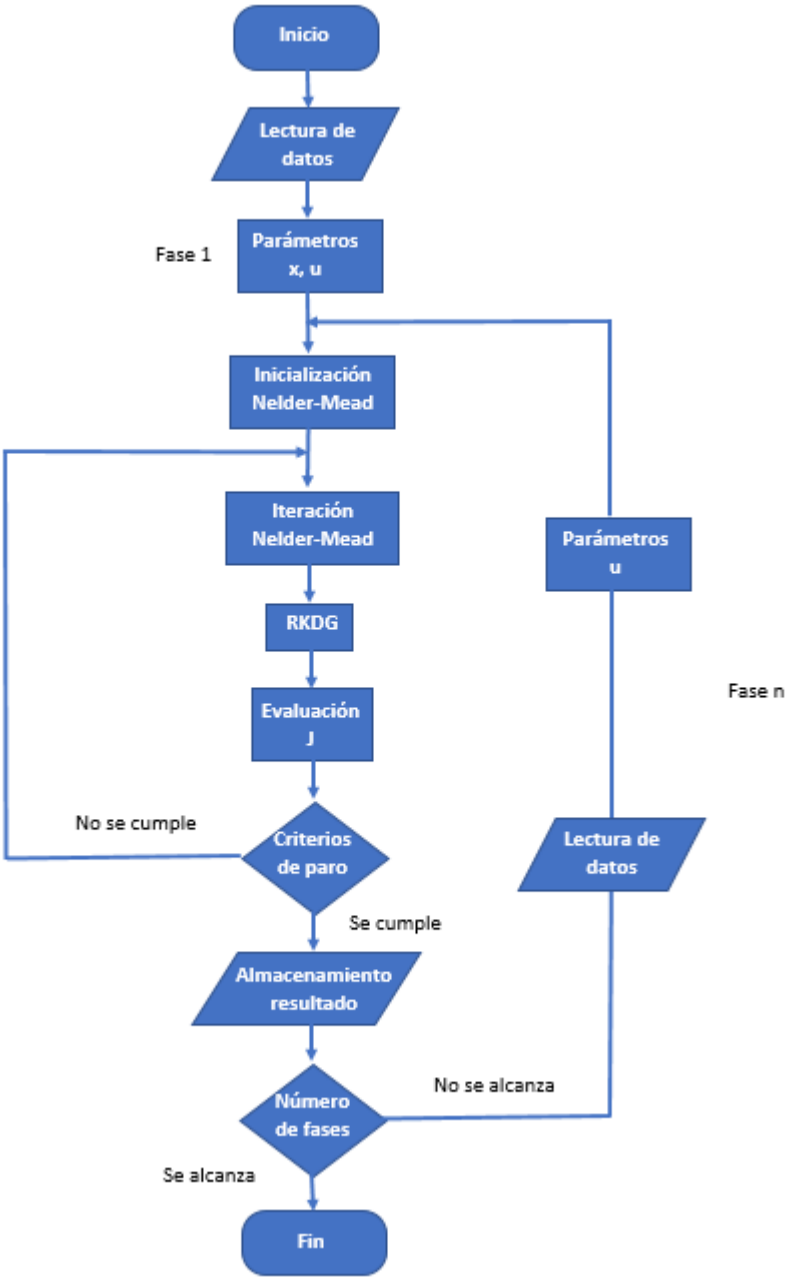

Figura 5.4: Flujo del proceso de optimización.

Capítulo 6

Resultados

En todas las pruebas realizadas se utilizaron condiciones de frontera periódicas y se aplicó el flujo numérico Engquist-Osher. La discretización espacial homogénea utilizada fue $\Delta x = 0.08$, mientras que la discretización temporal homogénea fue de $\Delta t = 0.001$, finalmente el orden de cuadratura empleado fue de $orden = 4$ y el grado de polinmios de Legendre fue $L = 2$. El programa se paralelizó con 4 hilos.

En todos los programas se trabajó con una malla dentro del intervalo $[-30, 50]$, por lo tanto se emplearon 1000 elementos espaciales (3000 coeficientes $u_j^m(t)$). En cuanto a la parte temporal se manejaron tiempos finales de $T = 10$ y $T = 50$, por lo que se tuvieron 10000 pasos de tiempo para $T = 10$ y 50000 para $T = 50$.

6.1. Solver

Para probar el funcionamiento del solver de la ecuación 4.1, tanto para el caso lineal como el no lineal (ecuación de Burgers no viscosa), se realizaron diversas pruebas con diferentes tipos de funciones iniciales, las cuales se dejaron evolucionar hasta un tiempo determinado.

En los experimentos realizados para el caso lineal se tomó $f(u) = 2u$, siendo 2 unidades la velocidad de desplazamiento de la onda, y la condición inicial u_0 se dejó evolucionar hasta un tiempo final de $T = 10$. Se realizaron 3 pruebas:

La primera prueba corresponde al análisis de una función suave, la cual no tuviera cambios bruscos de pendiente ni discontinuidades, eligiendo para esto la función seno

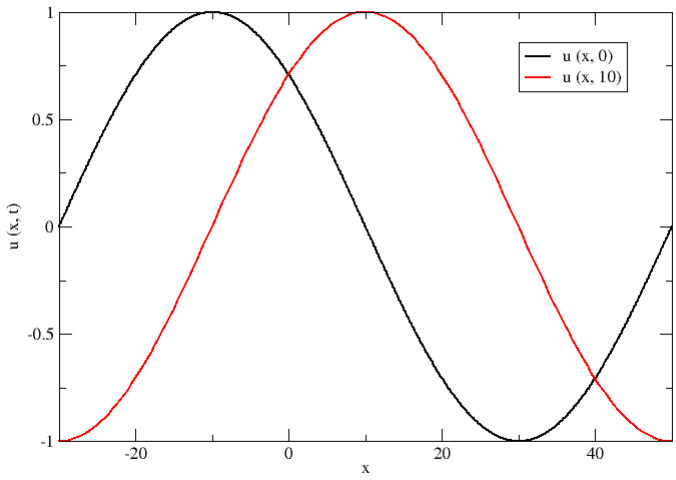

Figura 6.1: Resultado obtenido en $T = 10$ para el caso lineal, utilizando la ec. 6.1 como u_0.

(ec. 6.1), los resultados obtenidos por el solver se pueden observar en la figura 6.1.

$$u_0(x) = \sin(\frac{\pi}{40}(x + 30)). \tag{6.1}$$

Para analizar el comportamiento del solver con funciones con cambios de pendientes pronunciados, se utilizó la función descrita en la ecuación 6.2, los resultados de este experimento se haya en la figura 6.2.

$$u_0(x) = \begin{cases} \frac{3}{2000}(-e^{-(5\sqrt{20}+x)^2} + e^{-(2\sqrt{20}+x)^2} \\ +\pi x(erf(5\sqrt{20} - x) + erf(2\sqrt{20} - x))) & |x - 5| \leq 25, \\ 0 & \textit{en otro caso.} \end{cases} \tag{6.2}$$

Finalmente se probó el desempeño del solver con una función a trozos, la cual es el caso más difícil de tratar debido a la aparición de oscilaciones y picos en la región de la discontinuidad. Se utilizó una función escalón con varios niveles contenidos entre $(-1, 1)$, cuya expresión se encuentra en la ecuación 6.3, los resultados obtenidos se en este caso se observan en la figura 6.3.

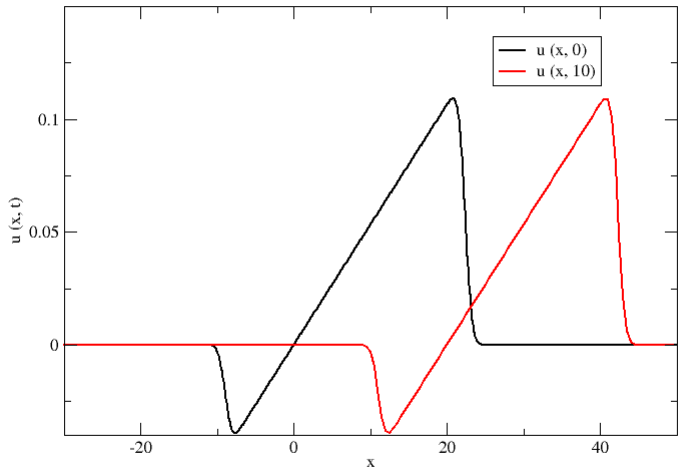

Figura 6.2: Resultado obtenido en $T = 10$ para el caso lineal, utilizando la ec. 6.2 como u_0.

$$u_0(x) = \begin{cases} 1 & -20 < x < 0, \\ -1 & 20 < x < 40, \\ 0 & \textit{en otro caso.} \end{cases} \tag{6.3}$$

Como se puede observar en las gráficas, los resultados fueron los esperados, ya que se obtuvo un desplazamiento hacia la derecha de 20 unidades, lo cual corresponde a la velocidad y tiempo utilizados en la expresión 6.4.

$$x = vT = (2)(10) = 20. \tag{6.4}$$

Para probar el caso correspondiente a la ecuación de Burgers no viscosa, se realizaron dos pruebas, la primera utilizando como u_0 a la función N (ec. 6.2), la cual fue dejada evolucionar hasta un tiempo final de $T = 50$ y la segunda prueba con la función seno como condición inicial (ec. 6.1), la cual se dejó evolucionar hasta $T = 10$, los resultados se encuentran en la figura 6.4 y la figura 6.5 respectivamente.

Los resultados mostrados en las gráficas también corresponden con el comportamiento esperado, ya que la dirección de desplazamiento de la onda es el correcto,

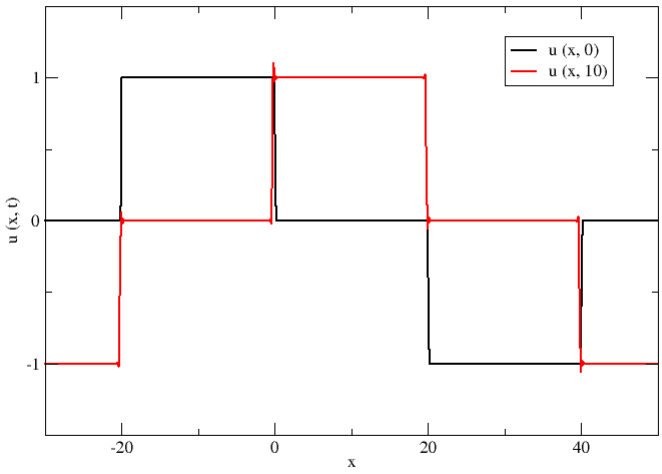

Figura 6.3: Resultado obtenido en $T = 10$ para el caso lineal, utilizando la ec. 6.3 como u_0.

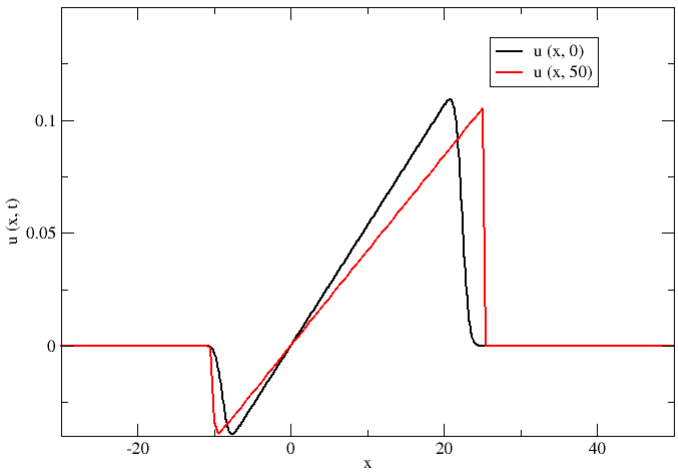

Figura 6.4: Resultado obtenido en $T = 50$ para la ecuación de Burgers no viscosa, utilizando la ec. 6.2 como u_0.

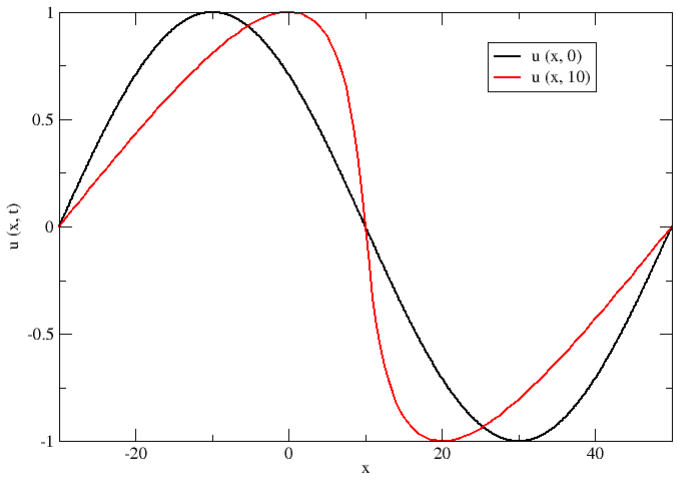

Figura 6.5: Resultado obtenido en $T = 10$ para la ecuación de Burgers no viscosa, utilizando la ec. 6.1 como u_0.

debido a que este corresponde con la velocidad de la onda, la cual es el valor que $u(x, t)$ tenga en ese momento y posición.

También se probó la eficiencia de la paralelización ejecutando el algoritmo del solver con diferente número de hilos y tiempos finales T. El speedup promedio del solver, evolucionando la condición inicial hasta $T = 10$ se muestran en la tabla 6.1, mientras que los resultados para $T = 50$ se encuentran en la tabla 6.2.

Hilos	Tiempo (s)	Speedup
1	39.697	1
2	21.34	1.86
4	11.715	3.39
8	6.411	6.19
12	4.585	8.66

Cuadro 6.1: Speedup promedio del solver, evolucionando la condición inicial hasta $T = 10$.

Hilos	Tiempo (s)	Speedup
1	196.479	1
2	103.638	1.90
4	55.511	3.54
8	29.982	6.55
12	21.553	9.12

Cuadro 6.2: Speedup promedio del solver, evolucionando la condición inicial hasta $T = 50$.

6.2. Optimizador

Para probar el funcionamiento del programa final (optimizador y solver) se realizaron pruebas sintéticas, al igual que en las pruebas realizadas para el solver se implementó tanto el caso lineal como el no lineal. Las funciones objetivo se generaron dejando evolucionar una condición inicial dada formada por 1000 pares ordenados. Las especificaciones usadas por el solver para obtener una función objetivo fueron las siguientes: condiciones de frontera periódicas, flujo numérico Engquist-Osher, discretización espacial homogénea $\Delta x = 0.08$, discretización temporal homogénea $\Delta t = 0.0001$, orden de cuadratura $orden = 5$, grado de polinmios de Legendre $L = 3$ y Runge-Kutta orden 2. Por lo tanto el nivel de resolución usado para generar la función objetivos es bastante alto, y es diferente al implementado para resolver el problema inverso.

Para generar las condiciones iniciales, en todos los casos se comenzó con un total de 6 pares ordenados, donde los valores de x_i están distribuidos casi de manera uniforme sobre la región a integrar y los valores de $u_i = 0$.

Cabe mencionar que la discretización de la función objetivo es diferente a la malla del solver, siendo esta de $\Delta x = 0.5$, sin embargo al calcular el funcional J (ec. 4.27), se utiliza la discretización del mallado del solver, por lo que es necesario realizar una interpolación sobre los datos de la función objetivo para hacer el cálculo de J.

Primero se probó el caso lineal utilizando como función objetivo el resultado obtenido de la evolución de la condición inicial seno (ec. 6.1) hasta $T = 10$. Se realizaron 2 fases, en las cuales el tamaño de paso de todas las variables a optimizar utilizadas fue de 1.0 en ambas fases.

En la tabla 6.3 se muestran los resultados obtenidos en esta prueba, donde *It. Máx* son las iteraciones máximas que se pueden hacer por fase, *Tol.* J_{ini} es la tolerancia que se tiene en cada fase tomando en cuenta el valor con J inicio ésta, *Tam. Simp.* es la tolerancia que se tiene sobre el tamaño del simplejo, J_{ini} y J_{fin} son el valor de J al iniciar y terminar la fase y finalmente *It.* es el número de iteraciones que se realizaron por fase.

	It. Máx.	Tol. J_{ini}	Tam. Simp.	J_{ini}	J_{fin}	It.
Fase 1	150	$0.01J$	0.05	19.9948	0.1638	40
Fase 2	150	$0.01J$	0.05	0.1638	0.0056	150

Cuadro 6.3: Resultados obtenidos en la primera prueba sintética.

Las gráficas obtenidos para $u(x,0)$ y $u(x,T)$ en la dos fases pueden observarse en la figura 6.6, donde $u(x,0)$ *real* es nuestra condición inicial u_0 y $u(x,10)$ *real* es nuestra función objetivo u^*, $u(x,0)$ *inicial* y $u(x,10)$ *inicial* corresponden al punto inicial del optimizador y $u(x,0)$ *fase* 1, $u(x,10)$ *fase* 1, $u(x,0)$ *fase* 2 y $u(x,10)$ *fase* 2, son los resultados que se obtuvieron por cada fase de optimizador, siendo las dos últimos el resultado obtenido de todo el proceso de optimización, los parámetros se refieren a los pares ordenados que generan $u(x,0)$ en cada fase.

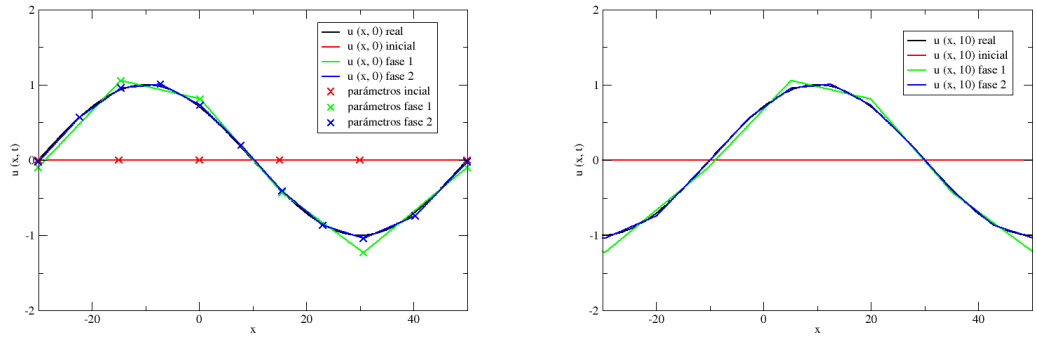

Figura 6.6: Resultados por cada fase del proceso de optimización correspondiente al caso lineal, utilizando como función objetivo el resultado de la evolución la condición incial ec. 6.1 hasta $T = 10$.

La gráfica del resultado final del optimizador, ya mostrando la condición inicial

original contra la condición inicial encontrada y la comparación de su evolución al tiempo T contra la función objetivo, se muestra en la figura 6.7.

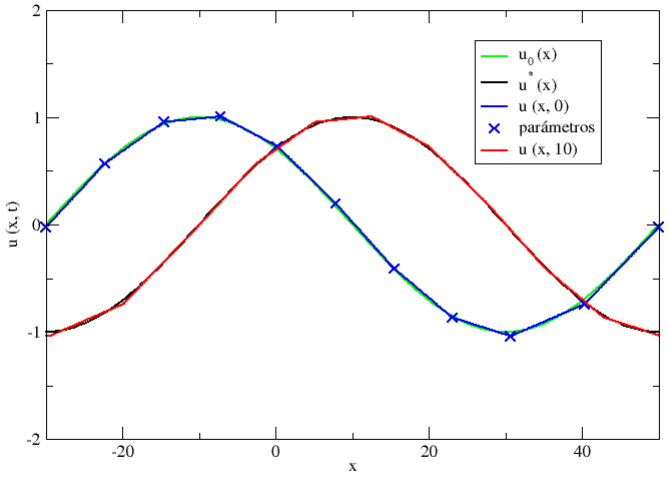

Figura 6.7: Resultado final correspondiente al caso lineal con la condición incial ec. 6.1 dejada evolucionar hasta $T = 10$.

Después se probó el caso no lineal, utilizando como función objetivo el resultado de la ecuación de Burgers no viscosa en $T = 10$ utilizando como condición inicial la ecuación 6.1, los resultados de esta prueba se pueden observar en la tabla 6.4.

	It. Máx.	Tol. J_{ini}	Tam. Simp.	J_{ini}	J_{fin}	It.
Fase 1	150	$0.01J$	0.05	19.9898	0.1947	38
Fase 2	150	$0.01J$	0.05	0.1947	0.0062	150

Cuadro 6.4: Resultados obtenidos en la segunda prueba sintética.

Del mismo modo que el caso anterior se realizaron dos fases, en las cuales se utilizó el mismo tamaño de paso para todas variables. Los gráficas para $u(x, 0)$ y $u(x, T)$ en la dos fases pueden observarse en la figura 6.8, mientras que la gráfica del resultado final se muestra en la figura 6.9.

Finalmente se realizó una prueba tomando como función objetivo el resultado de

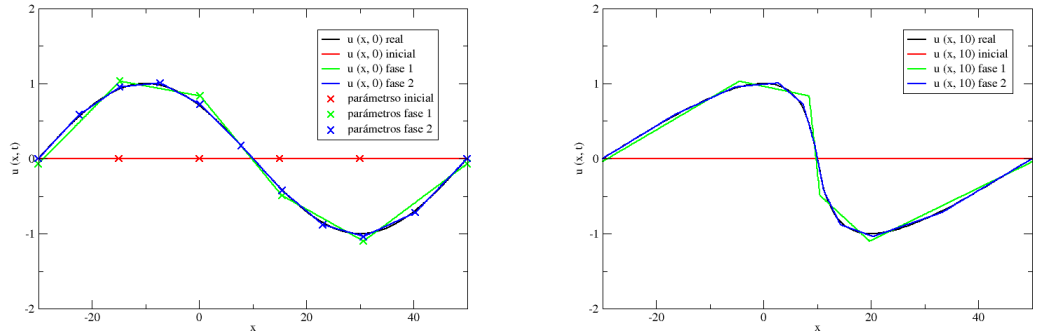

Figura 6.8: Resultados por cada fase del proceso de optimización correspondiente a la ecuación de Burgers no viscosa, utilizando como función objetivo el resultado de la evolución la condición incial ec. 6.1 hasta $T = 10$.

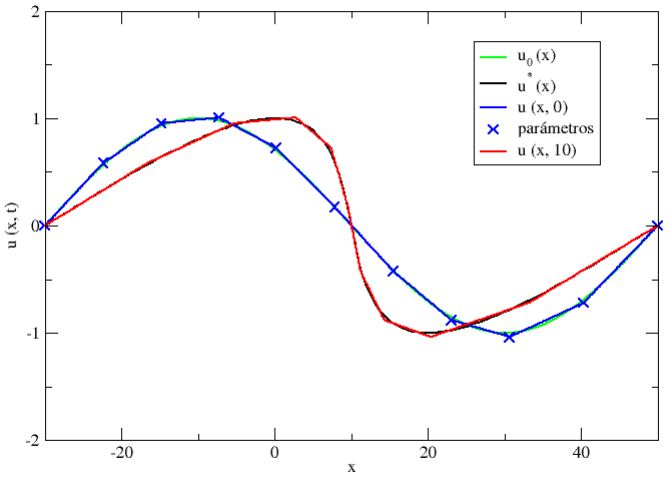

Figura 6.9: Resultado final correspondiente a la ecuación de Burgers no viscosa con la condición incial ec. 6.1 dejada evolucionar hasta $T = 10$.

la ecuación lineal en $T = 10$ utilizando como condición inicial la ecuación 6.2. Los resultados de esta prueba se pueden observar en la tabla 6.5.

	It. Máx.	Tol. J_{ini}	Tam. Simp.	J_{ini}	J_{fin}	It.
Fase 1	150	$0.01J$	0.05	0.0528	0.0017	150
Fase 2	150	$0.1J$	0.05	0.0017	0.0007	150
Fase 3	50	$0.1J$	0.05	0.0007	0.0002	50

Cuadro 6.5: Resultados obtenidos en la tercera prueba sintética.

En esta ocasión se realizaron 3 fases, debido a que fue más difícil ajustar los datos. En la primera fase, se usó un tamaño de paso de 1.0 para los parámetros x_i y de 0.001 para los u_i, debido a la diferencia de magnitud entre estos, en la segunda fase se usó un tamaño de paso de 1.0 para las variables u_i y en la tercera fase se usó un tamaño de 0.01 para éstas, debido a que el la cambio en las variables u_i ya no es tan grande como en las fase anterior.

La primera y tercera fase terminaron por que se alcanzo el número máximo de iteraciones permitidas, mientras que la segundar paró por el tamaño del simplejo. Las gráficas obtenidos para $u(x, 0)$ y $u(x, T)$ en la tres fases pueden observarse en la figura 6.10, mientras que la gráfica del resultado final se muestra en la figura 6.11.

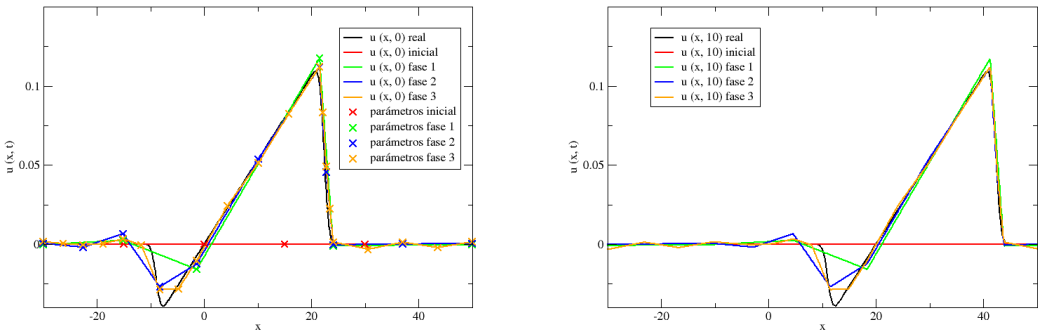

Figura 6.10: Resultados por cada fase del proceso de optimización correspondiente al caso lineal, utilizando como función objetivo el resultado de la evolución la condición incial ec. 6.2 hasta $T = 10$.

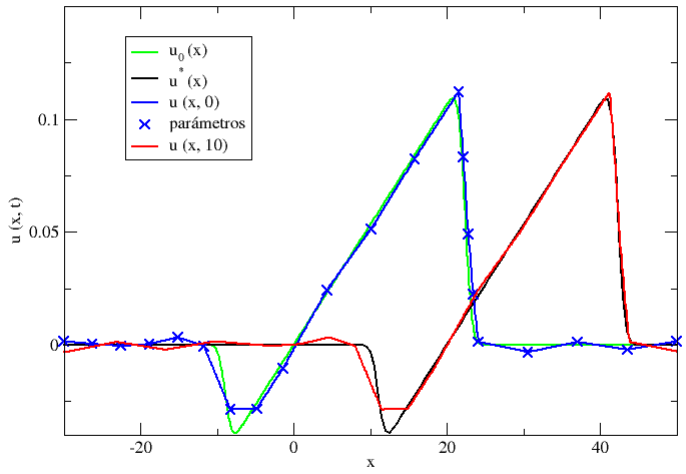

Figura 6.11: Resultado final correspondiente al caso lineal con la condición incial ec. 6.2 dejada evolucionar hasta $T = 10$.

Tanto para los casos lineales, como para el caso no lineal se comprobó que se cumpliera la condición CFL, obteniendo para los casos lineales

$$\Delta t \leq \frac{(0.2)(0.08)}{2.0} = 0.008, \tag{6.5}$$

mientras que para el caso no lineal

$$\Delta t \leq \frac{(0.2)(0.08)}{1.0} = 0.016, \tag{6.6}$$

por lo que en todos los casos la discretización temporal $\Delta t = 0.001$ cumple con la condición CFL.

6.3. Problema inverso

Para asegurar que el correcto funcionamiento del algoritmo no se debe a que se trabajaron con pruebas sintéticas, se probó el programa con la siguiente función u^* (ec. 6.7) utilizando la dinámica de Burgers no viscosa, éste es el mismo ejemplo

utilizado en [1], del cual no conocemos la expresión de la condición inicial.

$$u^*(x) = \begin{cases} \frac{3}{2000}(-e^{-(5\sqrt{20}+x)^2} + e^{-(2\sqrt{20}+x)^2} \\ +\pi x(erf(5\sqrt{20}-x) + erf(2\sqrt{20}-x))) & |x-5| \leq 25, \\ 0 & \textit{en otro caso.} \end{cases} \tag{6.7}$$

Se tomó como tiempo final $T = 50$, ya que es parecido al tiempo que toma al sonic boom propagarse por la atmósfera, el cual es un tiempo largo considerando la velocidad con la que se crea el ruptura.

Para resolver este problema inverso se utilizaron los mismos parámetros para el solver que en los casos anteriores. Se verificó que se cumple la condición CFL, en este caso $max_{j\in\mathcal{R}}|u_0^j| = 0.109$, por lo que se tomó como máximo el valor 0.15, para tener un poco más de tolerancia en caso de presentarse picos u oscilaciones en la evolución, por lo tanto, la condición de CFL para el tiempo queda como

$$\Delta t \leq \frac{(0.2)(0.08)}{0.15} = 0.106. \tag{6.8}$$

Por lo tanto, en nuestro caso se cumple con la condición CFL si se elige

$$\Delta t = 0.001 \leq 0.106. \tag{6.9}$$

Para resolver este el problema inverso se realizaron 3 fases. Para generar la condición inicial, se propusieron 6 pares de coordenadas en la primera fase cuyos x_i estaban distribuidos casi de manera uniforme dentro del mallado y los $u_i = 0$.

Los valores del tamaño de paso para la primera fase fueron de 1.0 para las variables x_i y de 0.01 para las u_i debido diferencia del orden de escala entre las x_i y las u_i, para la segunda fase se tomó el valor de 1.0 para el tamaño de paso, ya que solo se optimiza sobre las variables u_i, sin embargo en la fase 3 éste valor cambia a 0.01 debido a que como ya no hay tanto cambio en las variables u_i, el paso se tiene que hacer más fino. Los resultados obtenidos se pueden observar en la tabla 6.6.

Los gráficas para $u(x, 0)$ y $u(x, T)$ en la tres fases pueden observarse en la figura 6.12, mientras que la gráfica del resultado final se muestra en la figura 6.13.

La evolución que tiene el funcional J en todo el proceso de optimización y el

	It. Máx.	Tol. J_{ini}	Tam. Simp.	J_{ini}	J_{fin}	It.
Fase 1	250	$0.1J$	0.05	0.0528	0.0046	53
Fase 2	250	$0.1J$	0.05	0.0046	0.0012	214
Fase 3	250	$0.1J$	0.05	0.0012	0.0001	64

Cuadro 6.6: Resultados obtenidos para el problema inverso.

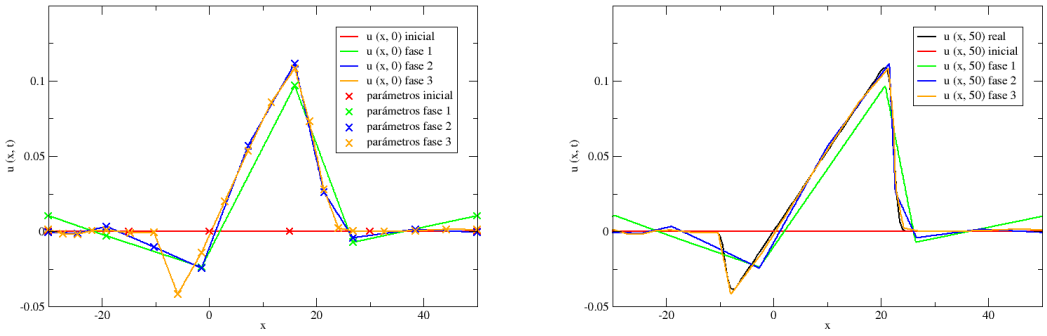

Figura 6.12: Resultados por cada fase del proceso de optimización correspondiente a la ecuación de Burgers no viscosa con la función objetivo u^* (ec. 6.7).

cambio en el tamaño del simplejo a través de éste pueden observarse por fases en la figura 6.14 y de forma completa y separada en las figuras 6.15 y 6.16.

Como se puede ver, el resultado obtenido está muy cerca de la función objetivo, por lo que el error obtenido es pequeño. Los resultados obtenidos por nuestro algoritmo son muy similares a los mostrados en la Fig. 10 del artículo [1], usando en nuestro caso, una cantidad de puntos menor para generar u_0, ya que en [1] la optimización se realiza sobre las u_i y se utiliza una discretización de muestreo para u_0 igual a la del mallado sobre la condición inicial.

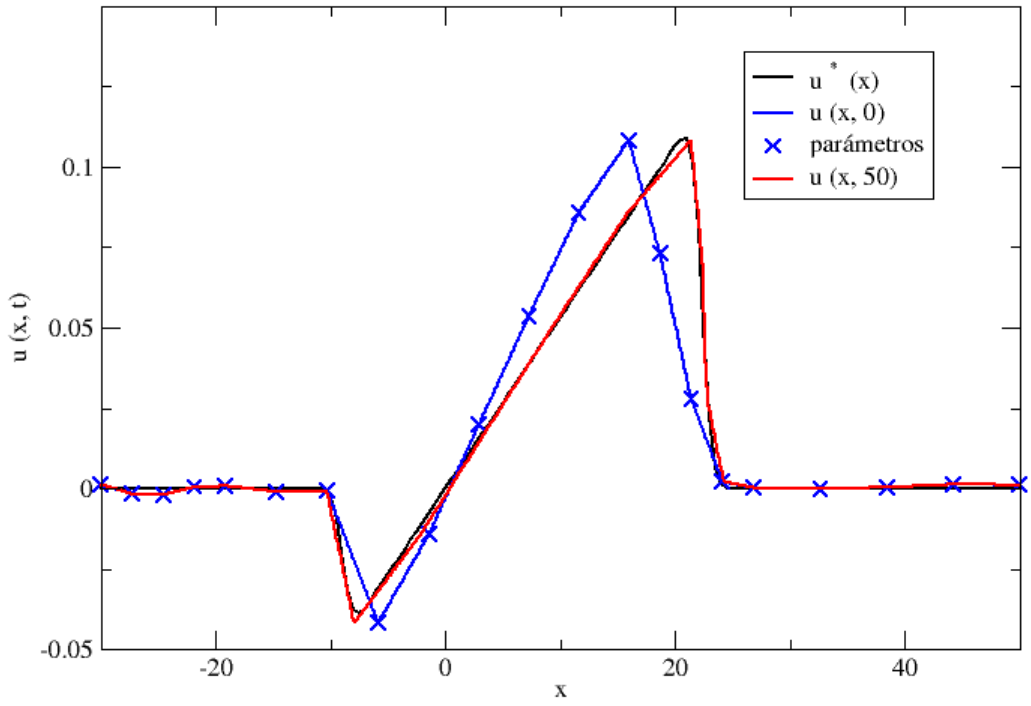

Figura 6.13: Resultado final correspondiente a la ecuación de Burgers no viscosa con la función objetivo u^* (ec. 6.7).

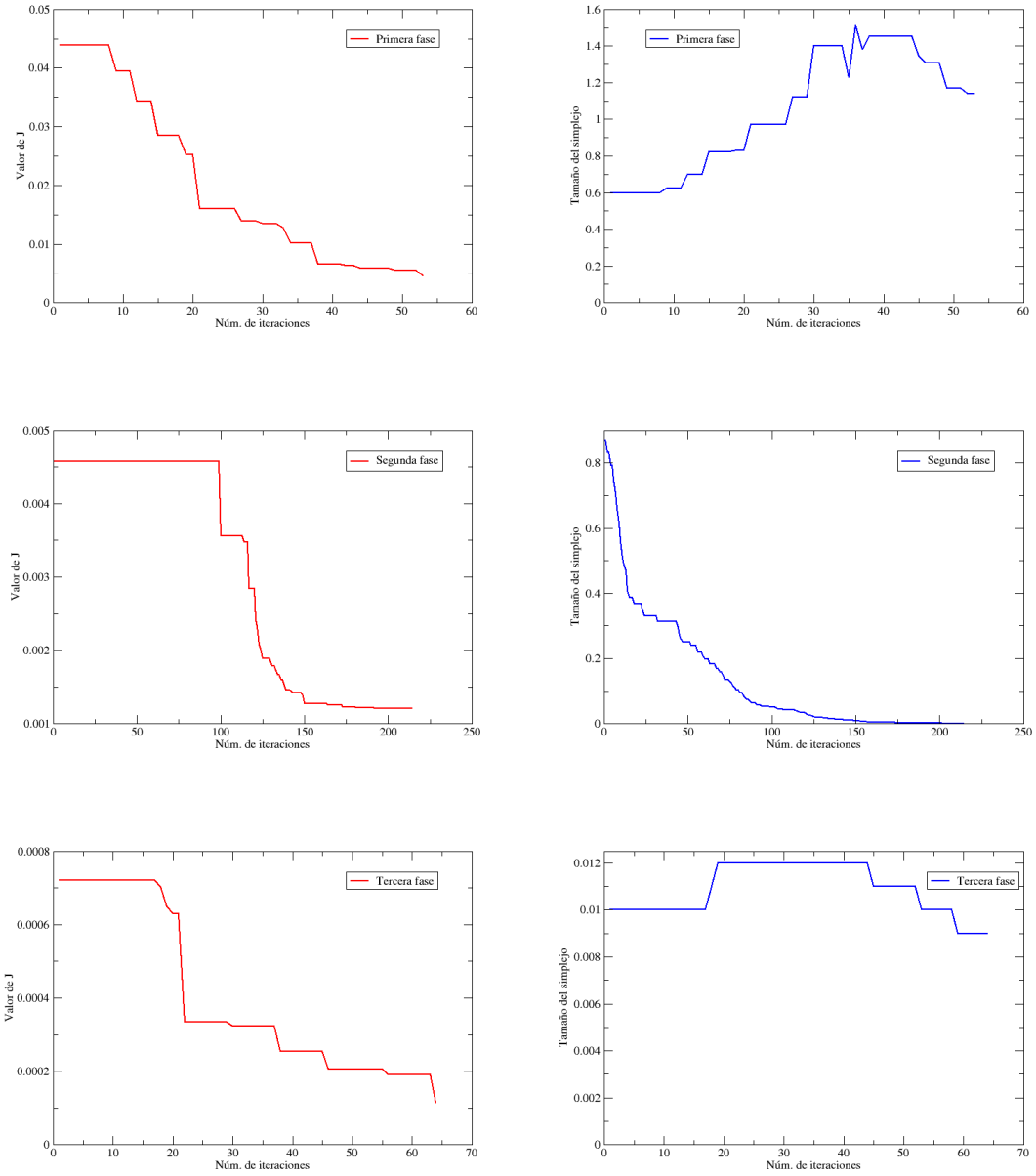

Figura 6.14: Evolución de J y variación del tamaño del simplejo a través de las diferentes fases del proceso de optimización.

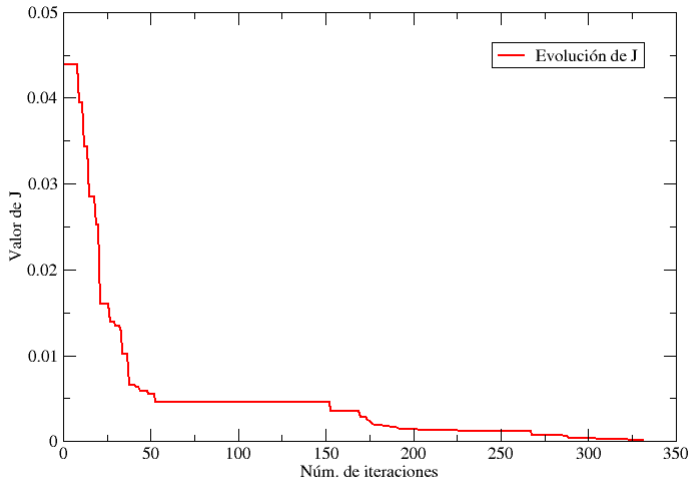

Figura 6.15: Evolución de J a través del proceso de optimización.

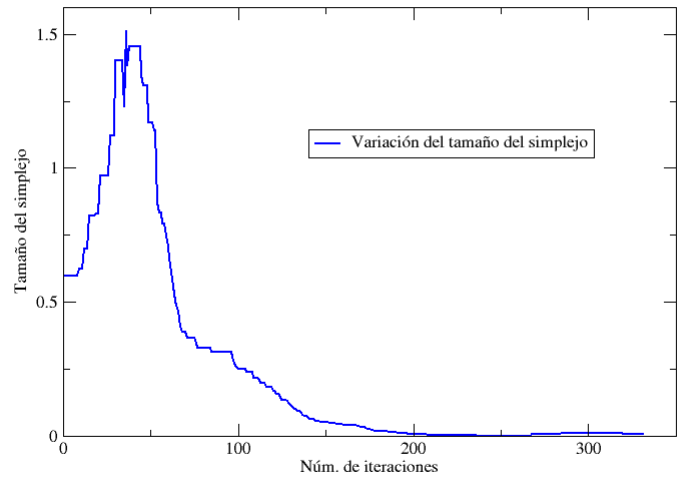

Figura 6.16: Variación del tamaño del simplejo a través del proceso de optimización.

Capítulo 7

Conclusión

Como se puede observar en los resultados obtenidos, el funcionamiento del programa es el correcto, ya que las soluciones tanto lineales como no lineales se comportan de la manera esperada. Tanto la parte del solver como la parte de optimización mostraron un buen desempeño, la primera logrando trabajar con discontinuidades y reducir la magnitud de las oscilaciones provocadas por los saltos de $u(x,t)$. La segunda, mostrando eficiencia debido al número no tan grande de iteraciones requeridas para alcanzar el óptimo, además del reducido número de variables necesarias para parámetrizar la condición inicial.

La propuesta sugerida para parametrizar y optimizar $u_0(x)$ en la tesis entregó buenos resultados, por lo que es una estrategia a tomar en cuenta cuando se esté optimizando sobre propuestas de condiciones iniciales o funciones que se deseen conocer. El hecho de que en la primera fase se optimizen tanto las x_i como las u_i nos permite crear un bosquejo burdo con un número pequeño de puntos la función a encontrar, permitiendo localizar las discontinuidades y los cambios de pendiente más bruscos de ésta. La ventaja dividir la optimización en distintas fases y que en las fases siguientes se aumenten los puntos coordenados con los que se describe $u_0(x)$, es que nos permite refinar este bosquejo y por lo tanto ir disminuyendo el valor de J, el cual representa el error que posee nuestra propuesta con respecto a los datos que se desean alcanzar.

Algunas de las ventajas que se tienen al cambiar las variables de optimización de x_i y u_i a solo u_i es que, en primer lugar, al momento de pasar de la primera

a la segunda fase, podemos conservar el número de variables con las que se está trabajando, evitando el aumento de dimensionalidad. Otra ventaja es que se evita que en un futuro existan muchas oscilaciones o estancamientos debido al cambio de los valores de todos los x_i de los pares ordenados, además de que al hacer este cambio se reduce la probabilidad de que dos puntos se empalmen o queden a una distancia menor al tamaño del elemento I_j.

Es muy importante tener un buen control de las variables x_i al momento de optimizar, ya que si no se cumple con la condición de que estas vayan en forma ascendente, también es importante que al momento de reconstruir $u(x,t)$ no se tengan elementos empalmados ya que esto ocasiona que la reconstrucción no se realice de forma correcta y el método RKDG falle. Otra condición que hay que tener en cuenta es que dos o más parámetros x_i no se pueden localizar en un sólo elemento I_j, ya que como la reconstrucción se hace tomando a $u_j^l(t)$ como constante, tener más puntos haría que $u_0(x)$ tuviera una discretización más fina que nuestra malla utilizada para el solver, lo cual no causa que el método falle, pero se considera como un error al momento de analizar los datos.

El cálculo de J necesita de la interpolación de la función objetivo $u^*(x)$, por que es necesario comparar la información de todos los nodos del mallado del solver con los valores de función objetivo para no perder ninguna información que el solver nos este arrojando, ya que la discretización más burda de la u^* podría ocultar esta. Las interpolaciones, a pesar de ser lineales, son buenas aproximaciones debido a la densidad de la malla con la que se trabaja. Una ventaja proveniente de usar una malla fina para el solver, es que se pueden aproximar los grados de libertad $u_j^l(t)$ como constantes sin perder mucha exactitud en la reconstrucción de u_h, ocasionando que las cuadraturas no necesiten un orden tan alto para calcular las integrales.

No se considera como un gran problema tener un cierto error en el ajuste, ya que al estar usando interpolaciones es muy difícil poder recrear la función exactamente, además ya que nuestra función objetivo proviene de un muestreo, esta puede contener ruido debido a mediciones en caso de que esta provenga de datos experimentales, lo cual es lo deseado, ya que lo que se quiere es poder simular los fenómeno de la naturaleza.

El método de Nelder-Mead es una buena elección como método de descenso, ya

que nos garantiza que el valor de J no va a incrementar, debido a que por iteración tenemos dos resultados posibles, los cuales son la disminución del valor de J o que éste permanezca igual. Además como ya se menciono anteriormente, el hecho de que este método no necesite el cálculo del gradiente facilita bastante su implementación para este tipo de problemas inversos.

Como todo método de solución numérico de ecuaciones diferenciales hay que tener siempre presente el cumplir con la condición CFL para evitar que el método diverja. También otro de los detalles a tomar en cuenta es la condición inicial que se toma como punto de inicio para el Nelder-Mead, ya que en algunos casos el método puede terminar sin descender o tardar mucho tiempo en alcanzar el óptimo. Se escogió $u_0(x) = 0$ como condición inicial, debido a que se considero como el inicio más neutro considerando el caso en que no se conociera nada acerca de la solución del problema, además los ejemplos que se manejaron no tenían valores máximos o mínimos muy alejados de 0.

Una de las partes del programa más difíciles de manejar fueron los parámetros para el criterio de paro y el tamaño de paso para los parámetros a optimizar, ya que el valor de J depende mucho de la magnitud de los datos utilizados, sumado al hecho de que debido al método de descenso, no se puede usar la variación de J como un criterio de paro, ya que hay regiones donde el valor de J no varia aún estando muy lejos del óptimo, como se puede observar en la sección de resultados, por lo que para hacer el algoritmo más general se decidió tomar una fracción de J como criterio de paro. En el caso del tamaño de paso, este es un poco difícil de manejar debido a que está relacionado con la velocidad de descenso, haciendo que si se escoge un mal tamaño de paso el algoritmo realice demasiadas iteraciones, por lo general para escogerlo se tomó en cuenta la escala de la variable a parametrizar, sin embargo a veces es necesario cambiar este valor dependiendo de la fase en la que se este ejecutando, ya que en algunas fases el descenso es más brusco y en otras más fino.

Es importante recordar que la ecuación de Burgers no viscoso no es reversible en el tiempo. Por lo tanto, en principio, no todos los estados conducen a un minimizador único. Observemos también que, en ausencia de viscosidad, las discontinuidades de choque pueden surgir en un tiempo finito incluso para datos iniciales uniformes.

La presencia de pequeñas oscilaciones en los datos iniciales no tienen consecuencia importantes, ya que se atenúan durante el proceso de evolución. Sin embargo, las discontinuidades fuera de lugar y, en términos más generales, los cuasi-shocks afectan severamente el proceso de optimización.

Para la parte de la paralelización se trabaja con OpenMP, en lugar de con MPI, ya que el primero presenta una mayor facilidad al momento de trabajar con el, además al momento de compararlos se obtuvieron mejores resultados en ahorro de tiempo para la parelelización hecha con OpenMP.

Finalmente debido al hecho de que el programa esta diseñado en forma modular, es muy simple modificar el programa para trabajar otros ejemplos en 1D, como implementar otros flujos numéricos, un Runge-Kutta de orden más alto o incluso otro modelo del teorema de transporte (ec. 3.4).

Capítulo 8

Trabajo futuro

Entre las mejoras que se le pueden hacer el programa está implementar el cambio de fase de forma automática, ya que actualmente es necesario volver a ejecutar el programa con los nuevos archivos, como el nuevo punto de partida para el optimizador el cual es la solución de la fase anterior, y el nuevo tamaño de paso para realizar la siguiente fase. Este cambio es fácil de realizar, ya que le algoritmo esta diseñado para duplicar automáticamente los puntos de la fase anterior. La razón por la que no se implementó esta automatización fue el hecho de que era más sencillo analizar como se tenían que ajustar los criterios de paro en cada fase si estas se ejecutaban de forma independiente.

En relación con lo mencionado anteriormente, es necesario implementar la condición de que al momento de duplicar los puntos, hay que verificar que dos o más puntos no queden en un sólo elemento I_j, ya que por el momento esta condición sólo se aplica durante la optimización de las x_i en la fase 1.

Algunos problemas interesantes que se podrían tratar son los problemas con condiciones de frontera no periódicas y los que presentan viscosidad, esta última sugerencia podría realizarse de forma relativamente sencilla debido a la estructura modular del código, sin embargo la primera es un poco más complicada de llevar a cabo ya que hay que modificar varias de las funciones ya implementadas. También se podría probar con otros flujos numéricos, para hacer el programa más general y ver como los diferentes tipos de flujos afectan a la solución.

Otras mejoras que se pueden realizar al algoritmo, es que en vez de realizar la

parelización con openMP se podría hacer con CUDA, logrando que el programa sea más rápido, ya que la estructura del RKDG se presta para que el cálculo de la información en cada elemento se realice en un núcleo independiente por cada paso de tiempo. También otra mejora que se podría realizar sería la modificación del funcional J, para que el cálculo de la diferencia entre las soluciones sea en forma relativa, similar al que se utiliza para calcular errores relativos, lo cual daría una gran ventaja al momento de imponer las condiciones de paro.

Finalmente creo que la evolución de este algoritmo sería la solución de problemas inversos en 2D, ya que con esto se podrían abarcar una gran cantidad de fenómenos que podrían ser simulados.

Bibliografía

[1] Allahverdi N, Pozo A, Zuazua E. Numerical Aspects of large-time optimal control of Burgers equation. ESAIM: M2AN; 13 Octubre 2015: 50, 1371-1401.

[2] Cockburn B, Shun CW. Runge-Kutta Discontinuos Galerkin Methods for Convection-Dominated Problems. Journal of Scientific Computing; Septiembre 2001: 16(3), 173-216.

[3] Gotlieb S, Shu CW. Total Variations Diminishing Runge-Kutta Schemes. Mathematical of Computation; Enero 1998: 221(67), 73-85.

[4] Van Groesen E, Molenaar J.Continuum Modeling in the Physical Sciences.10 ed. Estados Unidos. SIAM;2007

[5] Shearer M, Levy R. Partial Differential Equations An Introduction to Theory and Applications. 10 ed. Inglaterra. PRINCETON UNIVERSITY PRESS; 2015

[6] Ghil M, Malanotte-Rizzoli P. Data assimilation in meteorology and oceanography. Vol. 33 of Advances in Geophysics. Academic Press; 1991.

[7] Ignat LI, Pozo A, Zuazua E. Large-time asymptotics, vanishing viscosity and numerics for 1-D scalar conservation laws. Math. Comput; 2015: 84, 1633–1662.

[8] Kim YJ, Tzavaras AE. Diffusive N-waves and metastability in the Burgers equation. SIAM J. Math. Anal; 2001: 33, 607–633.

[9] Castro C, Palacios F, Zuazua E. An alternating descent method for the optimal control of the inviscid Burgers' equation in the presence of shocks. Math. Models Methods Appl. Sci; 2008: 18, 369–416.

[10] Cockburn B, Shu CW. The Runge–Kutta discontinuous Galerkin finite element method for conservation laws V: Multidimensional systems. J. Comput. Phys; 1998: 141, 199–224.

[11] Bassi F, Rebay S. High-order accurate discontinuous finite element solution of the 2D Euler equations. J. Comput. Phys; 1997: 138, 251–285.

[12] Biswas R, Devine KD, Flaherty J. Parallel, adaptive finite element methods for conservation laws. Appl. Numer. Math; 1994: 14, 255–283.

[13] Bassi F, Rebay S. A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier–Stokes equations. J. Comput. Phys; 1997: 131, 267–279.

[14] Cockburn B, Shu CW. The local discontinuous Galerkin method for timedependent convection-diffusion systems. SIAM J. Numer. Anal; 1998: 35, 2440–2463.

[15] Li Y, Osher S, Tsai R. Heat source identification based on l_1 constrained minimization. Inverse Probl. Imaging; 2014: 8, 199–221.

[16] Shu CW. TVD time discretizations. SIAM J. Sci. Stat. Comput; 1998; 9, 1073–1084.

[17] Hesthaven JS, Warburton T. Nodal Disconinuos Galerkin Methods Algorithms, Analysis, and Applications. 9 ed. Estados Unidos. Springer; 2008.

[18] Lagarias JC, Reeds JA, Wright MH, Wright PE. Convergence properties of the Nelder-Mead simplex method in low dimensions. SIAM J. OPTIM; Nov 1998: 9(1), 112-147.

[19] Mckinnon KIM, Convergence of the Nelder-Mead simplex method to a nonstationary point. SIAM J. OPTIM; May 1996: 9(1), 148-158.

[20] Shukla HS, Tamsir M, Srivastava VK, Kumar J. Numerical solution of two dimensional coupled viscous Burger equation using modified cubic B-spline differential quadrature method. API Advances; Nov 2014: 4(11), 117-134.