UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

POSGRADO EN CIENCIAS FÍSICAS

CCADET

SOLUCIÓN DE ECUACIONES DIFERENCIALES PARCIALES NO LINEALES UTILIZANDO MÉTODOS ESPECTRALES DE COLOCACIÓN CON POLINOMIOS DE CHEBYSHEV

GRADUACIÓN POR TESIS O ARTÍCULO DE INVESTIGACIÓN
QUE PARA OPTAR POR EL GRADO DE:
MAESTRÍA EN CIENCIAS FÍSICAS

PRESENTA:
ALEJANDRO ELIZONDO PEREA

DR. PABLO LUIS RENDÓN
CCADET

DRA. CATALINA STERN FORGACH
FACULTAD DE CIENCIAS

DR. CARLOS MÁLAGA IGUIÑIZ
FACULTAD DE CIENCIAS

MÉXICO, D. F. AGOSTO 2015

Resumen

Las ecuaciones diferenciales parciales (EDP) no lineales no suelen tener soluciones analíticas generales, por lo que es necesario hacer uso de herramientas computacionales para describirlas. Los métodos de colocación de Chebyshev ofrecen un balance de simplicidad computacional y gran precisión en sus soluciones. Al aplicar estos métodos a diferentes formas de la ecuación de Burgers y la ecuación de Korteweg-de Vries (KdV) y compararlos con soluciones analíticas se encontraron errores de órdenes de magnitud bajos, dándonos un nivel de precisión muy alto en su solución.

Agradecimientos

Gracias a mi supervisor el Dr. Pablo Rendón por guiarme en todo este proceso y por su gran apoyo, al Dr. Felipe Orduña y Roberto Velasco del CCADET por sumamente accesibles y otorgarme su tiempo para discutir problemas y darme un gran apoyo computacional. Gracias a la Dra. Catalina Stern y al Dr. Carlos Málaga por también haberme dado el apoyo y guianza necesaria para terminar este proyecto. De igual manera a todos los involucrados en el PAPIIT IN109214 y al CCADET por haber sido de gran apoyo durante todo este trayecto.

A mi familia, mi abuela Angelina Lazo y mi madre Rosa Aurora Perea, quienes me han apoyado durante toda mi vida y con quienes siempre estaré agradecido.

A mi amigo y colega Jorge de la Torre, que siempre ha prestado oído para preguntas y discusiones tanto científicas como personales.

Y en general quisiera agradecer a cualquier persona con interés científico, que gracias a ustedes todos nos alimentamos de ideas como un gran organismo. Todo para entretenernos y asombrarnos en este gran y genial universo.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Ecuaciones diferenciales parciales no lineales

Una ecuación diferencial es una ecuación que relaciona las derivadas de distintos órdenes de una función de una o más variables. Existen varias maneras de clasificar este tipo de ecuaciones. Se le llama ecuación diferencial ordinaria (EDO) a las ecuaciones diferenciales que tienen funciones que sólo dependen de una variable. Las ecuaciones diferenciales parciales (EDP) son ecuaciones diferenciales caracterizadas por contener funciones que dependen de varias variables, así como derivadas parciales de estas funciones con respecto a más de una de esas variables.

Las EDP son utilizadas para describir diversos fenómenos en los campos de las matemáticas aplicadas, biología, finanzas, ingeniería y en la física, donde en esta última es común encontrarlas describiendo fenómenos relacionados con el sonido, la transferencia de calor, electrodinámica, mecánica de fluidos, elasticidad mecánica y la mecánica cuántica.

Al igual que las ecuaciones algebráicas las ecuaciones diferenciales tienen un orden. Éste se determina según el grado de diferenciación máximo de la ecuación diferencial.

La clasificación de ecuaciones algebráicas se extiende a las ecuaciones diferenciales en términos de linealidad. Hay una diferencia crucial entre el análisis de una ecuación lineal a una no lineal. La manera más sencilla de distinguir linealidad en una ecuación es aplicando el principio de superposición de soluciones, el cual establece que si $u_1(x)$ y $u_2(x)$ son soluciones a una ecuación diferencial, entonces $u_1(x) + u_2(x)$ también es solución. Cualquier ecuación que no cumpla con el principio de linealidad cae en la categoría de no lineal.

Una manera más general de tratar la linealidad en ecuaciones no necesariamente diferenciales involucra el manejo del concepto de un operador lineal L. En el caso de las ecuaciones diferenciales el operador L es un operador diferencial que depende de la función $f = f(x_1, x_2, ..., x_n)$, de sus derivadas con respecto a las variables independientes, y de las variables independientes. La linealidad impone los dos requerimientos siguientes: [1]

$$L[u + v] = L[u] + L[v] \ , \tag{1.1}$$

$$L[cu] = cL[u] \tag{1.2}$$

donde c es una constante arbitraria, y u y v son funciones.

Las ecuaciones no lineales son particularmente difíciles de resolver explícitamente, por lo que los métodos para resolverlas de manera analítica son muy limitados.

Como se verá más delante, en este texto enfocaremos nuestro estudio en la solución de EDP no lineales, con énfasis principal en la la ecuación de Burgers y la ecuación de Korteweg–de Vries.

Una opción para estudiar una ecuación no lineal es aproximarla mediante ecuaciones lineales. A este método se le conoce como linealización. Esta técnica es empleada frecuentemente en la física y suele otorgar buenas aproximaciones del comportamiento del sistema. Obviamente no se decidió hacer utilizar este enfoque debido a que dicha aproximación eliminar la no linealidad y simplificar la ecuación no es parte del problema a analizar.

La otra opción que abordamos en el presente texto, consiste en resolver de manera aproximada la ecuación diferencial no lineal a través de un método computacional numérico.

1.2. Análisis y métodos numéricos

Históricamente los métodos numéricos han sido de gran utilidad para ayudar a resolver problemas de carácter científico. La primera evidencia de la aplicación de un método numérico data del año 7200 a.e.c. en Babilonia donde se aproxima el valor de $\sqrt{2}$ a seis decimales [2].

A diferencia de otros campos de las matemáticas, los métodos numéricos no se concentran en encontrar soluciones exactas, esto es debido a que en la práctica es frecuentemente

difícil encontrar este tipo de soluciones, por lo que los métodos numéricos se concentran más bien en funcionar como una alternativa cuando no se pueden encontrar soluciones analíticas o su solución es demasiado difícil.

El primer objetivo en mente al desarrollar la tecnología detrás de las computadoras fue el de poder llevar a cabo cálculos mecánicos llevarían mucho tiempo en llevar a cabo a una persona. Es por eso que a través del tiempo podemos distinguir que el desarrollo de nuevos métodos numéricos está totalmente relacionado con el avance tecnológico y computacional.

A través de este texto se hará uso de métodos espectrales, un tipo de método numérico orientado a la solución de ecuaciones diferenciales que se caracteriza por la obtención de resultados con un gran grado de precisión. Como se verá más delante, estos métodos son muy precisos y representan una carga computacional más ligera que los métodos de elementos finitos, sin embargo éstos se vuelven muy poco precisos cuando se emplean en geometrías complejas o al tratar de resolver sistemas con discontinuidades. [3]. Dichos métodos fueron desarrollados por el matemático americano Steve Orszag en 1969. [4]

1.3. Objetivos

Se escogió estudiar este tema por ser un buen punto de partida en el tema de métodos espectrales, pues engloba la complejidad matemática de describir fenómenos relacionados con la acústica y la mecánica de fluidos, así como el uso de un método particularmente preciso para resolver ecuaciones diferenciales. Como es claro, el tema a tratar está relacionado con el análisis numérico, área que se ha visto fuertemente desarrollada en las últimas décadas debido al creciente desarrollo tecnológico en procesadores lógicos computacionales.

La facilidad para poder llevar a cabo cálculos muy complejos con computadoras ordinarias de uso personal hacen de las simulaciones numéricas una de las principales herramientas para el estudio de sistemas no lineales.

En el texto presente se hará uso de un método espectral llamado método espectral por colocación de Chebyshev para resolver varios casos de la ecuación de Burgers. Como se explicará en los capítulos subsecuentes, la caracteríastica principal de los métodos espectrales es que éstos hacen uso de un grupo de funciones base en todo el dominio [3], siendo el método espectral por colocación de Chebyshev dicho método haciendo uso de los polinomios de Chebyshev.

Este método tiene la gran ventaja de ser bastante preciso aún cuando su desarrollo y aplicación suelen ser menos directos que en los casos de otros métodos numéricos, como los métodos por diferencias finitas o métodos de elementos finitos.

Una vez obtenidos los resultados otorgados por este método, se llevará a cabo una comparación con los resultados analíticos de la ecuación de Burgers. Ésto con el objetivo de encontrar el error relativo de la solución numérica. También se llevará a cabo el estudio de la respuesta de dicho error con respecto a parámetros de fineza de las matrices de diferenciación (parte esencial del cómputo numérico llevado a cabo por los métodos de colocación). Estos resultados se compararán con los obtenidos por *Khater et al.* [5].

Capítulo 2

Fundamentos

Viendo que nuestro estudio se concentra en el área de análisis numérico aplicado la ecuación de Burgers, muy conocida en la física, es necesario presentar el fondo de cada uno de estos conceptos. Es por esto que en esta sección se analizarán a detalle los conceptos detrás de los métodos espectrales, las ondas de choque y la ecuación de Burgers, así como la relaciones entre todos ellos.

2.1. Los métodos espectrales

Se les llama métodos espectrales (ME) al conjunto de técnicas de análisis de discretización espacial basados en la expansión de soluciones como coeficientes de ciertas funciones base [6]. Muy comúnmente son utilizados para resolver ecuaciones diferenciales.

Las funciones base usualmente tienen soporte global, lo que significa que el conjunto de funciones está definida en un dominio infinito y no son necesariamente cero [3]. Una vez obtenidos los coeficientes basados en estas funciones, éstos pueden ser considerados el *espectro* de la solución, por lo cual son llamados *métodos espectrales* [6].

Es por esta razón que los ME son particularmente poderosos, pues para calcular el valor de la derivada en un punto del espacio requieren información de todos los demás puntos del espectro, lo cual es muy difernete a los métodos de diferencias finitas. Esto significa que los ME usualmente tienen un orden de aproximación de orden muy alto. De hecho, se ME dan el valor exacto de la derivada de cierta función, considerando que el error que aparece es debido al truncamiento debido al conjunto de funciones base [6].

Desde el punto de vista computacional, los ME también tienen cierta ventaja. A diferencia de los métodos de diferencias finitas, o métodos de elementos finitos, los ME no

basan todo su poder de cómputo en la memoria de la computadora. Es por esta razón que son muy utilizados para analizar flujos en mecánica de fluidos.

Por otro lado, la principal desventaja en los ME es que son geométricamente menos flexibles que los métodos de órdenes menores [3]. Las discontinuidades en las funciones y problemas donde haya ondas de choque suelen aumentar considerablemente el error de cálculo.

Dicho lo anterior es importante notar que el objetivo de nuestro estudio es utilizar los ME para describir las ondas de choque. Siendo que las ondas de choque se caracterizan por la presencia de discontinuidades en algunas de las variables del fluido (temperatura, presión, densidad, etc.), es muy común recurrir a la *ecuación de Burgers* para aproximar dichas discontinuidades mediante un perfil suave en el que pueden aparecer pendientes pronunciadas. Esta ecuación permite dar un tratamiento continuo a la ecuación, aproximando la discontinuidad natural de las ondas de choque. El análisis de esta aproximación, así como la naturaleza detrás de las ondas de choque se comentarán a mayor detalle en las secciones siguientes.

2.1.1. Elección de funciones base

Por la misma naturaleza de los métodos espectrales, todas las funciones base deben de ser funciones suaves [3]. En la literatura es normal encontrarnos con las siguientes funciones utilizadas como funciones base: [6]

- Funciones trigonométricas (series de Fourier)
- Polinomios de Chebyshev
- Polinomios de Legendre
- Polinomios de bajo orden de Laguere
- B-Splines

Sin embargo, en este estudio sólo se hizo uso de los polinomios de Chebyshev como funciones base. Las razones por las cuales se escogieron éstos y no los demás se dará a conocer a continuación.

Es importante notar que las opciones más comunes como funciones base son las series de Fourier y los polinomios de Chebyshev. El principio para escogerlos es muy sencillo: si la función a analizar tiene una naturaleza periódica, siempre es mejor usar series de Fourier como herramienta, si no, los polinomios de Chebyshev son la elección más general. Como

regla general se suele hacer uso de los polinomios de Cheyshev como funciones base antes que cualquier otro set de polinomios. De haber periodicidad se puede hacer uso de otro set de funciones. [3]

Como se verá más adelante, la ecuación de Burgers no tiene condiciones de frontera periódicas por lo que los coeficientes de Chebyshev representan una elección ideal.

2.1.2. Colocación por Chebyshev

Los polinomios de Chebyshev son definidos en el dominio de $x \leq |1|$ de la siguiente manera:

$$T_n(x) = \cos(n \arccos x), \ n = 0, 1, 2, ... \tag{2.1}$$

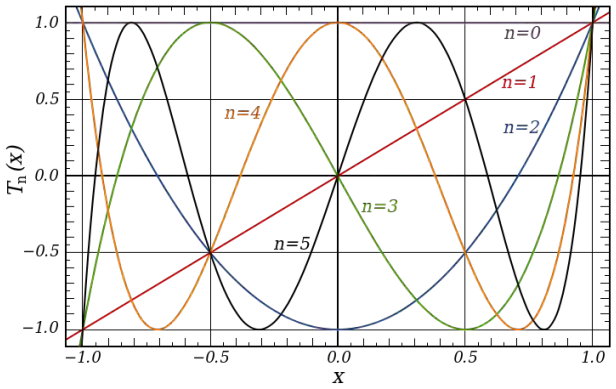

FIGURA 2.1: Primeros cinco polinomios de Chebyshev en un dominio de $-1 < x < 1$.

Se puede aproximar el valor de una función $u(x)$ haciendo uso de una serie finita de polinomios de Chevyshev, esto debido a las propiedades ortogonales de los polinomios que permiten reescribir una función como una suma infinta (que a fines prácticos puede ser aproximada y truncada). Dicha función $u(x)$ se escribiría de la siguiente manera:

$$u_N(x) = \sum_{n=0}^{N} a_n T_n(x), \text{ para una } N \text{ finita.} \tag{2.2}$$

siendo a_n el n-ésimo coeficiente de Chebyshev. Ahora bien, la solución más directa que se nos viene a la mente para trabajar con estos polinomios es manejar una distribución de puntos x_j equidistantes dado el dominio en el que se va a trabajar, sin embargo, la elección de una *malla equidistante* está asociada con el *fenómeno de Runge* [3]. El *fenómeno de Runge* es gráficamente muy similar al fenómeno de Gibs, y ocurre debido a

la inocurrencia de puntos suficientes donde evaluar regiones muy cercanas a los límites del dominio (particularmente son regiones poco suaves).

Para evitar problemas debido al *fenómeno de Runge* es necesario no hacer uso de una malla basada en una distribución más adecuada al tratamiento de suavidad del problema. Es por eso que se debe escoger una distribución de malla con mayor densidad en los límites del dominio. La distribución más usada para los polinomios de Chebysher es la *distribución de Gauss-Lobatto*, la cual se puede escribir la siguiente manera:

$$x_j = \cos\left(\frac{\pi j}{N}\right),\ j = 0, ..., N. \tag{2.3}$$

esto es, manejando el dominio de los polinomios T_n como $(-1, 1)$. Para casos donde el dominio es generalizado a (a, b), donde $a < b$, podemos utilizar la siguiente fórmula [6]:

$$x_j = \frac{1}{2}\left((a+b) - (b-a)\cos\left(\frac{\pi j}{N}\right)\right),\ j = 0, ..., N. \tag{2.4}$$

Gracias a las propiedades de ortogonalidad y manejo de funciones pares e impares, se puede llevar a cabo un tratamiento para relacionar los polinomios de Chebyshev con la expresión de la malla con los puntos de Gauss-Lobatto [3] [6]. Lo anterior describe bastante bien la intención detrás de los métodos de colocación en general, y ésta es de colocar un conjunto de puntos dado un dominio, de tal manera que convenientemente se ajusten a las propiedades de diferenciación de la función o polinimios a utilizar.

Así pues, los polinomios de Chebyshev T_n evaluados en los puntos de Gauss-Lobatto x_j están dados por:

$$T_n(x_j) = \cos\left(\frac{nj\pi}{N}\right),\ j, k = 0, ..., N. \tag{2.5}$$

Partiendo de la expresión anterior, se puede hacer uso del recurso de la transformada de Chebyshev para ejecutar derivadas con gran facilidad. Sin embargo, debido la practicidad del manejo computacional del método de colocación de Chebyshev se prefirió manejar la idea de construir la matriz de derivación D_{ij} correspondiente a los polinomios evaluados en los puntos escogidos (2.3). Dicha matriz nos servirá para llevar a cabo operaciones de diferenciación con un manejo computacional muy eficiente y sencillo.

Considerando que la derivada de la función discreta u_N se puede escribir como una multiplicación de matrices:

$$u'_N(x_i) = \sum_{j=0}^{N} D_{ij} u_N, \tag{2.6}$$

Entonces, mediante el uso de los polinomios de Chebyshev se pueden calcular los coeficientes de la matriz $\mathbf{D}$ y se pueden generalizar sus valores de la siguiente manera [6]:

$$D_{ij} = \begin{cases} \frac{c_i}{c_j} \frac{(-1)^{i+j}}{x_i - x_j}, & i \neq j \\ -\frac{x_i}{2(1-x_i^2)}, & 1 \leq i = j \leq N-1 \\ \frac{2N^2+1}{6}, & i = j = 0 \\ -\frac{2N^2+1}{6}, & i = j = N. \end{cases} \tag{2.7}$$

con el coeficiente c_i definido de la siguiente manera:

$$c_i = \begin{cases} 2, & i = 0, N \\ 1, & 1 \leq i \leq N-1. \end{cases} \tag{2.8}$$

Para el cómputo de esta matriz se escribieron las rutinas *chebdif.py* y *chebPy.py* que se pueden encontrar los apéndices.

Es importante agregar que en la literatura, los términos *método de colocación* y *métodos pseudoespectrales* son intercambiables. Sin embargo, es importante notar que un método de colocación busca encontrar una solución aproximada que satisfaga la ecuación evaluada en los puntos de colocación, mientras que un método es considerado pseudoespectral si no todas las partes de su algoritmo son ejecutadas en términos espectrales. Dicho lo anterior se puede considerar que un método de colocación siempre es pseudoespectral, mientras que un método pseudoespectral no es necesariamente un método de colocación. [7]

2.2. Ondas de choque

Físicamente, las ondas de choque se caracterizan por ser ondas que se propagan por un mecanismo de expansión-compresión dentro del medio donde la presión y densidad

cambian de manera súbita. La aparición de ondas de choque en la naturaleza a menudo se asocia con fenómenos no lineales que eventualmente dan lugar a una discontinuidad.

Una manera muy clara de visualizar a una onda de choque es como si se tratara de una discontinuidad en alguna de las propiedades que describen al medio (i.e. presión, delensidad) que viaja a una velocidad finita [8].

La descripción matemática de las ondas de choque es útil para varios campos de la física, principalmente la mecánica de fluidos, la aerodinámica y la acústica, aunque también es particularmente útil para describir fenómenos de tráfico y detonaciones [9].

El caso más sencillo de describir en la naturaleza es cuando un cuerpo alcanza la velocidad del sonido c a través del aire. Cuando $v = c$ el cuerpo viaja junto a las ondas que él mismo emite, razón por la cual se crea una *barrera de sonido*. A toda esta superposición de ondas viajando en el frente se les puede tratar como una onda de choque, puesto que en conjunto llevan a cabo una variación abrupta de las propiedades del medio (en el caso de un avión la variación súbita es de la presión) en variación espacial muy corta [8].

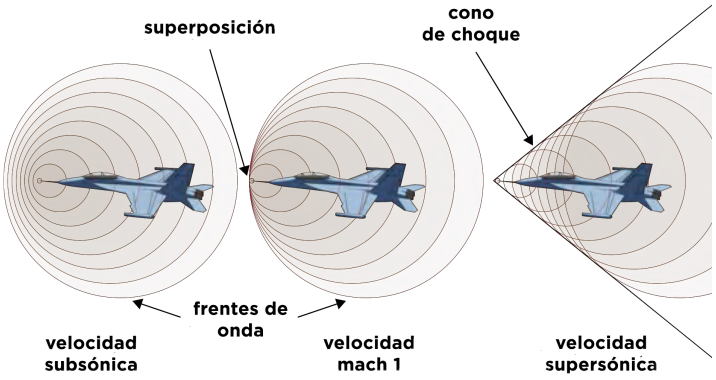

FIGURA 2.2: Descripción de cuando un jet viaja a velocidades subsónicas y supersónicas. Cuando el jet alcanza la velocidad del sonido, las ondas que emite se superponen junto con él creando un efecto de onda de choque.

2.3. Ecuación de Burgers

La ecuación de Burgers es un caso particular de la ecuación de Navier Stokes donde se hace uso de muchas simplifiaciones con el fin de describir el comportamiento interno de la no linealidad de la ecuación [10]. Para poder llegar a ella es necesario partir de la condición de incompresibilidad del fluido:

FIGURA 2.3: Cuando un jet viaja a la velocidad del sonido se crea una onda de choque que se puede visualizar como una fluctuación drástica en la presión del medio.

$$\nabla \cdot \mathbf{v} = 0 \tag{2.9}$$

donde ρ es la densidad del flujo, p la presión, $\mathbf{v}$ el vector de velocidad del flujo y μ es la viscosidad del fluido [10]. Podemos notar que el resultado de $\nabla \cdot \mathbf{v} = 0$ implica que el flujo es incompresible, por lo que la variación temporal de la densidad es cero ($\frac{\partial \rho}{\partial t} = 0$). También supongamos que los efectos de la gravedad son despreciables.

Expandiendo las operaciones vectoriales llegamos a la siguiente expresión:

$$\frac{\partial(\rho \mathbf{v})}{\partial t} + \nabla \cdot (\rho \mathbf{v}\mathbf{v}) + \nabla p - \mu \nabla^2 \mathbf{v} = 0 \ , \tag{2.10}$$

$$\rho \frac{\partial u}{\partial t} + \rho u \frac{\partial u}{\partial x} + \rho v \frac{\partial u}{\partial y} + \rho w \frac{\partial u}{\partial z} + \frac{\partial p}{\partial x} - \mu \Big(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \Big) = 0 \ , \tag{2.11}$$

donde (u, v, w) son los componentes de la velocidad en las coordenadas (x, y, z) respectivamente [10].

Haciendo un tratamiento unidimensional e ignorando los efectos del gradiente de presión la expresión se reduce a:

$$\rho\frac{\partial u}{\partial t} + \rho u\frac{\partial u}{\partial x} - \mu\frac{\partial^2 u}{\partial x^2} = 0 \ . \tag{2.12}$$

Si consideramos que la viscosidad cinemática, cantidad en la cual no participa ninguna fuerza, está definida como $\nu = \mu/\rho$, llegamos a la ecuación de Burgers:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2} \ , \tag{2.13}$$

donde $u = u(x,t)$. Para el caso donde la viscosidad se considera nula el término de la segunda derivada se cancela y obtenemos la ecuación de Burgers invíscida.

La ecuación de Burgers es una ecuación diferencial parcial (EDP) no lineal parabólica utilizada frecuentemente para describir fenómenos en los campos de acústica, mecánica de fluidos, ondas de tráfico, dinámica de gases y matemáticas aplicadas [11]. Es nombrada en referencia al físico americano-holandés Johannes Martinus Burgers que realizó sus estudios en en el área de mecánica de fluidos.

También, como se mencionó anteriormente, puede aparecer el caso donde $\nu = 0$, que quiere decir que el flujo no es viscoso y se suscitan discontinuidades en el flujo, describiendo ondas de choque [10]. Para el caso donde ν es suficientemente pequeña (del orden de 10^{-3}) [5], se puede aproximar el comportamiento de onda de choque sin tener discontinuidades en la función. Es por eso que dicha ecuación es utilizada para describir las ondas de choque y al mismo tiempo conservear las propiedades de suavidad de la función.

Otra característica sobre la ecuación de Burgers es que el componente que acompaña a la viscosidad da un comportamiento disipativo a la ecuación. Esto quiere decir que la onda de choque va perdiendo amplitud conforme esta evoluciona. Los efectos precisos de la disipación son debidos más puntualmente al fenómeno que trata de modelar la ecuación, pero en términos físicos ondulatorios dicha disipación es normalmente producto de la pérdida de energía de la onda por fricción o turbulencia.

Cabe agregar que existe una solución analítica para la ecuación de Burgers. Para el caso unidimensional es necesario hacer uso de la transformación de *Cole-Hopf* [10], como se muestra a continuación:

$$u(x,t) = -2\nu\frac{\partial \log v(x,t)}{\partial x} \tag{2.14}$$

Haciendo uso de esta transformación se puede llegar a la solución [9]:

$$u(x,t) = \frac{\int_{-\infty}^{\infty} \left(\frac{x-\tilde{x}}{t}\right) \exp\left(\frac{-G(x,\tilde{x},t)}{2\nu}\right)\, d\tilde{x}}{\int_{-\infty}^{\infty} \exp\left(\frac{-G(x,\tilde{x},t)}{2\nu}\right)\, d\tilde{x}}\ , \tag{2.15}$$

$$G(x,\tilde{x},t) = \frac{(x-\tilde{x})^2}{2t} + \int_0^{\tilde{x}} u_0(s)\ ds \tag{2.16}$$

Y u_0 es la condición inicial de u, por lo tanto la solución sólo se puede escribir de manera explícita si las integrales tienen una solución analítica.

También se puede demostrar que la siguiente función es solución para la ecuación de Burgers (2.13):

$$u(x,t) = C + A\tanh\left(-\frac{A}{2\nu}(x - Ct + B)\right) \tag{2.17}$$

Este tipo de solución es llamada choque de Taylor y podemos ver que esta expresión tiene la forma $u(x,t) = f(x-Ct)$, lo cual quiere decir que el perfil de la onda de choque se conserva y viaja a una velocidad constante C en una sola dirección.

En las secciones siguientes se mostrarán las soluciones numéricas a la ecuación de Burgers en los casos unidimensional y bidimensional, donde ambos casos se escriben como sistemas de ecuaciones acopladas, y también la ecuación KdV. Para todos estos casos existen soluciones analíticas que se pueden consultar en las referencias más adelante, y servirán para establecer la validez de nuestras soluciones numéricas.

2.3.1. Ecuación de Korteweg de Vries

Un caso que no tratamos a fondo en este estudio, pero es interesante mencionar es el de la ecuación Korteweg de Vries. Esta ecuación es utilizada para describir ondas en aguas no profundas, ondas en el óceano donde la densidad está estrificada [12] y ondas ión acústicas en plasmas [13]

Escrita de la siguiente manera,

$$\frac{\partial u}{\partial t} + \frac{\partial^3 u}{\partial x^3} + 6u\frac{\partial u}{\partial x} = 0\ , \tag{2.18}$$

la ecuación KdV se clasifica matemáticamente como una EDP no lineal donde, contrario a la ecuación de Burgers, aparecen efectos de dispersión. Puntalmente dicho comportamiento es debido a la tercera derivada de la ecuación y físicamente representa un fenómeno interesante: la diferencia de velocidad de fase entre ondas de amplitudes grandes comparadas con las de amplitudes pequeñas.

2.3.2. Ecuación de Korteweg de Vries-Burgers

Combinando los efectos de disipación (Burgers) y dispersión (KdV) obtenemos la ecuación de Korteweg de Vries-Burgers.

$$\frac{\partial u}{\partial t} + \mu \frac{\partial^3 u}{\partial x^3} + 2u \frac{\partial u}{\partial x} - \nu \frac{\partial^2 u}{\partial x^2} = 0 \ , \tag{2.19}$$

Donde ν es la viscosidad cinemática y μ la viscosidad dinámica. Como es de esperarse, los términos de μ y ν determinarán la tendencia disipativa y/o dispersiva de la ecuación.

Durante los análisis que veremos a continuación analizaremos la solución numérica de esta ecuación.

Capítulo 3

Metodología

En este capítulo se llevará a cabo un análisis del método de colocación basado en los polinomios de Chebyshev para resolver diferentes casos de la ecuación de Burgers. Como se explicará más delante, este método está basado en expandir las funciones como sumas de polinomios de Chebyshev para luego calcular las derivadas de la función haciendo uso de matrices de diferenciación (2.7). Una vez habiendo reescrito todas las variaciones espaciales en términos de matrices de diferenciación es necesario resolver la variación temporal del flujo. Para ésta se manejó el método de Runge-Kutta de cuarto orden, en vista de que la EDP se vuelve un sistema de ecuaciones diferenciales ordinarias (EDO), y no es recomendable hacer uso de métodos espectrales para resolver derivadas temporales, puesto que el dominio no está acotado, y una de las consideraciones más importantes al usar métodos espectrales es tener un dominio bien definido para poder hacer uso del caracter global de la diferenciación.

Las ecuaciones de Burgers a resolver son las siguientes [5]:

- Burgers en 1D

$$\frac{\partial u}{\partial t} + \alpha u \frac{\partial u}{\partial x} = \nu \frac{\partial^2 u}{\partial x^2} \ , \tag{3.1}$$

- Burgers en 2D

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + u\frac{\partial u}{\partial y} = \nu\Big(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\Big) \ , \tag{3.2}$$

- Sistema de ecuaciones de Burgers acopladas en 1D

$$\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} + 2u\frac{\partial u}{\partial x} + \alpha\frac{\partial uv}{\partial x} = 0 \tag{3.3}$$

$$\frac{\partial v}{\partial t} - \frac{\partial^2 v}{\partial x^2} + 2v\frac{\partial v}{\partial x} + \beta\frac{\partial uv}{\partial x} = 0 \ , \tag{3.4}$$

- Sistema de ecuaciones de Burgers acopladas en 2D

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = \nu\Big(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\Big) \tag{3.5}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = \nu\Big(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\Big) \ , \tag{3.6}$$

- KdV-Burgers

$$\frac{\partial u}{\partial t} + \alpha u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2} - \mu\frac{\partial^3 u}{\partial x^3} \ , \tag{3.7}$$

donde α, β, ν y μ son constantes arbitrarias.

Con el fin de poder determinar si nuestras soluciones se acercan a los resultados que deberíamos esperar, se llevará a cabo el cómputo de las diferencias con las soluciones analíticas para casos conocidos calculando el error relativo [5] [14]. De igual manera será necesario visualizar las diferencias gráficamente.

3.1. Soluciones numéricas de las distintas formas de la ecuación de Burgers usando el método espectral de Chebyshev

3.1.1. Ecuación de Burgers en 1D

Consideremos a la ecuación de Burgers:

$$\frac{\partial u}{\partial t} + \alpha u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2} \ , \tag{3.8}$$

con la condición inicial

$$u(x, 0) = f(x). \tag{3.9}$$

Como se mostró en el capítulo anterior, podemos hacer uso de la matriz de diferenciación de Chebyshev (2.7) para calcular las derivadas de $u(x,t)$. Notando que el método de colocación requiere de un conjunto de puntos no equidistantes entre sí, se sugiere manejar el siguiente patrón de puntos:

$$x_n = \frac{1}{2}\Big((a+b) - (b-a)\cos\Big(\frac{\pi n}{N}\Big)\Big),\ n = 0, 1, ...N\ , \tag{3.10}$$

debido a que los polinomios de Chebyshev están definidos con los extremos en $T_N = \pm 1$. Esta distribución de puntos guarda la misma relación que la de la ecuación (2.3), con la diferencia de que ahora se permite manejar espectros de puntos en el intervalo (a, b).

Una vez discretizando y manejando la matriz de diferenciación podemos llegar a la expresión siguiente para $u_x(x_i, t)$:

$$u_x(x_i, t) = \sum_{n=0}^{N} [D_x]_{in} u(x_n, t)\ , \tag{3.11}$$

donde $[D_x]$ representa la matriz de diferenciación para la variable x y $[D_x]_{in}$ es el valor del i-ésimo renglón y n-ésima columna de la matriz.

Visto esto de otra manera, la operación que se muestra anteriormente es el resultado de la multiplicación de una matriz cuadrada por un vector.

Podemos calcular la segunda derivada de $u(x,t)$ usando el resultado anterior:

$$u_{xx}(x_i, t) = \sum_{n=0}^{N} [D_x]_{in} u_x(x_n, t)\ , \tag{3.12}$$

$$u_{xx}(x_i, t) = \sum_{j=0}^{N} \Big(\sum_{n=0}^{N} [D_x]_{in}[D_x]_{nj} \Big) u(x_j, t)\ , \tag{3.13}$$

$$u_{xx}(x_i, t) = \sum_{j=0}^{N} [F_x]_{ij} u(x_j, t). \tag{3.14}$$

Viendo lo anterior nos damos cuenta que para esta matriz de diferenciación D_x, podemos encontrar derivadas de orden superior simplemente ejecutando una multiplicación por

sí misma [**?**]:

$$[F_x]_{ij} = \sum_{n=0}^{N} [D_x]_{in}[D_x]_{nj}, i, j = 0, 1, ..., N. \tag{3.15}$$

$$F_x = D_x^2. \tag{3.16}$$

Se sugiere por parte de Khater y Trefethen hacer una distinción en el cómputo de los valores de las derivadas en las condiciones de frontera. [5] [**?**]. La razón principal de ello es para separar las operaciones matriciales de la primera y última columna de la matriz D_x. Esto facilita notablemente su identificación y hacen el método más fácil de replicar para otras EDP [**?**]. Esta distinción para las condiciones de frontera se lleva a cabo definiendo $d_i(t)$ y $\tilde{d}_i(t)$ de la siguiente manera:

$$d_i(t) = [D_x]_{i0}u_0(t) + [D_x]_{iN}u_N(t) \ , \tag{3.17}$$

$$\tilde{d}_i(t) = [F_x]_{i0}u_0(t) + [F_x]_{iN}u_N(t). \tag{3.18}$$

Dicho de otra forma, tanto $d_i(t)$ y $\tilde{d}_i(t)$ representan el cómputo de la suma de multiplicación de la primera y última columna de la matriz **D** (o **F**) y el vector de valores de $u(t)$ en la frontera.

Una vez calculado lo anterior podemos sumarlo al cómputo de las diferencias para tener el valor completo:

$$u_x(x_i, t) = d_i(t) + \sum_{n=1}^{N-1} [D_x]_{in}u(x_n, t) \ , \tag{3.19}$$

$$u_{xx}(x_i, t) = \tilde{d}_i(t) + \sum_{n=1}^{N-1} [F_x]_{in}u(x_n, t). \tag{3.20}$$

Sustituyendo las expresiones anteriores (3.19) y (3.20) en (3.8) obtenemos lo siguiente:

$$\frac{du_i(t)}{dt} + \alpha u_i(t) \sum_{n=1}^{N-1} [D_x]_{in}u_n(t) - \nu \sum_{n=1}^{N-1} [F_x]_{in}u_n(t) + \alpha u_i(t)d_i(t) - \nu\tilde{d}_i(t) = 0. \tag{3.21}$$

Llamándole a la expresión $G(t, u(t)) = \frac{du_i(t)}{dt}$ y tomando en cuenta la condición inicial $u(0) = u_0$ podemos definir un sistema de ecuaciones diferenciales ordinarias, lo que significa que podemos resolver el sistema en el tiempo con el método de Runge-Kutta de orden cuatro [5]:

$$k_1 = G(t_n, u(t_n)) \tag{3.22}$$

$$k_2 = G(t_n + \Delta t/2, u(t_n) + k_1 \Delta t/2) \tag{3.23}$$

$$k_3 = G(t_n + \Delta t/2, u(t_n) + k_2 \Delta t/2) \tag{3.24}$$

$$k_4 = G(t_n + \Delta t, u(t_n) + k_3 \Delta t) \tag{3.25}$$

$$u(t_{n+1}) = u(t_n) + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{3.26}$$

Una vez calculada u(x,t) por este método podemos compararla con la solución analítica $\bar{u}(x,t)$. Para eso calculamos la norma del error relativo [5]:

$$|E| = \sqrt{\frac{\sum_{i=1}^{N_{ip}} (u_i - \bar{u}_i)^2}{\sum_{i=1}^{N_{ip}} \bar{u}_i^2}} \tag{3.27}$$

Donde N_{ip} es el número de puntos totales en el espectro de la solución. Para este cálculo no se toman en cuenta los puntos de las condiciones en la frontera, debido a que estos puntos son valores donde tanto el método numérico como la solución analítica comparten el mismo valor, por lo que su variación es nula y ésto sesga levemente el cálculo del error comparando la solución numérica y analítica.

3.1.2. Ecuación de Burgers en 2D

Consideremos la ecuación de Burgers en 2D:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + u\frac{\partial u}{\partial y} = \nu\Big(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\Big) \ , \tag{3.28}$$

con una función arbitraria como condición inicial

$$u(x, y, 0) = f(x, y). \tag{3.29}$$

Podemos encontrar una solución aproximada basándonos en el método de colocación por polinomios de Chebyshev. Para ello es necesario encontrar un conjunto de puntos para x y y. Estos puntos siguen la distribución de Gauss-Lobatto vista en (2.3), con un espectro en (a, b), dada por:

$$x_n = \frac{1}{2}\Big((a+b) - (b-a)\cos\Big(\frac{\pi n}{N}\Big)\Big),\ n = 0, 1, ...N\ , \tag{3.30}$$

$$y_m = \frac{1}{2}\Big((a+b) - (b-a)\cos\Big(\frac{\pi m}{M}\Big)\Big),\ m = 0, 1, ...M. \tag{3.31}$$

Una vez teniendo estos puntos podemos hacer uso de las matrices de diferenciación. Estas matrices serán diferentes para cada coordenada y serán llamadas D_x y D_y, esta última calculada con la distribución de y_m.

Llevando a cabo un tratamiento similar que para el caso de la ecuación de Burgers unidimensional podemos llegar a las siguientes expresiones para cada una de las derivadas espaciales:

$$u_x(x_i, y_j, t) = d_{ij} + \sum_{n=1}^{N-1} [D_x]_{in} u_{nj}(t)\ , \tag{3.32}$$

$$u_y(x_i, y_j, t) = e_{ij} + \sum_{m=1}^{M-1} [D_y]_{im} u_{mj}(t)\ , \tag{3.33}$$

$$u_{xx}(x_i, y_j, t) = \tilde{d}_{ij} + \sum_{n=1}^{N-1} [E_x]_{in} u_{nj}(t)\ , \tag{3.34}$$

$$u_{yy}(x_i, y_j, t) = \tilde{e}_{ij} + \sum_{m=1}^{M-1} [E_y]_{im} u_{mj}(t)\ , \tag{3.35}$$

donde d_{ij} y $\tilde{d}_{ij}$ están definidos por (3.17) y (3.18), respectivamente. Los valores de e_{ij} y $\tilde{e}_{ij}$ en las expresiones anteriores están relacionados con el cómputo de matriz de diferenciación para los puntos donde u_y y u_{yy} se evaluan en las condiciones de frontera. Dichos valores pueden ser calculados de la manera siguiente:

$$e_i(t) = [D_y]_{i0} u_0(t) + [D_y]_{iM} u_M(t)\ , \tag{3.36}$$

$$\tilde{e}_i(t) = [F_y]_{i0} u_0(t) + [F_y]_{iM} u_M(t). \tag{3.37}$$

Sustituyendo las expresiones (3.32), (3.33), (3.34) y (3.35) en (3.2) se obtiene:

$$\frac{du_{ij}(t)}{dt} = -u_{ij}(t)\left(d_{ij}(t) + \sum_{n=1}^{N-1}[D_x]_{in}u_{nj}(t)\right) - u_{ij}(t)\left(e_{ij}(t) + \sum_{m=1}^{M-1}[D_y]_{jm}u_{im}(t)\right) + \left(\tilde{d}_{ij}(t) + \sum_{n=1}^{N-1}[F_x]_{in}u_{nj}(t)\right) + \nu\left(\tilde{e}_{ij}(t) + \sum_{m=1}^{M-1}[F_y]_{jm}u_{im}(t)\right). \tag{3.38}$$

La expresión anterior junto con la condición inicial (3.29) evaluada en los puntos de colocación forman un sistema de ecuaciones diferenciales ordinarias que pueden ser resueltas para $u_{ij}(t)$ con el método de Runge-Kutta de cuarto orden, como es calculado para el caso unidimensional (3.22) con la diferencia de que el método incluye el cómputo en la dimensión de y para cada punto.

3.1.3. Sistema de ecuaciones de Burgers acopladas en 1D

Consideremos el siguiente sistema de ecuaciones de Burgers:

$$\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} + 2u\frac{\partial u}{\partial x} + \alpha\frac{\partial (uv)}{\partial x} = 0 \ , \tag{3.39}$$

$$\frac{\partial v}{\partial t} - \frac{\partial^2 v}{\partial x^2} + 2v\frac{\partial v}{\partial x} + \beta\frac{\partial (uv)}{\partial x} = 0 \ , \tag{3.40}$$

donde u y v son los perfiles de la primera y la segunda onda respectivamente. Se trata de un sistema de ecuaciones no lineales, de primer orden en tiempo, y de segundo orden en la coordenada espacial. Para este sistema manejemos las condiciones iniciales siguientes:

$$u(x,0) = f_1(x) \ , \tag{3.41}$$

$$v(x,0) = f_2(x). \tag{3.42}$$

Haciendo uso de las expresiones vistas para la ecuación de Burgers unidimensional con d y $\tilde{d}$ para las condiciones a la frontera (3.19) y (3.20), y sustituyéndolas en (3.39) y (3.40), e introduciendo el valor de r y $\tilde{r}$ expresado de la siguiente forma:

$$r_i(t) = [D_x]_{i0}v_0(t) + [D_x]_{iN}v_N(t) \ , \tag{3.43}$$

$$\tilde{r}_i(t) = [F_x]_{i0}v_0(t) + [F_x]_{iN}v_N(t) \ , \tag{3.44}$$

obtenemos las siguientes expresiones, que en conjunto forman un sistema de ecuaciones diferenciales ordinarias:

$$\frac{du_i(t)}{dt} = \sum_{n=1}^{N-1}[F_x]_{in}u_n(t) - 2u_i(t)\sum_{n=1}^{N-1}[D_x]_{in}u_n(t) - \alpha u_i(t)\sum_{n=1}^{N-1}[D_x]_{in}v_n(t)$$
$$- \alpha v_i(t)\sum_{n=1}^{N-1}[D_x]_{in}u_n(t) + \tilde{d}_i(t) - 2u_i(t)d_i(t) - \alpha u_i(t)r_i(t) - \alpha v_i(t)d_i(t)\ , \quad (3.45)$$

$$u_i(0) = f_1(x_i)\ , \quad (3.46)$$

$$\frac{dv_i(t)}{dt} = \sum_{n=1}^{N-1}[F_x]_{in}v_n(t) - 2v_i(t)\sum_{n=1}^{N-1}[D_x]_{in}v_n(t) - \beta u_i(t)\sum_{n=1}^{N-1}[D_x]_{in}v_n(t)$$
$$- \beta v_i(t)\sum_{n=1}^{N-1}[D_x]_{in}u_n(t) + \tilde{r}_i(t) - 2v_i(t)r_i(t) - \beta u_i(t)r_i(t) - \beta v_i(t)d_i(t)\ , \quad (3.47)$$

$$v_i(0) = f_2(x_i). \quad (3.48)$$

Este sistema se resuelve numéricamente usando el método de Runge-Kutta de cuarto orden.

3.1.4. Sistema de ecuaciones de Burgers acopladas en 2D

Consideremos el sistema de ecuaciones acopladas definidas por

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = \nu\Big(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\Big)\ , \quad (3.49)$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = \nu\Big(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\Big)\ , \quad (3.50)$$

con la siguientes condiciones iniciales:

$$u(x, y, 0) = f_1(x, y)\ , \quad (3.51)$$

$$v(x, y, 0) = f_2(x, y). \quad (3.52)$$

Y las condiciones de frontera:

$$u(x, y, t) = p_1(x, y, t)\ , \quad (3.53)$$

$$v(x, y, t) = p_2(x, y, t). \quad (3.54)$$

Siguiendo el mismo tratamiento que para el caso del sistema de ecuaciones acopladas unidimensional, ya que se trata del mismo tipo de ecuaciones, podemos sustituir las expresiones (3.19) y (3.20) en (3.49) y (3.50), así como las expresiones para las condiciones de frontera para cada una de las dimensiones (tal como se analizó anteriormente en las expresiones (3.17) y (3.18)).

Después de la sustitución se llega al siguiente sistema de ecuaciones:

$$\begin{aligned}\frac{du_{ij}(t)}{dt} &= -u_{ij}(t)\left(d_{ij}(t)+\sum_{n=1}^{N-1}[D_x]_{in}u_{nj}(t)\right)-v_{ij}(t)\left(e_{ij}(t)+\sum_{m=1}^{M-1}[D_y]_{jm}u_{im}(t)\right)\\ &\quad+\nu\left(\tilde{d}_{ij}(t)+\sum_{n=1}^{N-1}[F_x]_{in}u_{nj}(t)\right)+\nu\left(\tilde{e}_{ij}(t)+\sum_{m=1}^{M-1}[F_y]_{jm}u_{im}(t)\right), \qquad (3.55)\\ u_{ij}(0) &= f_1(x_i,y_j). \qquad (3.56)\end{aligned}$$

$$\begin{aligned}\frac{dv_{ij}(t)}{dt} &= -u_{ij}(t)\left(w_{ij}(t)+\sum_{n=1}^{N-1}[D_x]_{in}v_{nj}(t)\right)-v_{ij}(t)\left(k_{ij}(t)+\sum_{m=1}^{M-1}[D_y]_{jm}v_{im}(t)\right)\\ &\quad+\nu\left(\tilde{w}_{ij}(t)+\sum_{n=1}^{N-1}[F_x]_{in}v_{nj}(t)\right)+\nu\left(\tilde{k}_{ij}(t)+\sum_{m=1}^{M-1}[F_y]_{jm}v_{im}(t)\right), \qquad (3.57)\\ v_{ij}(0) &= f_2(x_i,y_j). \qquad (3.58)\end{aligned}$$

Las expresiones de frontera discretizadas en dos dimensiones se expresan de la manera siguiente:

$$\begin{aligned}d_{ij}(t) &= [D_x]_{i0}u_{0j}(t)+[D_x]_{iN}u_{Nj}(t), \qquad (3.59)\\ \tilde{d}_{ij}(t) &= [F_x]_{i0}u_{0j}(t)+[F_x]_{iN}u_{Nj}(t), \qquad (3.60)\\ e_{ij}(t) &= [D_y]_{j0}u_{i0}(t)+[D_y]_{jM}u_{iM}(t), \qquad (3.61)\\ \tilde{e}_{ij}(t) &= [F_y]_{j0}u_{i0}(t)+[F_y]_{jM}u_{iM}(t), \qquad (3.62)\\ w_{ij}(t) &= [D_x]_{i0}v_{0j}(t)+[D_x]_{iN}v_{Nj}(t), \qquad (3.63)\\ \tilde{w}_{ij}(t) &= [F_x]_{i0}v_{0j}(t)+[F_x]_{iN}v_{Nj}(t), \qquad (3.64)\\ k_{ij}(t) &= [D_y]_{j0}v_{i0}(t)+[D_y]_{jM}v_{iM}(t), \qquad (3.65)\\ \tilde{k}_{ij}(t) &= [F_y]_{j0}v_{i0}(t)+[F_y]_{jM}v_{iM}(t). \qquad (3.66)\end{aligned}$$

Sustituyendo las expresiones anteriores en el sistema podemos resolver para su derivada en el tiempo con Runge-Kutta de orden 4 y obtener el perfil de comportamiento de $u(x, y, t)$ y $v(x, y, t)$.

3.1.5. Ecuación Korteweg de Vries - Burgers (KdV-Burgers)

Escribimos la ecuación KdV-Burgers de la manera siguiente,

$$\frac{\partial u}{\partial t} + \alpha u \frac{\partial u}{\partial x} = \nu \frac{\partial^2 u}{\partial x^2} - \mu \frac{\partial^3 u}{\partial x^3} \ , \tag{3.67}$$

considerando la condición inicial descrita por

$$u(x, 0) = f(x) \ , \tag{3.68}$$

y las condiciones de frontera siguientes para $u(x, t)$ y $u_x(x, t)$

$$u(x, t) = g(t) \ , \tag{3.69}$$

$$u_x(x, t) = h(t) \ , \tag{3.70}$$

donde $g(t)$ y $h(t)$ son el perfil de la onda en la frontera.

Podemos llevar a cabo el mismo procedimiento de discretización proponiendo una solución numérica, así como se ha analizado en los cuatro casos anteriores. Sustituyendo (3.19) y (3.20) en (3.67) llegamos a la siguiente expresión:

$$\frac{du_i(t)}{dt} + \alpha u_i(t) \sum_{n=1}^{N-1} [D_x]_{in} u_n(t) - \nu \sum_{n=1}^{N-1} [F_x]_{in} u_n(t) + \mu \sum_{n=1}^{N-1} [G_x]_{in} u_n(t) + \alpha u_i(t) d_i(t) - \nu \tilde{d}_i(t) + \mu \tilde{\tilde{d}}_i(t) = 0 \ , \tag{3.71}$$

donde la matriz G_x se expresa de la siguiente manera:

$$[G_x]_{ij} = \sum_{n=0}^{N} [D_x]_{in}[F_x]_{nj}, i, j = 0, 1, ..., N \ , \tag{3.72}$$

con la condición inicial:

$$u_i(0) = f(x_i) \ , \tag{3.73}$$

considerando también la facilidad de su cómputo basado en la multiplicación de matrices de la manera siguiente:

$$G_x = D_x^3. \tag{3.74}$$

También es necesario notar que en la expresión (3.71) aparece el término $\tilde{\tilde{d}}_i(t)$. Siguiendo la lógica utilizada para calcular $\tilde{d}_i(t)$, podemos escribir:

$$\tilde{\tilde{d}}_i(t) = [G_x]_{i0}u_0(t) + [G_x]_{iN}u_N(t) \ , \tag{3.75}$$

que facilita calcular la multiplicación de la matriz G_x y el vector $u(t)$ evaluado en las condiciones de frontera.

Después de resolver para la derivada con respecto al tiempo de la ecuación (3.71), hacer uso de (3.73) y sustituir las expresiones para la evaluación de las condiciones en la frontera podemos utilizar el método numérico de Runge-Kutta de cuarto orden para obtener $u(x, t)$.

Capítulo 4

Resultados

En este capítulo mostraremos los resultados numéricos obtenidos mediante los esquemas descritos en el capítulo anterior. Las simulaciones numéricas fueron realizadas con la ayuda de las librerías `numpy` y `scipy`, incluidas en la distribución científica *Anaconda* para `Python`. Las rutinas y métodos utilizados pueden ser consultados en el Apéndice A.

4.1. Resultados numéricos de la ecuación de Burgers

4.1.1. Ecuación de Burgers unidimensional

Consideremos a la ecuación de Burgers dimensional (3.8) con la solución analítica siguiente, conocida como choque de Taylor: [14]

$$u(x,t) = \frac{c}{\alpha} + (2\eta/\alpha)\tanh(x - ct)) \tag{4.1}$$

Ésta es una típica solución de onda viajera, que representa un perfil que viaja a velocidad constante c sin cambiar de forma.

Asumiendo valores de $a = 0$, $b = 1$ y $c = 0.1$, con una $N = 10$ podemos comparar los resultados analíticos y numéricos calculando el error relativo (3.27). Para los valores de la tabla 4.1 variamos los valores de α y η, así como el valor de t para el cual se evalua $u(x,t)$.

También se analizó la relación que hay entre el tamaño de la matriz de diferenciación N y el error relativo calculado. Para llevar a cabo esto fue necesario fijar otras variables

$\alpha = 1.0$	η	$\lvert E\rvert_{t=0.1}$	$\lvert E\rvert_{t=0.2}$	$\lvert E\rvert_{t=0.3}$
	0.01	2.07×10^{-4}	4.07×10^{-4}	6.01×10^{-4}
	0.001	2.51×10^{-6}	4.97×10^{-6}	7.41×10^{-6}
	0.0001	7.50×10^{-8}	1.47×10^{-7}	2.16×10^{-7}

$\alpha = 0.1$	η	$\lvert E\rvert_{t=0.1}$	$\lvert E\rvert_{t=0.2}$	$\lvert E\rvert_{t=0.3}$
	0.01	6.29×10^{-4}	7.34×10^{-4}	5.02×10^{-4}
	0.001	1.37×10^{-6}	4.31×10^{-6}	5.40×10^{-6}
	0.0001	7.32×10^{-8}	7.27×10^{-7}	8.33×10^{-7}

TABLA 4.1: Resultados del cómputo del error relativo de la solución numérica a la ecuación de Burgers unidimensional

de la ecuación (3.8); en este caso se hizo el análisis fijando $\alpha = 0.1$, $c = 0.1$ y $\eta = 0.01$. Se puede observar en la figura 4.1 que cuando N aumenta el error relativo tiende a disminuir, como se esperaría que ocurriera. Es importante agregar que también se llevaron a cabo simulaciones con valores de N mayores a los mostrados en la gráfica. Sin embargo éstos tienden a darnos un valor de error relativo fuera de lo esperado, debido a que las operaciones matriciales empiezan a cargar con *overflow*.

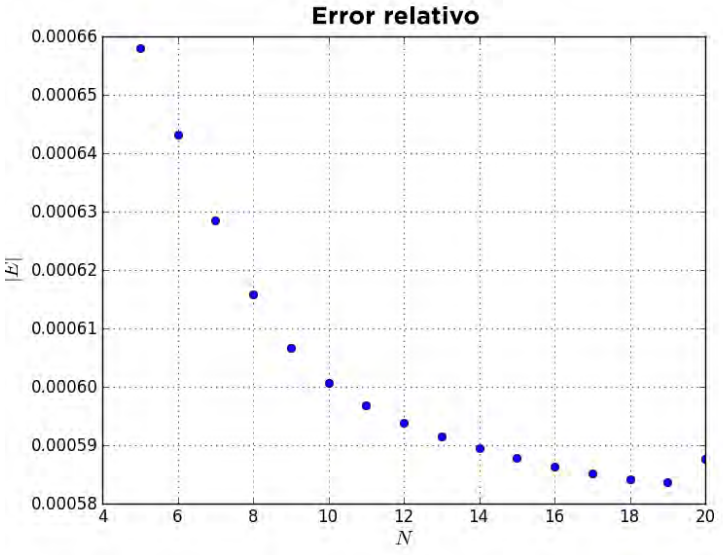

FIGURA 4.1: Error relativo variando el tamaño N de la matriz de diferenciación.

Otro caso que se analizó fue la comparación de soluciones por el método espectral de colocación contra el resultado analítico de la EDP. Esto se puede llevar a cabo manejando condiciones iniciales y de frontera que den lugar a un sistema integrable, para poder obtener una solución analítica como se mostró en (2.15).

$x = 0.25$	t	Chebyshev	Solución analítica
	0.4	0.317 52	0.317 52
	0.6	0.246 14	0.246 14
	0.8	0.199 55	0.199 56
	1.0	0.165 59	0.165 60
	3.0	0.027 75	0.027 75

$x = 0.75$	t	Chebyshev	Solución analítica
	0.4	0.645 62	0.645 62
	0.6	0.502 67	0.502 68
	0.8	0.385 34	0.385 34
	1.0	0.295 85	0.295 86
	3.0	0.030 44	0.030 44

TABLA 4.2: Resultados computados para $u(x,t)$ de las soluciones numéricas y analíticas de la ecuación de Burgers unidimensional.

Procediendo de la misma manera que Khater para poder comparar más fácilmente, utilizamos las siguientes condiciones iniciales:

$$u(x,0) = 4x(1-x), \qquad 0 \leq x \leq 1 \ , \tag{4.2}$$

y condición de frontera

$$u(0,t) = u(1,t) = 0, \qquad t > 0 \ , \tag{4.3}$$

para resolver y comparar analíticamente.

Usando el método de colocación por Chebyshev con valores de $N = 15$, $\Delta t = 0.001$ y $\eta = 0.1$ podemos ver que las diferencias entre el método numérico y la solución analítica son prácticamente nulas (tabla 4.2). Se han hecho también comparaciones de nuestro método de Chebyshev con métodos numéricos de elementos finitos [15] encontrando que, como se mencionó anteriormente, éstos últimos nos otorgan una precisión mucho menor (con un error de un orden de magnitud cuyo valor es aproximadamente 10^{-4}) que la encontrada con Chebyshev.

Si graficamos la evolución de la onda podemos ver como ésta va adquiriendo un perfil de onda de choque (figuras 4.2 y 4.3, para valores de $\eta = 0.01$ y $\eta = 0.1$).

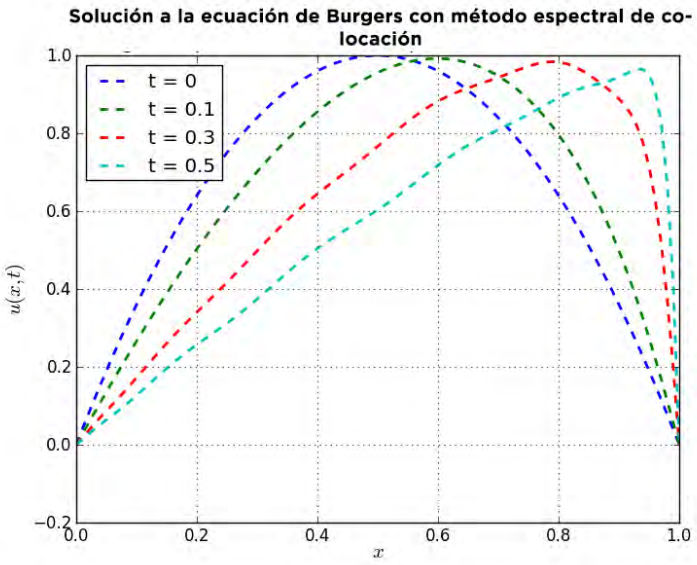

FIGURA 4.2: Perfil de la onda de choque descrita por la ecuación de Burgers unidimensional con $\eta = 0.01$

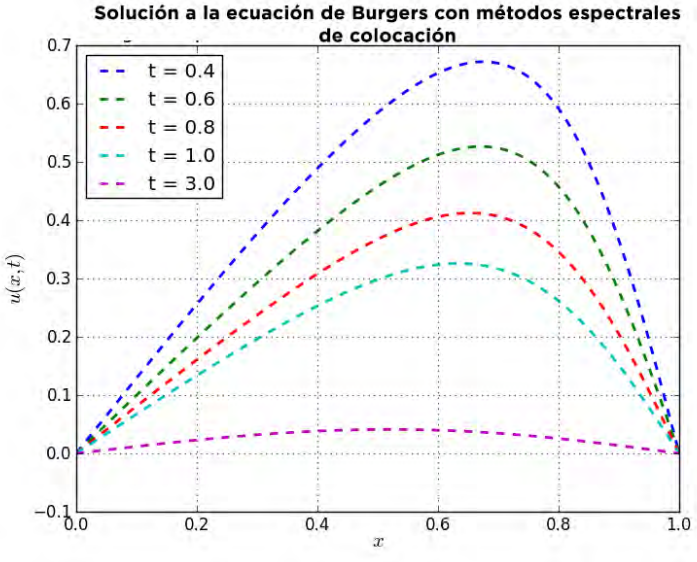

FIGURA 4.3: Perfil de la onda de choque descrita por la ecuación de Burgers unidimensional con $\eta = 0.1$

N	M	Δt	η	$\lvert E\rvert_{t=0.05}$	$\lvert E\rvert_{t=0.25}$
10	10	0.0005	1.0	2.73×10^{-7}	3.20×10^{-7}
10	10	0.0050	0.1	1.43×10^{-6}	6.18×10^{-6}
10	10	0.0100	0.1	6.38×10^{-6}	2.33×10^{-6}
30	30	0.0005	0.01	1.86×10^{-6}	5.27×10^{-4}

TABLA 4.3: Resultados del cómputo del error relativo de la solución numérica a la ecuación de Burgers en 2D

Analizando las figuras mencionadas anteriormente podemos ver como la viscosidad η está relacionada con la atenuación de la onda. Dicho de otra manera, entre mayor sea el valor de η la onda decae más rápidamente.

4.1.2. Ecuación de Burgers en 2D

Así como para el caso undimensional, también se utilizó el método de colocación por Chebyshev para resolver numéricamente el caso de la ecuación de Burgers en 2D 3.28. Los resultados de este método se compararon con la solución analítica [16] siguiente:

$$u(x,y,t) = \frac{1}{1+\exp\left((x+y-t)/(2\eta)\right)}\ , \tag{4.4}$$

definida en una región donde $a < x < b$ y η es el coeficente de viscosidad, que para este caso puede calcularse como $\eta = 1/Re$, donde Re es el número de Reynolds, que es una cantidad adimensional utilizada en la mecánica de fluidos para expresar el cociente entre las fuerzas inerciales y fuerzas viscosas, para poder así caracterizar flujos laminares y flujos turbulentos [8].

Manejando el método para dos dimensiones que se trató en el capítulo anterior podemos obtener un perfil de onda y su evolución en dos dimensiones. Una vez desarrollado el perfil de la onda bidimensional se calculó el error relativo para todos los puntos de la solución numérica contra la solución analítica. Es importante agregar que las condiciones iniciales y de frontera fueron establecidas en base a la solución analítica [16]. Para ello se variaron las dimensiones de la malla, N y M (previamente hemos utilizado únicamente mallas donde $N = M$) así como el paso temporal del método Δt y la viscosidad η. Los resultados se pueden observar en la tabla (4.3).

En al figura 4.4 podemos ver que mientras mayor sea el tamaño de las matrices de diferenciación, menor es el error relativo asociado a nuestra solución. Este resultado

es de esperarse en la mayoría de los métodos espectrales, sin embargo puede darse el caso en el que el error relativo aumente respecto al tamaño de la matriz. Ésto es un indicador de que el método utilizado para el cómputo de las matrices no es el más apropiado e inclusive puede tener problemas de desbordamiento aritmético (también llamado *overflow* aritmético, por su nombre en inglés).

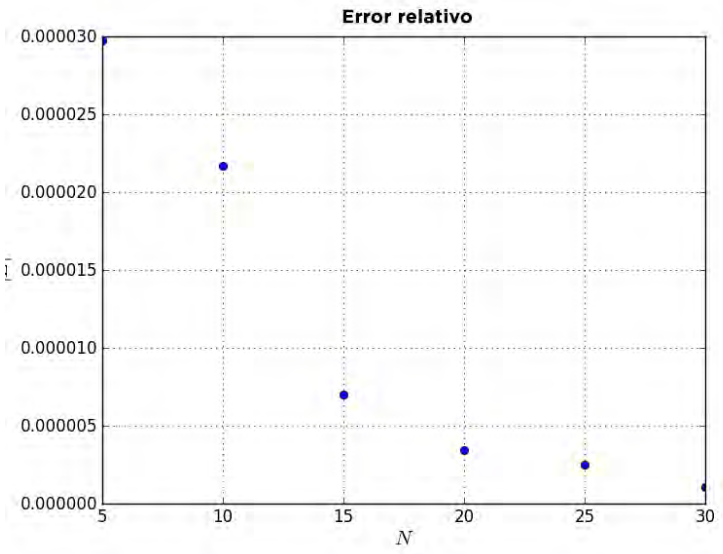

FIGURA 4.4: Error relativo para Burgers en 2D variando el tamaño N (donde $N = M$) de las matrices de diferenciación.

4.1.3. Sistema de ecuaciones de Burgers acopladas en 1D

Se resolvió el sistema de ecuaciones de Burgers acopladas descrito en (3.39) haciendo uso del método espectral basado en polinomios de Chebyshev, así como se vio en el capítulo anterior. Al igual que en los casos anteriores, nuestro objetivo principal fue comparar el perfil y evolución de la onda contra las soluciones analíticas [5].

Las soluciones al sistema son $u(x,t)$ y $v(x,t)$:

$$\begin{aligned} u(x,t) &= a_0 - 2A\left(\frac{2\alpha-1}{4\alpha\beta-1}\right)\tanh[A(x-2At)] \\ v(x,t) &= a_0\left(\frac{2\beta-1}{2\alpha-1}\right) - 2A\left(\frac{2\alpha-1}{4\alpha\beta-1}\right)\tanh[A(x-2At)]\ , \end{aligned} \tag{4.5}$$

donde $a < x < b$, y la constante $A = (1/2)\alpha_0(4\alpha\beta-1)/(2\alpha-1)$, con a_0, α y β, arbitrarios.

N	a	b	Δt	α	β	$\lvert E\rvert_{t=0.5}$	$\lvert E\rvert_{t=1.0}$
10	0	1	0.001	1.0	0.3	2.35×10^{-4}	9.12×10^{-4}
10	-10	10	0.1	0.1	0.3	5.23×10^{-4}	6.82×10^{-3}
20	-10	-10	0.01	0.1	0.3	3.68×10^{-3}	6.21×10^{-3}

TABLA 4.4: Resultados del cómputo del error relativo de la solución numérica al sistema de ecuaciones de Burgers acopladas.

N	a	b	Δt	α	β	$\lvert E\rvert_{t=0.5}$	$\lvert E\rvert_{t=1.0}$
10	0	1	0.001	1.0	0.3	4.63×10^{-4}	1.55×10^{-4}
10	-10	10	0.1	0.1	0.3	1.63×10^{-3}	7.53×10^{-3}
20	-10	-10	0.01	0.1	0.3	6.39×10^{-3}	6.11×10^{-2}

TABLA 4.5: Resultados del cómputo del error relativo de la solución numérica al sistema de ecuaciones de Burgers acopladas.

Después de llevar a cabo nuestra simulación calculamos el error relativo de acuerdo a (3.27). Los resultados se pueden revisar en las tablas (4.4) y (4.5) para $u(x,t)$ y $v(x,t)$, respectivamente [5].

También se llevaron a cabo comparaciones de los perfiles de la onda evaluados en $t = 1$ (4.5), y una visualización en tres dimensiones del perfil de la onda junto con su evolución temporal para $u(x,t)$ y $v(x,t)$ (4.6, 4.7). En estas figuras se puede notar que ambos perfiles para u y v tienen un comportamiento similar.

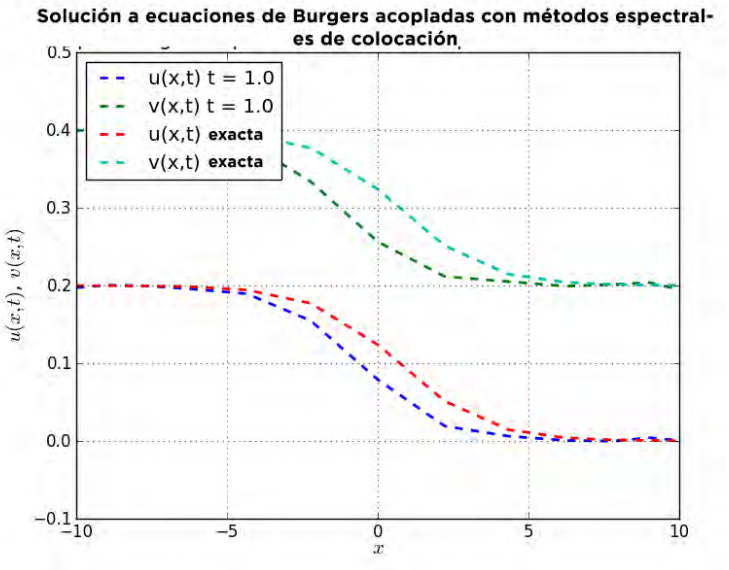

FIGURA 4.5: Comparación de perfiles de soluciones numéricas y analíticas para $u(x,t)$ y $v(x,t)$ evaluadas en $t = 1$ con los valores de $N = 15$, $\alpha = 1$, $\beta = 2$, en el dominio $x \in (-10, 10)$

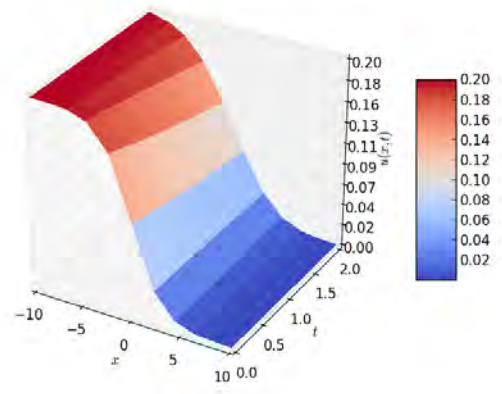

FIGURA 4.6: Perfil superficial de $u(x,t)$ para el sistema de ecuaciones de Burgers con los valores $N = 15$, $\alpha = 1$, $\beta = 2$, en el dominio $x \in (-10, 10)$

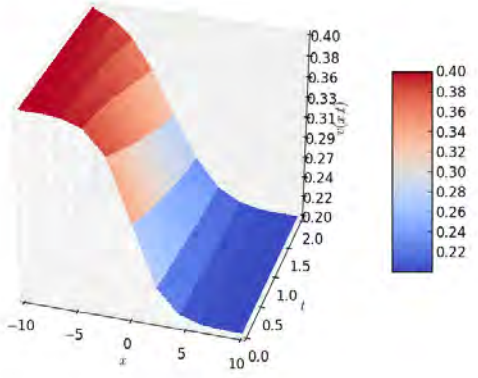

FIGURA 4.7: Perfil superficial de $v(x,t)$ para el sistema de ecuaciones de Burgers con los valores $N = 15$, $\alpha = 1$, $\beta = 2$, en el dominio $x \in (-10, 10)$

Al graficar la respuesta del error relativo con respecto a N podemos notar que, a diferencia de los casos anteriores, conforme el tamaño de la matriz de diferenciación va aumentando el error relativo también aumenta (aunque sea una variación muy pequeña), como se puede ver en la figura 4.8. Este resultado aunque puede considerarse contraintuitivo debido al aumento del error con respecto al tamaño de la matriz, sugiere un problema de desbordamiento aritmético producto de numerosas operaciones matriciales. Más delante se discutirá sobre alternativas para el cálculo matricial utilizado en nuestros algoritmos con la intención de disminuir el error de desbordamiento aritmético encontrado.

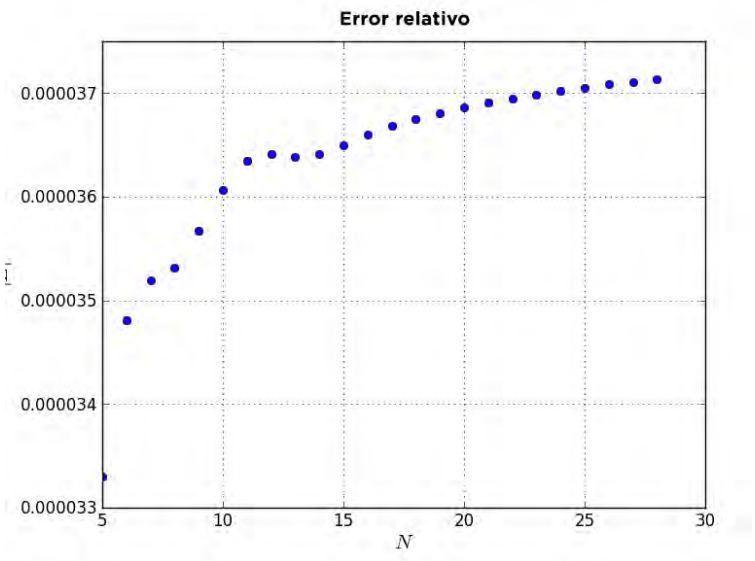

FIGURA 4.8: Error relativo para el sistema de ecuaciones de Burgers acopladas variando el tamaño N en la matriz de diferenciación.

$N = M$	Δt	η	$\lvert E\rvert_{t=0.1}$	$\lvert E\rvert_{t=2.0}$
10	0.0005	0.1	3.17×10^{-5}	9.35×10^{-5}
20	0.0010	0.01	8.34×10^{-6}	8.63×10^{-5}
30	0.0010	0.005	4.35×10^{-5}	1.72×10^{-4}

TABLA 4.6: Error relativo para $u(x,t)$ en el sistema de ecuaciones de Burgers acopladas en 2D

4.1.4. Sistema de ecuaciones de Burgers acopladas en 2D

Se consideró el sistema de ecuaciones de Burgers acopladas en 2D visto en el capítulo anterior (3.49). Este sistema tiene la soluciones analíticas siguientes [17]:

$$\begin{aligned} u(x,t) &= \frac{3}{4} - \frac{1}{4[1+\exp((-4x+4y-t)/32\eta)]}\ , \\ v(x,t) &= \frac{3}{4} + \frac{1}{4[1+\exp((-4x+4y-t)/32\eta)]}\ . \end{aligned} \tag{4.6}$$

Ambas ecuaciones están definidas en el dominio $a < x, y < b$, donde $\eta = 1/R$, y R como el número de Reynolds.

Se calculó el error para algunas combinaciones de N y M, así como el paso temporal y la viscoscidad η. Los resultados de estos cálculos se pueden observar en las tablas 4.6 y 4.7.

$N = M$	Δt	η	$\lvert E\rvert_{t=0.1}$	$\lvert E\rvert_{t=2.0}$
10	0.0005	0.1	2.30×10^{-5}	6.15×10^{-5}
20	0.0010	0.01	6.04×10^{-6}	4.81×10^{-5}
30	0.0010	0.005	3.14×10^{-5}	9.31×10^{-5}

TABLA 4.7: Error relativo para $v(x,t)$ en el sistema de ecuaciones de Burgers acopladas en 2D

Podemos visualizar las diferencias entre las soluciones analíticas y numéricas en la figura 4.9. También se incluyó la figura 4.10 para mostrar el perfil de la solución en 3D para $u(x,y,t)$ y $v(x,y,t)$.

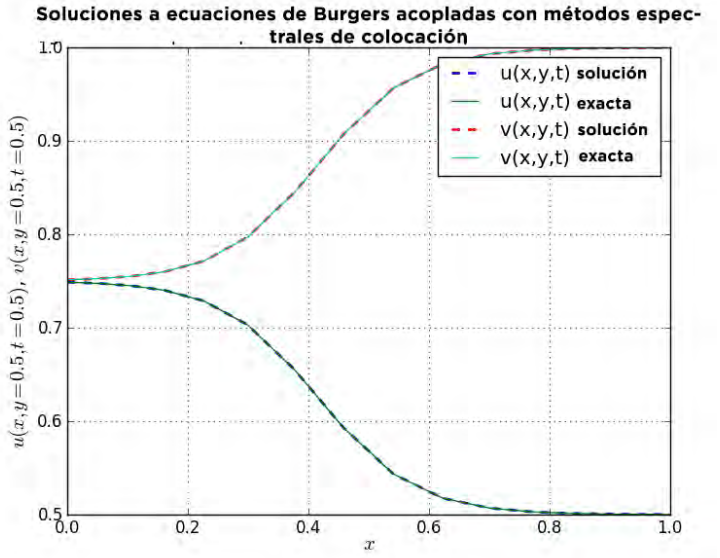

FIGURA 4.9: Perfil de las soluciones analítica y numérica para $u(x,y,t)$ y $v(x,y,t)$ fijadas en $y = 0.5$ y $t = 0.5$.

Además se analizó la tendencia del error relativo respecto al tamaño de la matriz de diferenciación para u y v. Los resultados se pueden observar en las figuras 4.11 y 4.15, respectivamente. Podemos ver que mientras amumenta el tamaño de la matriz el error tiende a disminuir.

Como se mencionó anteriormente, tanto para u como para v se puede como el error disminuye conforme el tamaño de las matrices de diferenciación aumentan, como era de esperarse.

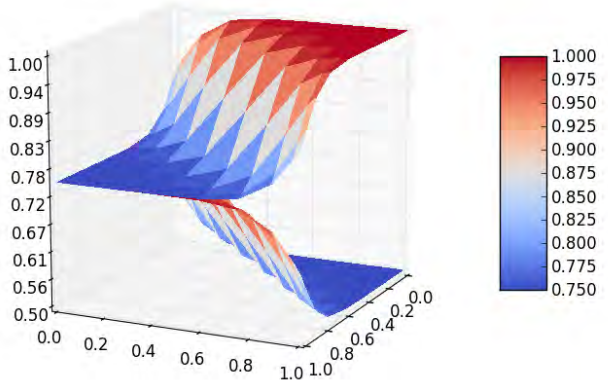

FIGURA 4.10: Perfil de las soluciones $u(x, y, t)$ y $v(x, y, t)$ con $y = 0.5$.

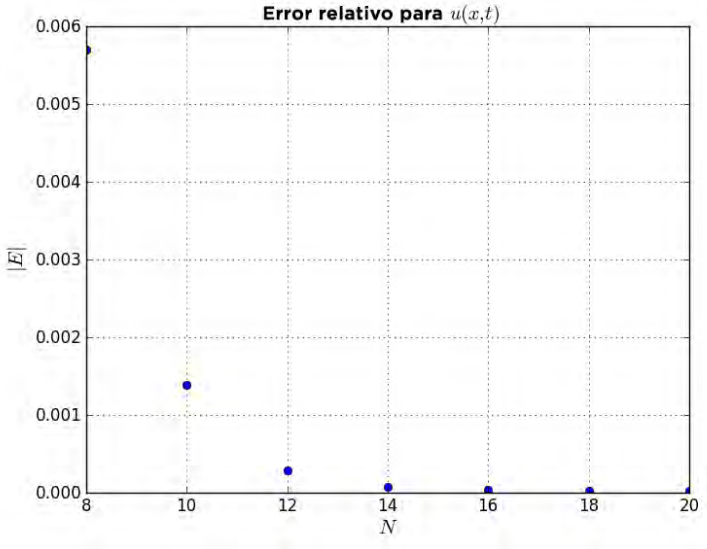

FIGURA 4.11: Error relativo contra N para $u(x, y, t)$ para el caso de la solución al sistema de ecuaciones de Burgers en 2D.

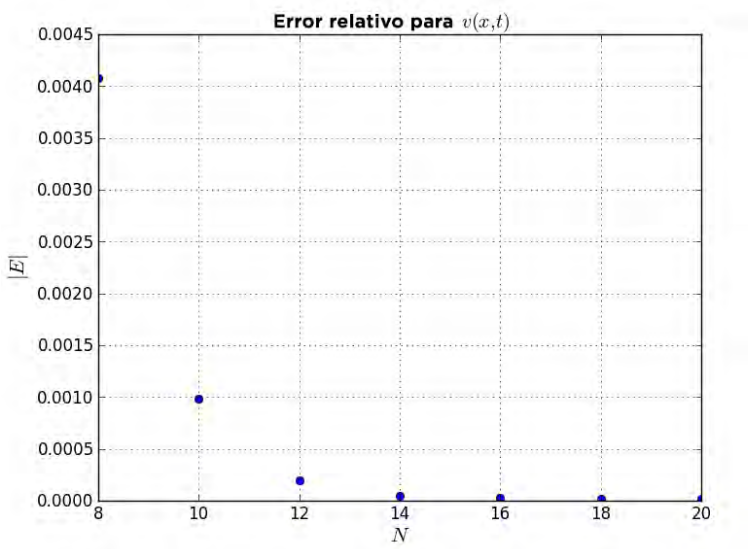

FIGURA 4.12: Error relativo contra N para $v(x, y, t)$ para el caso de la solución al sistema de ecuaciones de Burgers en 2D.

N	a	b	Δt	α	β	$\lvert E\rvert_{t=0.5}$	$\lvert E\rvert_{t=1.0}$
10	0	1	0.001	1.0	0.3	2.35×10^{-4}	9.12×10^{-4}
10	-10	10	0.1	0.1	0.3	5.23×10^{-5}	6.82×10^{-5}
20	-10	-10	0.01	0.1	0.3	3.68×10^{-5}	6.21×10^{-5}

TABLA 4.8: Resultados del cómputo del error relativo de la solución numérica al sistema de ecuaciones de Burgers acopladas.

4.1.5. Ecuación Korteweg de Vries - Burgers (KdV-Burgers)

La ecuación KdV-Burgers 3.7 vista en el capítulo anterior tiene la solución analítica siguiente [12]

$$u(x,t) = A\big[1 + \tanh(A(x - ct))\big] \ , \tag{4.7}$$

definida en la región $a < x < b$, donde las constantes A y c toman los valores $A = 1/6$ y $c = 2/9$.

Los resultados del cálculo del error relativo entre la solución analítica mostrada anteriormente y la solución numérica producto de nuestro método numérico se pueden observar en la tabla 4.8.

También se realizaron visualizaciones gráficas de la evolución de la onda para $N = 10$ (figura 4.13) y $N = 15$ (figura 4.14). Los valores de N fueron escogidos pensando en buscar si había una ganancia notable en la disminución del error relativo sin llevar un impacto en el tiempo de cómputo. Como podemos observar, el tamaño de la matriz no tiene un impacto significativo en el perfil de la onda para este orden de magnitud.

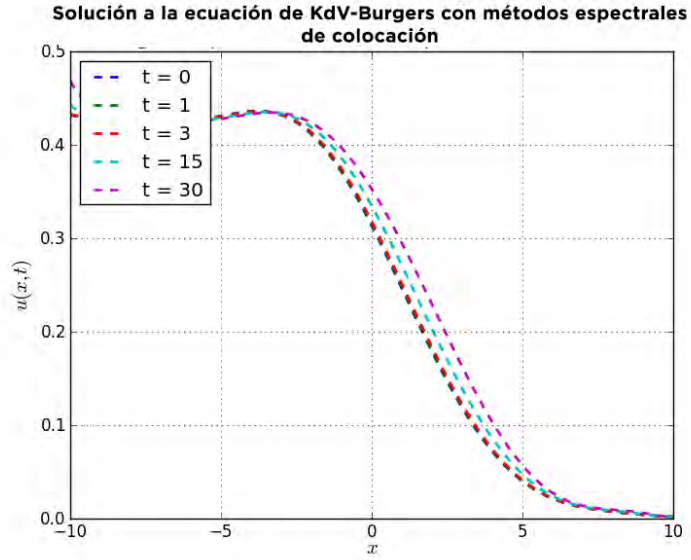

FIGURA 4.13: Evolución de la onda descrita por la ecuación de KdV-Burgers para $N = 10$ y $\Delta t = 0.001$.

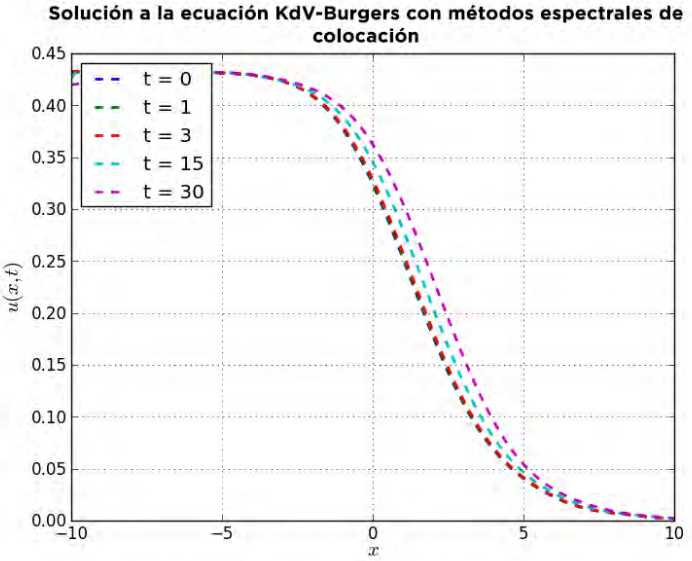

FIGURA 4.14: Evolución de la onda descrita por la ecuación de KdV-Burgers para $N = 15$ y $\Delta t = 0.001$.

Asimismo, se analizó la relación entre el error relativo $|E|$ y el tamaño N de la matriz de diferenciación.

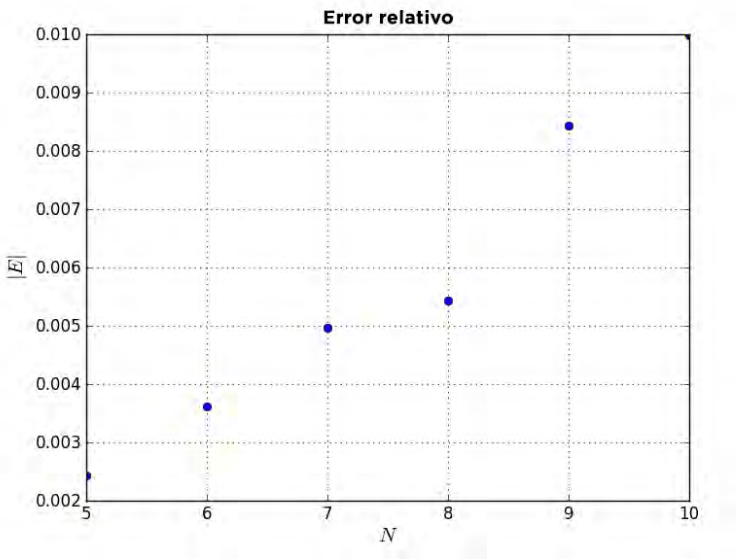

FIGURA 4.15: Error relativo contra N para la solución de la ecuación de KdV-Burgers en 1D

Podemos observar que, a diferencia de la mayoría de los casos analizados anteriormente, el error tiende a aumentar con respecto al tamaño de la matriz. Esto puede ser debido a problemas del cómputo de la matriz de diferenciación. Algunas de las operaciones llevadas a cabo hacen que los valores de las variables excedan los límites permitidos por la unidad de procesamiento, resultando en *overflow* aritmético. Sin embargo, es importante notar que el error relativo calculado sigue siendo de un orden muy pequeño (10^{-3}).

Capítulo 5

Conclusiones

Las EPD no lineles suelen requerir de métodos numéricos para poderlas resolver. Los métodos espectrales ofrecen una manera de resolverlas con gran precisión, pero sólo se les puede emplear bajo ciertas condiciones. Para poder describir el fenómeno de ondas de choque se puede usar el modelo de la ecuación de Burgers, que permite modelar los choques no como discontinuidades de la función, sino como variaciones abruptas con curvas bien definidas. Ésto permite utilizar métodos espectrales.

Las mayoría de las EDP no lineales no suelen tener soluciones analíticas tales como las estudiadas en este texto, y el número de ecuaciones no lineales que cuentan con soluciones analíticas generales es muy pequeño. Incluso en el caso de la ecuación KdV, que entra dentro de esta categoría, la mayor parte de las soluciones no se pueden expresar facilmente de forma analítica. Es por esta razón que resulta de interés resolver estas ecuaciones haciendo uso de una herramienta numérica de alta precisión. Tomando en cuenta lo anterior junto con el alto desarrollo en el desempeño computacional de ordenadores de uso personal en los últimos años, se escogió utilizar métodos espectrales para el análisis.

Podemos ver que para cada caso analizado - ecuaciones de Burgers en 1D y 2D, sistema de ecuaciones de Burgers acopladas en 1D y 2D y KdV Burgers en 1D - el error relativo comparado con las soluciones analíticas es de un orden de magnitud bajo (10^{-4}) y más preciso que los métodos de diferencias finitas utilizados en otros estudios [14] [15]. Es esencial agregar que aunque al aplicar métodos espectrales en estas ecuaciones obtengamos un error mucho menor que al resolverlas con métodos de diferencias finitas, esto no significa que siempre arrojarán mejores resultados, ya que, como se mencionó anteriormente, los métodos espectrales funcionan bajo condiciones muy específicas.

Analizando la relación entre el error relativo y el tamaño de la matriz de diferenciación obtuvimos resultados no esperados para los casos del sistema de ecuaciones de Burgers acopladas en 1D y KdV-Burgers, donde conforme el tamaño de la matriz aumenta también aumenta el error relativo. Es importante agregar que para estos casos muchas operaciones relacionadas con la matriz empezaron a tener problemas de desbordamiento de aritmética. Esto significa que las variables asignadas para cada valor de la matriz pueden superar el tamaño máximo que se les había especificado anteriormente y en consecuencia los resultados arrojados tienden a cargar un error. Otra explicación posible para entender por qué aparece este tipo de error es debido a que los polinimios utilizados para describir su diferenciación tengan un comportamiento asintótico.

El método espectral utilizado se muestra particularmente útil en problemas donde la precisión es importante. Su implementación no es tan directa como para los métodos de diferencias finitas lo cual es claramente una desventaja en casos donde el tiempo de implementación es escencial, sin embargo, siempre que se cumplen los requisitos para su manejo, dichos métodos arrojan resultados muy cercanos a las soluciones analíticas. Su poca versatilidad al emplearlos tal vez sea su principal consideración al decidir emplearlos.

5.1. Trabajo a futuro

Considerando el tema de los casos con desbordamiento aritmético en las operaciones matriciales, se piensa en tratar de resolver este problema haciendo uso de librerías de aritmética de números grandes, normalmente nativas del lenguaje de programación *C*, pero bien pueden ser manejadas con un lenguaje de alto nivel como Julia [18], orientado precisamente a resolver problemas de cómputo científico. Comparar los resultados usando este tipo de librerías con los obtenidos en este estudio podría ser muy útil para nuestros estudios futuros.

Otro estudio que podría arrojar más luz sobre el alcance y precisión de los métodos espectrales es el comparar nuestros resultados con los obtenidos con métodos espectrales de otro tipo, por ejemplo, usando métodos para obtener derivadas basados en la transformada rápida de Fourier o manejando otro conjunto de polinomios como funciones base (B-splines o polinomios de Legendre). Habría que llevar a cabo también una comparación entre los tiempos de ejecución dependiendo del método utilizado, pues en muchos casos esto puede influir en la decisión al emplearlos.

Al igual que la ecuación de Burgers y la ecuación Korteweg-de Vries-Burgers, existen muchas EDP no lineales que son usadas con frecuencia en los campos de la acústica y

mecánica de fluidos. La ecuación Benjamin-Ono que es una EDP no lineal utilizada para realizar modelos de ondas internas en aguas profundas [19], sería un buen ejemplo, ya que también se conoce una solución analítica con la cual se pueden hacer cálculos de error relativo.

Por otro lado, es particularmente interesante analizar una EDP no lineal en la cual se desconozca una solución analítica. Esta ecuación tendría que ser el producto de un modelo matemático para un fenómeno físico, lo cual implicaría llevar a cabo un estudio más profundo de la naturaleza empírica del fenómeno y de la ecuación en sí. En la solución de la ecuación bajo métodos espectrales tendría que haber un estudio del error relativo basado enteramente en observaciones empíricas del fenómeno, lo que posiblemente traiga en consideración un análisis estadístico, así como un manejo estocástico de la ecuación. Un caso a estudiar podría ser las ecuaciones para la descripción de solitoners como las ecuaciones senoidales de Gordon o ecuaciones de aguas superficiales. Dichas ecuaciones tienen algunas soluciones analíticas, pero bien también se pueden realizar experimentos para describir sus soluciones empíricamente y comparar los resultados con los otorgados con las soluciones numéricas.

Apéndice A

Código

A.1. chebPy.py

```
import numpy as np

def cheb(N, a = -1, b = 1):
    '''Chebushev polynomial differentiation matrix.
       Ref.: Trefethen's 'Spectral Methods in MATLAB' book.
    '''
    N = N-1
    x = 0.5*((a+b)-(b-a)*np.cos(np.pi*np.array(range(0,N+1))/N))
    c=np.zeros(N+1)
    c[0]=2.
    c[1:N]=1.
    c[N]=2.
    c = c * (-1)**np.linspace(0,N,N+1)
    X = np.tile(x, (N+1,1))
    dX = X - X.T
    D = np.dot(c.reshape(N+1,1),(1./c).reshape(1,N+1))
    D = D / (dX+np.eye(N+1))
    D = D - np.diag( D.T.sum(axis=0) )
    return D,x
```

A.2. burgers-1D.py

```
# -*- coding: utf-8 -*-

from __future__ import division
import numpy as np
from chebPy import *
from matplotlib import pyplot as plt
from scipy.integrate import odeint
import time
import datetime
```

```
# Interval
a, b = 0, 1

# Burger equations constants
alpha = 0.1
v = 0.01
c = 0.1

# Time to evaluate
t_eval = 0.3

# Matrix size
N = 10

# Derivative matrix based on Chebyshev polynomials
D, x = cheb(N,a,b)
D2   = np.dot(D,D)

u0 = c/alpha + (2*v)/alpha*np.tanh(x)

# Time
t_start = 0
t_end   = 3.5
delta_t = 0.01          # For Runge-Kutta

t_num   = (t_end - t_start)/delta_t
t               = np.linspace(t_start, t_end, t_num)

def F(u, t, alpha, v, D, D2, N):
        res = []
        for i in range(1, N-1) :
                d     =  D[i,0]*u[0] +   D[i,-1]*u[-1]
                d_bar = D2[i,0]*u[0] +  D2[i,-1]*u[-1]
                this = alpha * u[i] * np.sum(D[i,:][1:N-1] * u[1:N-1]) + v * np.
    sum(D2[i,:][1:N-1] * u[1:N-1]) + alpha * u[i] * d + v * d_bar
                res = np.append(res, this)
        res = np.append(0, res)
        res = np.append(res, 0)
        return res

u_sol = np.zeros((t_num, N))
u_sol[0, :] =  u0

for t_n in range(0, np.int(t_num - 1) ) :
        u_sol[t_n, :][0]  = c/alpha + (2*v)/alpha*np.tanh(a - c*(t_start + t_n*
    delta_t))
        u_sol[t_n, :][-1] = c/alpha + (2*v)/alpha*np.tanh(b - c*(t_start + t_n*
    delta_t))

        u_n = u_sol[t_n, :]
        k1 = F(u_n, t_n, alpha, v, D, D2 , N)
        k2 = F(u_n + k1*delta_t/2, t_n + 1/2*delta_t, alpha, v, D, D2, N)
        k3 = F(u_n + k2*delta_t/2, t_n + 1/2*delta_t, alpha, v, D, D2, N)
        k4 = F(u_n + k3*delta_t,   t_n + delta_t, alpha, v, D, D2, N)
```

```
        u_sol[t_n + 1, :] = u_n + 1/6*delta_t*(k1 + 2*k2 + 2*k3 + k4)

print "The solution has {0} elements in time, N = {1} and delta_t = {2}".format(
    u_sol.shape[0], u_sol.shape[1], delta_t)

xx = np.linspace(a, b, 301)
uu = np.polyval(np.polyfit(x, u0, N), xx)

t_to_eval = [0.1, 0.2, 0.25]
x_eval = 0.25

def error_norm_inf(u_compu, u_exact):
        return np.max(np.abs(u_compu - u_exact))

def error_norm_rel(u_compu, u_exact):
        up   = np.sum((u_compu - u_exact)**2)
        down = np.sum((u_exact)**2)
        return np.sqrt(up/down)

t_n_eval = (t_eval - t_start)/delta_t
t = t_eval
u_exact = c/alpha + (2*v)/alpha*np.tanh(x - c*t)

norm_rel  = error_norm_rel(u_sol[t_n_eval,:][1:N-1], u_exact[1:N-1])

print "alpha = {0}, Eta = {1}, t = {2}, N = {3} and delta_t = {4}".format(alpha,
    v, t_eval, N, delta_t)
print "The norm of relative errors is {1}".format(norm_rel)
```

A.3. burgers-2D.py

```
# -*- coding: utf-8 -*-

from __future__ import division
import numpy as np
from chebPy import *
from matplotlib import pyplot as plt

filename = "burgers_2D_collocation"

a, b = 0, 1.0

v = 0.1
delta_t = 0.0005
t_eval = 0.25
y_eval = 0.25

N = 10
M = 10
```

```
D_x, x = cheb(N,a,b)
D_y, y = cheb(M,a,b)
D_x2   = np.dot(D_x,D_x)
D_y2   = np.dot(D_y,D_y)

# Intial condition u0 = u(x,y,0)
u0 = 1.0 / (1.0 + np.exp((x + y)/(2*v)))

# Time
t_start = 0
t_end   = 0.5

t_num   = (t_end - t_start)/delta_t
t               = np.linspace(t_start, t_end, t_num)

# Using Runge-Kutta 4 manually
def F(u, t, v, D_x, D_x2, D_y, D_y2, N, M):
        res = np.zeros((N, M))
        for i in range(1, N-1) :
                for j in range(1, M-1) :
                        z_ij            =  D_x[i,0]*u[0,j] +  D_x[i,-1]*u[-1,j]
                        z_ij_bar        = D_x2[i,0]*u[0,j] + D_x2[i,-1]*u[-1,j]
                        q_ij            =  D_y[j,0]*u[i,0] +  D_y[j,-1]*u[i,-1]
                        q_ij_bar        = D_y2[j,0]*u[i,0] + D_y2[j,-1]*u[i,-1]

                        this = -u[i,j]*(z_ij + np.sum(D_x[i,:][1:N-1]*u[1:N-1, j
    ])) - u[i,j]*(q_ij + np.sum(D_y[j,:][1:M-1]*u[i,1:M-1])) + v*(z_ij_bar + np.
    sum(D_x2[i,:][1:N-1]*u[1:N-1, j])) + v*(q_ij_bar + np.sum(D_y2[j,:][1:M-1]*u[
    i,1:M-1]))
                        res[i,j] = this

        return res

u_sol = np.zeros((t_num, N, M))
u_sol[0, 0, :] =  u0

for t_n in range(0, np.int(t_num - 1) ) :
        u_sol[t_n, :][0]  = 1.0 / (1.0 + np.exp((a + a - t_n*delta_t)/(2*v)))
        u_sol[t_n, :][-1] = 1.0 / (1.0 + np.exp((b + b - t_n*delta_t)/(2*v)))

        u_n = u_sol[t_n, :]
        k1 = F(u_n, t_n, v, D_x, D_x2, D_y, D_y2, N, M)
        k2 = F(u_n + k1*delta_t/2, t_n + 1/2*delta_t, v, D_x, D_x2, D_y, D_y2, N,
     M)
        k3 = F(u_n + k2*delta_t/2, t_n + 1/2*delta_t, v, D_x, D_x2, D_y, D_y2, N,
     M)
        k4 = F(u_n + k3*delta_t,   t_n + delta_t, v, D_x, D_x2, D_y, D_y2, N, M)

        u_sol[t_n + 1, :] = u_n + 1/6*delta_t*(k1 + 2*k2 + 2*k3 + k4)

def error_norm_inf(u_compu, u_exact):
        return np.max(np.abs(u_compu - u_exact))

def error_norm_rel(u_compu, u_exact):
        up   = np.sum((u_compu - u_exact)**2)
```

```
        down = np.sum((u_exact)**2)
        return np.sqrt(up/down)

t_n_eval = (t_eval - t_start)/delta_t
t = t_eval
u_exact = 1.0 / (1.0 + np.exp((x + y - t)/(2*v)))

print u_sol[t_n_eval,:]
print u_exact

norm_inf  = error_norm_inf(u_sol[t_n_eval,:][1:N-1], u_exact[1:N-1])
norm_rel  = error_norm_rel(u_sol[t_n_eval,:][1:N-1], u_exact[1:N-1])

print "eta = {0}, y = {1}, t = {2}, N = {3}, M = {4} and delta_t = {5}".format(v,
     y_eval, t_eval, N, M, delta_t)
print "The norm infinity is {0}, while the norm of relative errors is {1}".format
    (norm_inf, norm_rel)
```

A.4. burgers-1D-coulpled.py

```
# -*- coding: utf-8 -*-

from __future__ import division
import numpy as np
from chebPy import *
from matplotlib import pyplot as plt
from scipy.integrate import odeint

# Interval
# a, b = -10, 10
a, b = -10, 10

# Burger equations constants
alpha = 1.0
beta  = 0.3

# Constants
a_0 = 0.05
A = 0.5*(a_0*(4*alpha*beta - 1))/(2*alpha - 1)

# Matrix size
N = 20

# Derivative matrix based on Chebyshev polynomials
D, x = cheb(N,a,b)
D2   = np.dot(D,D)

# Intial condition u0 = u(x,0) | ux0 = u_x(x,0) | v0 = v(x,0) | vx0 = v_x(x,0)
u0 = a_0 - 2*A*((2*alpha - 1)/(4*alpha*beta - 1))*np.tanh(A*(x))
v0 = a_0*((2*beta - 1)/(2*alpha - 1)) - 2*A*((2*alpha - 1)/(4*alpha*beta - 1))*np
    .tanh(A*(x))
```

```
ux0 = -2*A**2*((2*alpha - 1)/(4*alpha*beta - 1))*(1/np.cosh(A*x))**2
vx0 = ux0

# Time
t_start = 0
t_end   = 2
delta_t = 0.01          # For Runge-Kutta

t_num   = (t_end - t_start)/delta_t
t               = np.linspace(t_start, t_end, t_num)

# Diferential function for Runge-Kutta method
def F(u, v, t, alpha, beta, D, D2):
        res = []

        for i in range(1, N-1) :
                d     =  D[i,0]*u[0] +   D[i,-1]*u[-1]
                d_bar = D2[i,0]*u[0] +  D2[i,-1]*u[-1]
                r     =  D[i,0]*v[0] +   D[i,-1]*v[-1]
                r_bar = D2[i,0]*v[0] +  D2[i,-1]*v[-1]
                this = np.sum(D2[i,:][1:N-1] * u[1:N-1]) - 2 * u[i] * np.sum(D[i
    ,:][1:N-1] * u[1:N-1]) - alpha * u[i] * np.sum(D[i,:][1:N-1]*v[1:N-1]) -
    alpha * v[i] * np.sum(D[i,:][1:N-1] * u[1:N-1]) + d_bar - 2*u[i]*d - alpha*u[
    i]*r - alpha*v[i]*d
                res = np.append(res, this)
        res = np.append(0, res)
        res = np.append(res, 0)
        return res

def G(u, v, t, alpha, beta, D, D2):
        res = []

        for i in range(1, N-1) :
                d     =  D[i,0]*u[0] +   D[i,-1]*u[-1]
                d_bar = D2[i,0]*u[0] +  D2[i,-1]*u[-1]
                r     =  D[i,0]*v[0] +   D[i,-1]*v[-1]
                r_bar = D2[i,0]*v[0] +  D2[i,-1]*v[-1]
                this = np.sum(D2[i,:][1:N-1] * v[1:N-1]) - 2 * v[i] * np.sum(D[i
    ,:][1:N-1] * v[1:N-1]) - beta * u[i] * np.sum(D[i,:][1:N-1]*v[1:N-1]) - beta
    * v[i] * np.sum(D[i,:][1:N-1] * u[1:N-1]) + r_bar - 2*v[i]*r - beta*u[i]*r -
    beta*v[i]*d
                res = np.append(res, this)
        res = np.append(0, res)
        res = np.append(res, 0)
        return res

u_sol = np.zeros((t_num, N))
u_sol[0, :] =  u0

v_sol = np.zeros((t_num, N))
v_sol[0, :] =  v0

for t_n in range(0, np.int(t_num - 1) ) :
```

```
        u_sol[t_n, :][0]   = a_0 - 2*A*((2*alpha - 1)/(4*alpha*beta - 1))*np.tanh
    (A*(a - 2*A*(t_start + t_n*delta_t)))
        u_sol[t_n, :][-1]  = a_0 - 2*A*((2*alpha - 1)/(4*alpha*beta - 1))*np.tanh
    (A*(b - 2*A*(t_start + t_n*delta_t)))

        v_sol[t_n, :][0]   = a_0*((2*beta - 1)/(2*alpha - 1)) - 2*A*((2*alpha -
    1)/(4*alpha*beta - 1))*np.tanh(A*(a - 2*A*(t_start + t_n*delta_t)))
        v_sol[t_n, :][-1]  = a_0*((2*beta - 1)/(2*alpha - 1)) - 2*A*((2*alpha -
    1)/(4*alpha*beta - 1))*np.tanh(A*(b - 2*A*(t_start + t_n*delta_t)))

        u_n = u_sol[t_n, :]
        v_n = v_sol[t_n, :]

        k1 = F(u_n, v_n, t_n, alpha, beta, D, D2)
        l1 = G(u_n, v_n, t_n, alpha, beta, D, D2)

        k2 = F(u_n + k1*delta_t/2, v_n + l1*delta_t/2, t_n + 1/2*delta_t, alpha,
    beta, D, D2)
        l2 = G(u_n + k1*delta_t/2, v_n + l1*delta_t/2, t_n + 1/2*delta_t, alpha,
    beta, D, D2)

        k3 = F(u_n + k2*delta_t/2, v_n + l2*delta_t/2, t_n + 1/2*delta_t, alpha,
    beta, D, D2)
        l3 = G(u_n + k2*delta_t/2, v_n + l2*delta_t/2, t_n + 1/2*delta_t, alpha,
    beta, D, D2)

        k4 = F(u_n + k3*delta_t, v_n + l3*delta_t, t_n + delta_t, alpha, beta, D,
     D2)
        l4 = F(u_n + k3*delta_t, v_n + l3*delta_t, t_n + delta_t, alpha, beta, D,
     D2)

        u_sol[t_n + 1, :] = u_n + 1/6*delta_t*(k1 + 2*k2 + 2*k3 + k4)
        v_sol[t_n + 1, :] = v_n + 1/6*delta_t*(l1 + 2*l2 + 2*l3 + l4)

def error_norm_inf(u_compu, u_exact):
        return np.max(np.abs(u_compu - u_exact))

def error_norm_rel(u_compu, u_exact):
        up   = np.sum((u_compu - u_exact)**2)
        down = np.sum((u_exact)**2)
        return np.sqrt(up/down)

t_eval = 0.5
t_n_eval = (t_eval - t_start)/delta_t
t = t_eval
v_exact = a_0*((2*beta - 1)/(2*alpha - 1)) - 2*A*((2*alpha - 1)/(4*alpha*beta -
    1))*np.tanh(A*(x - 2*A*t))

norm_inf  = error_norm_inf(v_sol[t_n_eval,:], v_exact)
norm_rel  = error_norm_rel(v_sol[t_n_eval,:], v_exact)

print "a = {0}, b = {1}, delta_t = {2}, alpha = {3}, beta = {4}, N = {5} and t =
    {6}".format(a, b, delta_t, alpha, beta, N, t_eval)
print "The norm infinity is {0}, while the norm of relative errors is {1}".format
    (norm_inf, norm_rel)
```

A.5. burgers-2D-coupled.py

```
# -*- coding: utf-8 -*-

from __future__ import division
import numpy as np
from chebPy import *
from matplotlib import pyplot as plt
from scipy.integrate import odeint

a, b = 0, 1.0
eta = 0.005
delta_t = 0.001
t_eval = 2.0
y_eval = 0.5

N = 20
M = 20

D_x, x = cheb(N,a,b)
D_y, y = cheb(M,a,b)
D_x2   = np.dot(D_x,D_x)
D_y2   = np.dot(D_y,D_y)

# Intial condition u0 = u(x,y,0)
u0 = 3.0/4 - 1 / (4*(1 + np.exp((-4*x + 4*y )/(32*eta))))
v0 = 3.0/4 + 1 / (4*(1 + np.exp((-4*x + 4*y )/(32*eta))))

# Time
t_start = 0
t_end   = 2.01

t_num   = (t_end - t_start)/delta_t
t               = np.linspace(t_start, t_end, t_num)

# Using Runge-Kutta 4 manually
def F(u, v, t, eta, D_x, D_x2, D_y, D_y2, N, M):
        res = np.zeros((N, M))
        for i in range(1, N-1) :
                for j in range(1, M-1) :
                        z_ij            =  D_x[i,0]*u[0,j] +  D_x[i,-1]*u[-1,j]
                        z_ij_bar        = D_x2[i,0]*u[0,j] + D_x2[i,-1]*u[-1,j]
                        q_ij            =  D_y[j,0]*u[i,0] +  D_y[j,-1]*u[i,-1]
                        q_ij_bar        = D_y2[j,0]*u[i,0] + D_y2[j,-1]*u[i,-1]

                        this =  u[i,j]*(z_ij + np.sum(D_x[i,:][1:N-1]*u[1:N-1, j
    ])) + v[i,j]*(q_ij + np.sum(D_y[j,:][1:M-1]*u[i,1:M-1])) + eta*(z_ij_bar + np
    .sum(D_x2[i,:][1:N-1]*u[1:N-1, j])) + eta*(q_ij_bar + np.sum(D_y2[j,:][1:M
    -1]*u[i,1:M-1]))

                        res[i,j] = this

        return res

def G(u, v, t, eta, D_x, D_x2, D_y, D_y2, N, M):
```

```
        res = np.zeros((N, M))
        for i in range(1, N-1) :
                for j in range(1, M-1) :
                        w_ij            =  D_x[i,0]*v[0,j] +  D_x[i,-1]*v[-1,j]
                        w_ij_bar        = D_x2[i,0]*v[0,j] + D_x2[i,-1]*v[-1,j]
                        k_ij            =  D_y[j,0]*v[i,0] +  D_y[j,-1]*v[i,-1]
                        k_ij_bar        = D_y2[j,0]*v[i,0] + D_y2[j,-1]*v[i,-1]

                        this = u[i,j]*(w_ij + np.sum(D_x[i,:][1:N-1]*v[1:N-1, j])
    ) + v[i,j]*(k_ij + np.sum(D_y[j,:][1:M-1]*v[i,1:M-1])) + eta*(w_ij_bar + np.
    sum(D_x2[i,:][1:N-1]*v[1:N-1, j])) + eta*(k_ij_bar + np.sum(D_y2[j,:][1:M-1]*
    v[i,1:M-1]))
                        res[i,j] = this

        return res

u_sol = np.zeros((t_num, N, M))
v_sol = np.zeros((t_num, N, M))


for x_i in range(0, N - 1) :
        for y_i in range(0, M - 1) :
                u_sol[0, x_i, y_i] = 3.0/4 - 1 / (4*(1 + np.exp((-4*x[x_i] + 4*y[
    y_i] )/(32*eta))))
                v_sol[0, x_i, y_i] = 3.0/4 + 1 / (4*(1 + np.exp((-4*x[x_i] + 4*y[
    y_i] )/(32*eta))))

for t_n in range(0, np.int(t_num - 1) ) :
        u_sol[t_n,  0,  :]  = 3/4 - 1 / (4*(1 + np.exp((-4*a + 4*y - t_n*delta_t)
    /(32*eta))))
        u_sol[t_n, -1,  :]  = 3/4 - 1 / (4*(1 + np.exp((-4*b + 4*y - t_n*delta_t)
    /(32*eta))))

        u_sol[t_n,  :,  0]  = 3/4 - 1 / (4*(1 + np.exp((-4*x + 4*a - t_n*delta_t)
    /(32*eta))))
        u_sol[t_n,  :, -1]  = 3/4 - 1 / (4*(1 + np.exp((-4*x + 4*b - t_n*delta_t)
    /(32*eta))))

        v_sol[t_n,  0,  :]  = 3/4 + 1 / (4*(1 + np.exp((-4*a + 4*y - t_n*delta_t)
    /(32*eta))))
        v_sol[t_n, -1,  :]  = 3/4 + 1 / (4*(1 + np.exp((-4*b + 4*y - t_n*delta_t)
    /(32*eta))))

        v_sol[t_n,  :,  0]  = 3/4 + 1 / (4*(1 + np.exp((-4*x + 4*a - t_n*delta_t)
    /(32*eta))))
        v_sol[t_n,  :, -1]  = 3/4 + 1 / (4*(1 + np.exp((-4*x + 4*b - t_n*delta_t)
    /(32*eta))))

        u_n = u_sol[t_n, :]
        v_n = v_sol[t_n, :]

        k1 = F(u_n, v_n, t_n, eta, D_x, D_x2, D_y, D_y2, N, M)
        l1 = G(u_n, v_n, t_n, eta, D_x, D_x2, D_y, D_y2, N, M)
```

```
        k2 = F(u_n + k1*delta_t/2, v_n + l1*delta_t/2, t_n + 0.5*delta_t, eta,
    D_x, D_x2, D_y, D_y2, N, M)
        l2 = G(u_n + k1*delta_t/2, v_n + l1*delta_t/2, t_n + 0.5*delta_t, eta,
    D_x, D_x2, D_y, D_y2, N, M)

        k3 = F(u_n + k2*delta_t/2, v_n + l2*delta_t/2, t_n + 0.5*delta_t, eta,
    D_x, D_x2, D_y, D_y2, N, M)
        l3 = G(u_n + k2*delta_t/2, v_n + l2*delta_t/2, t_n + 0.5*delta_t, eta,
    D_x, D_x2, D_y, D_y2, N, M)

        k4 = F(u_n + k3*delta_t, v_n + l3*delta_t, t_n + delta_t, eta, D_x, D_x2,
     D_y, D_y2, N, M)
        l4 = G(u_n + k3*delta_t, v_n + l3*delta_t, t_n + delta_t, eta, D_x, D_x2,
     D_y, D_y2, N, M)

        u_sol[t_n + 1, :] = u_n + 1/6*delta_t*(k1 + 2*k2 + 2*k3 + k4)
        v_sol[t_n + 1, :] = v_n + 1/6*delta_t*(l1 + 2*l2 + 2*l3 + l4)


xx = np.linspace(a, b, 301)
yy = np.linspace(a, b, 301)

t_to_eval = [0.5]

def error_norm_inf(u_compu, u_exact):
        return np.max(np.abs(u_compu - u_exact))

def error_norm_rel(u_compu, u_exact):
        up   = np.sum((u_compu - u_exact)**2)
        down = np.sum((u_exact)**2)
        return np.sqrt(up/down)

t_n_eval = (t_eval - t_start)/delta_t
t = t_eval
y_n_eval = N/2 #0.5
u_exact = 3/4 - 1 / (4*(1 + np.exp((-4*x + 4*y[y_n_eval] - t)/(32*eta))))
v_exact = 3/4 + 1 / (4*(1 + np.exp((-4*x + 4*y[y_n_eval] - t)/(32*eta))))

# Error calculation
norm_inf_u  = error_norm_inf(u_sol[t_n_eval,:, y_n_eval], u_exact[:])
norm_rel_u  = error_norm_rel(u_sol[t_n_eval,:, y_n_eval], u_exact[:])

norm_inf_v  = error_norm_inf(v_sol[t_n_eval,:, y_n_eval], v_exact[:])
norm_rel_v  = error_norm_rel(v_sol[t_n_eval,:, y_n_eval], v_exact[:])

print "eta = {0}, y = {1}, t = {2}, N = {3}, M = {4} and delta_t = {5}".format(
    eta, y_eval, t_eval, N, M, delta_t)
print "The norm infinity for u(x,y,t) is {0}, while the norm of relative errors
    is {1}".format(norm_inf_u, norm_rel_u)
print "The norm infinity for v(x,y,t) is {0}, while the norm of relative errors
    is {1}".format(norm_inf_v, norm_rel_v)
```

A.6. kdv-burgers.py

```
# -*- coding: utf-8 -*-

from __future__ import division
import numpy as np
from chebPy import *
from matplotlib import pyplot as plt

# Interval
a, b = -10, 10

# KdV-Burger equations constants
eps = 0.1
v   = 0.1
mu  = 0.1

# Solution's constants
A = v**2/(25*eps*mu)
B = v/(10*mu)
C = 6*v**2/(25*mu)

# Time to evaluate
t_eval = 0.3

# Matrix size
N = 10

# Derivative matrix based on Chebyshev polynomials
D, x = cheb(N,a,b)
D2   = np.dot(D,D)
D3   = np.dot(D,D2)

D  = D[1:N-1,1:N-1]
D2 = D2[1:N-1,1:N-1]
D3 = D3[1:N-1,1:N-1]
x  = x[1:N-1]

# Intial condition u0 = u(x,0) | ux0 = u_x(x,0)
u0  = A*(9 - 6*np.tanh(B*(x)) - 3*(np.tanh(B*(x)))**2)
ux0 = -6*A*B*(np.tanh(B*x) + 1)*(1/np.cosh(B*x))**2

# Time
t_start = 0
t_end   = 2
delta_t = 0.001          # For Runge-Kutta

t_num   = (t_end - t_start)/delta_t
t               = np.linspace(t_start, t_end, t_num)

# Using Runge-Kutta 4 manually
def F(u, t, eps, v, mu, D, D2, D3):
        return eps * u * np.dot(D, u) + v * np.dot(D2, u) - mu * np.dot(D3, u)

u_sol = np.zeros((t_num, N-2))
```

```
u_sol[0, :] =  u0

for t_n in range(0, np.int(t_num - 1) ) :
        u_n = u_sol[t_n, :]
        k1 = F(u_n, t_n, eps, v, mu, D, D2, D3 )
        k2 = F(u_n + k1*delta_t/2, t_n + 1/2*delta_t, eps, v, mu, D, D2, D3)
        k3 = F(u_n + k2*delta_t/2, t_n + 1/2*delta_t, eps, v, mu, D, D2, D3)
        k4 = F(u_n + k3*delta_t,   t_n + delta_t, eps, v, mu, D, D2, D3)

        u_sol[t_n + 1, :] = u_n + 1/6*delta_t*(k1 + 2*k2 + 2*k3 + k4)

def error_norm_inf(u_compu, u_exact):
        return np.max(np.abs(u_compu - u_exact))

def error_norm_rel(u_compu, u_exact):
        up   = np.sum((u_compu - u_exact)**2)
        down = np.sum((u_exact)**2)
        return np.sqrt(up/down)

t_n_eval = (t_eval - t_start)/delta_t
t = t_eval
u_exact = A*(9 - 6*np.tanh(B*(x - C*t)) - 3*(np.tanh(B*(x - C*t)))**2)

norm_inf  = error_norm_inf(u_sol[t_n_eval,:], u_exact)
norm_rel  = error_norm_rel(u_sol[t_n_eval,:], u_exact)

print "Epsilon = {0}, Eta = {1}, Mu = {2}, t = {3}, N = {4} and delta_t = {5}".
    format(eps, v, mu, t_eval, N, delta_t)
print "The norm infinity is {0}, while the norm of relative errors is {1}".format
    (norm_inf, norm_rel)
```

Bibliografía

[1] Peter J. Olver. *Introduction to Partial Differential Equations*. Springer International Publishing, 2014.

[2] Babilonian collection ybc 7289. URL http://www.math.ubc.ca/~cass/Euclid/ybc/ybc.html.

[3] J.P. Boyd. *Chebyshev and Fourier Spectral Methods*. Dover Publishers, 2001.

[4] Steve Orszag. Numerical methods for the simulation of turbulence. *Lecture Notes in Physics*, 2:250–257, 1969.

[5] A.H. Khater et al. A chebyshev spectral collocation method for solving burgers'-type equations. *Journal of Computational and Applied Mathematics*, (222):333–350, May 2008.

[6] P. Schlatter. Spectral methods. Online Publication, Mar 2010. URL http://www.mech.kth.se/~ardeshir/courses/literature/Notes_Spectral_Methods.pdf.

[7] J. Shen and T. Tang. *Spectral and High-Order Methods with Applications*. Science Press, 2006.

[8] J. Anderson. *Fundamentals of Aerodynamics*. McGraw-Hill, 1991.

[9] J. Billingham and A.C. King. *Wave Motion*. Cambridge University Press, 2000.

[10] M. Landajuela. Burgers equation, 2011.

[11] G.B. Whithan. *Linear and Nonlinear Waves*. John Wiley and Sons, 1974.

[12] P.G. Drazin. *Solitons, An Introduction*. Cambridge University Press, 1983.

[13] John K. Hunter. *A model KdV problem: Ion acoustic waves*. UC Davis, 2006.

[14] A.A. Soliman. The modified extended direct algebraic method for solving nonlinear partial differential equations. *International Journal of Nonlinear Science*, 6(2):136–144, Jan 2008.

[15] S. Kutluay T. Özis, A. Esen. Numerical solution of burgers equation by quadratic b-spline finite elements. *Applied Mathematical and Computing*, 6(164):237–249, Jan 2005.

[16] O. Goyon. Multilevel schemes for solving unsteady equations. *International Journal for Numerical Methods in Fluids*, 22(22):937–959, Feb 1996.

[17] A.R. Bahadir. A fully implicit finite-difference scheme for two-dimensional burgers' equations. *Applied Mathematical and Computing*, 137(33):131–137, Feb 2003.

[18] J. Bezanson et al. Julia: A fresh approach to numerical computing. *CoRR*, abs/1411.1607, 2014. URL http://arxiv.org/abs/1411.1607.

[19] T. B. Benjamin. Internal waves of permanent form in fluids of great depth. *Journal of Fluid Mechanics*, 29:559–592, 9 1967. ISSN 1469-7645. doi: 10.1017/S002211206700103X. URL http://journals.cambridge.org/article_S002211206700103X.