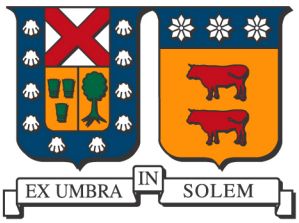
2016

ADAPTACIÓN TEÓRICA Y COMPUTACIONAL DE MÉTODO DE LATTICE BOLTZMANN PARA ECUACIONES DE SHALLOW WATER EN GPU

SALINAS EVANGELISTA, ÁLVARO SIMÓN

Universidad Técnica Federico Santa María

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
VALPARAÍSO - CHILE



Adaptación Teórica y Computacional del Método de *Lattice Boltzmann* para Ecuaciones de *Shallow Water* en GPU

ÁLVARO SIMÓN SALINAS EVANGELISTA

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE:
INGENIERO CIVIL INFORMÁTICO

PROFESOR GUÍA: CLAUDIO TORRES, Ph.D.
PROFESOR CO-GUÍA: ORLANDO AYALA, Ph.D.
PROFESOR CORREFERENTE: PATRICIO CATALÁN, Ph.D.

DICIEMBRE - 2016

Agradecimientos

Me gustaría expresar el más sincero agradecimiento al profesor Claudio Torres, mi profesor guía, por todo el tiempo dedicado al proyecto, por todos los conocimientos entregados a lo largo de mi formación como profesional, por haber creído en mí desde un comienzo y por todos los consejos y lecciones de vida que me ha dado durante estos años.

A mi profesor co-guía, Orlando Ayala, gracias por reunirse con nosotros cada semana durante estos últimos meses, siendo un apoyo fundamental en la realización de este trabajo, y gracias por compartir con nosotros sus conocimientos sobre el método de *Lattice Boltzmann* y dinámica de fluidos.

Quisiera también agradecer al Centro Científico Tecnológico de Valparaíso CCTVal por haber financiado este proyecto, permitiendo que se llevara a cabo.

Y finalmente, un agradecimiento especial a todos mis amigos y familiares, quienes me brindaron su compañía e infundieron alegría en los momentos buenos y me dieron su apoyo incondicional en los tiempos más difíciles. Sin ustedes no estaría aquí hoy, muchas gracias.

Dedicada a mi madre.
Gracias a ti estoy donde
estoy y soy quien soy.

Resumen

En el presente trabajo, se realiza una adaptación del método de *Lattice Boltzmann* para la resolución de las ecuaciones de *shallow water*. Se propone una variante híbrida del método, en la cual se utilizan dos funciones de distribución de equilibrio en distintas zonas del dominio con el objetivo de obtener un método "*well-balanced*" que soporte condiciones de borde abiertas. Se validan dichas propiedades mediante la simulación de un lago en reposo y un levantamiento central de agua cuyas ondas generadas atraviesan las fronteras de la simulación respectivamente. Adicionalmente, se comprueba la conservación de masa mediante la experimentación con condiciones de borde de tipo *Bounce-Back*. Finalmente, se implementa un código paralelizado en GPU, estudiando el aumento en la eficiencia computacional obtenido al disminuir los tiempos de ejecución del algoritmo.

Palabras clave: simulación numérica de tsunamis, ecuaciones de *shallow water*, método de *Lattice Boltzmann*, condiciones de borde abiertas, "*well-balanced*", GPU

Abstract

In the present work, an adaptation of the Lattice Boltzmann method is performed for the resolution of the shallow water equations. A hybrid variant of the method is proposed, in which two equilibrium distribution functions are used in different areas of the domain in order to obtain a well-balanced method that supports open boundary conditions. These properties are validated by the simulation of a lake at rest and a central water lifting which generated waves cross the boundaries of the simulation respectively. Additionally, mass conservation is proved by experimentation with Bounce-Back boundary conditions. Finally, a code parallelized in GPU is implemented, studying the increase in the computational efficiency obtained by decreasing the execution times of the algorithm.

Keywords: numerical simulation of tsunamis, shallow water equations, Lattice Boltzmann method, open boundary conditions, well-balanced, GPU

Glosario

Siglas

- CFL: Courant–Friedrichs–Lewy.
- CPU: Unidad de Procesamiento Central.
- CUDA: Arquitectura Unificada de Dispositivos de Cómputo.
- FLOPS: Operaciones de punto flotante por segundo.
- GPU: Unidad de Procesamiento Gráfico.
- NOAA: Administración Nacional Oceánica y Atmosférica.
- PCI: Interconexión de Componentes Periféricos.
- RAM: Memoria de Acceso Aleatorio.
- WENO: Esencialmente No Oscilatorio Ponderado.

Términos

- *Bounce-Back*: Tipo de condiciones de borde que simulan la presencia de un muro.
- *Colisión*: Paso fundamental del método de Lattice Boltzmann. Representa interacción entre partículas.
- D2Q9: Configuración del método de *Lattice Boltzmann* que representa dos dimensiones y nueve velocidades.
- *Dam-break*: Rotura de presa.
- *Kernel*: Subrutina que se ejecuta en GPU.
- *Lake at rest*: Lago en reposo.
- *Lattice Boltzmann*: Método para resolver ecuaciones diferenciales parciales.
- *Lattice gas cellular automata*: Autómata celular *Lattice-Gas*. Método para simulación de flujo de fluido.
- *Lattice unit* (*lu*): Unidad de medida para el espacio discretizado en el método de *Lattice Boltzmann*.
- *Shallow water*: Aguas someras o aguas poco profundas.

- *Streaming*: Paso fundamental del método de Lattice Boltzmann. Representa propagación de partículas.
- *Time step* (*ts*): Unidad de medida para el tiempo discretizado en el método de *Lattice Boltzmann*.
- *Upwind central scheme*: Esquema *upwind* (contra el viento) central.
- *Well-balanced*: Propiedad de un método que mantiene el estado inicial de altura de agua constante.

Variables

- b: Batimetría, altura del nivel del suelo.
- e: Vector de velocidad direccional.
- F: Término de fuerza.
- f: Función de distribución.
- $f^{(eq)}$: Función de distribución de equilibrio.
- g: Función de distribución.
- $g^{(eq)}$: Función de distribución de equilibrio.
- g: Aceleración de gravedad.
- h: Altura de agua desde la batimetría ($h = w - b$).
- i: Componente genérica de un vector.
- j: Componente genérica de un vector.
- L_x: Dimensión (número de nodos) del dominio discretizado a lo largo del eje x.
- L_y: Dimensión (número de nodos) del dominio discretizado a lo largo del eje y.
- p: Presión.
- S: Término fuente.
- t: Tiempo.
- u: Velocidad.
- v: Velocidad.
- V: Volumen.
- w: Altura de agua.
- x: Componente espacial.
- y: Componente espacial.
- α: Componente que representa una de las nueve direcciones del modelo D2Q9.
- ε: Variable que representa un valor pequeño en el estudio de perturbaciones.
- ρ: Densidad.
- ν: Viscosidad cinemática.
- τ: Tiempo de relajación para el paso de *colisión*.

Símbolos y Notaciones

- $\frac{\partial(\cdot)}{\partial *}$: Derivada parcial de $(\cdot)$ respecto a $*$.
- $\nabla(\cdot)$: Gradiente de $(\cdot)$, vector de sus derivadas parciales.
- $(\cdot)_*$: $*$-ésima componente del vector $(\cdot)$.

Índice general

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

La simulación numérica corresponde a una de las principales aplicaciones del área de la computación científica, teniendo como principales objetivos la reconstrucción de eventos y la predicción de sucesos futuros. Generalmente, los problemas de este tipo suponen el modelado de un sistema de ecuaciones diferenciales y la posterior resolución de éstas mediante la aplicación de un método numérico, requeriendo una gran capacidad de cómputo la mayoría de las veces. Dentro de este contexto, es posible encontrar los problemas de simulación de desastres naturales como tsunamis, sismos de gran intensidad y otros, siendo el primer ejemplo mencionado el foco de atención del presente trabajo de memoria.

Al simular tsunamis, existen diversas decisiones que deben ser tomadas para enfrentar correctamente el problema, destacando entre ellas la elección del sistema de ecuaciones a resolver, la selección del método numérico a implementar y la definición de la configuración que se utilizará, contemplando esta última la construcción de la malla mediante la cual se efectuará la discretización del problema, el tratamiento de variables y la determinación de parámetros. Teniendo esto en cuenta, para el desarrollo de este trabajo se ha decidido modelar el problema mediante las denominadas ecuaciones de *shallow water*, las cuales corresponden a un sistema de ecuaciones diferenciales parciales hiperbólicas que describen el flujo en la superficie de un fluido, siendo de gran utilidad al intentar reproducir una onda de agua. Por otro lado, entre los distintos métodos usualmente utilizados para la resolución de este sistema, se ha elegido el método de *Lattice Boltzmann* dado que es un gran candidato a la paralelización algorítmica con el fin de disminuir de forma drástica los tiempos de ejecución.

Así, el objetivo general de esta memoria es implementar el método de *Lattice Boltzmann* para la resolución de las ecuaciones de *shallow water* dentro del marco de la simulación de tsunamis. Para lograrlo, se han definido tres objetivos específicos a cumplir, siendo el primero de ellos codificar en un lenguaje de programación de alto nivel (Python) un algoritmo encargado de aplicar el método mencionado al problema a modo de prueba de concepto, permitiendo la obtención de resultados y posterior comparación con un algoritmo optimizado. En segundo lugar, ya habiendo obtenido una implementación funcional, se define como objetivo paralelizar el algoritmo mediante el uso de dispositivos gráficos (GPU) utilizando la arquitectura de cálculo paralelo CUDA en los lenguajes de programación C/C++. Finalmente, el tercer y último objetivo específico consiste en realizar simulaciones que permitan asegurar el cumplimiento de las propiedades fundamentales del problema y comparar el aumento de eficiencia computacional entre ambas versiones implementadas.

En cuanto a la estructura del presente documento, en el Capítulo 2 se presenta la definición del problema, contextualizando la importancia de la simulación numérica de tsunamis y describiendo el modelo elegido para la representación. A continuación, el Capítulo 3 corresponde al estado del arte, mencionando los principales trabajos relacionados presentes en la literatura. En el Capítulo 4 se expone un marco teórico para el método de *Lattice Boltzmann*, analizando sus aspectos más importantes. Los Capítulos 5 y 6 exhiben respectivamente, la solución propuesta al problema y el algoritmo implementado, incluyendo las decisiones tomadas ante la presencia de los obstáculos enfrentados. Luego, en el

Capítulo 7 se describen los experimentos realizados y se muestran sus resultados, mientras que en el Capítulo 8 se comparan las versiones secuencial y paralelizada del algoritmo, estudiando el efecto de la optimización. Finalmente, en los capítulos finales se concluye sobre el trabajo realizado y se señalan las referencias bibliográficas consultadas.

Capítulo 2

Definición del Problema

2.1. Simulación de Tsunamis

La simulación de tsunamis es un tema de bastante relevancia debido a que su uso está destinado a la evaluación de riesgos ante una catástrofe de esta índole. No cabe duda de que su importancia es aun mayor en un país tan sísmico como Chile. Es por esto que ya han surgido proyectos multidisciplinarios a gran escala para lograr avances importantes en el tema.

Chile siempre ha sido un foco de atención mundial debido al gran número de sismos que han afectado al país. El más recordado de ellos corresponde al megaterremoto ocurrido en Valdivia en el año 1960 [1], el cual registró una magnitud de 9,5 en escala Richter, siendo el más potente a lo largo de todo el mundo en la historia de la humanidad. Este desastre natural no solo se limitó al movimiento telúrico que de por sí produjo daños inmesurables, sino también dio pie a otras catástrofes como la erupción del volcán Puyehue y un devastador tsunami que azotó las costas del oceano Pacífico, afectando tanto a Chile como a territorios tan lejanos como son Hawái y Japón (su alcance puede ser apreciado en la Figura 2.1). Este evento costó la vida de cerca de dos mil personas y dejó más de dos millones de damnificados [2].

Si bien dicho evento se considera impactante, no corresponde a un suceso aislado, pues han ocurrido una gran cantidad de desastres naturales de este tipo a lo largo de la historia del país, contemplando más de 84 tsunamis en los últimos 100 años, de los cuales 4 han acontecido entre el año 2010 y el presente. En este contexto, la gran importancia de la investigación dedicada a esta área se vuelve evidente, siendo realmente necesario el desarrollo de herramientas que permitan reducir los riesgos y adquirir una mejor capacidad de reacción ante una catástrofe similar.

Otorgándole aun más relevancia al área, considerar solo a Chile al analizar la utilidad de ésta es ser demasiado localista, pues son muchos los países que se encuentran en situaciones similares, como Indonesia [3] o el ya mencionado Japón [4], siendo en realidad un tema de gran importancia e interés a escala mundial. Tan solo basta con remontarse al pasado 13 de Noviembre, donde un intenso sismo produjo un tsunami en las costas de Nueva Zelanda [5].

Las simulaciones numéricas se realizan con el objetivo de almacenar información relevante de eventos con distintas características, para luego, en el escenario de ocurrir un desastre real, relacionar los datos que se tienen en el momento con uno de los casos simulados y así tener un punto de comparación en base al cual tomar decisiones en el menor tiempo posible. De este modo, es posible obtener una gran cantidad de datos útiles sobre el tsunami que se avecina, como lo son la altura con la que llegará a la costa, la distancia de tierra que será cubierta por agua, la intensidad, velocidad o fuerza con la cual viene la ola, la estimación de zonas donde el daño puede ser mayor, entre otros.

Por otro lado, las simulaciones también son útiles para estudiar el fenómeno, permitiendo analizar y poner a prueba determinadas decisiones, como por ejemplo, la elección de terrenos destinados a la

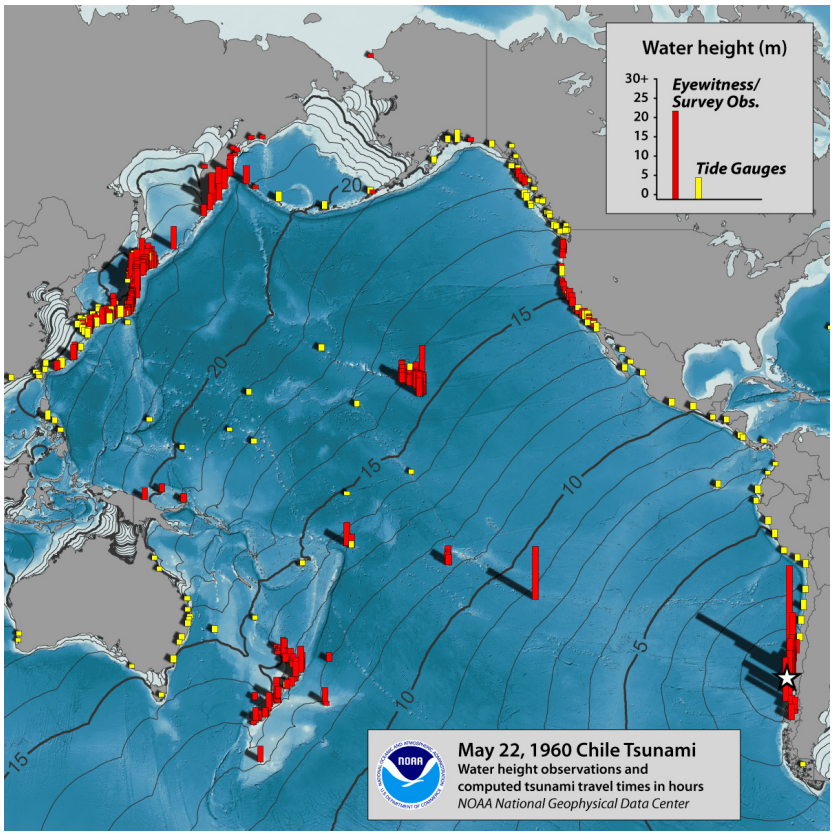


Figura 2.1: Altura de agua del tsunami provocado por el megaterremoto de Valdivia en el año 1960. Fuente: NOAA National Geophysical Data Center.

edificación, así como también evaluar y establecer políticas gubernamentales o estrategias de mitigación de efectos de tsunamis, como el levantamiento de terrenos o la construcción de muros con el fin de detener una posible inundación.

Para lograr estas simulaciones se utilizan métodos ampliamente validados. Entre los distintos enfoques y modelos que buscan representar el comportamiento de un tsunami, destacan las conocidas ecuaciones de *shallow water*, las cuales, en palabras simples, describen el flujo de la superficie de un fluido. Para resolverlas existen diversos métodos que utilizan determinados esquemas y representaciones. El problema radica en que este sistema no posee validez al representar la inundación de una zona seca, que es el elemento principal de un tsunami. A pesar de aquello, es posible obtener datos relevantes de una simulación que carezca de este elemento e incluso se puede implementar un método de inundación externo que se acople a la simulación, aunque esto último está considerado como trabajo futuro.

En este contexto, el problema atacado en esta memoria es la resolución de las ecuaciones de *shallow water* mediante el uso del método de *Lattice Boltzmann*, el cual ha adquirido gran relevancia recientemente [6, 7, 8, 9].

2.2. Representación del Problema: Ecuaciones de *shallow water*

Las ecuaciones de *shallow water*, también llamadas ecuaciones de aguas poco profundas y ecuaciones de aguas someras, son un sistema de ecuaciones diferenciales parciales hiperbólicas que describen el flujo de la superficie de un fluido. Su formulación es la siguiente:

$$\frac{\partial h}{\partial t} + \frac{\partial (hu)}{\partial x} + \frac{\partial (hv)}{\partial y} = 0 \tag{2.1}$$

$$\frac{\partial (hu)}{\partial t} + \frac{\partial}{\partial x}\left(hu^2 + \frac{1}{2}\mathrm{g}h^2\right) + \frac{\partial (huv)}{\partial y} = -\mathrm{g}h\frac{\partial b}{\partial x} \tag{2.2}$$

$$\frac{\partial (hv)}{\partial t} + \frac{\partial (huv)}{\partial x} + \frac{\partial}{\partial y}\left(hv^2 + \frac{1}{2}\mathrm{g}h^2\right) = -\mathrm{g}h\frac{\partial b}{\partial y} \tag{2.3}$$

donde $h(x, y, t)$ es la altura del fluido, $u(x, y, t)$ y $v(x, y, t)$ corresponden a las componentes x e y de la velocidad respectivamente, $b(x, y)$ representa la elevación de la batimetría y g es la constante de aceleración de gravedad. Esta corresponde a una formulación simple de las ecuaciones, pues el modelo puede ser complementado agregando términos adicionales, como la fuerza Coriolis, entre otros.

Las ecuaciones de *shallow water* se obtienen de las ecuaciones de Navier-Stokes, llamadas así por sus autores Claude-Louis Navier y George Gabriel Stokes. Estas ecuaciones describen el movimiento de fluidos viscosos. La primera de ellas es la ecuación de conservación de masa:

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

donde ρ es la densidad y $\boldsymbol{u}$ es la velocidad del fluido. Es a partir de esta ecuación desde donde se deriva (2.1).

La segunda ecuación corresponde a la conservación de momento y su formulación es:

$$\frac{\partial}{\partial t}(\rho \boldsymbol{u}) + \rho \boldsymbol{u} \cdot \nabla \boldsymbol{u} = -\nabla p + \mu \nabla^2 \boldsymbol{u} + \boldsymbol{F}$$

donde p es la presión, μ es la viscosidad dinámica y $\boldsymbol{F}$ es el término de fuerzas externas. Las ecuaciones (2.2) y (2.3) se derivan de esta ecuación.

En palabras simples, la derivación de las ecuaciones de *shallow water* se consigue al establecer una escala de longitud horizontal mucho más grande que la vertical, condición bajo la cual, según la conservación de masa, la velocidad vertical del fluido es muy pequeña y puede ser eliminada del modelo mediante integración. Lo mismo ocurre con el término de presión, ya que bajo estas condiciones, gracias a la ecuación de conservación de momento, es posible demostrar que los gradientes verticales de presión son asintóticamente hidrostáticos y los horizontales se deben al desplazamiento de la superficie, implicando que las velocidades en las profundidades del fluido son constantes.

Para resolver este sistema numéricamente, es necesaria una discretización del dominio mediante alguno de los métodos que serán presentados en la Sección 3.1. Una característica en común que deben presentar todos estos métodos es la propiedad de ser "*well-balanced*". Esto supone modelar correctamente una de las soluciones de estado estacionario más importantes, correspondiente a la de "*lake at rest*", o lago en reposo en español, que se define por:

$$u = 0, \qquad v = 0, \qquad w = h + b = \text{constante}$$

Esto significa que mientras no hayan fuerzas externas, si la altura del agua es constante a lo largo del dominio al tiempo $t = 0$, entonces debería mantenerse constante a medida que avanza el tiempo. Si bien esto suena lógico y algo simple de lograr, en realidad corresponde a un gran desafío, pues la mayoría de los métodos están sujetos a la generación de perturbaciones pequeñas que no permiten un nivel constante de agua. Los métodos que cumplen el estado estacionario y no generan dichas perturbaciones erróneas, son llamados "*well-balanced*".

Capítulo 3

Estado del Arte

3.1. Ecuaciones de *shallow water*

En la presente sección se discutirá sobre los principales enfoques utilizados para resolver el sistema conformado por las ecuaciones de *shallow water*. Como se mencionó en la Sección 2.2, para hallar una solución de este sistema, se hace necesaria la implementación de métodos numéricos que permitan generar una discretización del problema. De esta manera, es posible identificar en la literatura el establecimiento de tres corrientes principales, correspondientes a los métodos de diferencias finitas, volúmenes finitos y *Lattice Boltzmann*, adquiriendo esta última una relevancia considerable en la actualidad, lo cual será analizado en la Sección 3.2.

Comenzando por el primero de los métodos mencionados, el método de las diferencias finitas consiste en la estimación de la solución de una ecuación diferencial mediante la aproximación de derivadas por medio de diferencias finitas. Éstas son expresiones matemáticas en las que se reemplaza el término infinitesimal del cuociente que define una derivada por un valor finito.

Según su definición, la derivada de una función $f(x)$ es:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

En un dominio discretizado de forma equidistante, en donde el valor h corresponde a la distancia entre dos puntos consecutivos x_i y x_{i+1}, la aproximación de la derivada de una función $f(x)$ puede estimarse de tres formas según el tipo de diferencias finitas utilizado:

$$f'(x) \approx \frac{f(x+h) - f(x)}{h} = \frac{f(x_{i+1}) - f(x_i)}{\Delta x} \qquad \text{Diferencia Progresiva}$$

$$f'(x) \approx \frac{f(x) - f(x-h)}{h} = \frac{f(x_i) - f(x_{i-1})}{\Delta x} \qquad \text{Diferencia Regresiva}$$

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h} = \frac{f(x_{i+1}) - f(x_{i-1})}{2\Delta x} \qquad \text{Diferencia Centrada}$$

En la literatura es posible encontrar artículos no muy recientes en donde se aplica este método a la resolución de las ecuaciones de *shallow water*, siendo efectivamente éste uno de los primeros enfoques utilizados para afrontar el problema. Entre los más antiguos, destaca el trabajo de Sadourny [10] en el año 1975, en donde el autor estudia las propiedades de conservación de dos modelos numéricos basados en diferencias finitas. Es importante también mencionar el artículo de los autores Casulli y Cattani [11] del año 1994, en donde se realiza un análisis de estabilidad, eficiencia y precisión de un esquema semi-implícito en la resolución de las ecuaciones de *shallow water* de tres dimensiones.

Avanzando a la década del 2000, es posible encontrar el trabajo de Israeli et. al [12], en el cual se presenta un método incondicionalmente estable, eliminando la restricción producida por la condición CFL. También el trabajo de Rasulov et. al [13], en donde se aplica el método de diferencias finitas para resolver las ecuaciones de *shallow water* representadas mediante funciones discontinuas en una dimesión. Por su parte, los autores Delis y Katsaounis [14] emplean métodos de relajación como una generalización del método poniendo especial énfasis en el manejo del término fuente que representa el efecto de la batimetría. Se destaca adicionalmente, el artículo de Skiba y Filatov [15], en el cual se propone un método para construir distintos esquemas conservativos basados en diferencias finitas, considerando el modelo clásico de las ecuaciones de *shallow water*.

Ya en el año 2011, los autores Lu y Qiu [16] estudian la aplicación del método Lax-Wendroff en la discretización del tiempo a un esquema de diferencias finitas esencialmente no oscilatorio ponderado (WENO, por su sigla en inglés). Entre las principales publicaciones recientes se encuentran los trabajos de Luo y Gao, analizando en uno de ellos [17] los fenómenos de "*dam-break*" y de llanura aluvial a través de un esquema de diferencias finitas, y proponiendo en el otro [18], junto al autor Zhenghui Xie, un modelo de extrapolación de diferencias finitas basado en una descomposición ortogonal. Destaca también el trabajo de Li et. al [19], en donde se presenta un esquema de tipo WENO que mantiene la propiedad de "*well-balanced*" ante batimetrías no planas.

Por otra parte, continuando con el segundo método mencionado al comienzo de la sección, en el método de volúmenes finitos se construyen volúmenes de control en torno a los puntos que constituyen el dominio discretizado sobre el que se trabaja, cuidando que estos volúmenes no se traslapen entre sí. De este modo, la suma de los volúmenes de control considerados es equivalente al volumen total del fluido representado.

La metodología integra las ecuaciones diferenciales en cada volumen de control, obteniendo una versión discretizada de dicha ecuación. Si bien debe cumplir una consistencia en los flujos, es decir, el flujo que sale de un volumen de control hacia sus vecinos debe ser igual al que ellos reciben de él, su principal propiedad es que, de manera independiente al tamaño de la malla, el sistema de ecuaciones discretizadas resultante satisface en forma exacta las ecuaciones de conservación consideradas.

Haciéndose necesaria una función de flujo para determinar la interacción entre volúmenes vecinos pertenecientes la malla discreta, en el año 2000, Alexander Kurganov y Eitan Tadmor desarrollaron un novedoso esquema que ha adquirido bastante protagonismo: el denominado "*upwind central scheme*" [20].

Continuando con la labor de Kurganov, quien ha dedicado su trabajo al tema y posee bastantes publicaciones sobre el mismo, dos años después de la definición de la función mencionada, junto a Doron Levy presentó la aplicación de ésta a la resolución del sistema de ecuaciones de *shallow water* [21], demostrando que el método contaba con preservación de la positividad de la altura del agua, siendo capaz de manejar la presencia de zonas secas y simular la inundación. Ya en el año 2005, Kurganov y Guergana Petrova implementan el esquema en mallas triangulares para la resolución de sistemas de ecuaciones diferenciales parciales hiperbólicas de leyes de conservación [22], y tras dos años, en el 2007, ambos autores logran obtener un método "*well-balanced*" que presenta preservación de la positividad al mismo tiempo [23], algo no conseguido anteriormente.

Finalmente, en un trabajo conjunto con Steve Bryson, quien ya había trabajado en el año 2005 con las ecuaciones de *shallow water* en mallas no estructuradas [24], y otros autores, se logran unir todos los conceptos mencionados, implementando un esquema "*upwind central*" en un método de volúmenes finitos "*well-balanced*" y preservando la positividad de la altura del agua para resolver las ecuaciones de *shallow water* mediante la utilización de mallas triangulares no estructuradas [25]. En el trabajo mencionado, se presenta la siguiente formulación:

Considerando la altura de la superficie del agua $w = b + h$, y utilizando el vector $\boldsymbol{U} = (w, hu, hv)$, es posible reescribir (2.1), (2.2) y (2.3) como:

$$\boldsymbol{U}_t + \boldsymbol{F}(\boldsymbol{U}, b)_x + \boldsymbol{G}(\boldsymbol{U}, b)_y = \boldsymbol{S}(\boldsymbol{U}, b) \tag{3.1}$$

donde los flujos $\boldsymbol{F}$ y $\boldsymbol{G}$ se describen mediante:

$$\boldsymbol{F}(\boldsymbol{U}, b) = (hu, \frac{(hu)^2}{w-b} + \frac{1}{2}\mathrm{g}(w-b)^2, \frac{(hu)(hv)}{w-b})$$

$$\boldsymbol{G}(\boldsymbol{U}, b) = (hv, \frac{(hu)(hv)}{w-b}, \frac{(hv)^2}{w-b} + \frac{1}{2}\mathrm{g}(w-b)^2)$$

y el término fuente $\boldsymbol{S}$ está dado por:

$$\boldsymbol{S}(\boldsymbol{U}, b) = (0, -\mathrm{g}(w-b)b_x, -\mathrm{g}(w-b)by)$$

Considerando una malla triangular no estructurada y el método de volúmenes finitos, se define (x_j, y_j) como las coordenadas del centro de masa del volumen de control T_j y para valores de $k = 1, 2, 3$, $M_{jk} = (x_{jk}, y_{jk})$, l_{jk} y θ_{jk} respectivamente como el punto medio, el largo y el ángulo respecto a la horizontal del vector normal saliente del lado compartido con su k-esimo vecino.

De esta forma, aplicando un esquema semi-discreto a la aproximación de la solución de (3.1), se obtiene:

$$\bar{\boldsymbol{U}}_j(t) \approx \frac{1}{|T_j|}\int_{T_j} \boldsymbol{U}(x, y, t)\mathrm{d}x\mathrm{d}y \tag{3.2}$$

Finalmente, utilizando el esquema "*upwind central*" en su versión de segundo orden [22] como función de flujo para resolver (3.2), se llega a:

$$\begin{aligned}
\frac{\mathrm{d}\bar{\boldsymbol{U}}_j}{\mathrm{d}t} =& -\frac{1}{|T_j|}\sum_{k=1}^{3}\frac{\ell_{jk}\cos(\theta_{jk})}{a_{jk}^{\mathrm{in}} + a_{jk}^{\mathrm{out}}}\left[a_{jk}^{\mathrm{in}}\boldsymbol{F}\left(\boldsymbol{U}_{jk}(M_{jk}), b(M_{jk})\right) + a_{jk}^{\mathrm{out}}\boldsymbol{F}\left(\boldsymbol{U}_j(M_{jk}), b(M_{jk})\right)\right] \\
&- \frac{1}{|T_j|}\sum_{k=1}^{3}\frac{\ell_{jk}\sin(\theta_{jk})}{a_{jk}^{\mathrm{in}} + a_{jk}^{\mathrm{out}}}\left[a_{jk}^{\mathrm{in}}\boldsymbol{G}\left(\boldsymbol{U}_{jk}(M_{jk}), b(M_{jk})\right) + a_{jk}^{\mathrm{out}}\boldsymbol{G}\left(\boldsymbol{U}_j(M_{jk}), b(M_{jk})\right)\right] \\
&+ \frac{1}{|T_j|}\sum_{k=1}^{3}\ell_{jk}\frac{a_{jk}^{\mathrm{in}}a_{jk}^{\mathrm{out}}}{a_{jk}^{\mathrm{in}} + a_{jk}^{\mathrm{out}}}\left[\boldsymbol{U}_{jk}(M_{jk}) - \boldsymbol{U}_j(M_{jk})\right] + \overline{\boldsymbol{S}}_j
\end{aligned}$$

donde a_{jk}^{in} y a_{jk}^{out} son las velocidades direccionales locales y $\boldsymbol{U}_j(M_{jk})$ y $\boldsymbol{U}_{jk}(M_{jk})$ son los valores evaluados en el punto M_{jk} de la reconstrucción lineal por tramos:

$$\tilde{\boldsymbol{U}}(x, y) \coloneqq \bar{\boldsymbol{U}}_j + \left(\boldsymbol{U}_x\right)_j(x - x_j) + \left(\boldsymbol{U}_y\right)_j(y - y_j), \quad (x, y) \in T_j$$

de $\boldsymbol{U}$ en el tiempo t, esto es:

$$\boldsymbol{U}_j(M_{jk}) \coloneqq \lim_{(x,y)\to M_{jk};(x,y)\in T_j}\tilde{\boldsymbol{U}}(x, y), \qquad \boldsymbol{U}_{jk}(M_{jk}) \coloneqq \lim_{(x,y)\to M_{jk};(x,y)\in T_{jk}}\tilde{\boldsymbol{U}}(x, y)$$

Por otro lado, es importante destacar el estudio de reconstrucciones hidrostáticas presentes en la literatura. Así, se rescata la labor de Emmanuel Audusse et al. [26], quienes propusieron una reconstrucción hidostática con volúmenes finitos en un esquema "*well-balanced*" para resolver las ecuaciones de *shallow water*. Otros autores han presentado diversos enfoques, como una nueva discretización de la topografía en un modelo hidrostático que es "*well-balanced*" y maneja zonas secas [27], seguida de un estudio sobre las limitaciones del método al presentar un comportamiento anormal en ciertas condiciones [28].

En la actualidad, el método de volúmenes finitos corresponde a la corriente más popular, siendo la que ha presentado los mejores resultados en la resolución de las ecuaciones de *shallow water*. Entre las publicaciones más recientes sobre el tema destaca el trabajo de Dutykh y Clamond [29], quienes utilizan una discretización de volúmenes finitos al modelar una modificación de las ecuaciones de *shallow water*

que presenta batimetrías variantes en el tiempo. Por otro lado, el autor Yulong Xing [30] aplica un esquema de tipo WENO al método para canales de agua abiertos con geometría irregular. Por último, es posible encontrar el artículo de Beljadid et. al [31], en el cual se propone un nuevo enfoque para el análisis de estabilidad del método en mallas irregulares mediante la utilización de espectro, pseudo-espectro y descomposición en valores singulares.

Para finalizar, es sumamente importante mencionar las implementaciones más famosas en la actualidad destinadas a modelar las ecuaciones de *shallow water*. COMCOT [32] corresponde a la más popular de ellas, consistiendo en la utilización de un esquema de diferencias finitas con salto escalonado. También se destaca el modelo ANUGA [33], el cual se basa en el método de volúmenes finitos para simular la propagación de ondas cerca de la costa, con especial énfasis en el fenómeno de inundación.

3.2. Método de *Lattice Boltzmann*

El método de *Lattice Boltzmann* es una técnica de simulación relativamente nueva dentro del contexto de sistemas complejos de fluidos. Al contrario de los otros métodos analizados en la Sección 3.1, que resuelven numéricamente las ecuaciones de conservación de propiedades macroscópicas, como masa o momentum, los métodos de *Lattice Boltzmann* modelan el fluido sobre una malla discreta, analizando la propagación y colisión de colecciones ficticias de partículas presentes en él. Una explicación más detallada del método será presentada en el Capítulo 4, estando la presente sección dedicada a referenciar las principales aplicaciones del método presentes en la literatura, tanto a las ecuaciones de *shallow water* como a otros problemas de simulación.

La primera ecuación de *Lattice Boltzmann* fue presentada en el año 1988 por los autores Guy McNamara y Gianluigi Zanetti [34], en cuyo artículo proponían una mejora al método LGCA ("*Lattice gas cellular automata*"). La principal idea era eliminar el ruido estadístico presente en el método y con esto, reducir los tiempos de cómputo requeridos. A partir de entonces, con el artículo del año 1989 de los españoles Higuera y Jiménez [35] en donde se aplica la ecuación de *Lattice Boltzmann* a la hidrodinámica de lattices, surge el método de *Lattice Boltzmann* como una herramienta simple y eficiente para afrontar problemas de dinámicas de fluido, ganando de inmediato un gran interés. De esta forma, el primer libro dedicado enteramente al método fue publicado en el año 2001 a manos del autor Sauro Succi [36], y en él se tratan sus aspectos más relevantes.

Así mismo, han sido publicados otros libros sobre el método de *Lattice Boltzmann*, destacando entre ellos los de Jian Guo Zhou [37], Michael Sukop y Daniel Thorne [38] y Abdulmajeed Mohamad [39]. El primero de ellos es el que más se relaciona con este trabajo, pues en él se propone una adaptación del método para resolver las ecuaciones de *shallow water*. Los dos libros mencionados restantes tratan los aspectos importantes del método, su derivación a partir de las ecuaciones de Navier-Stokes y sus principales aplicaciones.

Aun así, la propuesta de Zhou no fue la primera, pues Rick Salmon ya había derivado un método para resolver el problema en el año 1999 [40]. A partir de estas dos publicaciones, la resolución de las ecuaciones de *shallow water* mediante la aplicación del método de *Lattice Boltzmann* ha ido adquiriendo importancia, pero siendo aun un caso de estudio relativamente reciente que cuenta con pocas publicaciones. En este contexto, es posible encontrar relevantes avances en el tema, como es el caso del artículo de Thömmes et. al [41], en donde se presenta una representación para el término fuente que permita manejar de forma precisa distintas configuraciones de batimetría. También se destaca el trabajo de Geveler et. al [42], en el cual se resuelven las ecuaciones de *shallow water* mediante del método de *Lattice Boltzmann* incluyendo objetos sólidos moviéndose a través del fluido, todo esto probando la paralelización del método en distintas arquitecturas. Ya en el año 2010, Zhou [43] propone una mejora a su trabajo anterior, permitiendo ahora manejar batimetrías no planas al incorporar de una manera consistente el nivel del suelo en la ecuación de *Lattice Boltzmann*.

Entre los trabajos más actuales sobre el tema, destaca el artículo de Li et. al [6], en el cual se propone una nueva función de equilibrio para resolver las ecuaciones de *shallow water*, modificando la función

propuesta por Zhou de modo que el término de presión hidrostática se calcule de manera conjunta a la inclinación de la batimetría. Siguiendo esta línea, una de las publicaciones más recientes corresponde a la de Hedjripour et. al [9] en el presente año, en donde se implementa un modelo asimétrico de *Lattice Boltzmann* en la resolución unidimensional de las ecuaciones de *shallow water*.

Es importante mencionar que en la actualidad, el método de *Lattice Boltzmann* no solo es utilizado en problemas de dinámicas de fluido, sino en diversos problemas que involucren la resolución de ecuaciones diferenciales parciales. Dentro de las aplicaciones del método es posible encontrar diversas áreas, como la termodinámica, más específicamente problemas de transferencia de calor [44, 45], electrohidrodinámica [46, 47], aerodinámica [48, 7], acústica [49, 8], entre otras.

Finalmente, es posible encontrar librerías de código abierto basadas en el método de Lattice Boltzmann en el marco de la dinámica de fluidos computacional, destacando entre ellas Palabos [50] y OpenLB [51]. Del mismo modo, existe también *software* privado destinado a este propósito, siendo PowerFLOW [52] el más popular al ser ampliamente utilizado en la industria automotriz y de transporte, debido a su aplicación validada en temas como aerodinámica, aeroacústica, gestión térmica, control climático y utilidades para el sistema de tren motriz.

Capítulo 4

Marco Teórico: El Método de *Lattice Boltzmann*

4.1. Introducción al Método

Es posible realizar una clasificación de los métodos numéricos utilizados en simulaciones de acuerdo a la escala de dimensión con la que representan el dominio del problema. Por un lado está la macroescala, en la cual el dominio es discretizado en volúmenes, celdas, elementos, entre otros, con el fin de considerar la velocidad, presión, temperatura y las demás variables a analizar como un valor promedio dentro del volumen finito analizado. En el otro extremo, podría considerarse la interacción de cada partícula (átomo o molécula) con las demás, lo que corresponde a una microescala. Debido a que en un extremo se debe lidiar con la imprecisión provocada por la discretización y en el otro se necesita un excesivo nivel de recursos computacionales, se hace necesario encontrar un punto medio. Es aquí donde surge el método de *Lattice Boltzmann*, en el cual se consideran las interacciones entre colecciones de partículas, las cuales están definidas por una función de distribución. Esto se denomina mesoescala.

En el método de *Lattice Boltzmann*, la unidad con la que se mide el tiempo en el *lattice* se denomina *time step* (ts) y el espacio es discretizado en *lattice units* (lu). Las posiciones de las colecciones de partículas corresponden a nodos de una malla regular equiespaciada. De acuerdo a la configuración utilizada en este trabajo, las partículas pueden moverse en 8 direcciones para llegar a otro nodo. Este modelo se conoce como D2Q9 (ver Figura 4.1) debido a que representa 2 dimensiones y contiene 9 velocidades.

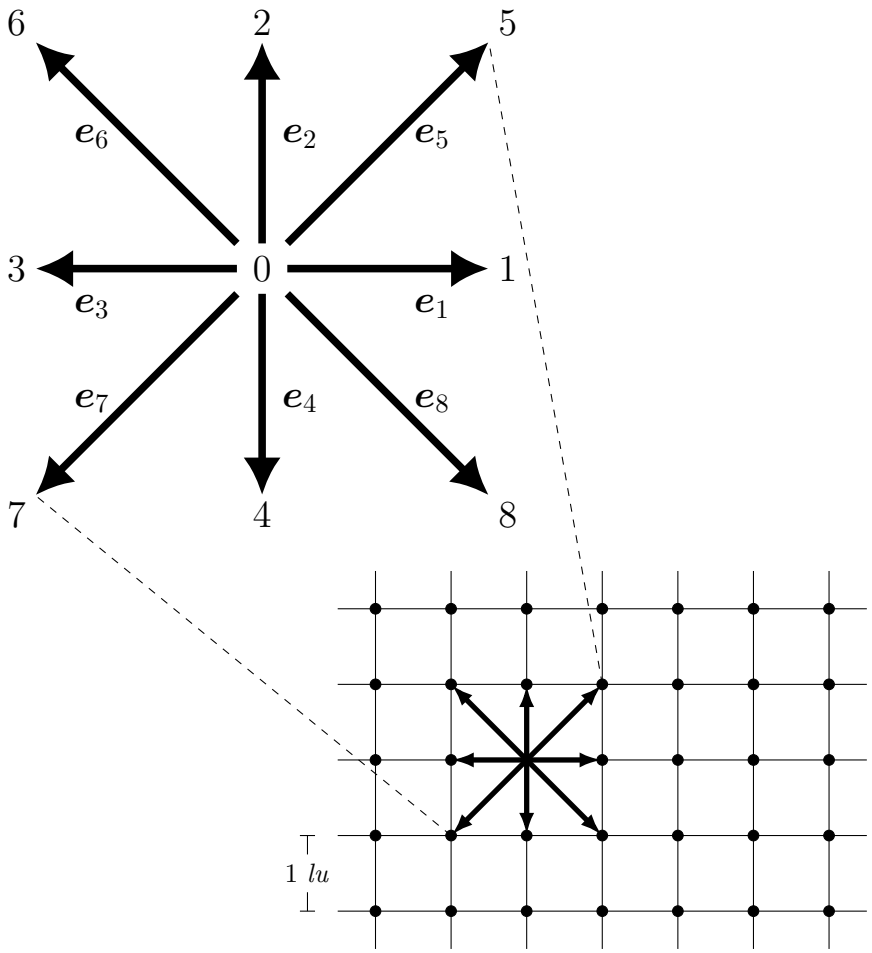


Figura 4.1: Modelo D2Q9.

La magnitud de las velocidades $\boldsymbol{e}_1$, $\boldsymbol{e}_2$, $\boldsymbol{e}_3$ y $\boldsymbol{e}_4$ es igual a 1 $lu\ ts^{-1}$, mientras que la de $\boldsymbol{e}_5$, $\boldsymbol{e}_6$, $\boldsymbol{e}_7$ y $\boldsymbol{e}_8$ corresponde a $\sqrt{2}\ lu\ ts^{-1}$. Así mismo, es posible apreciar que los vectores $\boldsymbol{e}_a$ están definidos como:

$$\boldsymbol{e}_\alpha = \begin{cases} (0,0), & \alpha = 0 \\ \left(\cos\dfrac{(\alpha-1)\pi}{4}, \sin\dfrac{(\alpha-1)\pi}{4}\right), & \alpha = 1,2,3,4 \\ \left(\sqrt{2}\cos\dfrac{(\alpha-1)\pi}{4}, \sqrt{2}\sin\dfrac{(\alpha-1)\pi}{4}\right), & \alpha = 5,6,7,8 \end{cases} \tag{4.1}$$

Aquí es donde aparece la denominada función de distribución f. En esta configuración, describir dicha función equivale a pensar en un histograma que representa la frecuencia de ocurrencia con la que las partículas de un determinado nodo toman una de las 9 velocidades posibles. De este modo, cada uno de los nueve componentes de la función de distribución corresponde a una densidad del fluido con una dirección específica, por lo que la densidad macroscópica ρ de cada nodo se calcula como:

$$\rho = \sum_{\alpha=0}^{8} f_\alpha \tag{4.2}$$

Del mismo modo, la velocidad macroscópica $\boldsymbol{u}$ no es más que el promedio de las velocidades mi-

croscópicas $\boldsymbol{e}_\alpha$ ponderadas por las densidades direccionales f_α obtenidas de la función de distribución:

$$\boldsymbol{u} = \frac{1}{\rho}\sum_{\alpha=0}^{8} f_\alpha \boldsymbol{e}_\alpha \tag{4.3}$$

En líneas generales, el método de *Lattice Boltzmann* consiste esencialmente en la realización de dos pasos llamados *streaming* y *colisión*. El primero de ellos corresponde a la etapa que representa el movimiento de las partículas, desplazando las densidades de un nodo a otro de acuerdo a los valores de la función de distribución para cada una de las direcciones posibles. Por otra parte, la *colisión* representa la interacción entre las partículas, pudiendo cambiar sus velocidades direccionales al llegar al siguiente nodo.

El enfoque más simple del método utiliza la aproximación BGK (Bhatnagar-Gross-Krook) para la implementación de la *colisión*, originando la siguiente ecuación de *Lattice Boltzmann*:

$$\underbrace{f_\alpha(\boldsymbol{x}+\boldsymbol{e}_\alpha\Delta t, t+\Delta t) = f_\alpha(\boldsymbol{x},t)}_{streaming} - \underbrace{\frac{[f_\alpha(\boldsymbol{x},t) - f_\alpha^{(eq)}(\boldsymbol{x},t)]}{\tau}}_{colisión} \tag{4.4}$$

donde τ es el denominado tiempo de relajación y $f_\alpha^{(eq)}$ corresponde a la función de distribución de equilibrio.

4.2. Función de Distribución de Equilibrio

La función de distribución de equilibrio es una parte fundamental del método y su determinación juega un rol esencial en su implementación, pues es ésta quien define qué ecuaciones se están resolviendo. En otras palabras, LBM mantiene su estructura algorítmica general para resolver diversos problemas como difusión de calor, flujos turbulentos, fluidos compresibles, entre otras simulaciones, pero lo único que debe cambiar para adaptarse de un problema a otro, es su función de distribución de equilibrio. Es por esto que a continuación se determinará esta función en base al problema que se intenta resolver, es decir, las ecuaciones de *shallow water* vistas en la Sección 2.2.

4.2.1. Primera Función de Distribución de Equilibrio ($f^{(eq)}$)

En la literatura es posible encontrar dos funciones de distribución de equlibrio que son útiles para resolver el problema con el que se está lidiando y, como se verá más adelante, en este trabajo se hará uso de ambas al combinarlas en distintas partes del dominio. La primera de ellas fue propuesta por Jian Guo Zhou [37] y está definida como:

$$f_\alpha^{(eq)} = \begin{cases} h - \dfrac{5\mathrm{g}h^2}{6} - \dfrac{2h}{3}\boldsymbol{u}_i^2, & \alpha = 0 \\ \dfrac{\mathrm{g}h^2}{6} + \dfrac{h}{3}\boldsymbol{e}_{\alpha i}\boldsymbol{u}_i + \dfrac{h}{2}\boldsymbol{e}_{\alpha i}\boldsymbol{e}_{\alpha j}\boldsymbol{u}_i\boldsymbol{u}_j - \dfrac{h}{6}\boldsymbol{u}_i^2, & \alpha = 1,2,3,4 \\ \dfrac{\mathrm{g}h^2}{24} + \dfrac{h}{12}\boldsymbol{e}_{\alpha i}\boldsymbol{u}_i + \dfrac{h}{8}\boldsymbol{e}_{\alpha i}\boldsymbol{e}_{\alpha j}\boldsymbol{u}_i\boldsymbol{u}_j - \dfrac{h}{24}\boldsymbol{u}_i^2, & \alpha = 5,6,7,8 \end{cases} \tag{4.5}$$

Para probar que al utilizar esta función de distribución de equilibrio las variables macroscópicas calculadas en las ecuaciones (4.2) y (4.3) corresponden a la solución de las ecuaciones de *shallow water*, se necesita realizar la expansión de Chapman-Enskog de la ecuación de *Lattice Boltzmann* definida en (4.4) con el ebjetivo de recuperar las ecuaciones (2.1), (2.2) y (2.3).

Asumiendo que Δt es pequeño e igual a ε:

$$\Delta t = \varepsilon$$

y considerando que, debido a que se está trabajando con las ecuaciones de *shallow water*, se hace necesaria la adición de un término que permita recuperar el término fuente y las demás fuerzas externas que pudiesen aplicarse al modelo, es posible reescribir la ecuación (4.4) como:

$$f_\alpha(\boldsymbol{x}+\boldsymbol{e}_\alpha\varepsilon, t+\varepsilon) = f_\alpha(\boldsymbol{x},t) - \frac{[f_\alpha(\boldsymbol{x},t) - f_\alpha^{(eq)}(\boldsymbol{x},t)]}{\tau} + \frac{\varepsilon}{N_\alpha}\boldsymbol{e}_{\alpha i}\boldsymbol{F}_i \tag{4.6}$$

donde $\boldsymbol{e}_{\alpha i}$ corresponde a la i-ésima componente del vector de velocidad $\boldsymbol{e}_\alpha$, $\boldsymbol{F}_i$ corresponde a la componente del término de fuerza en la dirección i, y N_α es una constante determinada por:

$$N_\alpha = \sum_{\alpha=0}^{8} \boldsymbol{e}_{\alpha i}^2$$

lo que en el modelo D2Q9 lleva a $N_\alpha = 6$.

Aplicando a la ecuación (4.6) las siguientes expansiones:

$$f_\alpha(\boldsymbol{x}+\boldsymbol{e}_\alpha\varepsilon, t+\varepsilon) = \sum_{n=0}^{\infty} \frac{\varepsilon^n}{n!} D_t^n f_\alpha(\boldsymbol{x},t)$$

$$f_\alpha = \sum_{n=0}^{\infty} \varepsilon^n f_\alpha^{(n)}$$

con el operador D_t definido como $D_t = (\partial_t + \boldsymbol{e}_\alpha \cdot \nabla)$, se obtiene la siguiente ecuación:

$$\sum_{n=0}^{\infty} \frac{\varepsilon^n}{n!} D_t^n \left(\sum_{m=0}^{\infty} \varepsilon^m f_\alpha^{(m)} \right) = \sum_{n=0}^{\infty} \varepsilon^n f_\alpha^{(n)} - \frac{1}{\tau}\left(\sum_{n=0}^{\infty} \varepsilon^n f_\alpha^{(n)} - f_\alpha^{(eq)} \right) + \frac{\varepsilon}{6}\boldsymbol{e}_{\alpha i}\boldsymbol{F}_i$$

El siguiente paso es tomar los términos que dependen de hasta el segundo orden de ε. Así, considerando solo los términos de $O(1)$ se obtiene:

$$\underbrace{\cancel{f_\alpha^{(0)}}}_{1} = \underbrace{f_\alpha^{(0)}}_{1} - \frac{1}{\tau}\left(f_\alpha^{(0)} - f_\alpha^{(eq)} \right)$$

Eliminando los términos calcelados se reduce a:

$$f_\alpha^{(0)} = f_\alpha^{(eq)} \tag{4.8}$$

Tomando ahora aquellos términos de hasta $O(\varepsilon)$ se llega a la siguiente ecuación:

$$\underbrace{\cancel{f_\alpha^{(0)}}}_{1} + \underbrace{\cancel{\varepsilon f_\alpha^{(1)}}}_{2} + \varepsilon D_t f_\alpha^{(0)} = \underbrace{f_\alpha^{(0)}}_{1} + \underbrace{\cancel{\varepsilon f_\alpha^{(1)}}}_{2} - \frac{1}{\tau}\left(\underbrace{\cancel{f_\alpha^{(0)}}}_{3} + \varepsilon f_\alpha^{(1)} \underbrace{\cancel{f_\alpha^{(eq)}}}_{3} \right) + \frac{\varepsilon}{6}\boldsymbol{e}_{\alpha i}\boldsymbol{F}_i$$

Nuevamente, al eliminar los términos cancelados se obtiene:

$$D_t f_\alpha^{(0)} = -\frac{1}{\tau} f_\alpha^{(1)} + \frac{1}{6}\boldsymbol{e}_{\alpha i}\boldsymbol{F}_i \tag{4.9}$$

Por último, considerando todos los términos que dependan de hasta $O(\varepsilon^2)$, la ecuación generada es:

$$\underbrace{\cancel{f_\alpha^{(0)}}}_{1} + \underbrace{\cancel{\varepsilon f_\alpha^{(1)}}}_{2} + \underbrace{\cancel{\varepsilon^2 f_\alpha^{(2)}}}_{3} +$$

$$\varepsilon D_t f_\alpha^{(0)} + \varepsilon^2 D_t f_\alpha^{(1)} + \frac{\varepsilon^2}{2} D_t^2 f_\alpha^{(0)} = \underbrace{\cancel{f_\alpha^{(0)}}}_{1} + \underbrace{\cancel{\varepsilon f_\alpha^{(1)}}}_{2} + \underbrace{\cancel{\varepsilon^2 f_\alpha^{(2)}}}_{3} + \frac{\varepsilon}{6}\boldsymbol{e}_{\alpha i}\boldsymbol{F}_i$$

$$- \frac{1}{\tau}\left(\underbrace{\cancel{f_\alpha^{(0)}}}_{4} + \varepsilon f_\alpha^{(1)} + \varepsilon^2 f_\alpha^{(2)} - \underbrace{\cancel{f_\alpha^{(eq)}}}_{4} \right)$$

Cancelando términos se obtiene:

$$\underbrace{\cancel{\varepsilon D_t f_\alpha^{(0)}}}_{1} + \varepsilon^2 D_t f_\alpha^{(1)} + \frac{\varepsilon^2}{2} D_t^2 f_\alpha^{(0)} = \underbrace{-\frac{1}{\tau}\cancel{\varepsilon f_\alpha^{(1)}}}_{1} - \frac{1}{\tau}\varepsilon^2 f_\alpha^{(2)} + \underbrace{\frac{\varepsilon}{6}\cancel{\boldsymbol{e}_{\alpha i}\boldsymbol{F}_i}}_{1}$$

Al aplicar la ecuación (4.9):

$$D_t f_\alpha^{(1)} + \frac{1}{2} D_t \left(-\frac{1}{\tau} f_\alpha^{(1)} + \frac{1}{6}\boldsymbol{e}_{\alpha i}\boldsymbol{F}_i \right) = -\frac{1}{\tau} f_\alpha^{(2)}$$

Finalmente, lo obtenido puede reescribirse como:

$$\left(1 - \frac{1}{2\tau}\right) D_t f_\alpha^{(1)} = -\frac{1}{\tau} f_\alpha^{(2)} - \frac{1}{2} D_t \left(\frac{1}{6}\boldsymbol{e}_{\alpha i}\boldsymbol{F}_i \right) \tag{4.10}$$

Para continuar con el desarrollo, se hace necesario mencionar las siguientes restricciones:

$$\sum_{\alpha=0}^{8} f_\alpha^{(0)} = h \tag{4.11a}$$

$$\sum_{\alpha=0}^{8} \boldsymbol{e}_{\alpha i} f_\alpha^{(0)} = h\boldsymbol{u}_i \tag{4.11b}$$

$$\sum_{\alpha=0}^{8} \boldsymbol{e}_{\alpha i}\boldsymbol{e}_{\alpha j} f_\alpha^{(0)} = \frac{1}{2}\mathrm{g}h^2\delta_{ij} + h\boldsymbol{u}_i\boldsymbol{u}_j \tag{4.11c}$$

$$\sum_{\alpha=0}^{8} f_\alpha^{(n)} = 0, \qquad \forall n > 0 \tag{4.11d}$$

$$\sum_{\alpha=0}^{8} \boldsymbol{e}_{\alpha i} f_\alpha^{(n)} = 0, \qquad \forall n > 0 \tag{4.11e}$$

y es útil recordar que por la simetría representada en la ecuación (4.1) es posible obtener las siguientes igualdades:

$$\sum_{\alpha=0}^{8} \boldsymbol{e}_{\alpha i} = 0$$

$$\sum_{\alpha=0}^{8} \boldsymbol{e}_{\alpha i}\boldsymbol{e}_{\alpha j} = 6\delta_{ij}$$

$$\sum_{\alpha=0}^{8} \boldsymbol{e}_{\alpha i}\boldsymbol{e}_{\alpha j}\boldsymbol{e}_{\alpha k} = 0$$

Es así como al realizar la sumatoria $\sum \left[(4.8) + \varepsilon \times (4.9) + \varepsilon^2 \times (4.10)\right]$ en torno a α, se obtiene:

$$\underbrace{\sum_{\alpha=0}^{8} f_\alpha^{(0)}}_{1} + \varepsilon \sum_{\alpha=0}^{8} D_t f_\alpha^{(0)} +$$

$$\varepsilon^2 \left(1 - \frac{1}{2\tau}\right) \sum_{\alpha=0}^{8} D_t f_\alpha^{(1)} = \underbrace{\sum_{\alpha=0}^{8} f_\alpha^{(eq)}}_{1} - \varepsilon \frac{1}{\tau} \cancelto{0}{\sum_{\alpha=0}^{8} f_\alpha^{(1)}} + \varepsilon \frac{1}{6} \boldsymbol{F}_i \cancelto{0}{\sum_{\alpha=0}^{8} \boldsymbol{e}_{\alpha i}}$$

$$- \varepsilon^2 \frac{1}{\tau} \cancelto{0}{\sum_{\alpha=0}^{8} f_\alpha^{(2)}} - \varepsilon^2 \frac{1}{2} \sum_{\alpha=0}^{8} D_t \left(\frac{1}{6} \boldsymbol{e}_{\alpha i} \boldsymbol{F}_i\right)$$

Al eliminar los términos cancelados y aquellos cuyo valor es cero, simplificando por ε la ecuación se convierte en:

$$\sum_{\alpha=0}^{8} D_t f_\alpha^{(0)} + \varepsilon \left(1 - \frac{1}{2\tau}\right) \sum_{\alpha=0}^{8} D_t f_\alpha^{(1)} = -\varepsilon \frac{1}{2} \sum_{\alpha=0}^{8} D_t \left(\frac{1}{6} \boldsymbol{e}_{\alpha i} \boldsymbol{F}_i\right)$$

Expandiendo los términos se obtiene:

$$\frac{\partial}{\partial t} \left(\sum_{\alpha=0}^{8} f_\alpha^{(0)}\right) + \frac{\partial}{\partial \boldsymbol{x}_j} \left(\sum_{\alpha=0}^{8} \boldsymbol{e}_{\alpha j} f_\alpha^{(0)}\right) +$$

$$\varepsilon \left(1 - \frac{1}{2\tau}\right) \left[\frac{\partial}{\partial t} \left(\cancelto{0}{\sum_{\alpha=0}^{8} f_\alpha^{(1)}}\right) + \right.$$

$$\left. \frac{\partial}{\partial \boldsymbol{x}_j} \left(\cancelto{0}{\sum_{\alpha=0}^{8} \boldsymbol{e}_{\alpha j} f_\alpha^{(1)}}\right)\right] = -\varepsilon \frac{1}{12} \left[\frac{\partial}{\partial t} \left(\boldsymbol{F}_i \cancelto{0}{\sum_{\alpha=0}^{8} \boldsymbol{e}_{\alpha i}}\right)\right.$$

$$\left. + \frac{\partial}{\partial \boldsymbol{x}_j} \left(\boldsymbol{F}_i \cancelto{6\delta_{ij}}{\sum_{\alpha=0}^{8} \boldsymbol{e}_{\alpha j} \boldsymbol{e}_{\alpha i}}\right)\right]$$

Al reemplazar los valores correspondientes:

$$\frac{\partial}{\partial t} \left(\sum_{\alpha=0}^{8} f_\alpha^{(0)}\right) + \frac{\partial}{\partial \boldsymbol{x}_j} \left(\sum_{\alpha=0}^{8} \boldsymbol{e}_{\alpha j} f_\alpha^{(0)}\right) = -\frac{\varepsilon}{2} \frac{\partial \boldsymbol{F}_j}{\partial \boldsymbol{x}_j}$$

Si se considera precisión de primer orden para el término de fuerza, al aplicar las restricciones definidas en (4.11a) y (4.11b) se obtiene:

$$\frac{\partial h}{\partial t} + \frac{\partial \left(h\boldsymbol{u}_j\right)}{\partial \boldsymbol{x}_j} = 0$$

lo que corresponde a la ecuación de *shallow water* presentada en (2.1).

Del mismo modo, al calcular $\sum \boldsymbol{e}_{\alpha k}\left[(4.8) + \varepsilon \times (4.9) + \varepsilon^2 \times (4.10)\right]$ se tiene:

$$\cancel{\underbrace{\sum_{\alpha=0}^{8} \boldsymbol{e}_{\alpha k} f_\alpha^{(0)}}_{1}} + \varepsilon \sum_{\alpha=0}^{8} \boldsymbol{e}_{\alpha k} D_t f_\alpha^{(0)} +$$

$$\varepsilon^2 \left(1 - \frac{1}{2\tau}\right) \sum_{\alpha=0}^{8} \boldsymbol{e}_{\alpha k} D_t f_\alpha^{(1)} = \cancel{\underbrace{\sum_{\alpha=0}^{8} \boldsymbol{e}_{\alpha k} f_\alpha^{(eq)}}_{1}} - \varepsilon \frac{1}{\tau} \cancelto{0}{\sum_{\alpha=0}^{8} \boldsymbol{e}_{\alpha k} f_\alpha^{(1)}}$$

$$+ \varepsilon \frac{1}{6} \boldsymbol{F}_i \cancelto{6\delta_{ik}}{\sum_{\alpha=0}^{8} \boldsymbol{e}_{\alpha k} \boldsymbol{e}_{\alpha i}} - \varepsilon^2 \frac{1}{\tau} \cancelto{0}{\sum_{\alpha=0}^{8} \boldsymbol{e}_{\alpha k} f_\alpha^{(2)}}$$

$$- \varepsilon^2 \frac{1}{2} \sum_{\alpha=0}^{8} \boldsymbol{e}_{\alpha k} D_t \left(\frac{1}{6} \boldsymbol{e}_{\alpha i} \boldsymbol{F}_i\right)$$

Evaluando los términos, eliminando los cancelados y simplificando por ε se obtiene:

$$\sum_{\alpha=0}^{8} \boldsymbol{e}_{\alpha k} D_t f_\alpha^{(0)} + \varepsilon \left(1 - \frac{1}{2\tau}\right) \sum_{\alpha=0}^{8} \boldsymbol{e}_{\alpha k} D_t f_\alpha^{(1)} = \boldsymbol{F}_k - \varepsilon \frac{1}{2} \sum_{\alpha=0}^{8} \boldsymbol{e}_{\alpha k} D_t \left(\frac{1}{6} \boldsymbol{e}_{\alpha i} \boldsymbol{F}_i\right)$$

Al expandir:

$$\frac{\partial}{\partial t}\left(\sum_{\alpha=0}^{8} \boldsymbol{e}_{\alpha k} f_\alpha^{(0)}\right) +$$

$$\frac{\partial}{\partial \boldsymbol{x}_j}\left(\sum_{\alpha=0}^{8} \boldsymbol{e}_{\alpha k} \boldsymbol{e}_{\alpha j} f_\alpha^{(0)}\right) +$$

$$\varepsilon \left(1 - \frac{1}{2\tau}\right) \left[\frac{\partial}{\partial t}\left(\cancelto{0}{\sum_{\alpha=0}^{8} \boldsymbol{e}_{\alpha k} f_\alpha^{(1)}}\right) +\right.$$

$$\left.\frac{\partial}{\partial \boldsymbol{x}_j}\left(\sum_{\alpha=0}^{8} \boldsymbol{e}_{\alpha k} \boldsymbol{e}_{\alpha j} f_\alpha^{(1)}\right)\right] = \boldsymbol{F}_k - \varepsilon \frac{1}{12} \left[\frac{\partial}{\partial t}\left(\boldsymbol{F}_i \cancelto{6\delta_{ik}}{\sum_{\alpha=0}^{8} \boldsymbol{e}_{\alpha k} \boldsymbol{e}_{\alpha i}}\right)\right.$$

$$\left. + \frac{\partial}{\partial \boldsymbol{x}_j}\left(\boldsymbol{F}_j \cancelto{0}{\sum_{\alpha=0}^{8} \boldsymbol{e}_{\alpha k} \boldsymbol{e}_{\alpha j} \boldsymbol{e}_{\alpha i}}\right)\right]$$

Finalmente, evaluando se llega a:

$$\frac{\partial}{\partial t}\left(\sum_{\alpha=0}^{8}\boldsymbol{e}_{\alpha k}f_{\alpha}^{(0)}\right)+$$
$$\frac{\partial}{\partial \boldsymbol{x}_j}\left(\sum_{\alpha=0}^{8}\boldsymbol{e}_{\alpha k}\boldsymbol{e}_{\alpha j}f_{\alpha}^{(0)}\right)=\boldsymbol{F}_k-\frac{\varepsilon}{2}\frac{\partial \boldsymbol{F}_k}{\partial t}-\frac{\varepsilon}{2\tau}(2\tau-1)\frac{\partial}{\partial \boldsymbol{x}_j}\left(\sum_{\alpha=0}^{8}\boldsymbol{e}_{\alpha k}\boldsymbol{e}_{\alpha j}f_{\alpha}^{(1)}\right)$$

Con lo que, nuevamente considerando precisión de primer orden para el término de fuerza, al reemplazar las restricciones (4.11b) y (4.11c) se obtiene:

$$\frac{\partial h\boldsymbol{u}_i}{\partial t}+\frac{\partial\left(h\boldsymbol{u}_i\boldsymbol{u}_j\right)}{\partial \boldsymbol{x}_j}=-\mathrm{g}\frac{\partial}{\partial \boldsymbol{x}_i}\left(\frac{h^2}{2}\right)-\frac{\partial}{\partial \boldsymbol{x}_j}\Lambda_{ij}+\boldsymbol{F}_i \tag{4.13}$$

en donde $\boldsymbol{F}_i$ se expresa como:

$$\boldsymbol{F}_i=-\mathrm{g}h\frac{\partial b}{\partial \boldsymbol{x}_i} \tag{4.14}$$

recuperando las ecuaciones de *shallow water* presentadas en (2.2) y (2.3). La única diferencia visible respecto a dichas ecuaciones es la presencia del término dependiente de Λ_{ij}, el cual representa el efecto de la viscosidad del fluido y puede ser agregado al modelo sin mayores complicaciones. Según el desarrollo realizado se tiene:

$$\Lambda_{ij}=\frac{\varepsilon}{2\tau}(2\tau-1)\sum_{\alpha=0}^{8}\boldsymbol{e}_{\alpha k}\boldsymbol{e}_{\alpha j}f_{\alpha}^{(1)}$$

Tomando como referencia el trabajo de Zhou [37], este término puede expresarse como:

$$\Lambda_{ij}\approx-\nu\left[\frac{\partial(hu_i)}{\partial x_j}+\frac{\partial(hu_j)}{\partial x_i}\right]$$

donde ν es la viscosidad cinemática y se define de la siguiente manera:

$$\nu=\frac{\Delta t}{6}(2\tau-1)$$

Finalmente, la ecuación (4.13) puede reescribirse como:

$$\frac{\partial h\boldsymbol{u}_i}{\partial t}+\frac{\partial\left(h\boldsymbol{u}_i\boldsymbol{u}_j\right)}{\partial \boldsymbol{x}_j}=-\mathrm{g}\frac{\partial}{\partial \boldsymbol{x}_i}\left(\frac{h^2}{2}\right)+\nu\frac{\partial^2(hu_i)}{\partial \boldsymbol{x}_j^2}+\boldsymbol{F}_i$$

4.2.2. Segunda Función de Distribución de Equilibrio ($g^{(eq)}$)

La segunda función de equilibrio utilizada fue propuesta por Shaotian Li et al. [6] y se define como:

$$g_{\alpha}^{(eq)}=\begin{cases}h-\dfrac{2h}{3}\boldsymbol{u}_i^2, & \alpha=0\\ \dfrac{h}{3}\boldsymbol{e}_{\alpha i}\boldsymbol{u}_i+\dfrac{h}{2}\boldsymbol{e}_{\alpha i}\boldsymbol{e}_{\alpha j}\boldsymbol{u}_i\boldsymbol{u}_j-\dfrac{h}{6}\boldsymbol{u}_i^2, & \alpha=1,2,3,4\\ \dfrac{h}{12}\boldsymbol{e}_{\alpha i}\boldsymbol{u}_i+\dfrac{h}{8}\boldsymbol{e}_{\alpha i}\boldsymbol{e}_{\alpha j}\boldsymbol{u}_i\boldsymbol{u}_j-\dfrac{h}{24}\boldsymbol{u}_i^2, & \alpha=5,6,7,8\end{cases} \tag{4.15}$$

Como es posible apreciar, el único cambio introducido en está función respecto a la primera analizada es la eliminación de los términos que dependen de la aceleración de gravedad g. Esto solo supone un cambio en la restricción presentada en (4.11c), modificándola de la siguiente manera:

$$\sum_{\alpha=0}^{8}\boldsymbol{e}_{\alpha i}\boldsymbol{e}_{\alpha j}g_{\alpha}^{(0)}=h\boldsymbol{u}_i\boldsymbol{u}_j$$

Como consecuencia de esto, desaparece el primer término del lado derecho en la ecuación (4.13), transformándose en:

$$\frac{\partial h\boldsymbol{u}_i}{\partial t} + \frac{\partial \left(h\boldsymbol{u}_i\boldsymbol{u}_j\right)}{\partial \boldsymbol{x}_j} = -\frac{\partial}{\partial \boldsymbol{x}_j}\Lambda_{ij} + \boldsymbol{F}_i$$

Es por esto que no se hace necesaria la aplicación de la expansión de Chapman-Enskog, pues basta con agregar este término faltante a $\boldsymbol{F}_i$ para que el resultado sea el mismo y nuevamente se recupere la ecuación de *shallow water* correcta. Por lo tanto, el nuevo término de fuerza corresponde a:

$$\boldsymbol{F}_i = -\mathrm{g}h\frac{\partial b}{\partial \boldsymbol{x}_i} - \mathrm{g}h\frac{\partial h}{\partial \boldsymbol{x}_i} = \mathrm{g}h\frac{\partial (b+h)}{\partial \boldsymbol{x}_i} \tag{4.16}$$

Otro cambio que debe introducirse al utilizar esta función de distribución de equilibrio es la aparición del término fuente expresado como:

$$S_\alpha = \begin{cases} 0, & \alpha = 0 \\ \dfrac{1}{6\boldsymbol{e}_{\alpha i}\boldsymbol{e}_{\alpha i}}\boldsymbol{e}_{\alpha i}\boldsymbol{F}_{\alpha i}, & \alpha = 1,\ldots,8 \end{cases} \tag{4.17}$$

donde $\boldsymbol{F}_{\alpha i}$ corresponde a la i-ésima componente del término de fuerza escalada por un peso ω_α, es decir:

$$\boldsymbol{F}_{\alpha i} = \omega_\alpha \boldsymbol{F}_i$$

Esta nueva adición al modelo hace necesario reescribir la ecuación de *Lattice Boltzmann* en (4.6) de la siguiente manera:

$$g_\alpha(\boldsymbol{x} + \boldsymbol{e}_\alpha\varepsilon, t+\varepsilon) = g_\alpha(\boldsymbol{x}, t) - \frac{\left[g_\alpha(\boldsymbol{x}, t) - g_\alpha^{(eq)}(\boldsymbol{x}, t)\right]}{\tau} + \varepsilon S_\alpha \tag{4.18}$$

donde g corresponde a la función de distribución del método cuando es utilizada la función de equilibrio $g^{(eq)}$, no debiendo confundirse con la aceleración de gravedad g.

A diferencia de lo ocurrido con la primera función de distribución de equilibrio presentada, en donde $N_\alpha = 6$ para todo valor de α, el término fuente definido en (4.17) corresponde a $\frac{1}{6}\boldsymbol{e}_{\alpha i}\boldsymbol{F}_{\alpha i}$ para $\alpha = 1,\ldots,4$ y $\frac{1}{12}\boldsymbol{e}_{\alpha i}\boldsymbol{F}_{\alpha i}$ para $\alpha = 5,\ldots,8$. Por este motivo, es necesario definir dos nuevas restricciones para que la expansión de Chapman-Enskog sea totalmente equivalente:

$$\sum_{\alpha=0}^{8} S_\alpha = 0 \tag{4.19a}$$

$$\sum_{\alpha=0}^{8} \boldsymbol{e}_{\alpha i} S_\alpha = F_i \tag{4.19b}$$

La satisfacción de dichas restricciones puede lograrse mediante una correcta elección de los pesos ω_α antes mencionados. De esta forma, se tiene:

$$\begin{aligned}\sum_{\alpha=0}^{8} S_\alpha &= S_0 + S_1 + S_2 + S_3 + S_4 + S_5 + S_6 + S_7 + S_8 \\ &= 0 + \frac{1}{6}\left(\omega_1\boldsymbol{F}_1 + \omega_2\boldsymbol{F}_2 - \omega_3\boldsymbol{F}_1 - \omega_4\boldsymbol{F}_2\right) + \frac{1}{12}\left[\omega_5(\boldsymbol{F}_1 + \boldsymbol{F}_2)\right. \\ &\quad \left. + \omega_6(-\boldsymbol{F}_1 + \boldsymbol{F}_2) + \omega_7(-\boldsymbol{F}_1 - \boldsymbol{F}_2) + \omega_8(\boldsymbol{F}_1 - \boldsymbol{F}_2)\right]\end{aligned}$$

Al definir los pesos $\omega_a = \omega_1 = \omega_2 = \omega_3 = \omega_4$ y $\omega_b = \omega_5 = \omega_6 = \omega_7 = \omega_8$, la anterior ecuación puede reescribirse como:

$$\begin{aligned}\sum_{\alpha=0}^{8} S_\alpha &= \frac{\omega_a}{6}\left(\boldsymbol{F}_1 + \boldsymbol{F}_2 - \boldsymbol{F}_1 - \boldsymbol{F}_2\right) \\ &\quad + \frac{\omega_b}{12}\left[(\boldsymbol{F}_1 + \boldsymbol{F}_2) + (-\boldsymbol{F}_1 + \boldsymbol{F}_2) + (-\boldsymbol{F}_1 - \boldsymbol{F}_2) + (\boldsymbol{F}_1 - \boldsymbol{F}_2)\right] \\ &= 0\end{aligned}$$

Con lo que se cumple (4.19a). Del mismo modo, se tiene lo siguiente:

$$\begin{aligned}\sum_{\alpha=0}^{8} \boldsymbol{e}_{\alpha i} S_\alpha &= 0 + \sum_{\alpha=1}^{4} \boldsymbol{e}_{\alpha i} S_\alpha + \sum_{\alpha=5}^{8} \boldsymbol{e}_{\alpha i} S_\alpha \\ &= \frac{1}{6}\boldsymbol{F}_j \sum_{\alpha=1}^{4} \boldsymbol{e}_{\alpha i}\boldsymbol{e}_{\alpha j}\omega_\alpha + \frac{1}{12}\boldsymbol{F}_j \sum_{\alpha=5}^{8} \boldsymbol{e}_{\alpha i}\boldsymbol{e}_{\alpha j}\omega_\alpha \\ &= \frac{2\omega_a}{6}\boldsymbol{F}_j\delta_{ij} + \frac{4\omega_b}{12}\boldsymbol{F}_j\delta_{ij} \\ &= \frac{\omega_a + \omega_b}{3}\boldsymbol{F}_i\end{aligned}$$

por lo que $\omega_a + \omega_b = 3$ para que se satisfaga (4.19b).

Finalmente, utilizando pesos que cumplan con las restricciones descritas anteriormente, se logra que la adición del término fuente al modelo no afecte la expansión de Chapman-Enskog, obteniendo así un método que resuelve las ecuaciones de *shallow water*.

4.3. Condiciones de Borde

Al resolver una ecuación diferencial, lo que realmente se obtiene es una familia de soluciones que satisfacen dicha ecuación, por lo que se hace necesario especificar condiciones iniciales del problema y/o condiciones en la frontera del dominio para obtener una solución única que se ajuste al caso particular. Al utilizar el método de *Lattice Boltzmann* para resolver las ecuaciones de *shallow water*, las condiciones iniciales se especifican simplemente por el hecho de otorgarle al algoritmo los datos de altura de agua, forma de la batimetría y velocidades macroscópicas en todo el dominio al tiempo $t = 0$. Por otro lado, como en la mayoría de los problemas de simulación, las condiciones de borde o de frontera son mucho más complicadas de implementar, debido a que dependen tanto de la representación como del método utilizado, y a que varían dependiendo del comportamiento que se desea reproducir.

En *Lattice Boltzmann*, la implementación de condiciones de borde se reduce a encontrar las tres componentes faltantes de la función de distribución, ya que no se pueden obtener vía *streaming* debido a la ausencia de nodos adyacentes. Esta situación se ve representada en la Figura 4.2, en donde las componentes faltantes están simbolizadas por flechas segmentadas.

A continuación, se presentarán los tres tipos de condiciones de borde más utilizadas para el problema que se intenta solucionar.

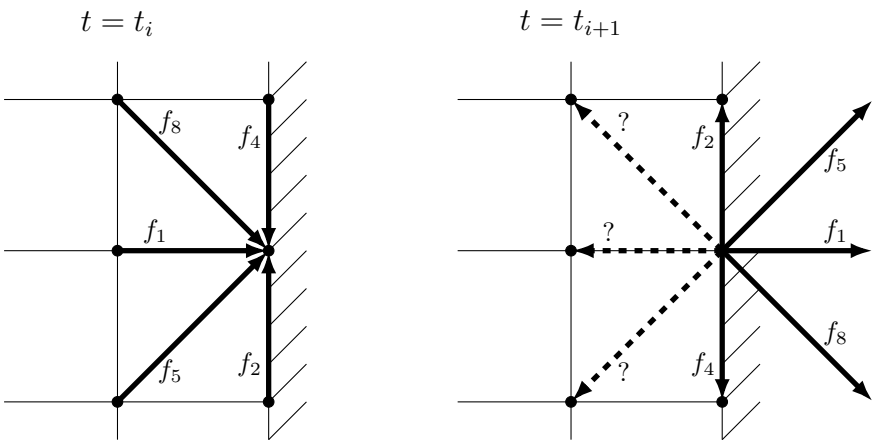


Figura 4.2: Componentes faltantes de f.

4.3.1. Condiciones Periódicas

Este tipo de condiciones de frontera es comúnmente la más sencilla de implementar. Lo que se intenta representar con su uso es un dominio infinito, al unir bordes opuestos de manera que el fin del dominio en uno de ellos signifique el comienzo del mismo en el otro. En el problema que se está resolviendo son de gran utilidad al simular canales infinitos cuya forma de la batimetría se repite cada cierta longitud del dominio, es decir, es periódica.

Un ejemplo básico para entender el comportamiento deseado de estas condiciones de frontera sería una onda de fluido que viaja a lo largo del dominio con una determinada dirección. Al llegar al borde, desaparece a través de éste, pero apareciendo por el extremo opuesto, continuando su viaje con la misma dirección.

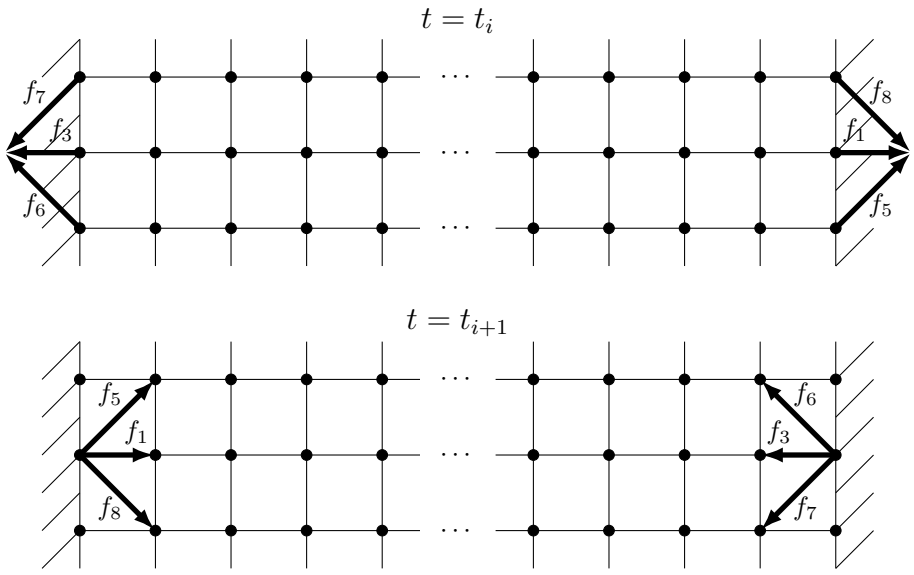


Figura 4.3: Condiciones de borde periódicas.

Como en la mayoría de los métodos, la implementación de condiciones de borde periódicas supone

asignar un nodo situado en el borde como nodo adyacente a aquel que se encuentra en el borde opuesto a la misma altura. Esto puede visualizarse en la Figura 4.3.

4.3.2. Condiciones de Tipo *Bounce-Back*

Las condiciones de borde de tipo *Bounce-Back* representan la presencia de un muro u objeto sólido que impide el paso de fluido a través de él. Esto supone una velocidad nula en la frontera del dominio, por lo que su implementación se reduce a reflejar las componentes de la función de distribución en dirección opuesta, como es posible apreciar en la Figura 4.4.

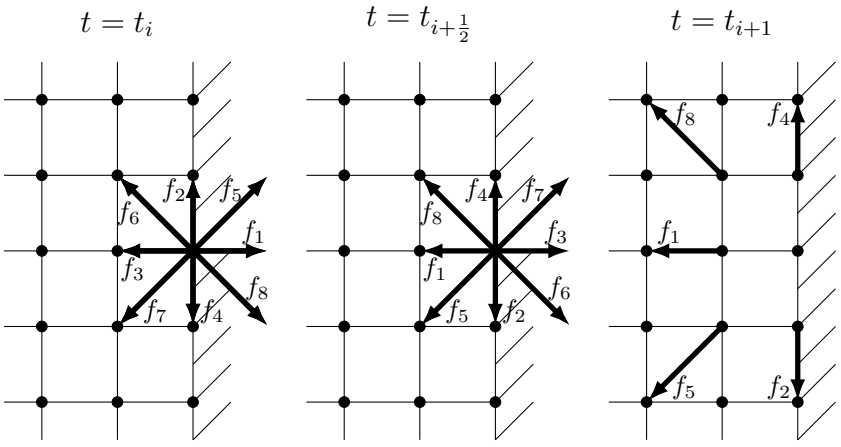


Figura 4.4: Condiciones de borde *Bounce-Back*.

Es importante mencionar que esta reflexión no se realiza en *streaming*, sino en un paso intermedio, por lo que si bien no corresponde a una discretización más fina del tiempo, es útil la representación de un tiempo medio para efectos de visualización.

Para este tipo de condiciones de borde, un ejemplo básico que ayuda a su comprensión sería nuevamente una onda de fluido que viaja a través del dominio, pero en esta ocasión, al llegar al borde rebotaría, adquiriendo una dirección de movimiento reflejada respecto a la anterior, siendo el ángulo de incidencia igual al ángulo reflejado.

4.3.3. Condiciones Abiertas

Las condiciones de borde abiertas representan el caso más común y necesario de comportamiento en la frontera. Como su nombre lo dice, este tipo de condiciones de borde significan un dominio abierto, lo que en palabras simples se puede describir como permitir al fluido seguir su curso como si el dominio continuara a pesar de que éste no se encuentra representado en la simulación.

De forma análoga a los casos anteriormente estudiados, un ejemplo simple que permite comprender el comportamiento de este tipo de condiciones de frontera sería una onda de fluido que viaja a lo largo del dominio y al llegar al borde desaparece a través de éste, sin reflejarse o volver a aparecer por algún otro extremo.

En cuanto al método de *Lattice Boltzmann* refiere, no se ha llegado aun a un consenso sobre la implementación de este tipo de condiciones de borde, pues diversos intentos han llevado a inestabilidades numéricas en ciertos casos. De esta forma, se utilizará la implementación más aceptada por presentar mayor estabilidad. Ésta corresponde a una copia de las componentes faltantes de la función de distribución desde el nodo adyacente interno, lo que puede visualizarse de manera más clara en la

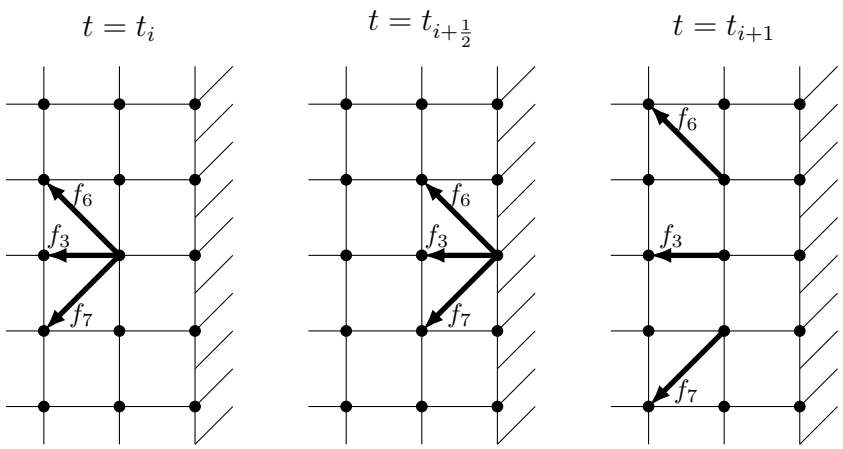


Figura 4.5: Condiciones de borde abiertas.

Figura 4.5. Al igual que las condiciones de tipo *Bounce-Back*, esto se ejecuta en un paso intermedio, por lo que se muestra un tiempo medio para efectos de visualización.

Capítulo 5

Propuesta de Solución

Como ya se mencionó anteriormente, la solución propuesta consiste en un nuevo modelo que combina el uso de las funciones de equilibrio definidas en (4.5) y (4.15), las cuales fueron denominadas $f^{(eq)}$ y $g^{(eq)}$ respectivamente. Si bien no es común utilizar más de una función de equilibrio en el método de *Lattice Boltzmann*, esta decisión fue tomada para enfrentar el mayor problema surgido durante la implementación de la solución y la posterior experimentación.

La principal diferencia entre ambas funciones recae en su comportamiento ante dos situaciones muy necesarias. Por un lado, el problema con el que se debe lidiar al utilizar $f^{(eq)}$ corresponde a que la aproximación de la derivada de h en el término

$$-\mathrm{g}h\frac{\partial h}{\partial \boldsymbol{x}_i}$$

se obtiene a partir de los términos presentes en la función de distribución de equilibrio, lo cual no es equivalente a la aproximación de la derivada de la batimetría que se calcula en el término de fuerza, cuya estimación se realiza mediante diferencias finitas. Esto genera inestabilidades numéricas en caso de existir batimetría no plana. El caso más simple para ejemplificar esta situación es considerar un nivel de agua constante con velocidad nula. Como se vio en la Sección 2.2, para que el método sea "*well-balanced*" se requiere que $w = h + b$ se mantenga constante a lo largo del tiempo.

Debido a que, según la definición del caso, todos los nodos poseen inicialmente el mismo valor de w, es posible afirmar que:

$$\frac{\partial w}{\partial \boldsymbol{x}_i} = \frac{\partial (h+b)}{\partial \boldsymbol{x}_i} = 0$$

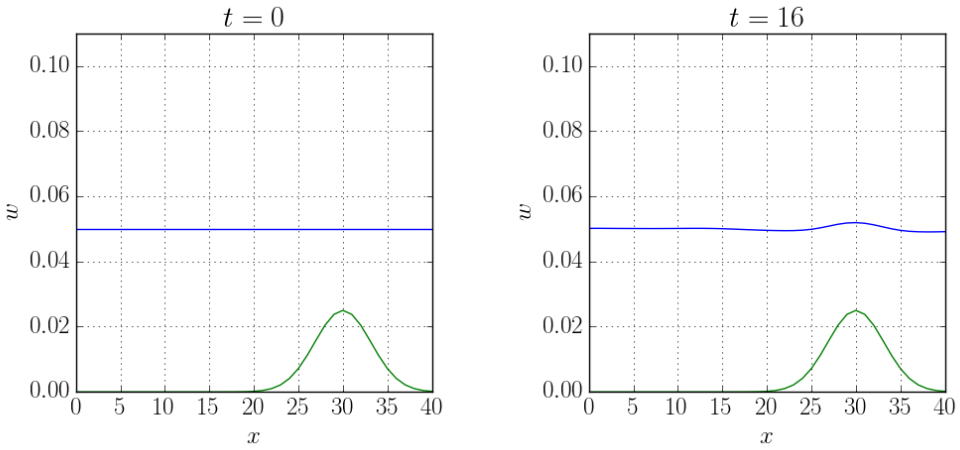
por lo que en realidad, para que se mantenga el estado estacionario, solo se necesita el cumplimiento de lo siguiente:

$$-\frac{\partial h}{\partial \boldsymbol{x}_i} = \frac{\partial b}{\partial \boldsymbol{x}_i}$$

proveniendo efectivamente estos valores desde la función de distribución de equilibrio y el término de fuerza (ver ecuaciones (4.13) y (4.14)).

Si bien esto es analíticamente correcto, al utilizar distintas aproximaciones numéricas en un dominio discretizado, esta igualdad no se cumple, generándose pequeñas perturbaciones indeseadas como puede verse en la Figura 5.1.

En el otro extremo, al utilizar la función $g^{(eq)}$, las estimaciones de las derivadas de b y h se realizan de forma equivalente, calculándose incluso de manera conjunta dentro de $\boldsymbol{F}_i$, como es posible apreciar en (4.16). Esto es esencial, pues evita la presencia de las perturbaciones generadas al usar $f^{(eq)}$ y permite de esta forma obtener un método "*well-balanced*".



(a) Altura de agua w constante al inicio de la simulación

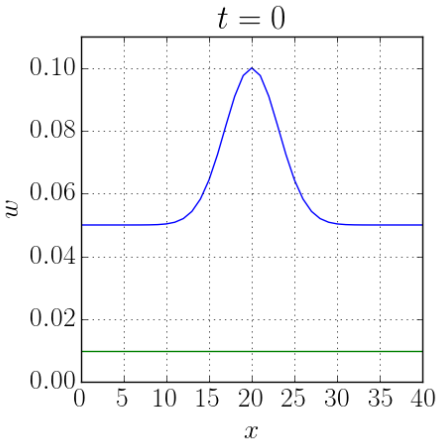
(b) Perturbaciones generadas en un tiempo más avanzado

Figura 5.1: Resultados al usar $f^{(eq)}$ en tiempos $t = 0$ y $t = 16$. La línea azul representa el nivel del agua w y en verde se visualiza la batimetría b.

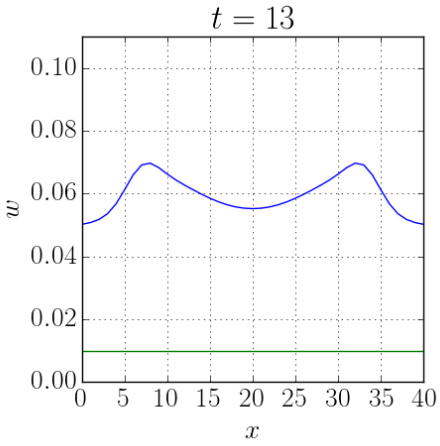
A pesar de que esto pareciera solucionar el conflicto, también existe un problema no menor al decidir utilizar $g^{(eq)}$, el cual corresponde al comportamiento del modelo frente a condiciones de borde abiertas.

Al utilizar este tipo de condiciones de borde, implementadas según se explicó en la Sección 4.3.3, en conjunto con dicha función de distribución de equilibrio, se generan grandes inestabilidades que no permiten una correcta simulación. Tales inestabilidades no corresponden a pequeñas perturbaciones, sino a oscilaciones que explotan exponencialmente en amplitud, como es posible visualizar en la Figura 5.2.

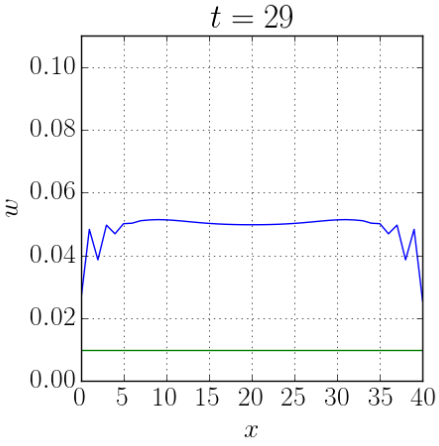
Es importante destacar que la función $f^{(eq)}$ soporta totalmente esta implementación de condiciones de borde abiertas, no presentando el comportamiento obtenido al utilizar $g^{(eq)}$. En este contexto, en donde la ventaja de una función es la desventaja de la otra, pero ambas características son deseables y requeridas, se hace necesaria una solución que permita evitar la decisión de optar por una de ellas a cambio de renunciar a la otra.



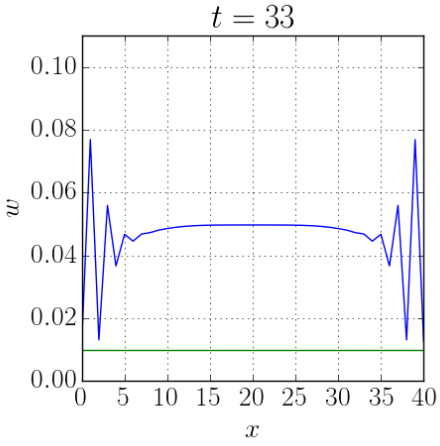
(a) La simulación comienza con un levantamiento de agua al centro del dominio



(b) Se generan dos ondas iguales que se mueven en direcciones opuestas hacia los bordes



(c) Al llegar a los bordes, se comienzan a generar oscilaciones indeseadas

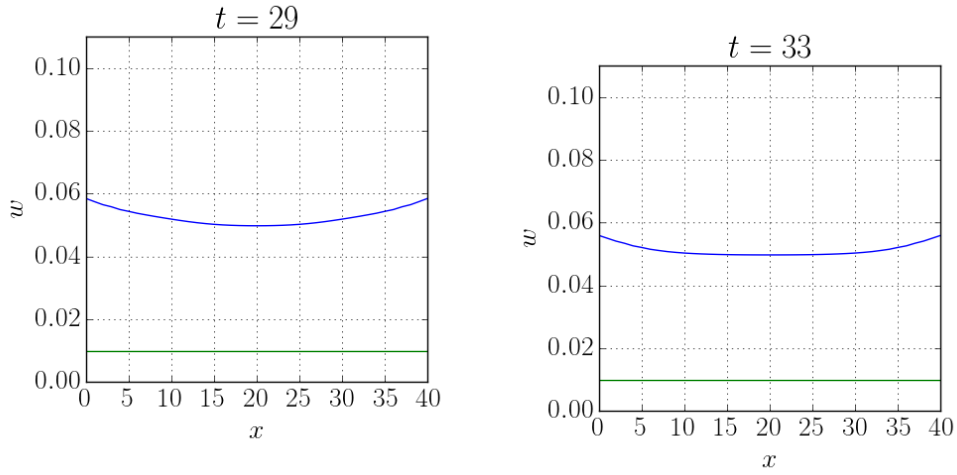


(d) Pocos *time steps* después, el tamaño de las oscilaciones es mucho mayor

Figura 5.2: Comportamiento de $g^{(eq)}$ frente a condiciones de borde abiertas.

Debido a todo esto, como ya se anunció en varias ocasiones, es que se decide utilizar ambas funciones de distribución de equilibrio. De esta forma, para asegurar el funcionamiento de las condiciones de borde abiertas, los nodos ubicados en la frontera donde se desea implementar este tipo de condición presentan la función de equilibrio $f^{(eq)}$, mientras que el resto de los nodos que componen el dominio utilizan la función $g^{(eq)}$. Como se aprecia en la Figura 5.3, esta modificación permite manejar las condiciones de borde deseadas, no presentando las inestabilidades vistas anteriormente, a pesar de ser exactamente la misma simulación. Sin embargo, es posible notar que aun con esta implementación se generarán inestabilidades en la frontera si existe batimetría no plana, siendo este un caso bastante común, por lo que esta opción solo supone una solución parcial al problema.

La segunda decisión adoptada corresponde a la generación de un dominio fantasma, es decir, la



(a) Correcto funcionamiento de condiciones de borde abiertas

(b) No se genera ningún tipo de inestabilidad

Figura 5.3: Condiciones de borde abiertas con $f^{(eq)}$ solo en los nodos de frontera. Imágenes correspondientes a 5.2c y 5.2d, siendo la misma simulación.

adición de nodos más allá de la frontera. El objetivo de este dominio agregado es que en él la batimetría se suavice hasta llegar a una forma plana, para conseguir el correcto comportamiento de las condiciones de borde abiertas sin que exista la preocupación por la existencia de una batimetría con relieve. Un ejemplo del efecto que produce este dominio fantasma en la batimetría es presentado en la Figura 5.4.

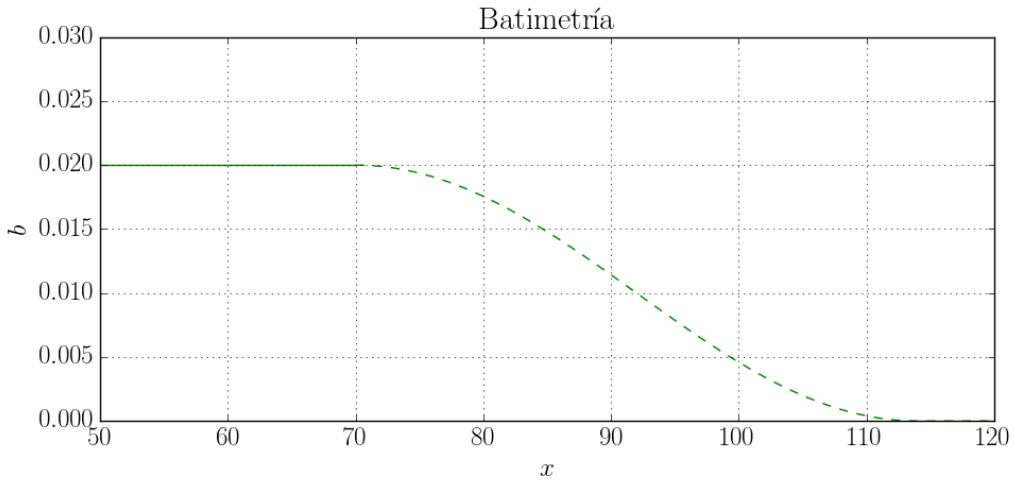


Figura 5.4: Efecto del dominio fantasma (línea verde segmentada) en la batimetría. La línea sólida corresponde al dominio real.

Es importante mencionar que, sin importar la forma que presente la batimetría en esta extensión del dominio, la solución al problema sigue siendo válida en el dominio real. La función elegida para suavizar la batimetría corresponde a una sinusoidal al cuadrado ($\sin(\cdot)^2$), dejando siete nodos iguales a cero en el extremo del dominio fantasma con el fin de asegurar un vecindario plano a los nodos en la frontera. Cabe destacar que el método implementado es sensible a derivadas muy altas de la batimetría, por lo que el largo del dominio añadido depende proporcionalmente de la máxima altura de batimetría registrada en la frontera del dominio real, evitando así inclinaciones cercanas a la verticalidad.

En la Figura 5.5 se muestra un esquema del dominio completo de la simulación utilizando condiciones de borde abiertas en las fronteras este y oeste.

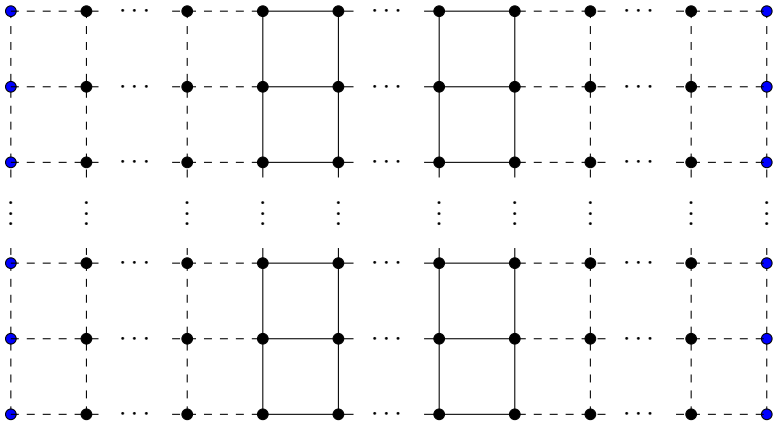


Figura 5.5: Dominio completo de la simulación con condiciones de borde abiertas.
El dominio real se representa mediante líneas solidas, mientras que el dominio fantasma corresponde a las líneas segmentadas. Nodos azules utilizan $f^{(eq)}$, mientras que los negros utilizan $g^{(eq)}$.

Capítulo 6

Algoritmo

6.1. Parámetros Iniciales

A continuación se listarán los parámetros que necesita recibir el algoritmo para comenzar la simulación:

- L_x y L_y: dimensiones del dominio en el eje horizontal y vertical respectivamente. Los índices de los nodos (i, j) poseen los rangos $i = 0, \ldots, L_x - 1$ y $j = 0, \ldots, L_y - 1$.
- T_{MAX}: tiempo máximo por el que se ejecutará la simulación. Adicionalmente, pueden incluirse otras condiciones para detener el algoritmo, como la presencia de un estado estacionario, a partir de la cual, se sabe que la solución no será modificada con el paso del tiempo.
- τ: tiempo de relajación, el cual se utiliza en el paso de *streaming-colisión* como puede verse en la ecuación (4.4).
- g: aceleración de gravedad que se utilizará en la simulación.
- b: arreglo bidimensional que contiene la altura de la batimetría en cada nodo del dominio.
- w: arreglo bidimensional con la altura del agua en cada nodo del dominio. Con los valores de w y b se calcula el valor de h de acuerdo a $h_{ij} = w_{ij} - b_{ij}$.
- $\boldsymbol{u}_x$ y $\boldsymbol{u}_y$: arreglos bidimensionales con las velocidades horizontal y vertical respectivamente en cada nodo del dominio.
- $\hat{\boldsymbol{F}}_x$ y $\hat{\boldsymbol{F}}_y$: componentes horizontal y vertical de los términos constantes que forman parte del vector de fuerza $\boldsymbol{F}$. Estos valores solo serán distintos de cero cuando se requiera implementar fuerzas externas, como la fuerza Coriolis o el efecto de roce del viento, entre otras posibles. De esta forma, es posible representar el término de fuerza de la ecuación de *Lattice Boltzmann* como:

  $$\boldsymbol{F}_i = -\mathrm{g}h\frac{\partial b}{\partial x_i} + \hat{\boldsymbol{F}}_i \qquad \text{o} \qquad \boldsymbol{F}_i = -\mathrm{g}h\frac{\partial (b+h)}{\partial x_i} + \hat{\boldsymbol{F}}_i$$

  dependiendo de si se está utilizando la función de distribución de equilibrio $f^{(eq)}$ o $g^{(eq)}$ respectivamente.

6.2. Estructura del Algoritmo

El algoritmo implementado consiste en un bucle principal que realiza llamadas a subrutinas, las cuales se encargan de ejecutar los pasos presentes en el metodo de *Lattice Boltzmann*. Es importante mencionar que algunas funciones no serán mencionadas debido a que son externas al método, como por ejemplo la lectura de datos y la generación de gráficos, o bien porque su implementación difiere de acuerdo al comportamiento deseado, como es el caso de las condiciones de borde, las cuales serán explicadas en la Sección 6.3.

La estructura medular del algoritmo puede verse en el Algoritmo 1.

Algoritmo 1 Estructura principal.

Entrada: $L_x, L_y, T_{MAX}, \tau, \mathrm{g}, b, w, \boldsymbol{u}_x, \boldsymbol{u}_y, \hat{\boldsymbol{F}}_x, \hat{\boldsymbol{F}}_y$
Salida:
1: $t \leftarrow 0$
2: $\boldsymbol{e}_x, \boldsymbol{e}_y, h, g, g^{eq}, S \leftarrow$ Setup$(L_x, L_y, \mathrm{g}, b, w, \boldsymbol{u}_x, \boldsymbol{u}_y, \hat{\boldsymbol{F}}_x, \hat{\boldsymbol{F}}_y)$
3: **mientras** $t \leq T_{MAX}$ **hacer**
4: $\quad g \leftarrow$ Streaming-Colision$(L_x, L_y, \tau, \boldsymbol{e}_x, \boldsymbol{e}_y, g, g^{eq}, S)$
5: $\quad h, \boldsymbol{u}_x, \boldsymbol{u}_y \leftarrow$ Solucion$(L_x, L_y, \boldsymbol{e}_x, \boldsymbol{e}_y, g)$
6: $\quad g^{eq} \leftarrow$ FEQ$(L_x, L_y, \boldsymbol{e}_x, \boldsymbol{e}_y, h, \boldsymbol{u}_x, \boldsymbol{u}_y)$
7: $\quad S \leftarrow$ TerminoFuente$(L_x, L_y, \mathrm{g}, b, h, \boldsymbol{e}_x, \boldsymbol{e}_y, \hat{\boldsymbol{F}}_x, \hat{\boldsymbol{F}}_y)$
8: **fin mientras**

La subrutina Setup corresponde a la inicialización de todas las variables necesarias para comenzar el ciclo que no son dadas como parámetro. Como se aprecia en el Algoritmo 2, esta función se encarga de generar los vectores $\boldsymbol{e}_x$ y $\boldsymbol{e}_y$, calcular h, y determinar los valores de g, g^{eq} y S para la primera iteración. Cabe mencionar que, en el comienzo, la función de distribución g es solo una copia de la función de distribución de equilibrio g^{eq}.

Algoritmo 2 Subrutina Setup.

Entrada: $L_x, L_y, \mathrm{g}, b, w, \boldsymbol{u}_x, \boldsymbol{u}_y, \hat{\boldsymbol{F}}_x, \hat{\boldsymbol{F}}_y$
Salida: $\boldsymbol{e}_x, \boldsymbol{e}_y, h, g, g^{eq}, S$
1: $\boldsymbol{e}_x \leftarrow [0, 1, 0, -1, 0, 1, -1, -1, 1]$
2: $\boldsymbol{e}_y \leftarrow [0, 0, 1, 0, -1, 1, 1, -1, -1]$
3: **para todo** i **en** $\{0, \ldots, L_x - 1\}$ **hacer**
4: $\quad$ **para todo** j **en** $\{0, \ldots, L_y - 1\}$ **hacer**
5: $\quad\quad h_{ij} \leftarrow w_{ij} - b_{ij}$
6: $\quad$ **fin para**
7: **fin para**
8: $g^{eq} \leftarrow$ FEQ$(L_x, L_y, \boldsymbol{e}_x, \boldsymbol{e}_y, h, \boldsymbol{u}_x, \boldsymbol{u}_y)$
9: $g \leftarrow g^{eq}$
10: $S \leftarrow$ TerminoFuente$(L_x, L_y, \mathrm{g}, b, h, \boldsymbol{e}_x, \boldsymbol{e}_y, \hat{\boldsymbol{F}}_x, \hat{\boldsymbol{F}}_y)$

La primera función llamada desde el ciclo principal del algoritmo es Streaming-Colision. Este procedimiento, presentado en el Algoritmo 3, se encarga de realizar conjuntamente los pasos de *streaming* y *colisión* del método de *Lattice Boltzmann*.

Algoritmo 3 Subrutina Streaming-Colision.

Entrada: $L_x, L_y, \tau, \boldsymbol{e}_x, \boldsymbol{e}_y, g, g^{eq}, S$
Salida: g

para todo i **en** $\{0, \ldots, L_x - 1\}$ **hacer**
  para todo j **en** $\{0, \ldots, L_y - 1\}$ **hacer**
    para todo α **en** $\{0, \ldots, 8\}$ **hacer**
      $i_\alpha \leftarrow i + \boldsymbol{e}_{x\alpha}$
      $j_\alpha \leftarrow j + \boldsymbol{e}_{y\alpha}$
      si $0 \leq i_\alpha \leq L_x - 1$ **y** $0 \leq j_\alpha \leq L_y - 1$ **entonces**
        $g^{temp}_{i_\alpha j_\alpha \alpha} \leftarrow g_{ij\alpha} - (g_{ij\alpha} - g^{eq}_{ij\alpha})/\tau + S_{ij\alpha}$
      fin si
    fin para
  fin para
fin para
$g \leftarrow g^{temp}$

Luego de realizar las etapas de *streaming* y *colisión*, es necesario calcular las variables macroscópicas mediante la subrutina Solucion, cuyo funcionamiento se muestra en el Algoritmo 4. Aquí se calculan los valores de h, $\boldsymbol{u}_x$ y $\boldsymbol{u}_y$ de acuerdo a las ecuaciones (4.2) y (4.3).

Algoritmo 4 Subrutina Solucion.

Entrada: $L_x, L_y, \boldsymbol{e}_x, \boldsymbol{e}_y, g$
Salida: $h, \boldsymbol{u}_x, \boldsymbol{u}_y$

para todo i **en** $\{0, \ldots, L_x - 1\}$ **hacer**
  para todo j **en** $\{0, \ldots, L_y - 1\}$ **hacer**
    $h_{ij} \leftarrow 0$
    $\boldsymbol{u}_{xij} \leftarrow 0$
    $\boldsymbol{u}_{yij} \leftarrow 0$
    para todo α **en** $\{0, \ldots, 8\}$ **hacer**
      $h_{ij} \leftarrow h_{ij} + g_{ij\alpha}$
      $\boldsymbol{u}_{xij} \leftarrow \boldsymbol{u}_{xij} + \boldsymbol{e}_{x\alpha} g_{ij\alpha}$
      $\boldsymbol{u}_{yij} \leftarrow \boldsymbol{u}_{yij} + \boldsymbol{e}_{y\alpha} g_{ij\alpha}$
    fin para
    $\boldsymbol{u}_{xij} \leftarrow \boldsymbol{u}_{xij}/h_{ij}$
    $\boldsymbol{u}_{yij} \leftarrow \boldsymbol{u}_{yij}/h_{ij}$
  fin para
fin para

La siguiente función invocada corresponde a FEQ, la cual se encarga de calcular la función de distribución de equilibrio a partir de las variables macroscópicas computadas anteriormente. Como se explico en el Capítulo 5, se han utilizado dos funciones de distribución de equilibro distintas, estado la función presentada en (4.5), es decir, $f^{(eq)}$ implementada en la frontera del dominio en aquellos casos especiales en donde se desee aplicar condiciones de borde abiertas, por lo que en esta subrutina solo se visualiza el calculo de la función de distribución presente en (4.15), es decir, $g^{(eq)}$. La subrutina FEQ puede verse en el Algoritmo 5.

Algoritmo 5 Subrutina FEQ.

Entrada: $L_x, L_y, \boldsymbol{e}_x, \boldsymbol{e}_y, h, \boldsymbol{u}_x, \boldsymbol{u}_y$
Salida: g^{eq}

$k_1 \leftarrow 3{,}0$
$k_2 \leftarrow 4{,}5$
$k_3 \leftarrow 1{,}5$
para todo i **en** $\{0, \ldots, L_x - 1\}$ **hacer**
    para todo j **en** $\{0, \ldots, L_y - 1\}$ **hacer**
        $g^{eq}_{ij0} \leftarrow h_{ij} - 4k_3 h_{ij}(\boldsymbol{u}^2_{xij} + \boldsymbol{u}^2_{yij})/9$
        para todo α **en** $\{1, \ldots, 8\}$ **hacer**
            $u_1 \leftarrow \boldsymbol{e}_{x\alpha}\boldsymbol{u}_{xij} + \boldsymbol{e}_{y\alpha}\boldsymbol{u}_{yij}$
            $u_2 \leftarrow u_1^2$
            $u_3 \leftarrow \boldsymbol{u}^2_{xij} + \boldsymbol{u}^2_{yij}$
            $g^{eq}_{ij\alpha} \leftarrow h_{ij}(k_1 u_1 + k_2 u_2 - k_3 u_3)/9$
            si $\alpha \geq 5$ **entonces**
                $g^{eq}_{ij\alpha} \leftarrow g^{eq}_{ij\alpha}/4$
            fin si
        fin para
    fin para
fin para

Finalmente, la función TerminoFuente se encarga de calcular las componentes del término fuente S según lo especifica la ecuación (4.17). Nuevamente, debe recordarse que, cuando se tienen condiciones de borde abiertas, en la frontera se utiliza otra función de distribución de equilibrio, por lo que el término de fuerza cambia y como consecuencia, también lo hace el término fuente. De todas formas, de manera análoga a la subrutina anterior, en el Algoritmo 6 solo se muestra el caso en donde se utiliza la función $g^{(eq)}$.

Para la aproximación de las derivadas se utiliza diferencias finitas centradas:

$$\begin{aligned}\frac{\partial(b(i,j)+h(i,j))}{\partial x} &= \frac{(b_{i+1j}+h_{i+1j}) - (b_{i-1j}+h_{i-1j})}{2} \\ \frac{\partial(b(i,j)+h(i,j))}{\partial y} &= \frac{(b_{ij+1}+h_{ij+1}) - (b_{ij-1}+h_{ij-1})}{2}\end{aligned}$$

excepto en los bordes, debido a la falta de nodos adyacentes, en donde se utilizan diferencias progresivas o regresivas según sea el caso:

$$\begin{aligned}\frac{\partial(b(i,j)+h(i,j))}{\partial x} &= (b_{i+1j}+h_{i+1j}) - (b_{ij}+h_{ij}), && i = 0 \\ \frac{\partial(b(i,j)+h(i,j))}{\partial x} &= (b_{ij}+h_{ij}) - (b_{i-1j}+h_{i-1j}), && i = L_x - 1 \\ \frac{\partial(b(i,j)+h(i,j))}{\partial y} &= (b_{ij+1}+h_{ij+1}) - (b_{ij}+h_{ij}), && j = 0 \\ \frac{\partial(b(i,j)+h(i,j))}{\partial y} &= (b_{ij}+h_{ij}) - (b_{ij-1}+h_{ij-1}), && j = L_y - 1\end{aligned}$$

En el caso de existir condiciones de borde periódicas (ver Sección 4.3.1), sí se utilizan diferencias centradas en la frontera.

Algoritmo 6 Subrutina TerminoFuente.

Entrada: $L_x, L_y, \mathrm{g}, b, h, \boldsymbol{e}_x, \boldsymbol{e}_y, \hat{\boldsymbol{F}}_x, \hat{\boldsymbol{F}}_y$
Salida: S

para todo i **en** $\{0, \ldots, L_x - 1\}$ **hacer**
  para todo j **en** $\{0, \ldots, L_y - 1\}$ **hacer**
    si $i = 0$ **entonces**
      $w_x \leftarrow (h_{i+1j} + b_{i+1j}) - (h_{ij} + b_{ij})$
    si no si $i = L_x - 1$ **entonces**
      $w_x \leftarrow (h_{ij} + b_{ij}) - (h_{i-1j} + b_{i-1j})$
    si no
      $w_x \leftarrow ((h_{i+1j} + b_{i+1j}) - (h_{i-1j} + b_{i-1j}))/2$
    fin si
    si $j = 0$ **entonces**
      $w_y \leftarrow (h_{ij+1} + b_{ij+1}) - (h_{ij} + b_{ij})$
    si no si $j = L_y - 1$ **entonces**
      $w_y \leftarrow (h_{ij} + b_{ij}) - (h_{ij-1} + b_{ij-1})$
    si no
      $w_y \leftarrow ((h_{ij+1} + b_{ij+1}) - (h_{ij-1} + b_{ij-1}))/2$
    fin si
    $\boldsymbol{F}_x \leftarrow \hat{\boldsymbol{F}}_x - \mathrm{g} h_{ij} w_x$
    $\boldsymbol{F}_y \leftarrow \hat{\boldsymbol{F}}_y - \mathrm{g} h_{ij} w_y$
    $S_{ij0} \leftarrow 0$
    para todo α **en** $\{1, \ldots, 8\}$ **hacer**
      $S_{ij\alpha} \leftarrow 1{,}5(\boldsymbol{e}_{x\alpha} F_x + \boldsymbol{e}_{y\alpha} F_y)/6$
      si $\alpha \geq 5$ **entonces**
        $S_{ij\alpha} \leftarrow S_{ij\alpha}/2$
      fin si
    fin para
  fin para
fin para

6.3. Condiciones de Borde

A continuación, se describirán los tres tipos de condiciones de borde implementados en el algoritmo presentado.

6.3.1. Condiciones Periódicas

La inclusión de este tipo de condiciones de borde en el algoritmo se realiza mediante la modificación de las subrutinas Streaming-Colision y TerminoFuente. Un ejemplo de como implementar condiciones de borde periódicas horizontales es presentado en los Algoritmos 8 y7, en donde las líneas en rojo corresponden a las modificaciones hechas a las subrutinas originales. Para condiciones verticales, el ejemplo es análogo, pero trabajado con el índice j en vez de i.

Algoritmo 7 Subrutina TerminoFuente con condiciones periódicas.

Entrada: $L_x, L_y, \mathrm{g}, b, h, \boldsymbol{e}_x, \boldsymbol{e}_y, \hat{\boldsymbol{F}}_x, \hat{\boldsymbol{F}}_y$
Salida: S

para todo i **en** $\{0, \ldots, L_x - 1\}$ **hacer**
  para todo j **en** $\{0, \ldots, L_y - 1\}$ **hacer**
    si $i = 0$ **entonces**
      $w_x \leftarrow ((h_{i+1j} + b_{i+1j}) - (h_{L_x-1j} + b_{L_x-1j}))/2$
    si no si $i = L_x - 1$ **entonces**
      $w_x \leftarrow ((h_{0j} + b_{0j}) - (h_{i-1j} + b_{i-1j}))/2$
    si no
      $w_x \leftarrow ((h_{i+1j} + b_{i+1j}) - (h_{i-1j} + b_{i-1j}))/2$
    fin si
    si $j = 0$ **entonces**
      $w_y \leftarrow (h_{ij+1} + b_{ij+1}) - (h_{ij} + b_{ij})$
    si no si $j = L_y - 1$ **entonces**
      $w_y \leftarrow (h_{ij} + b_{ij}) - (h_{ij-1} + b_{ij-1})$
    si no
      $w_y \leftarrow ((h_{ij+1} + b_{ij+1}) - (h_{ij-1} + b_{ij-1}))/2$
    fin si
    $\boldsymbol{F}_x \leftarrow \hat{\boldsymbol{F}}_x - \mathrm{g} h_{ij} w_x$
    $\boldsymbol{F}_y \leftarrow \hat{\boldsymbol{F}}_y - \mathrm{g} h_{ij} w_y$
    $S_{ij0} \leftarrow 0$
    para todo α **en** $\{1, \ldots, 8\}$ **hacer**
      $S_{ij\alpha} \leftarrow 1{,}5(\boldsymbol{e}_{x\alpha} F_x + \boldsymbol{e}_{y\alpha} F_y)/6$
      si $\alpha \geq 5$ **entonces**
        $S_{ij\alpha} \leftarrow S_{ij\alpha}/2$
      fin si
    fin para
  fin para
fin para

Algoritmo 8 Subrutina Streaming-Colision con condiciones periódicas.

Entrada: $L_x, L_y, \tau, \boldsymbol{e}_x, \boldsymbol{e}_y, g, g^{eq}, S$
Salida: g

para todo i **en** $\{0, \ldots, L_x - 1\}$ **hacer**
  para todo j **en** $\{0, \ldots, L_y - 1\}$ **hacer**
    para todo α **en** $\{0, \ldots, 8\}$ **hacer**
      $i_\alpha \leftarrow i + \boldsymbol{e}_{x\alpha}$
      $j_\alpha \leftarrow j + \boldsymbol{e}_{y\alpha}$
      si $i_\alpha = -1$ **entonces**
        $i_\alpha \leftarrow Lx - 1$
      si no si $i_\alpha = L_x$ **entonces**
        $i_\alpha \leftarrow 0$
      fin si
      si $0 \leq j_\alpha \leq L_y - 1$ **entonces**
        $g^{temp}_{i_\alpha j_\alpha \alpha} \leftarrow g_{ij\alpha} - (g_{ij\alpha} - g^{eq}_{ij\alpha})/\tau + S_{ij\alpha}$
      fin si
      fin para
    fin para
  fin para
  $g \leftarrow g^{temp}$

6.3.2. Condiciones de Tipo *Bounce-Back*

La implementación de condiciones de borde de tipo *Bounce-Back* en el algoritmo requiere de la llamada a una nueva subrutina que debe ubicarse al final del ciclo, siendo la última función ejecutada antes de llamar a Streaming-Colision. Un ejemplo de esta nueva subrutina simulando un muro en el extremo izquierdo del dominio se encuentra presente en el Algoritmo 9. Por simetría, se hace evidente la implementación de esto en las demás fronteras del dominio.

Algoritmo 9 Subrutina BounceBack-Oeste.

Entrada: L_y, g
Salida: g

para todo j **en** $\{0, \ldots, L_y - 1\}$ **hacer**
  $g_{0j1}, g_{0j3} \leftarrow g_{0j3}, g_{0j1}$
  $g_{0j2}, g_{0j4} \leftarrow g_{0j4}, g_{0j2}$
  $g_{0j5}, g_{0j7} \leftarrow g_{0j7}, g_{0j5}$
  $g_{0j6}, g_{0j8} \leftarrow g_{0j8}, g_{0j6}$
fin para

6.3.3. Condiciones Abiertas

Implementar este tipo de condiciones de borde en el algoritmo es bastante más complicado en comparación a los otros casos analizados. Debido a la gran cantidad de cambios que sufre el método, su participación en el algoritmo no se reduce a un reemplazo de líneas o la creación de una subrutina que se encargue de la implementación. A continuación, aunque no se mostrará un algoritmo en pseudocódigo, se listarán los pasos necesarios para lograr que el método soporte condiciones de borde abiertas:

- Primero que todo, se debe agregar una subrutina a la función Setup con el objetivo de crear el dominio fantasma. Este procedimiento debe ampliar los arreglos de batimetría, altura de agua y

velocidades, aumentando su dimensión para extender el dominio real.

- Se deben modificar las componentes del término fuente S que van a incidir en los nodos de frontera vía *streaming*. Esto significa calcular solo la derivada de b y eliminar la de h.
- Del mismo modo, se deben modificar aquellas componentes de la función de distribución que van dirigidas a los nodos de frontera. Este cambio supone utilizar las componentes presentadas en (4.5).
- Para los dos pasos anteriores se aplican condiciones de borde periódicas y se especifican las componentes que van desde una frontera a la otra de manera que se simule la copia de componentes presentada en la Sección 4.3.3. Un ejemplo de esto es considerar condiciones de borde abiertas en la frontera oeste, es decir, $x = 0$. En este caso, la componente f_1 de un nodo en $x = L_x - 1$ (borde opuesto) se iguala a la componente f_1 del nodo que está a la misma altura en $x = 1$, simulando que luego del *streaming*, el nodo en $x = 0$ le copió dicha componente a aquel en $x = 1$.

Realizando estos pasos se logra implementar correctamente este tipo de condiciones de frontera, permitiendo una correcta combinación entre ambas funciones de distribución de equilibrio.

Capítulo 7

Experimentación Numérica

En la presente sección se analizarán cuatro casos de estudio que han sido simulados. Éstos son la simulación de flujo a través de un canal a modo de experimentación preliminar, el estado estacionario de lago en reposo con el fin de comprobar si el método es "*well-balanced*", un levantamiento central de agua con condiciones de tipo periódicas para verificar el cumplimiento de la conservación de masa, y propagación de ondas con condiciones de borde abiertas para comprobar el comportamiento de estas condiciones de frontera de acuerdo a la solución implementada (ver Capítulo 5).

7.1. Experimentación Preliminar: Flujo en un Canal

En esta sección se simula y analiza un importante aunque simple tipo de flujo, el cual corresponde al que ocurre dentro de un cilindro o entre dos superficies paralelas. La Ley de Poiseuille permite determinar el flujo laminar estacionario del fluido que pasa a través de este canal, encontrando que en las paredes o bordes la velocidad es 0 y que el máximo valor de ésta se alcanza en el centro del dominio.

Si bien este experimento consiste en un problema totalmente distinto al que se intenta resolver en este trabajo, se considera importante de presentar a modo de experimentación preliminar, cuyo objetivo es el entendimiento del método de *Lattice Boltzmann* en un escenario de menos complejidad. Esto debido a que, como se mencionó en la Sección 4.2, la implementación del método es la misma, cambiando únicamente su función de distribución de equilibrio, pero manteniendo su estructura algorítmica y el uso del modelo D2Q9. De este modo, la principal diferencia es que al resolver este problema se está buscando una solución a las ecuaciones de Navier-Stokes, presentadas en la Sección 2.2, obteniendo la siguiente función de distribución de equilibrio:

$$f_\alpha^{(eq)}(x) = \omega_\alpha \rho(x) \left[1 + 3\boldsymbol{e}_\alpha \cdot \boldsymbol{u} + \frac{9}{2}(\boldsymbol{e}_\alpha \cdot \boldsymbol{u})^2 - \frac{3}{2}\boldsymbol{u}^2 \right]$$

donde los pesos ω_α corresponden a 4/9 para $\alpha = 0$, 1/9 para $\alpha = 1, 2, 3, 4$ y 1/36 para $\alpha = 5, 6, 7, 8$.

En un canal de ancho $2a$ como el de la Figura 7.1, la velocidad se calcula como:

$$u(x) = \frac{G}{2\mu}(a^2 - x^2)$$

donde μ es la viscosidad y G es el gradiente de presión, el cual en este caso se considerará gravitacional en una tubería vertical, es decir, $G = \rho g$.

A continuación se analiza el experimento realizado en [38], en donde se simula un fluido con número de Reynolds Re = 4,4 y una velocidad máxima de 0,1 $lu\ ts^{-1}$. Para lograr obtener estos valores, y

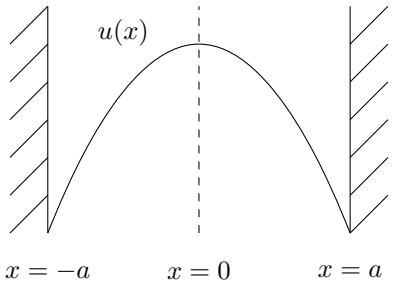


Figura 7.1: Perfil de velocidad para el flujo en un canal.

considerando que se utiliza $\tau = 1$, lo que produce una viscosidad de $1/6\ lu^2\ ts^{-1}$, se necesita un ancho del canal de 11 lu y una gravedad g igual a $1{,}102 \times 10^{-3}\ lu\ ts^{-2}$. En las paredes del canal se utilizó condiciones de borde de tipo *Bounce-Back* y en los extremos de éste (entrada y salida de fluido) fueron implementadas condiciones periódicas. El resultado corresponde al alcance de un estado estacionario luego de aproximadamente 720 *time steps* con una velocidad máxima de 0,1 $lu\ ts^{-1}$, como se aprecia en la Figura 7.2.

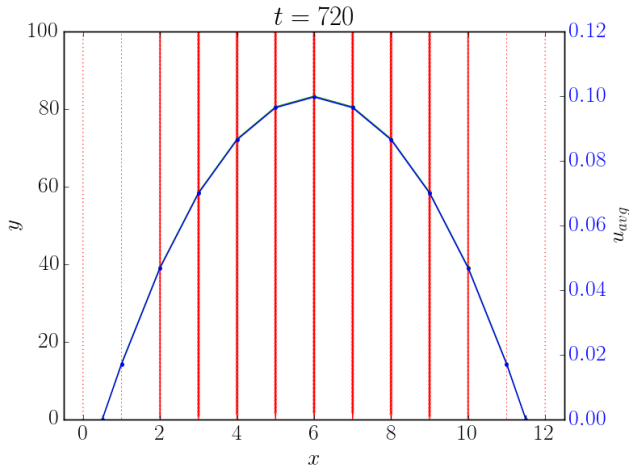


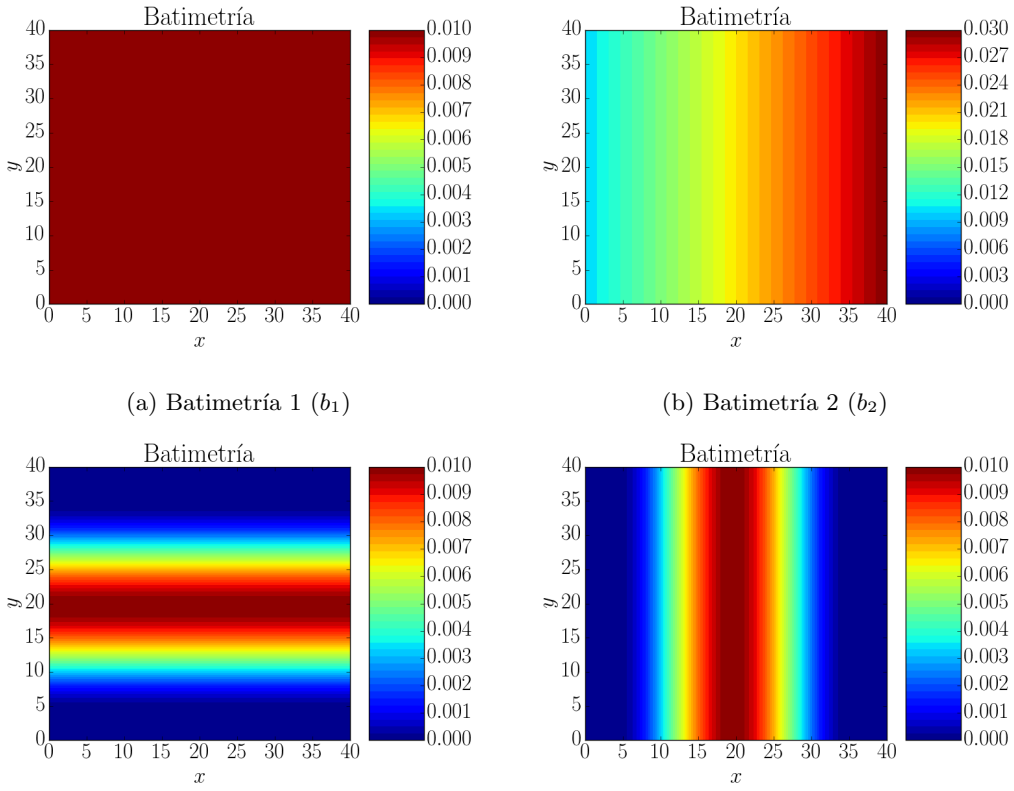
Figura 7.2: Estado estacionario del flujo en un canal.

7.2. Lago en Reposo

Este conjunto de simulaciones permiten verificar si el método cumple con la propiedad de ser "*well-balanced*". Las pruebas consisten en especificar una altura de agua $w_0 = 0{,}05$ constante en todo el dominio y luego de algunos *time steps* se verifica si en nivel de agua se mantuvo en su nivel inicial. Es importante aclarar que, debido a que el método realiza exactamente los mismos pasos en cada *time step*, basta con que no cambie en el primero de ellos para asegurar que no lo hará en el futuro. Aun así, se eligió de manera aleatoria el tiempo $t = 16$ para la comprobación.

Esta simulación se ejecutó para distintas configuraciones de batimetría y condiciones de borde, las cuales son listadas a continuación.

Batimetrías:



(a) Batimetría 1 (b_1)

(b) Batimetría 2 (b_2)

(c) Batimetría 3 (b_3)

(d) Batimetría 4 (b_4)

Figura 7.3: Distintas configuraciones de batimetría para simular el lago en reposo.

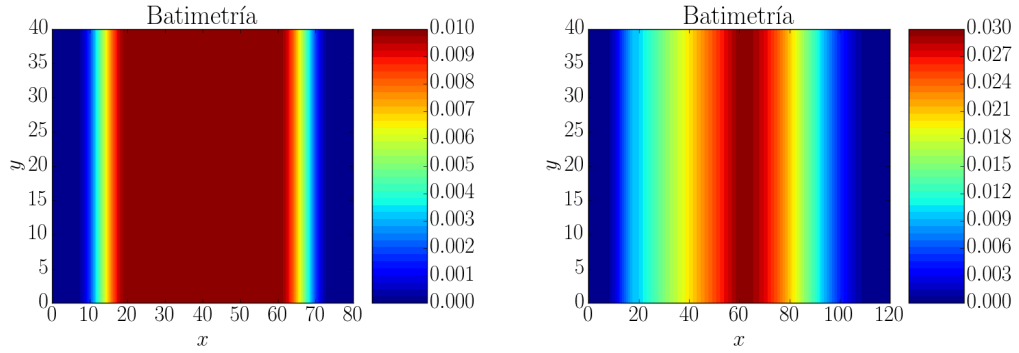
Las distintas batimetrías utilizadas son presentadas en la Figura 7.3. A continuación son listadas y descritas:

- b_1: corresponde a una superficie totalmente plana de altura 0,01.
- b_2: es una rampa a lo largo del eje horizontal cuya altura mínima es de 0,01 y la máxima tiene un valor de 0,03.
- b_3: consiste en una función sinusoidal al cuadrado en el eje vertical que se mantiene constante a lo largo del eje horizontal. Su altura máxima es de 0,01.
- b_4: idéntica a b_3 pero rotada en 90°.

Condiciones de Borde:

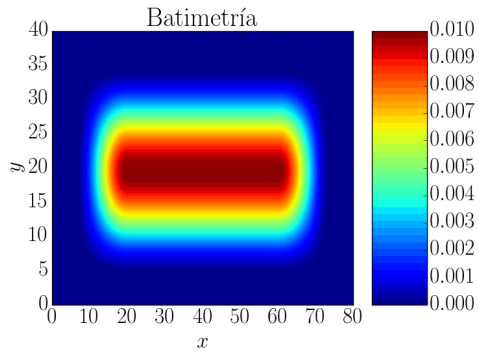
Se utilizaron los tres tipos de condiciones de borde analizados en la Sección 6.3. Éstas fueron implementadas en las fronteras este y oeste, mientras que las fronteras norte y sur siempre mantuvieron condiciones de borde periódicas con el fin de representar un dominio infinito a lo largo del eje vertical.

En la Figura 7.4 es posible apreciar los dominios fantasma añadidos a las batimetrías b_1, b_2, b_3 y b_4. Es posible apreciar que, debido a la forma de b_4 no se hace necesario el uso de dominio fantasma aun con condiciones de borde abiertas.

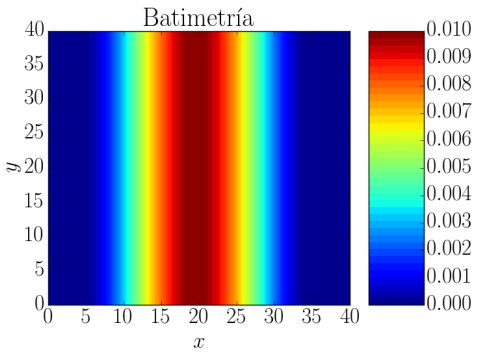


(a) Batimetría 1 (b_1)

(b) Batimetría 2 (b_2)



(c) Batimetría 3 (b_3)



(d) Batimetría 4 (b_4)

Figura 7.4: Dominios fantasma para cada configuración de batimetría en simulación de lago en reposo.

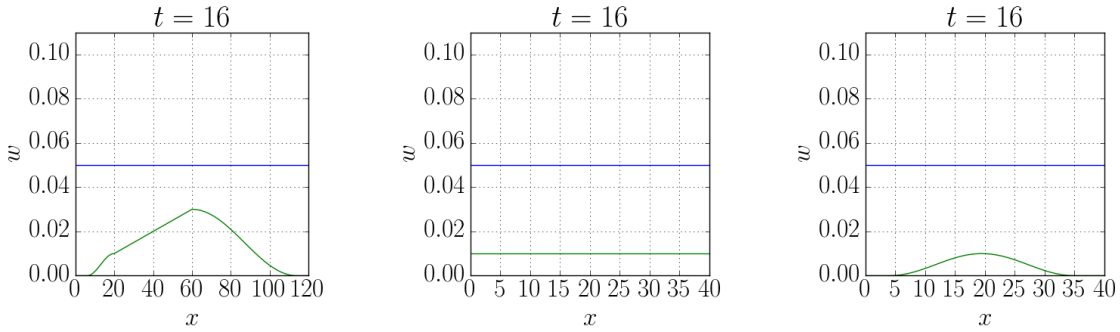
Resultados

Como se puede apreciar en la Tabla 7.1, efectivamente se cumple la propiedad de "*well-balanced*", pues todos los valores de w se mantuvieron constantes a después de 16 *time steps*.

Tabla 7.1: Resultados de simulación de lago en reposo en $t = 16$.
P: condiciones periódicas. BB: condiciones *Bounce-Back*. A: condiciones abiertas.

	b_1			b_2		
	P	BB	A	P	BB	A
máx(w)	0.05	0.05	0.05	0.05	0.05	0.05
mín(w)	0.05	0.05	0.05	0.05	0.05	0.05
	b_3			b_4		
	P	BB	A	P	BB	A
máx(w)	0.05	0.05	0.05	0.05	0.05	0.05
mín(w)	0.05	0.05	0.05	0.05	0.05	0.05

Algunas imágenes para visualizar los resultados pueden ser vistas en la Figura 7.5.



(a) b_2 con condiciones de borde abiertas

(b) b_3 con condiciones de tipo *Bounce-Back*

(c) b_4 con condiciones de frontera periódicas

Figura 7.5: Algunos resultados de la simulación de lago en reposo.

7.3. Conservación de Masa

Este experimento corresponde a realizar un levantamiento de agua en el centro del eje x con condiciones de borde de tipo periódicas implementadas en las fronteras norte y sur y condiciones de tipo *Bounce-Back* en los bordes este y oeste. El objetivo es verificar que el volumen de agua se mantiene constante a lo largo del tiempo, esto debido a que las condiciones de borde utilizadas no permiten que el agua escape, entrando al dominio por una frontera a medida que sale por la opuesta, en el caso de las coniciones periódicas, o rebotando hacia el interior del dominio, en el caso de las de tipo *Bounce Back*.

Para esta simulación, se utilizará una batimetría plana de altura $b = 0{,}01$ y una altura inicial de agua con valor $w = 0{,}05$ en un dominio de dimensión 40×40 *lattice units*. El levantamiento de agua se modelará de acuerdo a la siguiente función exponencial:

$$w(x, y) = 0{,}05 + 0{,}05e^{-\frac{(x-20)^2}{20}} \tag{7.1}$$

alcanzando una altura máxima de 0,1 y siendo invariable a lo largo del eje y.

La mejor manera para comprobar el cumplimiento de la conservación de masa es sumar todas las alturas de agua h a lo largo del tiempo y verificar que este valor se vuelva constante. En otras palabras, lo que se busca es:

$$h_{\text{total}} = \sum_{i=0}^{40} \sum_{j=0}^{40} h(i, j) = \text{constante}$$

lo que efectivamente se cumple, como es posible apreciar en la Figura 7.6 y la tabla 7.2.

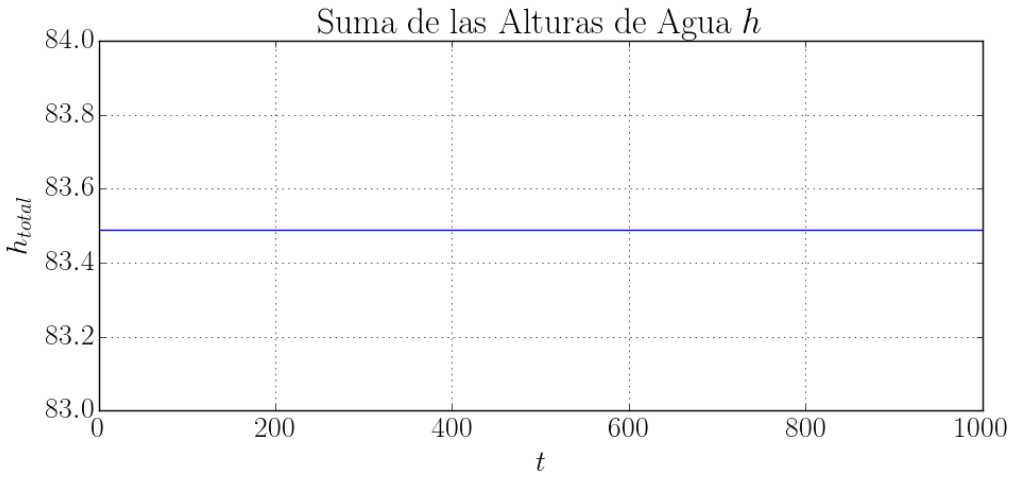


Figura 7.6: Suma de las alturas de agua h a lo largo de 1000 *time steps*.

Tabla 7.2: Resultados de simulación de conservación de masa.

t	h_{total}
0	83.489641919
150	83.489641919
300	83.489641919
450	83.489641919
600	83.489641919
750	83.489641919
900	83.489641919

Si bien se ha comprobado que la suma de las alturas de agua no se ve modificada con el paso del tiempo, la aproximación del volumen de agua es un escenario muy distinto. Con las especificaciones mencionadas anteriormente, el volumen del agua puede ser calculado como:

$$\begin{aligned} V &= \int_0^{40}\int_0^{40} w(x,y) - b(x,y)dxdy \\ &= \int_0^{40}\int_0^{40} 0{,}05 + 0{,}05e^{-\frac{(x-20)^2}{20}} - 0{,}01dxdy \\ &= 40\int_0^{40} 0{,}04 + 0{,}05e^{-\frac{(x-20)^2}{20}}dx \\ &= 40\left(1{,}6 + 0{,}05\int_0^{40} e^{-\frac{(x-20)^2}{20}}dx\right) \\ &\approx 79{,}8533 \end{aligned}$$

Para aproximar el valor de la integral en el dominio discretizado se utilizará la regla de Simpson, definida como:

$$\int_a^b f(x)dx = \frac{b-a}{6}\left[f(a) + 4f\left(\frac{b+a}{2}\right) + f(b)\right]$$

De esta manera, beneficiándose de la invariabilidad a lo largo del eje y, se fijará un valor al centro de dicho eje, se aproximará la integral a lo largo del eje x y se escalará por 40 de la forma:

$$\begin{aligned} \hat{V} &= 40\sum_{i=0}^{14}\frac{2}{6}\left[h(2i,20) + 4h(2i+1,20) + h(2i+2,20)\right] \\ &= \frac{40}{3}\sum_{i=0}^{14}\left[h(2i,20) + 4h(2i+1,20) + h(2i+2,20)\right] \end{aligned}$$

donde $h(i,j) = h_{ij} = w_{ij} - b_{ij}$.

Como es posible apreciar en la Figura 7.7, el volumen de agua aproximado tiene un valor inicial muy cercano al volumen calculado de forma analítica, pero comienza a oscilar en torno a su estado estacionario ($\hat{V} \approx 79{,}4664236656$), el cual difiere del valor mencionado. Esto se debe al efecto de la discretización del dominio.

El mismo efecto es presentado en la Figura 7.8, en donde se grafica el error de la aproximación respecto al resultado analítico:

$$E = |\hat{V} - V|$$

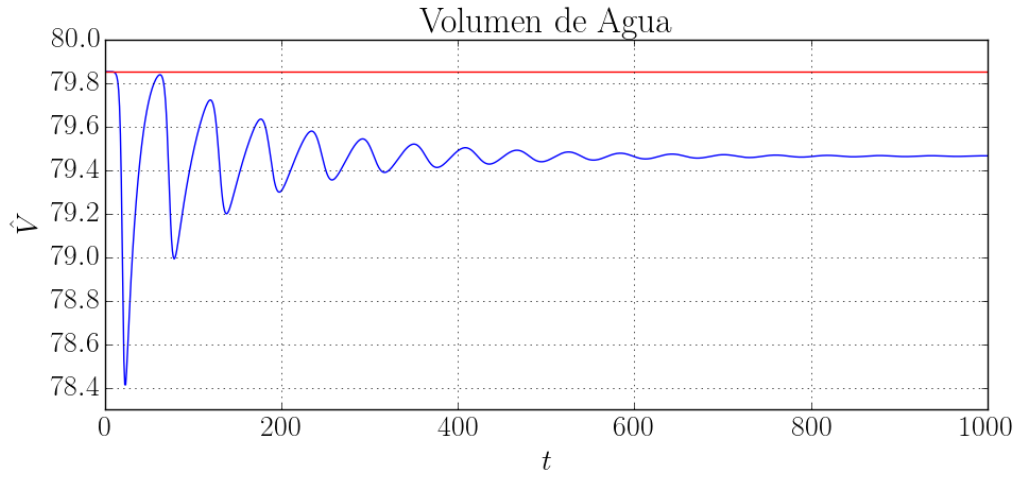


Figura 7.7: Aproximación del volumen de agua $\hat{V}$ a lo largo de 1000 *time steps*.

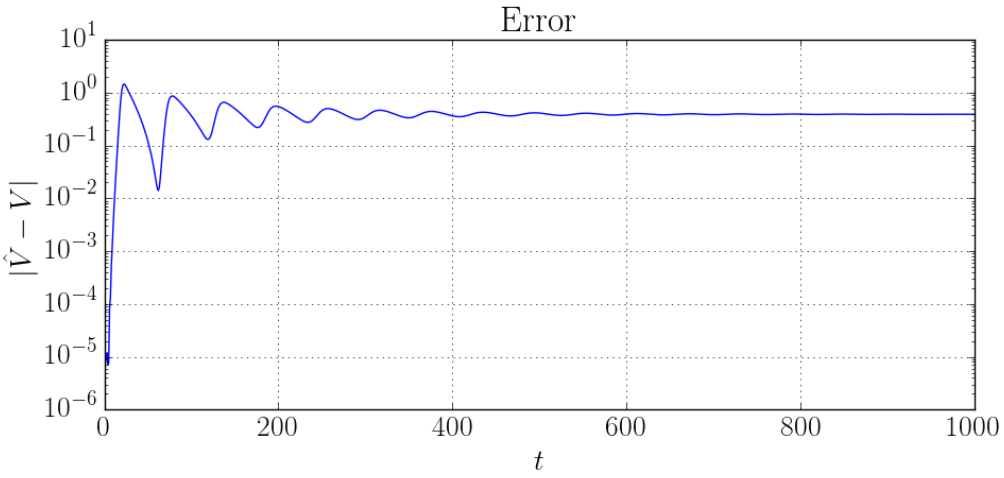
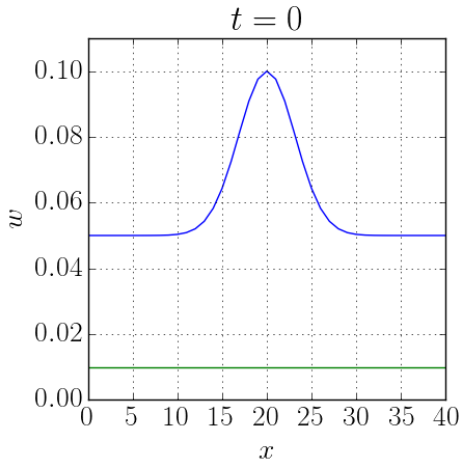
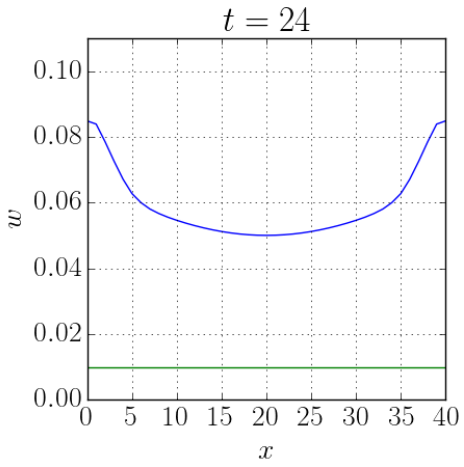


Figura 7.8: Error de aproximación del volumen de agua a lo largo de 1000 *time steps*.

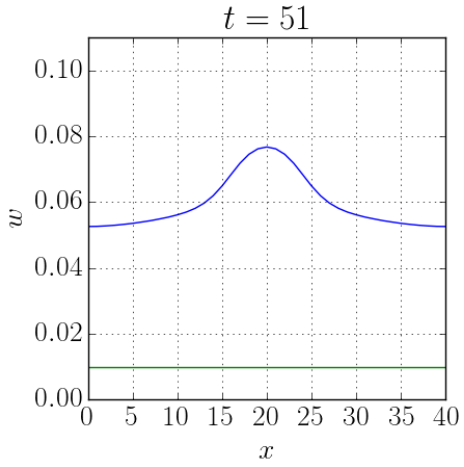
Algunas imágenes explicativas de la simulación son presentadas en la Figura 7.9. En ellas es posible ver la configuración inicial, el efecto de las condiciones de borde, y el cómo la onda pierde altura con el paso del tiempo hasta llegar a un estado estacionario.



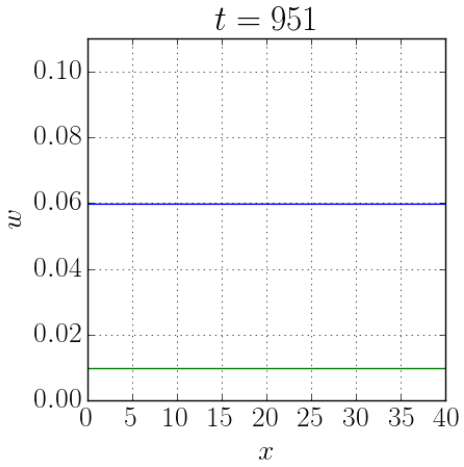
(a) Altura inicial de agua según la ecuación (7.1) en el tiempo $t = 0$



(b) Al llegar al borde, las ondas rebotan en dirección opuesta



(c) Las ondas vuelven al centro, pero con una menor altura debido al efecto de la gravedad



(d) Al finalizar la simulación, las ondas se han disipado totalmente, alcanzando un estado estacionario

Figura 7.9: Simulación para verificar la conservación de masa.

7.4. Condiciones de Borde Abiertas

El presente experimento tiene como único objetivo comprobar el correcto funcionamiento del método ante condiciones de borde abiertas. Con ello, se pondrá a prueba la solución propuesta en este trabajo, contemplando el uso de dos funciones de distribución de equilibrio distintas y la implementación del dominio fantasma. Para lograr esto, se simula la propagación de ondas esperando que desaparezcan a través de las fronteras del dominio.

La simulación se realizó utilizando una batimetría que consiste en una función sinusoidal al cuadrado en el eje vertical que se mantiene constante a lo largo del eje horizontal. Su altura máxima es de 0,01, siendo idéntica a la batimetría b_3 del experimento de lago en reposo en la Sección 7.2. Por otro lado, al igual que en el experimento de conservación de masa en la Sección 7.3, se utilizó una altura inicial de agua con valor $w = 0{,}05$ en un dominio de dimensión 40×40 *lu*, con un levantamiento modelado de acuerdo a la ecuación (7.1).

Nuevamente se procede a ilustrar en la Figura 7.10 la suma de las alturas de agua h, debido a que permite analizar los distintos periodos de tiempo involucrados en la simulación. De esta forma, es posible apreciar que:

- En los primeros 40 *time steps* la suma de las alturas de agua se mantiene constante debido a que las ondas aun no llegan a las fronteras del dominio.
- Entre los tiempos 40 y 80, la cantidad de agua baja de forma muy acelerada debido a que las ondas atraviesan los bordes.
- A partir de los 80 *time steps* y hasta los 300, se generan oscilaciones debido a la anterior propagación de las ondas.
- Finalmente, en el resto de la simulación se alcanza un estado estacionario.

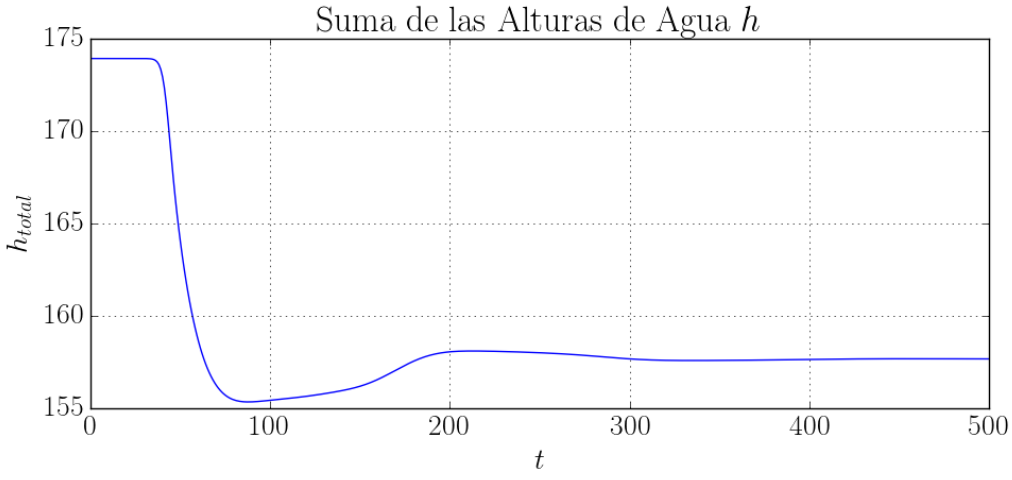
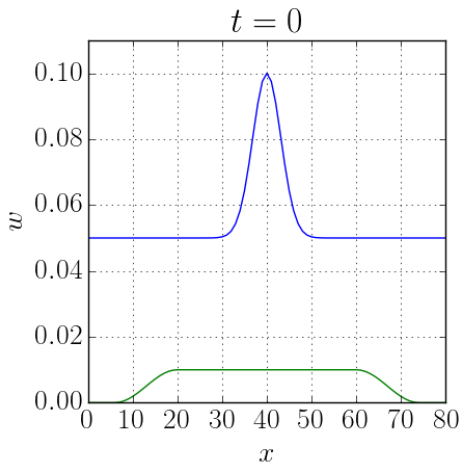
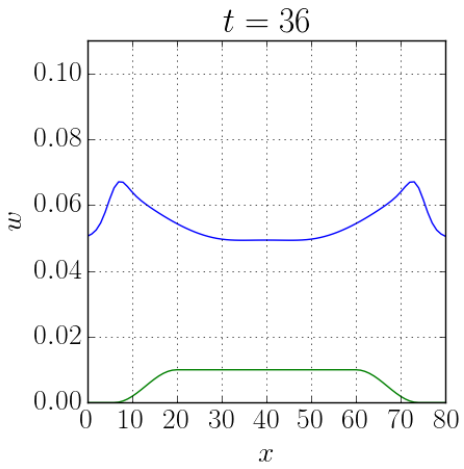


Figura 7.10: Suma de las alturas de agua h a lo largo de 500 *time steps*.

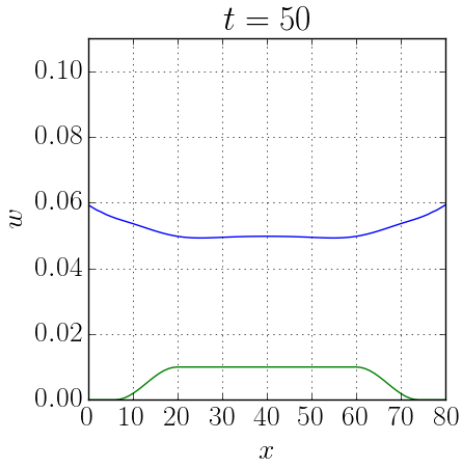
Como se puede ver en la Figura 7.11, en donde se ilustran tiempos importantes de la simulación, el resultado de ésta fue exitoso. El modelo soporta totalmente condiciones de frontera abiertas, permitiendo que las ondas atraviesen el borde del dominio sin generar perturbaciones indeseables. Al desaparecer, el nivel de agua presenta oscilaciones que disminuyen en amplitud hasta alcanzar el estado estacionario en donde la altura de agua se iguala a la inicial ($w = 0{,}05$), siendo este un comportamiento normal y esperado y no debiendo confundirse con perturbaciones producidas por la inestabilidad del método.



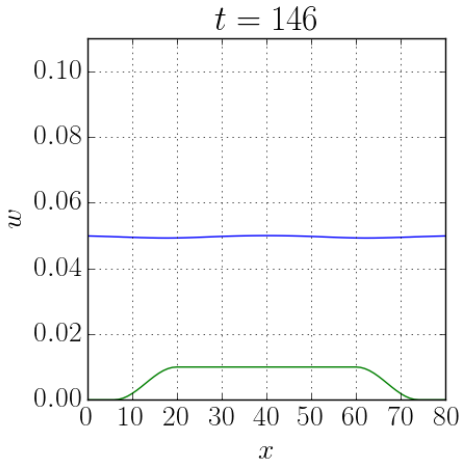
(a) Altura inicial de agua según la ecuación (7.1) en el tiempo $t = 0$



(b) Las ondas se acercan a las fronteras del dominio



(c) Las ondas atraviesan los bordes, desapareciendo permanentemente del dominio



(d) Las ondas han desaparecido completamente dejando una pequeña oscilación

Figura 7.11: Simulación con condiciones de borde abiertas.

Capítulo 8

GPU

8.1. Introducción a CUDA

CUDA [53] es una arquitectura de cálculo paralelo de NVIDIA que aprovecha la potencia de las unidades de procesamiento gráfico (GPU, por su sigla en inglés) para obtener un mayor rendimiento computacional. Debido a la necesidad de renderizar gráficos tridimensionales de alta definición en tiempo real, las GPU se han convertido en dispositivos multinúcleo de multiprocesamiento altamente paralelos, presentando un poder de cómputo mucho mayor a las unidades de procesamiento central o CPU (ver Figura 8.1).

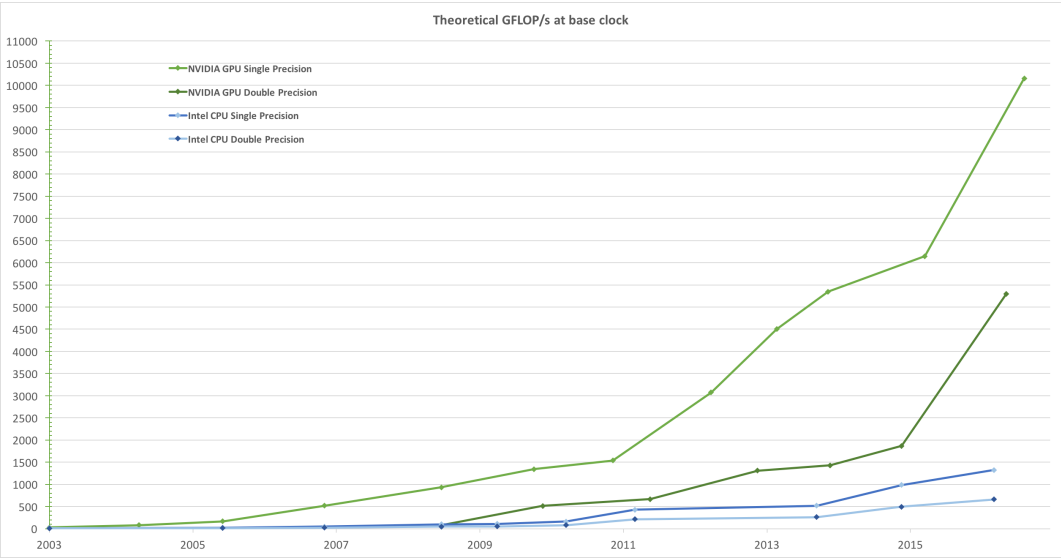


Figura 8.1: Operaciones de punto flotante por segundo (FLOPS) en CPU y GPU.
Fuente: CUDA C Programming Guide v8.0

Al contrario de una CPU, la cual corresponde a un gran procesador con un alto número de instrucciones optimizadas y una amplia memoria caché, una GPU contiene un gran número de pequeños procesadores con pocas instrucciones trabajando en paralelo, lo que permite procesar una gran cantidad de datos al mismo tiempo, pero ejecutando la misma instrucción para cada uno de ellos. En la Figura 8.2 es posible apreciar gráficamente la diferencia anteriormente descrita entre las mencionadas unidades de procesamiento.

Figura 8.2: Comparación gráfica entre las arquitecturas de CPU y GPU.
Fuente: CUDA C Programming Guide v8.0.

En este contexto, para obtener la capacidad computacional de las GPU en un determinado *software*, NVIDIA provee una extensión de los lenguajes C/C++ gracias a la cual es posible definir funciones especiales que serán ejecutadas en estos dispositivos. Dichas funciones se denominan "*kernels*" y su objetivo es paralelizar las instrucciones que contienen, en un número determinado de hebras o "*threads*". Estas últimas se agrupan en bloques que son procesados de forma independiente, lo que hace surgir la necesidad de sincronizar los cálculos cuando una determinada fórmula requiere del resultado de otra anterior, debido a que una hebra de algún bloque puede intentar utilizar un valor que aun no ha sido calculado por otra hebra de un bloque distinto. Afortunadamente, antes de comenzar la ejecución de un "*kernel*" existe una sincronización implícita, pues se requiere que hayan terminado de procesarse todas las hebras del "*kernel*" anterior, permitiendo afrontar la problemática previamente descrita mediante la separación de las fórmulas dependientes en distintos "*kernels*".

8.2. Implementación y Resultados

Se implementó una adaptación del código secuencial en Python al lenguaje C++, manteniendo la misma estructura algorítmica presentada en 6.2. De este modo, para obtener una alta eficiencia computacional en la versión final del código, fueron probadas distintas configuraciones para cada subrutina o "*kernel*", eligiendo en cada caso la que presentó un menor tiempo de ejecución. Estas diferencias corresponden a la compensación entre una alta paralelización y el tipo de sentencias que se deben utilizar para manejar los datos. El ejemplo que ilustra de mejor manera este escenario es el "*kernel*" encargado de calcular la función de distribución de equilibrio. Debido a la definición de esta función presente en (4.15), el cálculo de sus componentes difiere de acuerdo al valor de α, requeriendo tres instrucciones distintas (una para $\alpha = 0$, otra para $\alpha = 1, 2, 3, 4$ y una última para $\alpha = 5, 6, 7, 8$), las cuales pueden ser manejadas mediante una sentencia condicional de tipo "*if-else if-else*". El problema radica en que, si bien esto permite paralelilzar el código de manera que cada hebra se encargue de calcular, para un único nodo, una componente en particular de la función de equilibrio ($L_x \times L_y \times 9$ hebras), este tipo de instrucciones son computacionalmente costosas, por lo que debe decidirse si adoptar esta opción o buscar una variante que implique un menor tiempo de cómputo. Así, otra forma de implementar lo anteriormente descrito corresponde a utilizar una paralelización de menor nivel, en la cual cada hebra tenga que calcular las nueve componentes de la función de equilibrio en un nodo del dominio ($L_x \times L_y$ hebras), lo que significa un mayor número de instrucciones, pero limitándose a realizar solo operaciones aritméticas. Para este caso en específico, la segunda opción resultó ser más eficiente, presentando el "*kernel*" un tiempo promedio de ejecución de 77,359 microsegundos, mientras que la implementación con sentencias condicionales demoró 105,53 microsegundos, habiendo sido ambos tiempos medidos en una misma simulación con igual tamaño de dominio. Otro factor que afecta en el rendimiento de la primera opción es que, al presentar un mayor nivel de paralelización y,

por ende, requerir un mayor número de hebras, para dominios con gran cantidad de nodos, el número de núcleos de la GPU no son suficientes para procesar todas las hebras al mismo tiempo, generándose una secuencialidad al tener que esperar ciertas hebras que otras finalicen su procesamiento.

Antes de exponer los resultados de esta nueva implementación, es importante mencionar que todos los experimentos fueron ejecutados utilizando una tarjeta gráfica NVIDIA Tesla K20M con 5GB de memoria RAM y sistema de comunicación PCI Express 2.0 x16. Por otra parte, la versión 7.5 de CUDA fue utilizada en la compilación.

Se realizaron dos distintas comparaciones entre la implementación realizada en CUDA respecto al código secuencial en lenguaje Python. La primera de ellas corresponde a la validación del nuevo código, buscando probar que los resultados obtenidos entre ambas implementaciones son equivalentes. Para esto, se ejecutó una simulación simple con un levantamiento central de agua definido por una función exponencial sobre batimetría plana, en un dominio de 40×40 *lattice units*. Para determinar la equivalencia entre los resultados se utilizó la norma infinita del error, es decir, la mayor diferencia entre los valores de altura de agua w a lo largo de todos los nodos que componen la malla de la simulación. En la Figura 8.3 se presenta esta comparación a lo largo de 100 *time steps*.

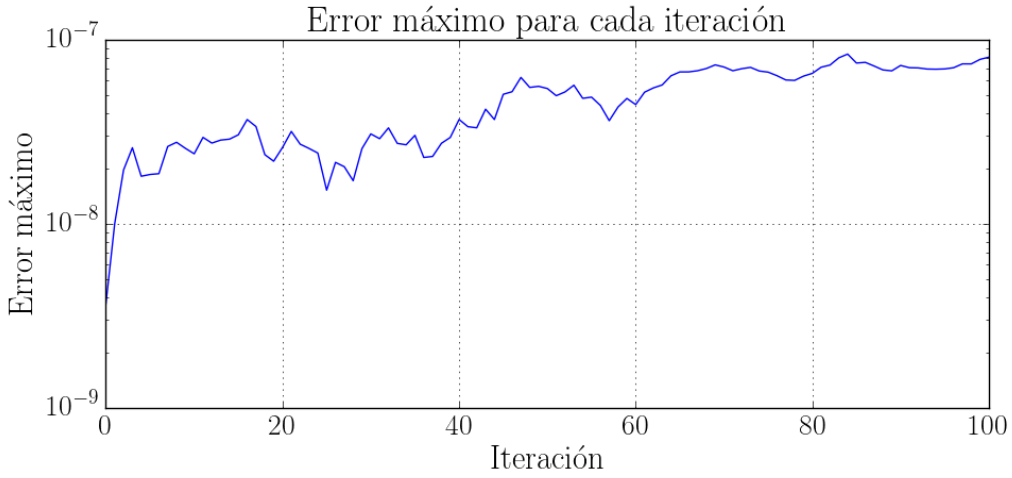


Figura 8.3: Comparación de resultados obtenidos en CUDA y Python.

Es posible verificar que el error no excede el orden de 10^{-7} para las primeras 100 iteraciones, lo que corresponde a resultados muy similares, con pequeñas diferencias numéricas atribuidas a la precisión utilizada, pues en Python se usó precisión doble, mientras que en CUDA fue usada precisión simple. De esta forma, se puede afirmar que ambas implementaciones realizan un procesamiento equivalente y aplican el método de *Lattice Boltzmann* en la resolución de las ecuaciones de *shallow water*.

La siguiente comparación realizada es respecto a los tiempos de ejecución de las implementaciones. Si bien el objetivo de codificar el algoritmo en Python era entender el funcionamiento del método mediante el uso de un lenguaje de programación simple y de alto nivel, se consideró injusto comparar de forma directa ambas implementaciones, debido a que el uso de las estructuras e instrucciones básicas, como listas y ciclos "*for*", no constituye una buena referencia de lo que un código secuencial es capaz de realizar. De este modo, también se realizó una mejora al código secuencial de Python mediante la utilización del módulo NumPy [54], el cual provee estructuras y funciones optimizadas para computación matemática y vectorial.

Los tiempos por iteración fueron calculados como el tiempo promedio de 100 iteraciones del método para una misma simulación con distintos tamaños de dominio, teniendo éstos dimensiones de 40×40

lu (1681 nodos), 100×100 *lu* (10201 nodos), 200×200 *lu* (40401 nodos), 500×500 *lu* (251001 nodos) y 1000×1000 *lu* (1002001 nodos). Como es posible apreciar en la Figura 8.4, la optimización realizada al código secuencial logró reducir el tiempo de ejecución en casi dos órdenes de magnitud, correspondiendo a una gran mejora en términos de eficiencia computacional. Aun así, la implementación paralela en CUDA es muy superior a ambas versiones secuenciales, disminuyendo el tiempo de cómputo en más de dos órdenes de magnitud respecto al código secuencial optimizado, confirmando de esta manera que el método de *Lattice Boltzmann* posee una gran capacidad de paralelización debido a que los cálculos requeridos se realizan de forma local, sin requerir datos provenientes de todo el dominio.

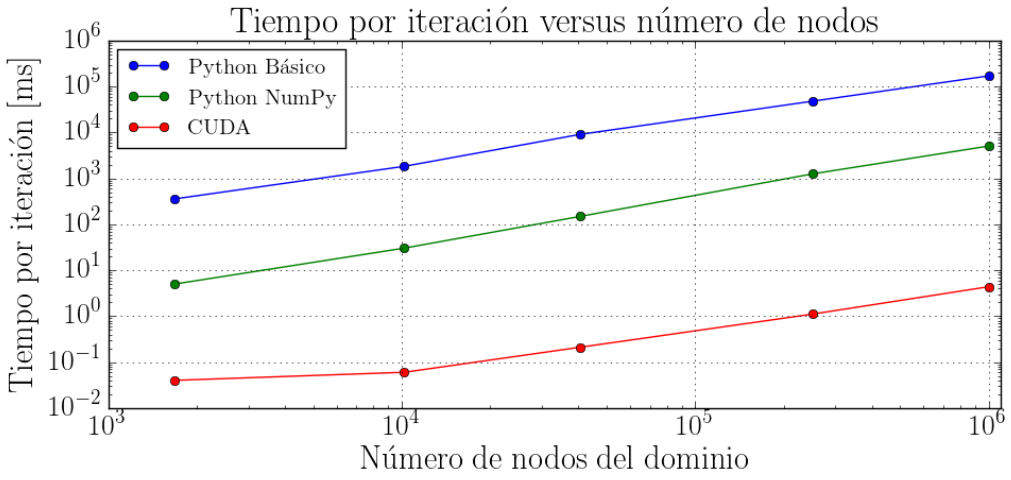


Figura 8.4: Comparación de tiempos de ejecución en CUDA y Python.

También es posible notar una relación lineal entre el tiempo de ejecución y el número de nodos que conforman el dominio. Resultados numéricos de la medición del tiempo son presentados en la Tabla 8.1.

Tabla 8.1: Comparación de tiempos de ejecución en CUDA y Python.
Tiempo por iteración en milisegundos.

	Número de nodos del dominio				
	1681	10201	40401	251001	1002001
Python Básico	358,3499	1838,5399	9089,2400	48152,7700	171548,1200
Python NumPy	5,0099	30,3900	148,5299	1257,2999	5089,5800
CUDA	0,040	0,060	0,210	1,110	4,390

En último lugar, se presenta un análisis más profundo de la implementación en CUDA, comparando los tiempos de ejecución de cada "*kernel*". Éstos corresponden al tiempo promedio en microsegundos que demora una llamada a cada "*kernel*" incluido en la comparación, la cual es exhibida en la Tabla 8.2. La simulación a partir de la cual se tomaron las mediciones presentadas corresponde a la misma del estudio anterior en un dominio de 40×40 *lattice units*, manteniéndose la proporción de los tiempos para los demás tamaños de éste.

Tabla 8.2: Comparación de tiempos de ejecución de cada "*kernel*".

"*kernel*"	Tiempo por llamada [μs]
Streaming-Colision	5.3780
Solucion	3.8230
FEQ	13.851
TerminoFuente	5.1940
Total	**28.2460**

Como es posible apreciar, el "*kernel*" encargado de calcular la función de distribución de equilibrio es el que demora más tiempo en su ejecución, debiéndose esto a la complejidad de su cómputo y al gran número de cálculos requeridos como se explicó al comienzo de esta sección. Los siguientes "*kernels*" que más tardan por cada llamada son los de Streaming-Colision y TerminoFuente, siendo sus tiempos de ejecución muy similares. Finalmente, debido a su simplicidad de cálculo, el "*kernel*" en el cual se calculan las variables macroscópicas del problema es el de más rápida ejecución. Cabe destacar que la diferencia entre la suma de los tiempos de los "*kernels*" presentes en la Tabla 8.2 y el tiempo por iteración de 40 μs presentado en la Tabla 8.1 se debe a pequeñas instrucciones adicionales que corresponden a copia de variables y también a procesos internos implícitos que realiza CUDA respecto al manejo de memoria.

Conclusiones

En líneas generales, es posible afirmar que se ha cumplido a cabalidad con el objetivo de este trabajo, logrando aplicar el método de *Lattice Boltzmann* a la resolución de las ecuaciones de *shallow water*. En paralelo al proceso de estudio del método, se desarrolló una implementación en lenguaje Python, permitiendo encontrar el principal problema enfrentado, el cual consistió en que una de las funciones de distribución de equilibrio propuestas no permitía obtener un modelo "*well-balanced*" ante la presencia de batimetría no plana, mientras que la segunda función propuesta no soportaba condiciones de borde abiertas. Debido a que cada una de dichas funciones presentaba un buen comportamiento en el escenario en donde la otra fallaba, se decidió solucionar el problema mediante la combinación de ambas en distintas zonas del dominio.

Gracias al planteamiento mencionado, se logró obtener un metodo "*well-balanced*" que soporta condiciones de borde abiertas, siendo ambas propiedades muy importantes en el problema que se intenta resolver. De forma adicional al resultado teórico, el cumplimiento de estas porpiedades fue validado mediante diversos experimentos numéricos. En este contexto, se logró verificar que el método fuera "*well-balanced*" mediante la simulación de un lago en reposo, en donde los resultados fueron satisfactorios al mantenerse el estado estacionario requerido. Por otro lado, se experimentó con condiciones de borde abiertas, comprobando que efectivmente el comportamiento fue el deseado, no generando inestabilidades y perturbaciones que estropeasen la simulación. Finalmente, también se confirmó que se trata de un método conservativo al evidenciar que la masa total de agua se mantuvo constante durante la disipación de una onda propagada a través de un dominio cerrado implementado con condiciones de borde de tipo *Bounce-Back*.

Finalmente, se implementó una versión paralelizada del algoritmo mediante el uso de GPU con el objetivo de obtener una mayor eficiencia computacional. Se validó que los resultados de esta nueva implementación fueran equivalentes a los presentados por el código secuencial mediante la ejecución de una misma simulación en ambas versiones. Adicionalmente, se optimizó la implementación secuencial en Python utilizando las herramientas optimizadas del módulo NumPy, con el fin de realizar una comparación más justa de los tiempos de procesamiento. Los resultados fueron bastante satisfactorios, debido a que la versión secuencial disminuyó considerablemente su tiempo de ejecución gracias a la optimización realizada, pero aun así, la implementación en CUDA alcanzó una eficiencia computacional mucho mayor, demostrando así la utilidad y superioridad de la paralelización obtenida al ejecutar instrucciones en dispositivos de procesamiento gráfico.

Trabajo Futuro

Teniendo una implementación funcional del método capaz de resolver las ecuaciones de *shallow water*, se necesita continuar el trabajo realizado para poder hacer simulaciones numéricas de tsunamis. Para lograr esto, se requiere que el método sea capaz de lidiar con la inundación, la cual consiste en la característica principal del fenómeno y provee la información más importante como es el alcance del agua o las zonas que serán cubiertas por ella. Dado que el sistema de ecuaciones con el que se representa el problema deja de ser válido para las denominadas "zonas secas", se hace necesaria la

formulación de un método que maneje esta situación y se acople al método actual. Este escenario puede suponer la elección de otro sistema de ecuaciones a resolver, la inclusión de nuevos parámetros con el fin de adaptar la representación al nuevo problema que se quiere solucionar o la modificación del método mismo. Debido a la importancia de esto para completar el trabajo realizado, el manejo de la inundación será el principal y más próximo objeto de estudio, pretendiendo obtener un método capaz de lidiar con esta situación.

Luego de haber resuelto lo anteriormente mencionado, se pretende aumentar la complejidad de la experimentación, validando la solución propuesta tanto en contextos artificiales como en un escenario de tsunami real. Para esto, se evaluará el desempeño del método implementado en los casos de referencia más conocidos, como lo son los escenarios de "*dam break*" y "*run up*", y también al simular con los datos de un tsunami real. Adicionalmente, se espera comparar el desempeño con otras implementaciones que busquen resolver el mismo problema, siendo COMCOT una de las más conocidas.

Finalmente, se espera poder seguir realizando optimizaciones de bajo nivel en la implementación paralelizada, valiéndose de herramientas más avanzadas proporcionadas por la arquitectura CUDA, con el objetivo de reducir aun más los tiempos computacionales y aumentar la eficiencia del algoritmo.

Bibliografía

[1] National Geophysical Data Center. Southern Chile Earthquake and Tsunami, 22 May 1960 [En línea]. <https://www.ngdc.noaa.gov/hazard/22may1960.html>. [Visitado el 15 de Diciembre de 2016].

[2] National Geophysical Data Center. 22 DE MAYO DE 1960 PUERTO MONTT, VALDIVIA CHILE EARTHQUAKE AND TSUNAMI [En línea]. <https://www.ngdc.noaa.gov/hazardimages/event/show/25>. [Visitado el 15 de Diciembre de 2016].

[3] National Geophysical Data Center. Sumatra, Indonesia Earthquake and Tsunami, 26 December 2004 [En línea]. <https://www.ngdc.noaa.gov/hazard/26dec2004.html>. [Visitado el 15 de Diciembre de 2016].

[4] National Geophysical Data Center. Great Tohoku, Japan Earthquake and Tsunami, 11 March 2011 [En línea]. <https://www.ngdc.noaa.gov/hazard/11mar2011.html>. [Visitado el 15 de Diciembre de 2016].

[5] United States Geological Survey. M7.8 - 56km NNE of Amberley, New Zealand [En línea]. <http://earthquake.usgs.gov/earthquakes/eventpage/us1000778i>. [Visitado el 15 de Diciembre de 2016].

[6] S. Li y P. Huang y J. Li. A modified lattice Boltzmann model for shallow water flows over complex topography. *International Journal for Numerical Methods in Fluids*, 77:441–458, 2015.

[7] S. Wood y R. Deiterding. *A dynamically adaptive lattice Boltzmann method for flapping wing aerodynamics*, pages 1082–1088. Korean Society of Mechanical Engineers, 2015.

[8] E. Salomons y W. Lohman y H. Zhou. Simulation of sound waves using the lattice boltzmann method for fluid flow: Benchmark cases for outdoor sound propagation. *PLoS ONE*, 11(1):1–19, 2016.

[9] A. H. Hedjripour y D. P. Callaghan y T. E. Baldock. Generalized transformation of the lattice boltzmann method for shallow water flows. *Journal of Hydraulic Research*, 54(4):371–388, 2016.

[10] R. Sadourny. The Dynamics of Finite-Difference Models of the Shallow-Water Equations. *Journal of Atmospheric Sciences*, 32(4):680–689, 1975.

[11] V. Casulli y E. Cattani. Stability, Accuracy and Efficiency of a Semi-Implicit Method for Three-Dimensional Shallow Water Flow. *Computers and Mathematics with Applications*, 27(4):99–112, 1994.

[12] M. Israeli y N. Naik y M. Cane. An Unconditionally Stable Scheme for the Shallow Water Equations. *Monthly Weather Review*, 128(3):810–823, 2000.

[13] M. Rasulov y Z. Aslan y O. Pakdil. Finite differences method for shallow water equations in a class of discontinuous functions. *Applied Mathematics and Computation*, 160(2):343–353, 2005.

[14] A. Delis y Th. Katsaounis. Numerical solution of the two-dimensional shallow water equations by the application of relaxation methods. *Applied Mathematical Modelling*, 29(8):754–783, 2005.

[15] Y. Skiba y D. Filatov. Conservative arbitrary order finite difference schemes for shallow-water flows. *Journal of Computational and Applied Mathematics*, 218:579–591, 2008.

[16] C. Lu y J. Qiu. Simulations of Shallow Water Equations with Finite Difference Lax-Wendroff Weighted Essentially Non-oscillatory Schemes. *Journal of Scientific Computing*, 47:281–302, 2011.

[17] Z. Luo y J. Gao. The numerical simulations based on the NND finite difference scheme for shallow water wave equations including sediment concentration. *Computer Methods in Applied Mechanics and Engineering*, 294:245–258, 2015.

[18] Z. Luo y J. Gao y Z. Xie. Reduced-order finite difference extrapolation model based on proper orthogonal decomposition for two-dimensional shallow water equations including sediment concentration. *Journal of Mathematical Analysis and Applications*, 429(2):901–923, 2015.

[19] G. Li y V. Caleffi y Z. Qi. A well-balanced finite difference WENO scheme for shallow water flow model. *Applied Mathematics and Computation*, 265:1–16, 2015.

[20] A. Kurganov y E. Tadmor. New high resolution central schemes for nonlinear conservation laws and convection-diffusion equations. *Journal of Computational Physics*, 160:241–282, 2000.

[21] A. Kurganov y D. Levy. Central-upwind schemes for the Saint-Venant system. *Mathematical Modelling and Numerical Analysis*, 36:397–425, 2002.

[22] A. Kurganov y G. Petrova. Central-upwind schemes on triangular grids for hyperbolic systems of conservation laws. *Numerical Methods for Partial Differential Equations*, 21(3):536–552, 2005.

[23] A. Kurganov y G. Petrova. A second-order well-balanced positivity preserving central-upwind scheme for the Saint-Venant system. *Communications in Mathematical Sciences*, 5:133–160, 2007.

[24] S. Bryson y D. Levy. Balanced central schemes for the shallow water equations on unstructured grids. *SIAM J. Scientific Computing*, 27(2):532–552, 2005.

[25] S. Bryson y Y. Epshteyn y A. Kurganov y G. Petrova. Well-balanced positivity preserving central-upwind scheme on triangular grids for the Saint-Venant system. *ESAIM: Mathematical Modelling and Numerical Analysis*, 45(3):423–446, 2011.

[26] E. Audusse y F. Bouchut y M.-O. Bristeau y R. Klein y B. Perthame. A fast and stable well-balanced scheme with hydrostatic reconstruction for shallow water flows. *SIAM J. Sc. Comp.*, 25(6):2050–2065, 2004.

[27] C. Berthon y F. Foucher. Hydrostatic upwind schemes for shallow-water equations. *Springer Proceedings in Mathematics*, 4:97–105, 2011.

[28] O. Delestre y S. Cordier y F. Darboux y F. James. A limitation of the hydrostatic reconstruction technique for Shallow Water equations. *Comptes Rendus Mathématique, Elsevier Masson*, 350(13–14):677–681, 2012.

[29] D. Dutykh y D. Clamond. Modified shallow water equations for significantly varying seabeds. *Applied Mathematical Modelling*, 40(23–24):9767–9787, 2016.

[30] Y. Xing. High order finite volume WENO schemes for the shallow water flows through channels with irregular geometry. *Journal of Computational and Applied Mathematics*, 299:229–244, 2016.

[31] Hazim Qiblawey A. Beljadid y A. Mohammadiana. Stability analysis of unstructured finite volume methods for linear shallow water flows using pseudospectra and singular value decomposition. *Advances in Water Resources*, 96:127–144, 2016.

[32] School of Civil and Environmental Engineering, Cornell University. COMCOT User Manual [En línea]. <https://piazza-resources.s3.amazonaws.com/ha634t4ntvkmt/hberwqu7fjf1of/COMCOT_user_manual_v1_6.pdf?AWSAccessKeyId=AKIAIEDNRLJ4AZKBW6HA&Expires=1481817620&Signature=9%2FarNKV2ZtiS9b%2Fsp54sS4W9ovw%3D>. [Visitado el 15 de Diciembre de 2016].

[33] ANUGA [En línea]. <https://anuga.anu.edu.au/>. [Visitado el 15 de Diciembre de 2016].

[34] G.McNamara y G. Zanetti. Use of the boltzmann equation to simulate lattice-gas automata. *Physical Review Letters*, 61:2332–2335, 1988.

[35] F. J. Higuera y J. Jiménez. Boltzmann approach to lattice gas simulations. *EPL (Europhysics Letters)*, 9(7):663, 1989.

[36] S. Succi. *The Lattice Boltzmann Equation: for Fluid Dynamics and Beyond*. Oxford University Press, 1 edition, 2001.

[37] J. G. Zhou. *Lattice Boltzmann Method for Shallow Water Flows*. Springer Berlin Heidelberg, 1 edition, 2004.

[38] M. C. Sukop y D. T. Thorne Jr. *Lattice Boltzmann Modeling: An Introduction for Geoscientists and Engineers*. Springer Berlin Heidelberg, 2 edition, 2006.

[39] A. A. Mohamad. *Lattice Boltzmann Method: Fundamentals and Engineering Applications with Computer Codes*. Springer Berlin Heidelberg, 1 edition, 2011.

[40] R. Salmon. The lattice boltzmann method as a basis for ocean circulation modeling. *Journal of Marine Research*, 57:503–535, 1999.

[41] G. Thömmes y M. Seaïd y M. K. Banda. Lattice boltzmann methods for shallow water flow applications. *International Journal for Numerical Methods in Fluids*, 55:673–692, 2007.

[42] M. Geveler y D. Ribbrock y D. Göddeke y S. Turek. *Lattice-Boltzmann Simulation of the Shallow-Water Equations with Fluid-Structure Interaction on Multi- and Manycore Processors*, pages 92–104. Springer Berlin Heidelberg, 2010.

[43] J. G. Zhou. Enhancement of the labswe for shallow water flows. *Journal of Computational Physics*, 230:394–401, 2010.

[44] M. Varmazyar y M. Bazargan. Development of a thermal lattice Boltzmann method to simulate heat transfer problems with variable thermal conductivity. *International Journal of Heat and Mass Transfer*, 59:363–371, 2013.

[45] A. RABIENATAJ y M. FARHADI y M. JOURABIAN. Lattice Boltzmann Simulation of Heat Transfer Enhancement During Melting by Using Nanoparticles. *IJST, Transactions of Mechanical Engineering*, 37(M1):23–37, 2013.

[46] A. Kupershtokha y D. Medvedev. Lattice Boltzmann equation method in electrohydrodynamic problems. *Journal of Electrostatics*, 64:581–585, 2006.

[47] G. Tang y X. Li y W. Tao. Microannular electro-osmotic flow with the axisymmetric lattice Boltzmann method. *Journal of Applied Physics*, 108(11):114903–114903–11, 2010.

[48] H. Aono y A. Gupta y D. Qi y W. Shyy. The lattice boltzmann method for flapping wing aerodynamics. *40th Fluid Dynamics Conference and Exhibit, Chicago, Illinois*, 2010.

[49] E. Magnus. The lattice boltzmann method with applications in acoustics. Master's thesis, Norwegian University of Science and Technology, Noruega, 2009.

[50] Palabos [En línea]. <`http://www.palabos.org/`>. [Visitado el 15 de Diciembre de 2016].

[51] OpenLB Open source lattice Boltzmann code [En línea]. <`http://optilb.org/openlb/`>. [Visitado el 15 de Diciembre de 2016].

[52] PowerFLOW [En línea]. <`http://exa.com/en/product/simulation-tools/powerflow-cfd-simulation`>. [Visitado el 15 de Diciembre de 2016].

[53] NVIDIA Corporation. CUDA C Programming Guide v8.0 [En línea]. <`http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html`>. [Visitado el 15 de Diciembre de 2016].

[54] NumPy [En línea]. <`http://www.numpy.org/`>. [Visitado el 15 de Diciembre de 2016].