UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

Localización de fronteras en dominios irregulares para aplicar el método de lattice Boltzmann

T E S I S
QUE PARA OPTAR POR EL GRADO DE:
MAESTRO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

P R E S E N T A:
A I T O R L A N D E R D E I C A Z A A S T I Z

Director de Tesis:
Dr. Luis Miguel De La Cruz Salas
Instituto de Geof ísica, UNAM

Ciudad Universitaria, CD. MX. Febrero, 2017

Prefacio

El estudio del comportamiento de los fluidos por medio de técnicas numéricas se ha trabajado con una gran variedad de métodos. Muchos de ellos se centran en resolver las ecuaciones de Navier-Stokes, por medio de su discretización. Un método que ha cobrado fuerza últimamente es el de las Redes de Boltzmann o de Lattice Boltzmann. De la enorme cantidad de trabajos que existen al respecto, hay una cantidad menor que considera el problema concreto de interacción del fluido con las paredes que lo contienen, entendiendo por interacción una influencia mutua entre uno y otro.

Un problema que es necesario resolver para empezar a trabajar con fronteras móviles es la identificación de los nodos que quedan al interior de una frontera, los que quedan en su exterior, y cuáles de esos dos conjuntos de nodos están suficientemente cerca de la frontera.

El presente trabajo está organizado como se describe a continuación:

Capítulo 1 Se describe el problema a resolver. También se mencionan los objetivos de la tesis y se resaltan las hipótesis tomadas en cuenta.

Capítulo 2 Se describe el método de Lattice de Boltzmann junto con su precisión. Además se discute un método para considerar fronteras curvas. Finalmente, se discute la interacción que tiene el fluido con las fronteras.

Capítulo 3 Se discute la implementación del método para localizar fronteras de dominios irregulares, y cómo a partir de estas, se encuentra identifica los tipos de nodos del dominio total.

Capítulo 4 Se presentan los resultados obtenidos.

Capítulo 5 Se discuten los resultados obtenidos, y se proponen mejoras al algoritmo.

Agradecimientos

Agradezco en primer lugar a mi tutor Luis Miguel de la Cruz Salas por sus importantes consejos, su guía y señalarme caminos más prometedores a lo largo de todo este proceso, pero sobre todo por la confianza en mi capacidad para lograr este trabajo, que me impulsó más de una vez a seguir adelante.

A mis sinodales Jorge Alberto Márques Flores, Miguel Ángel Padilla Castañeda, Ricardo Atahualpa Solórzano Kraemer y Edgar Vázquez Luis, cuyas correcciones y rápidez para leer el trabajo me ayudaron en más de una forma.

A mi familia que me apoyaron, y que también me ayudaron a revisar y encontrar errores en mi código.

A mis amigos Jeourkgeleks, Miguel B., Nadia R., Miguel P., Hugo A., Eduardo H., Saúl H., Isabel G., Fernando V., Majo, Arturo V., Karen, Vladimir, Kat, Jesús P., Jair, Lety, Luis M., Gus, Elisa, Fausto, Mirna, Pedro M., Andrés P., Israel, Diego e Isael, que sin su amistad no habría llegado a este punto. Cada uno fue fundamental para esto.

A Lulú, Amalia y Cecilia, que nos demostraron su cariño y dan ejemplo de cómo debe ser un personal administrativo más humano.

Finalmente al Consejo Nacional de Ciencia y Tecnología (Conacyt) por haberme otorgado una beca para realizar mis estudios de Maestría.

Aitor.

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO I

Introducción

I.1. Descripción del problema

I.1.1. Motivación

Típicamente, un problema físico no se ve restringido a un dominio estático, entendiendo por dominio estático, un dominio que no cambia con el tiempo. Su dominio puede cambiar por muchas razones, pero estas se pueden englobar en tres categorías. La frontera del dominio se modifica por lo que contiene el dominio o por lo que está afuera, o por ambos motivos.

Un problema físico que es ejemplo del último caso es un problema médico conocido como aneurisma. Un aneurisma es un ensanchamiento o abombamiento anormal de una parte de una arteria debido a debilidad en la pared del vaso sanguíneo que provoca que éste protruya (sobresalga) o se abombe.[1] Se denomina aneurisma cerebral cuando ocurre en un vaso sanguíneo del cerebro,[2] aunque en realidad estos pueden ocurrir en cualquier arteria del cuerpo.

No es claro qué causa los aneurismas o cuándo un aneurisma es dañino para una persona, por lo que un estudio de los fenómenos físicos implicados puede dar algunas respuestas.

Puede pensarse a las arterias como pequeños tubos con paredes elásticas en los que se transporta la sangre, que es un fluido no-newtoniano ([1], [8]) .

En la sangre la cantidad de partículas es muy grande, como es típico de cualquier fluido, por lo que un análisis individual de cada partícula sería totalmente impráctico por la carga de peso computacional que requeriría. La alternativa es suponer a la sangre como un fluido continuo en el espacio con ciertas características físicas que pueden reconocerse, tales como su viscocidad, tensión superficial, compresibilidad, capilaridad y conducción térmica.

El entendimiento de cómo se comporta el fluido, en el tratamiento de una persona con un aneurisma, puede dar información de cómo debería colocarse una malla para la recuperación óptima del paciente, o de alguna otra alternativa.

Lo discutido anteriormente aplica a dominios que cambian con el tiempo y muestra la importancia de tener un método que ayude a describir el dominio[3]. Sin embargo, si el método sirve para describir el dominio para un instante de tiempo, este es perfectamente aplicable a un problema donde la frontera sea fija.

En problemas con frontera fija, la tarea de describir el dominio se tiene que hacer una sola vez. Pero cuando el dominio tiene cierta complejidad, el dominio es muy grande o son muchos los dominios en que hay que hacer pruebas, conviene un método automatizado para hacer las descripciones de ellos.

Un ejemplo en que muchos dominios se quisieran investigar, es el de los perfiles de alas de avión. Puede ser que los perfiles a investigar ya se tengan caracterizados (en su dominio espacial), o que un solo perfil se modifique y se desee observar el efecto de la modificación.

Otro ejemplo es el análisis de un dominio con obstáculos por el que fluye un fluido. Los obstáculos pueden ser de formas muy variadas, por lo que la localización de fronteras para estos obstáculos irregulares es muy útil.

[1] `https://medlineplus.gov/spanish/ency/article/001122.htm`

[2] `https://medlineplus.gov/spanish/ency/article/001414.htm`

[3] Para el método de Lattice Boltzmann que aborda esta tesis, se entiende por descripción del dominio, a una categorización de nodos, repartidos espacialmente, en un dominio más grande que contiene al dominio original.

I.1.2. Simplificaciones al problema real

Una simplificación al problema es tratarlo como si fuera bidimensional. Una ventaja de esto es que la visualización es más sencilla y el poder de cómputo necesario se reduce. Para problemas físicos con una simetría de translación[4], la restricción a dos dimensiones no debe modificar cualitativamente los resultados físicos que se esperan. Sin embargo, cuando el problema carece de esta simetría, la interacción con la dimensión que se elimina podría ser importante.

Con esta simplificación las fronteras, que representan las paredes de la arteria, son ahora unidimensionales. Se piensa que dichas fronteras existen arriba y abajo en el dominio. Del lado izquierdo y derecho entra y sale el fluido.

Una modelación de un aneurisma, requeriría que las paredes representativas de la arteria cambien sus propiedades debido al adelgazamiento de las mismas paredes. Este adelgazamiento puede ocurrir en sólo una sección de ella, y para simplificar más el problema, uniforme. Esto quiere decir que en las partes donde no hay adelgazamiento, se puede pensar a la pared como rígida, y que no estarán presentes heterogenidades en la constitución de la pared de la arteria. También, por simplicidad, se supone que la sección transversal de la arteria es constante.

De los numerosos métodos para encontrar la solución a dicho problema una posibilidad es uno conocido como el método de *Lattice Boltzmann*.

La principal razón por la que se utiliza este método es porque es fácilmente paralelizable por la localidad de las operaciones que realiza, y además en numerosos casos da resultados muy buenos que modelan adecuadamente el comportamiento de fluidos.

Sin embargo, este método se ha usado poco para modelar fluidos no-newtonianos, siendo la sangre uno de ellos. El tipo de interacción entre las partículas de dichos fluidos es mucho más complejo, y posiblemente menos local de lo que el método soporta en su forma más estándar.

Así, para simplificar más el problema, se puede considerar que el fluido entre las paredes de la arteria es un fluido newtoniano, para asemejar más al uso estándar del método de Lattice Boltzmann. Ésta simplificación tiene como resultado una alteración en el flujo de la sangre en la arteria (véase [1]).

Finalmente, el flujo en el sistema circulatorio no es sostenido, sino que está determinado por el mismo ritmo del corazón. Puede suponerse algún flujo de inserción cuyo ritmo no sea constante, sino oscilante, como por ejemplo un flujo senoidal.

I.1.3. ¿Qué falta?

Un problema de ecuaciones diferenciales está mal planteado cuando las condiciones a la frontera no están prescritas adecuadamente.

En esencia, el problema se transforma en encontrar la forma en que la pared de la arteria interactúa con el fluido.

Desde el punto de vista físico, las partículas del fluido empujan esta pared adelgazada o debilitada, y qué tanto se deforme depende de las propiedades elásticas de esta pared y de otras propiedades del fluido.

[4]Una simetría de translación para un problema físico significa que si se hace una translación del sistema de referencias, la geometría y características del problema se ven exactamente iguales.

El campo de velocidades es una de las piezas claves en el método de Lattice Boltzmann, y además, en el rango de trabajo del método, el fluido es incompresible, por lo que la densidad también es conocida. En esencia, esto es suficiente para conocer el momento por unidad de volumen (o área, en el caso bidimensional) que lleva el fluido en algún punto de la red.

Conocido el momento que lleva el fluido, y con un modelo del intercambio de momento del fluido con las paredes de la arteria, el movimiento de la pared puede ser encontrado.

Este modelo de intercambio de momento entre la pared y el fluido, puede ser bastante complejo. Los problemas dinámicos empiezan cuando uno considera la colisión entre dos partículas que colisionan, porque en general, las direcciones después de la colisión no son conocidas, y entonces el problema está subdeterminado. También, el tipo de colisión con la pared puede ser otra complicación. La colisión puede ser inelástica (y en general lo es), con lo que existe una disipación de energía en la colisión.

Una vez determinado el modelo específico que dicta el intercambio de momento entre la frontera y el fluido, el movimiento de la frontera puede encontrarse. Sin embargo, esto sólo resuelve el problema en un paso de tiempo.

También, el fluido debe afectarse de cierta forma con el intercambio de momento. Esta parte, puede encapsularse en un problema de condiciones de frontera.

Otra parte a considerar es que al moverse y deformarse la frontera, el dominio englobado por la frontera se modifica. Esto más que nada tiene repercusiones para el método de Lattice Boltzmann, que funciona a través de un modelo de red de nodos. Entonces, la interacción de la frontera con el fluido se descompone en tres partes principales: movimiento de la frontera, condiciones de frontera y modificación del dominio. Cada una de estas relacionada con las otras dos.

La siguiente sección muestra los objetivos alcanzados por la tesis.

I.2. Objetivos de la tesis

- Investigar condiciones de frontera que contemplen fronteras curvas o rectas para englobar dominios irregulares.
- Discutir un modelo de colisión que permita conectar el movimiento de la frontera con las condiciones de frontera.
- Implementar un método para la categorización de nodos de acuerdo a la posición específica de la frontera y requerimentos de las condiciones de frontera.

I.3. Aportes del trabajo

Aparte de los objetivos descritos en la sección anterior, los aportes del trabajo son los siguientes:

- Una recopilación del material teórico indispensable para entender el funcionamiento y alcances del método de Lattice Boltzmann.

- Implementación de un método asociativo que permita generar a partir de una imagen de un dominio bidimensional (del tipo y-simple o con una pequeña transformación, también x-simple), un conjunto de puntos que representen la frontera del conjunto.
- Implementación de un método que permita, a partir de un conjunto de coordenadas de puntos, identificar los nodos frontera. Estos nodos frontera se usan para fijar condiciones de frontera específicas.
- Implementación de un método que identifique los nodos interiores y exteriores delimitados por la frontera. La determinación del tipo de los nodos frontera, mencionada en el punto anterior, permite la identificación rápida de los demás nodos. Como estos nodos frontera están cerca de la frontera, se les puede aplicar un criterio de la regla del producto cruz, que normalmente sólo es aplicable para polígonos convexos, para determinar si son interiores o exteriores.[5]
- Implementación de una función que elija del conjunto de nodos frontera, los que un método requiere para aplicar condiciones de frontera.
- Extensión de la identificación de los tipos de nodos en el dominio total, para regiones con huecos.
- Implementación del método de Lattice Boltzmann mostrando la utilidad del método incluso para las condiciones de frontera de más fácil implementación.

[5]El criterio de la regla del producto cruz, para determinar si un punto está dentro de un polígono convexo en un plano, pide que el signo de la componente resultante del producto cruz entre un vector tangencial a una arista del polígono, y un vector que localice el punto desde el orígen del primer vector, sea siempre el mismo para cualquier arista que se elija. Los vectores tangenciales a la frontera del polígono tienen un sentido específico, que depende de una convención inicial que establece cómo se recorre el polígono.

CAPÍTULO II

Marco teórico

A continuación se presenta una pequeña reseña histórica del origen del método y una descripción de su predecesor, conocida como la red del autómata de gas. Se cree conveniente porque ayuda a comprender el orden en que se hacen los pasos del método de Lattice Boltzmann y el porqué se hacen. Actualmente el método se sostiene por sí solo, y de hecho se ha visto su equivalencia con las ecuaciones de Navier-Stokes (véase Chen et al. [5])

II.1. Historia del método de Lattice Boltzmann

El método de Lattice Boltzmann surge a partir de la red del autómata de gas (LGA, por sus siglas en inglés). Este es un método que combina elementos de autómatas y la ecuación de Boltzmann para modelar la evolución de un fluido.

El primer LGA fue presentado por Hardy, Pomeau y de Pazzis en 1973[1] [13], y en él se considera un gas formado por partículas que se mueven entre los puntos de una red cuadrada. Sin embargo, el movimiento de estas partículas se restringe específicamente a movimientos directos entre puntos vecinos de la red. Esta restricción discretiza las direcciones de las velocidades que pueden tener las partículas. En cuanto a la magnitud de estas velocidades, se consideran unitarias, en el sentido de que por cada paso de tiempo las partículas avanzan completamente al punto vecino.

Los movimientos que siguen las partículas están dictados por reglas de interacción específicas. También, el estado en un punto de la red está determinado por un número Booleano que representa si está ocupado o no ese punto de la red. El estado del punto de la red cambia de acuerdo a su estado actual y a los estados de los puntos adyacentes de la red, considerando las velocidades de las partículas en esos puntos de la red. Esto constituye la parte del nombre que hace alusión a los autómatas.

Así, inicialmente todas las partículas tienen una velocidad que determina hacia qué puntos vecinos se mueven. Para avanzar la simulación se hace un paso temporal en el que todas las partículas se mueven de acuerdo con su velocidad. A este paso se le conoce como el paso de *flujo* (ver figuras II.1b y II.1c), e implica sobre el problema una discretización temporal.

Después, las partículas intentan ocupar el punto vecino al que su velocidad está dirigida. El éxito de la ocupación depende de si hay otras partículas que quieran ocupar el mismo punto de la red. Este caso corresponde a una colisión entre partículas, como por ejemplo, una partícula llegando a un punto de la red desde la izquierda y otra partícula llegando desde la derecha. Según las reglas de este autómata, en este caso, las partículas salen en ángulos rectos a la dirección que llegaron[2]. A este paso se le conoce como el paso de *colisión* (véase las figuras II.1a y II.1b).

Sin embargo, los mayores problemas de LGA son: ruido intrínseco, falta de invariancia Galileana, presiones dependientes de la velocidad (es decir, soluciones no físicas) y grandes viscosidades numéricas.

En 1986, Frisch, Hasslacher y Pomeau (FHP) obtuvieron las ecuaciones correctas de Navier-Stokes usando una red hexagonal. También, las ecuaciones de Lattice Boltzmann se usaron junto con LGA para calcular viscosidad (Frisch et al. [10]). Para eliminar ruido estadístico, en 1988, McNamara y Zanetti [22] se deshicieron del operador Booleano de LGA que involucraba las variables de ocupación de las partículas al despreciar las correlaciones entre partículas e introducir funciones de distribución promedio, dando origen al método de Lattice Boltzmann.

[1] También conocido como HPP por las siglas de sus autores.

[2] Hay muchas variantes del autómata, por lo que la dirección específica a la que salen puede cambiar de uno a otro, o bien, consideran otro tipo particular de regla.

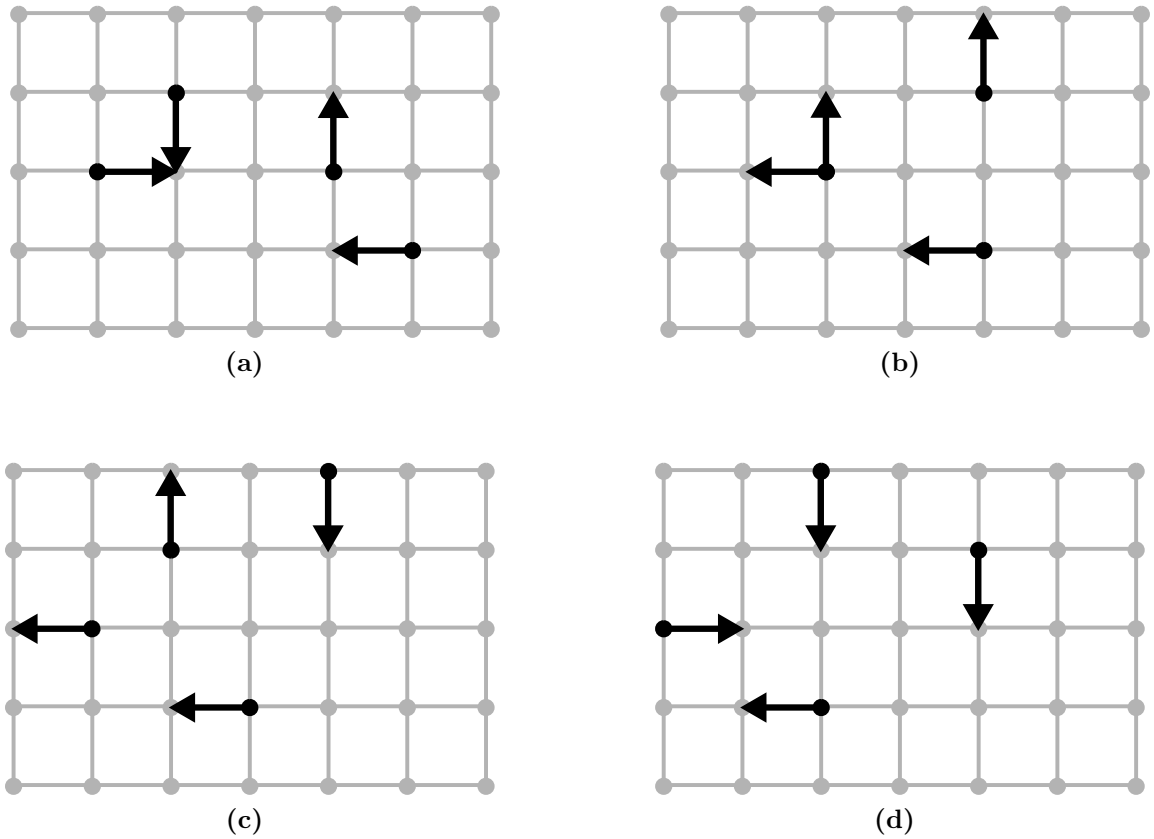


Figura II.1: *Cuatro partículas y sus velocidades en el autómata de gas, moviéndose de un punto a otro en la red. (a) Dos de las partículas entrarán en colisión al siguiente paso de tiempo. (b) Las partículas se mueven de acuerdo a su colisión. Las dos partículas que entraron en colisión cambian su velocidad debido a las reglas específicas del modelo. Una partícula rebotará con la pared. (c) Al rebotar la partícula con la pared, invierte dirección de su velocidad. Dos partículas más rebotarán con la pared. (d) Las dos partículas que rebotan en la pared, invierten su velocidad. Las otras dos partículas avanzan según su velocidad.*

Higuera y Zanetti [15] hicieron una simplificación importante al método de Lattice Boltzmann al presentar una ecuación de Lattice Boltzmann[3] con un operador de colisión linealizado que supone que la distribución está cerca del estado de equilibrio local. Una versión particularmente simple del operador de colisión linealizado basado en el modelo de colisión de Bhatnagar-Gross-Krook (BGK) se introdujo independientemente por muchos autores, que incluyen a Koelman [17] y Chen et al. [6].

II.2. El método de Lattice Boltzmann

Como se vió en la sección anterior, el método de Lattice Boltzmann, históricamente no deriva directamente de la ecuación de Boltzmann (que se analiza enseguida). Más bien, utiliza la ecuación de Boltzmann para dictar las reglas del comportamiento físico que siguen las partículas.

Así, la física del problema se modela a través de la ecuación de Boltzmann, y las simplificaciones que se hagan en ella están íntimamente ligadas a los resultados que se espera simular.

[3] Véase la ecuación (II.55)

II.2.1. Ecuación de Boltzmann

La ecuación de Boltzmann, también conocida como la ecuación de transporte de Boltzmann, describe la distribución estadística de partículas en un fluido. Es una ecuación para la evolución temporal de la función de distribución de partículas en el espacio fase $f(\vec{r}, \vec{\xi}, t)$, donde $\vec{r}$ es un punto en el espacio, $\vec{\xi}$ es la velocidad (microscópica, i.e., la velocidad que lleva una sola partícula), y t es el tiempo. Entonces, esta función de distribución es una función de 7 variables, y da el número de partículas por unidad de volumen en el espacio fase de una partícula. Es el número de partículas por unidad de volumen, tales que su velocidad (microscópica) es aproximadamente (ξ_x, ξ_y, ξ_z) cerca del punto (x, y, z) y del tiempo t. La ecuación continua clásica de Boltzmann es una ecuación integro-diferencial para una función de distribución de una partícula $f(\vec{r}, \vec{\xi}, t)$ que se escribe como

$$\frac{\partial f}{\partial t} + \vec{\xi} \cdot \frac{\partial f}{\partial \vec{r}} + \vec{F} \cdot \frac{\partial f}{\partial \vec{\xi}} = \Omega(f) \tag{II.1}$$

donde $\vec{\xi}$ es la velocidad de la partícula, $\vec{F}$ es la fuerza de cuerpo y $\Omega(f)$ es la integral de colisión.

Las fuerzas de cuerpo, que son fuerzas que actúan en cada elemento de volumen del dominio donde se encuentre materia, tales como la fuerza de gravedad, se ignoran en esta tesis.

Por esto, el tercer término en la ecuación (II.1) se elimina, y la ecuación queda como

$$\frac{\partial f}{\partial t} + \vec{\xi} \cdot \frac{\partial f}{\partial \vec{r}} = \Omega(f) \tag{II.2}$$

En cuanto a la integral de colisión $\Omega(f)$, o término de colisión, es una cantidad que describe las colisiones entre partículas. Las "reglas" de cómo se distribuyen las partículas después de su colisión se describen con este término. Con esto se ve la importancia de la elección en este término, y se explica que la física principal del problema esté contenida aquí. Sin embargo, la discretización en el espacio de velocidades que se hace en la ecuación (II.2), modifica en cierta medida lo que se espera observar, y los resultados deben tratarse con cuidado. Se habla más de esto en la siguiente sección donde se discretiza la ecuación (II.2).

Es de esperar que este término de colisión sea muy complejo, dada la gran cantidad de partículas que hay en un fluido. Para facilitar las soluciones numéricas y analíticas de la ecuación de Boltzmann, este término de colisión se reemplaza por lo que se conoce como aproximación BGK o modelo de relajación temporal único (*single-relaxation-time model*). Está dado por la siguiente expresión:

$$\Omega_{\mathrm{BGK}}(f) = -\frac{f - g}{\tau}, \tag{II.3}$$

donde τ es un tiempo de relajación único asociado con la relajación post-colisión a un estado de equilibrio para la función f, representado por g. Este modelo se entiende como una evolución del sistema a estados de equilibrio sucesivos. La expresión (II.3) fue propuesta por He y Luo [14].

Las siglas BGK corresponden a las iniciales de los apellidos Bhatnagar, Gross y Krook, que publicaron un artículo (Bhatnagar et al. [2]) donde discuten un modelo de colisión para gases. Cuando el método de Lattice Boltzmann usa la aproximación BGK también se le conoce por la sigla LBGK (véase II.6).

Con esto, la ecuación de Boltzmann queda con la siguiente forma:

$$\frac{\partial f}{\partial t} + \vec{\xi} \cdot \frac{\partial f}{\partial \vec{r}} = -\frac{f - g}{\tau}. \tag{II.4}$$

Alternativamente, hay otra formulación en la que se trabaja con múltiples tiempos de relajación, que se conoce como MRT (*Multi-Relaxation-Time*, véase [19]).

A partir de la función de distribución $f(\vec{r},\vec{\xi},t)$ se obtienen las variables hidrodinámicas macroscópicas ρ, $\vec{u}$ y T, es decir, la densidad, la velocidad promedio del fluido y la temperatura, respectivamente. Estas son los momentos (de la velocidad microscópica ξ) de la función de distribución f:

$$\rho = \int f \,\mathrm{d}\vec{\xi}, \tag{II.5a}$$

$$\rho\vec{u} = \int \vec{\xi} f \,\mathrm{d}\vec{\xi}, \tag{II.5b}$$

$$\rho\varepsilon = \frac{1}{2}\int (\vec{\xi}-\vec{u})^2 f \,\mathrm{d}\vec{\xi}. \tag{II.5c}$$

La energía ε puede escribirse en términos de la temperatura T para el modelo BGK:

$$\varepsilon = \frac{D_0}{2}RT = \frac{D_0}{2}N_A k_B T, \tag{II.6}$$

donde R, D_0, N_A, y k_B son la constante de los gases ideales, el número de grados de libertad de una partícula, el número de Avogadro, y la constante de Boltzmann, respectivamente.

Resumiendo un poco lo anterior, la física del problema se obtiene totalmente de la función $f(\vec{r},\vec{\xi},t)$. Si se encuentra ésta, se considera el problema en cuestión resuelto. El método secciona en dos partes principales la forma para encontrar esta función, conocidas como el paso de flujo y el paso de colisión.

A continuación se discretiza la ecuación de Boltzmann tanto en el tiempo como en su espacio fase. La discretización en el espacio fase impone una discretización para las velocidades, además de una discretización espacial.

II.2.2. Discretización de la ecuación de Boltzmann

En el siguiente análisis, se usa la ecuación de Boltzmann con el término de colisión con el modelo BGK, es decir, la ecuación (II.4). Como función de equilibrio se toma la distribución de Maxwell-Boltzmann:

$$g(\vec{r},\vec{\xi},t) \equiv \frac{\rho}{(2\pi RT)^{D/2}} \exp\left(-\frac{(\vec{\xi}-\vec{u})^2}{2RT}\right), \tag{II.7}$$

con D la dimensión del sistema. Dado que la derivada temporal a lo largo de la línea característica de $\vec{\xi}$ puede escribirse como[4]

$$\frac{\mathrm{d}}{\mathrm{d}t} \equiv \frac{\partial}{\partial t} + \vec{\xi}\cdot\nabla, \tag{II.8}$$

entonces la ecuación (II.4) se reescribe como una ecuación diferencial ordinaria:

$$\frac{\mathrm{d}f}{\mathrm{d}t} + \frac{1}{\tau}f = \frac{1}{\tau}g. \tag{II.9}$$

Usando el método de factor integrante, se multiplica por una función $\mu = \mu(t)$

$$\mu\frac{\mathrm{d}f}{\mathrm{d}t} + \frac{\mu}{\tau}f = \frac{\mu}{\tau}g. \tag{II.10}$$

[4]Para una función $\phi(x,y,z,t)$ con $x = x(t)$, $y = y(t)$, $z = z(t)$, por regla de la cadena se tiene $\frac{\mathrm{d}\phi(x,y,z,t)}{\mathrm{d}t} = \frac{\partial\phi}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial\phi}{\partial y}\frac{\partial y}{\partial t} + \frac{\partial\phi}{\partial z}\frac{\partial z}{\partial t} + \frac{\partial\phi}{\partial t}\frac{\partial t}{\partial t} = \nabla\phi\cdot(\frac{\partial x}{\partial t},\frac{\partial y}{\partial t},\frac{\partial z}{\partial t}) + \frac{\partial\phi}{\partial t} = \frac{\partial\phi}{\partial t} + \vec{\xi}\cdot\nabla\phi = \left(\frac{\partial}{\partial t} + \vec{\xi}\cdot\nabla\right)\phi$

El miembro izquierdo de (II.10) se ve como $\frac{\mathrm{d}\mu f}{\mathrm{d}t} = \mu\frac{\mathrm{d}f}{\mathrm{d}t} + \frac{\mathrm{d}\mu}{\mathrm{d}t}f$, por lo que

$$\frac{\mathrm{d}\mu}{\mathrm{d}t} = \frac{1}{\tau}\mu, \tag{II.11}$$

es decir,

$$\mu(t) = e^{\frac{t}{\tau}}. \tag{II.12}$$

De esta forma, la ecuación (II.10) se reescribe como

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(e^{\frac{t}{\tau}}f\right) = \frac{e^{\frac{t}{\tau}}}{\tau}g. \tag{II.13}$$

Integrando (II.13) respecto del tiempo de t_0 a $t_0 + \delta_t$ el miembro izquierdo queda como

$$\int_{t_0}^{t_0+\delta_t} \frac{\mathrm{d}}{\mathrm{d}t}\left(e^{\frac{t}{\tau}}f\right)\ \mathrm{d}t = \left(e^{\frac{t}{\tau}}f\right)\Big|_{t_0}^{t_0+\delta_t} = e^{\frac{t_0+\delta_t}{\tau}}f\Big|_{t_0+\delta_t} - e^{\frac{t_0}{\tau}}f\Big|_{t_0} = e^{\frac{t_0}{\tau}}\left(e^{\frac{\delta_t}{\tau}}f\Big|_{t_0+\delta_t} - f\Big|_{t_0}\right) \tag{II.14}$$

Por lo tanto, se tiene

$$e^{\frac{t_0}{\tau}}\left(e^{\frac{\delta_t}{\tau}}f\Big|_{t_0+\delta_t} - f\Big|_{t_0}\right) = \frac{1}{\tau}\int_{t_0}^{t_0+\delta_t}\left(e^{\frac{t}{\tau}}g(t)\right)\ \mathrm{d}t \tag{II.15}$$

o

$$e^{\frac{\delta_t}{\tau}}f\Big|_{t_0+\delta_t} - f\Big|_{t_0} = \frac{1}{\tau}\int_{t_0}^{t_0+\delta_t}\left(e^{\frac{t-t_0}{\tau}}g(t)\right)\ \mathrm{d}t. \tag{II.16}$$

Haciendo el cambio de variable $t' = t - t_0$ en la integral del lado derecho

$$e^{\frac{\delta_t}{\tau}}f(t_0+\delta_t) - f(t_0) = \frac{1}{\tau}\int_0^{\delta_t}\left(e^{\frac{t'}{\tau}}g(t_0+t')\right)\ \mathrm{d}t'. \tag{II.17}$$

Despejando ahora $f(t_0 + \delta_t)$

$$f(t_0+\delta_t) = \frac{1}{\tau}e^{-\frac{\delta_t}{\tau}}\int_0^{\delta_t}\left(e^{\frac{t'}{\tau}}g(t_0+t')\right)\ \mathrm{d}t' + e^{-\frac{\delta_t}{\tau}}f(t_0). \tag{II.18}$$

Como t_0 es cualquier tiempo, se descarta el subíndice. También $f(t+\delta_t) = f(\vec{r}+\vec{\xi}\delta_t, \vec{\xi}, t+\delta_t)$ y $g(t+t') = g(\vec{r}+\vec{\xi}t', \vec{\xi}, t+t')$.

Suponiendo que δ_t es suficientemente pequeño y que, localmente, g es suficientemente suave, se hace la siguiente aproximación[5]

$$g(\vec{r}+\vec{\xi}t', \vec{\xi}, t+t') = \left(1-\frac{t'}{\delta_t}\right)g(\vec{r},\vec{\xi},t) + \frac{t'}{\delta_t}g(\vec{r}+\vec{\xi}\delta_t, \vec{\xi}, t+\delta_t) + \mathcal{O}(\delta_t^2), \qquad 0 \le t' \le \delta_t. \tag{II.19}$$

Se desprecian los término de orden $\mathcal{O}(\delta_t^2)$. Usando esta aproximación en (II.18), se tiene

$$\begin{aligned} f(\vec{r}+\vec{\xi}\delta_t, \vec{\xi}, t+\delta_t) - f(\vec{r},\vec{\xi},t) = &\left(e^{-\frac{\delta_t}{\tau}} - 1\right)\Big[f(\vec{r},\vec{\xi},t) - g(\vec{r},\vec{\xi},t)\Big] \\ &+ \left(1 + \frac{\tau}{\delta_t}(e^{-\frac{\delta_t}{\tau}} - 1)\right)\Big(g(\vec{r}+\vec{\xi}\delta_t, \vec{\xi}, t+\delta_t) - g(\vec{r},\vec{\xi},t)\Big) \end{aligned} \tag{II.20}$$

[5]Esta aproximación se obtiene de utilizar el teorema de Taylor para el tiempo $t + t'$ alrededor de t, y cambiando la primer derivada por su aproximación en diferencias finitas.

Expandiendo $e^{-\frac{\delta_t}{\tau}}$ en su serie de Taylor, y después descartando los términos de orden $\mathcal{O}(\delta_t^2)$ o más pequeños, se llega a

$$f(\vec{r}+\vec{\xi}\delta_t,\vec{\xi},t+\delta_t)-f(\vec{r},\vec{\xi},t)=-\frac{1}{\tau}\left[f(\vec{r},\vec{\xi},t)-g(\vec{r},\vec{\xi},t)\right] \tag{II.21}$$

donde $\tau \equiv \tau/\delta_t$ es el tiempo de relajación adimensional (y su unidad es δ_t). Por lo tanto, la ecuación (II.21) tiene precisión a primer orden en δ_t. Esta es la ecuación de evolución de la función de distribución f con el tiempo discretizado.

Aunque g está escrita como una función explícita del tiempo, su dependencia con el tiempo solamente está en las variables hidrodinámicas ρ, $\vec{u}$ y T, $(g(\vec{r},\vec{\xi},t) = g(\vec{r},\vec{\xi};\rho,\vec{u},T))$. Es decir, para construir la función de distribución de equilibrio, primero es necesario encontrar ρ, $\vec{u}$ y T.

Para evaluar numéricamente los momentos hidrodinámicos dados por las ecuaciones (II.5), se necesita discretizar el espacio $\vec{\xi}$ de forma apropiada. Para lograr esto, se usa el análisis multi-escala de Chapman-Enskog (véase Kruger [18]), donde se encuentra que los momentos hidrodinámicos se pueden calcular con la función de equilibrio:

$$\int h(\vec{\xi})f(\vec{r},\vec{\xi},t)\ \mathrm{d}\vec{\xi}=\int h(\vec{\xi})g(\vec{r},\vec{\xi},t)\ \mathrm{d}\vec{\xi}, \tag{II.22}$$

donde $h(\vec{\xi}) = A + \vec{B}\cdot\vec{\xi} + C\vec{\xi}\cdot\vec{\xi}$, es una combinación lineal de cantidades conservadas, y A, C son dos constantes arbitrarias y B es un vector constante arbitrario.

Por la ecuación (II.22), los momentos de la distribución (II.5) se encuentran con

$$\rho=\int f\,\mathrm{d}\vec{\xi}=\int g\,\mathrm{d}\vec{\xi}, \tag{II.23a}$$

$$\rho\vec{u}=\int \vec{\xi}f\,\mathrm{d}\vec{\xi}=\int \vec{\xi}g\,\mathrm{d}\vec{\xi}, \tag{II.23b}$$

$$\rho\varepsilon=\frac{1}{2}\int(\vec{\xi}-\vec{u})^2 f\,\mathrm{d}\vec{\xi}=\frac{1}{2}\int(\vec{\xi}-\vec{u})^2 g\,\mathrm{d}\vec{\xi}. \tag{II.23c}$$

Con una discretización apropiada, la integración en el espacio de momentos (con función de peso g), se aproxima con cuadraturas hasta una cierta precisión, esto es

$$\int \psi(\vec{\xi})g(\vec{r},\vec{\xi},t)\,\mathrm{d}\vec{\xi}=\sum_\alpha W_\alpha\psi(\vec{\xi}_\alpha)g(\vec{r},\vec{\xi}_\alpha,t), \tag{II.24}$$

donde $\psi(\vec{\xi})$ es un polinomio de $\vec{\xi}$, W_α es el coeficiente de peso de la cuadratura, y $\vec{\xi}_\alpha$ es el conjunto de velocidades discretas, o también se ve como las abscisas de la cuadratura. De acuerdo con esto, los momentos hidrodinámicos de las ecuaciones (II.23) se encuentran con

$$\rho=\sum_\alpha f_\alpha=\sum_\alpha g_\alpha, \tag{II.25a}$$

$$\rho\vec{u}=\sum_\alpha \vec{\xi}_\alpha f_\alpha=\sum_\alpha \vec{\xi}_\alpha g_\alpha, \tag{II.25b}$$

$$\rho\varepsilon=\frac{1}{2}\sum_\alpha(\vec{\xi}_\alpha-\vec{u})^2 f_\alpha=\frac{1}{2}\sum_\alpha(\vec{\xi}_\alpha-\vec{u})^2 g_\alpha \tag{II.25c}$$

donde

$$f_\alpha \equiv f_\alpha(\vec{r},t) \equiv W_\alpha f(\vec{r},\vec{\xi}_\alpha,t), \tag{II.26a}$$

$$g_\alpha \equiv g_\alpha(\vec{r},t) \equiv W_\alpha g(\vec{r},\vec{\xi}_\alpha,t). \tag{II.26b}$$

Por lo que si se encuentra f_α o g_α con sus respectivas velocidades ξ_α, es posible calcular ρ, $\vec{u}$, T.

Con esto, si la ecuación (II.21) se evalua en $\vec{\xi} = \vec{\xi}_\alpha$ y se multiplica por W_α, se obtiene la ecuación

$$f_\alpha(\vec{r}+\vec{\xi}_\alpha\delta_t,\vec{\xi}_\alpha,t+\delta_t) - f_\alpha(\vec{r},\vec{\xi}_\alpha,t) = -\frac{1}{\tau}[f_\alpha(\vec{r},\vec{\xi}_\alpha,t) - g_\alpha(\vec{r},\vec{\xi}_\alpha,t)]. \tag{II.27}$$

Se habla de esta ecuación más adelante.

La función de equilibrio g_α discretizada es desconocida, pero g es conocida hasta cierto punto, y es posible utilizarla para buscar la aproximación descrita por (II.24), lo que lleva a determinar la función g_α discretizada, a partir de (II.26b). Por otra parte, es necesario encontrar la función de distribución de equilibrio g_α porque se usa en (II.21) para continuar con la discretización en el espacio fase. Sin embargo, es a través de la cuadratura (II.24) que se construyen la estructura de red y la función de distribución de equilibrio de la ecuación de Lattice Boltzmann.

Lo que caracteriza a la ecuación de Lattice Boltzmann son tres componentes: (i) una ecuación de evolución en la forma de la ecuación (II.21) con tiempo discretizado, y un espacio fase en el que el espacio de configuraciones (las coordenadas) es de una estructura de red (por eso el método es *lattice*) y un espacio de momentos (las velocidades), que se reduce a un pequeño conjunto de momentos discretos (velocidades discretas); (ii) restricciones de conservación, en la forma de las ecuaciones (II.25); (iii) una adecuada función de distribución de equilibrio que lleve a las ecuaciones de Navier-Stokes.

Para encontrar la discretización de g, se empieza por aproximar la función a segundo orden en la velocidad $\vec{u}$, conocida como aproximación de bajo número Mach [14]:

$$\begin{aligned} g &= \frac{\rho}{(2\pi RT)^{D/2}} \exp\left(-\frac{(\vec{\xi}-\vec{u})^2}{2RT}\right) = \frac{\rho}{(2\pi RT)^{D/2}} \exp\left(-\vec{\xi}^{\,2}/2RT\right) \exp\left\{(\vec{\xi}\cdot\vec{u})/RT - \vec{u}^{\,2}/2RT\right\} \\ &= \frac{\rho}{(2\pi RT)^{D/2}} \exp\left(-\vec{\xi}^{\,2}/2RT\right) \left\{1 + \frac{(\vec{\xi}\cdot\vec{u})}{RT} + \frac{(\vec{\xi}\cdot\vec{u})^2}{2(RT)^2} - \frac{\vec{u}^{\,2}}{2RT}\right\} + \mathcal{O}(\vec{u}^{\,3}) \end{aligned} \tag{II.28}$$

Truncando la expresión a segundo orden, se tiene

$$f^{(\mathrm{eq})} = \frac{\rho}{(2\pi RT)^{D/2}} \exp\left(-\vec{\xi}^{\,2}/2RT\right) \left\{1 + \frac{(\vec{\xi}\cdot\vec{u})}{RT} + \frac{(\vec{\xi}\cdot\vec{u})^2}{2(RT)^2} - \frac{\vec{u}^{\,2}}{2RT}\right\} \tag{II.29}$$

Hay dos consideraciones a tomar en cuenta para la discretización del espacio fase. Primero, la discretización del espacio de momentos está acoplada con la del espacio de configuraciones, de tal forma que se obtiene una estructura de red. Segundo, la cuadratura debe ser suficientemente precisa, de tal forma que las restricciones de conservación dadas por la ecuación (II.25) se mantienen de forma exacta, y además, para que las simetrías necesarias, requeridas por las ecuaciónes de Navier-Stokes, se mantengan.

Al derivar las ecuaciones de Navier-Stokes a partir de la ecuación de Boltzmann por medio del análisis de Chapman-Enskog, los primeros dos órdenes de aproximación de la función de distribución se consideran (es decir, $f^{(\mathrm{eq})}$ y $f^{(1)}$). Dada la función de distribución de equilibrio (II.29), la cuadratura usada para evaluar los momentos hidrodinámicos debe ser capaz de calcular los siguientes momentos con respecto de $f^{(\mathrm{eq})}$ de forma exacta (ver He y Luo [14]):

$$\rho: \quad 1, \ \ \xi_i, \ \ \xi_i\xi_j, \tag{II.30a}$$
$$\vec{u}: \quad \xi_i, \ \ \xi_i\xi_j, \ \ \xi_i\xi_j\xi_k, \tag{II.30b}$$
$$T: \quad \xi_i\xi_j, \ \ \xi_i\xi_j\xi_k, \ \ \xi_i\xi_j\xi_k\xi_l, \tag{II.30c}$$

donde ξ_i es la i-ésima componente de $\vec{\xi}$ en coordenadas cartesianas. Estos momentos se calculan con la función de peso $\exp\left(-\vec{\xi}^{\,2}/2RT\right)$.

Calcular los momentos hidrodinámicos de $f^{(\mathrm{eq})}$ es equivalente a evaluar la siguiente integral en general (véase He y Luo [14]):

$$I = \int \psi(\vec{\xi}) f^{(\mathrm{eq})} \,\mathrm{d}\vec{\xi} = \frac{\rho}{(2\pi RT)^{D/2}} \int \psi(\vec{\xi}) \exp\left(-\vec{\xi}^{\,2}/2RT\right) \left\{ 1 + \frac{(\vec{\xi}\cdot\vec{u})}{RT} + \frac{(\vec{\xi}\cdot\vec{u})^2}{2(RT)^2} - \frac{\vec{u}^{\,2}}{2RT} \right\} \mathrm{d}\vec{\xi}, \tag{II.31}$$

donde $\psi(\vec{\xi})$ es un polinomio de $\vec{\xi}$. La integral (II.31) tiene la siguiente estructura

$$\int e^{-x^2} \psi(x) \,\mathrm{d}x, \tag{II.32}$$

que puede calcularse numéricamente con una cuadratura de tipo Gaussiana.

Antes de seguir discutiendo la discretización del espacio fase, conviene revisar los tipos de esquemas de velocidades y su notación en la literatura.

Esquemas $DnQm$

Los esquemas de velocidades que hay en la literatura del método de Lattice Boltzmann determinan el tipo de red que se usa. Estas velocidades tienen una magnitud determinada, tal que las partículas pasan de un punto de la red a otro vecino. Existen muchas formas de hacer la discretización en el esquema de velocidades, pero lo usual es que las velocidades consideradas sólo estén dirigidas a los vecinos cercanos. También, puede considerarse una velocidad $\vec{0}$ para considerar las partículas que se quedan estáticas.

Una forma de clasificar qué tipo de red se está usando es el esquema $DnQm$. Dn simboliza que la dimensión que utiliza el problema es n. Qm simboliza que se utilizan m velocidades para las partículas.

Por ejemplo, un esquema de 3 dimensiones, y 15 velocidades se representaría con $D3Q15$. Las 15 velocidades consideradas en este esquema son: una de la partícula en reposo en un nodo particular, 6 de los vecinos de este nodo que se encuentran arriba, abajo, izquierda, derecha, al frente y atrás, y las velocidades en las direcciones de los vecinos cercanos en las direcciones oblicuas que suman 8. Si se piensa al nodo que considera la partícula en reposo como si estuviera en el centro de un cubo, los 6 vecinos quedarían en las caras de dicho cubo y los 8 vecinos en direcciones oblicuas, quedarían en los vértices.

Para dos dimensiones, uno de los esquemas más usados es el $D2Q9$ que considera una partícula en reposo, las velocidades hacia arriba, abajo, izquierda y derecha, y las 4 velocidades oblicuas. Este esquema considera 3 magnitudes de velocidad distintas[6]. Las direcciones de estas velocidades se acoplan perfectamente a una red cuadrada. Este esquema es el que se describe en detalle porque es el que se usa en este trabajo.

[6] Una de esas magnitudes es cero, debida a la velocidad $\vec{0}$, otra es la relativa a los primeros vecinos, y la otra velocidad es la relativa a los segundos vecinos.

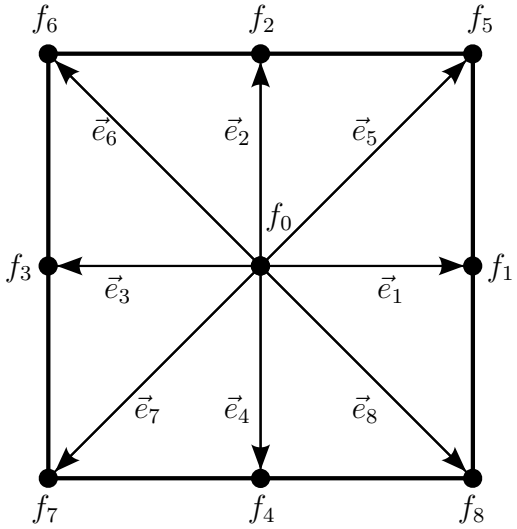


Figura II.2: *Modelo D2Q9: modelo de red bidimensional cuadrada de 9 velocidades.*

Un esquema como el $D2Q6$ o el $D2Q7$, a diferencia del $D2Q9$, se acoplan con una red triangular. Puesto que estos esquemas presentan mayor dificultad para implementar la red en un programa, se deja de lado. El procedimiento que se presenta en la siguiente sección se sigue con ciertas similitudes para los esquemas $D2Q6$ y $D2Q7$, sin embargo, en esos casos es conveniente usar coordenadas polares, véase por ejemplo He y Luo [14].

Discretización de g para el esquema $D2Q9$

Para la discretización de g bajo el esquema $D2Q9$, se usa un sistema coordenado cartesiano. Esto implica que la función polinomial $\psi(\vec{\xi})$ que se ve en la ecuación (II.31) se escribe como

$$\psi_{m,n}(\vec{\xi}) = \xi_x^m \xi_y^n \tag{II.33}$$

donde ξ_x y ξ_y son las componentes x y y de $\vec{\xi}$. Esta función polinomial representa cualquiera de los momentos de las ecuaciones (II.30). Así, la integral de los momentos, definida por la ecuación (II.31), se desarrolla a continuación. Se usa que $\vec{\xi}^2 = \xi_x^2 + \xi_y^2$ y que $D = 2$.

$$
\begin{aligned}
I = \int \psi_{m,n}(\vec{\xi}) f^{(\mathrm{eq})}\,\mathrm{d}\vec{\xi} &= \frac{\rho}{(2\pi RT)^{D/2}} \int \xi_x^m \xi_y^n e^{(-\vec{\xi}^{\,2}/2RT)} \left\{ 1 + \frac{(\vec{\xi}\cdot\vec{u})}{RT} + \frac{(\vec{\xi}\cdot\vec{u})^2}{2(RT)^2} - \frac{\vec{u}^{\,2}}{2RT} \right\} \mathrm{d}\vec{\xi}, \\
&= \frac{\rho}{2\pi RT} \Bigg\{ \int \xi_x^m \xi_y^n e^{(-\vec{\xi}^{\,2}/2RT)} \left(1 - \frac{\vec{u}^{\,2}}{2RT} \right) \mathrm{d}\vec{\xi} + \int \xi_x^m \xi_y^n e^{(-\vec{\xi}^{\,2}/2RT)} \frac{(\xi_x u_x + \xi_y u_y)}{RT}\,\mathrm{d}\vec{\xi} \\
&\qquad + \int \xi_x^m \xi_y^n e^{(-\vec{\xi}^{\,2}/2RT)} \frac{(\xi_x^2 u_x^2 + 2\xi_x u_x \xi_y u_y + \xi_y^2 u_y^2)}{2(RT)^2}\,\mathrm{d}\vec{\xi} \Bigg\} \\
&= \frac{\rho}{2\pi RT} \Bigg\{ \left(1 - \frac{\vec{u}^{\,2}}{2RT} \right) \iint \xi_x^m \xi_y^n e^{(-\xi_x^2/2RT)} e^{(-\xi_y^2/2RT)}\,\mathrm{d}\xi_x\,\mathrm{d}\xi_y \\
&\qquad + \iint \xi_x^m \xi_y^n e^{(-\xi_x^2/2RT)} e^{(-\xi_y^2/2RT)} \frac{(\xi_x u_x + \xi_y u_y)}{RT}\,\mathrm{d}\xi_x\,\mathrm{d}\xi_y \\
&\qquad + \iint \xi_x^m \xi_y^n e^{(-\xi_x^2/2RT)} e^{(-\xi_y^2/2RT)} \frac{(\xi_x^2 u_x^2 + 2\xi_x u_x \xi_y u_y + \xi_y^2 u_y^2)}{2(RT)^2}\,\mathrm{d}\xi_x\,\mathrm{d}\xi_y \Bigg\} \qquad \text{(II.34)} \\
&= \frac{\rho}{2\pi RT} \Bigg\{ \left(1 - \frac{\vec{u}^{\,2}}{2RT} \right) \int \xi_x^m e^{(-\xi_x^2/2RT)}\,\mathrm{d}\xi_x \int \xi_y^n e^{(-\xi_y^2/2RT)}\,\mathrm{d}\xi_y \\
&\qquad + \frac{1}{RT} \Bigg[u_x \int \xi_x^{m+1} e^{(-\xi_x^2/2RT)}\,\mathrm{d}\xi_x \int \xi_y^n e^{(-\xi_y^2/2RT)}\,\mathrm{d}\xi_y \\
&\qquad\qquad + u_y \int \xi_x^m e^{(-\xi_x^2/2RT)}\,\mathrm{d}\xi_x \int \xi_y^{n+1} e^{(-\xi_y^2/2RT)}\,\mathrm{d}\xi_y \Bigg] \\
&\qquad + \frac{1}{2(RT)^2} \Bigg[u_x^2 \int \xi_x^{m+2} e^{(-\xi_x^2/2RT)}\,\mathrm{d}\xi_x \int \xi_y^n e^{(-\xi_y^2/2RT)}\,\mathrm{d}\xi_y \\
&\qquad\qquad + 2u_x u_y \int \xi_x^{m+1} e^{(-\xi_x^2/2RT)}\,\mathrm{d}\xi_x \int \xi_y^{n+1} e^{(-\xi_y^2/2RT)}\,\mathrm{d}\xi_y \\
&\qquad\qquad + u_y^2 \int \xi_x^m e^{(-\xi_x^2/2RT)}\,\mathrm{d}\xi_x \int \xi_y^{n+2} e^{(-\xi_y^2/2RT)}\,\mathrm{d}\xi_y \Bigg] \Bigg\}.
\end{aligned}
$$

Haciendo

$$
I_m = \int_{-\infty}^{+\infty} e^{-\zeta^2} \zeta^m\,\mathrm{d}\zeta, \qquad \text{con} \qquad \zeta = \xi_\gamma / \sqrt{2RT} \qquad \text{(II.35)}
$$

la expresión se simplifica:

$$
\begin{aligned}
I = \frac{\rho}{2\pi RT}\Bigg\{ & \left(1-\frac{\vec{u}^{\,2}}{2RT}\right)\left(\sqrt{2RT}\right)^{(m+1+n+1)} I_m I_n \\
& + \frac{1}{RT}\left[u_x\left(\sqrt{2RT}\right)^{(m+2+n+1)} I_{m+1}I_n + u_y\left(\sqrt{2RT}\right)^{(m+1+n+2)} I_m I_{n+1}\right] \\
& + \frac{1}{2(RT)^2}\Bigg[u_x^2\left(\sqrt{2RT}\right)^{(m+3+n+1)} I_{m+2}I_n + 2u_xu_y\left(\sqrt{2RT}\right)^{(m+2+n+2)} I_{m+1}I_{n+1} \\
& \qquad + u_y^2\left(\sqrt{2RT}\right)^{(m+1+n+3)} I_m I_{n+2}\Bigg]\Bigg\} \\
= \frac{\rho}{\pi}\left(\sqrt{2RT}\right)^{(m+n)}\Bigg\{ & \left(1-\frac{\vec{u}^{\,2}}{2RT}\right) I_m I_n + \frac{\sqrt{2RT}}{RT}(u_x I_{m+1}I_n + u_y I_m I_{n+1}) \\
& + \frac{2RT}{2(RT)^2}(u_x^2 I_{m+2}I_n + 2u_xu_y I_{m+1}I_{n+1} + u_y^2 I_m I_{n+2})\Bigg\} \\
= \frac{\rho}{\pi}\left(\sqrt{2RT}\right)^{(m+n)}\Bigg\{ & \left(1-\frac{\vec{u}^{\,2}}{2RT}\right) I_m I_n + \frac{2(u_x I_{m+1}I_n + u_y I_m I_{n+1})}{\sqrt{2RT}} \\
& + \frac{(u_x^2 I_{m+2}I_n + 2u_xu_y I_{m+1}I_{n+1} + u_y^2 I_m I_{n+2})}{RT}\Bigg\}.
\end{aligned}
\tag{II.36}
$$

Las integrales como la de la ecuación (II.35) pueden evaluarse con ayuda de la cuadratura de Gauss-Hermite[7]:

$$
I_m = \sum_{i=1}^{3} \omega_i \zeta_i^m \tag{II.37}
$$

Las raíces de la cuadratura son:

$$
\zeta_1 = -\sqrt{3/2}, \qquad \zeta_2 = 0, \qquad \zeta_3 = \sqrt{3/2}, \tag{II.38}
$$

y los correspondientes coeficientes de peso son:

$$
\omega_1 = \sqrt{\pi}/6, \qquad \omega_2 = 2\sqrt{\pi}/3, \qquad \omega_3 = \sqrt{\pi}/6, \tag{II.39}
$$

Ahora, para el siguiente desarrollo, se usa la fórmula (II.37) con la convención de suma de índice repetido de Einstein (ver apéndice B), es decir,

$$
I_m = \omega_i \zeta_i^m. \tag{II.40}
$$

El objetivo es encontrar la expresión para las funciones discretizadas g_α, por lo que se busca una expresión similar a (II.24).

Usando (II.40) en (II.36), se tiene

[7] En general, la integral $\int_{-\infty}^{\infty} e^{-x^2} f(x)\,\mathrm{d}x$ puede aproximarse con la fórmula $\sum_{i=1}^{n} \omega_i f(x_i)$, donde n es el número de puntos de muestra usados. x_i son las raíces del polinomio de Hermite $H_n(x)$ y los pesos asociados ω_i están dados por $\omega_i = \frac{2^{n-1} n! \sqrt{\pi}}{n^2 [H_{n-1}(x_i)]^2}$. Estas raíces pueden consultarse rápidamente en `https://www.wolframalpha.com`, buscando "HermiteH_n(x)", sustituyendo n por el número de puntos de muestra.

$$
\begin{aligned}
I = \frac{\rho}{\pi}\left(\sqrt{2RT}\right)^{(m+n)} \Bigg\{ & \left(1 - \frac{\vec{u}^{\,2}}{2RT}\right)\omega_i\zeta_i^m\omega_j\zeta_j^n + \frac{2(u_x\omega_i\zeta_i^{m+1}\omega_j\zeta_j^n + u_y\omega_i\zeta_i^m\omega_j\zeta_j^{n+1})}{\sqrt{2RT}} \\
& + \frac{u_x^2\omega_i\zeta_i^{m+2}\omega_j\zeta_j^n + 2u_xu_y\omega_i\zeta_i^{m+1}\omega_j\zeta_j^{n+1} + u_y^2\omega_i\zeta_i^m\omega_j\zeta_j^{n+2}}{RT}\Bigg\} \\
= \frac{\rho}{\pi}\omega_i\omega_j\Bigg\{ & \left(1 - \frac{\vec{u}^{\,2}}{2RT}\right)\xi_i^m\xi_j^n + \frac{2}{(\sqrt{2RT})^2}(u_x\xi_i^{m+1}\xi_j^n + u_y\xi_i^m\xi_j^{n+1}) \\
& + \frac{u_x^2\xi_i^{m+2}\xi_j^n + 2u_xu_y\xi_i^{m+1}\xi_j^{n+1} + u_y^2\xi_i^m\xi_j^{n+2}}{(\sqrt{2RT})^2RT}\Bigg\}.
\end{aligned}
\tag{II.41}
$$

En la última igualdad se usó que $\zeta_\gamma = \xi_\gamma/\sqrt{2RT}$. Tomando $\vec{\xi}_{i,j} = (\xi_i, \xi_j)$, se tiene que $\psi(\vec{\xi}_{i,j}) = \psi_{m,n}(\vec{\xi}_{i,j}) = \xi_i^m\xi_j^n$. Usando esto y escribiendo los signos de suma:

$$
I = \sum_{i,j=1}^{3}\frac{\rho}{\pi}\omega_i\omega_j\psi(\vec{\xi}_{i,j})\Bigg\{\left(1 - \frac{\vec{u}^{\,2}}{2RT}\right) + \frac{(u_x\xi_i + u_y\xi_j)}{RT} + \frac{u_x^2\xi_i^2 + 2u_xu_y\xi_i\xi_j + u_y^2\xi_j^2}{2(RT)^2}\Bigg\} \tag{II.42}
$$

que se reescriben como

$$
I = \sum_{i,j=1}^{3}\frac{\rho}{\pi}\omega_i\omega_j\psi(\vec{\xi}_{i,j})\Bigg\{1 + \frac{(\vec{\xi}_{i,j}\cdot\vec{u})}{RT} + \frac{(\vec{\xi}_{i,j}\cdot\vec{u})^2}{2(RT)^2} - \frac{\vec{u}^{\,2}}{2RT}\Bigg\}. \tag{II.43}
$$

Las diferentes combinaciones entre los valores de i y j dan valores específicos para el producto $\omega_i\omega_j$. También, como $\vec{\xi}_{i,j} = (\xi_i, \xi_j) = \sqrt{2RT}(\zeta_i, \zeta_j)$, las combinaciones entre los valores de i y j dan distintos vectores. Se define la siguiente notación

$$
\vec{e}_\alpha = \begin{cases} (0,0), & & \alpha = 0 \\ (\cos\theta_\alpha, \operatorname{sen}\theta_\alpha)c, & \theta_\alpha = (\alpha-1)\pi/2, & \alpha = 1,2,3,4 \\ \sqrt{2}(\cos\theta_\alpha, \operatorname{sen}\theta_\alpha)c, & \theta_\alpha = (\alpha-5)\pi/2 + \pi/4, & \alpha = 5,6,7,8 \end{cases} \tag{II.44}
$$

y

$$
w_\alpha = \frac{\omega_i\omega_j}{\pi} = \begin{cases} 4/9, & i = j = 2, & \alpha = 0 \\ 1/9, & i = 1, j = 2,\ldots, & \alpha = 1,2,3,4 \\ 1/36, & i = j = 1,\ldots, & \alpha = 5,6,7,8, \end{cases} \tag{II.45}
$$

donde $c = ||\sqrt{2RT}\zeta_1|| = ||\sqrt{2RT}\zeta_3|| = \sqrt{3RT}$. c es la "velocidad de la luz" en el sistema, y usualmente se hace unitaria. También

$$
c = \frac{\delta_x}{\delta_t}. \tag{II.46}
$$

Otra velocidad que se utiliza es la *velocidad del sonido* del sistema c_s. Para esta configuración está definida por medio de $RT = c_s^2 = c^2/3$.

Usando (II.44) y (II.45) se define

$$
f_\alpha^{(\mathrm{eq})} = \rho w_\alpha\Bigg\{1 + \frac{3(\vec{e}_\alpha\cdot\vec{u})}{c^2} + \frac{9(\vec{e}_\alpha\cdot\vec{u})^2}{2c^4} - \frac{3\vec{u}^{\,2}}{2c^2}\Bigg\}. \tag{II.47}
$$

Con esto, la ecuación (II.43) se reescribe de la siguiente forma:

$$I = \sum_{\alpha=0}^{8} \rho\psi(\vec{\xi}_\alpha) f_\alpha^{(\mathrm{eq})}. \tag{II.48}$$

Comparando con el lado derecho de (II.24):

$$\sum_\alpha W_\alpha \psi(\vec{\xi}_\alpha) g(\vec{r}, \vec{\xi}, t) = \sum_\alpha \rho\psi(\vec{\xi}_\alpha) f_\alpha^{(\mathrm{eq})}, \tag{II.49}$$

y tomando $g(\vec{r}, \vec{\xi}_\alpha, t) \approx f^{(\mathrm{eq})}(\vec{\xi}_\alpha)$, y observando que $f^{(\mathrm{eq})}(\vec{\xi}_\alpha) = \frac{\rho}{(2\pi RT)^{D/2}} \exp\left(-\vec{\xi}_\alpha{}^2/2RT\right) f_\alpha^{(\mathrm{eq})}$, (ver (II.29)),

$$\sum_\alpha W_\alpha \psi(\vec{\xi}_\alpha) \frac{\rho}{(2\pi RT)^{D/2}} \exp\left(-\vec{\xi}_\alpha{}^2/2RT\right) f_\alpha^{(\mathrm{eq})} = \sum_\alpha \rho\psi(\vec{\xi}_\alpha) f_\alpha^{(\mathrm{eq})}, \tag{II.50}$$

se reconoce a W_α:

$$W_\alpha = (2\pi RT)^{D/2} \exp\left(\vec{\xi}_\alpha{}^2/2RT\right). \tag{II.51}$$

Con esto, se observa que la obtención de las ecuaciones (II.25) es directa.

II.2.3. Recuperación de variables físicas

Las variables hidrodinámicas se recuperan con ayuda de las fórmulas:

$$\rho = \sum_\alpha f_\alpha, \tag{II.52a}$$

$$\rho\vec{u} = \sum_\alpha \vec{\xi}_\alpha f_\alpha, \tag{II.52b}$$

$$\rho\varepsilon = \frac{1}{2} \sum_\alpha (\vec{\xi}_\alpha - \vec{u})^2 f_\alpha. \tag{II.52c}$$

Cada una de estas variables se calculan en los nodos de la red, siempre y cuando f_α sea conocida para todas las posibles velocidades. En cuanto a las velocidades $\vec{\xi}_\alpha$, estas se toman en la forma de las ecuaciones (II.44), donde es fácil controlar "la velocidad de la luz" c del sistema.

Así, tenemos

$$\rho = \sum_\alpha f_\alpha, \tag{II.53a}$$

$$\rho\vec{u} = \sum_\alpha \vec{e}_\alpha f_\alpha, \tag{II.53b}$$

$$\rho\varepsilon = \frac{1}{2} \sum_\alpha (\vec{e}_\alpha - \vec{u})^2 f_\alpha. \tag{II.53c}$$

Sin embargo, con la construcción hasta el momento, estas variables no se pueden calcular directamente para nodos cercanos a las fronteras, porque se desconoce alguno o algunos de los valores de f_α. Para puntos que no forman parte de la red, se hace alguna interpolación o extrapolación de su valor.

Para ayudar a comprender un poco más el método, en la siguiente sección se revisa el funcionamiento del algoritmo en regiones alejadas de la frontera.

II.3. Una revisión del algoritmo

Para obtener la ecuación con la que trabaja el algoritmo, en la ecuación (II.27) se sustituye la función de equilibrio g_α por la forma truncada $f_\alpha^{(\mathrm{eq})}$:

$$f_\alpha(\vec{r}+\vec{\xi}_\alpha\delta_t,\vec{\xi}_\alpha,t+\delta_t)-f_\alpha(\vec{r},\vec{\xi}_\alpha,t)=-\frac{1}{\tau}\left[f_\alpha(\vec{r},\vec{\xi}_\alpha,t)-f_\alpha^{(\mathrm{eq})}(\vec{r},\vec{\xi}_\alpha,t)\right]. \tag{II.54}$$

Usando la forma alternativa para la velocidad microscópica ξ_α, dada por (II.44), esta última ecuación toma la forma

$$f_\alpha(\vec{r}+\vec{e}_\alpha\delta_t,\vec{e}_\alpha,t+\delta_t)=f_\alpha(\vec{r},\vec{e}_\alpha,t)-\frac{1}{\tau}[f_\alpha(\vec{r},\vec{e}_\alpha,t)-f_\alpha^{(\mathrm{eq})}(\vec{r},\vec{e}_\alpha,t)]. \tag{II.55}$$

La ecuación (II.55) se actualiza en dos pasos en el algoritmo:

$$\text{Paso de colisión:}\qquad \widetilde{f}_\alpha(\vec{r},\vec{e}_\alpha,t)=f_\alpha(\vec{r},\vec{e}_\alpha,t)-\frac{1}{\tau}[f_\alpha(\vec{r},\vec{e}_\alpha,t)-f_\alpha^{(\mathrm{eq})}(\vec{r},\vec{e}_\alpha,t)], \tag{II.56a}$$

$$\text{Paso de flujo:}\qquad f_\alpha(\vec{r}+\vec{e}_\alpha\delta_t,\vec{e}_\alpha,t+\delta_t)=\widetilde{f}_\alpha(\vec{r},\vec{e}_\alpha,t), \tag{II.56b}$$

donde f_α y $\widetilde{f}_\alpha$ denotan el estado previo a la colisión y el estado posterior a la colisión de la función de distribución, respectivamente. El paso de colisión es local[8]. El paso de flujo no es local y corresponde a la propagación de partículas después de la colisión de nodos vecinos.

Que el paso de colisión sea local implica que no se necesita información de los nodos vecinos, por lo que es un candidato perfecto a paralelizar. El paso de flujo también es paralelizable, pero requiere que se conozca la función de distribución después de la colisión en los nodos vecinos. Es decir, es necesario esperar a que los cálculos del paso de colisión se realicen en todos los nodos vecinos para calcular el paso de flujo. Usualmente esto se traduce en la necesidad de partir en dos funciones separadas estos pasos.

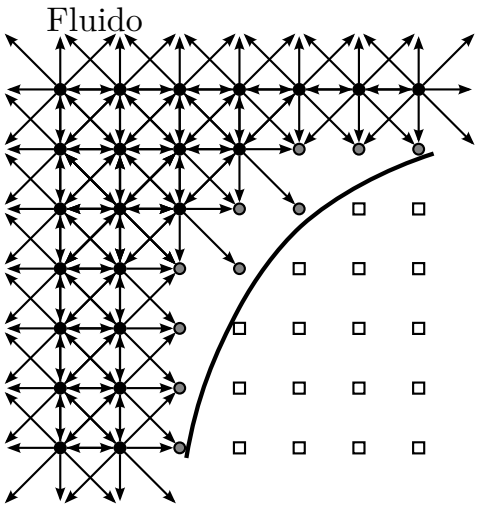


Figura II.3: *Paso de flujo: Las poblaciones f_α migran de un punto a otro en la red. Sin embargo, para los nodos cercanos a la frontera (en gris), se desconoce al menos una de estas poblaciones. El paso estándar de flujo trabaja sin problemas con los nodos fluidos de bulto (en negro).*

[8]Es decir, para calcularlo se utiliza únicamente la información en el mismo punto, al contrario del paso de flujo que no es local, porque necesita información de puntos vecinos.

En lo que sigue, se entiende por poblaciones a las funciones f_α que tiene cada nodo.

En el paso de flujo (ver figura II.3) se observa que el cálculo hecho en el paso de colisión, para un nodo específico, se asigna a $f_\alpha(\vec{r}+\vec{e}_\alpha\delta_t, \vec{e}_\alpha, t+\delta_t)$. Entonces, las funciones f_α en los nodos vecinos se modifican. Un nodo modifica una y sólo una función f_α de cada nodo vecino, y de sí misma. f_1 se modifica en el nodo a la derecha; f_2 en el nodo arriba; f_3 en el nodo a la izquierda; f_4 en el nodo abajo; f_5 en el nodo arriba a la derecha; f_6 en el nodo arriba a la izquierda; f_7 en el nodo abajo a la izquierda; f_8 en el nodo abajo a la derecha; f_0 en el mismo nodo. Alternativamente, las funciones f_α de un nodo en particular, se modifican debido a sus vecinos.[9] Esta forma alternativa plantea la ecuación (II.56b) de la siguiente forma

$$f_\alpha(\vec{r}, \vec{e}_\alpha, t+\delta_t) = \widetilde{f}_\alpha(\vec{r}-\vec{e}_\alpha\delta_t, \vec{e}_\alpha, t). \tag{II.57}$$

De esta manera, para nodos alejados de las fronteras, se conoce la evolución de cada función f_α y se usan las ecuaciones (II.52) para encontrar las variables hidrodinámicas (ver figura II.3).

Sin embargo, los nodos que son vecinos a las fronteras presentan un problema en esta serie de argumentos. Entendiendo por enlace a la conexión entre dos nodos, se define como nodo frontera a un nodo que tiene al menos un enlace con un nodo vecino que se intersecta con la frontera. Así, un nodo vecino a la frontera es un nodo frontera.

El problema que presentan los nodos frontera es que se desconoce la función f_α proveniente de los enlaces que se intersectan por la frontera.

La sección siguiente discute cómo manejar las condiciones de frontera del problema, y se muestra cómo hacer el cálculo de las variables hidrodinámicas en esos nodos frontera.

II.4. Condiciones de frontera

La dificultad de las condiciones de frontera para los métodos de Lattice Boltzmann es encontrar una formulación para las funciónes de distribución f_α que no provienen de los nodos del dominio fluido.

Como el método de Lattice Boltzmann sólo se define para puntos equidistantes en mallas cartesianas, las condiciones de frontera primero se introdujeron para paredes rectas. Estas condiciones de frontera se dividen en dos familias, de rebote (*bounce-back*) y las condiciones de frontera mojadas (ver Jahanshaloo et al. [16]). Para las primeras, los nodos de frontera, nodos en los que se aplican las condiciones de frontera, están fuera del dominio fluido, mientras que para las segundas condiciones, están en la frontera pero siguen siendo parte del dominio fluido. Las primeras sólo ayudan a mantener la cerradura del método, en el sentido de que para estos nodos frontera no se aplica el método de Lattice Boltzmann descrito, pero dicen cómo son las funciones f_α que los nodos frontera necesitan para aplicar el método. Las segundas condiciones para los nodos de frontera aplican un paso de colisión antes de un paso de flujo, similar a los nodos en el cuerpo del fluido. Sin embargo, si la frontera no está alineada con las líneas de la malla, una aproximación de la frontera en escalera sólo daría un esquema de primer orden de precisión espacial (ver Ginzbourg y Adler [11]). Para una frontera curva, los nodos de la red pueden dividirse en nodos en el dominio fluido (nodos fluidos) y nodos fuera del dominio fluido. Para hacer una distinción entre los primeros nodos y los segundos, a los segundos se les llama nodos sólidos, representando la

[9] Cuando se implementa esta parte, es necesario incorporar varios arreglos temporales que guarden cómo se modifican las funciones f_α de cada nodo. De otra forma se incurre en el error de modificar una función f_α particular modificada previamente por un nodo vecino.

región fuera del dominio fluido como si fuera un dominio sólido. Se refiere a la frontera misma como una "pared" en analogía a una frontera sin resbalamiento en una pared sólida (ver figura II.4). Hay muchos intentos para formular condiciones que permitan incrementar la precisión de primer orden a segundo. Una descripción de estos puede verse en Verschaeve y Müller [29] y Jahanshaloo et al. [16].

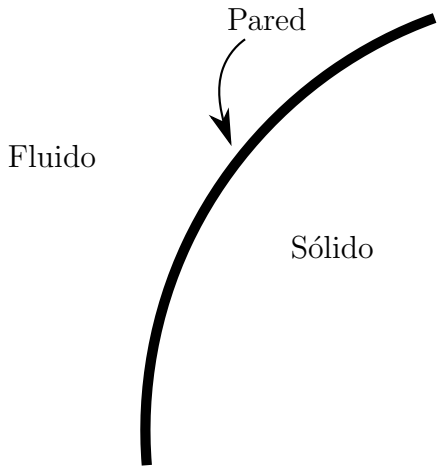


Figura II.4: *Regiones del dominio computacional.*

El tratamiento de frontera que se usa se aplica tanto a la interpolación y extrapolación de la velocidad en el nodo frontera. Como se le aplica un paso de colisión a los nodos frontera, se piensa a este tratamiento como una condición de frontera mojada. Está basado en un formalismo derivado por Lätt [20], [21] que usa una variante previa por Skordos [26]. La estrategia del presente tratamiento de fronteras curvas consiste en interpolar o extrapolar de forma precisa cantidades macroscópicas en los nodos frontera y reconstruir las poblaciones con el formalismo de reconstrucción de Lätt [20], antes de aplicar los usuales pasos de colisión y de flujo. Este tratamiento, desarrollado por Verschaeve y Müller [29] se entiende como una generalización de la condición de frontera de diferencia finita para paredes rectas de Lätt et al. [21]).

El tratamiento que se usa reduce el problema de la precisión y estabilidad a un asunto más técnico de interpolación. Más adelante se describen estos métodos de interpolación.

Se describe en la siguiente sección brevemente en qué consiste la reconstrucción de Lätt.

II.4.1. Reconstrucción de Lätt

Para los nodos frontera una parte de las poblaciones f_α son desconocidas. Las únicas poblaciones conocidas son las que provienen de nodos fluidos. Una alternativa para trabajar con estos nodos consistiría en buscar la manera de relacionar las poblaciones conocidas con las desconocidas en un nodo. Otra alternativa, que es la que se usa en esta tesis, consiste en reconstruir todas las poblaciones a partir de variables macroscópicas. Estas variables macroscópicas son la velocidad del fluido $\vec{u}$, la densidad ρ y el tensor de razón de deformación $\mathcal{S}$.

Cada población puede considerarse constituida por dos partes, una parte en equilibrio $f_\alpha^{(\mathrm{eq})}$ y otra parte fuera de equilibrio $f_\alpha^{(\mathrm{neq})}$. Dada la velocidad $\vec{u}$ y la densidad ρ, se calcula $f_\alpha^{(\mathrm{eq})}$ con la ecuación (II.47). Para calcular la parte fuera de equilibrio se usa

$$f_\alpha^{(\mathrm{neq})} \approx -\frac{\rho\tau w_\alpha}{c_s^2}\mathcal{Q}_\alpha : \mathcal{S}, \tag{II.58}$$

donde el tensor $\mathcal{Q}_\alpha$ se define como sigue

$$\mathcal{Q}_\alpha = \vec{e}_\alpha \vec{e}_\alpha - c_s^2 \mathbf{I}, \tag{II.59}$$

y el tensor de razón de deformación $\mathcal{S}$ es

$$\mathcal{S} = \frac{(\nabla \vec{u} + (\nabla \vec{u})^T)}{2}. \tag{II.60}$$

Finalmente se reconstruyen las poblaciones con

$$f_\alpha = f_\alpha^{(\mathrm{eq})} + f_\alpha^{(\mathrm{neq})}. \tag{II.61}$$

Este procedimiento se llama método de Lattice Boltzmann regularizado ([20]). El método de reconstrucción de Lätt se usa para las condiciones de frontera de no resbalamiento. Para los nodos fluidos de cuerpo, se usa el método estandar descrito en la sección II.3.

Se desarrolla a continuación en notación tensorial (y con la notación de Einstein) el producto diádico $\mathcal{Q}_\alpha : \mathcal{S}$. Para no sobrecargar la notación, al vector $\vec{e}_\alpha$ que usualmente se denotaría por $(e_\alpha)_i$, se le denota simplemente por e_i.

En notación tensorial $\mathcal{S}$ es

$$\mathcal{S}_{ij} = \frac{1}{2}\left(\frac{\partial u_j}{\partial x_i} + \frac{\partial u_i}{\partial x_j}\right). \tag{II.62}$$

El tensor $\mathcal{Q}_\alpha$ es

$$(\mathcal{Q}_\alpha)_{ij} = e_i e_j - c_s^2 \delta_{ij}. \tag{II.63}$$

Así el producto $\mathcal{Q}_\alpha : \mathcal{S}$ es

$$\begin{aligned}
(\mathcal{Q}_\alpha)_{ij}\,\mathcal{S}_{ij} &= \left(e_i e_j - c_s^2 \delta_{ij}\right) \frac{1}{2}\left(\frac{\partial u_j}{\partial x_i} + \frac{\partial u_i}{\partial x_j}\right) \\
&= \frac{1}{2}\left(e_i e_j \frac{\partial u_j}{\partial x_i} - c_s^2 \frac{\partial u_i}{\partial x_i} + e_i e_j \frac{\partial u_i}{\partial x_j} - c_s^2 \frac{\partial u_i}{\partial x_i}\right) \\
&= \frac{1}{2}\left(2 e_i e_j \frac{\partial u_j}{\partial x_i} - 2 c_s^2 \frac{\partial u_i}{\partial x_i}\right) \\
&= e_i e_1 \frac{\partial u_1}{\partial x_i} + e_i e_2 \frac{\partial u_2}{\partial x_i} - c_s^2 \left[\frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2}\right] \\
&= e_1 e_1 \frac{\partial u_1}{\partial x_1} + e_2 e_1 \frac{\partial u_1}{\partial x_2} + e_1 e_2 \frac{\partial u_2}{\partial x_1} + e_2 e_2 \frac{\partial u_2}{\partial x_2} - c_s^2 \left[\frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2}\right]
\end{aligned} \tag{II.64}$$

En el anterior desarrollo se usó que $\delta_{ij}\frac{\partial u_i}{\partial x_j} = \frac{\partial u_i}{\partial x_j}$ por las propiedades de la delta de Kronecker y también que $e_i e_j \frac{\partial u_i}{\partial x_j} = e_j e_i \frac{\partial u_j}{\partial x_i}$ dado que tiene dos parejas de índices mudos y estas pueden intercambiarse libremente.

Se discute en la siguiente sección la categorización que recibe cada nodo y también se describe cómo encontrar $\vec{u}$, ρ y $\mathcal{S}$ para cada nodo frontera, para el método LBGK (véase II.2.1 y II.6).

II.4.2. Condición de no resbalamiento en fronteras curvas para LBGK

Se restringe la discusión al caso bidimensional, concretamente para la configuración $D2Q9$. Como sucede para paredes rectas, el problema principal de las condiciones de frontera para LBGK reside en el hecho que para los nodos cercanos a la frontera, como se ve en la figura II.5, algunas de las poblaciones son desconocidas después del paso de flujo. Sin embargo, para el caso de fronteras curvas, existen diferentes elecciones de cómo se pueden definir los nodos frontera.

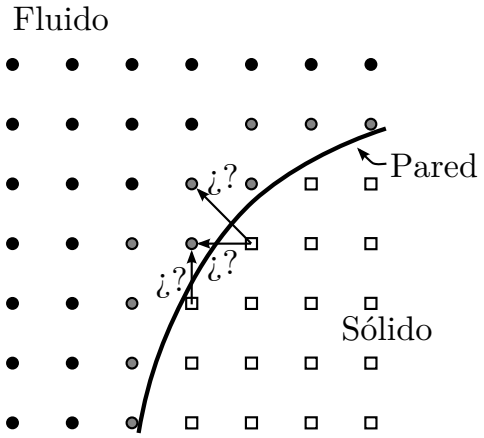


Figura II.5: *Algunos nodos cercanos a la frontera para los cuales se muestra de dónde provienen las poblaciones desconocidas.*

Para continuar la discusión, se redefine la numeración de las poblaciones f_α y de los vectores $\vec{e}_\alpha$ como lo muestra la figura II.6.

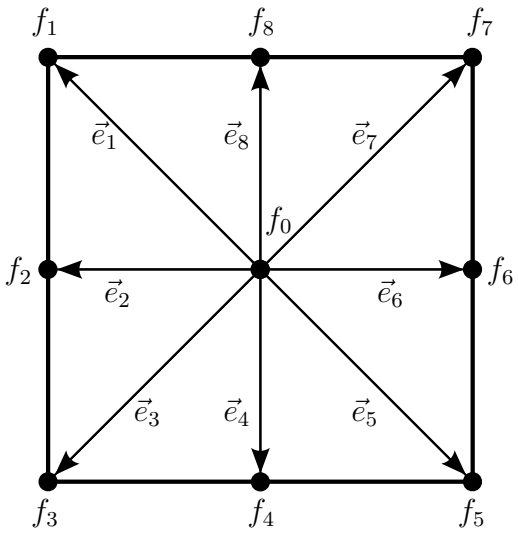


Figura II.6: *Redefinición de direcciones de velocidades para las densidades de probabilidad f_α*

También se redefine $\vec{e}_\alpha$ y w_α de la siguiente forma:

$$\vec{e}_\alpha = \begin{cases} (0,0), & & \alpha = 0 \\ (\cos\theta_\alpha, \operatorname{sen}\theta_\alpha)c, & \theta_\alpha = \pi + \left(\frac{\alpha-2}{2}\right)\frac{\pi}{2}, & \alpha = 2,4,6,8 \\ \sqrt{2}(\cos\theta_\alpha, \operatorname{sen}\theta_\alpha)c, & \theta_\alpha = \frac{3\pi}{4} + \left(\frac{\alpha-1}{2}\right)\frac{\pi}{2}, & \alpha = 1,3,5,7 \end{cases} \tag{II.65}$$

y

$$w_\alpha = \frac{\omega_i \omega_j}{\pi} = \begin{cases} 4/9, & \alpha = 0 \\ 1/9, & \alpha = 2, 4, 6, 8 \\ 1/36, & \alpha = 1, 3, 5, 7. \end{cases} \tag{II.66}$$

Una de las formas para definir los nodos frontera es como los nodos en el dominio fluido que tienen un enlace intersectado por la frontera, como se muestra en la figura II.7a. Entonces hay una distinción entre nodos fluidos de cuerpo (círculos negros) y nodos frontera fluidos (círculos grises) en los que se necesita encontrar las poblaciones desconocidas después del paso de flujo. Los nodos sólidos no toman parte en esta configuración y están desprovistos de cualquier dinámica. Como los nodos frontera están en medio de los nodos fluidos, para los cuales la velocidad $\vec{u}$ es conocida después del paso de flujo, y la pared, en la que se imponen las condiciones de frontera para $\vec{u}$, esta elección para los nodos frontera está entonces asociada con una interpolación de $\vec{u}$ como se ve después. Esta configuración se llama configuración 1. Para el nodo frontera P en cuestión (ver figura II.7a), las poblaciones desconocidas son f_1, f_2 y f_8, pues estas no provienen de ningún nodo fluido. Para un nodo frontera, se clasifica las poblaciones cuyos vectores $\vec{e}_\alpha$ de la red apuntan a nodos fluidos de cuerpo, en un conjunto $\mathcal{F}$ que contiene los índices de dichos vectores. Estos índices se llaman índices fluidos. Por ejemplo, para el nodo P en la figura II.7a se tiene:

$$\mathcal{F} = \{1\}. \tag{II.67}$$

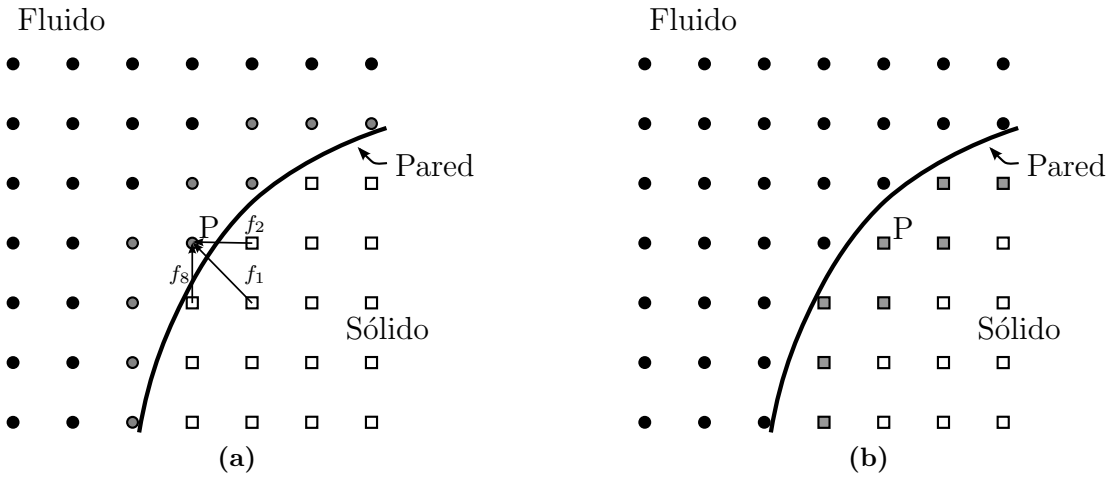


Figura II.7: ***(a)*** *Configuración 1: Los nodos del dominio computacional están tanto en el dominio fluido (círculos) como en el dominio sólido (cuadrados). En este caso se escogen como nodos frontera (en gris), los nodos fluidos que tienen un enlace intersectado por la pared. Círculos negros: Nodos fluidos de cuerpo; Círculos grises: Nodos frontera fluidos; Cuadrados blancos: Nodos sólidos; P es un nodo frontera usado como ejemplo en el texto. Además, se muestra las poblaciones desconocidas del nodo P.* ***(b)*** *Configuración 2: Los nodos frontera (gris) ahora se definen como los nodos sólidos que tienen un enlace intersectado por la pared. Círculos negros: Nodos fluidos; Cuadrados grises: Nodos frontera sólidos; Cuadrados blancos: Nodos sólidos de cuerpo; P es un nodo frontera usado como ejemplo en el texto.*

Los índices α de los vectores $\vec{e}_\alpha$ que apuntan a sí mismo u otros nodos frontera fluidos, definen los elementos del conjunto $\mathcal{B}$ de índices de frontera. De nuevo, para el punto P en la figura II.7a, se tiene

$$\mathcal{B} = \{0, 2, 3, 7, 8\}. \tag{II.68}$$

Finalmente, los vectores que apuntan a nodos sólidos forman el conjunto de índices sólidos $\mathbf{S}$. Para el punto P, $\mathbf{S} = \{4, 5, 6\}$. El conjunto de todos los índices $\mathbf{Q}$ entonces se escribe como:

$$\mathbf{Q} = \mathcal{F} \cup \mathcal{B} \cup \mathbf{S}. \tag{II.69}$$

El superíndice *op* se usa para denotar los conjuntos de índices de los vectores opuestos. Por lo tanto, el conjunto de índices de poblaciones desconocidas (los índices desconocidos) es:

$$\mathbf{S}^{\text{op}} = \{1, 2, 8\}. \tag{II.70}$$

Una segunda forma para definir los nodos frontera es que estos sean los nodos que se encuentran en el dominio sólido y tienen un enlace intersectado por la pared. En este caso se usa una extrapolación para $\vec{u}$. A esta configuración se le llama configuración 2. Para el nodo frontera P en la figura II.7b, los conjuntos definidos anteriormente son:

$$\mathcal{F} = \{1, 2, 8\}, \qquad \mathbf{S} = \{5\}, \qquad \mathcal{B} = \{0, 3, 4, 6, 7\}. \tag{II.71}$$

Otra alternativa más (configuración 3) consiste en definir los nodos frontera como los nodos adyacentes a la pared con la distancia mínima a ella, como se muestra en la figura II.8.

Esto se hace comparando los nodos de la configuración 1 y la configuración 2 entre sí y seleccionando primero aquellos nodos de ambos conjuntos que tienen sus enlaces intersectados por la pared con una distancia más pequeña que la mitad de la distancia entre dos nodos de diferentes conjuntos. Luego los huecos entre estos nodos con distancia mínima se llenan, observando el enlace entre dos nodos alternativos (uno en la región fluida y el otro en la región sólida), y escogiendo el nodo con la menor distancia a la intersección del enlace y la frontera. Esto genera una lista de nodos frontera tal que todos los nodos fluidos están separados de los nodos sólidos. En este caso, para los nodos frontera en el dominio fluido, se interpola $\vec{u}$ (los nodos frontera fluidos, círculos grises), y para los nodos frontera en el dominio sólido (los nodos frontera sólidos, cuadrados grises) se extrapola $\vec{u}$. Los conjuntos de índices para el nodo P en la figura II.8, son

$$\mathcal{F} = \{1, 2, 3, 8\}, \qquad \mathbf{S} = \{5\}, \qquad \mathcal{B} = \{0, 4, 6, 7\}. \tag{II.72}$$

Al contrario que para paredes rectas, los conjuntos $\mathcal{F}$, $\mathbf{S}$ y $\mathcal{B}$ son diferentes para cada nodo frontera. Además, no todos los nodos frontera se sitúan directamente en la pared, por lo que tienen una velocidad distinta de cero, que complica la situación más, puesto que esta velocidad es desconocida a priori.

II.4.3. Cálculo de $\vec{u}$, ρ y $\mathcal{S}$ en los nodos frontera

Se ha probado que las ecuaciones básicas del método regularizado de Lattice Boltzmann (II.47), (II.58) y (II.61) tienen una amplia aplicabilidad en el método de Lattice Boltzmann ([20]). Además de aplicarse en el método regularizado de Lattice Boltzmann, que muestra una estabilidad mayor, las ecuaciones mencionadas se usaron por Lätt para desarrollar un tratamiento de refinamiento de malla ([20]) y un tratamiento de condiciones de frontera para paredes rectas ([21]). La propiedad fundamental de ésta reconstrucción de funciones de distribución de partículas f_α es el hecho que sólo se necesitan cantidades macroscópicas para especificar la forma de f_α. Estas cantidades macroscópicas son la densidad del fluido ρ, la velocidad del fluido $\vec{u}$ y el tensor de razón de deformación $\mathcal{S}$. Si se conocen para cada paso temporal en los nodos cercanos a la frontera (los nodos frontera), la frontera puede representarse de forma muy precisa usándolas en las ecuaciones de reconstrucción (II.47), (II.58) y (II.61). Por lo tanto, la precisión del método depende principalmente de la precisión de las aproximaciones de ρ, $\vec{u}$ y $\mathcal{S}$. Una vez que

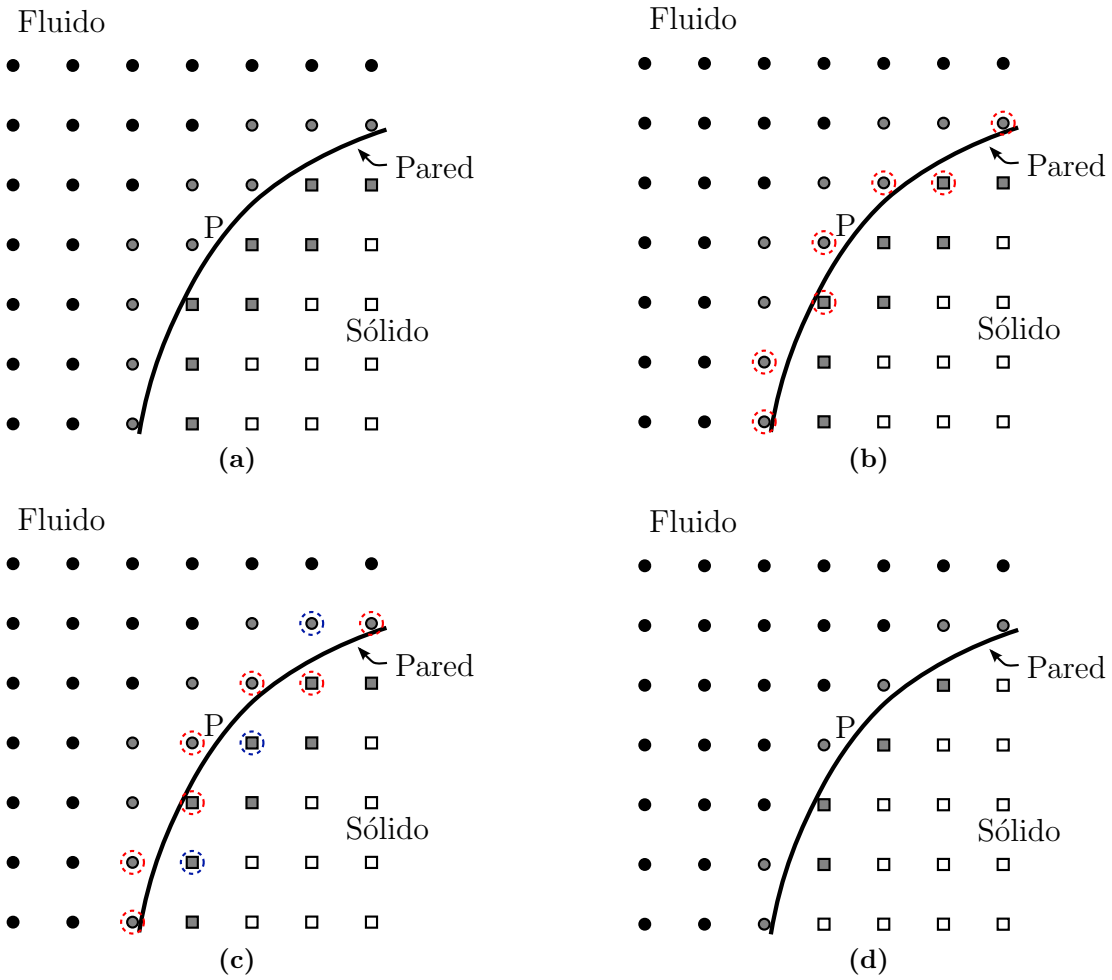


Figura II.8: *Configuración 3: Los nodos frontera (gris) son los nodos sólidos y fluidos con la distancia mínima a la intersección con la pared del enlace intersectado. Círculos negros: Nodos fluidos de cuerpo; Círculos grises: Nodos frontera fluidos; Cuadros grises: Nodos frontera sólidos; Cuadrados blancos: Nodos sólidos de cuerpo; P es el nodo frontera de interés. (a) Se consideran todos los nodos frontera. (b) Se marcan los nodos frontera que esten a una distancia menor o igual a la mitad de la separación entre nodos. (c) De los nodos frontera sin marcar, para algunos casos se puede conectar un nodo frontera sólido con un nodo frontera fluido. Entre estos dos nodos, se marca el más cercano a la frontera. (d) Cualquier nodo frontera no marcado, se transforma en un nodo de bulto del tipo correspondiente. Con esto, los nodos tienen el tipo correcto para esta configuración.*

se encuentran aproximaciones suficientemente precisas, se usan las ecuaciones (II.47), (II.58) y (II.61) para determinar todas las funciones de distribución f_α en los nodos frontera. Después, se aplica el paso usual de colisión (II.56a) seguido del paso de flujo (II.56b). El paso de flujo ahora se ve como muestra la figura II.9. Los nodos sólidos de cuerpo son nodos inactivos que no entran en cálculos.

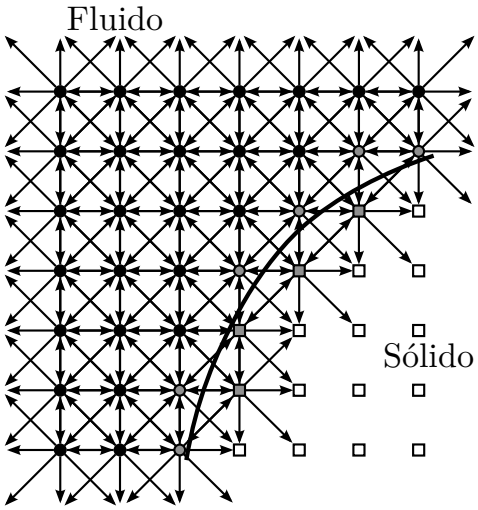


Figura II.9: *Paso de flujo para todos los nodos del domino que no son nodos sólidos de bulto.*

Los esquemas para derivar ρ, $\vec{u}$ y $\mathcal{S}$ se describen a continuación.

Cálculo de $\vec{u}$

Después del paso de flujo, la velocidad $\vec{u}$ se calcula con las ecuaciones (II.53a) y (II.53b). En el caso de la condición de no resbalamiento, la velocidad se especifica en la pared (condición de frontera de Dirichlet). Sin embargo, en los nodos frontera, independientemente de la configuración escogida, la velocidad es desconocida, porque no se tienen poblaciones que provengan de nodos sólidos de cuerpo, y no pueden usarse las ecuaciones (II.53a) y (II.53b) para calcularlas. Por lo tanto, es necesario interpolar o extrapolar la velocidad en el nodo frontera, usando la velocidad en los nodos fluidos de cuerpo cercanos al nodo frontera a lo largo de un enlace, como H_1 y H_2 en la figura II.10, o h_1 y h_2 en la misma figura. Se escoge el índice cuyo enlace asociado forma el menor ángulo con la normal a la pared.

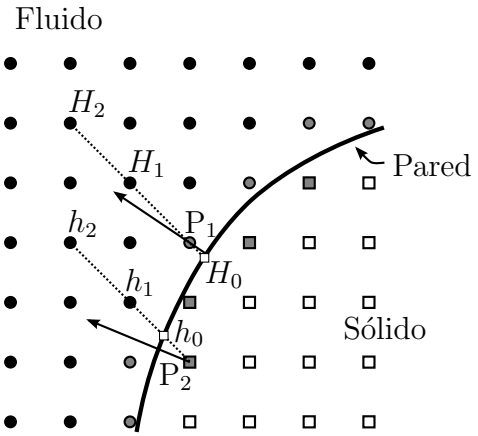


Figura II.10: *Interpolación a lo largo de un enlace entre puntos de la malla. Los puntos H_0, H_1 y H_2 se usan para la interpolación en el nodo frontera P_1. Los puntos h_0, h_1 y h_2 se usan para la extrapolación en el nodo frontera P_2.*

Así, se determina la dirección $\vec{d}$ del enlace de la siguiente forma: $\vec{d} = \frac{\vec{e}_j}{||\vec{e}_j||}$ donde j es el índice para el que $\frac{\vec{e}_i \cdot \vec{n}}{||\vec{e}_i||}$ es máximo, con $i \in \mathcal{F}$, y donde $\vec{n}$ es la normal a la pared que pasa por el nodo frontera. En la

figura II.10 este nodo frontera puede ser P_1 o P_2 que muestran los casos para un nodo frontera fluido y para un nodo frontera sólido. $\vec{d}$ forzosamente existe por la forma en que están definidos los nodos frontera. En el caso raro en que se tenga dos máximos, se escoge el de menor índice.

La interpolación o extrapolación de $\vec{u}$ en P es directa usando los polinomios de Lagrange de interpolación cuadrática (Burden y Faires [3]). Primero se calcula las distancias dP, dH_1 y dH_2 de los nodos P, H_1 y H_2 al origen H_0 (o de los nodos P, h_1 y h_2 al origen h_0), la intersección del enlace con la pared (ver figura II.10). dP es positivo en el caso de interpolación, es decir, P es un nodo frontera fluido (p. ej. P_1), o negativo en el caso de una extrapolación, es decir, P es un nodo frontera sólido (p. ej. P_2). Como los nodos H_1 y H_2 son nodos fluidos de cuerpo (o h_1 y h_2 para el nodo frontera sólido), puede calcularse la velocidad $\vec{u}_1$ y $\vec{u}_2$ respectivamente, en estos nodos. La velocidad $\vec{u}_0$ en el punto H_0 (o h_0) está dada (condición de frontera de Dirichlet). Así, se calcula un estimado de $\vec{u}$ en el nodo frontera P evaluando el polinomio de interpolación en dP:

$$\vec{u}(dP) = \vec{u}_0 l_0(dP) + \vec{u}_1 l_1(dP) + \vec{u}_2 l_2(dP), \tag{II.73}$$

donde $l_i(dP)$, $i = 0, 1, 2$ son los polinomios de interpolación de Lagrange de las distancias $dH_0(= 0)$, dH_1 y dH_2 evaluados en dP:

$$l_0(dP) = \frac{(dP - dH_1)(dP - dH_2)}{dH_1 dH_2}, \tag{II.74}$$

$$l_1(dP) = \frac{dP(dP - dH_2)}{dH_1(dH_1 - dH_2)}, \tag{II.75}$$

$$l_2(dP) = \frac{dP(dP - dH_1)}{dH_2(dH_2 - dH_1)}. \tag{II.76}$$

Cálculo del tensor de razón de deformación $\mathcal{S}$

Para el cálculo del tensor de razón de deformación $\mathcal{S} = (\nabla\vec{u} + (\nabla\vec{u})^T)/2$ se usan diferencias finitas con esténciles a lo largo de las líneas de la red (o malla). Por ejemplo, para calcular $\nabla\vec{u}$ se calculan los términos $a_{ij} = \frac{\partial u_j}{\partial x_i}$, para $i = 1, 2$ y $j = 1, 2$.

Para un nodo frontera fluido P_1 como el que se muestra en la figura II.11, se usan valores en nodos fluidos de cuerpo, de frontera, o intersecciones de las líneas de la malla con la pared, dando diferencias finitas centrales o hacia atrás o adelante, dependiendo de la posición del nodo frontera respecto a la pared frontera.

La forma de determinar esas derivadas parciales con intervalos desiguales se explica a continuación:

Si $\{x_0, x_1, \ldots, x_n\}$ son $(n + 1)$ puntos en un intervalo I, y se tiene una función $f \in \mathcal{C}^{n+1}(I)$, es decir, $n + 1$ veces derivable con todas las derivadas continuas, la función f puede expresarse como

$$f(x) = \sum_{k=0}^{n} f(x_k) L_{n,k}(x) + \frac{(x - x_0)\cdots(x - x_n)}{(n + 1)!} f^{(n+1)}(\xi(x)), \tag{II.77}$$

para algún $\xi(x)$ en I, donde $L_{n,k}$ denota el k-ésimo coeficiente del polinomio de Lagrange para f en $\{x_0, x_1, \ldots, x_n\}$, dado por

$$L_{n,k} = \prod_{\substack{i=0 \\ i \neq k}}^{n} \frac{(x - x_i)}{(x_k - x_i)}. \tag{II.78}$$

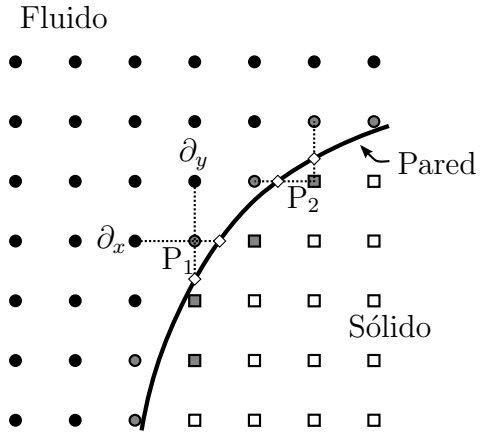


Figura II.11: *Cálculo del tensor de razón de deformación $\mathcal{S}$ en el nodo frontera fluido P_1 y en el nodo frontera sólido P_2. Los puntos en la línea horizontal se usan para calcular $\partial_x \vec{u}$ y los puntos en la línea horizontal para $\partial_y \vec{u}$.*

Derivando (II.77)

$$\begin{aligned} f'(x) =& \sum_{k=0}^{n} f(x_k) L'_{n,k}(x) + D_x \left[\frac{(x-x_0)\cdots(x-x_n)}{(n+1)!} \right] f^{(n+1)}(\xi(x)) \\ &+ \frac{(x-x_0)\cdots(x-x_n)}{(n+1)!} D_x[f^{(n+1)}(\xi(x))]. \end{aligned} \tag{II.79}$$

El último término no es posible calcularlo o estimarlo pues se desconoce la derivada $(n+2)$ de f en el punto $\xi(x)$. Sin embargo, si x es alguno de los puntos x_j, ese término se hace cero porque su coeficiente se hace cero. Si además se evalúa en $x = x_j$, la fórmula (II.79) se vuelve

$$f'(x_j) = \sum_{k=0}^{n} f(x_k) L'_{n,k}(x_j) + \frac{f^{(n+1)}(\xi(x_j))}{(n+1)!} \prod_{\substack{k=0 \\ k\neq j}}^{n} (x_j - x_k), \tag{II.80}$$

que es la fórmula de (n+1) puntos para aproximar $f'(x_j)$.

Para tres puntos x_0, x_1, x_2, esta fórmula es

$$\begin{aligned} f'(x_j) =& f(x_0) \left[\frac{2x_j - x_1 - x_2}{(x_0 - x_1)(x_0 - x_2)} \right] + f(x_1) \left[\frac{2x_j - x_0 - x_2}{(x_1 - x_0)(x_1 - x_2)} \right] \\ &+ f(x_2) \left[\frac{2x_j - x_0 - x_1}{(x_2 - x_0)(x_2 - x_1)} \right] + \frac{1}{6} f^{(3)}(\xi(x_j)) \prod_{\substack{k=0 \\ k\neq j}}^{2} (x_j - x_k), \end{aligned} \tag{II.81}$$

donde $j = 0, 1, 2$, y ξ_j indica que este punto depende de x_j.

Para algunos casos, como cuando la frontera es una pared recta (localmente, pero abarcando varios nodos de la red), se usan sólo nodos de la red. En estas circunstancias, los nodos de la red son equidistantes y la fórmula para calcular la derivada se simplifica. Sea h la separación entre los puntos donde están los nodos. Según el tipo de nodos que rodeen al nodo donde se quiere calcular la derivada se usa alguna de las siguientes expresiones

$$f'(x_0) = \frac{1}{2h}[-3f(x_0) + 4f(x_0 + h) - f(x_0 + 2h)] + \frac{h^2}{3} f^{(3)(\xi_0)} \tag{II.82}$$

$$f'(x_0) = \frac{1}{2h}[f(x_0 + h) - f(x_0 - h)] - \frac{h^2}{6} f^{(3)(\xi_1)} \tag{II.83}$$

$$f'(x_0 + 2h) = \frac{1}{h}\left[\frac{1}{2}f(x_0) - 2f(x_0 + h) + \frac{3}{2}f(x_0 + 2h)\right] + \frac{h^2}{3} f^{(3)(\xi_2)}, \tag{II.84}$$

donde ξ_0, ξ_1 y ξ_2 son puntos intermedios entre x_0 y $x_0 + 2h$. Por supuesto, al quitar el último término en las 4 ecuaciones anteriores, se obtiene la aproximación a segundo orden. Para el caso en que se usa la expresión (II.81), la aproximación es mejor porque las distancias entre puntos son menores, pero el costo es que los cálculos se hacen un poco más pesados.

Cálculo de ρ

Para encontrar una aproximación para ρ se usan las poblaciones conocidas que llegan a un nodo frontera en el paso de flujo. Estas poblaciones conocidas se identifican como poblaciones postflujo. Para fines prácticos, se identifica a todas las poblaciones (de cada nodo) después del paso de flujo, como poblaciones postflujo, y se identifican con el superíndice pf, i.e. f_α^{pf}. A estas poblaciones se les aplica a continuación el paso de reconstrucción que las convierte a las poblaciones que recibe la ecuación (II.56a) encargada de aplicar el paso de colisión. Se recuerda que este paso de reconstrucción únicamente aplica para los nodos frontera. Equivalentemente, se aplica a todos los nodos, pero para los nodos fluidos de cuerpo este paso corresponde a la identidad.

De las ecuaciones (II.47), (II.58), se observa que la densidad puede separarse en (II.61), de tal manera que se tiene lo siguiente:

$$f_\alpha = \rho \mathfrak{g}_\alpha = \rho(\mathfrak{g}_\alpha^{\text{eq}}(\vec{u}) + \mathfrak{g}_\alpha^{\text{neq}}(\mathcal{S})), \tag{II.85}$$

donde $\mathfrak{g}_\alpha^{\text{eq}}$ se define como:

$$\mathfrak{g}_\alpha^{\text{eq}}(\vec{u}) = f_\alpha^{\text{eq}}(1, \vec{u}), \tag{II.86}$$

y $\mathfrak{g}_\alpha^{\text{neq}}$ como

$$\mathfrak{g}_\alpha^{\text{neq}}(\mathcal{S}) = f_\alpha^{\text{neq}}(1, \mathcal{S}). \tag{II.87}$$

Nótese que $\mathfrak{g}_\alpha^{\text{eq}}$ y $\mathfrak{g}_\alpha^{\text{neq}}$ pueden calcularse tan pronto como se conozcan $\vec{u}$ y $\mathcal{S}$ en el nodo frontera. Denotando como $\mathcal{K}$ al conjunto de índices conocidos, a saber, los índices de las poblaciones que se conocen después del paso de flujo. Una posible elección para $\mathcal{K}$ son los conjuntos opuestos de índices fluidos y de frontera:

$$\mathcal{K} = \mathcal{F}^{\text{op}} \cup \mathcal{B}^{\text{op}}, \tag{II.88}$$

pues estas poblaciones no provienen de nodos sólidos. Otra elección es escoger sólo las poblaciones que provienen de nodos fluidos:

$$\mathcal{K} = \mathcal{F}^{\text{op}}. \tag{II.89}$$

En la referencia [30] se muestra que, en la frontera, las poblaciones que vienen de nodos frontera pueden llevar a inestabilidades numéricas. Por lo tanto, se escoge $\mathcal{K} = \mathcal{F}^{\text{op}}$. La estrategia para encontrar una formulación local de ρ es aproximar ρ imponiendo las condiciones siguientes:

$$f_\alpha^{\text{pf}} = \rho \mathfrak{g}_\alpha, \qquad \alpha \in \mathcal{K}, \tag{II.90}$$

donde f_α^{pf} es el valor de la población α después del paso de flujo y antes de sustituirse por $\rho\mathfrak{g}_\alpha$. Si la cardinalidad de $\mathcal{K}$ es mayor que 1, el sistema de ecuaciones, compuesto por (II.90), está sobredeterminado. En este caso, una solución puede encontrarse en términos de mínimos cuadrados:

$$\rho = \frac{\sum_{\alpha\in\mathcal{K}} f_\alpha^{\mathrm{pf}}\mathfrak{g}_\alpha}{\sum_{\beta\in\mathcal{K}}(\mathfrak{g}_\beta)^2}. \tag{II.91}$$

También puede formularse una solución de (II.90) en términos de un promedio pesado con:

$$\rho = \frac{1}{|\mathcal{K}|}\sum_{\alpha\in\mathcal{K}} \frac{f_\alpha^{\mathrm{pf}}}{\mathfrak{g}_\alpha}, \tag{II.92}$$

donde $|\mathcal{K}|$ es la cardinalidad de $\mathcal{K}$. Ambas formas de aproximar ρ, ecuaciones (II.91) y (II.92), comparten la desventaja de no coincidir con la definición de la densidad, (II.53a), si $\mathcal{K} = \mathcal{Q}$, esto es, en el caso de un nodo fluido de cuerpo. Otra aproximación para la densidad se escribe como la suma de todas las poblaciones conocidas y dividido entre la suma correspondiente de $\mathfrak{g}_\alpha$:

$$\rho = \frac{\sum_{\alpha\in\mathcal{K}} f_\alpha^{\mathrm{pf}}}{\sum_{\alpha\in\mathcal{K}}(\mathfrak{g}_\alpha^{\mathrm{eq}} + \mathfrak{g}_\alpha^{\mathrm{neq}})}. \tag{II.93}$$

Para $\mathcal{K} = \mathcal{Q}$, esta aproximación coincide con la definición de densidad, la ecuación (II.53a).

Estas tres aproximaciones para ρ, (II.91), (II.92) y (II.93) no conservan masa, en el sentido que la suma de poblaciones que fluyen de los nodos frontera en los nodos fluidos de cuerpo después de el paso de colisión no igualan necesariamente la suma de poblaciones que fluyen hacia los nodos fluidos de cuerpo después del paso de flujo.

Sin embargo de todo esto, se trabaja la aproximación (II.93), que da los mejores resultados en Verschaeve y Müller [29].

II.4.4. Fronteras móviles

El método considerado para tratar las condiciones de frontera sugiere su posible uso en fronteras con movimiento. Para problemas con fronteras móviles, el fluido tiene la misma velocidad que la pared en movimiento, por lo que, reconocidos los nodos frontera en cada etapa del movimiento de la pared, y conocida la velocidad de las fronteras, en los nodos frontera se pueden reconstruir las poblaciones f_α, y con ello, el dominio tendrá retroalimentación del movimiento que se lleva en las fronteras.

Así que, al contrario de un dominio estático, los nodos del dominio no tienen un único tipo a lo largo del tiempo. Para trabajar dicho problema, es entonces fundamental tener un método que identifique los tipos de los nodos de acuerdo con las reglas descritas anteriormente.

La frontera del dominio se puede mover debido a varias causas. Por ejemplo, una posibilidad es que una de las paredes del dominio sea un émbolo de una jeringa y entonces se mueva por una fuerza exterior a la influencia del dominio. O posiblemente un pistón de un motor empujado por los gases de combustión. También, las paredes de una vena o arteria con el bombeo de sangre.

Concretamente, hay un conjunto de situaciones en que es el fluido al interior el que mueve la frontera, mediante el intercambio de momento con ella. Ahora, como se discutió en II.2.3, la densidad de momento se puede obtener en cada nodo fluido de bulto, y además, por medio de interpolación y extrapolación, en los nodos frontera. Esto hace posible una forma de encontrar el momento que el fluido le intercambia a la frontera, y por lo tanto, una forma de encontrar cómo se mueve.

Por supuesto, la naturaleza de la frontera determina su propia respuesta, y en unas situaciones disipará una mayor parte de la energía transmitida por el fluido, que en otras.

Por otra parte, la construcción de la frontera con partículas representativas permitiría una interacción más localizada entre la frontera y el medio, sin descartar la interacción que la frontera tiene con ella misma ni la interacción que el mismo fluido tiene con sí mismo.

En un modelo simple, la frontera se considera como pequeñas masas unidas por resortes. Según el modelo particular con este enfoque, puede considerarse que la influencia del medio en una partícula de la frontera será amortiguada por 2 o más resortes.

La influencia del medio en la frontera puede entenderse con un modelo de colisión entre las partículas del medio y las partículas de la frontera. Si la frontera está conformada por pequeñas masas, como se sugiere en el párrafo anterior, entonces la colisión puede darse entre partículas representativas del medio y con las partículas representativas de la frontera. Para terminar de determinar el problema, la dirección de las partículas representativas de la frontera después de la colisión puede ser perpendicular a la normal a la frontera antes de la colisión, y la dirección de las partículas representativas del medio, tal que el ángulo en el que incidieron sea igual al ángulo de salida con la normal, similar a la ley de Snell para la reflexión de rayos. Impuestas estas dos direcciones, las velocidades después de la colisión se pueden encontrar y con ello el intercambio de momento con la pared.

Así, conocidas las fuerzas que una partícula siente (el cambio de momento de ésta por unidad de tiempo), con un método como Runge-Kutta (véase Burden y Faires [3]) es posible determinar el movimiento que la partícula seguirá. Esto por supuesto es aplicable a cualquier partícula de la frontera, por lo que se puede estimar el movimiento general de la frontera.

II.4.5. Otros tipos de condiciones de frontera

Una de las condiciones de frontera más usadas es la de rebote o *bounce back*. Esta condición de frontera es útil para imponer una condición de no-resbalamiento en una frontera. Físicamente esto quiere decir que no hay movimiento de flujo en la frontera. La condición de rebote establece que cuando una partícula llega a un nodo en una pared, la partícula se regresa de vuelta al nodo fluido del que provenía. Considerando de nuevo las funciones de distribución en el esquema $D2Q9$ definidas en la figura II.12, esta condición implica las siguientes igualdades:

$$f_1 = f_5 \tag{II.94}$$
$$f_2 = f_6 \tag{II.95}$$
$$f_3 = f_7 \tag{II.96}$$
$$f_4 = f_8. \tag{II.97}$$

Por supuesto, las funciones de distribución paralelas a una pared no sufren este rebote. Esta reflección completa garantiza que tanto la componente tangencial como la componente normal de la velocidad del fluido en la pared se anulen.

Otro tipo de condiciones de frontera son las condiciones periódicas. Éstas normalmente se establecen cuando la frontera queda en el borde del dominio. Si se observa los nodos en el borde de un dominio rectangular, por ejemplo, los nodos en el lado izquierdo y derecho del dominio rectangular, los nodos en el lado izquierdo no tienen nodos más a su izquierda que los provean con las poblaciones f_5, f_6 o f_7 en el paso de flujo; los nodos en el lado derecho no tienen nodos más a su derecha que los provean con las

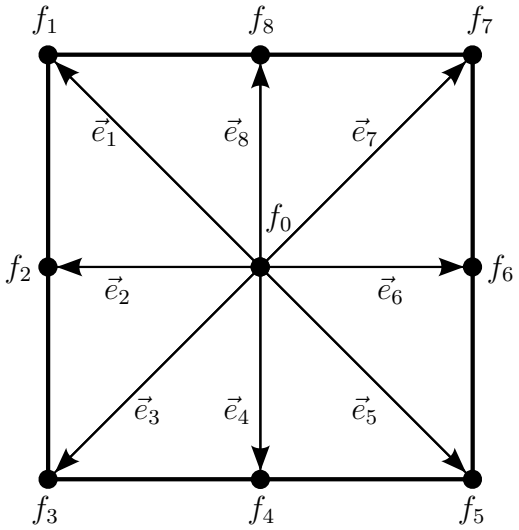


Figura II.12: *Redefinición de direcciones de velocidades para las densidades de probabilidad f_α*

poblaciones f_1, f_2 o f_3. Para proveer estas poblaciones, lo que se hace es conectar los nodos del borde izquierdo con los del derecho, haciendo que las poblaciones faltantes las provea el otro. Esto representa condiciones de frontera periódicas. En este caso se tiene, del lado izquierdo

$$f_5(1, y, t) = f_5(L, y, t) \tag{II.98}$$
$$f_6(1, y, t) = f_6(L, y, t) \tag{II.99}$$
$$f_7(1, y, t) = f_7(L, y, t), \tag{II.100}$$

y del lado derecho

$$f_1(L, y, t) = f_1(1, y, t) \tag{II.101}$$
$$f_2(L, y, t) = f_2(1, y, t) \tag{II.102}$$
$$f_3(L, y, t) = f_3(1, y, t). \tag{II.103}$$

Otra alternativa para las poblaciones de entrada (f_5, f_6 y f_7 en el ejemplo anterior, que son las que se imponen del lado izquierdo) es establecerlas con las poblaciones de equilibrio $f_\alpha^{(\mathrm{eq})}$ dadas por la ecuación (II.47). Para las poblaciones de salida (f_1, f_2 y f_3 en el ejemplo anterior, que son las que se imponen del lado derecho), se puede copiar las poblaciones a la izquierda. Esta no es una manera muy limpia de imponerlas, pero cierra el problema de tal forma que ya no hay poblaciones desconocidas.

II.5. Condiciones iniciales

Al igual que con las condiciones de frontera, las condiciones iniciales de las funciones de densidad de probabilidad f_α, son un tema que no se aborda de manera directa, como por ejemplo en un problema de campos de velocidades donde las velocidades están dadas y se trabaja directamente sobre ellas.

Usualmente se propone inicializar las poblaciones f_α a las poblaciones de equilibrio $f_\alpha^{(\mathrm{eq})}$, con las velocidades y densidades dadas (véase [24] y [28]). Esto parece razonable si uno supone que la perturbación que sufre el fluido no existía antes y empieza a aparecer justo en el momento en que se inicia la simulación.

Hay un gran problema con la inicialización a partir de las poblaciones de equilibrio. Estas implican que inicialmente todos los momentos cinéticos están en equilibrio, los cuáles dependen sólo de los momentos hidrodinámicos. Pero esto no es cierto cuando los flujos de entrada ρ_0 y $\vec{u}_0$ tienen gradientes distintos de cero (véase [24]). También, se muestra en [24], [26], [27], [31] y [4] que es crucial hacer modificaciones a los campos macroscópicos iniciales para asegurar un comportamiento suave de la solución y para evitar errores iniciales, que de otra forma persistirán a través de la simulación.

Por otra parte, uno podría abordar el problema usando la reconstrucción de Lätt. Dadas las velocidades, la densidad, y el tensor de razón de deformación iniciales, las poblaciones iniciales se pueden reconstruir. Pero de aquí también es recuperable la inicialización en la poblaciones de equilibrio si las velocidades alrededor son las mismas, pues el tensor de razón de deformación en estos casos tiene todas sus componentes iguales a cero y, con ello, la recontrucción de poblaciones equivale a establecerlas como las poblaciones de equilibrio. Esto es porque la parte $f_\alpha^{(\mathrm{neq})}$ se anula, al menos al orden de la aproximación dada, aunque también, presumiblemente, en su forma exacta.

II.6. Discusión del método

Una característica que define la versión particular del método de Lattice Boltzmann considerado, es su operador de colisión. Cuando se usa la expresión

$$\Omega_{\mathrm{BGK}}(f) = -\frac{f-g}{\tau}, \tag{II.104}$$

se llama método de Lattice Boltzmann con la aproximación BGK (LBGK) o modelo de tiempo de relajación sencillo (SRT). Recuérdese que este término de colisión particular es un modelo de colisión para gases. Esto limita los problemas que se pueden resolver en gran medida. De hecho, esto da una conexión entre la presión y la densidad por medio de la ecuación de estado de un gas ideal

$$p = c_s^2 \rho, \tag{II.105}$$

con ρ la densidad, p la presión, y c_s la velocidad del sonido definida anteriormente.

También, para lograr la discretización de la ecuación de transporte de Boltzmann en $\vec{\xi}$, se mencionó el uso del análisis multi-escala de Chapman-Enskog. Un resultado del análisis multi-escala de Chapman-Enskog (véase [29] y [18]) es que las cantidades macroscópicas obedecen las ecuaciones de Navier-Stokes para un fluido incompresible, para números pequeños de Mach y Knudsen. De hecho, el esquema las resuelve hasta un error compuesto de las siguientes tres contribuciones:

- El error espacial escala como δx^2, donde δx es el espaciamiento de la malla.
- El error temporal escala como δt^2, donde δt es el paso temporal.
- El error en la compresibilidad escala como Ma^2, donde Ma es el número de Mach.

Como el número de Mach escala como $\delta t/\delta x$ ($c_s = c/\sqrt{3} = \delta x/(\sqrt{3}\delta t)$ y $Ma = \frac{u_0}{c_s}$), el paso temporal necesita ser escalado como $\delta t \propto \delta x^2$ para recobrar las ecuaciones de Navier-Stokes incompresibles a segundo órden en el espacio.

El tiempo de relajación, que fija la razón de aproximación al equilibrio, está relacionado con la viscocidad cinemática[10] por[25]

$$\tau = \frac{6\nu + 1}{2}. \tag{II.106}$$

ν está medida en unidades de la red. Se observa que τ es el valor crítico para asegurar que la viscocidad cinemática sea no negativa. Las inestabilidades numéricas pueden ocurrir para τ cerca de este valor crítico.

II.7. Modificaciones al algoritmo para considerar fronteras móviles

La estructura de un algoritmo más general se muestra en la figura II.13

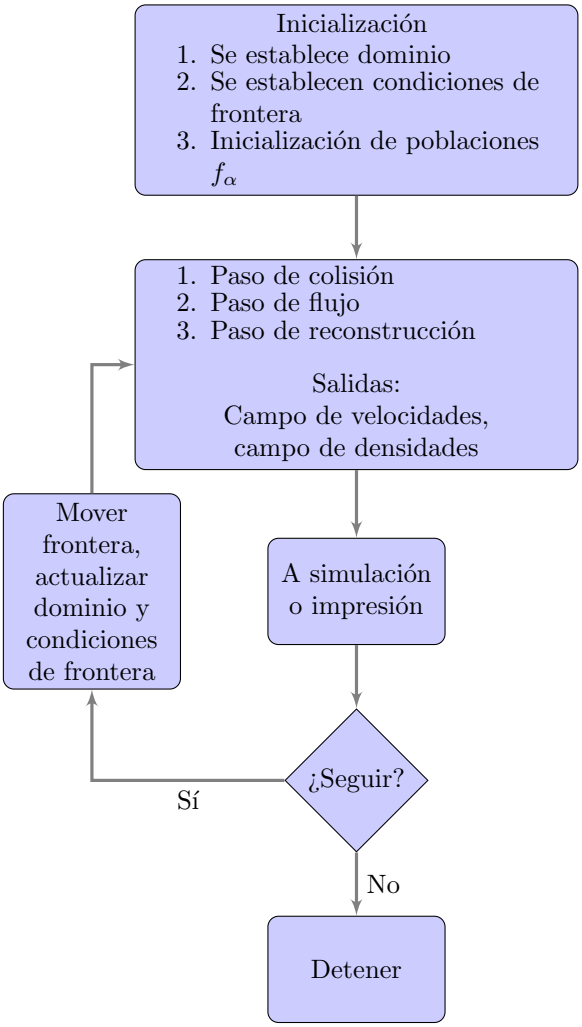


Figura II.13: *Diagrama de flujo del algoritmo.*

[10]La viscocidad cinemática se define como la razón de la viscocidad dinámica μ entre la densidad del fluido ρ. La viscocidad dinámica de un fluido expresa su resistencia a flujos cortantes, donde capas adyacentes se mueven paralelas entre sí con diferentes velocidades.

Quitando el paso que sigue del cuadro condicional "¿Seguir?", el algoritmo es el de un problema con fronteras estáticas. Así, la modificación se encarga de trabajar aisladamente el problema de las fronteras móviles.

Como se observa en la figura II.13, lo que se necesita hacer es mover la frontera, actualizar el dominio y actualizar las condiciones de frontera. Una forma de lograr el movimiento de la frontera, como se comentaba en la subsección II.4.4, es asignar partículas a la frontera y mover estas de acuerdo al momento que reciban del medio fluido. Luego del movimiento de la frontera, las partículas en ella, según su movimiento modifican las condiciones de frontera en sus correspondientes posiciones. Para actualizar el dominio, es necesario identificar los nodos que están dentro de la frontera del dominio. Para esta parte, se desarrollan métodos en el siguiente capítulo acorde a los requerimientos de la configuración 3 para el tratamiento de condiciones de frontera en fronteras curvas, discutidas en la sección II.4.2.

CAPÍTULO III

Implementación

La implementación del método de Lattice Boltzmann se puede separar en dos partes. Una de las partes es la que aplica el método de Lattice Boltzmann estándar. La otra parte genera una estructura adecuada para que el método funcione adecuadamente. Esta segunda parte se encarga de mover las fronteras, categorizar de acuerdo con esto los nodos y modificar condiciones de frontera. La primer parte, en cierto sentido, sólo trabaja con la información que le llega del dominio. Naturalmente, la información procesada se transmite a la segunda parte para mover las fronteras, categorizar de nuevo los nodos y modificar las condiciones de frontera. Así, hay una comunicación entre estas dos partes, pero una parte no se ejecuta hasta que la otra ha terminado de ejecutarse.

Ambas partes se implementan con el esquema $D2Q9$.

Una discusión más amplia del algoritmo se encuentra en II.6.

A continuación se describe la implementación de la parte correspondiente al método de Lattice Boltzmann.

III.1. Implementación del método de Lattice Boltzmann

En las secciones II.3 y II.7 se revisó el algoritmo que sigue el método de Lattice Boltzmann. El algoritmo estándar del método consiste de dos pasos fundamentales. Uno es el paso de colisión y otro es el paso de flujo. Para implementar las condiciones de frontera se incorpora un nuevo paso que reconstruye las poblaciones f_α en los nodos frontera, que son nodos en la proximidad de la frontera. Estos tres pasos trabajan con las poblaciones f_α, $\widetilde{f}_\alpha$ y f_α^{pf}. Sin embargo, en la implementación se usan 9 arreglos para manejar las 9 poblaciones y otros 9 arreglos auxiliares, que se usan en el paso de flujo.

El arreglo k-ésimo fk contiene los valores que f_k tiene para cada nodo del dominio a un tiempo determinado. Un nodo en el dominio se representa por una pareja de índices que indican su posición relativa a otros nodos del dominio, tal como se hace para mallados estructurados (y la red es uno). Esto es relativamente sencillo dado que, de entrada, los dominios donde viven estos nodos son rectangulares. Un nodo no entra en la simulación si este se categoriza como nodo sólido de bulto. La desventaja es, que se requiere una mayor cantidad de espacio en memoria para guardar estos arreglos que consideran nodos que no entran en la simulación. La forma más natural para escoger estos arreglos sería con arreglos bidimensionales. Sin embargo, con miras a hacer del algoritmo lo más adecuado para paralelizar, estos arreglos son unidimensionales, o lineales. El valor que un nodo posee de una población específica, se asocia a una posición específica en este arreglo. La forma de lograr esto es con un polinomio de redireccionamiento. Este polinomio de redireccionamiento toma dos índices (i, j), los mismo índices que identifican al nodo, y a partir de ellos calcula la posición que un valor debe tomar en el arreglo. Si NX es el número de nodos en la dirección x, este polinomio se puede escoger como $PL(i, j) = jNX + i$, si i es el índice que representa la posición horizontal del nodo en el dominio y j la posición vertical. Con un polinomio de redireccionamiento de esta forma, cada nodo tiene asociado un valor entero distinto. La figura III.1 muestra la asociación de cada pareja de índices, en un pequeño dominio cuadrado, a un número entero distinto (en rojo) que se usa como posición el arreglo lineal.

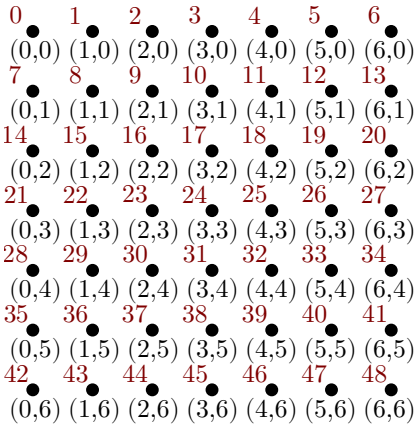


Figura III.1: *Arreglo cuadrado de nodos identificados por una pareja de índices. Esta pareja de índices se asocia al valor entero presentado en rojo por medio del polinomio de redireccionamiento* $PL(i,j) = 7j + i$.

Nótese la convención seguida para asignar los índices a un nodo. Un nodo con índice j más grande que otro está más abajo, y viceversa. Un nodo con índice i más grande, está a la derecha, y viceversa. La única dificultad que genera la inversión en el índice j es una inversión al momento de calcular distancias verticales.

Se revisa a continuación la implementación de los pasos de colisión, de flujo y de reconstrucción.

III.1.1. Paso de colisión

Como se describe en la sección II.3 el paso de colisión corresponde con la siguiente ecuación

$$\widetilde{f}_\alpha(\vec{r},\vec{e}_\alpha,t) = f_\alpha(\vec{r},\vec{e}_\alpha,t) - \frac{1}{\tau}[f_\alpha(\vec{r},\vec{e}_\alpha,t) - f_\alpha^{(\mathrm{eq})}(\vec{r},\vec{e}_\alpha,t)]. \tag{III.1}$$

Esta se reescribe de la siguiente forma

$$\widetilde{f}_\alpha(\vec{r},\vec{e}_\alpha,t) = \left(1 - \frac{1}{\tau}\right) f_\alpha(\vec{r},\vec{e}_\alpha,t) + \frac{1}{\tau} f_\alpha^{(\mathrm{eq})}(\vec{r},\vec{e}_\alpha,t), \tag{III.2}$$

donde $f_\alpha^{(\mathrm{eq})}(\vec{r},\vec{e}_\alpha,t)$ está dado por la ecuación (II.47):

$$f_\alpha^{(\mathrm{eq})} = \rho w_\alpha \left\{ 1 + \frac{3(\vec{e}_\alpha\cdot\vec{u})}{c^2} + \frac{9(\vec{e}_\alpha\cdot\vec{u})^2}{2c^4} - \frac{3\vec{u}^{\,2}}{2c^2} \right\}. \tag{III.3}$$

(III.2) se aplica para cada una de las velocidades microscópicas discretizadas $\vec{e}_\alpha$ dadas por la ecuación (II.65):

$$\vec{e}_\alpha = \begin{cases} (0,0), & & \alpha = 0 \\ (\cos\theta_\alpha, \operatorname{sen}\theta_\alpha)c, & \theta_\alpha = \pi + \left(\frac{\alpha-2}{2}\right)\frac{\pi}{2}, & \alpha = 2,4,6,8 \\ \sqrt{2}(\cos\theta_\alpha, \operatorname{sen}\theta_\alpha)c, & \theta_\alpha = \frac{3\pi}{4} + \left(\frac{\alpha-1}{2}\right)\frac{\pi}{2}, & \alpha = 1,3,5,7 \end{cases} \tag{III.4}$$

con

$$w_\alpha = \frac{\omega_i\omega_j}{\pi} = \begin{cases} 4/9, & \alpha = 0 \\ 1/9, & \alpha = 2,4,6,8 \\ 1/36, & \alpha = 1,3,5,7. \end{cases} \tag{III.5}$$

Es decir, en realidad se tienen 9 ecuaciones. Una para cada velocidad $\vec{e}_\alpha$.[1]

Sin embargo, para calcular cada una de las funciones de equilibrio $f_\alpha^{(\mathrm{eq})}$, se necesita conocer la densidad y la velocidad. Estas se encuentran con (II.53a) y (II.53b):

$$\rho = \sum_\alpha f_\alpha, \tag{III.6a}$$

$$\rho\vec{u} = \sum_\alpha \vec{e}_\alpha f_\alpha. \tag{III.6b}$$

La ecuación (III.6b) son en realidad dos ecuaciones. Para el esquema $D2Q9$, al usar las velocidades (III.4), estas dos ecuaciones se expanden de la siguiente forma:

$$\begin{aligned}\rho u_x = \sum_\alpha (e_\alpha)_x f_\alpha = 0\cdot f_0 + c\sqrt{2}\cos\left(\frac{3\pi}{4}\right) f_1 + c\cos(\pi) f_2 + c\sqrt{2}\cos\left(\frac{5\pi}{4}\right) f_3 + c\cos\left(\frac{3\pi}{2}\right) f_4 \\ + c\sqrt{2}\cos\left(\frac{7\pi}{4}\right) f_5 + c\cos(0) f_6 + c\sqrt{2}\cos\left(\frac{\pi}{4}\right) f_7 + c\cos\left(\frac{\pi}{2}\right) f_8,\end{aligned} \tag{III.7}$$

$$\begin{aligned}\rho u_y = \sum_\alpha (e_\alpha)_y f_\alpha = 0\cdot f_0 + c\sqrt{2}\,\mathrm{sen}\left(\frac{3\pi}{4}\right) f_1 + c\,\mathrm{sen}(\pi) f_2 + c\sqrt{2}\,\mathrm{sen}\left(\frac{5\pi}{4}\right) f_3 + c\,\mathrm{sen}\left(\frac{3\pi}{2}\right) f_4 \\ + c\sqrt{2}\,\mathrm{sen}\left(\frac{7\pi}{4}\right) f_5 + c\,\mathrm{sen}(0) f_6 + c\sqrt{2}\,\mathrm{sen}\left(\frac{\pi}{4}\right) f_7 + c\,\mathrm{sen}\left(\frac{\pi}{2}\right) f_8.\end{aligned} \tag{III.8}$$

Que se reducen a

$$\begin{aligned}\rho u_x &= 0\cdot f_0 - cf_1 - cf_2 - cf_3 + 0\cdot f_4 + cf_5 + cf_6 + cf_7 + 0\cdot f_8 \\ &= c(-f_1 - f_2 - f_3 + f_5 + f_6 + f_7),\end{aligned} \tag{III.9a}$$

$$\begin{aligned}\rho u_y &= 0\cdot f_0 + cf_1 + 0\cdot f_2 - cf_3 - cf_4 - cf_5 + 0\cdot f_6 + cf_7 + cf_8 \\ &= c(f_1 - f_3 - f_4 - f_5 + f_7 + f_8).\end{aligned} \tag{III.9b}$$

Al dividir las ecuaciones (III.9) entre la densidad calculada con (III.6a), se obtienen las componentes x y y de la velocidad $\vec{u}$.

Este paso se calcula para todo nodo, excepto para los nodos sólidos de cuerpo.

En resumen: En todo nodo que no sea sólido de cuerpo, se calcula primero su densidad, y la velocidad $\vec{u}$. A continuación se calculan las 9 poblaciones de equilibrio $f_\alpha^{(\mathrm{eq})}$. Finalmente con (III.2) se calcula $\widetilde{f}_\alpha(\vec{r}, \vec{e}_\alpha, t)$, que se asigna a los mismos arreglos usados para las poblaciones f_α.

En C, este paso queda como

```
void collide(void){
    int P;
    double rho, rhoux, rhouy, ux, uy, uCuadrada;
    double f0eq, f6eq, f8eq, f2eq, f4eq, f7eq, f1eq, f3eq, f5eq;

    /** Constantes. */
    double rtau = 1.0/tau;
    double rtau2 = 1.0 - rtau;
    double w0 = 4.0/9.0;
```

[1] Los vectores e_α son: $e_0 = (0,0)$, $e_1 = c(-1,1)$, $e_2 = c(-1,0)$, $e_3 = c(-1,-1)$, $e_4 = c(0,-1)$, $e_5 = c(1,-1)$, $e_6 = c(1,0)$, $e_7 = c(1,1)$, $e_8 = c(0,1)$.

```
    double wpar = 1.0/9.0;
    double wimpar = 1.0/36.0;
    double c=1.0;


    for (int i = 0; i < NY; ++i) {
        for (int j = 0; j < NX; ++j) {
            P = PL(j,i);
            if (M[P] != 0) {

                /** Se encuentran la densidad y la velocidad del fluido.*/
                rho = f0[P] + f6[P] + f8[P] + f2[P] + f4[P] + f7[P] + f1[P] + f3[P] + f5[P];
                rhoux = c*(f6[P] - f2[P] + f7[P] - f1[P] - f3[P] + f5[P]);
                rhouy = c*(f8[P] - f4[P] + f7[P] + f1[P] - f3[P] - f5[P]);
                ux = rhoux/rho;
                uy = rhouy/rho;
                velx[P] = ux;
                vely[P] = uy;

                /** Para graficar, se encuentra la magnitud de la velocidad del fluido. */
                velplot[P] = sqrt(ux*ux + uy*uy);

                /** Se calculan las poblaciones de equilibrio. */
                uCuadrada = 1.5*(ux*ux + uy*uy);
                f0eq = rho * w0 * (1.0 - uCuadrada);
                f2eq = rho * wpar * (1.0 - 3.0*ux + 4.5*ux*ux - uCuadrada);
                f4eq = rho * wpar * (1.0 - 3.0*uy + 4.5*uy*uy - uCuadrada);
                f6eq = rho * wpar * (1.0 + 3.0*ux + 4.5*ux*ux - uCuadrada);
                f8eq = rho * wpar * (1.0 + 3.0*uy + 4.5*uy*uy - uCuadrada);
                f1eq = rho * wimpar * (1.0 + 3.0*(-ux + uy)
                                       + 4.5*(-ux + uy)*(-ux + uy) - uCuadrada);
                f3eq = rho * wimpar * (1.0 + 3.0*(-ux - uy)
                                       + 4.5*(-ux - uy)*(-ux - uy) - uCuadrada);
                f5eq = rho * wimpar * (1.0 + 3.0*(ux - uy)
                                       + 4.5*(ux - uy)*(ux - uy) - uCuadrada);
                f7eq = rho * wimpar * (1.0 + 3.0*(ux + uy)
                                       + 4.5*(ux + uy)*(ux + uy) - uCuadrada);

                /** Se hace el paso de colisi'on. */
                f0[P] = rtau2 * f0[P] + rtau * f0eq;
                f1[P] = rtau2 * f1[P] + rtau * f1eq;
                f2[P] = rtau2 * f2[P] + rtau * f2eq;
                f3[P] = rtau2 * f3[P] + rtau * f3eq;
                f4[P] = rtau2 * f4[P] + rtau * f4eq;
                f5[P] = rtau2 * f5[P] + rtau * f5eq;
                f6[P] = rtau2 * f6[P] + rtau * f6eq;
                f7[P] = rtau2 * f7[P] + rtau * f7eq;
                f8[P] = rtau2 * f8[P] + rtau * f8eq;
            }
        }
    }

}
```

III.1.2. Paso de flujo

Para este paso, se usa la forma alternativa para el paso de flujo (II.57), encontrada en II.3:

$$f_\alpha(\vec{r}, \vec{e}_\alpha, t + \delta_t) = \widetilde{f}_\alpha(\vec{r} - \vec{e}_\alpha \delta_t, \vec{e}_\alpha, t). \tag{III.10}$$

Como se observa de la ecuación (III.10), este paso trabaja con el resultado del paso anterior. A pesar de que esta asignación es simple, hay que tomar una precaución. Se necesitan arreglos auxiliares en donde se guarda el valor obtenido. Esto es necesario porque otros nodos hacen uso de los valores $\widetilde{f}_\alpha(\vec{r}, \vec{e}_\alpha, t)$. De otra forma se modifican los valores a asignar. También, este paso sólo se aplica a los nodos

Del lado derecho de la ecuación (III.10), la posición es $\vec{r} - \vec{e}_\alpha \delta_t$. Si el punto $\vec{r}$ corresponde a la posición del nodo P, $\vec{r} - \vec{e}_\alpha \delta_t$ corresponde a uno de sus 8 nodos vecinos, pues $\vec{e}_\alpha \delta_t$ es un vector de tamaño equivalente a la separación entre nodos, δ_x cuando $\alpha = 2, 4, 6, 8$ y $\sqrt{2}\delta_x$ cuando $\alpha = 1, 3, 5, 7$.[2] Así, por ejemplo, para el vector $\vec{e}_1 = c(-1, 1)$, el producto $\vec{e}_1 \delta_t = \delta_x(-1, 1)$ pues $c = \delta_x/\delta_t$. Si P se identifica con los índices (i, j) el nodo vecino que corresponde a $\vec{r} - \vec{e}_1 \delta_t$ es el $(i+1, j+1)$. De hecho, una forma rápida para determinar el nodo vecino correspondiente, consiste en tomar $c = 1$ para el vector $\vec{e}_\alpha$, invertir el signo de la coordenada x y sumar al nodo (i, j) (la inversión de signo en x y no en y se debe a la inversión ya mencionada en el índice j). Una fórmula que resume esto es:

$$P_{\text{vecino}(\alpha)} = (-\hat{\imath} \cdot \vec{e}_\alpha|_{c=1})\,\hat{\imath} + (\hat{\jmath} \cdot \vec{e}_\alpha|_{c=1})\,\hat{\jmath} + (i, j) = (-e_{\alpha_x}, e_{\alpha_y})|_{c=1} + (i, j). \tag{III.11}$$

El valor de la población f_k para el nodo con índices (i, j), se coloca en la posición $PL(i, j)$ del arreglo fk, es decir, en $fk[PL(i, j)]$. Si se llama $tmpfk$ al arreglo temporal con el que después se hace la asignación final, el código correspondiente a esto, para un nodo específico P, se ve como sigue:

```
P = PL(i,j);
tmpf6[P] = f6[PL( i-1,  j   )];
tmpf8[P] = f8[PL(  i  , j+1 )];
tmpf2[P] = f2[PL( i+1,  j   )];
tmpf4[P] = f4[PL(  i  , j-1 )];
tmpf7[P] = f7[PL( i-1, j+1 )];
tmpf1[P] = f1[PL( i+1, j+1 )];
tmpf3[P] = f3[PL( i+1, j-1 )];
tmpf5[P] = f5[PL( i-1, j-1 )];
```

Después de hacer estas operaciones para cada nodo P, cada uno de estos valores tmpfk[P] se asigna de vuelta a fk[P], para cada nodo P.

Hay dos últimas observaciones. Aunque algún nodo sólido de cuerpo se encuentre alrededor del nodo P, y su población que esté asignando al nodo P sea un número sin sentido, no importa, pues el paso siguiente, el de reconstrucción genera la población correcta (al menos de forma aproximada). Las poblaciones que se usan para el paso de reconstrucción provienen de nodos que no son nodos sólidos de bulto. La segunda observación es que un nodo frontera puede estar en la frontera del dominio total, así que no existen algunos nodos. Se asigna un número cualquiera para la población correspondiente, porque de cualquier forma no es necesaria para la reconstrucción. Para simplificar, y no modificar la estructura mostrada más arriba, se definen cuatro nuevos índices: im1, ip1, jm1, jp1. "i minus 1", "i plus 1", etc. im1 es igual a (i-1), ip1 es igual a (i+1), etc., a menos que el índice que generen sea un índice que no esté en el dominio, v.g. im1=0-1=-1, o jp1=(Nx-1)+1=Nx. En esos casos, se asigna la misma variable i o j, v.g. ip1=i o jm1=j.

[2] El caso cuando $\alpha = 0$ equivale a la identidad, por eso no se realiza.

En C, este paso queda como

```
void stream(void){
int im1,ip1,jm1,jp1,P;

/** Mueve las poblaciones fk a arreglos auxiliares. */
for (int i = 0; i < NY; ++i) {
    /** En el borde del dominio, no se mueven los valores de fk. */
    if (i == 0) {
        im1 = 0;
    } else {
        /**i minus 1*/
        im1 = i-1;
    }
    if (i == (NY-1)) {
        ip1 = NY-1;
    } else {
        /**j plus 1*/
        ip1=i+1;
    }

    for (int j = 0; j < NX; ++j) {
        /** Se calcula el 'indice en el arreglo lineal. */
        P  = PL(j,i);
        if (M[P] != 0) {

            /** En el borde del dominio, no se mueven los valores de fk. */
            if (j == 0) {
                jm1=0;
            } else {
                /**j minus 1*/
                jm1 = j-1;
            }
            if (j == (NX-1)){
                jp1=NX-1;
            } else {
                /** j plus 1*/
                jp1 = j+1;
            }
                /** Se asignan las poblaciones fk a arreglos auxiliares. */
            tmpf6[P] = f6[PL(jm1,i)];
            tmpf8[P] = f8[PL(j,ip1)];
            tmpf2[P] = f2[PL(jp1,i)];
            tmpf4[P] = f4[PL(j,im1)];
            tmpf7[P] = f7[PL(jm1,ip1)];
            tmpf1[P] = f1[PL(jp1,ip1)];
            tmpf3[P] = f3[PL(jp1,im1)];
            tmpf5[P] = f5[PL(jm1,im1)];

        }
    }
}

/** Se hace ahora s'i el paso de flujo. */
for (int i = 0; i < NY; ++i) {
    for (int j = 1; j < NX; ++j) {
        P = PL(j,i);
```

```
            if (M[P] != 0) {
                f6[P] = tmpf6[P];
                f8[P] = tmpf8[P];
                f2[P] = tmpf2[P];
                f4[P] = tmpf4[P];
                f7[P] = tmpf7[P];
                f1[P] = tmpf1[P];
                f3[P] = tmpf3[P];
                f5[P] = tmpf5[P];
            }
        }
    }

}
```

III.2. Implementación de las condiciones de frontera

La implementación de las condiciones de frontera está ligada, de acuerdo con el esquema seguido para el tratamiento de fronteras curvas, al paso de reconstrucción. La razón es que en los nodos frontera la velocidad se aproxima con una interpolación o extrapolación usando la velocidad en la frontera. Esto, por supuesto, impone la condición de frontera. Pero ahora, quien se encarga de transmitir la información correspondiente de la condición de frontera son los nodos frontera.

En el caso cuando se opera un nodo frontera fluido, parece natural que en los pasos subsecuentes éste entre en los pasos de colisión y de flujo. Sin embargo, en un nodo frontera sólido parece absurdo pensar que exista fluido pasando por él, y con ello, seguir con los pasos de colisión y de flujo. Uno puede pensar casos donde la frontera se moja o algo similar y ahí sí tendría sentido, pero es de esperar que el comportamiento físico del fluido absorbido por la pared sea distinto que el simulado con la ecuación de Boltzmann. Así, queda descartado que el fluido pase por ese tipo de nodo. La forma de entender por qué se considera operar estos nodos es que, para resolver una ecuación diferencial se necesita cumplir las condiciones de frontera. Si se cumplen las condiciones de frontera (y la solución es única), lo que pase fuera del dominio donde interesa conocer la solución al problema, se descarta libremente.

El paso de reconstrucción trata específicamente con los nodos frontera.

III.2.1. Paso de reconstrucción

El paso de reconstrucción empieza por hacer una interpolación o extrapolación de las velocidades en los nodos frontera. La función encargada de esto es `calculaVelocidadEnNodosFrontera(...)`. La extrapolación se lleva a cabo en los nodos frontera sólidos y la interpolación en los nodos frontera fluidos.

Esta extrapolación o interpolación es diferente de acuerdo con la posición relativa del nodo con la frontera. Como se discutió en la sección II.4.3, la velocidad se interpola o extrapola de acuerdo a la dirección que hace un menor ángulo con la normal a la frontera.

Para encontrar la dirección de interpolación/extrapolación, los únicos nodos disponibles para tomar valores de $\vec{u}$ son los nodos fluidos, que se marcan con 1. Es decir, el conjunto de velocidades $\vec{e}_i$, las velocidades microscópicas de la discretización del método de Lattice Boltzmann, está restringido a

ciertas direcciones. Dada esta restricción, se realiza el producto punto de $\hat{e}_i$ con el vector normal a la frontera $\vec{n}$, y se identifica el vector que da el máximo valor. La razón es que el máximo valor en el producto punto corresponde a identificar el vector $\vec{e}_j$ que hace el menor ángulo con la normal $\vec{n}$.

Entonces se necesita el vector normal a la frontera para encontrar la dirección de interpolación/extrapolación. Los métodos de las secciones III.3 y III.4 se encargan de encontrar un ajuste por segmentos de recta a la frontera, asociando a cada nodo frontera, la pendiente y ordenada al origen del segmento de recta más cercano. También se asocia a cada nodo, un sentido, que hace referencia a un sentido específico con el que se recorre la curva frontera. A partir de la pendiente a de la recta ajustada, se construye un vector tangencial a la frontera, dado por la pareja de coordenadas $(1, a)$.[3] Este vector tangencial tiene una componente x positiva, por lo que se dice que apunta a la derecha. Este vector es adecuado cuando la frontera (cercana) se recorre en ese mismo sentido o "a la derecha", registrado en el arreglo `sentido[]` con un `+1`. Cuando la frontera se recorre en el sentido contrario, es decir, hacia coordenadas en x menores, se debe invertir el vector tangencial. Así, el vector tangencial en estos casos es el $(-1, -a)$. También, el valor asignado al segmento de recta que ajusta la frontera, en el arreglo `sentido[]`, es `-1`. Si el valor en el arreglo es, para un nodo particular, s, entonces el vector tangencial es $s(1, a) = (s, s * a)$. Por lo tanto, el vector perpendicular a este, que apunta a la región interior es $\vec{n} = (s * a, -s)$.

Tabla III.1: *Arreglos para la búsqueda del máximo de* $\frac{\vec{e}_i \cdot \vec{n}}{||\vec{e}_i||}$

i	Ix[i]	Iy[i]	Cx[i]	Cy[i]	Cnorma[i]
0	0	0	0.0	0.0	0.0
1	-1	-1	-c	c	$\sqrt{2c^2}$
2	-1	0	-c	0.0	c
3	-1	1	-c	-c	$\sqrt{2c^2}$
4	0	1	0.0	-c	c
5	1	1	c	-c	$\sqrt{2c^2}$
6	1	0	c	0.0	c
7	1	-1	c	c	$\sqrt{2c^2}$
8	0	-1	0.0	c	c

Para identificar qué nodos vecinos son nodos fluidos, se usan los arreglos `Ix[i]` y `Iy[i]` de la tabla III.1. El índice `i` en estos arreglos corresponde a la dirección que tiene el vector $\vec{e}_i$. Así, `PL(i+Ix[r],j+Iy[r])` da el nodo vecino en la dirección `r`, y si el nodo en esa posición es fluido, se prueba como candidato a máximo del producto punto $\frac{\vec{e}_i \cdot \vec{n}}{||\vec{e}_i||}$, tomando `i=r`.

Para calcular el producto punto, se usan los arreglos `Cx[i]`, `Cy[i]` y `Cnorma[i]`, en la tabla III.1. La entrada `i` contiene la componente x, la componente y y la norma del vector $\vec{e}_i$, respectivamente. Entonces, el producto punto se calcula con `sentido[P]*(a*Cx[r]-Cy[r])/(Cnorma[r])`, donde `sentido[P]` es el sentido de la frontera en la proximidad del nodo (i, j).

En C, la identificación del máximo se escribe de la siguiente forma:

[3] Cuando a corresponde a una pendiente infinita, este vector tangencial no sirve. Pero para ese caso especial, un vector tangencial es el $(0, 1)$. De acuerdo al sentido asignado, que en este caso dice si la frontera se recorre hacia arriba o hacia abajo, el vector normal es el $(1, 0)$ o el $(-1, 0)$. Si la frontera se recorre hacia arriba, el interior del dominio queda hacia la derecha, por lo que el vector normal es el $(1, 0)$; si se recorre hacia abajo, el interior queda hacia la izquierda, y el vector normal es el $(-1, 0)$. Si s vale `1` en el primer caso, y `-1` para el segundo, el vector normal se escribe en general como $(s, 0)$.

```
    for (int r = 1; r < 9; ++r) {
      if (M[PL(i+Ix[r],j+Iy[r])] == 1) {
        if ((max2 = sentido[P]*(a*Cx[r]-Cy[r])/(Cnorma[r])) > max) {
          max = max2;
          rmax = r;
        }
      }
    }
```

Cuando la pendiente corresponde a una recta vertical, el producto punto se modifica porque el vector normal a la frontera es $(s, 0)$. Así, la búsqueda del máximo del producto punto se modifica como sigue:

```
    for (int r = 1; r < 9; ++r) {
      if (M[PL(i+Ix[r],j+Iy[r])] == 1) {
        if ((max2 = sentido[P]*(Cx[r])/(Cnorma[r])) > max) {
          max = max2;
          rmax = r;
        }
      }
    }
```

El índice `rmax` encontrado da la dirección de interpolación/extrapolación.

Con este índice se calcula la distancia entre el nodo y el segmento de recta que ajusta la frontera cercana. Se calcula encontrando el punto de intersección de una recta con la dirección de $\vec{e}_{rmax}$, y luego usando la fórmula de distancia entre dos puntos. Si el nodo es un nodo de frontera sólido, la distancia se hace negativa. Si la distancia es cero, entonces la frontera queda sobre el nodo, y la velocidad en el nodo frontera está dada por la velocidad impuesta en la frontera. Si es distinta de cero, la velocidad en el nodo se calcula según la ecuación (II.73), $(\vec{u}(dP) = \vec{u}_0 l_0(dP) + \vec{u}_1 l_1(dP) + \vec{u}_2 l_2(dP))$. Para calcular (II.74), (II.75), (II.76), se necesita las distancias $dH1$ y $dH2$, que se calculan de la siguiente forma

```
  if (rmax == 2 || rmax == 4 || rmax == 6 || rmax == 8) {
    dH1 = delta + dP;
    dH2 = 2.0*delta + dP;
  } else {
    dH1 = sqrt(2.0)*delta + dP;
    dH2 = 2.0*sqrt(2.0)*delta + dP;
  }
```

Cuando `rmax` es 2, 4, 6 u 8, la distancia entre nodos es `delta`, y cuando tiene otro valor, es $\sqrt{2}$`delta`. Cuando el nodo es uno de frontera sólido, `dP` es negativo, y la distancia al primer y segundo nodo se reducen, como se necesita.

Finalmente, se hace la interpolación/extrapolación de cada componente de la velocidad para el nodo $P = (i, j)$:

```
  velx[P] = l_0(dP,dH1,dH2)*Uox[P]
          + l_1(dP,dH1,dH2)*velx[PL(i+Ix[rmax],j+Iy[rmax])]
          + l_2(dP,dH1,dH2)*velx[PL(i+2*Ix[rmax],j+2*Iy[rmax])];

  vely[P] = l_0(dP,dH1,dH2)*Uoy[P]
          + l_1(dP,dH1,dH2)*vely[PL(i+Ix[rmax],j+Iy[rmax])]
          + l_2(dP,dH1,dH2)*vely[PL(i+2*Ix[rmax],j+2*Iy[rmax])];
```

donde `velx[P]` es la componente en x de la velocidad, `vely[P]` la componente en y, y `Uox[P]` y `Uoy[P]` son las componentes de la velocidad en la frontera cercana.

Si la frontera queda en el borde del dominio total, es decir en `i=0, i=NX-1, j=0, o j=NY-1`, sólo se restringe la búsqueda de `rmax`, según el punto en el borde.

Luego, se necesita calcular las componentes del tensor de razón de deformación $\mathcal{S}$. Según se vio al final de II.4.1, en la ecuación (II.64), para calcularlas, se necesita encontrar $\frac{\partial u_x}{\partial x}$, $\frac{\partial u_x}{\partial y}$, $\frac{\partial u_y}{\partial x}$ y $\frac{\partial u_y}{\partial y}$.

Para calcular estas, se identifica la orientación de la frontera identificando los puntos de intersección del segmento de recta ajustado a la frontera con una recta vertical y una recta horizontal que pasen por el nodo, en el que se calcula $\mathcal{S}$.

Si la frontera queda sobre el nodo, se usan las velocidades de los nodos vecinos a izquierda y derecha, y arriba y abajo. En este caso, para calcular las derivadas se usa alguna de las ecuaciones (II.82), (II.83) o (II.84), según el caso, pues los nodos son equidistantes, por supuesto, sin el último término que es desconocido y da el error. En otro caso, se usa la ecuación (II.81), de nuevo, sin el último término. Cuando la frontera no queda sobre un nodo, y la recta ajustada es horizontal o vertical, es posible usar alguna de las ecuaciones (II.82), (II.83) o (II.84), según el caso.

A continuación se calculan $\mathfrak{g}_\alpha^{\text{eq}}$ y $\mathfrak{g}_\alpha^{\text{neq}}$ (véase la sección II.4.3):

$$\mathfrak{g}_\alpha^{(\text{eq})} = w_\alpha \left\{ 1 + \frac{3(\vec{e}_\alpha \cdot \vec{u})}{c^2} + \frac{9(\vec{e}_\alpha \cdot \vec{u})^2}{2c^4} - \frac{3\vec{u}^{\,2}}{2c^2} \right\}. \tag{III.12}$$

y

$$f_\alpha^{(\text{neq})} \approx -\frac{\tau w_\alpha}{c_s^2} \mathcal{Q}_\alpha : \mathcal{S}. \tag{III.13}$$

El producto diádico $\mathcal{Q}_\alpha : \mathcal{S}$ es, según la ecuación (II.64):

$$(\mathcal{Q}_\alpha)_{ij}\, \mathcal{S}_{ij} = (e_\alpha)_x (e_\alpha)_x \frac{\partial u_x}{\partial x} + (e_\alpha)_y (e_\alpha)_x \frac{\partial u_x}{\partial y} + (e_\alpha)_x (e_\alpha)_y \frac{\partial u_y}{\partial x} + (e_\alpha)_y (e_\alpha)_y \frac{\partial u_y}{\partial y} - c_s^2 \left[\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} \right]. \tag{III.14}$$

Con estas expresiones se calcula cada una de las expresiones $\mathfrak{g}_\alpha = \mathfrak{g}_\alpha^{\text{eq}} + \mathfrak{g}_\alpha^{\text{neq}}$, según la dirección de la velocidad α.

Finalmente se calcula ρ usando alguna de las expresiones (II.91), (II.92) o (II.93), haciendo las sumas indicadas sobre las poblaciones conocidas, como se indica.

III.3. Identificación de frontera

Matemáticamente, para identificar la frontera de un dominio basta con encontrar los puntos para los que si se traza una bola $B_\epsilon = \{(x,y) | \sqrt{(x^2+y^2)} < \epsilon\}$, de cualquier radio $\epsilon > 0$, siempre se tienen puntos tanto en el interior del dominio como en el exterior. Sin embargo, probar esta característica es en la práctica imposible, primero por la cantidad infinita de puntos por los que una frontera está conformada y segundo porque si las pruebas se detienen en un valor definido ϵ_0, no hay garantía de que este punto sea de frontera, porque siempre existe la posibilidad de que con un radio menor no haya puntos en la bola. Una alternativa consiste en tomar una curva que una dos puntos, uno en el dominio y otro fuera de él, y avanzar sobre la curva, por ejemplo, del punto exterior al punto interior, y la intersección de esta curva con el dominio es el punto frontera. Para simplificar las cosas, esta curva puede ser un

segmento de recta. En cualquier caso, la cantidad de puntos que computacionalmente son manejables es una cantidad finita. En lo que sigue se restringe la discusión a dominios en $\mathbb{R}^2$ o dominios planos.

Un conjunto de figuras o dominios en que es fácil representar la frontera son los polígonos. Basta con sus vértices, y una lista de aristas que los unan, para describir perfectamente su frontera, a pesar de estar conformados por una cantidad infinita de puntos. Una de las características de los polígonos es que están conformados exclusivamente por segmentos rectos. Comparativamente, en general, una figura con fronteras curvas no es posible representarla con una cantidad finita de puntos. Por supuesto, la frontera podría ser representada por una curva definida a pedazos. Pero elegir cada una de las curvas que mejor representen al dominio tiene muchas dificultades. Además, se necesita mantener la información de qué puntos o regiones de frontera están caracterizados por cuál curva, si el método busca, por ejemplo, la normal a la frontera por esos puntos. También, la obtención de la normal a la superficie por un punto particular puede requerir un método extra. Dada la simplicidad de un polígono, la posibilidad de tomar uno como aproximación del dominio presenta simplificaciones. A saber, basta sólo con dos parámetros para determinar el segmento de recta que conforma la frontera. La precisión que se pueda obtener con esta aproximación poligonal del dominio depende del número de puntos que se colocan en ella.

Para el trabajo se consideró la alternativa de aproximar el dominio por polígonos. De hecho, una de las formas que tiene el programa para determinar la frontera de un dominio, implícitamente trabaja con un polígono.

Con ayuda de la biblioteca "`stb_image.h`" se extrae un mapa de bits, que se convierte en un arreglo binario que contiene la información de qué puntos de la imagen son interiores y cuáles son exteriores. A este arreglo binario se llama máscara binaria. Interpretando los unos como pixel negro y los otros como pixel blanco, gráficamente la máscara binaria se vería como la imagen derecha en la figura III.2. De hecho, para hacer ese tipo de asignación, los pixeles dentro de la imagen deben tener una característica que los distinga de los otros. Podría ser cierta cantidad de rojo, verde o azúl. Se optó por tomar los blancos como exteriores y los negros como interiores. Así, la entrada correcta que recibe el programa es una imagen como la derecha, y no como la izquierda de la figura III.2.

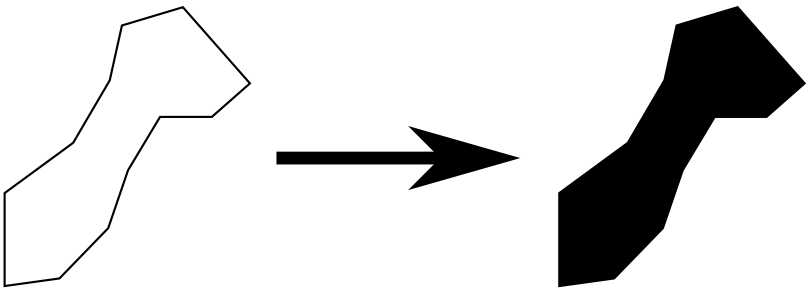


Figura III.2: *Entrada de imagen correcta al programa.*

Para un método como el de Lattice Boltzmann esta información puede ser suficiente para determinar en qué puntos tiene que trabajar. Sin embargo, los métodos que trabajan directamente con esta información tienen una precisión limitada y entonces limitan el rango posible de aplicabilidad del método. En el capítulo anterior se discutió un método que trabaja con cierto conjunto de nodos específicos, en tres configuraciones distintas. Aquí se presenta la implementación de la configuración 3 presentada, pues se observa en [29] que da los mejores resultados. Para su método no basta con la elección de estos nodos, sino que además necesita información de la frontera. El conjunto de puntos extraido es útil para esto también.

La función para la determinación de puntos en la frontera a partir de la máscara binaria extraida de la imagen se describe en la siguiente subsección, junto con una función que se usa previamente, que cuenta cuántos puntos colocan en la frontera. Con el objetivo de darle un significado un poco más físico a estos puntos de frontera, apartir de ahora se les llama partículas a estos puntos, y con ello se retoma el llamar pared a la frontera del dominio.

III.3.1. Generación de partículas en la pared del dominio.

Previamente a la asignación de coordenadas a las partículas en la pared, se cuenta el número de partículas total a colocar. La función `conteoParticulasPared(...)` se encarga de esto. Presupone que la máscara binaria proviene de una imagen con un dominio del tipo y-simple. Un dominio y-simple (véase la figura III.3a) es un dominio en $\mathbb{R}^2$ en el que si se traza una recta vertical cualquiera (con excepción de los extremos del dominio en x), únicamente intersecta a la frontera del dominio en dos puntos; en otras palabras, la región queda entre dos funciones de x. Esto es un poco restrictivo en cuanto a la generalidad del dominio que se quiere trabajar. Hay una manera de extender el programa a regiones que no son y-simples. La forma se discute más adelante, pero no se implementa.

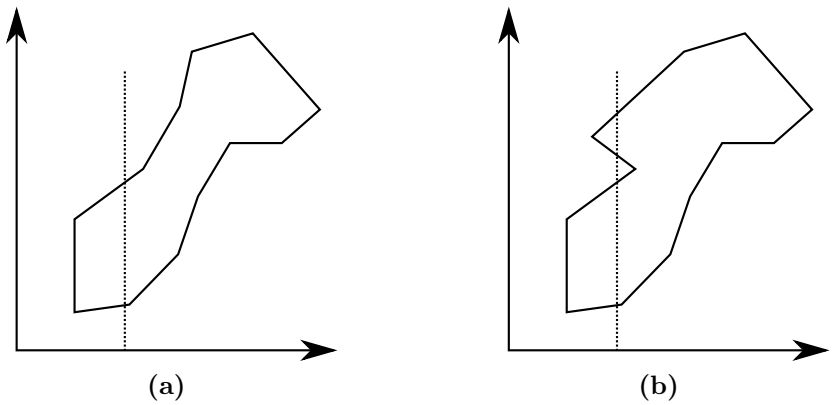


Figura III.3: *(a) Dominio y-simple: Cualquier recta vertical intersecta únicamente dos veces a la frontera del dominio, con excepción de las que pasan por las abscisas de los extremos. (b) Dominio que no es y-simple: Existen rectas verticales que intersectan más de dos veces la frontera del dominio, además de las que pasan por las abscisas de los extremos.*

Con la máscara binaria cumpliendo las propiedades discutidas, el conteo de partículas se hace determinando la primer y última columna con unos. Claro, se supone que el dominio original no es disjunto (que se puede unir cualquier punto del dominio con otro por medio de una curva cuyos puntos estén en el dominio), y que por ello la máscara binaria posee una región donde cada columna tiene más de dos unos. Es una revisión por fuerza bruta, porque no se conoce de entrada qué columna tiene unos ni en qué renglones, de tenerlos.

Con la información del índice de la primer columna donde hay unos y de la última columna, se cuenta el número de unos en cada una de estas columnas. Si cada pixel en el borde de la máscara binaria representa una partícula, el número de partículas en la pared izquierda N1, en la derecha N2, y en la parte de arriba y abajo del dominio N3 se encuentra contando el número de unos en la primer columna, el número de unos en la última columna, y el número de columnas entre la primer y última columna.

Sumando estas y las partículas extra, que se colocan entre cada partícula asociada a un pixel, se obtiene el número total de partículas.

A continuación se generan arreglos con el tamaño del número de partículas. Uno de ellos para guardar la coordenada x y el otro para la coordenada y.

La siguiente función que se ejecuta es la de `generacionDeParticulasEnPared(...)`. Esta función recorre el bitmap columna por columna. Llena primero las posiciones de los nodos en la frontera de arriba y abajo. Para ello, identifica los pixeles en la frontera de arriba, y automáticamente asigna una posición con base en sus índices (i, j). Prende una bandera que indica que ya está dentro del interior del dominio, y cuando encuentra un cero, asigna una posición de acuerdo con los índices en la posición de arriba, i.e. $(i, j-1)$. Para los de la frontera de arriba, estas posiciones se van colocando en orden, saltando las posiciones de las partículas intermedias que falta por asignar. Para los de la frontera de abajo, estos se colocan de atrás para adelante en el arreglo en la posición que les corresponde para hacer que toda la frontera se recorra en una sola dirección.

Luego, las partículas de la frontera izquierda y la derecha son asignadas. Los nodos de la pared derecha se escriben en orden inverso para seguir la misma dirección en la frontera. De nuevo, en el arreglo que guarda las posiciones escritas, se deja un espacio para las partículas intermedias.

Finalmente, se asigna las posiciones de las partículas intermedias, tomando los puntos de las partículas asignadas previamente, y colocando equidistantemente las coordenadas de partículas intermedias en el segmento que une a esos dos puntos.

En la siguiente subsección se describe la forma de identificar rápidamente la región en que una partícula se encuentra.

III.3.2. Identificación de localidades para una partícula.

La red de nodos permea a todo el espacio en que se hace la simulación. Incluso, puede extenderse a regiones fuera del dominio en que se busca la solución del problema. Ciertos nodos externos al dominio se consideran como parte de muchos métodos para resolver las condiciones de frontera del método de Lattice Boltzmann. Por practicidad, uno puede considerar una red de nodos que cubra todo un dominio rectangular, aunque el dominio físico donde se quiere resolver el problema sea una porción de él, y no su totalidad. Las partículas que se determinaron con las funciones de las subsección anterior viven en este espacio generalizado. De hecho, estas partículas determinan a final de cuentas la asignación de tipo que cada nodo del dominio general recibe.

La forma en la que estas partículas determina si un nodo se debe etiquetar de una forma u otra se explica a continuación. En general, una partícula se encuentra espacialmente entre cuatro nodos (véase figura III.4). O, de otra forma, si se unen los nodos (i, j), $(i, j+1)$, $(i+1, j+1)$ y $(i+1, j)$ con aristas (quitando las diagonales), cualquier partícula que se encuentre contenida en el cuadrado resultante tiene cercanos los mismos nodos. Ahora, esa partícula en realidad forma parte de la pared, así que está unida con otras dos partículas de la pared. Esas dos partículas pueden estar dentro de la localidad descrita. Si se extiende este razonamiento, en algún momento una de estas partícula se une con otra en una localidad similar. Si el espaciamiento entre estas partículas no es muy grande, la localidad de la última partícula tiene como vecino al menos un nodo de la primer localidad. Es posible, sin embargo, que la partícula se encuentre sóla en la localidad. Pero el resultado sigue siendo el mismo. La pared, formada por las partículas, necesariamente cruza la localidad. Esto muestra que el enlace entre cualquier par de nodos necesariamente se intersecta por la frontera, lo que los convierte automáticamente en nodos frontera.

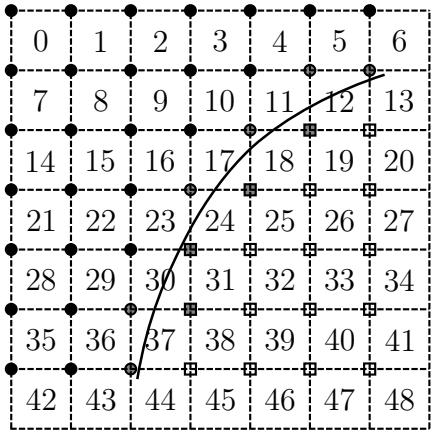


Figura III.4: *Esquema de localidades del tipo 1.*

Entonces, por una parte la localidad de las partículas establece cuáles son nodos frontera, y por otra, establece qué puntos se encuentran cercanos entre sí, y con esto ajustar alguna curva a la frontera.

Hay casos especiales en que una partícula pertenece a dos o cuatro localidades. Esto sucede cuando la partícula queda sobre una arista horizontal o una arista vertical entre dos nodos vecinos, para el primer caso, o cuando queda precisamente sobre un nodo, para el segundo caso. Surge entonces la pregunta de a qué nodo debería asociarse una partícula. Pero tomando en cuenta que la presencia de una partícula determina que se establezca un nodo como nodo frontera, no se descarta ninguna, y se considera que una partícula en esa situación está en todas las localidades compartidas.

Para el caso de dos localidades vecinas, ocupadas por la frontera, hay entonces nodos frontera en común. Para estos nodos, ambas localidades quedan igual de cerca, y en cada una se ajusta una curva a la frontera en ambas localidades. En estos casos, la decisión no es clara de qué curva de ajuste debe asignarse a los nodos en común. Una posibilidad sería utilizar una curva de ajuste para todas estas localidades vecinas. Pero hay otras posibilidades.

El esquema de localidades descritas en los párrafos anteriores se llaman localidades de tipo 1. Su uso principal consiste en la identificación del tipo de los nodos.

Un segundo esquema de localidades, que se llaman del tipo 2, consiste en el espacio formado si se colocan los vértices de la localidad en los centros de cada una de las 4 localidades de tipo 1, vecinas a un nodo (véase figura III.5).

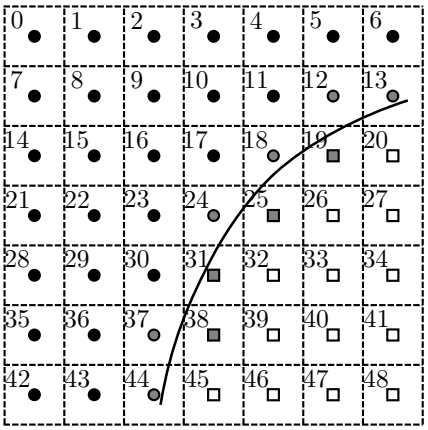


Figura III.5: *Esquema de localidades del tipo 2.*

Para este segundo esquema, se garantiza que las partículas en la localidad sean las más próximas al nodo. Entonces, la curva que se ajusta a la pared también queda más próxima al nodo.

Cualquiera de los dos esquemas permite identificar a un nodo frontera. Sin embargo, es posible que alguno de estos o ambos no detecten a todos los nodos frontera por falta de partículas. Por ejemplo (véase figura III.5), para el esquema de localidades tipo 2, si una partícula se encuentra en la localidad 24 y la siguiente en la 18, y ninguna de esas partículas es compartida por la localidad 25, entonces el nodo 25 no se etiqueta como nodo frontera. Algo similar ocurre con las localidades de tipo 1. Sin embargo, por la naturaleza de las localidades de tipo 1, es menos probable que falten de etiquetar nodos frontera. Cada una de estas localidades etiqueta 4 nodos, mientras que las del tipo 2 sólo etiquetan uno.

Una manera de evitar el problema es pensar en localidades de un tamaño mayor, pero el ajuste local de la frontera es más tardado por considerar más puntos.

Otra forma es considerar un tipo de localidad y corregir con el otro. El enfoque de hecho es usar las localidades de tipo 1 para marcar los nodos frontera, y también con estas, escoger los nodos con los que trabaja la configuración 3 usada por el método de condiciones de frontera descrito en el capítulo II. Para los nodos frontera marcados, (más adelante se describe el marcado de los nodos) a los que no se les haya asignado ciertos valores, se trabajan en el esquema de tipo 2.

Las dos funciones que se encargan de la asignación de localidades de cada tipo a las partículas son `identificaLocalidadesDeParticulasConfiguracion1(...)` e `identificaLocalidadesDeParticulasConfiguracion2(...)`, y trabajan de forma muy similar.

Además de la forma mostrada en las figuras III.4 y III.5 para referirnos a una localidad, existe una forma por medio de índices para referirnos a estas. Nótese que en el esquema de localidades tipo 1 al nodo en la esquina superior izquierda de cada localidad su número asociado por el polinomio de redireccionamiento $PL(i,j) = i + j * NX$ es el mismo que el de la localidad correspondiente. En otras palabras la localidad número $PL(i,j)$ siempre tiene de esquina superior izquierda al nodo (i,j). En el esquema de localidades tipo 2, sólo hay un nodo contenido por la localidad $PL(i,j)$, el (i,j). Así, en el esquema 1, los índices de la localidad son los del nodo de la esquina superior izquierda, y en el esquema 2, los índices del nodo que está en su centro.

Si se encuentra estos índices para cada partícula, se reconstruye a partir de ellos su localidad. Para encontrarlos en el esquema 1 se usa lo siguiente:

```
m = ParticulasFronterax[k]/delta;
l = ( (NY-1)*delta - ParticulasFronteray[k] )/delta;
```

`m` y `l` son los índices que tiene la k-ésima partícula con coordenadas (`ParticulasFronterax[k]`,`ParticulasFronteray[k]`), `delta` es una constante que se define en el preámbulo del programa, y `NY` es el número de nodos en la dirección vertical. La operación usa un *cast* al resultado de la operación de la derecha. Por ejemplo, si `ParticulasFronterax[k]` es igual a `m*delta+r`, la operación devuelve `m`, porque el número resultante de la división `ParticulasFronterax[k]/delta` es de la forma `m + r/delta` y `r/delta` se descarta por no ser entero. Si la coordenada x del nodo es `Px(m)`, cuando `Px(m)` $\leq$ `ParticulasFronterax[k]` $<$ `Px(m) + delta` el índice que se le asigna es `m`. Algo análogo sucede con `l`, pero la estructura de la operación es diferente porque el índice `l` con menor valor corresponde con la mayor altura del dominio.

Para el caso de las localidades de tipo 2, la identificación de índices se hace con:

```
m=(ParticulasFronterax[k]+delta/2)/delta;
l=((NY-1)*delta - ParticulasFronteray[k]+delta/2)/delta;
```

Ahora, si `ParticulasFronterax[k]` es igual a `m2*delta + r`, y se suma con `delta/2`, es posible que `r+delta/2` sea igual a `delta + r2` que, al dividirse entre `delta` de `m2 + 1 + r2/delta`, descartando de nuevo la parte fraccional `r2/delta`. La situación descrita sucede cuando `r` es mayor o igual a `delta/2`. Si la coordenada x del nodo es `Px(m)`, cuando `Px(m) - delta/2` $\leq$ `ParticulasFronterax[k]` $<$ `Px(m) + delta/2` el índice que se le asigna es `m`. Analogamente para `l`, y de nuevo la estructura de la operación es diferente por la misma razón mencionada.

Nótese que con esta operación, una partícula en el borde derecho o en el borde de abajo de la localidad de tipo 1 o 2, no se asigna a esa localidad, sino a la de la derecha, en el primer caso, y a la de abajo, en el segundo. Pero ya que si una partícula se encuentra en el borde entre dos localidades o en el vértice entre cuatro localidades, se le asigna las dos o cuatro según el caso. Para las localidades de tipo 2, cualquier partícula en el dominio se le asigna una localidad que al menos tiene parte de ella en el dominio (la localidad cero tiene tres cuartas partes fuera del dominio, pero esto no representa ningún problema). En el caso de las localidades de tipo 1, para partículas en el borde derecho del dominio general, o en el borde inferior, estrictamente se debería asignar la localidad `PL(NX-1,l)` y `PL(m,NY-1)`, respectivamente. Pero como dichas partículas también están en la localidad a la izquierda y de arriba, respectivamente, se considera que sólo están en estas. Una partícula en la esquina inferior derecha sólo se asigna a la localidad `PL(NX-2,NY-2)`.

La forma de identificar si una partícula está entre dos localidades para la localidad de tipo 1 es probando las igualdades

```
delta*m == ParticulasFronterax[k];
l*delta == ( (NY-1)*delta - ParticulasFronteray[k] );
```

o para evitar problemas de redondeo

```
fabs( delta*m - ParticulasFronterax[k] ) > epsilon;
fabs( l*delta - ( (NY-1)*delta - ParticulasFronteray[k] ) ) > epsilon;
```

con `epsilon` una constante pequeña, que determina cuándo se considera que dos números son iguales. Si se cumple la primera, quiere decir que la partícula está entre dos localidades vecinas horizontalmente; si se cumple la segunda, la partícula está entre dos localidades vecinas verticalmente; si se cumplen las dos, quiere decir que tiene cuatro localidades vecinas.

La identificación para cuando una partícula está entre dos localidades del tipo 2 es análoga.

En cualquiera de los dos esquemas, para guardar las localidades en las que está cada partícula, se usa los arreglos `Loc` y `Loc2` para los esquemas 1 y 2, respectivamente, que cada uno tiene el tamaño de cuatro veces el número total de partículas. Cada cuatro entradas de estos arreglos están asignadas a una partícula específica. Es decir, las entradas `4*k`, `4*k+1`,`4*k+2` y `4*k+3` corresponden a la partícula `k`. Así, si se quiere saber la partícula a la que le corresponde una localidad ℓ, que se encuentra en la entrada `i` de alguno de los arreglos de localidades, se calcula `i`$/4$ y se obtiene su parte entera.

Las partículas que sólo se encuentran en una localidad, sus otras tres entradas asignadas son -1. Las que comparten otra localidad, sus dos últimas entradas asignadas son -1. Las que comparten cuatro localidades, tienen todas sus entradas asignadas distintas de -1.

Para encontrar todas las partículas que tienen una misma localidad y hacer más eficiente el algoritmo, los arreglos `Loc` y `Loc2` se ordenan. Sin embargo, como se necesita la ordenación que poseen por construcción para poder identificar a qué partícula corresponde cada localidad, no es posible hacer la modificación directa en los arreglos. Por ello, se usan arreglos de índices auxiliares. Estos arreglos se inicializan según lo siguiente

```
for (int i = 0; i < 4*Ntotales; ++i) {
    ind1[i] = i;
}
```

De tal forma que si se hace la composición de un arreglo con otro, se cumple que `Loc[ind1[i]] = Loc[i]`. Si se intercambia el elemento `r` por el que está en la posición `s`, la composición de arreglos para esos valores da `Loc[ind1[r]]=Loc[s]` y `Loc[ind1[s]]=Loc[r]`, es decir, el intercambio de los elementos de `ind1` genera el intercambio de posiciones cuando se considera la composición de arreglos `Loc[ind1[]]`. Con esto, el intercambio necesario para ordenar el arreglo `Loc` se lleva a cabo en `ind1`, y cuando aparezca la combinación `Loc[ind1[k]]` se entiende como si se usara directamente un arreglo de localidades ordenadas, por ejemplo, `LocOrd[k]`.

Para ordenar los arreglos de localidades, se usa la función `radixSort(...)`. Esta función toma el arreglo de localidades, el arreglo de índices, y un arreglo auxiliar de índices. También necesita un arreglo donde se registra el dígito considerado, y un arreglo más que registra el número de aparición de un dígito. De hecho, esta función es una modificación al algoritmo estándar. La única modificación es que trabaja con un arreglo de enteros que puede contener varios -1.

El algoritmo de radix sort es un algoritmo que dígito por dígito. Hay dos versiones. Una que empieza con los dígitos de órdenes superiores y va trabajando sucesivamente con los órdenes inferiores, y una que empieza con los dígitos de órdenes inferiores y va trabajando sucesivamente con los órdenes superiores. La idea del algoritmo es que se hace una ordenación parcial por dígito. La versión implementada trabaja primero con las unidades y avanza a órdenes superiores.

Empieza por inicializar los arreglos de índices usados. El siguiente paso es calcular el número de dígitos que tiene el número mayor del arreglo, que es conocido porque no hay más de `NX*NY-1` localidades. Luego, extrae el primer dígito de cada número. Estos los coloca en un arreglo para evitar hacer el cálculo de nuevo. Después, inicializa el arreglo `cuentaDigitos`, que es el encargado de poner cada elemento en su lugar. Este es de tamaño 11, en vez del típico tamaño de 10. Tiene un elemento más para guardar el número de apariciones de -1. A continuación, cuenta cada aparición de los dígitos en el arreglo `digitos` antes encontrado. Cuando encuentra -1, suma en la última posición del arreglo. Entonces, la última posición del arreglo da el número de negativos en el arreglo `Loc`. A `cuentaDigitos` le hace una suma prefija una vez ha terminado el paso anterior. Así, ahora el arreglo contiene la primer posición en que se coloca cada elemento del arreglo que está ordenando. Como ya se mencionó, el intercambio para la ordenación se lleva a cabo en el arreglo de índices. También, el intercambio necesita de un arreglo auxiliar, o se sobreescriben posiciones que luego se intercambian con el valor de la sobreescritura y no con el valor original que debían intercambiar. Antes de hacer el intercambio de punteros para los arreglos de índices, se copian los índices que contienen las posiciones de los elementos con -1. De esta forma, en los intercambios sucesivos, donde ya no se trabaja con esas posiciones, ya no se altera su orden. De no hacer este paso, en el paso siguiente del ciclo de ordenamiento, las últimas posiciones donde deberían estar los índices de los elementos con -1, están en el arreglo de índices `ind`, pero en el intercambio con `indAux`, el orden correcto lo guarda este. Si el ciclo terminara en ese paso, el orden correcto para esas posiciones está en `indAux`. El paso final del ciclo es el intercambio de arreglos de índices, como ya se mencionó. En la siguiente vuelta, haciendo el intercambio de índices para las últimas posiciones, se trabaja ahora con los dígitos en la segunda posición. Como los elementos con -1 ya no pueden ordenarse más, ya no se tocan y el arreglo se reduce en tamaño, lo que ayuda a hacer más eficiente el algoritmo. Todos los pasos siguientes son los mismos, con excepción de que ya no se trabaja con los -1. Una última característica importante del algoritmo se menciona. Aunque en el arreglo final quedan las localidades con un mismo valor juntas, la localidad correspondiente a la k-ésima partícula

y la localidad correspondiente a la $(k+n)$-ésima partícula, con un mismo valor, guardan un orden, tal que aparece primero la localidad para la partícula con índice menor, es decir, la k-ésima.

III.3.3. Ajuste de segmentos de recta a la frontera del dominio

Dos funciones se encargan de esta parte. La primera es `rectificaFrontera(...)`. Esta revisa todas las localidades de tipo 2 del dominio y revisa qué partículas tienen la misma localidad. La función `binarySearch(...)` se encarga de hacer la búsqueda. En caso de no encontrar la localidad, para el nodo asociado a la localidad de tipo 2, los valores de `pendientes[PL(i,j)]`, `ordenadasAlOrigen[PL(i,j)]` y `sentido[PL(i,j)]` se asignan a cero. Esto es importante para el paso de corrección. Esto es, porque un nodo frontera con `sentido[PL(i,j)]` igual a cero, es un nodo que en el esquema de localidades del tipo 2, no tiene partículas en la cercanía, y por lo tanto, no hay manera de hacer un ajuste de la frontera. Pero por ser un nodo de frontera, en el esquema de localidades del tipo 1 sí tiene partículas cerca de él.

En lo que sigue, para la función `rectificaFrontera(...)` se trabaja exclusivamente con localidades del tipo 2, por lo que sólo se refiere a ellas como localidades, sobreentendiendo que son de este tipo.

Para representar la frontera, una forma sencilla es ajustando un segmento de recta a las partículas que estén en la localidad. El número mínimo de partículas necesarias para identificar a un nodo como de frontera es uno. Por lo que si en la localidad, asociada a un nodo, existen partículas, la búsqueda binaria regresa el índice del arreglo `Loc2[ind2[]]` donde se encuentra la localidad. Para facilitar un poco la discusión, se considera un arreglo auxiliar con las localidades ordenadas. Sea este `Loc2Ord[]`. Entonces, la búsqueda binaria regresa, el índice `ix` donde al evaluar en el arreglo, está la localidad. Sin embargo, la búsqueda binaria regresa el primer elemento que encuentra con la localidad buscada. Como se busca todas las partículas con la misma localidad, se busca hacia atrás en `Loc2Ord` el primer elemento que tiene la misma localidad y también hacia adelante. Respecto del elemento con índice `ix`, el primer elemento con la misma localidad es el de índice `ix-rrizq` y el último elemento con la misma localidad es el de índice `ix+rrder`. Si `rrizq` y `rrder` son iguales a cero, quiere decir que la partícula se encuentra sola en la localidad. Si uno es igual a 1 y el otro es igual a cero, entonces hay dos partículas en la localidad. Si los dos son distintos de cero, hay más de 2 puntos en la localidad.

Si `N` cuenta el número de partículas en la localidad, cuando `N = 1`, con sólo esa partícula no se puede obtener una rectificación de la frontera. Entonces, se consideran las dos partículas vecinas en la pared, y a partir de ellas se obtiene la pendiente de una recta, que se impone, pase por el punto en la localidad. Si `N = 2`, la recta para ajustar la frontera se obtiene con la ecuación de la recta que pasa por dos puntos. Si `N >2`, la mejor recta que aproxima a ese conjunto de puntos se obtiene con el método de mínimos cuadrados. Para cualquiera de estos casos, hay un caso especial que se trata de forma distinta. Cuando la pendiente de la recta es muy grande, correspondiente a una recta vertical, la ordenada al origen carece de sentido. Sin embargo, para describir ese tipo de rectas, además de su pendiente, se necesita en qué abscisa está. Para no perder esa información, en estos caso, la ordenada al origen `b` guarda la abscisa por donde pasa la recta.

Hay otros detalles importantes a cuidar. Cuando la primer partícula o última se encuentran solas en una localidad, la partícula anterior a la primera es la última del arreglo, y la partícula siguiente a la última es la primera. También, si la primer y última partícula se encuentran en la misma localidad, la última se considera que precede a la primera. Recuérdese que el arreglo `Loc2Ord` guarda de todas maneras un orden relativo entre localidades con el mismo valor. Por lo que es perfectamente diferenciable cuál es

la primer partícula y cuál es la última. En general, esto también ayuda para ver qué partícula precede a otra. La importancia de esto es que con esta información se construye en qué sentido se recorre la frontera, y eso ayuda para determinar qué punto es interior y cuál es exterior.

La forma de obtener el sentido en que se recorre la frontera es tomar la primer partícula en la localidad y la última, y restar la coordenada x de la primera a la segunda, y según el signo es el sentido asignado. 1 si es positivo y -1 si es negativo. Si las partículas tienen la misma coordenada x, se resta la coordenada y de la última partícula a la de la primera. El arreglo `sentido[]` guarda esto de acuerdo con cada nodo en el dominio total. Por supuesto, los únicos lugares donde se escribe algo es en las posiciones de los nodos frontera.

Para terminar la explicación de la implementación de esta función, se recuerda que para obtener qué partícula se está trabajando, se divide entre cuatro el índice del arreglo de localidades. En este caso el índice es `ind1[ix]` para la primer localidad encontrada con la búsqueda binaria, `ind1[ix-rrizq]` para la primer partícula con la misma localidad y `ind1[ix+rrder]` para la última partícula con la misma localidad. Todos los elementos de `ind1` entre el elemento `ix-rrizq` y el elemento `ix+rrder` son partículas con la misma localidad. Entonces la partícula correspondiente se encuentra con `ind1[k]/4`, donde `k` toma valores entre `ix-rrizq` y `ix+rrder`.

Ahora, como ya se mencionó antes, el análisis por localidades del tipo 2 sólo logra asignar `pendientes[]`, `ordenadasAlOrigen[]` y `sentido[]` a ciertos nodos frontera. La función `correccionANodosNoContemplados(...)` se encarga de corregir eso.

`correccionANodosNoContemplados(...)` recorre todas las localidades de tipo 1 que tienen todas las partículas. Por supuesto, para evitar repetir la revisión de nodos, el arreglo se recorre en orden, y una vez termina con una localidad avanza a la siguiente.

Lo primero que hace es tomar una localidad de tipo 1 y avanza en el arreglo para ver cuántas partículas comparten la misma localidad. Esta información sirve, por una parte, para indicar la cantidad de partículas que considera uno de estos nodos frontera no trabajados, y por otra, para avanzar a la siguiente localidad con valor diferente.

Con la localidad de tipo 1, se obtiene el nodo asociado a ésta (el de la esquina superior izquierda). Por ejemplo, es el `(m,l)`, y se obtiene con

```
l = pos/NX;
m = pos - l*NX;
```

`pos` es la localidad trabajada, y `NX` es el número de nodos en la dirección horizontal. En este caso, todas las variables trabajadas son enteras. La razón de estas fórmulas es que la localidad `pos` se expresa con sus índices de la forma `pos = l*NX + m`. Al tomar la división entre `NX` se descarta `m/NX` pues `m` es un número entre 0 y `NX-1`. Conocido el índice `l`, se resta `l*NX` para obtener `m`.

Conocidos estos índices, se sabe que los nodos que rodean la localidad son `(m,l)`, `(m+1,l)`, `(m+1,l+1)` y `(m,l+1)`, por lo que esos son los candidatos a corregir. Un nodo de estos, no trabajado, tiene dos características. Una, que el valor del arreglo `sentido[]` para el nodo es cero, y que la localidad del tipo 2 con los mismos índices del nodo, no posee partículas. La razón de revisar ambas propiedades es que un nodo está en cuatro localidades del tipo 1, y para cada una de ellas se puede asociar una pendiente, ordenada al origen y sentido de la frontera rectificada. Así, a pesar de que se le haya asignado previamente una pendiente, ordenada al origen y sentido, se sigue analizando si el segmento de recta ajustado a la frontera en otra localidad vecina es mejor. Se piensa como un ajuste mejor si el segmento de recta tiene una distancia menor al nodo frontera.

Además de la pendiente, ordenada al origen y sentido de la frontera, se corrigen las condiciones de frontera. Para una localidad dada, estas variables se calculan una sola vez, siempre que sea necesario. Esta es la razón de la bandera `estaCalculado`, y que el cálculo de estas se lleve a cabo en una función dentro de cada condicional para cada uno de los cuatro nodos de la localidad. La función se llama `correcciones(...)` y calcula la pendiente, ordenada al origen y sentido de las partículas en la localidad de tipo 1, de manera totalmente análoga a la función `rectificaFrontera(...)`. También calcula las condiciones a la frontera, impuesta por un arreglo que contiene las velocidades de las partículas en la frontera. La forma en que trabaja, es considerando los índices de las partículas móviles. Si alguno de los índices de la localidad se encuentra entre el rango de partículas móviles establecido, la velocidad de la partícula correspondiente se suma. Al final, se divide la suma de velocidades de todas las partículas en la localidad con índices en el rango permitido, entre el número de partículas móviles de la localidad.

III.3.4. Identificación de los tipos de nodo del dominio

Para la identificación de los tipos de nodo del dominio, se usan tres funciones. La primer función identifica y marca los nodos frontera según el esquema de localidades de tipo 1. Esta es la función `identificaNodosFrontera(...)`. La siguiente función, usa la información de cuáles son los nodos frontera para hacer una asignación más eficiente. Esta es `identificaTipoDeNodos(...)`. La última función, marca los nodos según la configuración 3 descrita en el capítulo II. Esta es la función `marcaNodos(...)`.

El funcionamiento de la primer función es simple. Se recorren las localidades de tipo 1 en orden, omitiendo las localidades ya trabajadas. Los nodos que delimitan la localidad son los nodos frontera. Se marcan con -1, en un arreglo `M`, para que no haya confusión con posibles marcados anteriores. Por supuesto, el arreglo `M` que guarda el tipo de cada nodo debe ser inicializado desde el principio del programa, y de hecho, basta con esa vez, a pesar de que pueda ser móvil la frontera del dominio. La razón es que sólo esta función marca con -1 los nodos frontera. De la búsqueda de localidades, por supuesto se omiten las últimas entradas del arreglo `LocOrd`, pues ahí es donde quedaron los -1 indicando que no había más localidades asignadas a las partículas.

La segunda función, `identificaTipoDeNodos(...)` trabaja determinando el tipo de cada nodo con índice vertical j. Es decir, los nodos en un mismo renglón en el arreglo rectangular de nodos. Al principio de cada rengón, la bandera `MarcaEncontrada` se inicializa apagada. Esta bandera señala si se ha encontrado un nodo de frontera, previamente marcado con -1. Si esta bandera está apagada, automáticamente el tipo del nodo se asigna a cero, que indica que el nodo es un nodo sólido de bulto. Esta bandera se prende cuando se detecta un nodo frontera. Así, para todos los nodos en una línea, antes de encontrar un nodo frontera, su tipo se marca con 0. Una línea sin nodos frontera se marca completamente con ceros.

En el momento en que se encuentra un nodo frontera, se prende la bandera `MarcaEncontrada` y el comportamiento de identificación de nodos cambia. Para el nodo frontera, se identifica su tipo por medio de la función `identificaTipoNodosFrontera2()`. Esta es una función que regresa únicamente los valores 2 o 3. Un valor 2 significa que el nodo es un nodo frontera sólido. Un valor 3 significa que el nodo es un nodo frontera fluido. El funcionamiento de esta función se explica más adelante.

Después de encontrar el primer nodo frontera, la bandera `MarcaEncontrada` está prendida y los nodos siguientes se trabajan de la siguiente forma. Si el nodo revisado es nodo frontera, la identificación se vuelve a hacer por medio de la función `identificaTipoNodosFrontera2()`. Esto sigue hasta que se encuentra un nodo que no sea de frontera. Para ese nodo, la asignación de tipo depende del nodo

anterior, que es de frontera. Si el tipo del nodo anterior es 2, quiere decir que el nodo es un nodo sólido de bulto. si el tipo del nodo anterior es 3, el nodo es un nodo fluido de bulto, que se identifica con el valor 1. Por construcción, un nodo de frontera sólido (2) sólo tiene como vecinos a otros nodos de frontera sólidos, a nodos de frontera fluidos (3) o a nodos sólidos de bulto (0). Un nodo de frontera fluido (3), sólo tiene como vecinos a otros nodos frontera fluidos, nodos de frontera sólidos o nodos fluidos de bulto (1). Los nodos que siguen tienen el mismo tipo ya establecido, excepto si se encuentra de nuevo un nodo frontera, donde hay que volver a hacer la identificación de tipo, para luego seguir haciendo la identificación de los nodos faltantes, de la misma forma. La operación de identificación de tipo es la más costosa, así que sólo se usa para los nodos frontera.

Respecto de la función `identificaTipoNodosFrontera2()`, su funcionamiento está basado en una propiedad geométrica, que se atribuye a dominios en $\mathbb{R}^2$ a los que se les asigna un curva frontera, que se recorre de una manera específica (véase figura III.6). Esta propiedad geométrica es la orientabilidad de un área. Se dice que un área está orientada positivamente si su curva frontera se recorre en sentido antihorario, o positivo, y que está orientada negativamente si su curva frontera se recorre en sentido horario o negativo.

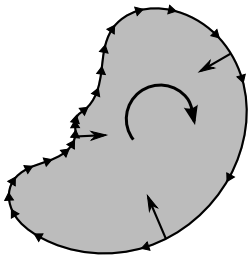


Figura III.6: *Curva frontera de un dominio en* $\mathbb{R}^2$.

Cuando la curva se recorre en sentido horario, si se toma un vector tangencial a la curva en un punto $\vec{\mathbf{x}}$, de tal forma que siga el sentido horario, cualquier punto interior al dominio en la localidad de $\vec{\mathbf{x}}$, queda a la derecha de la frontera; cualquier punto en el exterior al dominio, en la localidad de $\vec{\mathbf{x}}$, queda a la izquierda. Así, si se determina de qué lado, respecto de la curva frontera, queda el nodo frontera a identificar, se identifica su tipo. El signo de un producto cruz de hecho es el que ayuda a determinar de qué lado. Si además del vector tangencial $\vec{\mathbf{v}}_t$ en el punto $\vec{\mathbf{x}}$, se toma un vector que vaya de ese punto al nodo a identificar, sea este $\vec{\mathbf{v}}_N$,el signo del producto cruz $\vec{\mathbf{v}}_t \times \vec{\mathbf{v}}_N$, de ser positivo, nos indica que el nodo está en el exterior, y por lo tanto, es un nodo frontera sólido; si es negativo, el nodo es interior, y es un nodo frontera fluido.

Sin embargo, obtener un nodo en la recta que ajusta localmente a la frontera, es una operación un tanto costosa computacionalmente.

La información necesaria para generar el vector tangencial a la curva, que puede ser de cualquier tamaño, ya se posee. Porque se tiene la pendiente y ordenada al origen de la recta tangencial a la frontera. Además, se tiene el sentido en que se recorre la curva para cada nodo. De hecho, con excepción de cuando la pendiente es infinita, si `a` es la pendiente, y `b` su ordenada al origen, un vector tangencial a la frontera con el sentido `s` es `s(1,a)`. El segundo vector, el que determina la posición del nodo, se construye de acuerdo al sentido `s` que tiene la curva en el punto cercano al nodo. Si la curva se recorre de izquierda a derecha (`s=1`), se extiende el segmento de recta que ajusta a la frontera hasta $x = 0$ (véase figura III.7). El punto de intersección es el `(0,b)`. Un vector que parta de este punto al nodo, que está en `(Px(i),Py(j))`, es el `(Px(i)-0,Py(j)-b)`. El signo del producto cruz de `(1,a)` con `(Px(i),Py(j)-b)`

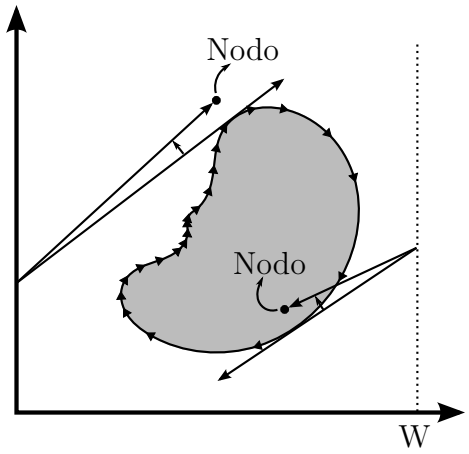


Figura III.7: *Vectores tangenciales a la curva frontera de una región en* $\mathbb{R}^2$.

establece el tipo del nodo frontera. Si la curva se recorre de derecha a izquierda (`s=-1`), la extensión del segmento que ajusta la frontera se hace a la derecha, hasta, por ejemplo $x = W$, con W el ancho del dominio (véase figura III.7). Ahora el punto de intersección es el `(W,aW+b)`. El vector de este punto al nodo es `(Px(i)-W,Py(j)-aW-b)`. El signo del producto cruz de `-(1,a)` con `(Px(i)-W,Py(j)-aW-b)` determina el tipo de nodo frontera. De hecho, cuando `s=1` el signo se determina calculando `Py(j) - b - a*Px(i)`, y cuando `s=-1`, el signo se determina calculando `-Py(j) + b + a*Px(i)`. Nótese que el signo en la segunda expresión no depende del valor escogido para `W`. Además, las expresiones son iguales con excepción del signo.

En el argumento del párrafo anterior, se usa que el signo del producto cruz de un vector, que lleva la dirección de una recta, con otro vector, que va de un punto cualquiera en la recta a un punto fijo en el espacio, siempre es el mismo. Particularmente, el punto cualquiera, puede ser el punto de tangencia de la recta (que es la recta ajustada a la frontera), y el punto fijo en el espacio, el nodo frontera.

Si el valor absoluto de `Py(j) - b - a*Px(i)` es muy cercano a cero, entonces se considera a los vectores colineales, y el tipo de nodo se asigna por convención como 3 (nodo frontera fluido), pues el nodo queda sobre la frontera.

Si la pendiente `a` es la asignada a una recta vertical (o mayor), la recta es vertical. En este caso, si el signo `s=-1`, y el nodo queda a la derecha, el nodo es exterior, y por lo tanto, es un nodo de frontera sólido; si queda a la izquierda, el nodo es de frontera fluido. Cuando `s=1`, la situación se invierte. Si el nodo queda sobre la frontera, de nuevo se asigna el tipo de nodo a 3.

Se advierte que este método presupone que las rectas ajustadas a la frontera son aproximadamente tangenciales a la frontera.

La función `marcaNodos()` hace la selección de los nodos más cercanos a la frontera. Para lograr esto, por cada localidad de tipo 1 donde haya partículas, ajusta una recta al conjunto de puntos en ella, de forma análoga a la función `rectificaFrontera(...)`. Con la pendiente y ordenada al origen, se procede al marcado de nodos. Para toda la descripción de esta función, se usa únicamente las localidades en el esquema tipo 1, por lo que el tipo de la localidad se sobreentiende de aquí en adelante.

La recta ajustada intersecta a la frontera de la localidad de 6 maneras diferentes (véase la figura III.8). Se llama y_a a la ordenada de la intersección de esta recta, con una recta vertical que pase por el nodo (l, m). y_b es la ordenada de la intersección de esta recta con una recta vertical que pase por el nodo $(l + 1, m)$. x_c la abscisa de la intersección de la recta ajustada con la recta horizontal que pasa por el

nodo (l, m). Y finalmente, x_d es la abscisa de la intersección de la recta ajustada con una recta horizontal que pasa por el nodo $(l, m+1)$. Cuando la recta ajustada es vertical, únicamente intersecta a las rectas horizontales que pasan por los nodos (l, m) y $(l, m+1)$. Cuando la recta es horizontal, únicamente tiene intersección con las rectas verticales que pasan por los nodos (l, m) y $(l+1, m)$. Estos casos se consideran aparte del caso más general.

Así, para el caso cuando la recta es vertical (figura III.9b), para hacer el marcado de nodos, como la recta equidista de las parejas de nodos (l, m), $(l, m+1)$ y $(l+1, m)$, $(l+1, m+1)$, la función marca una pareja u otra. Marca la que esté más cercana a la recta. Si la recta ajustada es horizontal (figura III.9a), ahora las parejas de nodos son (l, m), $(l+1, m)$ y $(l, m+1)$, $(l+1, m+1)$. De nuevo, se marca la pareja de nodos que estén más cercanas.

En el caso más general, y_a, y_b, x_c y x_d todas existen. Sin embargo, estas ordenadas y abscisas de intersección no siempre corresponden a intersecciones en las fronteras de la localidad. Pero lo que sí se garantiza, como la recta ajustada está contenida en la localidad, es que al menos un par de estas corresponda a intersecciones de la frontera de la localidad. Por ejemplo, en la figura III.8a, las abscisas de intersección y_a y y_b corresponden con puntos de intersección con la frontera. Entonces, basta con ver cuáles de los valores y_a, y_b, x_c y x_d quedan entre los nodos.

Las combinaciones posibles en que y_a, y_b, x_c y x_d quedan entre los nodos son: $[y_a,y_b]$, $[x_c,x_d]$, $[y_a,x_c]$, $[y_a,x_d]$, $[y_b,x_c]$ y $[y_b,x_d]$. Los casos $[y_a,y_b]$ y $[x_c,x_d]$ de hecho pueden ser ambos válidos en el caso en que la recta vaya de nodo a nodo opuestos por la diagonal. En estos casos, los nodos marcados son los que pasan por los nodos. En los demás casos, $[y_a,x_c]$, $[y_a,x_d]$, $[y_b,x_c]$ y $[y_b,x_d]$, si la recta ajustada pasa por algún nodo, este se marca. En general, se marca el nodo más cercano al punto de intersección de la recta con la localidad. Ahora, la configuración 3, la que requiere los nodos más cercanos a la frontera, requiere que no haya huecos, o en otras palabras, que haya suficientes nodos marcados para que no sea posible pasar de un nodo sólido de bulto a un nodo fluido de bulto sin pasar por un nodo de frontera marcado. Cuando la recta ajustada es vertical u horizontal, esto se logra automáticamente. Cuando la recta es oblicua, si dos nodos opuestos por la diagonal no están marcados, uno puede ir de uno de los nodos al otro por la diagonal que los une. Si $a > 0$, se necesita que alguno de los nodos (l, m) o $(l+1, m+1)$ esté marcado. Si $a < 0$, se necesita que alguno de los nodos $(l+1, m)$ o $(l, m+1)$ esté marcado. De hecho, entre las parejas de nodos mencionadas, se debe marcar el que tenga más cercana la recta. Para los casos $[y_a,x_c]$, $[y_a,x_d]$, $[y_b,x_c]$ y $[y_b,x_d]$, la elección es directa (véase las figuras III.8c, III.8d, III.8e y III.8f), porque cualquier punto del segmento de recta que queda en la localidad, queda más cerca de ese nodo. Sin embargo, hay una situación donde es posible que haya dos nodos opuestos por la diagonal que no estén marcados y no represente un problema para la configuración. Esta situación se da, en los casos $[y_a,x_c]$, $[y_a,x_d]$, $[y_b,x_c]$ y $[y_b,x_d]$, cuando los puntos de intersección de la recta ajustada quedan más cerca de un mismo nodo.

Falta por determinar qué nodo marcar cuando la recta de ajuste es oblicua e intersecta en uno de los casos $[y_a,y_b]$, $[x_c,x_d]$. Considere la figura III.10. Los nodos marcados son $(l+1, m)$ y $(l, m+1)$. Se necesita marcar alguno de los nodos (l, m) o $(l+1, m+1)$. Sea y_c la altura del centro de la localida y y_m la altura que tiene la recta a la mitad de la localidad. Si $y_m < y_c$ la recta queda más cerca del nodo $(l+1, m+1)$, y si $y_m > y_c$ la recta queda más cerca del nodo (l, m). Por lo tanto, la elección de qué nodo marcar se hace con este método.

El método de marcado se discute a continuación. Siendo que los tipos de cada nodo se guardan en el arreglo `M`, con números del 0 al 3, para no usar otro arreglo más, el marcado se hace con otros números. La convención seguida permite convertir un nodo frontera sólido o fluido a un nodo de bulto simplemente restándole 2 al valor del tipo que tiene. Pero la opción de convertir a nodo de bulto uno de frontera se

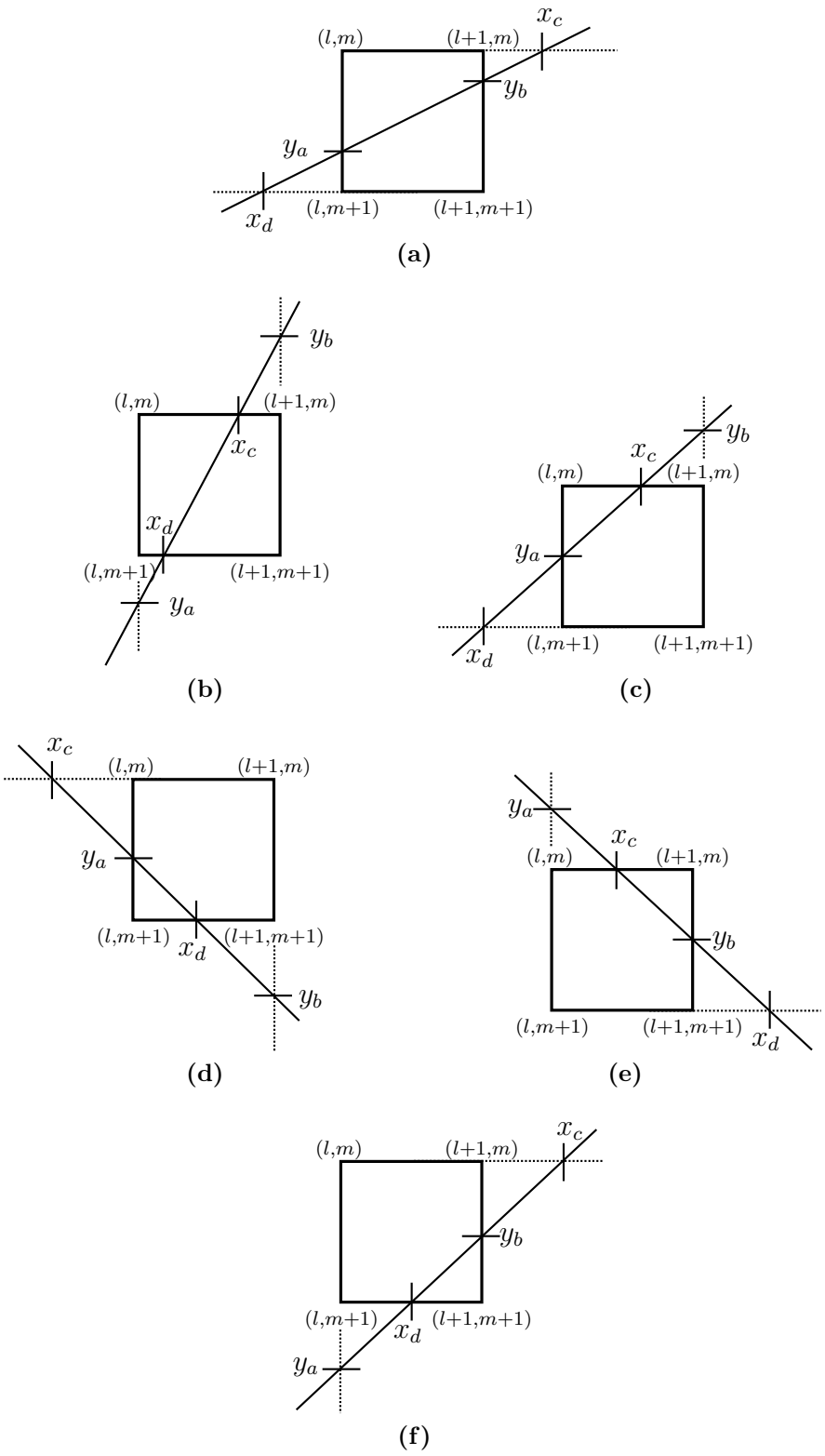


Figura III.8: *Posibles intersecciones de la recta de ajuste con las fronteras de una localidad. Para todos los casos, la recta es oblicua. (a) La recta intersecta a izquierda y derecha a alturas distintas. (b) Intersecta arriba y abajo con distancias horizontales distintas. (c) Intersecta arriba y a la izquierda. (d) Intersecta a la izquierda y abajo. (e) Intersecta arriba y a la derecha. (f) Intersecta a la derecha y abajo.*

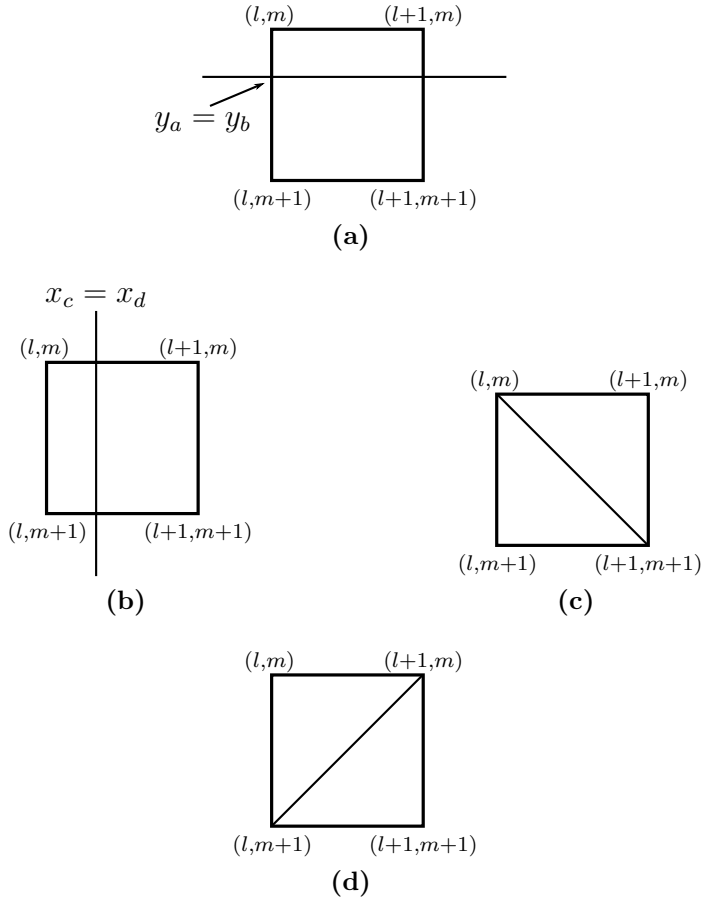


Figura III.9: *Posibles intersecciones de la recta de ajuste con las fronteras de una localidad, para casos especiales. (a) La recta intersecta a izquierda y derecha a la misma altura. (b) Intersecta arriba y abajo con distancias horizontales iguales. (c) y (d) Intersecta en los vértices de la localidad.*

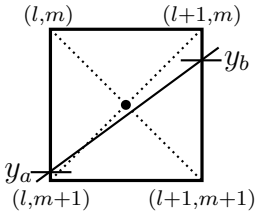


Figura III.10: *Cuando la recta de ajuste es oblicua, e intersecta a la localidad a la izquierda y a la derecha, o arriba y abajo, se marca otro nodo. El criterio para determinar qué nodo marcar se basa en comparar la altura del centro de la localidad con la altura que tiene la recta a la mitad de la localidad. En el caso particular mostrado en la figura, como la recta queda por debajo del centro de la localidad, esta queda más cerca del nodo* $(l+1, m+1)$ *y por lo tanto se marca.*

descarta, porque otra localidad puede requerir que un nodo de estos sea marcado, y si es desmarcado, se hace de bulto. Así que, lo que se hace, es sumar 2 a cada nodo que se quiera marcar, siempre que no se haya marcado antes. Es decir, un nodo frontera sólido marcado tiene el valor 4, y un nodo frontera fluido tiene el valor 5. Al final de marcar los nodos por los métodos descritos, si se resta 2 a los nodos no marcados, estos adquieren el tipo de los nodos de bulto, y si se resta dos a los nodos marcados, estos adquieren el valor 2 o 3. De esta forma, se obtienen los nodos que requiere la configuración 3.

A partir de la configuración 3, las otras dos configuraciones son muy fáciles de encontrar con lo que ya

se tiene construido. De hecho, la modificación para obtener los nodos frontera sólidos es muy simple. En vez de aplicar la función `marcaNodos(...)`, simplemente se regresan al valor 1 los nodos frontera marcados con 3. El caso cuando se desee escoger los nodos frontera fluidos, requiere una convención como el caso de la configuración 3, que quizá tenga mucho más peso aquí. La convención a establecer es, cómo etiquetar un nodo que caiga sobre la pared del dominio. También, si estos se etiquetan como nodos frontera fluidos, se debe establecer qué pasa con los nodos frontera fluidos alrededor, porque un enfoque es pensar que estos ya no son de frontera, ya que la pared se encuentra "un poco más allá" de los nodos frontera fluidos que caen sobre ella.

III.4. Dominios más generales

Como se mencionó en III.3.1, el programa se puede extender a regiones más generales que no son y-simples. La aplicación a regiones x-simples es casi inmediata. Recuérdese que un dominio en $\mathbb{R}^2$ es x-simple si al trazar una recta horizontal cualquiera, (con excepción de los extremos del dominio en y), únicamente intersecta a la frontera del dominio en dos puntos, o visto de otra forma, la región está entre dos funciones de y. Así, si el dominio a trabajar es una región del tipo x-simple, lo único que hay que hacer es rotar la imagen de entrada 90 grados.

De hecho, la limitación mencionada sólo se da cuando la entrada al programa es una imagen. Si las posiciones de la frontera, y órden en que se recorre la frontera están dadas, el programa no tiene problema.

La razón de la limitación es la forma en que se hace el reconocimiento de la frontera. Una posición de un punto en la frontera superior se asigna, y a continuación se asigna uno en la frontera inferior. Sin embargo, cuando el dominio es como el mostrado en la figura III.11, los puntos inmediatamente abajo no son necesariamente los de la curva en su frontera inferior. Así, la función `generacionDeParticulasEnPared(...)` asigna un punto de la frontera superior como si fuera uno de la frontera inferior.

En la práctica, la región está dada, y en apariencia puede parecer del tipo y-simple. Más aún, la región interior podría tener pixeles blancos, o islas de pixeles blancos. Para evitar estos dos problemas, se hace un paso de preprocesamiento al mapa de bits en el que, para cada columna, se encuentra el primer uno en la frontera superior y el último en la frontera inferior. Todas las posiciones intermedias en el mapa de bits se hacen uno. Este paso se hace por fuerza bruta, porque de nuevo, para cada columna, no hay una estimación de dónde pueda estar el primer uno o último en el mapa de bits.

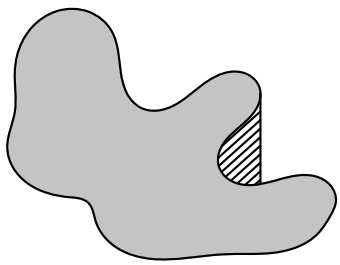


Figura III.11: *Región en $\mathbb{R}^2$ que no puede ser vista como región de tipo y-simple.*

Con este paso de preprocesamiento, si la región dada al programa es como la de la figura III.11, la región sombreada queda considerada como parte de la región.

Si, en definitiva, la región que se quiere trabajar es como la de la figura III.11, entonces es necesario seccionar la región en tres subregiones del tipo y-simple (véase la figura III.12). El seccionamiento de la región debe dejar cuatro puntos en común entre una subregión, que se considera como la primaria, porque es la que se une con las otras, y las otras dos subregiones, las secundarias, compartiendo así dos puntos con cada una. Esto, para reconocer precisamente dónde se unen las subregiones. Para la subregión primaria, llamada $\mathcal{A}$, se encuentra todos sus puntos frontera. Para las subregiones secundarias, $\mathcal{B}$ y $\mathcal{C}$, del conjunto de puntos en la frontera, se substrae la frontera común con $\mathcal{A}$, con excepción de los cuatro puntos mencionados anteriormente. En lo sucesivo, se entiende por frontera de $\mathcal{B}$ o $\mathcal{C}$ al conjunto resultante.

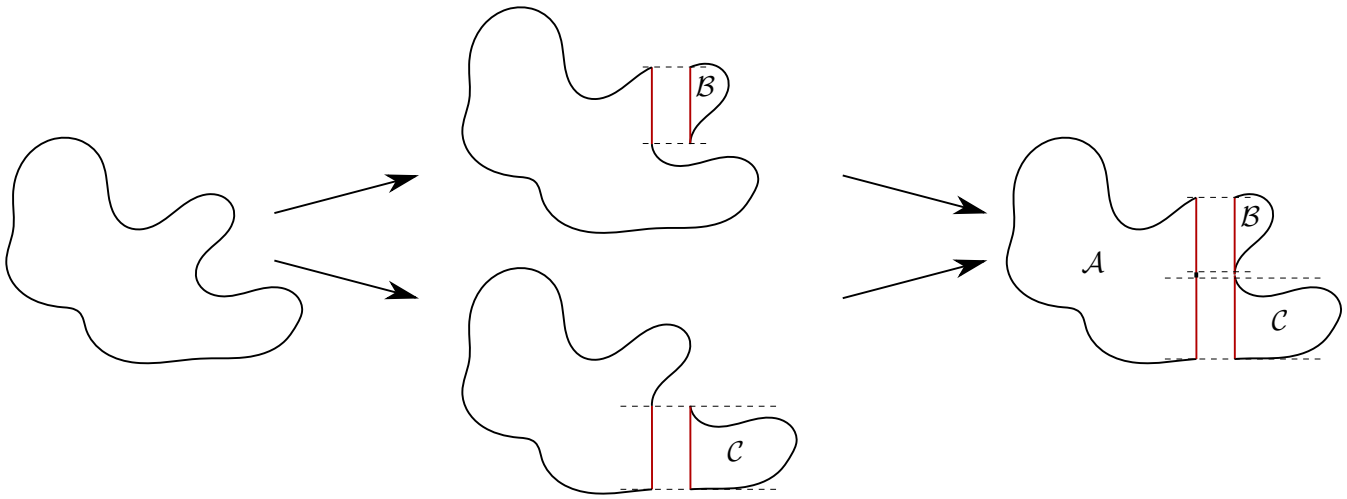


Figura III.12: *Seccionamiento de dominio en $\mathbb{R}^2$ en dos subregiones del tipo $y-$simple.*

La reconstrucción de la unión de las fronteras, es como sigue. Para una subregión, por ejemplo la $\mathcal{B}$, se toma el primer punto de la frontera de $\mathcal{B}$. Este corresponde con uno de los dos puntos que se evitó substraer de la frontera común entre $\mathcal{A}$ y $\mathcal{B}$. A continuación se busca en la lista de puntos de frontera de la subregión $\mathcal{A}$. El segundo punto común, el otro que no se substrajo de la lista de puntos de la frontera, se busca también en la lista de puntos de frontera de la subregión $\mathcal{A}$. De hecho, es el último punto de la frontera de $\mathcal{B}$. Todos los puntos intermedios a estos se eliminan y en su lugar se colocan (en orden) los de la frontera de $\mathcal{B}$. Por supuesto, los puntos en común se escriben una sola vez. Esto mismo se aplica con la subregión $\mathcal{C}$.

Las fronteras de $\mathcal{A}$, $\mathcal{B}$ y $\mathcal{C}$ deben escribirse de tal forma que tengan la misma orientación. Para $\mathcal{B}$ y $\mathcal{C}$ esto implica que, según de qué lado de la región original se extrajeron, es el orden en que se escriben sus arreglos de coordenadas. Recuérdese que cuando se habla de las fronteras de $\mathcal{B}$ y $\mathcal{C}$, las curvas frontera están abiertas por la substracción mencionada anteriormente. Por ejemplo, si la extracción se hizo del lado derecho, en los arreglos de coordenadas se escribe primero la frontera superior, luego la pared derecha y al final la frontera inferior. Si la extracción se hizo del lado izquierdo, primero se escribe la frontera inferior, luego la pared izquierda y finalmente la frontera superior.

Como se mencionó anteriormente, todo esto no se implementa. Sin embargo, de la discusión anterior, se observa que la restricción original impuesta al programa, de que trabaje exclusivamente con regiones del tipo y-simple, es un paso importante para un programa que trabaje regiones más generales.

Ahora, ninguna de las regiones discutidas anteriormente tiene fronteras interiores o "huecos". La figura III.13 muestra un dominio con huecos.

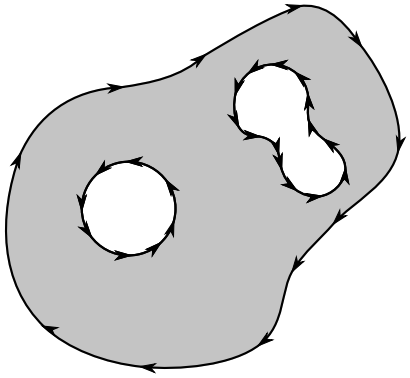


Figura III.13: *Región en $\mathbb{R}^2$ con "huecos".*

En realidad, incorporar huecos (de tipo y-simples) a la región es sencillo, dado todo lo que ya se tiene. Sólo hay que hacer unas cuantas modificaciones. Las modificaciones son sólo dos.

Antes de discutir las modificaciones, se considera la modelación típica de una región en $\mathbb{R}^2$ con hoyos, para entender por qué se hacen.

Hasta el momento, se ha discutido que la curva exterior sigue una cierta orientación. Que con esa orientación, la región interior se entendía que está "siempre a la derecha". Sin embargo, si la orientación de la curva se invierte, la región interior queda ahora "siempre a la izquierda". Pero para un hueco, esto ocurre al revés. En los huecos de la figura III.13, si se sigue la orientación de la curva mostrada, la región interior está "siempre a la derecha", y si se invierte la dirección en que se recorre la curva, la región interior está "siempre a la izquierda". Al considerar la frontera exterior y la interior a la vez, si la frontera exterior tiene la orientación prescrita previamente, y la frontera interior tiene una orientación invertida, al recorrer las dos fronteras en esas orientaciones, la región interior está "siempre a la derecha" de la frontera exterior, y "siempre a la derecha" de la frontera interior.

Esto sugiere que para asignar tipo a los nodos frontera cercanos a la frontera interior, lo único que se tiene que hacer es invertir de signo el arreglo `sentido[]`, correspondiente a invertir el sentido en que se recorre la curva.

La segunda modificación se realiza en la función `identificaTipoDeNodos(...)`. Esta función determina el tipo de cada nodo por renglón. El tipo que establece a los nodos, cuando no ha encontrado ningún nodo frontera, es cero. Ahora se pide que el tipo se establezca de acuerdo con un número Booleano. Este número Booleano es 0 si la frontera corresponde a una frontera exterior, y 1 si la frontera corresponde a una frontera interior.[4]

Para una misma curva, estas modificaciones tienen por efecto la inversión de tipos para los nodos. Es decir, lo que antes era 2, después del cambio es 3 y viceversa, y lo que antes era 0, ahora es 1, y viceversa.

Cuando se contemplan las dos curvas a la vez, la región interior a la curva interior debe tener ceros, la región intermedia entre la curva interior y la exterior debe tener unos, y la región exterior a la frontera exterior debe contener ceros. También, los nodos frontera de la frontera interior, para la región interior, deben ser 2, para la región entre la frontera interior y la exterior deben ser 3, y para la región exterior a la frontera exterior deben ser 2.

Se observa ahora lo siguiente. Si se multiplica el tipo de un nodo sólido de bulto (0) por el tipo de un

[4]Por convención, la orientación dada a la curva exterior es negativa. Para la interior, es positiva. Con esto, el número Booleano mencionado recibe el nombre `OP`, que indica si la curva tiene orientación positiva.

nodo fluido de bulto (1), se obtiene cero. Si se multiplica el tipo de un nodo fluido de bulto por un nodo de cualquier otro tipo, se obtiene el tipo del otro nodo.

Si entonces se multiplica el tipo asignado en una configuración, por el tipo del nodo correspondiente en la otra configuración, el tipo de los nodos de bulto más interiores es cero, el tipo de los nodos de bulto exteriores también, el tipo de los nodos de bulto en la región intermedia es uno, y los tipos de las fronteras se quedan sin alterar, porque para la otra configuración el nodo correspondiente tiene tipo 1. Este resultado es precisamente el que se busca.

Además del arreglo que guarda los tipos de nodos, se modifican los arreglos `sentidos[]`, `pendientes[]` y `ordenadasAlOrigen[]`. Los nuevos arreglos se obtiene sumando entre sí los arreglos de cada configuración. De hecho, los arreglos de `pendientes[]` y `ordenadasAlOrigen[]` pueden ser exactamente los mismos usados para ambas curvas.

Se resume ahora todo el proceso de la obtención de los distintos arreglos que caracterizan la frontera. El método de condiciones de frontera descrito en el capítulo II trabaja con esta información.

III.4.1. Resumen de la obtención de arreglos que caracterizan la frontera

Existen dos enfoques iniciales. En uno, las posiciones de las partículas en la frontera están dadas. En el otro, se construyen a partir de la imagen. Para el segundo, primero se obtiene el mapa de bits de la imagen, se preprocesa la imagen (para limpiar impurezas o modificar a conveniencia las pequeñas inconsistencias para el uso del programa), se usa la función `conteoParticulasPared(...)` para contar el número de partículas que se colocan por defecto en las paredes superior, inferior, izquierda y derecha del dominio. Finalmente, con la información obtenida, se usa `generacionDeParticulasEnPared(...)`, que genera las posiciones de las partículas en la frontera.

A partir de las coordenadas de los puntos frontera, se identifica la localización de cada una en dos configuraciones de localidades distintas. Para esto se usan las funciones `identificaLocalidadesDeParticulasConfiguracion1(...)` e `identificaLocalidadesDeParticulasConfiguracion2(...)`. Los arreglos de localidades obtenidos se ordenan con la función `radixSort(...)`.

A continuación, se rectifica la frontera con `rectificaFrontera(...)`. Para terminar la rectificación de la frontera, se aplica un paso de corrección que añade pendientes, ordenadas al origen y sentido a nodos frontera que se pasaron por alto.

Con las localidades de las partículas frontera, se identifica los nodos frontera con `identificaNodosFrontera(...)`. Luego, con la información de la rectificación de la frontera, y con la variable Booleana `OP` fijada a 0, se establece el tipo de todos los nodos con `identificaTipoDeNodos(...)`. A continuación, se escogen los nodos a usar por la configuración 3. De esto se encarga la función `marcaNodos(...)`.

Si se desea agregar un hueco, todos los pasos anteriores se repiten, pero la variable Booleana `OP` se fija a uno, y al arreglo que guarda los sentidos en que se recorre la curva del hueco, se le invierten los signos. El arreglo final con las dos curvas frontera se obtiene multiplicando entre sí los arreglos que guardan los tipos de nodo para cada tipo de curva. El arreglo de pendientes, ordenadas al origen y sentido se obtienen sumando entre sí estos mismos arreglos de la curva frontera exterior y la interior.

III.5. Asignación de condiciones de frontera

La asignación de las condiciones de frontera se hace por medio de tres funciones. Una de estas asigna velocidades a las partículas de la frontera, y las otras dos se encargan de encontrar y promediar las velocidades para partículas en la misma localidad.

La primera se llama `inicializaVelocidadesDeParticulasMoviles(...)`. En esta, se asignan las componentes de las velocidades de las partículas en dos arreglos, uno para la componente x y otro para la componente y. Recuérdese que las posiciones de las partículas en la frontera se registran de una forma específica en dos arreglos, uno para la componente x y otro para la componente y, empezando por las posiciones de las partículas en la frontera superior, seguido por las posiciones de las partículas en la frontera derecha, luego por las posiciones de las partículas en la frontera inferior, y finalmente por las posiciones de las partículas en la frontera izquierda. De forma análoga, las componentes de las velocidades se registran con el mismo criterio. Así, las entradas i-ésimas de los arreglos mencionados dan la posición y velocidad de una partícula específica.

Tomando lo anterior en cuenta, se puede asignar velocidades del fluido a todas las partículas en la frontera izquierda y en la frontera derecha, por ejemplo. Si se piensa a los arreglos con las componentes de la velocidad como divididos en cuatro conjuntos, el primer conjunto con las componentes de las velocidades de las partículas en la frontera superior, el segundo con las componentes de las velocidades de las partículas en la frontera derecha, etcétera, se deben modificar las velocidades del segundo y cuarto conjunto de ambos arreglos. Por supuesto, según el caso puede ser necesario modificar otra combinación de conjuntos, un solo conjunto o partes de los conjuntos.

Las otras dos funciones son `inicializaCondicionesDeFrontera(...)` y `correccionANodosNoContemplados(...)`. Como ya se comentó, estas funciones utilizan las localidades de las partículas. La primer función utiliza las localidades en el esquema tipo 2, y la segunda en el tipo 1. Así, la primer función recorre todas localidades del tipo 2, y suma las velocidades en cada localidad, y divide entre el número de partículas de la localidad. La segunda función hace lo mismo, pero sólo trabaja en las localidades de tipo 1 que corresponden a nodos que son de frontera, y que no tenían partículas alrededor en el esquema de localidades de tipo 2.

CAPÍTULO IV

Resultados

IV.1. Obtención de coordenadas a partir de una imagen

Para obtener el mapa de bits con el que se hace la asociación a coordenadas espaciales, se usa la biblioteca `stb_image.h`. En este caso, se considera la luminancia de un pixel. Si esta luminancia es más grande que un cierto umbral, se asigna al arreglo `bitmap[]` en la posición correspondiente un cero, si no, un 1. La imagen que se trabaja es la que se ve en la figura IV.1.

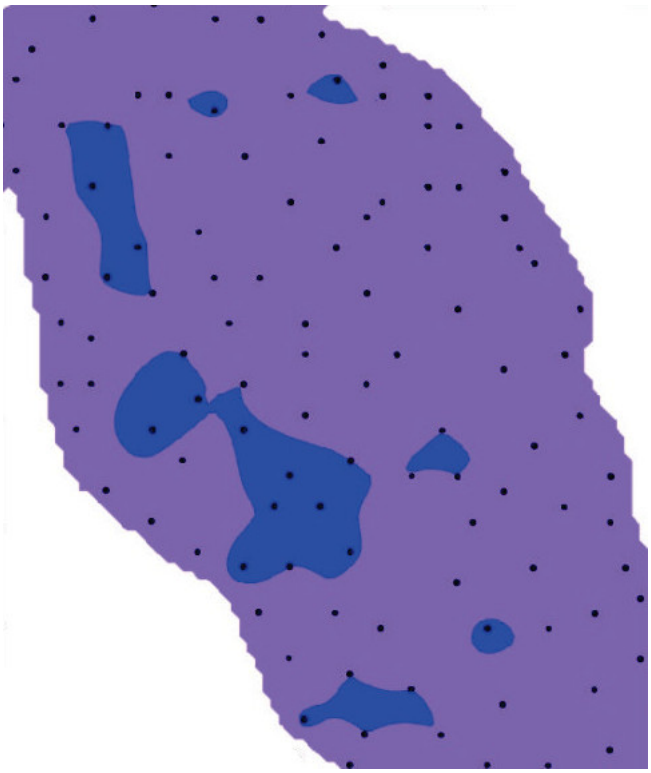


Figura IV.1: *Imagen original de prueba.*

De esta, el mapa de bits extraido es el que se ve en la figura IV.2.

Figura IV.2: *Mapa de bits extraido.*

Para obtener la siguiente imagen, se imprime en un archivo el arreglo `bitmap[]`, colocando un asterisco cuando la entrada vale uno, y colocando un punto cuando vale cero. El mapa de bitmaps mostrado en realidad no parece un mal candidato que cumpla con lo que se menciona en el capítulo III. Sin embargo, una inspección cercana al mapa de bits extraido muestra que hay dos problemas. La figura IV.3 muestra un acercamiento de una región problemática en la figura.

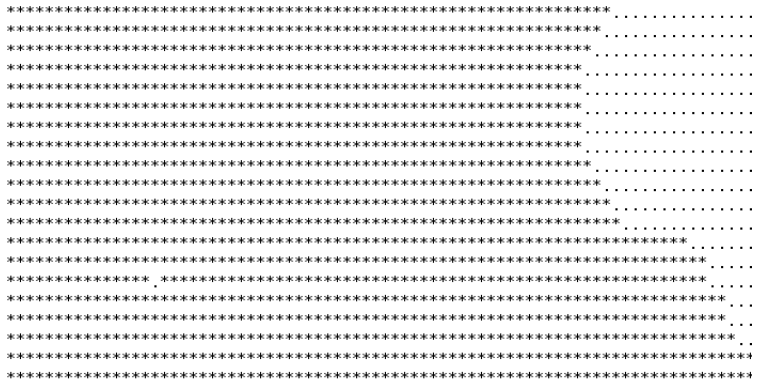


Figura IV.3: *Acercamiento al mapa de bits.*

Se observa que en la región interior hay un punto, indicando que en el mapa de bits hay un cero. También, del lado derecho, se observa una curva ligera que retrae la frontera hacia el interior. El punto en la región interior lleva al programa a interpretar ese punto como el punto en la frontera inferior en esa columna. Entonces, ese punto crea una discontinuidad en la frontera obtenida. Sobre el segundo problema, la consecuencia es similar. Del segundo problema, se hace evidente que la imagen de entrada

no cumple con los requerimientos de que el dominio mostrado sea una región y-simple. Pero este es un problema que no es evidente de observar la figura IV.2. Mucho menos es evidente los huecos internos. Entonces, se aplica un paso de preprocesamiento al mapa de bits extraido. El resultado se muestra en la figura IV.4.

Figura IV.4: *Mapa de bits con preprocesamiento.*

Desde luego, esto modifica el dominio que se quiere trabajar. Sin embargo, con un método como el descrito en la sección III.4 es posible obtener el dominio tal como se requiere.

A continuación, para obtener las coordenadas de la frontera se ejecuta la función `generacionDeParticulasEnPared(...)`. El resultado es el que se muestra en la figura IV.5.

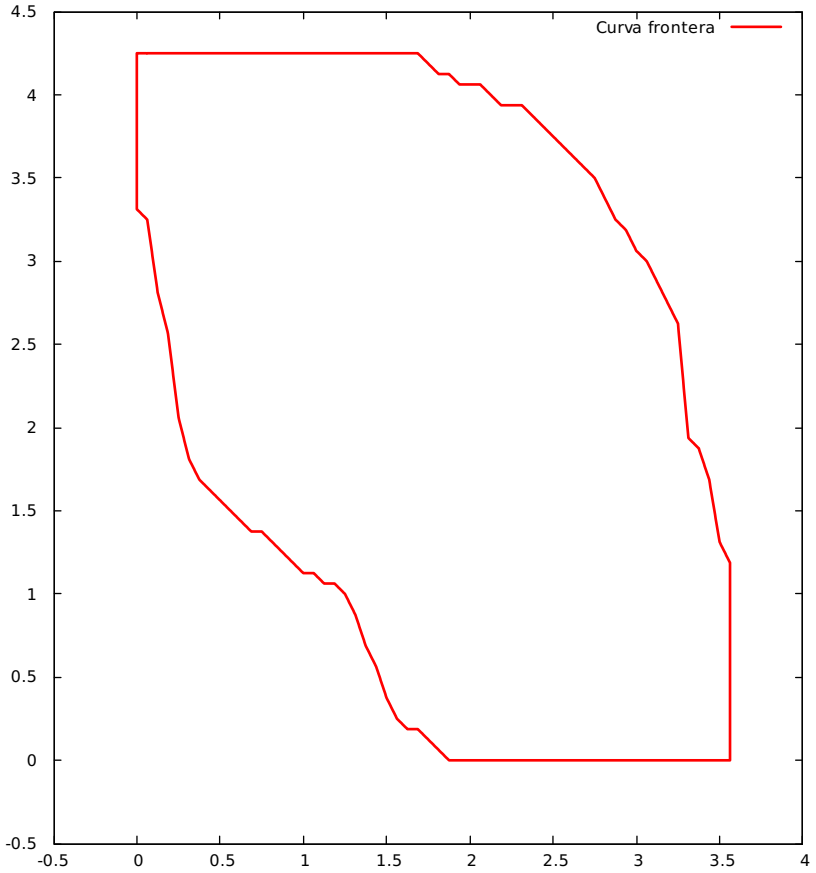


Figura IV.5: *Puntos que describen la frontera de la imagen de entrada.*

Hay un aumento de puntos debido a la misma función. Esto es relevante más adelante.

IV.2. Identificación de localidades y su ordenamiento.

La figura IV.6 muestra los arreglos de localidades de los puntos en la frontera obtenidos en la sección anterior para los dos tipos de localidades. Recuérdese que cada 4 elementos del arreglo que guarda las localidades están asociadas a una misma partícula.

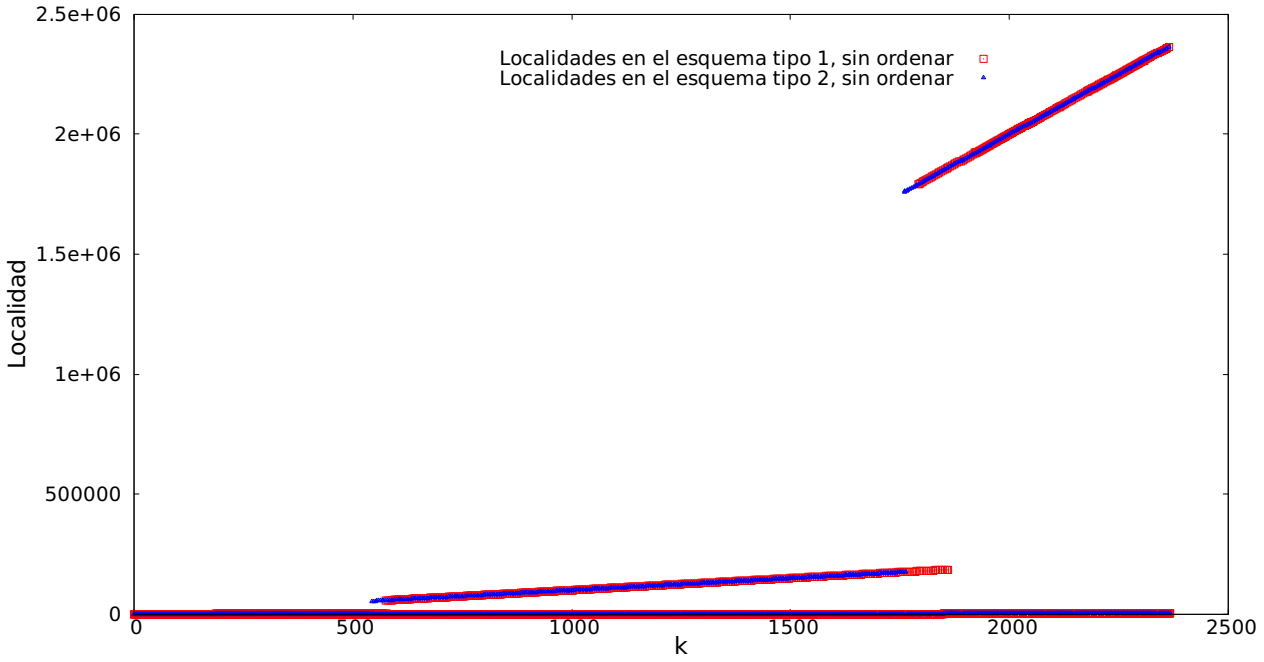


Figura IV.6: *Localidades en los esquemas tipo 1 y 2 en que las partículas se encuentran.*

Se observa que el conjunto de localidades en un esquema y en otro muestran comportamientos bastante similares. Sin embargo, la comparación directa entre uno y otro arreglo, que contienen las localidades, muestra que no son iguales. Estos comportamientos son de esperar porque espacialmente no difieren mucho un tipo de localidad del otro.

Se muestra en la figura IV.7 las mismas localidades después de ser ordenadas por `radixSort(...)`.

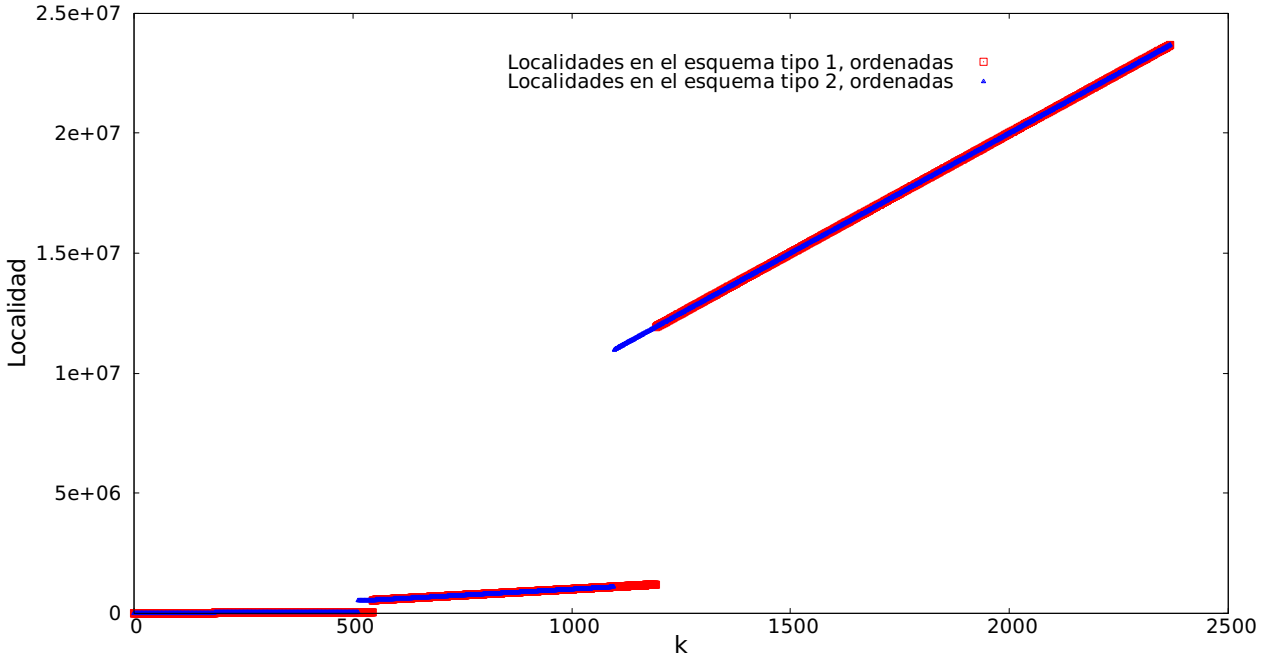


Figura IV.7: *Localidades en los esquemas tipo 1 y 2, ordenadas.*

Se observa claramente que en efecto se ordenó cada conjunto.

IV.3. Rectificación de la frontera.

A partir de las localidades tipo 2 y las posiciones de cada partícula frontera, se procede a la rectificación de la frontera. Graficando pequeños de segmentos de recta, cercanos a cada nodo frontera, se obtiene lo que muestra la figura IV.8.

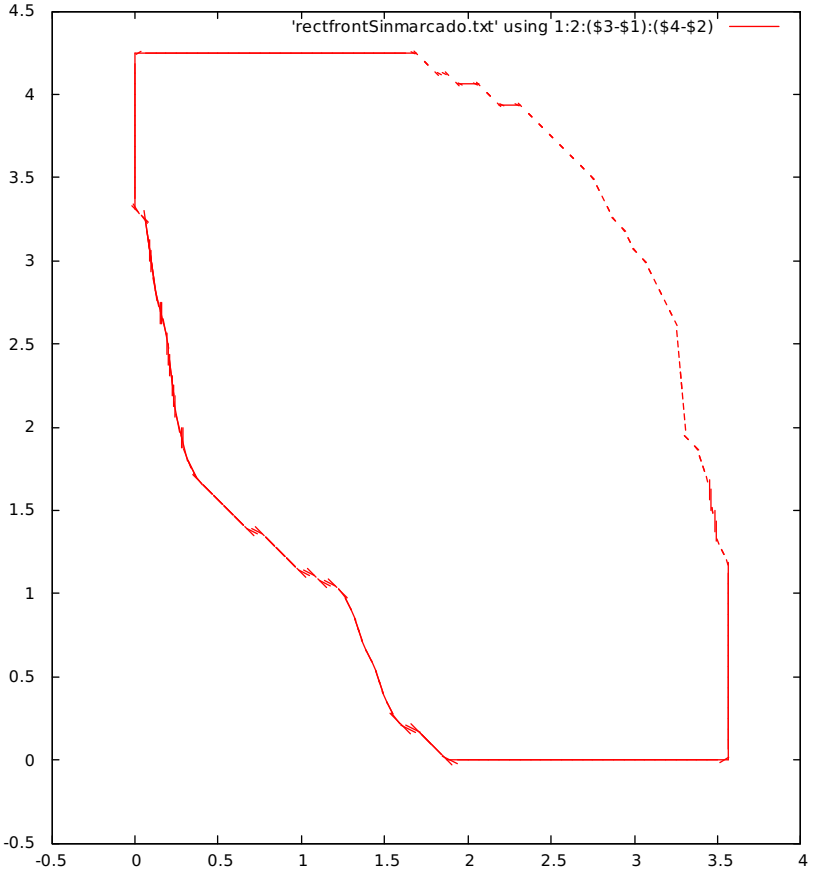


Figura IV.8: *Ajuste de segmentos de recta a la frontera.*

Se muestra en la figura IV.9 un acercamiento. Por supuesto, localmente dan una aproximación a la frontera, por lo que los segmentos de recta ajustarán mejor unas secciones de la curva que otra. Además, es posible encontrar unos cuantos huecos debido a que hay nodos frontera que no son identificados por falta de partículas en su localidad.

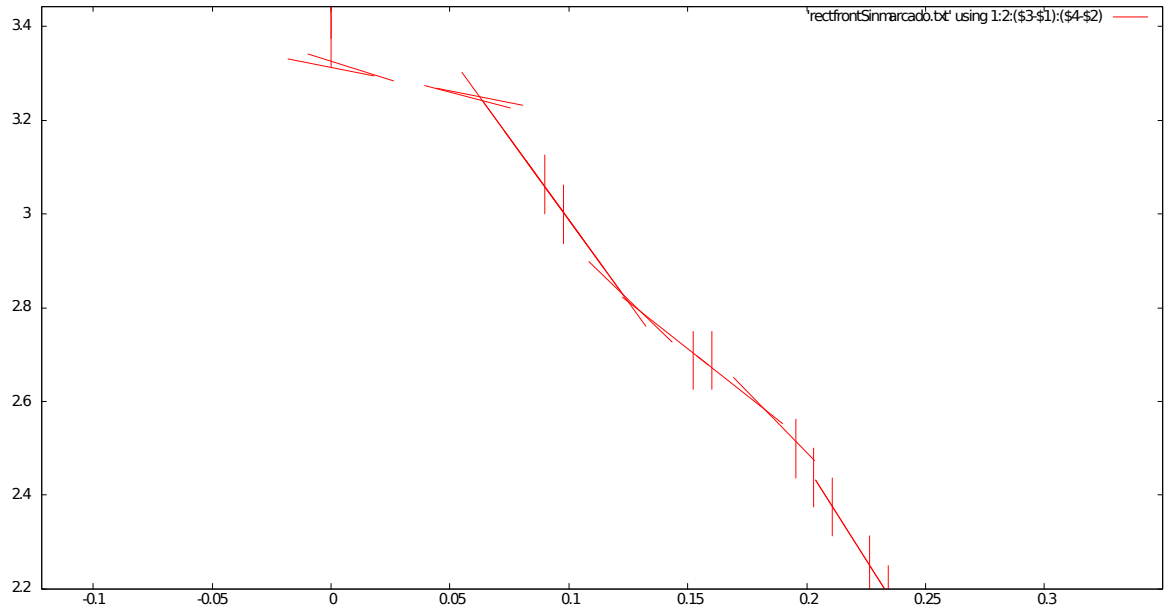


Figura IV.9: *Acercamiento a una región donde se observa el ajuste de segmentos de recta a la frontera.*

A continuación, ejecutando la función `correccionANodosNoContemplados(...)` se obtiene la figura IV.10.

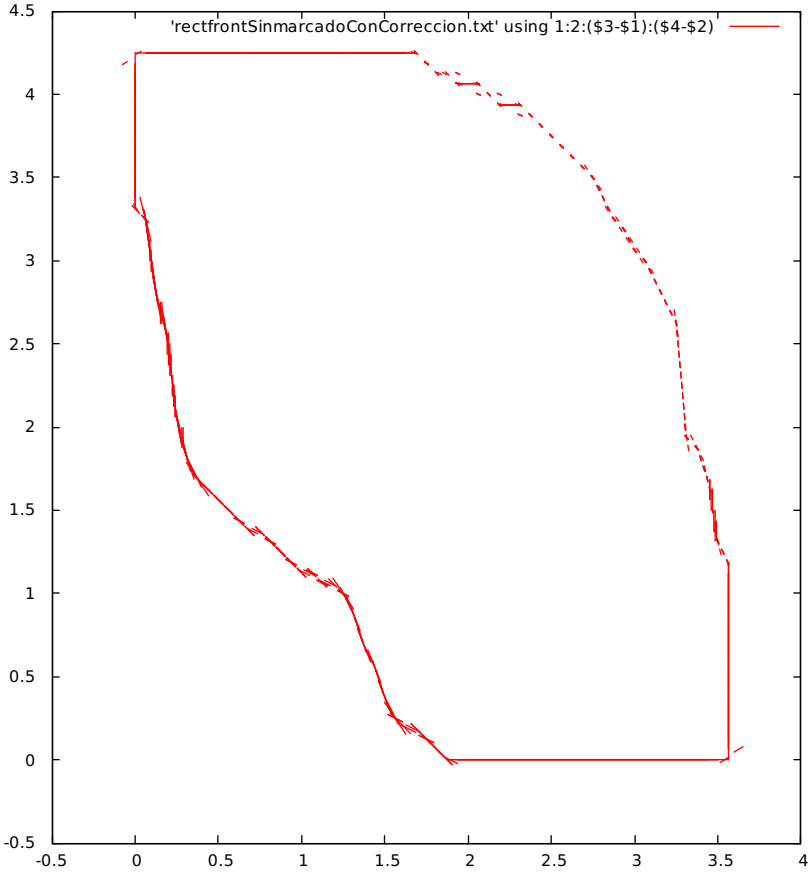


Figura IV.10: *Ajuste de segmentos de recta a la frontera, después de el paso de corrección.*

También, en la figura IV.11 se muestra un acercamiento.

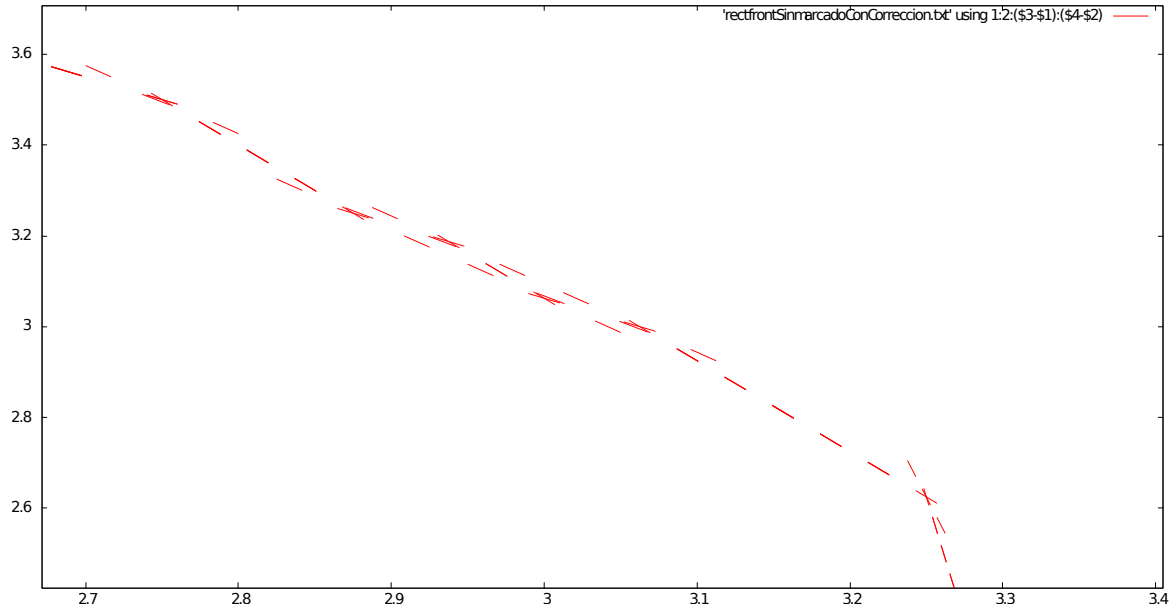


Figura IV.11: *Acercamiento a una región donde se observa el ajuste de segmentos de recta a la frontera, después de el paso de corrección.*

Nótese que los espacios entre los segmentos de recta son más reducidos. Sin embargo, se señala que las figuras mostradas son sólo *cualitativas*. La razón es que para obtenerlas se usó el punto de las rectas de ajuste a la altura de los nodos. Este punto puede no ser el más cercano al nodo. Pero es el punto más cercano a esos segmentos el que realmente ajusta la curva frontera. El resultado es que hay pequeños segmentos de recta que, a pesar de corresponder a la recta de ajuste, están centrados en un punto incorrecto.

IV.4. Identificación de tipos de nodo.

A continuación, se ejecuta la función `identificaNodosFrontera(...)`. El resultado se muestra en la figura IV.12.

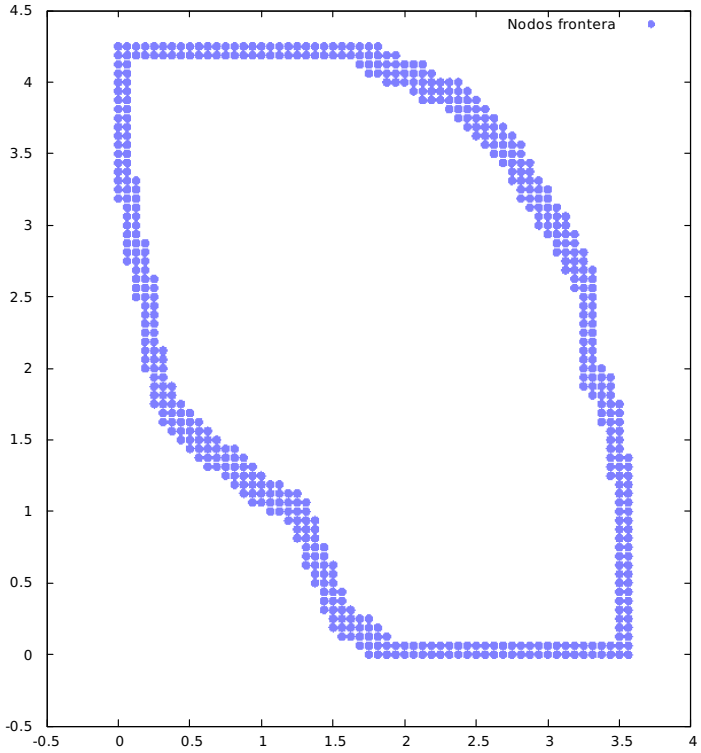


Figura IV.12: *Identificación de nodos frontera.*

Luego, con la información de cuáles son los nodos frontera, se identifica el tipo de todos los nodos con la función `identificaTipoDeNodos(...)`. El resultado se muestra en la figura IV.13.

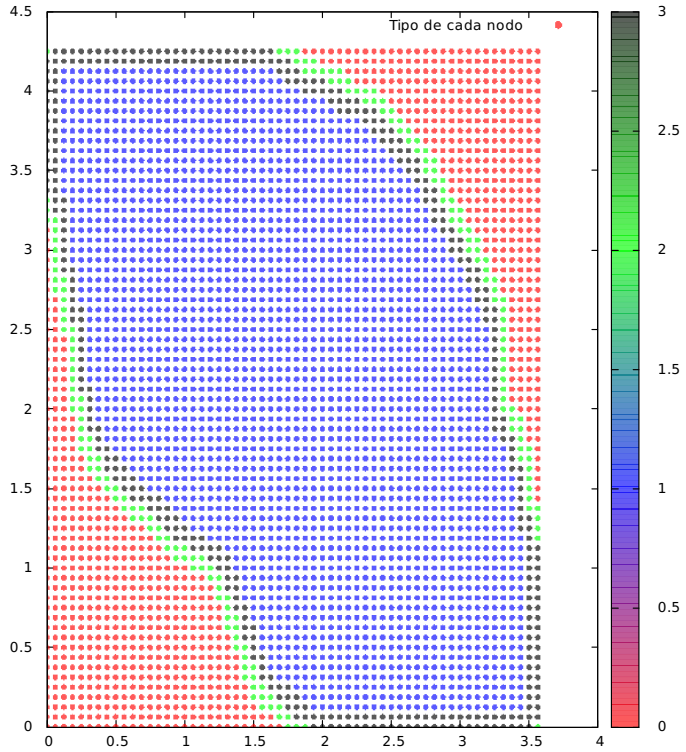


Figura IV.13: *Tipo de cada nodo del dominio general.*

IV.5. Elección de nodos para la configuración 3.

Para hacer la elección de nodos como en la configuración 3 descrita en la sección II.4.2, usamos la función `marcaNodos(...)`. El resultado se muestra en las figuras IV.14 y IV.15.

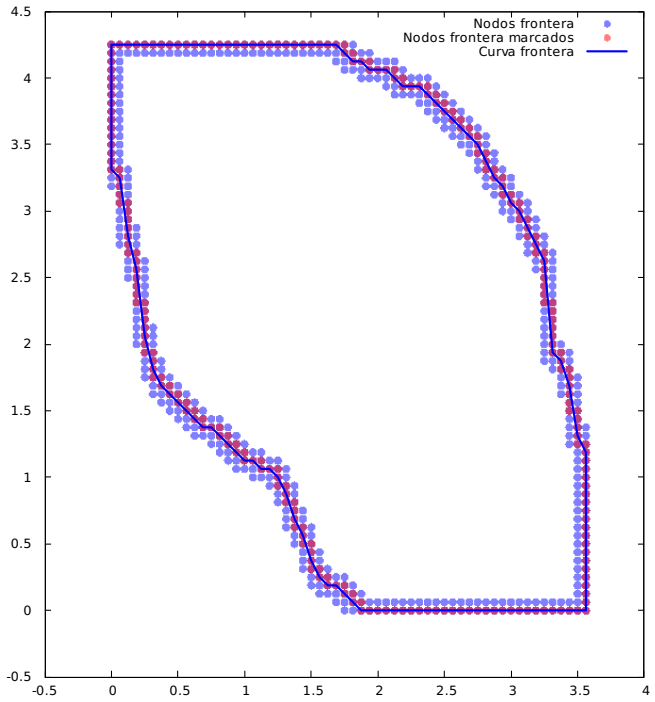


Figura IV.14: *Superposición de nodos frontera y nodos frontera marcados. Se indica la frontera del dominio.*

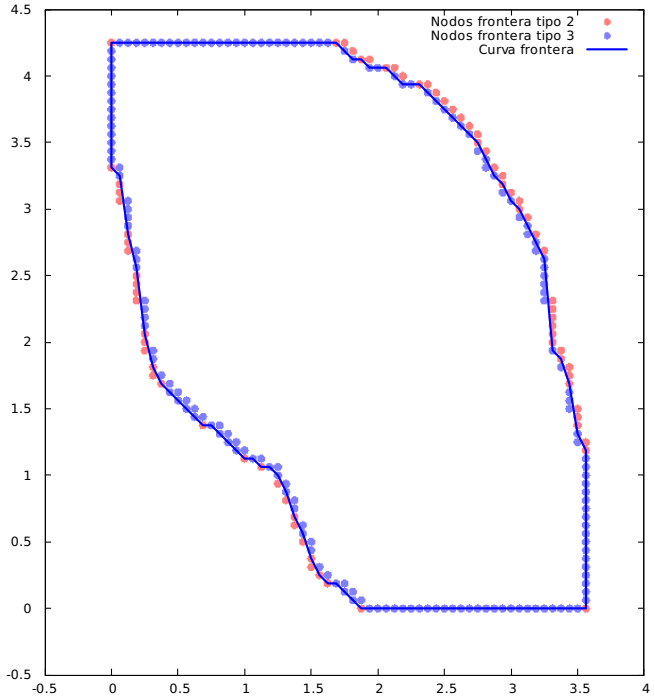


Figura IV.15: *Nodos frontera marcados y sus correspondientes tipos. Se indica la frontera del dominio.*

Con los nodos elegidos, se observa en la figura IV.16 la rectificación usando los segmentos de recta de sólo esos nodos.

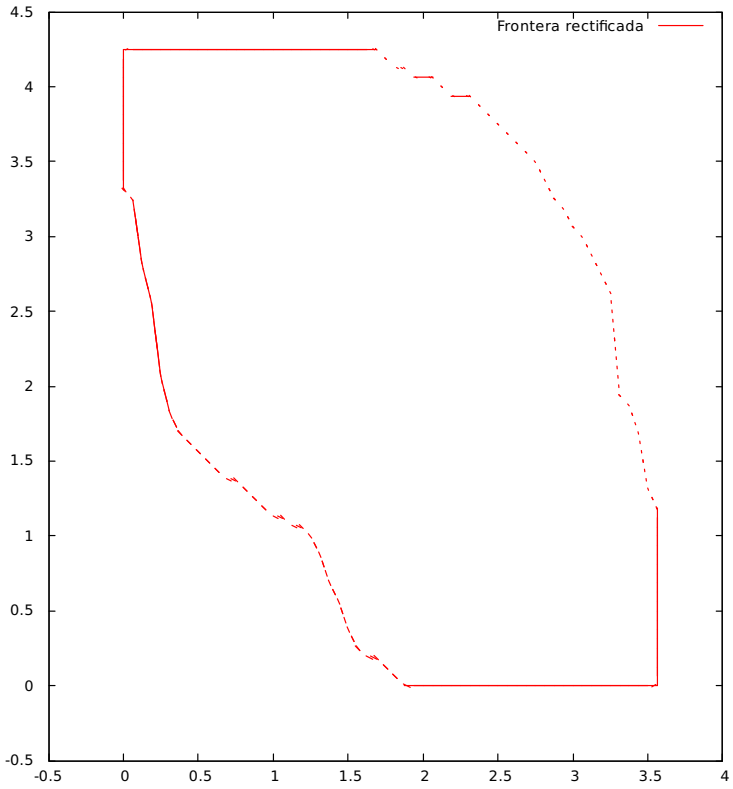


Figura IV.16: *Rectificación final de la frontera.*

IV.6. Asignación de condiciones de frontera por medio de las partículas en la frontera

La elección de las condiciones de frontera puede variar mucho de problema a problema. Aquí, para ilustrar una elección común, se asigna a las partículas en la frontera izquierda y derecha una velocidad en x. Esto se hace por medio de la función `inicializaVelocidadesDeParticulasMoviles(...)`. Cuando se posee la información de la o las localidades (en ambos esquemas) en que una partícula se encuentra, la función `inicializaCondicionesDeFrontera(...)` y la función `correccionANodosNoContemplados(...)` asignan el promedio de las velocidades al nodo cercano a la frontera, de las partículas en la correspondiente localidad. En la figura IV.17 se resaltan las partículas a las que poseen una componente x de la velocidad distinta de cero.

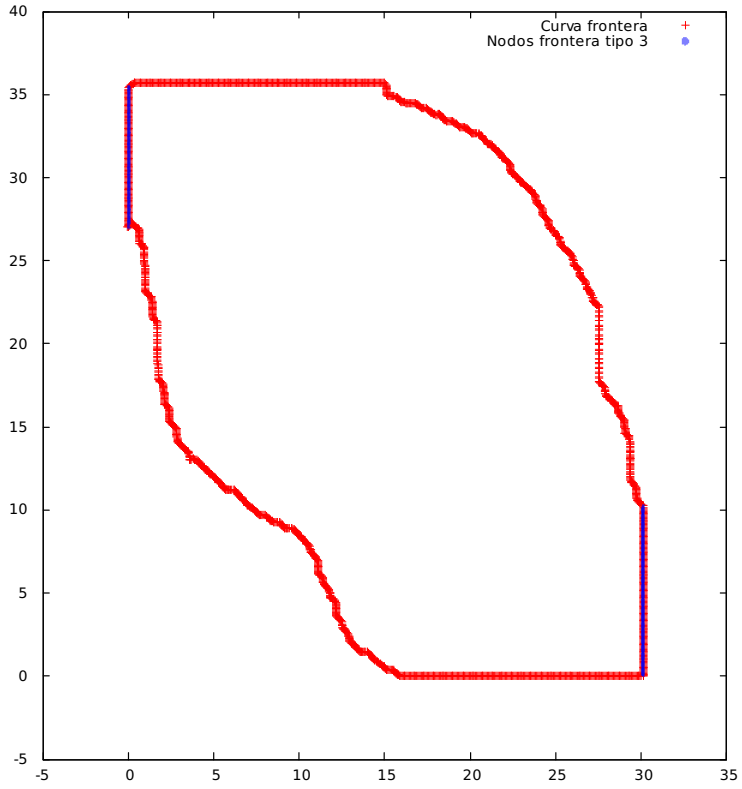


Figura IV.17: *Región de la frontera a la que se le asignó una componente de la velocidad en x distinta de cero.*

IV.7. Otros ejemplos

Para ilustrar la utilidad de colocar partículas extra, basta con ver la figura IV.18. El principal problema de esa figura proviene de las regiones donde la frontera es vertical. Para el caso cuando la entrada son las coordenadas de la frontera, también se puede agregar partículas intermedias, y se evita esa clase de problemas.

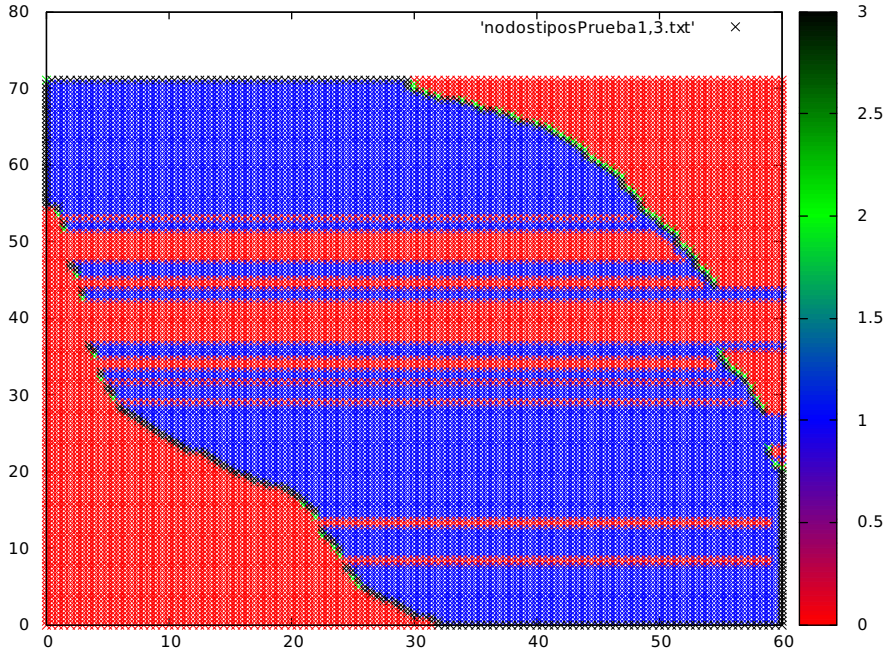


Figura IV.18: *La misma imagen usada para las pruebas anteriores, pero sin colocar partículas intermedias.*

La figura IV.19 muestra otro ejemplo más de aplicación.

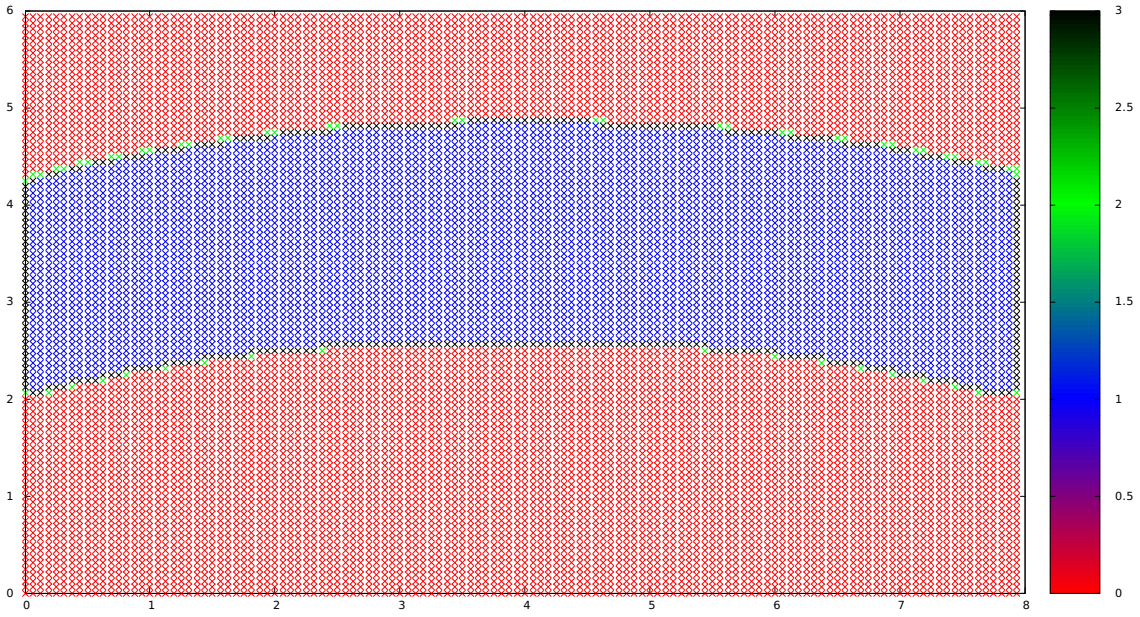


Figura IV.19: *Ejemplo de un dominio curvo.*

También, el programa puede considerar huecos en el dominio. Cada hueco debe entrar como imagen separada, y en general las mismas reglas que aplican para estos son las que se piden para la primer región. La figura IV.20 muestra un ejemplo. Obsérvese que la región parece tener tres huecos, y dos de ellos están muy pegados. En realidad se trata de un solo hueco, pero el tramo que los une es muy delgado. Aquí la resolución de la imagen de entrada contrajo ese tramo y el resultado es que no haya

nodos de bulto (aunque sólidos) en él. Por supuesto, el mismo problema puede surgir para una figura donde los nodos interiores sean nodos fluidos de bulto, y eso sí respresenta un gran problema. Esta situación ilustra que la resolución de la imagen de entrada determina la cantidad de nodos puestos en una región.

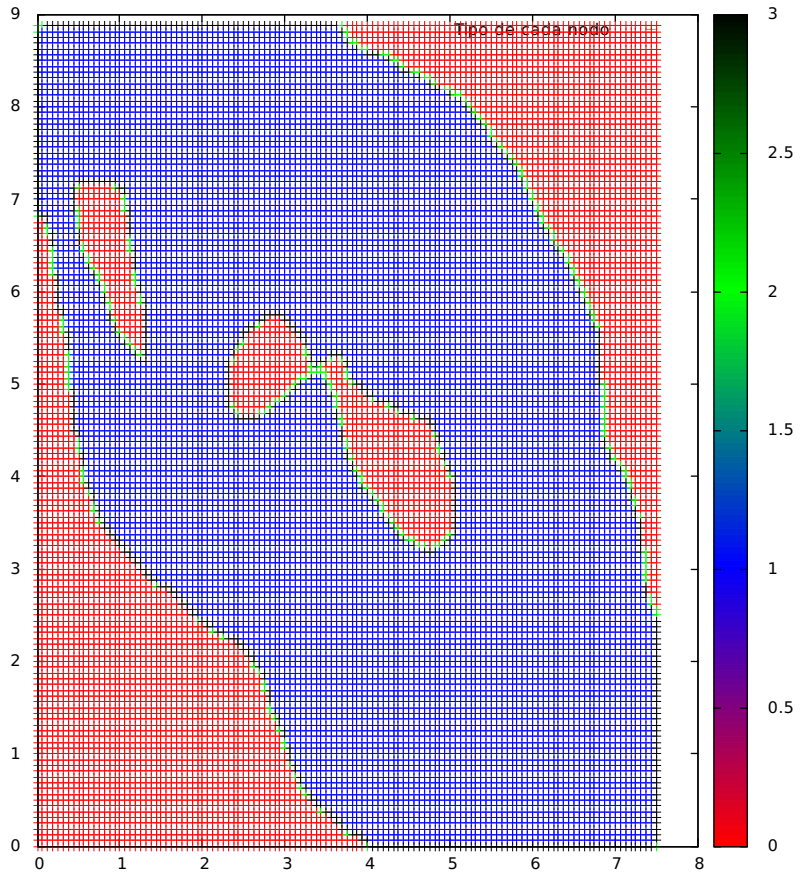


Figura IV.20: *Tipos de nodos para una región con huecos.*

El último ejemplo muestra el perfil del ala de un avión. Este perfil se obtuvo de `http://m-selig.ae.illinois.edu/ads/coord_database.html` y corresponde al perfil de ala del USA-34. En este caso, la entrada al programa son coordenadas.

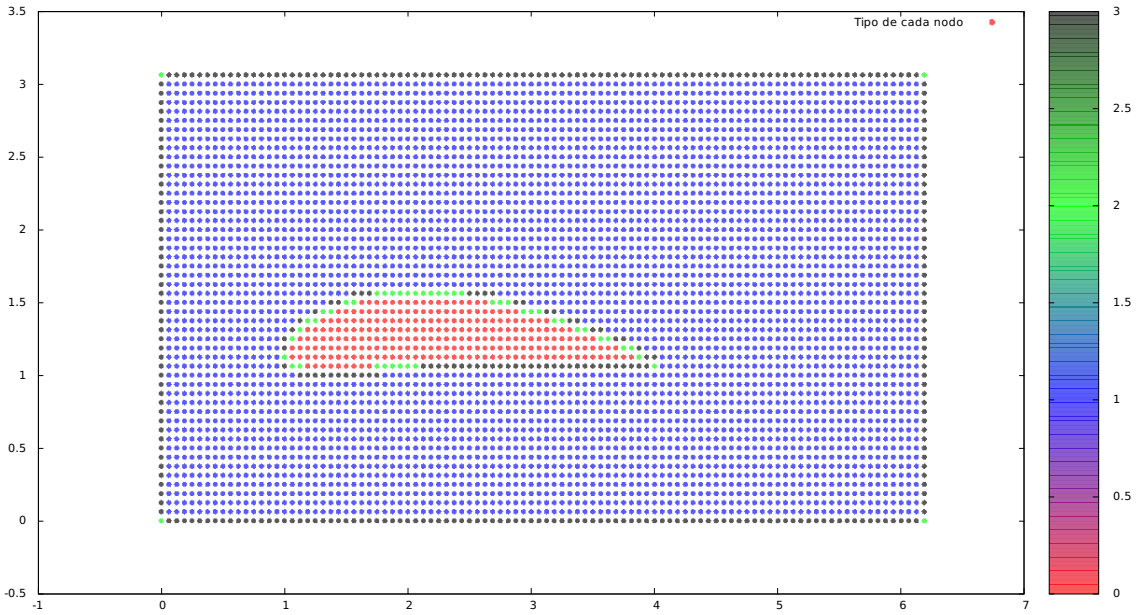


Figura IV.21: *Perfil de ala de avión.*

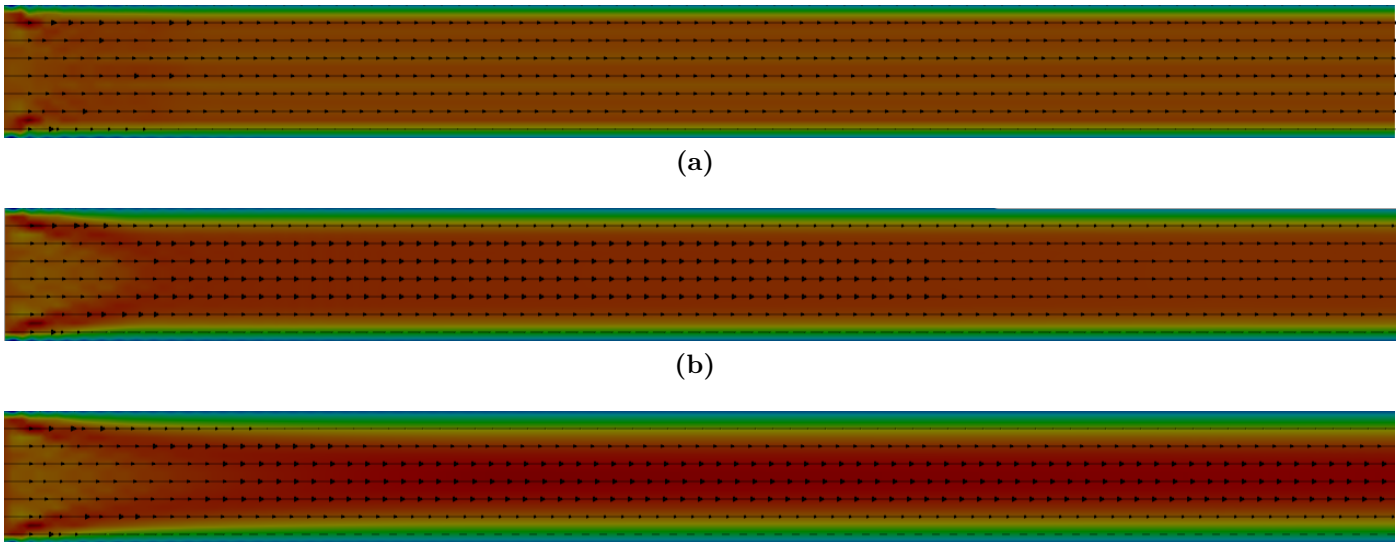
IV.8. Simulaciones con Lattice Boltzmann

Cuando en un problema particular no se cuenta con una imagen que lo describa sino del conjunto de puntos que conforma la frontera, el programa descrito se aplica exactamente igual, pero omitiendo la primer sección del capítulo presente. Lo que se obtiene es la construcción de los nodos más cercanos a la frontera (configuración 3), y esto puede ser útil para algún otro esquema de condiciones de frontera aparte del de frontera curva, como por ejemplo, el de rebote o *bounce back*. La colección de nodos frontera sólidos (configuración 2) en un caso puede considerarse muy lejana, pues la frontera está antes. O considerando el conjunto de nodos frontera fluidos (configuración 1) como frontera, estos pueden considerarse muy cerca del fluido. Un enfoque donde se carga el peso en ambas partes consiste en considerar los nodos frontera más cercanos a la frontera como los nodos donde se llevan a cabo las condiciones de frontera.

De cualquier forma, como ejemplo de la aplicabilidad del programa, se presentan a continuación una serie de simulaciones.

Flujo de Poiseuille

El flujo de Poiseuille es un flujo laminar que se presenta cuando un fluido es viscoso. En este flujo, el patrón de magnitud de velocidades varía respecto del eje del tubo en el cuál está contenido el fluido, siendo más grandes entre más cerca estén del centro del eje y más pequeñas entre más cerca estén de las paredes.



(a)

(b)

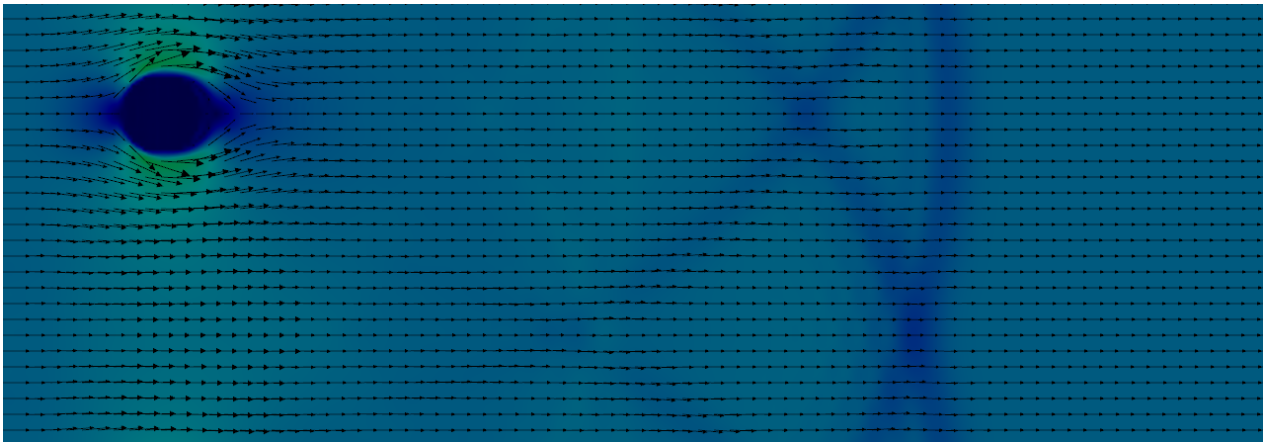
(c)

Figura IV.22: *Flujo de Poiseuille en tres instantes de tiempo.*

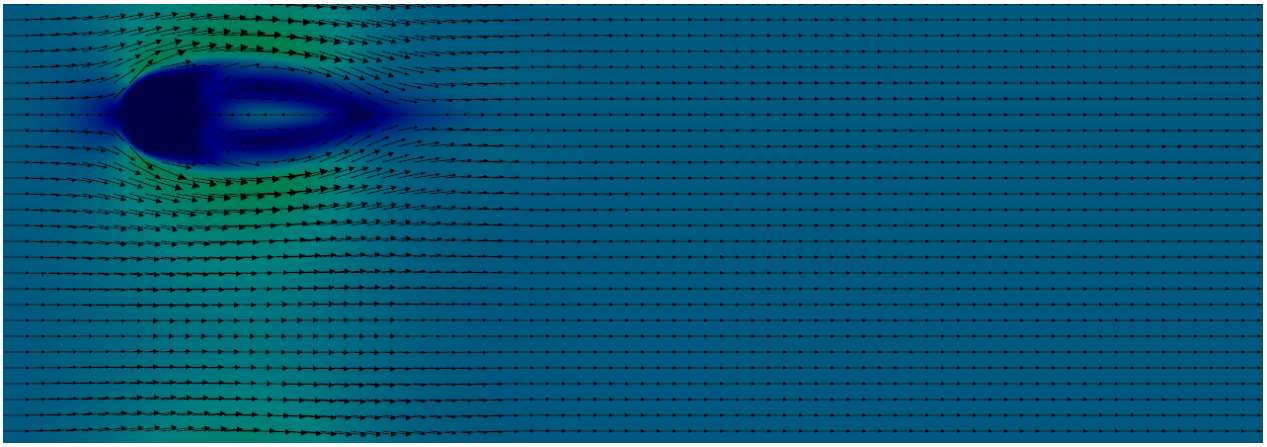
Calle de vórtices de von Kármán

En la dinámica de fluidos, la calle de vértices de von Kármán es un patrón repetitivo de vórtices giratorios causados por la separación inestable de flujo de un fluido alrededor de un cuerpo romo.

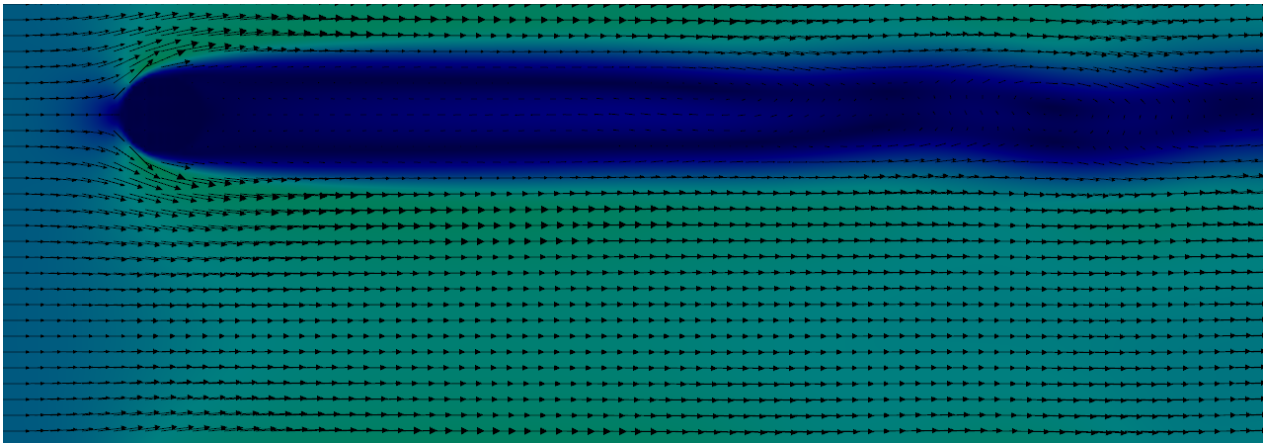
Sólo se forma en un cierto rango de velocidades de flujo, especificado por un rango de números de Reynolds (véase apéndice A), típicamente por encima de un valor *Re* de 90.



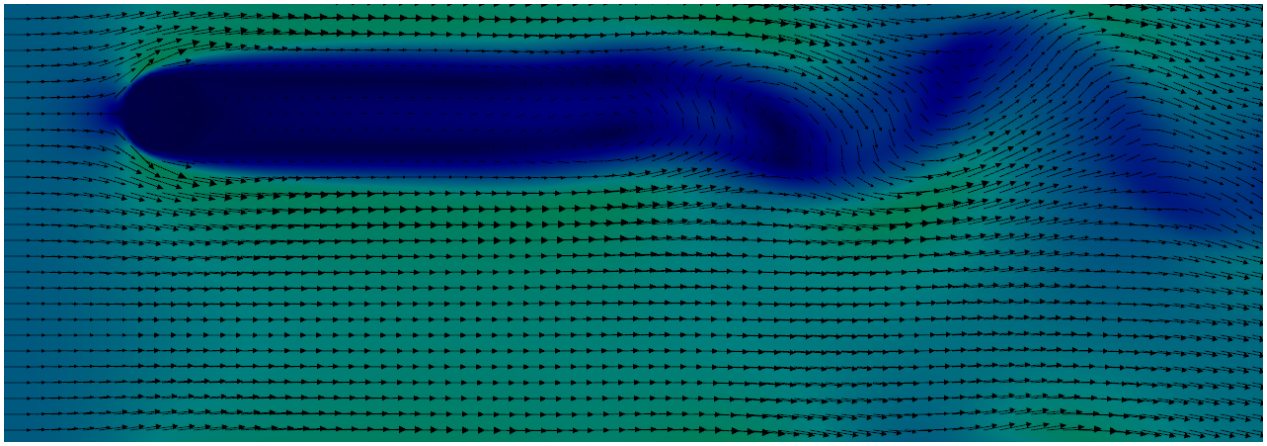
(a)



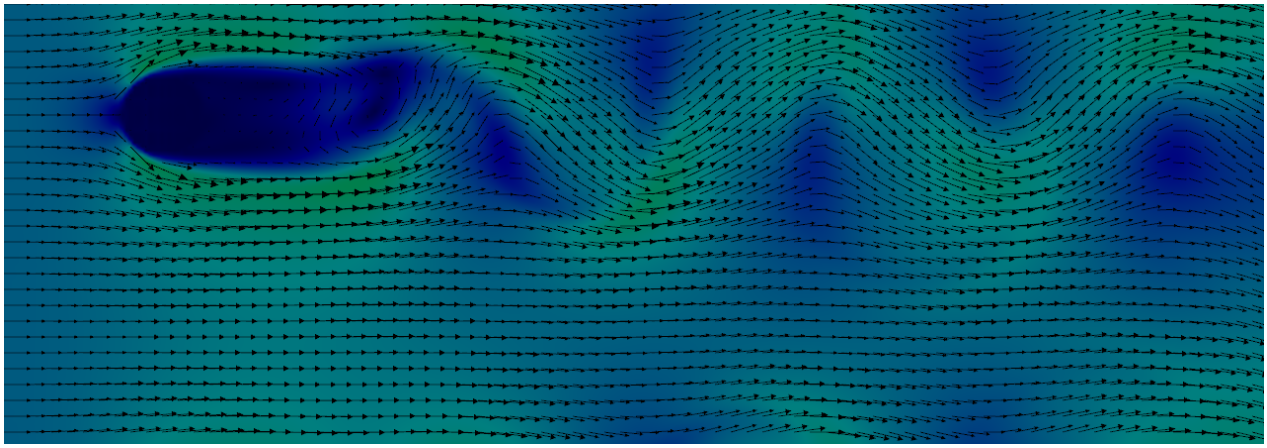
(b)



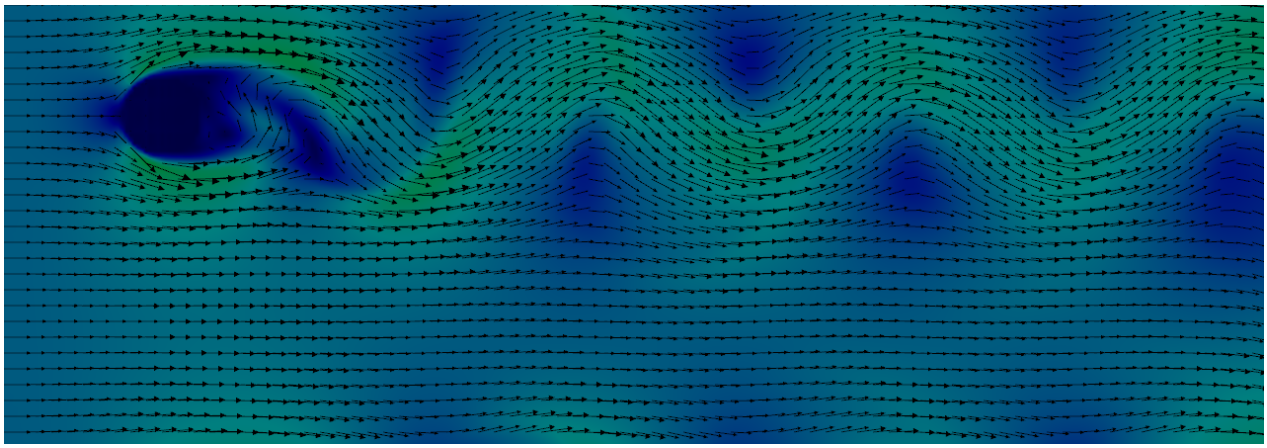
(c)



(d)



(e)



(f)

Figura IV.23: *Calle de vórtices de Von Kármán en seis instantes de tiempo.*

Cavidad con tapa móvil

Un problema común para pruebas de referencia es el de la cavidad con tapa móvil. El problema consiste en una cavidad con un fluido inicialmente estático. Uno de los lados de la cavidad, llamado tapa, se

empieza a mover con una velocidad fija. Esto genera una circulación del flujo al interior de la cavidad. Según la velocidad de la tapa, y dimensiones de la cavidad (finalmente, según el número de Reynolds), se formarán uno o más vórtices. Los parámetros usados son los mismos que se usan en el ejemplo planteado en el apéndice A.

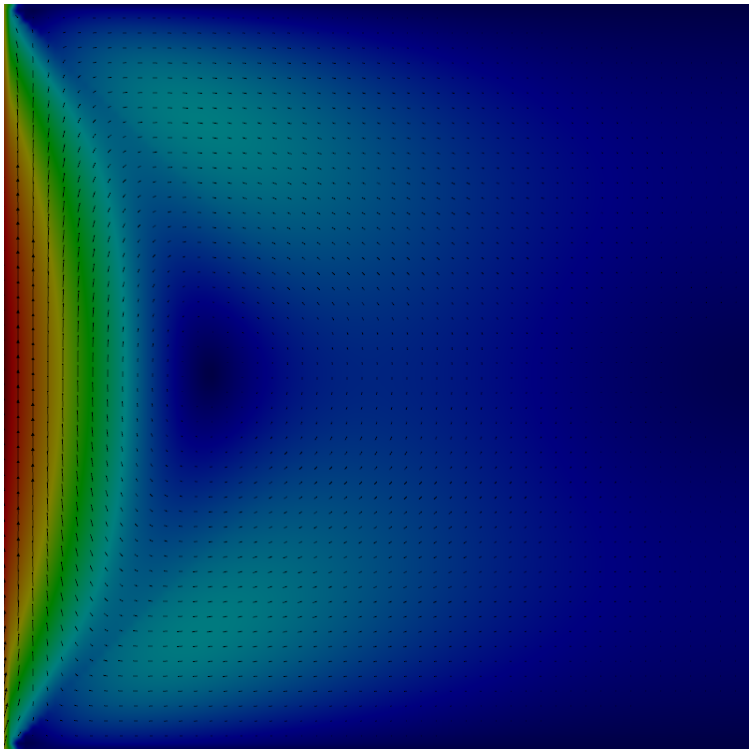


Figura IV.24: *Campo de velocidades dentro de una cavidad con tapa móvil. La tapa móvil se encuentra a la izquierda de la cavidad.*

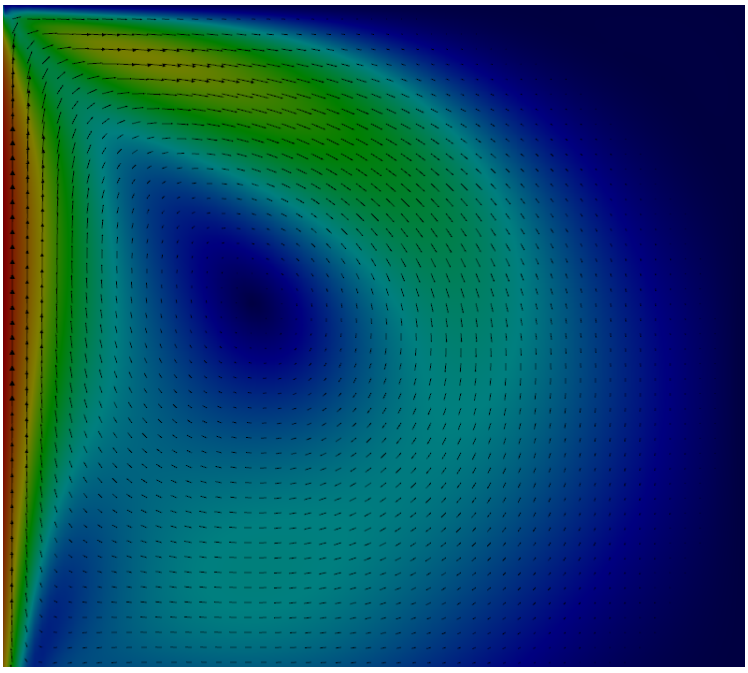


Figura IV.25: *Campo de velocidades dentro de una cavidad con tapa móvil, a un tiempo más avanzado.*

Flujo alrededor del perfil de un ala de avión

Los distintos modelos de aviones implican una construcción diferente para los perfiles de sus alas. Estos perfiles crearán un flujo de velocidades distinto en cada caso. Una manera de investigar este flujo es suponer que el ala está fija, y es el flujo el que avanza hacia el ala (invarianza Galileana). Para las situaciones ejemplificadas en las figuras IV.26 y IV.27, una pared está cerca del ala, por lo que el flujo se modifica aún más.

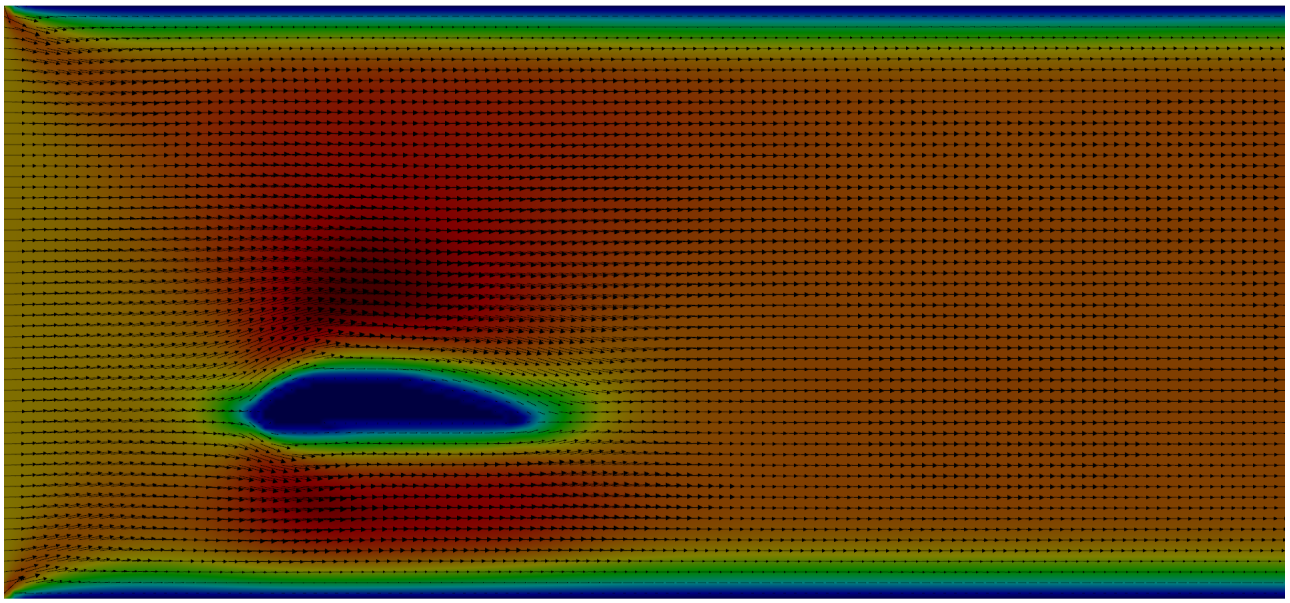


Figura IV.26: *Flujo alrededor del perfil del ala de un avión.*

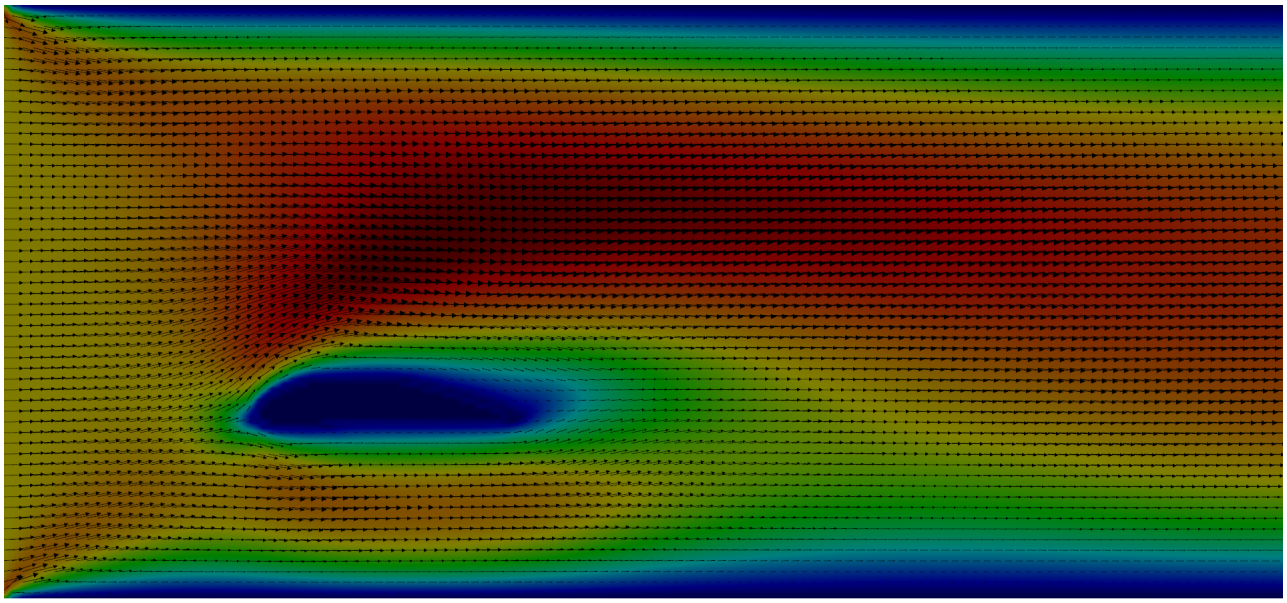
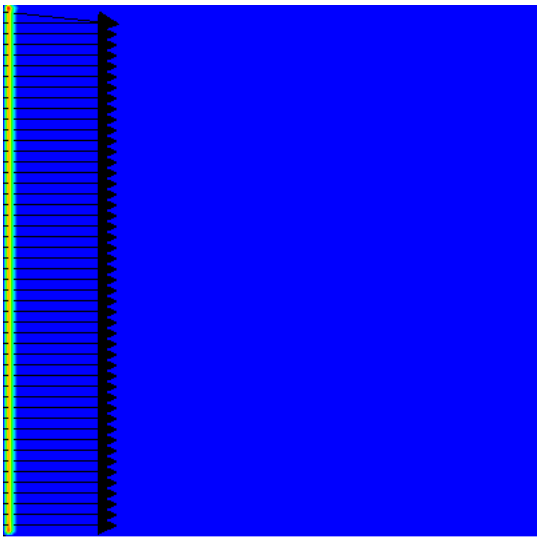


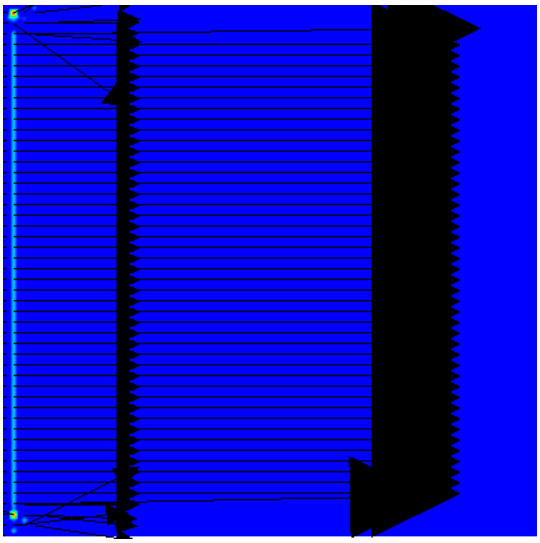
Figura IV.27: *Flujo alrededor del perfil del ala de un avión a un tiempo más avanzado.*

Prueba con condiciones de frontera de no resbalamiento para fronteras curvas

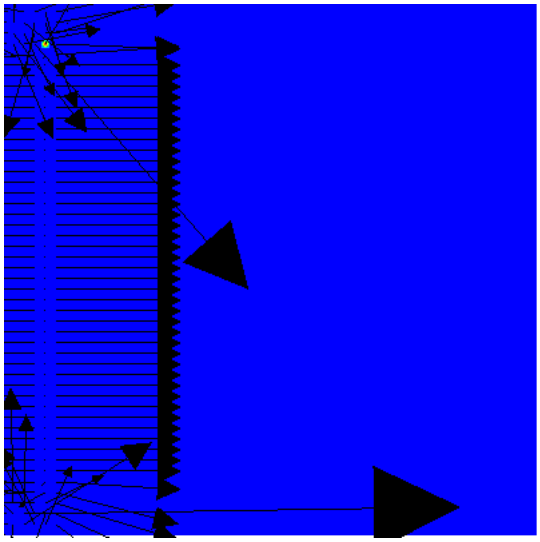
Las condiciones de frontera de no resbalamiento para fronteras curvas en principio son aplicables a fronteras rectas. Así, el problema probado es el de la cavidad con tapa móvil. Los mismos parámetros usados en la prueba anterior para la cavidad con tapa móvil se usan aquí.



(d)



(e)



(f)

Figura IV.28: *Campo de velocidades dentro de una cavidad con tapa móvil para tiempos iniciales. La tapa móvil se encuentra a la izquierda de la cavidad. (a) Campo de velocidades para el primer paso de tiempo. (b) Campo de velocidades para el paso 5 de tiempo. (c) Campo de velocidades para el paso 10 de tiempo.*

Se observa que la simulación se vuelve rápidamente inestable, y las magnitudes de las velocidades al primer paso de tiempo ya superan la velocidad del sonido del sistema. También se observa que en las esquinas de la cavidad pegadas a la pared móvil las velocidades empiezan a crecer mucho.

CAPÍTULO V

Discusión y conclusiones

Los resultados mostrados en el capítulo IV muestran gran parte de la utilidad del programa. Es decir, la definición de la red de nodos para aplicar el método de Lattice Boltzmann. Existe una gran cantidad de métodos ([9], [23], [12], [7] [21]) que trabajan con la información del tipo de un nodo.

Además, un método que trabaje el problema de fronteras móviles en algún momento necesitará hacer una recatalogación de los nodos, incluso si no requiere conocer específicamente si un nodo es del tipo de frontera sólido o si es de frontera fluido.

También, un dominio con fronteras complejas representa un reto para imponer las condiciones de frontera. El método de partículas empleado se presta para cargar esa información, y que estas condiciones sean modificadas como reacción al medio o quizá, modificadas por algo externo.

En este trabajo se hizo una extensa labor recopilatoria de lo que forma parte del método actual. Aunque las variantes del método son muchas, lo que se presenta aquí representa buena parte de lo que se encuentra en la literatura de forma más común.

Como se discutía en la última sección del capítulo IV, el método se presta particularmente para usar unas de las condiciones de frontera más rápidas de implementar, cuando todo lo que se posee son las coordenadas de un objeto, y posiblemente del medio en el que este se encuentra.

Sin embargo, uno de los objetivos futuros es implementar métodos de condiciones de frontera con mayor precisión. A pesar de que mucho del énfasis del trabajo se puso en un método particular para tomar en cuenta las condiciones de frontera, las pruebas con él no fueron exitosas. La razón posible del fallo en esta parte, se cree que tiene que ver con la inicialización del problema. Aunque esta razón puede intentar descartarse porque el mismo método parece comportarse adecuadamente con las condiciones de frontera de "Bounce-Back", condiciones periódicas, condición de entrada y de salida. Sin embargo, las pruebas hechas con las condiciones de frontera para fronteras curvas dan origen a velocidades relativamente grandes. Obsérvese que la velocidad del sonido para el sistema es $c_s = 1/\sqrt{3}$. Así, un gradiente grande entre velocidades entre la frontera y el medio ocasionan velocidades mayores a esta velocidad. Pero que la velocidad del medio sea mayor a la velocidad del sonido del sistema resultará en que haya mayor compresibilidad del fluido en el dominio. Esto, por el análisis multi-escala de Chapman-Enskog está prohibido porque una de sus consecuencias es que el fluido a tratar sea incompresible. Entonces, inicialmente ni siquiera se están cumpliendo las condiciones requeridas.

Respecto del método para delimitar el dominio, varias mejoras son posibles. Una de ellas es un tanto sensible a la separación entre nodos de la red. La razón es que las localidades en que una partícula se encuentra pueden cambiar, y también el número de partículas en cada localidad. La consecuencia es que, el sentido en que se recorre la curva puede ser mal asignado. Esto lleva finalmente a que un nodo sea erroneamente etiquetado, y consecuentemente, los nodos interiores o exteriores en el mismo renglón también.

Hay varias razones para elegir un ajuste por segmentos de recta a la frontera. Una de ellas es que es mucho más sencillo. Pero la principal razón es que la resolución del método no va más allá de la separación de los nodos. Así, aunque una colección de partículas de la frontera tenga posiciones oscilantes dentro de las localidades comprendidas entre los nodos, el método hará un promedio de las propiedades que pueda proveer la frontera.

Para el método de condiciones de frontera curva, surgen problemas con ciertos tipos de regiones. Si la región introducida al programa, ya sea por medio de coordenadas o una imagen, posee una frontera no muy suave, es decir, con picos, es probable que catalogue un cúmulo de cuatro nodos o más, como nodos de frontera. Esto aisla nodos frontera y los hace inútiles para el método. Una versión futura debería identificar estos problemas y hacer una selección más precisa de los nodos frontera.

Otro punto importante es que aunque se poseen simulaciones usando el método de Lattice Boltzmann, se descarta toda intención de hacer comparaciones cualitativas con alguna prueba de referencia. El problema de las condiciones iniciales, si se trata mal, con buena probabilidad introducirá un error que hará inútiles los resultados. Esto no quiere decir que se crea que cualitativamente los resultados de las simulaciones no digan nada. Dan alguna información, pero se cree que otros fenómenos son enmascarados.

Gran parte de los códigos implementados están pensados para paralelizarse con unas ligeras modificaciones. Particularmente, el paso de colisión, el paso de flujo, el paso de reconstrucción, las funciones de identificación de tipo de nodos y de marcado de nodos son paralelizables. Esto reparte la carga de trabajo, que burdamente se puede decir que es de orden mayor o igual a $NX \times NY$, el número de nodos en el dominio.

Finalmente, no sólo el método de Lattice Boltzmann se puede beneficiar de los resultados obtenidos. Encontrar la interfaz entre dos fluidos es algo común en los problemas computacionales de la física de fluidos. Muchos métodos utilizan partículas para rastrear la frontera, y reconocer las regiones en que cada fluido se encuentra.

APÉNDICE A

Apéndice: Adimensionalización y unidades del método de lattice Boltzmann

El acercamiento presentado aquí consiste de dos pasos. Un sistema físico primero se convierte a un sistema sin dimensiones, que es independiente de las escalas físicas originales, pero también independiente de los parámetros de simulación. En un segudno paso, el sistema sin dimensiones se convierte en una simulación discreta. Las relaciones entre estos tres sistemas (el físico (P), el que no tiene dimensiones (D) y el discreto (LB)) se hacen a través de números sin dimensiones, o números independientes de la escala. Las soluciones a las ecuaciones incompresibles de Navier-Stokes, por ejemplo, dependen solamente de un parámetro adimensional, que es el llamado número de Reynolds (Re). De esta manera, los tres sistemas (P), (D) y (LB) se definen de manera que tengan el mismo número de Reynolds. La transición de (P) a (D) se realiza a través de la elección de una escala de longitud característica l_0 y una escala de tiempo t_0, y el paso de (D) a (LB) a mediante la elección de un paso discreto δ_x y del paso δ_t . He aquí una representación gráfica de esta relación:

$$\text{Sistema físico (P)} \overset{Re,l_0,t_0}{\longleftrightarrow} \text{Sistema adimensional (D)} \overset{Re,\delta_x,\delta_t}{\longleftrightarrow} \text{Sistema discreto (LB)}$$

Este no es el único acercamiento posible. Obviamente, se podría pasar directamente de (P) a (LB). Sin embargo, hay varias razones que motivan el paso intermedio y también para hacer explícito el sistema adimensional. En primer lugar, las variables discretas δ_x y δ_t son parámetros muy importantes que tienen influencia en la precisión y estabilidad de la simulación. No dependen de las escalas del sistema físico considerado, y ciertamente tampoco de la elección de las unidades físicas. Además, es muy probable que se caiga un día en una situación en la que no se tenga ningún sistema físico que sirva de referencia. Este es, por ejemplo, el caso cuando se intenta reproducir los resultados de un cálculo de referencia (benchmark) que fue obtenido por otro modelo distinto a lattice Boltzmann. Y finalmente, liberarse de las unidades físicas originales es una abstracción importante que predispone el ánimo para lo que se va a hacer, es decir, cálculos numéricos.

A.1. Ejemplo 1: Flujo de fluido incompresible

A.1.1. Sistema de Ecuaciones

En un fluido incompresible, la densidad toma un valor constante $\rho = \rho_0$ que no varía ni en el tiempo ni en el espacio. Las ecuaciones del movimiento, las ecuaciones de Navier-Stokes, se gobiernan por las leyes de conservación de la masa y del momento. La ley de conservación de masa establece que el campo de velocidades tiene divergencia nula:

$$\nabla_p \cdot u_p = 0. \tag{A.1}$$

Aquí, u es la velocidad, y el subíndice "p" indica que las variables y las derivadas se calculan en unidades físicas. La ley de conservación de la cantidada de movimiento conduce a la siguiente relación:

$$\partial_{t_p} u_p + (u_p \cdot \nabla_p) u_p = -\frac{1}{\rho_{0p}} \nabla_p p_p + \nu_p \nabla_p^2 u_p, \tag{A.2}$$

donde p_p es la presión y ν_p la viscosidad cinemática en unidades físicas.

A.1.2. Formulación adimensional

Las ecuaciones (A.1) y (A.2) ahora se re-escriben en forma adimensional. Para ello se introducen tanto una escala de longitud l_0 como una escala de tiempo t_0 que son representativas del flujo. La longitud l_0 podría representar, por ejemplo, el tamaño de un obstáculo que está sumergido en el fluido y t_0 podría ser el tiempo que necesita un "escalar pasivo" del fluido para recorrer una distancia l_0. Las variables físicas como el tiempo t_p y el vector de posición r_p, se sustituyen por su contrapartidas adimensionales:

$$t_d = \frac{t_p}{t_{0,p}} \qquad y \qquad r_d = \frac{r_p}{l_{0,p}} \tag{A.3}$$

De la misma manera, se introduce un cambio de unidades para las otras variables, basado en un análisis dimensional:

$$u_p = \frac{l_{0,p}}{t_{0,p}} u_d, \qquad \partial_{t_p} = \frac{1}{t_{0,p}} \partial_{t_d}, \qquad \nabla_p = \frac{1}{l_{0,p}} \nabla_d, \qquad y \qquad p_p = \rho_0 \frac{l_{0,p}^2}{t_{0,p}^2} p_d. \tag{A.4}$$

Usando este cambio de variables en las ecuaciones (A.1) y (A.2) conduce a la versión adimensional de las ecuaciones de Navier-Stokes:

$$\partial_{t_d} u_d + (u_d \cdot \nabla_d) u_d = -\nabla_d p_d + \frac{1}{Re} \nabla_d^2 u_d \tag{A.5}$$

$$\nabla_d \cdot u_d = 0, \tag{A.6}$$

donde el número de Reynolds, adimensional, se ha definido como

$$Re = \frac{l_0^2}{t_0 \nu} \tag{A.7}$$

evaluado en cualquier sistema de unidades. Dos flujos que obedecen a las ecuaciones de Navier-Stokes son equivalentes si están enmarcados en la misma geometría (excepto por un factor de escala) y tienen el mismo número de Reynolds.

Note que al expresar las variables de referencia en el sistema adimensional, se encuentra que $l_{0,d} = 1$ y $t_{0,d} = 1$. Para ayudar la intuición, se puede considerar al sistema adimensional como "el sistema en el que l_0 y t_0 son unitarios". También se observa que la viscosidad en el sistema adimensional es $\nu_d = \frac{1}{Re}$.

A.1.3. Discretización del sistema adimensional

El intervalo discreto de espacio δ_x se define como la longitud de referencia dividida por el número de celdas N utilizadas para discretizar esta longitud. Del mismo modo, δ_t se define como el tiempo de referencia dividido por el número de pasos de iteración N_{iter} necesarios para alcanzar este tiempo. Recuérdese que ambas variables de referencia son unitarias en el sistema adimensional. Así, los parámetros de discretización son

$$\delta_x = 1/N \qquad \text{y} \tag{A.8}$$
$$\delta_t = 1/N_{iter}. \tag{A.9}$$

Otras variables, como la velocidad y la viscosidad, se convierten fácilmente entre (D) y (LB) mediante un análisis adimensional:

$$u_d = \frac{\delta_x}{\delta_t} u_{lb} \qquad \text{y} \tag{A.10}$$
$$\nu_d \equiv \frac{1}{Re} = \frac{\delta_x^2}{\delta_t} \nu_{lb}, \tag{A.11}$$

y así,

$$u_{lb} = \frac{\delta_t}{\delta_x} u_d \qquad \text{y} \tag{A.12}$$
$$\nu_{lb} = \frac{\delta_t}{\delta_x^2} \frac{1}{Re}. \tag{A.13}$$

Finalmente, definiendo la velocidad de referencia $u_0 \equiv \frac{l_0}{t_0}$, se encuentra, por definición,

$$u_{0,d} = 1 \qquad \text{y} \tag{A.14}$$
$$\nu_{0,lb} = \frac{\delta_t}{\delta_x}. \tag{A.15}$$

A.1.4. Simulación del montaje.

Implementemos numéricamente un flujo 2D, confinado dentro de una caja de tamaño 3cm × 3cm, puesto en movimiento por la tapa superior, que se desplaza a la derecha a una velocidad de 2cm / min. La viscosidad del fluido es de 5 cm2 / min (mermelada de frambuesa).

1. Defina una longitud y un tiempo característicos. Dada la geometría del problema, tiene sentido definir una longitud de referencia l_0 como la longitud de la caja, y una velocidad de referencia u_0 como la velocidad de la tapa superior. Así, $l_{0,p} = 3$ cm, $u_{0,p} = 2$cm/min. El tiempo característico es $t_{0,p} = l_{0,p}/u_{0,p} = \frac{3}{2}$min.

2. Calcule el número de Reynolds: $Re = u_{0,p}l_{0,p}/\nu_p = 2 \cdot 3/5 = 6/5$.

3. Escoja los parámetros de discretización. Digamos que queremos una rejilla que consta de 101 × 101 nodos. Si la ubicación física de la frontera está sobre un nodo de rejilla (como es el caso, por ejemplo, con la condición de frontera Zou / He), el tamaño del sistema es de 100 × 100 en las variables de red. Por lo tanto, el espaciamiento de la malla discreta es $\delta_x = 1/100$. Además, elegimos una resolución de tiempo dada, digamos $\delta_t = 2\Delta 10^{-4}$.

4. Calcule el valor de las variables en la simulación. La velocidad $u_l b$, que se utiliza para implementar la condición de frontera en la tapa superior, se evalúa a partir de la ecuación (A.12). La viscosidad de la red ν_{lb} se calcula a partir de la ecuación (A.13). Si se utiliza el tiempo de relajación simple BGK, el tiempo de relajación se calcula a través de la relación $\tau = \nu_{lb}/c_s^2 + 1/2$, donde la velocidad del sonido es una constante de la red: $c_s^2 = 1/3$.

A.1.5. Discusión: ¿Cómo escoger δ_t?

No hay una forma directa e intuitiva de elegir δ_t. En otros esquemas numéricos diferentes al de LB, δ_t está a menudo vinculado con δ_x por consideraciones de estabilidad. En esquemas explícitos de paso temporal, es común utilizar la relación $\delta_t \approx \delta_x^2$ para mantener el modelo numéricamente estable.

En el método de lattice Boltzmann, δ_t y δ_x están conectados por una restricción diferente. De la discusión anterior, es claro que la velocidad, medida en unidades de la red, es $u_{0,lb} \sim \delta_t/\delta_x$. Incluso si se permite la posibilidad de que el fluido sea compresible, el valor de u_{lb} no puede ser mayor que la velocidad del sonido c_s, porque el modelo no soporta flujos supersónicos. Esto conduce a la restricción $\delta_t < \delta_x/\sqrt{3}$. Una restricción aún más fuerte aparece para la simulación de flujos incompresibles, es decir, cuando se desea explícitamente resolver la ecuación (A.2). El modelo LB es una solución de fluido casi compresible. Esto significa que entra en un régimen ligeramente compresible para resolver la ecuación de presión del fluido. Sin embargo, los efectos de compresibilidad afectan la precisión numérica. Como estos efectos varían como el cuadrado del número de Mach, Ma^2, se mantienen bajo control manteniendo el número de Mach bajo. El número de Mach no es otra cosa que $u_{0,lb}/c_s$, lo que significa que es proporcional a $u_{0,lb}$. Por lo tanto, el error de compresibilidad $\varepsilon(Ma)$ se escala, como $\varepsilon(Ma) \sim \delta_t^2/\delta_x^2$. Ahora imagine que aumenta la resolución de la red. La razón para hacer esto es muy probablemente para reducir el error numérico del modelo. Como el modelo LB es de segundo orden, el error de la red es $\varepsilon(\delta_x) \sim \delta_x^2$. Por supuesto, no tiene sentido la reducción del error de la red si después de esto el error de compresibilidad sigue presente y ahora es el importante. Por lo tanto, una decisión sensata es mantener ambos términos de error en el mismo orden: $\varepsilon(Ma) \sim \varepsilon(\delta_x)$. Esto conduce inmediatamente a la relación

$$\delta_t \sim \delta_x^2, \tag{A.16}$$

lo cual, no es de extrañar, es la misma restricción que se obtiene por otros sistemas explícitos.

En la práctica, un buen valor para empezar, en un tamaño de red de $N \sim 100$, es $u_{lb} \sim 0.02$. Si la exactitud de la simulación no es muy importante, puede aumentar este valor un poco para aumentar

δ_t, y así acelerar la simulación. Pero si empieza a aumentar la resolución de la red, nunca olvide escalar la resolución de tiempo impuesta por la ecuación (A.16), o no se obtiene ningún beneficio en absoluto de aumentar el número de nodos de la red.

APÉNDICE B

Apéndice: Manejo tensorial

En la notación tensorial es útil trabajar sólo con componentes. Un vector puede escribirse usando una sola letra que represente bajo sustitución la coordenada particular a tratar. Por ejemplo, podemos referirnos a un vector $\mathbf{u}$, escribiendo u_i. El subíndice i puede tomar los valores 1, 2 y 3, que en coordenadas cartesianas corresponderán a las componentes x, y y z. Para referirnos a la coordenada x del vector $\mathbf{u}$, simplemente escribimos u_1; para la coordenada y, u_2, etc.

También, en la notación tensorial, seguido es conveniente usar la *convención de sumas*. Esta convención dice que cuando existe un subíndice repetido, se está indicando que se debe hacer una suma en los valores 1, 2 y 3 para ese sufijo. El sufijo usado se considera un subíndice mudo porque no importa que letra sea, si está repetida indicará suma para ese sufijo. Esta notación es útil para ahorrarnos signos de suma. Por ejemplo, para la dilatación tenemos que $\Theta = \sum \epsilon_{ii} = \epsilon_{11} + \epsilon_{22} + \epsilon_{33}$. Usando la convención de sumas podemos escribir simplemente ϵ_{ii} o también, por ejemplo, ϵ_{kk}. Esta notación puede usarse también para derivadas, como por ejemplo en la ecuación

$$\rho \frac{\partial^2 u_i}{\partial t^2} = \frac{\partial \sigma_{ij}}{\partial x_j}. \tag{B.1}$$

Se presenta una tabla en la que se muestra la notación vectorial y la notación en índices para distintas operaciones. Los vectores se representan en negritas y los tensores con una doble raya debajo de ellos.

En la tabla B.1 aparece el tensor alternante ε_{ijk}, este se define

$$\varepsilon_{ijk} = \begin{cases} 0 & \text{si dos de los índices } i,\ j \text{ o } k \text{ son iguales,} \\ 1 & \text{si } i,\ j,\ k \text{ son distintos y aparecen en orden cíclico,} \\ -1 & \text{si } i,\ j,\ k \text{ son distintos y no están en orden cíclico.} \end{cases} \tag{B.2}$$

Por ejemplo, $\varepsilon_{112} = 0$, $\varepsilon_{231} = 1$, y $\varepsilon_{321} = -1$.

El tensor δ_{ij} es un tensor unitario, y se define de la siguiente manera

$$\delta_{ij} = \begin{cases} 0 & \text{si dos de los índices } i \neq j, \\ 1 & \text{si } i = j. \end{cases} \tag{B.3}$$

A δ_{ij} se le conoce también como el tensor de substitución, porque bajo contracción (hacer dos índices

de un tensor iguales y llevar a cabo la suma), corresponde a hacer una sustitución del subíndice que queda apareado de la delta con el que no quedó apareado.

Tabla B.1: *Notación en índices de relaciones tensoriales y vectoriales.*

Notación tensorial	**Notación en índices**
$\mathbf{v}$	v_i
$\underline{\underline{\sigma}}$	σ_{ij}
$\alpha = \mathbf{u} \cdot \mathbf{v}$	$\alpha = u_k v_k$
$\mathbf{w} = \underline{\underline{\sigma}} \cdot \mathbf{u}$	$w_i = \sigma_{ij} u_j$
$\mathbf{w}^T = \mathbf{v}^T \cdot \underline{\underline{\sigma}}$	$w_i = \sigma_{ji} v_j$
$\alpha = \underline{\underline{\sigma}} : \underline{\underline{\tau}}$	$\alpha = \sigma_{ij} \tau_{ij}$
$\mathbf{w} = \mathbf{u} \times \mathbf{v}$	$w_i = \varepsilon_{ijk} u_j v_k$
$\mathbf{u} = \nabla \alpha$	$u_i = \frac{\partial \alpha}{\partial x_i}$
$\alpha = \nabla \cdot \mathbf{u}$	$\alpha = \frac{\partial u_i}{\partial x_i}$
$\underline{\underline{e}} = \nabla \mathbf{u}$	$e_{ij} = \frac{\partial u_j}{\partial x_i}$
$\mathbf{v} = \nabla^2 \mathbf{u}$	$v_i = \frac{\partial^2}{\partial x_j \partial x_j} u_i$
$\mathbf{u} = \nabla \cdot \underline{\underline{\sigma}}$	$u_i = \frac{\partial}{\partial x_j} \sigma_{ji}$
$\underline{\underline{\tau}} = \underline{\underline{\sigma}} \cdot \underline{\underline{e}}$	$\tau_{ij} = \sigma_{ik} e_{kj}$
$\underline{\underline{A}} = \mathbf{uv}$	$A_{ij} = u_i v_j$

Referencias

[1] Ales J. Apostolidis, Adam P. Moyer, y Antony N. Beris. Non-Newtonian effects in simulations of coronary arterial blood flow. *Journal of Non-Newtonian Fluid Mechanics*, 233:155–165, 2016.

[2] P.L. Bhatnagar, E.P. Gross, y M. Krook. A model for collision processes in gases. I. Small amplitude processes in charged and neutral one-component systems. *Physical Review*, 94:511–525, 1954.

[3] R. L. Burden y J. D. Faires. *Numerical Analysis*. Brooks/Cole, 9 edition, 2011.

[4] A. Caiazzo. *Asymptotic analysis of lattice Boltzmann method for fluid-structure interaction problems*. PhD thesis, TU Kaiserslautern and Scuola Normale Superiore, Pisa, 2007.

[5] Hudong Chen, Shiyi Chen, y William H. Matthaeus. Recovery of the Navier-Stokes equations using lattice-gas Boltzmann method. *Physical Review A*, 45:5339–5342, 1991.

[6] S. Chen, H. Chen, Martinez, y Matthaeus. Lattice Boltzmann model for simulation of magnetohydrodynamics. *Physical Review Letters*, 67:3776–3780, 1991.

[7] S. Chen, D. Martinez, y R. Mei. On boundary conditions in lattice boltzmann methods. *Physics of Fluids*, 8:2517–2536, 1996.

[8] Siamak N. Doost, Liang , Zhong, Boyang Su, y Yosry S. Morsi. The numerical analysis of non-Newtonian blood flow in human patient-specific left ventricle. *Computer Methods and Programs in Biomedicine*, 127:232–247, 2016.

[9] O. Filippova y D. Hänel. Boundary fitting and local grid refinement for lattice-BGK models. *International Journal of Modern Physics C*, 9:1271–1279, 1998.

[10] U. Frisch, B. Hasslacher, y Y. Pomeau. Lattice-gas automata for the Navier-Stokes equations. *Physical Review Letters*, 56:1505–1508, 1986.

[11] I. Ginzbourg y P.M. Adler. Boundary flow condition analysis for the three-dimensional lattice Boltzmann model. *Journal de Physique*, 4:191–214, 1994.

[12] Z. Guo, C. Zheng, y B. Shi. An extrapolation method for boundary conditions in lattice Boltzmann method. *Physics of Fluids*, 14:2007–2010, 2002.

[13] J. Hardy, Y. Pomeau, y O. de Pazzis. Time evolution of a two-dimensional classical lattice system. *Physical Review Letters*, 31:276–279, 1973.

[14] Xiaoyi He y Li-Shi Luo. Theory of the lattice Boltzmann method: From the Boltzmann equation to the lattice Boltzmann equation. *Physical Review E*, 56:6811–6817, 1997.

[15] F. Higuera y G. Zanetti. Boltzmann approach to lattice gas simulations. *Europhysics Letters*, 9: 663–668, 1989.

[16] Leila Jahanshaloo, Nor Azwadi Che Sidik, Alireza Fazeli, y Mahmoud Pesaran H.A. An overview of boundary implementation in lattice Boltzmann method for computational heat and mass transfer. *International Communications in Heat and Mass Transfer*, 78:1–12, 2016.

[17] J.M.V.A. Koelman. A simple lattice Boltzmann scheme for Navier-Stokes fluid flow. *Europhysics Letters*, 15:603–607, 1991.

[18] T. Kruger. *Computer Simulation Study of Collective Phenomena in Dense Suspensions of Red Blood Cells under Shear*. Springer Spektrum, 1 edition, 2012.

[19] P. Lallemand. Theory of the lattice Boltzmann method: dispersion, dissipation, isotropy, Galilean invariance, and stability. *Physical Review E*, 61:6546–6562, 2000.

[20] J. Lätt. *Hydrodinamic Limit of Lattice Boltzmann Equations*. PhD thesis, Université de Genève, 2007.

[21] J. Lätt, B. Chopard, O. Malaspinas, M. Deville, y A. Michler. Straight velocity boundaries in the lattice Boltzmann method. *Physical Review E*, 77, 2008.

[22] G. McNamara y G. Zanetti. Use of a Boltzmann equation to simulate lattice-gas automata. *Physical Review Letters*, 61:2332, 1988.

[23] R. Mei, L.S. Luo, y W. Shyy. An accurate curved boundary treatment in the lattice Boltzmann method. *Journal of Computational Physics*, 155:307–330, 1999.

[24] R. Mei, L.S. Luo, P. Lallemand, y D. d'Humières. Consistent initial conditions for lattice Boltzmann simulations. *Computers & Fluids*, 35:855–862, 2006.

[25] D. A. Perumal y A. K. Dass. Simulation of flow in two-sided lid-driven square cavities by the lattice boltzmann method. *Advances in Fluid Mechanics VII, Oxford University Press*, 45-54, 2008.

[26] P.A. Skordos. Initial and boundary conditions for the lattice Boltzmann method. *Physical Review E*, 48:4823–4842, 1993.

[27] P. Van Leemput, K. Lust, y I.G. Kevrekidis. Coarse-grained numerical bifurcation analysis of lattice Boltzmann models. *Physica D: Nonlinear Phenomena*, 210(1-2):58–76, 2005.

[28] P. Van Leemput, M. Rheinländer, y M. Junk. Smooth initizalization of lattice Boltzmann schemes. *Computers and Mathematics with Applications*, 58:867–882, 2009.

[29] J. C.G. Verschaeve y B. Müller. A curved no-slip boundary condition for the lattice Boltzmann method. *Journal of Computational Physics*, 229:6781–6803, 2010.

[30] J.C.G. Verschaeve. Analysis of the lattice Boltzmann Bhatnager-Gross-Krook no-slip boundary condition: ways to improve accuracy and stability. *Physical Review E*, 80, 2009.

[31] Z. Yang. *Analysis of lattice Boltzmann boundary conditions*. PhD thesis, Universität Konstanz, 2007.