VNIVERSIDAD NACIONAL
AVTNOMA DE
MEXICO

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

PROGRAMA DE POSGRADO EN CIENCIA E INGENIERIA DE LA COMPUTACION

PARALELIZACION DEL ALGORITMO DE RETICULOS DE BOLTZMANN EN EL ESTUDIO DE LA DINAMICA DE UNA CELULA DE SANGRE ROJA EN UNA ARTERIOLA

TESIS
QUE PARA OPTAR POR EL GRADO DE:
MAESTRO EN CIENCIAS (COMPUTACION)

PRESENTA:
ROBERTO FEDERICO ORTIGOZA DOMINGUEZ

TUTOR:
SUEMI RODRIGUEZ ROMO
FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLAN

MÉXICO, D. F. MAYO 2013

Indice

Introducción.

Aparte de ser apasionante, la simulación computacional del flujo sanguíneo es un campo con mucho futuro y con grandes perspectivas ya que, además de ser un área de constante innovación tecnológica, promete a los investigadores médicos, biólogos, y demás científicos interesados en el tema, ampliar su conocimiento y entendimiento de los procesos físicos que ocurren en dicho fenómeno, amén de proporcionarles herramientas para analizar sistemáticamente y a detalle los mecanismos biomecánicos que ocurren en el desarrollo de ciertas enfermedades relacionadas con el flujo sanguíneo, inclusive en lugares y circunstancias en los que sería muy difícil acceder por otros medios que no sean la simulación computacional [1], [2] y [3]. Se espera que en los años venideros el constante crecimiento, depuración e innovación en esta área la haga madurar al punto de proporcionar puntos de vista ahora impensables, como de hecho ya esta ocurriendo en otras áreas del conocimiento humano.

Asimismo la simulación de todo tipo de procesos físicos y químicos mediante el *método de los retículos* o *redes de Boltzmann* [4] está ganando terreno y popularidad entre la comunidad científica animada por los excelentes resultados que ha entregado desde hace ya algunos años, y no es para menos pues se caracteriza por tener un costo computacional bajo, una relativamente sencilla implementación y flexibilidad para adaptar el método a una gran variedad de problemas reales, eso sin contar con que carece de los problemas de ruido y otros que acarrearon sus antecesores, los autómatas celulares, que hay mencionarlo, aún éstos están lejos de haberse explorado en su totalidad, por lo que aún queda mucho por hacer en el entendimiento y el horizonte de aplicación **[5]**.

El presente trabajo se presenta como un primer intento en la simulación de una célula de sangre roja en una arteriola que sirva como base para una expansión futura a la simulación completa del flujo sanguíneo que incluya la mayoría de sus componentes principales y muestre a detalle los mecanismos interactivos entre los mismos, y esto para distintas geometrías y distintas escalas de arterias. Asimismo se estudia el movimiento de una célula de sangre roja en una arteriola dada, determinando y analizando las variables hidrodinámicas relevantes que intervienen en el proceso mediante la implementación de una arquitectura de paralelización que consta de una combinación de las *Unidades Gráficas de Proceso* (GPUs por sus siglas en inglés) y el estándar basado en el paso de mensajes, MPI, donde se trató de aprovechar la intrínseca localidad del método de los retículos de Boltzmann para paralelizar la parte esencial de un programa secuencial basado en el método de manera casi natural.

Se espera que este trabajo ayude al lector interesado en una modesta introducción al método de las redes de Boltzmann y su aplicabilidad a la simulación de procesos, sugiriéndole que no se desanime por la inicial complejidad que note conforme avance en su estudio considerando que aún los ahora expertos tuvieron las mismas (y quizá más) dificultades al inicio de su camino como sucede en muchos aspectos de la vida misma.

Objetivo.

Diseño y construcción de un programa informático que utilizará una arquitectura computacional paralela para simular el movimiento de una célula de sangre roja en una arteriola, con la finalidad de estudiar la dinámica de dicho sistema que tenga como objetivo último, coadyuvar a los avances en la investigación médica para descubrir las causas mecánicas de distintas enfermedades de tipo circulatorio, por ejemplo. Y finalmente realizando lo anterior mediante el exitoso método de las redes de Boltzmann.

Metas.

- Diseñar y construir un programa informático que simule el movimiento de una célula de sangre roja en una arteriola, y mediante el cual se puedan extraer datos relevantes acerca de la dinámica del sistema célula + vena. El algoritmo de simulación (especialmente la parte hidrodinámica de la misma) estará basado en el método de las redes de Boltzmann.

- Diseñar y construir una arquitectura computacional novedosa y eficiente con el fin de paralelizar el algoritmo de simulación anterior utilizando tecnologías de paralelización disponibles.

- Realizar los experimentos que se consideren pertinentes para comprobar la validez de la simulación así como para garantizar su eficiencia.

Hipótesis de Trabajo.

Se puede simular computacionalmente mediante el método de las redes de Boltzmann el movimiento de una célula de sangre roja en un microvaso, por ejemplo en una arteriola, y en consecuencia estudiar la dinámica de dicha célula, considerándose el sistema como una partícula inmersa en un fluido Newtoniano. Asimismo, es factible la paralelización eficiente del programa construido mediante una arquitectura combinada de cómputo que conste de la Arquitectura de Dispositivos de Cómputo Unificado y el estándar de la Interfaz de Paso de Mensajes.

Alcances.

Aunque el fin último del presente trabajo es, como ya se mencionó, ayudar a aumentar el conocimiento y entendimiento del fenómeno del flujo sanguíneo humano, tiene como finalidad inmediata hacer un aporte principalmente en el área de las ciencias computacionales por lo que, aún cuando el algoritmo de simulación que utiliza las redes de Boltzmann es original, esta fuertemente basado en el propuesto por [1], entre otros autores. Asimismo la parte mecánica de la simulación se puede considerar como rústica en el sentido de que no se utilizaron técnicas avanzadas para simular con exceso de detalle el movimiento de la célula, sino más bien se toma como base el movimiento resultante de aplicar la suma de las fuerzas involucradas sobre el centro de masa aplicando las leyes de la mecánica clásica, esto en parte debido al interés primordial mencionado al principio de éste párrafo.

Notas Respecto a las Referencias Bibliográficas.

Dada la importancia que tienen para efectos de respetar en lo posible los derechos de autor, se especifica que cuando la referencia sea respecto a una frase o concepto exactas, la misma se escribirá en negritas, cuando sea respecto a todo un contexto, se mencionará la frase “lo anterior basado en” o “lo anterior de” inmediatamente antes de la referencia de los autores principales, y se referirá a todo el texto anterior a la referencia hasta antes de la última referencia bibliográfica mencionada, cuando no se mencione ninguna de las frases anteriores antes de la referencia de los autores principales, la misma se refiere al texto anterior desde el inicio del párrafo donde se encuentre, no se toman en cuenta las referencias incluidas en los pies de nota, los demás casos se entenderán dependiendo del contexto en que se hallen.

Capítulo 1

Flujo Sanguíneo y Redes de Boltzmann.

1.1 Flujo Sanguíneo.

El entendimiento del flujo sanguíneo en geometrías complejas, desde las arterias coronarias a las venas microcapilares ha atraído la atención desde hace centurias, comenzando con las investigaciones de Leonardo da Vinci **[2]**. Y esto no ha sido en vano pues el estudio y la comprensión de los procesos (por ejemplo el proceso de adhesión de los leucocitos en áreas de inflamación o patología en el cuerpo) que ocurren en dicho flujo permitirían el desarrollo de nuevas estrategias para el tratamiento a tiempo de diversas enfermedades relacionadas tales como la arterioesclerosis, la arterogénesis, la artritis y el cáncer [2] y [6]. Los estudios *in vivo*, *in vitro*[1] y las simulaciones por computadora proveen una valiosa fuente de información acerca del funcionamiento de los sistemas biológicos en general, sin embargo los estudios *in vivo* están sujetos a consideraciones éticas y los datos obtenidos mediante los estudios *in vitro* son limitados debido a, por ejemplo, restricciones de acceso óptico a las áreas de interés [3].

Los ingentes avances en la tecnología (que incluye la miniaturización, el incremento en la capacidad de procesamiento y de almacenamiento), en la ciencia de la computación y las técnicas computacionales, han permitido que sea cada vez más precisa y confiable la información que las simulaciones computacionales pueden proveer, asimismo gracias al continuo avance en las técnicas de paralelización, la obtención de dicha información es cada vez más rápida y eficiente. Por lo que se intuye que el avance que se dará en el campo de la simulación de sistemas biológicos será cada vez más vertiginoso.

1.2 Ecuación de Boltzmann.

[1] *In vivo* significa que la investigación y la experimentación se hace en un organismo vivo, en cambio *in vitro* significa que dicha investigación y experimentación se hace fuera de un organismo vivo, aislando una parte del mismo dentro de un ambiente artificial controlado (v.g. un tubo de ensayo) [W1], [W2] y [W3].

En este contexto el método de las redes de Boltzmann, que fue inicialmente utilizada para calcular los coeficientes de viscosidad de los Autómatas en Redes Celulares de Gas por Wolfram (1986) y Frish et al. (1987) y que fue presentada como método de cálculo numérico para la simulación de flujo de fluidos por MacNamara y Zanetti (1988) en donde se suprimían variaciones fluctuantes en los resultados que arrojaban los autómatas celulares substituyendo los campos booleanos por funciones de distribución [5], es una metodología que aparte de abordar problemas que involucren flujo de fluidos en geometrías complejas a un costo computacional bajo y de describir muchos fenómenos que ocurren en sistemas fluídicos como los flujos laminares, los solitones, muy probablemente la turbulencia (a bajos números de Reynolds), etc. [7] y [8], ha mostrado ser eficiente y útil para solucionar diversos problemas que involucren sistemas fluídicos multicomponente y multifase [8] y [9]: desde la liberación de nanopartículas de microcápsulas móviles hasta la simulación del flujo a través de colonias de coral, pasando por la simulación de flujo sanguíneo en las redes microvasculares, la disipación térmica en una placa metálica, etc.

La metodología de las redes de Boltzmann se ha comparado con experimentos, Dinámica Stokesiana[2], y métodos de diferencias finitas obteniéndose resultados acordes con los que arrojan éstos [10].

A diferencia de los métodos numéricos tradicionales tales como el método de las diferencias finitas, volumen finito, etc. los cuales se basan en el cálculo de las variables hidrodinámicas mediante la discretización de las ecuaciones continuas de movimiento del fluido, el método de las redes de Boltzmann o *lattice de Boltzmann* (MRB) se funda en un enfoque mecánico estadístico, es decir se pregunta ¿por qué no en lugar de considerar toda la información que puede haber en un sistema de 10^{23} partículas (por ejemplo un gas), mejor se considera la evolución temporal del sistema mediante la "interacción" de los promedios estadísticos de partículas[3] que puede haber en una región determinada del espacio, con una característica hidrodinámica dada y en un intervalo de tiempo determinado? [4], es decir parte de que el comportamiento macroscópico de un fluido es el resultado de la suma de las interacciones microscópicas de las muchas partículas que conforman el sistema.

En 1872 el insigne científico Austriaco Ludwig Boltzmann (1844 - 1906) desarrolló la ecuación que lleva su nombre, la cual fue una de sus muchas contribuciones al campo de la mecánica estadística.

Consideremos un fluido como un sistema de *N* partículas todas de masa *m*, donde la trayectoria libre media de las mismas es muy grande comparada con la distancia a la cual las partículas interactúan (colisionan) entre si (por ejemplo las que componen a un gas diluido) y que evoluciona en el tiempo mediante la colisiones que suceden entre las mismas y sujeto quizá a la influencia de un agente externo,

[2] Es una técnica de solución para la ecuación acoplada de N cuerpos de Langevin, la cual describe mecánicamente el movimiento browniano de partículas suspendidas en un fluido Newtoniano incompresible [11].

[3] En este contexto se refiere a la cantidad $1/m\ f(\mathbf{x}, \mathbf{v}, t)\ \Delta^3 x \Delta^3 v$.

podemos entonces derivar la ecuación de Boltzmann definiendo primeramente un *espacio fase* [12] para nuestro sistema compuesta por la posición **x** y la velocidad **v** (recuerde que hay partículas de gas moviéndose, por lo que se les puede asociar una cierta velocidad), a continuación podemos considerar la existencia de una función de densidad de probabilidad definida en dicho espacio fase más el tiempo f (**x**, **v**, t), tal que el número probable de partículas n en una localidad entre **x** y **x** + Δ_1**x**, con una velocidad entre **v** y **v** + Δ_1**v** y en un tiempo t es f (**x**, **v**, t) $\Delta^3{}_1x\Delta^3{}_1v$ (donde el volumen $\Delta^3{}_1x\Delta^3{}_1v$ es igual a $\Delta_1x_x\,\Delta_1x_y\,\Delta_1x_z\,\Delta_1v_x\,\Delta_1v_y\,\Delta_1v_z$) [4].

Supongamos ahora que aplicamos una fuerza externa **F** a este probable conjunto de partículas y que no colisionan entre si al aplicarla, entonces en un tiempo Δt posterior tendremos el mismo número probable de partículas pero para la localidad entre **x** + **a**Δt = **x** + (Δ**x**/Δt)Δt = **x** + Δ**x** y **x** + Δ**x** + Δ_1**x**, con la velocidad entre **v** + (**F**/m) Δt = **v** + (Δ**v**/Δt) Δt = **v** + Δv y **v** + Δ**v** + Δ_1**v**, y en el tiempo t + Δt, lo que significa que:

$$f(\mathbf{x}+\Delta\mathbf{x},\mathbf{v}+\Delta\mathbf{v},t+\Delta t)\Delta^3{}_1x\Delta^3{}_1v = f(\mathbf{x},\mathbf{v},t)\Delta^3{}_1x\Delta^3{}_1v \qquad (1.1)$$

Para visualizar mejor lo anterior imagínese un conjunto de partículas contenido en una área del espacio fase compuesto sólo por las direcciones X de la posición y la velocidad, $\Delta_1x_x\Delta_1v_x$ (con cada partícula situada en un punto dentro de dicha área entre x_x y x_x + Δ_1x_x, y entre v_x y v_x + Δ_1v_x), entonces, con la aplicación de una fuerza externa, cada partícula del conjunto se moverá, en el tiempo t + Δt, a una localidad dentro del área del espacio fase formada por los puntos (x_x + Δx_x, v_x + Δv_x), (x_x + Δx_x, v_x + Δv_x + Δ_1v_x), (x_x + Δx_x + Δ_1x_x, v_x + Δv_x + Δ_1v_x), y (x_x + Δx_x + Δ_1x_x, v_x + Δv_x), (nótese que Δx_x y Δ_1x_x no necesariamente deben coincidir, ya que la primera se refiere al *desplazamiento* del área de espacio fase, en tanto que la segunda es la *longitud* de dicha área en la coordenada de la posición espacial, lo mismo aplica para Δv_x y Δ_1v_x), donde esto es justamente lo que se esta expresando en la ecuación (1.1) pero para el caso de tres dimensiones.

Si consideramos ahora las colisiones observamos que al término f (**x**, **v**, t) $\Delta^3{}_1x\Delta^3{}_1v$ debemos sumar y restar la masa de aquellas partículas que para la nueva posición del área del espacio fase lleguen desde otras colisiones o se vayan hacia otras distribuciones en el tiempo Δt, llamemos a la tasa de cambio temporal de esta ganancia menos pérdida de partículas $C\Delta^3{}_1x\Delta^3{}_1v$, siendo C una constante a la cual se le conoce como el *término de colisión*, entonces tenemos:

$$f(\mathbf{x}+\Delta\mathbf{x},\mathbf{v}+\Delta\mathbf{v},t+\Delta t)\Delta^3{}_1x\Delta^3{}_1v = f(\mathbf{x},\mathbf{v},t)\Delta^3{}_1x\Delta^3{}_1v + C\Delta^3{}_1x\Delta^3{}_1v\Delta t \qquad (1.2)$$

reacomodando términos y tomando el límite cuando $\Delta t \rightarrow 0$ tenemos, de (1.2):

[4] También se podría ver la función de distribución como la densidad probable de partículas (por unidad de volumen del espacio fase), $f(\mathbf{x},\mathbf{v},t) = \frac{n}{\Delta^3{}_1x\Delta^3{}_1v}$, para partículas que se encuentran entre (**x**, **v**) y (**x**+ Δ_1**x**, **v** + Δ_1**v**) **[12]**.

$$\lim_{\Delta t\to 0}\frac{f(\mathbf{x}+\Delta\mathbf{x},\mathbf{v}+\Delta\mathbf{v},t+\Delta t)-f(\mathbf{x},\mathbf{v},t)}{\Delta t}=\lim_{\Delta t\to 0}\frac{C}{\Delta t}\Delta t \qquad (1.3)$$

$$\frac{df(\mathbf{x},\mathbf{v},t)}{dt}=\frac{Df(\mathbf{x},\mathbf{v},t)}{Dt}=C$$

donde el símbolo $\frac{D}{Dt}$ denota la derivada material o total respecto a *t* y es igual a:

$$(\frac{d\mathbf{x}}{dt}\cdot\frac{\partial}{\partial\mathbf{x}}+\frac{d\mathbf{v}}{dt}\cdot\frac{\partial}{\partial\mathbf{v}}+\frac{\partial}{\partial t})f(\mathbf{x},\mathbf{v},t)=C \qquad (1.4)$$

identificando $d\mathbf{x}/dt = \mathbf{v}$ y $d\mathbf{v}/dt = \mathbf{a} = \mathbf{F}/m$ y substituyendo en la ecuación anterior, tenemos:

$$(\mathbf{v}\cdot\frac{\partial}{\partial\mathbf{x}}+\frac{\mathbf{F}}{m}\cdot\frac{\partial}{\partial\mathbf{v}}+\frac{\partial}{\partial t})f(\mathbf{x},\mathbf{v},t)=C \qquad (1.5)$$

con el término de colisión aún por determinar.

La ecuación (1.5) es la ecuación cinética para la función de distribución de *una partícula individual* a la cual se le conoce como *ecuación de transporte de Boltzmann,* según [13] o simplemente *ecuación de Boltzmann* [4] (pueden existir funciones de distribución para dos o más partículas f_{12}, f_{123}, etc., tales que, por ejemplo $f_{12}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{v}_1, \mathbf{v}_2, t)$ es la función de densidad de probabilidad de que dos partículas se encuentren, una en una localidad entre $\mathbf{x}_1$ y $\mathbf{x}_1 + \Delta_1\mathbf{x}_1$, con una velocidad entre $\mathbf{v}_1$ y $\mathbf{v}_1 + \Delta_1\mathbf{v}_1$, y la otra en una localidad entre $\mathbf{x}_2$ y $\mathbf{x}_2 + \Delta_1\mathbf{x}_2$, con una velocidad entre $\mathbf{v}_2$ y $\mathbf{v}_2 + \Delta_1\mathbf{v}_2$, ambas en un tiempo *t*).

Debido a que la determinación del operador de colisión *C* origina considerar la serie interminable de distribuciones de probabilidad f_{12}, f_{123}, etc., conocida como la jerarquía BBGYK después de Bogoliubov, Born, Green, Yvon y Kirkwood quienes la establecieron entre 1945 y 1946, Boltzmann hizo las siguientes consideraciones para truncar la jerarquía BBGYK:

- Se considera que el sistema a analizar es un gas diluido donde las partículas que lo componen son puntos (aunque con masa).

- Dichas partículas interactúan sólo mediante colisiones binarias (entre dos partículas solamente) vía un potencial interpartícula de corto alcance, es decir la distancia a la cual las partículas pueden interactuar es muy pequeña comparada con la trayectoria libre media de las mismas, por lo que las velocidades de las partículas no están relacionadas entre si antes de la colisión, y después de la misma sólo por virtud de la conservación de la masa (a esta suposición se le conoce como *caos molecular* o *Stosszahlansatz*).

- Ninguna fuerza externa al sistema de partículas influye en las colisiones entre ellas.

Bajo estas suposiciones se tiene el siguiente término de ganancias menos pérdidas que, para las funciones de distribución, solo depende de las funciones de distribución binaria de las dos partículas que intervienen en la colisión, f_{12} y $f_{1'2'}$ (correspondientes a las funciones de distribución binaria de las partículas 1 y 2 antes y después de que ocurra la colisión):

$$C_{12} \equiv G - P = \int\int (f_{1'2'} - f_{12}) \left|\mathbf{v}_1 - \mathbf{v}_2\right| \sigma(\Omega) d\Omega d\mathbf{v}_2 \qquad (1.6)$$

todo lo anterior según [4] y [5], donde $\mathbf{v}_1$ y $\mathbf{v}_2$ son las velocidades de las partículas 1 y 2 respectivamente antes de colisionar y $\sigma(\Omega)$ es la sección transversal diferencial que definimos como aquella sección de área transversal que expuesta a un haz de partículas a colisionar, éstas serán dispersadas (debido a la colisión) en un correspondiente ángulo sólido de dispersión Ω, el cuál esta formado por el ángulo de dispersión θ, una $d\theta$ y el ángulo azimutal ϕ [4] y [W4]. En la Figura 1 se muestran esquemáticamente dichos ángulos ejemplificados mediante dos partículas, 1 y 2 (que formarían parte del haz de partículas a colisionar), que colisionan entre si. La figura fue realizada, aparte de las dos últimas fuentes, en base a lo mostrado en [14].

En la naturaleza observamos como diferentes sistemas, aún con la acción de fuerzas o agentes externos tienden a un cierto equilibrio. Para nuestro sistema en partícular supondremos que partirmos de que las partículas del gas se encuentran inicialmente en un estado de no equilibrio y veremos si el modelo matemático propuesto en las ecuaciones anteriores puede evolucionar con el tiempo bajo ciertas condiciones, a un estado de equilibrio, para ello primero se menciona que pueden distinguirse dos tipos de equilibrio: el equilibrio local para un punto en el espacio, en el cual las variaciones temporales y espaciales de variables como la temperatura, la presión o la velocidad de un cierto vecindario alrededor de dicho punto son prácticamente nulas y el equilibrio global en el cual los valores de las mismas variables son las mismas para todo el sistema. Para nuestro caso el equilibrio local es definido como un estado tal que existe una función de distribución local f^{eq}, en donde las ganancias y las pérdidas de partículas debido a colisiones son las mismas de modo que el término de colisión desaparece:

$C(f^{eq}, f^{eq}) = 0$

según [4]. Ahora bien, en la ecuación (1.6) observamos que el término de colisión involucra la diferencia entre las funciones de distribución de dos partículas $f_{1'2'}$ después de que ha ocurrido la colisión y f_{12} antes de que haya ocurrido la colisión, bajo la suposición Stosszahlansatz Boltzmann asume que $f_{12} = f_1 f_2$, y lo mismo para $f_{1'2'}$ por lo que para el estado de equilibrio local, de (1.6) tenemos:

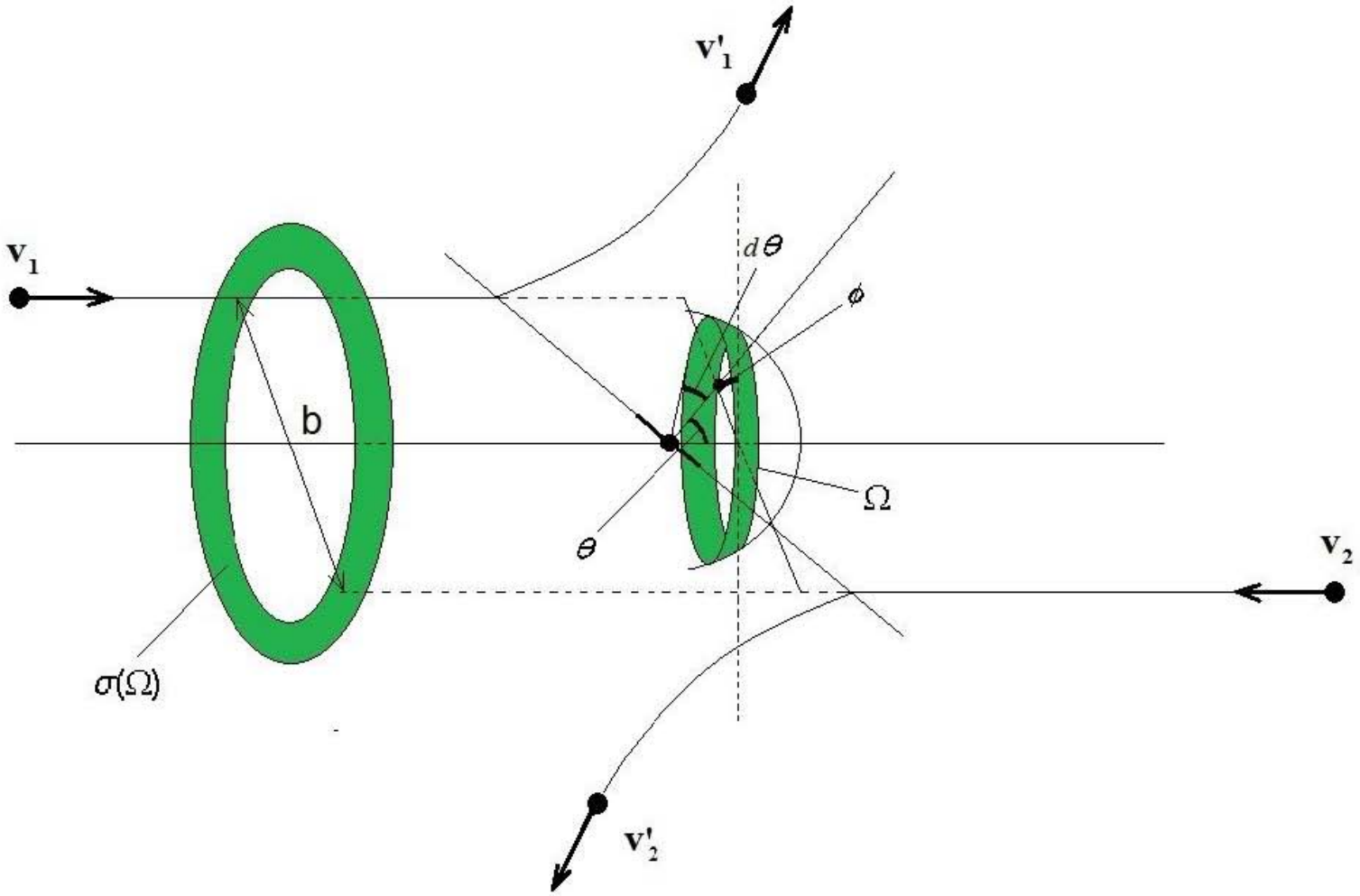


Figura 1. Esquema de colisión entre dos partículas, 1 y 2, con velocidades iniciales $\mathbf{v}_1$ y $\mathbf{v}_2$ y velocidades finales $\mathbf{v'}_1$ y $\mathbf{v'}_2$ antes y después de la colisión, respectivamente. Nótese la sección del plano de colisión compuesto por el segmento de recta b, las líneas punteadas y la prolongación de uno de los lados del ángulo ϕ. Fuente: elaboración propia.

$$\int\int (f_{1'} f_{2'} - f_1 f_2)\left|\mathbf{v}_1 - \mathbf{v}_2\right|\sigma(\Omega)d\Omega d\mathbf{v}_2 = 0$$

lo que significa que,

$$f_{1'} f_{2'} - f_1 f_2 = 0$$

$$f_{1'}\, f_{2'} = f_1\, f_2 \qquad (1.7)$$

si asumimos que nunca ocurre que $\mathbf{v}_1 = \mathbf{v}_2$ para todas las colisiones de nuestro sistema[5]. Si tomamos logaritmos naturales en la ecuación anterior resulta:

$$\ln f_{1'} + \ln f_{2'} = \ln f_1 + \ln f_2 \qquad (1.8)$$

Ahora bien, Cercignani (1988) mostró que existen funciones de distribución f tales que cuando se substituyen en el término de colisión lo anulan, siendo la forma de estas funciones:

$$f(\mathbf{v}) = exp(a_1 + \mathbf{a}_2 \cdot \mathbf{v} + a_3 v^2) \qquad (1.9)$$

se observa que los términos a_1, $\mathbf{a}_2 \cdot \mathbf{v}$ y $a_3 v^2$ son combinaciones lineales de las *invariantes de colisión dinámicas* 1, $m\mathbf{v}$, $mv^2/2$ (conservación de masa, momento y energía cinética, la suma de valores para ambas partículas no cambian después de la colisión). Si aplicamos logaritmo natural a la ecuación (1.9) tenemos:

$$\ln f(\mathbf{v}) = a_1 + \mathbf{a}_2 \cdot \mathbf{v} + a_3 v^2 \qquad (1.10)$$

lo cual es congruente con (1.8), pues de ésta ecuación observamos que el ln f es una invariante de colisión a nivel microscópico (la suma de los logaritmos no cambia después de la colisión), y de (1.10) observamos que ln f es función de las invariantes de colisión dinámicas mencionadas.

La *distribución de Maxwell* es un caso especial entre las posibles funciones de distribución dadas por (1.9):

$$f^{eq} = \frac{\rho}{m(2\pi RT)^{D/2}} e^{-\frac{(\mathbf{v}-\mathbf{u})^2}{2RT}} \qquad (1.11)$$

para D dimensiones espaciales, donde ρ es la densidad del fluido para todo el rango de velocidades:

$$\rho = m\int f(\mathbf{x}, \mathbf{v}, t)d\mathbf{v}, \qquad (1.12)$$

[5] Nótese que bajo la suposición de Stosszahlansatz asumimos que no hay relación entre las velocidades de la partícula antes de que ocurra la colisión, más no hemos dicho nada después de que ha ocurrido la colisión (en realidad Cercignani (1988) menciona que bajo condiciones adicionales, también asumidas para la derivación de la ecuación de Boltzmann, la suposición Stosszahlansatz sigue siendo válida aún para este caso).

donde *R* es la constante específica de los gases[6], *T* la temperatura del fluido, y **u** es la velocidad macroscópica del fluido:

$$\mathbf{u} = \frac{m}{\rho}\int \mathbf{v} f(\mathbf{x},\mathbf{v},t)d\mathbf{v} \quad [7] \qquad (1.13)$$

Es necesario considerar que en 1872 Boltzmann propuso que la cantidad:

$$H(t) = -\int\int f(\mathbf{x},\mathbf{v},t)\ln(f(\mathbf{x},\mathbf{v},t))d\mathbf{v}d\mathbf{x} \qquad (1.14)$$

satisface la siguiente ecuación:

$$\frac{dH}{dt} \geq 0 \qquad (1.15)$$

donde H(t) representa la entropía del sistema y f (**x**, **v**, t) es cualquier distribución que satisface la ecuación (1.5), el signo de la igualdad se alcanza cuando f (**x**, **v**, t) es la distribución f^{eq} siempre que la velocidad y la temperatura macroscópicas sean constantes espacialmente y exista equilibrio local. A lo anterior se le llama el *teorema H.*

La ecuación (1.15) expresa el hecho de que la entropía del sistema siempre aumenta conforme el tiempo aumenta y alcanza su valor máximo cuando la distribución f (**x**, **v**, t) es igual a la función de distribución f^{eq} bajo las condiciones mencionadas, es decir cuando el sistema alcanza el estado de equilibrio global. Nótese que, dado que el teorema H se cumple para cualquier función que satisfaga la ecuación (1.5), entonces cualquier representación matemática basada en dicha ecuación y que se proponga como modelo para la evolución de nuestro sistema debe cumplir también con dicho teorema, todo lo anterior de [4] y [5].

Dado que el término de colisión es una expresión integral que no ayuda al cómputo numérico, se han propuesto varias aproximaciones, entre las cuales la más conocida es la BGK, propuesta por Bhatnagar, Gross y Krook (1954) y que ha resultado ser muy útil y eficiente para utilizarse en, como ya se mencionó, una amplia gama de problemas de transporte en flujo de fluidos (incluso para hallar, en algunos casos, soluciones analíticas a los mismos), aparte de que simplifica enormemente el término de colisión, lo anterior de [4], [5] y [8].

---

[6] También se puede escribir la distribución *Maxweliana* utilizando la constante de Boltzmann, k_B, la cual es un puente entre la física microscópica y la macroscópica que relaciona la energía de una partícula individual y la temperatura, y que se relaciona con *R* mediante $R = \frac{k_B N_A}{M} = \frac{k_B}{m}$. donde N_A es el número de avogadro y M la masa molar de la partícula.

[7] [4] También proporciona la expresión para la energía interna del gas por unidad de volumen (densidad de energía), $\rho e = m\int f \frac{v^2}{2} d\mathbf{v}$.

La simplificación BGK propone que, bajo la suposición de Stosszahlansatz, y considerando que en un gas diluido la diferencia entre la distribución de probabilidad actual y la distribución de probabilidad del estado de equilibrio es muy pequeña, f (**x**, **v**, t) $- f^{eq}$ (**x**, **v**, t) << f^{eq} (**x**, **v**, t), la distribución resultante en el siguiente paso de tiempo será la distribución en el paso de tiempo actual más una cantidad proporcional a la diferencia mencionada, de la ecuación (1.2) se observa que entonces el término de colisión sería, en cada paso de tiempo, proporcional a ésta diferencia (siendo esta diferencia lo que le falta a la distribución f (**x**, **v**, t) para llegar o *relajarse* a la distribución de equilibrio f^{eq} (**x**, **v**, t)) y utilizando una constante[8] para todos los pasos, por lo que el término de colisión queda de la siguiente manera:

$$C = -\frac{f - f^{eq}}{\tau} \tag{1.16}$$

donde τ es dicha constante de proporcionalidad a la que se le llama el *tiempo de relajación,* lo anterior de [15] y [4]. De acuerdo con [5] la aproximación BGK hace que una función de distribución f (**x**, **v**, t), que cumpla con la ecuación (1.5), tienda con el tiempo a la distribución Maxweliana, lo cual esta de acuerdo con el teorema H.

Substituyendo la aproximación BGK para el término de colisión en la ecuación (1.5), se obtiene:

$$(\mathbf{v}\cdot\frac{\partial}{\partial\mathbf{x}}+\frac{\mathbf{F}}{m}\cdot\frac{\partial}{\partial\mathbf{v}}+\frac{\partial}{\partial t})\,f(\mathbf{x},\mathbf{v},t) = -\frac{f(\mathbf{x},\mathbf{v},t) - f^{eq}(\mathbf{x},\mathbf{v},t)}{\tau} \tag{1.17}$$

1.3 Ecuación de Navier-Stokes.

Las ecuaciones de Navier-Stokes representan la forma diferencial de la conservación del momento lineal y describen el movimiento de una partícula del fluido en cualquier instante de tiempo.

La ecuación vectorial para un fluido supuesto incompresible es:

$$\frac{\partial\mathbf{V}}{\partial t} + (\mathbf{V}\cdot\nabla)\mathbf{V} = \mathbf{a}_F - \frac{1}{\rho}\nabla p + \nu\nabla^2\mathbf{V} \tag{1.18}$$

obsérvese que las dimensiones de cada término son L/T^2.

[8] [4] Indica que en realidad el fenómeno de la relajación de un estado de no equilibrio a un estado de equilibrio en el flujo de un fluido ocurre a varias escalas de tiempo (que se verán más adelante) y que la variable temporal representativa de dicha relajación τ varía de acuerdo a estas escalas por lo que escoger τ como una constante equivale a englobar todas estas escalas de tiempo en un solo valor.

Los dos términos del lado izquierdo de la igualdad son la derivada material o total de la velocidad respecto del tiempo, donde dicha derivada material esta compuesta de la derivada temporal $\partial \mathbf{V} / \partial t$ de una partícula de fluido en una posición fija del espacio, y la derivada *convectiva* (convección en este caso se refiere al cambio posicional de un punto a otro en el espacio) $(\mathbf{V} \cdot \nabla)\mathbf{V}$ la cual nos indica la variación de la velocidad respecto al cambio de posición para un tiempo fijo determinado, cabe notar que es en base a este término que pueden considerarse las perturbaciones en el flujo debido a la turbulencia en la ecuación de Navier Stokes. El primer término del lado derecho de la igualdad $\mathbf{a}_F$ representa la aceleración debida a las fuerzas externas que en este caso es la gravedad; el término $-1/\rho \nabla p$ es la aceleración debida a la presión actuante en el flujo, el último término $\nu \nabla^2 \mathbf{V}$ es la desaceleración o resistencia al flujo que sufre el fluido debido a la acción de fuerzas viscosas en él, por último ν es la viscosidad cinemática del fluido, lo anterior de [16].

Como ya se mencionó, las ecuaciones de Navier-Stokes junto con las leyes de conservación de la masa modelan la mecánica de mucho tipos de fluidos de manera adecuada (por ejemplo los flujos estacionarios, laminares y algunos con turbulencia, compresibles o incompresibles), y sirven para resolver muchos problemas de flujo de fluidos.

1.4 Redes de Boltzmann.

Lo presentado en esta sección esta basado en lo similar mostrado en [5].

Supongamos que queremos discretizar la ecuación (1.17), para ello, podemos considerar un conjunto de velocidades discretas, $\mathbf{v}_i$, y un conjunto de funciones de distribución asociadas, f_i ($\mathbf{x}$, t), que estarán regidas por (despreciando por el momento el término de la fuerza externa) la *ecuación de Boltzmann discreta*: [9]

$$(\mathbf{v}_i \cdot \frac{\partial}{\partial \mathbf{x}} + \frac{\partial}{\partial t}) f_i(\mathbf{x},t) = -\frac{f_i(\mathbf{x},t) - f_i^{eq}(\mathbf{x},t)}{\tau} \qquad (1.19)$$

También consideremos que nuestro sistema, que será el espacio físico donde se encuentra el fluido, es discretizado como una red o retículo compuesto como una serie de puntos o nodos de la red y en donde las distribuciones f_i "viajan" o se propagan sólo a través de los enlaces que unen entre si todos los nodos, colisionando solamente en los nodos con las distribuciones entrantes a los mismos. La forma en que los enlaces unen entre si todos los nodos está dada por el conjunto de velocidades discretas $\mathbf{v}_i$ seleccionadas (en la siguiente sección se mencionarán

[9] Es necesario mencionar que las unidades de f_i ($\mathbf{x}$, t) serán ahora las de f ($\mathbf{x}$, $\mathbf{v}$, t) x v, es decir l^{-3}, l representa unidades de longitud.

algunas formas conocidas de redes que dan nombre a los distintos *modelos de redes de Boltzmann*).

Ahora bien, podemos discretizar a su vez la ecuación (1.19) mediante diferencias finitas divididas hacia atrás para la variable *t*, hacia adelante para la variable **x** y tomando como referencia el punto (**x,** $t + \Delta t$) [17]:

$$\begin{aligned}&\frac{f_i(\mathbf{x},t+\Delta t)-f_i(\mathbf{x},t)}{\Delta t}+v_{ix}\frac{f_i(x_x+\Delta x_x,t+\Delta t)-f_i(x_x,t+\Delta t)}{\Delta x_x}\\&+v_{iy}\frac{f_i(x_y+\Delta x_y,t+\Delta t)-f_i(x_y,t+\Delta t)}{\Delta x_y}\\&+v_{iz}\frac{f_i(x_z+\Delta x_z,t+\Delta t)-f_i(x_z,t+\Delta t)}{\Delta x_z}=-\frac{1}{\tau}(f_i(\mathbf{x},t)-f_i^{eq}(\mathbf{x},t))\end{aligned} \qquad (1.20)$$ [10]

Si ahora seleccionamos Δt y $\Delta \mathbf{x}$ de modo que $\Delta \mathbf{v}_i = \Delta \mathbf{x}/\Delta t$ [11], es decir si hacemos que el espaciado de la red de discretización dividida entre el paso de tiempo (Δt) sea igual a la velocidad de las partículas, tenemos:

$$\begin{aligned}&\frac{f_i(\mathbf{x},t+\Delta t)-f_i(\mathbf{x},t)}{\Delta t}+\frac{\Delta x_x}{\Delta t}\frac{f_i(\mathbf{x}+\Delta x_x,t+\Delta t)-f_i(\mathbf{x},t+\Delta t)}{\Delta x_x}\\&+\frac{\Delta x_y}{\Delta t}\frac{f_i(\mathbf{x}+\Delta x_y,t+\Delta t)-f_i(\mathbf{x},t+\Delta t)}{\Delta x_y}\\&+\frac{\Delta x_z}{\Delta t}\frac{f_i(\mathbf{x}+\Delta x_z,t+\Delta t)-f_i(\mathbf{x},t+\Delta t)}{\Delta x_z}=-\frac{1}{\tau}(f_i(\mathbf{x},t)-f_i^{eq}(\mathbf{x},t))\end{aligned} \qquad (1.21)$$

A continuación, sabemos del cálculo que:

$$\begin{aligned}&\Delta f_i(\mathbf{x},t+\Delta t)=\frac{\partial f_i(\mathbf{x},t+\Delta t)}{\partial x_x}dx_x+\frac{\partial f_i(\mathbf{x},t+\Delta t)}{\partial x_y}dx_y+\frac{\partial f_i(\mathbf{x},t+\Delta t)}{\partial x_z}dx_z\\&+\varepsilon\Delta x_x+\varepsilon_1\Delta x_y+\varepsilon_2\Delta x_z\end{aligned}$$

donde ε, ε_1 y ε_2 tienden a cero cuando Δx_x, Δx_y y Δx_z tienden a cero respectivamente, a continuación despreciamos los tres últimos sumandos del lado derecho de la ecuación anterior y discretizando observamos que:

$$\Delta f_i(\mathbf{x},t+\Delta t)=f_i(\mathbf{x}+\Delta x,t+\Delta t)-f_i(\mathbf{x},t+\Delta t)\approx$$

[10] En la ecuación se emplea una notación abreviada i.e. en lugar de escribir $f_i(x_x+\Delta x_x, x_y, x_z, t+\Delta t)$ se escribe $f_i(x_x+\Delta x_x, t+\Delta t)$.

[11] Obsérvese que esto implica que $\Delta v_{ix} = \Delta x_x/\Delta t$, $\Delta v_{iy} = \Delta x_y/\Delta t$, y que $\Delta v_{iz} = \Delta x_z/\Delta t$.

$$\frac{f_i(\mathbf{x}+\Delta x_x, t+\Delta t)-f_i(\mathbf{x}, t+\Delta t)}{\Delta x_x}\Delta x_x+\frac{f_i(\mathbf{x}+\Delta x_y, t+\Delta t)-f_i(\mathbf{x}, t+\Delta t)}{\Delta x_y}\Delta x_y$$
$$+\frac{f_i(\mathbf{x}+\Delta x_z, t+\Delta t)-f_i(\mathbf{x}, t+\Delta t)}{\Delta x_z}\Delta x_z \qquad (1.22)$$

relacionando lo anterior con los sumandos respectivos de la ecuación (1.21) y substituyendo en dicha ecuación queda:

$$\frac{f_i(\mathbf{x}, t+\Delta t)-f_i(\mathbf{x}, t)+f_i(\mathbf{x}+\Delta\mathbf{x}, t+\Delta t)-f_i(\mathbf{x}, t+\Delta t)}{\Delta t}=-\frac{1}{\tau}(f_i(\mathbf{x}, t)-f_i^{eq}(\mathbf{x}, t))$$

simplificando y reordenando términos queda:

$$f_i(\mathbf{x}+\Delta\mathbf{x}, t+\Delta t)=f_i(\mathbf{x}, t)+\frac{\Delta t}{\tau}(f_i^{eq}(\mathbf{x}, t)-f_i(\mathbf{x}, t)) \qquad (1.23)$$

con:

$$m\sum_i f_i(\mathbf{x}, t)=\rho(\mathbf{x}, t) \qquad (1.23a)$$

$$m\sum_i \mathbf{v}_i f_i(\mathbf{x}, t)=\rho(\mathbf{x}, t)\mathbf{u}(\mathbf{x}, t)=\mathbf{j}(\mathbf{x}, t) \qquad (1.23b)$$

donde **u**(**x**, *t*) es la velocidad macroscópica del fluido y **j**(**x**, *t*) es la densidad macroscópica de momento (momento por unidad de volumen) en el punto (**x**, *t*) (nótese que f_i, ρ, **u** y **j** dependen de **x** y *t,* a diferencia de las $\mathbf{v}_i$ que son invariantes en el espacio y el tiempo). La ecuación (1.23) es la llamada *ecuación de redes de Boltzmann*.

Para finalizar esta sección obsérvese cómo la ecuación (1.23) indica como se lleva a cabo la evolución temporal de nuestro sistema, en dicha ecuación se identifican dos sucesos que ocurren en dos tiempos diferentes: en el tiempo *t*, en el nodo **x**, ocurre el así llamado *paso de colisión*, en cual se suma $f_i(\mathbf{x}, t)$ con el término de colisión $\frac{\Delta t}{\tau}(f_i^{eq}(\mathbf{x}, t)-f_i(\mathbf{x}, t))$ (nótese como se evalúa lo anterior sólo de manera local) y posteriormente, en el tiempo $t + \Delta t$, ocurre el *paso de propagación*, en el cuál se propaga o transfiere el resultado del paso de colisión al nodo adyacente $\mathbf{x} + \Delta\mathbf{x}$, en el sentido de la velocidad $\mathbf{v}_i$., estos pasos se repiten consecutivamente para todas las direcciones *i* de todos los nodos de nuestra red en cada paso de tiempo. Si nuestro sistema está definido dentro de ciertos límites espaciales, entonces será necesario aplicar condiciones de frontera adecuados para cada límite, existen varias condiciones de frontera, con distintos grados de exactitud, y apropiadas a determinados tipos de aplicaciones, por ejemplo bounce-back en el enlace, bounce-back de medio enlace, libre de deslizamiento, deslizamiento friccional, inamuro, etc. [4].

1.5 Expansión de Chapman-Enskog.

Es posible asegurar que la ecuación (1.23) puede utilizarse para simular el flujo de un fluido mediante un método que recupera de las ecuaciones de Navier-Stokes analizando el sistema a distintas escalas de tiempo (niveles microscópico y macroscópico) mediante tratamiento perturbativo siendo este procedimiento conocido por los nombres de sus autores: *análisis multiescala* de Chapman-Enskog[12], quienes lo desarrollaron entre 1910 y 1920. Como una rápida aproximación al método (el cual será desarrollado para números de Mach pequeños, por lo que se puede considerar que el sistema estará cerca del equilibrio, y para modelos isotérmicos, ambas consideraciones asumidas para nuestro modelo), para dos dimensiones solamente. Consideremos primero que el fenómeno de flujo de un fluido ocurre a varias escalas de tiempo (lo restante expuesto en la sección esta basado principalmente en [4] y [5], enfatizando las frases exactas donde es necesario):

- En primer lugar está el tiempo en que ocurre una colisión, en el cual están involucrados el diámetro molecular efectivo y la velocidad típica de la partícula[13]. Este tiempo (respecto a los demás) ocurre bastante rápido[14] y que para el caso de la teoría de Boltzmann se considera nulo, es decir se supone que las colisiones ocurren instantáneamente [4].

- Enseguida esta el tiempo que ocurre entre colisiones donde esta involucrada la trayectoria libre media de las moléculas y la velocidad típica de las mismas, en esta escala de tiempo ocurre la relajación hacia el equilibrio local ya que pocas colisiones son necesarias para alcanzar dicho equilibrio. Este tiempo, al igual que el anterior, también ocurre rápido pero más lento que el mismo.

- También está el tiempo donde ocurre la advección y la propagación de las ondas sónicas, el cual es rápido pero menor que los anteriores.

- Finalmente está el tiempo donde ocurre la difusión en el fluido, el cual es bastante más lento que los anteriores.

En cuanto a la escala espacial sólo se considerará una escala, aquella en la cual ocurren la advección y la difusión, la cual es similar para ambas.

En base a las escalas anteriores podemos definir unas nuevas variables cuya función es separar o diferenciar las diferentes escalas de tiempo y espacio en las cuales vamos a definir ahora la ecuación (1.23), y desarrollar esta ecuación en cada escala mediante expansiones alrededor de un número pequeño característico que

[12] [18] Indica que la historia comienza desde David Hilbert, el cual fue el primero en probar la existencia de una clase de soluciones a la ecuación de transporte de Boltzmann.

[13] Depende desde luego del tipo de fluido de que estemos hablando, (por ejemplo gas o líquido) y bajo que condiciones físicas se encuentra (presión, temperatura).

[14] [4] También menciona que en este tiempo ocurre una rápida relajación de la función de distribución de muchas partículas a la función de distribución de una sola partícula $f_{123..N} \rightarrow f_1$.

relaciona la escala micro y macroscópica (que muy adecuadamente puede ser el número de Knudsen), así tenemos:

$t_1 = \varepsilon t$ (advección y ondas sónicas), (1.24a)

$t_2 = \varepsilon^2 t$ (difusión), (1.24b)

$\mathbf{x}_1 = \varepsilon \mathbf{x}$ (advección y difusión) (1.24c)

La función de distribución estará ahora en función de las nuevas variables definidas. La relajación hacia el equilibrio local se considera en la escala ε^0, la cual se realizará en muy pocos pasos de tiempo [5]. El parámetro de expansión o perturbativo servirá también como una etiqueta para distinguir los términos que corren en las distintas escalas de tiempo.

En base a las definiciones anteriores tendríamos las siguientes equivalencias para los operadores diferenciales (obsérvese que dado que las nuevas ecuaciones estarán en función de las nuevas variables definidas y dichas variables están en función de *t*, puede entonces utilizarse la regla de la cadena para la establecer las equivalencias):

$$\begin{aligned} \partial/\partial t &= \varepsilon\, \partial/\partial t_1 + \varepsilon^2\, \partial/\partial t_2 \\ \partial/\partial \mathbf{x} &= \varepsilon\, \partial/\partial \mathbf{x}_1 \end{aligned} \qquad (1.25)$$

También debemos considerar que para poder obtener las ecuaciones de Navier-Stokes a partir de la ecuación (1.23), debemos contar con una adecuada forma discretizada de la distribución de equilibrio, la cual para números de Mach pequeños se obtiene a partir de la expansión en serie de Taylor hasta los términos de segundo orden de la distribución de Maxwell, ecuación (1.11) [18] y [19]:

$$f^{eq} = \frac{\rho}{m(2\pi RT)^{D/2}} e^{-\frac{v^2}{2RT}} \times \left[1 + \frac{\mathbf{v}\cdot\mathbf{u}}{RT} + \frac{(\mathbf{v}\cdot\mathbf{u})^2}{2(RT)^2} - \frac{u^2}{2RT} \right] \qquad (1.26)$$

La distribuciones discretizadas de la ecuación (1.26) toman diferentes formas dependiendo del modelo específico de redes de Boltzmann seleccionado y se obtienen mediante un procedimiento que involucra la utilización de una *cuadratura de Gauss-Hermite* (la cual es una forma de calcular integrales definidas e indefinidas de forma aproximada cuando la función de ponderación es $W(x) = e^{-x^2}$ **[20] y [W5]**), siendo este procedimiento válido para modelos isotérmicos [19]. También es necesario conservar la isotropía, es decir los *tensores de la red* (o *tensor de la lattice*, el cual es igual a $\sum_i v_{ic1} v_{ic2} .. v_{icn}$, donde v_{ic1}, v_{ic2}, … son las componentes cartesianas de la velocidad v_i) de segundo y cuarto rango del modelo seleccionado debe ser isotrópicos (las componentes del tensor deben ser las mismas en cualquier sistema de coordenadas) para que la función de equilibrio sea adecuada para recuperar las ecuaciones de Navier-Stokes [5]. De acuerdo con este mismo autor los

modelos que poseen la isotropía necesaria para recuperar las ecuaciones de Navier-Stokes o bien pueden adquirirla de alguna manera (esto se podría hacer, por ejemplo, dando a cada velocidad v_{ic1}, v_{ic2}, … un determinado peso pi que conduzcan a ser isotrópicos los tensores de la lattice de segundo y cuarto rango) son: D2Q7[15] (red triangular), D2Q9 (red cuadrada incluyendo velocidad cero), D2Q13, D2Q21, D3Q15 (tercera dimensión) y D3Q19, en tanto que otros modelos que no son isotrópicos por ningún medio son por ejemplo el D2Q4 (red cuadrada con cuatro velocidades únicamente).

Por razones que se mencionarán al principio del capítulo 3 el modelo a utilizar en el presente trabajo será el D2Q9. Para este modelo la forma final de la distribución de equilibrio es la siguiente:

$$f_i^{eq}(\mathbf{x},t) = f_i^{eq}(\rho(\mathbf{x},t),\mathbf{u}(\mathbf{x},t)) = \frac{\rho}{m} w_i \times [1 + \frac{3(\mathbf{c}_i \cdot \mathbf{u})}{c^2} + \frac{9(\mathbf{c}_i \cdot \mathbf{u})^2}{2c^4} - \frac{3u^2}{2c^2}] \qquad (1.27)$$

donde w_i = $4/9$ para las partículas en reposo, $1/9$ para i = 1, 2, 3, 4 y $1/36$ para i = 5, 6, 7 y 8, c es un constante igual a $c = \sqrt{3RT} = \sqrt{\frac{3k_B T}{m}}$ y u es la velocidad macroscópica (definida en la ecuación (1.13)).

A la constante c se le llama la *velocidad básica de la lattice* (o red) y en su versión adimensional normalmente se escoge de 1 *ul/pt* (unidad de *lattice* (o red) por paso de tiempo, siendo ambas unidades adimensionales de medida básicas que se utilizan en los modelos basados en las redes de Boltzmann), y se relaciona con las $\mathbf{c}_i$ de la siguiente forma:

$$\mathbf{c}_0 = (0,0), \mathbf{c}_{1,3} = (\pm c, 0), \mathbf{c}_{2,4} = (0, \pm c), \mathbf{c}_{5,6,7,8} = (\pm c, \pm c)$$

para el modelo D2Q9, lo anterior de [21].

Solo para efectos de estar en consonancia con la literatura base en el tema se utilizará en adelante **c**, *c en lugar de* **v**, *v para designar la velocidad de las partículas*.

En la Figura 2 se muestran, para un nodo de la red, los vectores $\mathbf{c}_i$ para el modelo D2Q9.

Ahora bien, continuando con la intensión de recuperar las ecuaciones de Navier-Stokes apartir de la ecuación de redes de Boltzmann, comenzaremos por expandir el término $f_i(\mathbf{x} + \Delta\mathbf{x}, t + \Delta t)$ de la ecuación (1.23) en serie de Taylor

[15] Se utilizará la notación introducida por Quian et al. (1992) $D_{Dim}Q_{Vel}$, donde *Dim* es el número de dimensiones espaciales y *Vel* es el número de velocidades, incluyendo la velocidad cero (partículas en reposo).

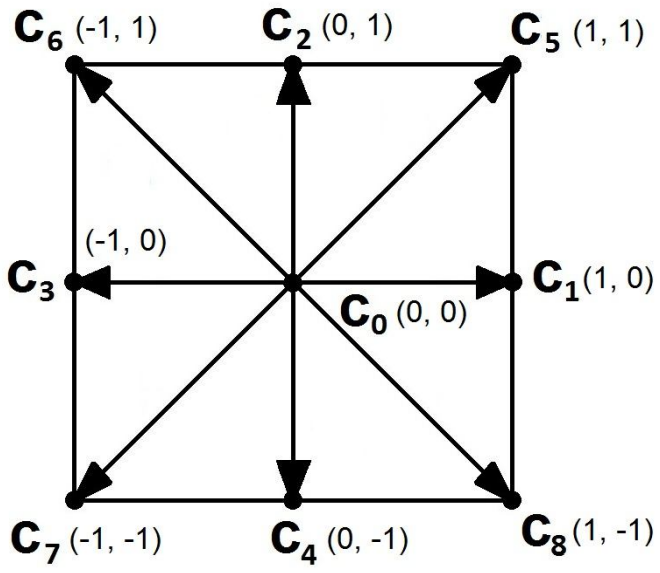


Figura 2. Velocidades $\mathbf{c}_i$ para el modelo D2Q9, fuente: [15].

alrededor de $f_i(\mathbf{x},t)$ hasta los términos de segundo orden, quedando dicho término de la siguiente forma:

$$f_i(\mathbf{x}+\Delta\mathbf{x},t+\Delta t)=f_i(\mathbf{x}+\mathbf{c}_i\Delta t,t+\Delta t)=f_i(\mathbf{x},t)+\Delta t\frac{\partial f_i}{\partial t}+\Delta t c_{i\alpha}\frac{\partial f_i}{\partial x_\alpha}$$
$$+\frac{(\Delta t)^2}{2}\left(\frac{\partial^2 f_i}{\partial t^2}+2c_{i\alpha}\frac{\partial^2 f_i}{\partial t\partial x_\alpha}+c_{i\alpha}c_{i\beta}\frac{\partial^2 f_i}{\partial x_\alpha\partial x_\beta}\right)+O\left(\frac{\partial^3 f_i}{\partial ...}\right) \quad (1.28)$$

Dado que para números de Mach pequeños la distribución resultante esta cercana a la distribución de equilibrio, podemos expandir f_i alrededor de la distribución de equilibrio:

$$f_i=f_i^{eq}+\varepsilon f_i^{(1)}+\varepsilon^2 f_i^{(2)}+O(\varepsilon^3) \quad (1.29)$$

Si multiplicamos por la masa m la ecuación anterior y sumamos para todos los i nos damos cuenta que necesariamente $\sum_i f_i^{(1)}(\mathbf{x},t)=0$, $\sum_i f_i^{(2)}(\mathbf{x},t)=0$, ..., ya que $\sum_i f_i^{eq}(\mathbf{x},t)=\rho$, si ahora multiplicamos la misma ecuación por $m\mathbf{c}_i$ y sumamos para todos los i nos damos cuenta que $\sum_i \mathbf{c}_i f_i^{(1)}(\mathbf{x},t)=0$, ..., debido a que $\sum_i \mathbf{c}_i f_i^{eq}(\mathbf{x},t)=\mathbf{j}$, tenemos entonces:

$$\sum_i f_i^{(1)}(\mathbf{x},t)=0,\quad \sum_i \mathbf{c}_{i\alpha} f_i^{(1)}(\mathbf{x},t)=0,$$
$$\sum_i f_i^{(2)}(\mathbf{x},t)=0,\quad \sum_i \mathbf{c}_{i\alpha} f_i^{(2)}(\mathbf{x},t)=0,$$ [16] (1.30)

... ...

donde $f^{(1)}, f^{(2)},\cdots$ son distribuciones (perturbaciones) que distan de la distribución de equilibrio. Despreciando en (1.28) los términos de tercer orden, en (1.29) los términos $O(\varepsilon^3)$, substituyendo (1.29) en (1.28), y aplicando (1.25) a la ecuación resultante obtenemos en forma compacta:

$$f_i(\mathbf{x}+\Delta\mathbf{x},t+\Delta t)= f_i(\mathbf{x},t)+\Delta t[\varepsilon\frac{\partial}{\partial t_1}+\varepsilon^2\frac{\partial}{\partial t_2}+\varepsilon c_{i\alpha}\frac{\partial}{\partial x_{1\alpha}}+\frac{\Delta t}{2}(\varepsilon^2\frac{\partial^2}{\partial t_1^{\,2}}$$
$$+2\varepsilon^2 c_{i\alpha}\frac{\partial^2}{\partial t_1\partial x_{1\alpha}}+\varepsilon^2 c_{i\alpha}c_{i\beta}\frac{\partial^2}{\partial x_{1\alpha}\partial x_{1\beta}})](f_i^{eq}+\varepsilon f_i^{(1)}+\varepsilon^2 f_i^{(2)}) \quad (1.31)$$

Ahora bien, según [5], la ecuación de redes de Boltzmann que incluye el término de la fuerza externa es:

$$f_i(\mathbf{x}+\Delta\mathbf{x},t+\Delta t)= f_i(\mathbf{x},t)+\frac{\Delta t}{\tau}(f_i^{eq}(\mathbf{x},t)-f_i(\mathbf{x},t))$$
$$+\frac{\Delta t c_{i\alpha}}{12mc^2}\left(K_\alpha(\mathbf{x},t)+K_\alpha(\mathbf{x}+\Delta\mathbf{x},t+\Delta t)\right) \quad (1.32)$$

donde **K** es la *fuerza de masa*[17] que actúa sobre el fluido, desarrollando en serie de Taylor dicho término de fuerza externa obtenemos:

$$\frac{\Delta t c_{i\alpha}}{12mc^2}\left(K_\alpha(\mathbf{x},t)+K_\alpha(\mathbf{x}+\Delta\mathbf{x},t+\Delta t)\right)=\frac{\Delta t c_{i\alpha}}{6mc^2}K_\alpha(\mathbf{x},t)+\frac{(\Delta t)^2 c_{i\alpha}}{12mc^2}(\frac{\partial K_\alpha(\mathbf{x},t)}{\partial t}$$
$$+c_{i\beta}\frac{\partial K_\alpha(\mathbf{x},t)}{\partial x_\beta})+O\left((\Delta t)^3\right)$$

despreciando los términos $O((\Delta t)^3)$ en la ecuación anterior, estableciendo la misma ecuación proporcional a ε, aplicando (1.25), despreciando los términos $O(\varepsilon^3)$, substituyendo en (1.32) e igualando con la ecuación (1.31) queda:

$$f_i(\mathbf{x}+\Delta\mathbf{x},t+\Delta t)-f_i(\mathbf{x},t)=\Delta t[\varepsilon\frac{\partial}{\partial t_1}+\varepsilon^2\frac{\partial}{\partial t_2}+\varepsilon c_{i\alpha}\frac{\partial}{\partial x_{1\alpha}}+\frac{\Delta t}{2}(\varepsilon^2\frac{\partial^2}{\partial t_1^{\,2}}$$
$$+2\varepsilon^2 c_{i\alpha}\frac{\partial^2}{\partial t_1\partial x_{1\alpha}}+\varepsilon^2 c_{i\alpha}c_{i\beta}\frac{\partial^2}{\partial x_{1\alpha}\partial x_{1\beta}})](f_i^{eq}+\varepsilon f_i^{(1)}+\varepsilon^2 f_i^{(2)})$$

[16] Es decir las perturbaciones $f_i^{(1)}$ y $f_i^{(2)}$ no contribuirán a la densidad de masa y momento.
[17] Sus unidades están dadas en *unidades de fuerza* / *unidades de volumen.*

$$= -\frac{\Delta t}{\tau}(\varepsilon f_i^{(1)} + \varepsilon^2 f_i^{(2)}) + \varepsilon \frac{\Delta t c_{i\alpha}}{6mc^2} K_\alpha(\mathbf{x},t) + \varepsilon^2 \frac{(\Delta t)^2 c_{i\alpha}}{12mc^2}(\frac{\partial K_\alpha(\mathbf{x},t)}{\partial t_1} + c_{i\beta}\frac{\partial K_\alpha(\mathbf{x},t)}{\partial x_\beta}) \quad (1.33)$$

De la ecuación anterior observamos que para el primer orden en ε se tiene:

$$\Delta t(\frac{\partial}{\partial t_1} + c_{i\alpha}\frac{\partial}{\partial x_{1\alpha}})(f_i^{(0)}) = -\frac{\Delta t}{\tau} f_i^{(1)} + \frac{\Delta t c_{i\alpha}}{6mc^2} K_\alpha(\mathbf{x},t) \quad (1.34)$$

donde $f^{eq} = f^{(0)}$. Si ahora sumamos para todos los i tenemos:

$$\sum_i (\frac{\partial}{\partial t_1} + c_{i\alpha}\frac{\partial}{\partial x_{1\alpha}}) f_i^{(0)} = \sum_i (-\frac{1}{\tau} f_i^{(1)} + \frac{c_{i\alpha}}{6mc^2} K_\alpha(\mathbf{x},t)) \quad (1.35)$$

Sabiendo que $\sum_i c_{i\alpha} = 0$ para el modelo D2Q9 (observar Figura 2) y aplicando (1.23a), (1.23b) y (1.30) a la ecuación anterior obtenemos:

$$\sum_i \frac{\partial f_i^{(0)}}{\partial t_1} = \frac{\partial \sum_i f_i^{(0)}}{\partial t_1} = \frac{1}{m}\frac{\partial \rho}{\partial t_1}$$

$$\sum_i \frac{\partial c_{i\alpha} f_i^{(0)}}{\partial x_{1\alpha}} = \frac{1}{m}\frac{\partial j_\alpha}{\partial x_{1\alpha}}$$

$$-\frac{1}{\tau}\sum_i f_i^{(1)} = 0$$

$$\frac{1}{6mc^2} K_\alpha(\mathbf{x},t)\sum_i c_{i\alpha} = 0$$

substituyendo lo anterior en (1.35) queda:

$$\frac{\partial \rho}{\partial t_1} + \frac{\partial j_\alpha}{\partial x_{1\alpha}} = 0$$

así para el primer orden en ε se obtiene en notación vectorial:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \mathbf{j} = 0 \quad (1.36)$$

la cual es la ecuación de conservación de la masa o de continuidad.

Si multiplicamos ahora la ecuación (1.34) por $c_{i\beta}$ y sumando para todos los *i*:

$$\sum_i c_{i\beta}\left(\frac{\partial}{\partial t_1}+c_{i\alpha}\frac{\partial}{\partial x_{1\alpha}}\right)f_i^{(0)}=\sum_i c_{i\beta}\left(-\frac{1}{\tau}f_i^{(1)}+\frac{c_{i\alpha}}{6mc^2}K_\alpha(\mathbf{x},t)\right) \qquad (1.37)$$

Aplicando nuevamente (1.30) y (1.23b) a la ecuación anterior se tiene:

$$\sum_i \frac{\partial c_{i\beta}f_i^{(0)}}{\partial t_1}=\frac{1}{m}\frac{\partial j_\beta}{\partial t_1}$$

$$-\frac{1}{\tau}\sum_i c_{i\beta}f_i^{(1)}=0$$

$$\frac{1}{6mc^2}\sum_i c_{i\alpha}c_{i\beta}K_\alpha(\mathbf{x},t)=\frac{K_\alpha(\mathbf{x},t)}{6mc^2}\sum_i c_{i\alpha}c_{i\beta}=\frac{K_\alpha(\mathbf{x},t)}{6mc^2}6c^2\delta_{\beta\alpha}=\frac{K_\alpha(\mathbf{x},t)\delta_{\beta\alpha}}{m}=\frac{K_\beta(\mathbf{x},t)}{m}$$

substituyendo lo anterior en (1.37) se tiene:

$$\frac{\partial j_\beta}{\partial t_1}+\frac{\partial P_{\beta\alpha}{}^{(0)}}{\partial x_{1\alpha}}=K_\beta(\mathbf{x},t) \qquad (1.38)$$

donde según [5], el tensor de flujo de momento, $P_{\beta\alpha}{}^{(0)}$, es igual a:

$$P_{\beta\alpha}{}^{(0)}=m\sum_i c_{i\beta}c_{i\alpha}f_i^{(0)}=\frac{1}{\rho}\begin{pmatrix} j_1{}^2 & j_1j_2 \\ j_1j_2 & j_2{}^2\end{pmatrix}+\frac{c^2\rho}{3}\delta_{\beta\alpha}=\rho u_\alpha u_\beta+p\delta_{\beta\alpha} \qquad (1.39)$$

ya que $\begin{pmatrix} j_1{}^2 & j_1j_2 \\ j_1j_2 & j_2{}^2\end{pmatrix}=\rho^2u_\alpha u_\beta$, siendo j_1 y j_2 las componentes cartesianas de la densidad de momento, $p=RT\rho=\frac{k_BT\rho}{m}=\frac{c^2\rho}{3}$ es la presión y $\rho(\mathbf{x},t)=\rho$ se considerará constante (lo que quiere decir que la ecuación a obtener será para fluidos incompresibles).

Substituyendo (1.39) en (1.38), utilizando de nuevo (1.23b) en el primer sumando del lado izquierdo de la misma ecuación (1.38) obtenemos:

$$\rho\frac{\partial u_\beta}{\partial t_1}+\rho\frac{\partial u_\alpha u_\beta}{\partial x_{1\alpha}}+\frac{\partial p}{\partial x_{1\beta}}=K_\beta(\mathbf{x},t)$$

$$\frac{\partial u_\beta}{\partial t_1}+\frac{\partial u_\alpha u_\beta}{\partial x_{1\alpha}}=-\frac{1}{\rho}\frac{\partial p}{\partial x_{1\beta}}+\frac{K_\beta(\mathbf{x},t)}{\rho}$$

desarrollando el término $\frac{\partial u_\alpha u_\beta}{\partial x_{1\alpha}}$ y aplicando la ecuación de continuidad (1.36) se observa que el mismo se reduce a $u_\alpha \frac{\partial u_\beta}{\partial x_{1\alpha}}$, además $\frac{K_\beta(\mathbf{x},t)}{\rho} = a_\beta$, donde a_β es la aceleración del fluido debida a la fuerza de masa, por lo que se obtiene en notación vectorial:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} = -\frac{1}{\rho}\nabla p + \mathbf{a} \qquad (1.40)$$

Nótese que la ecuación anterior es exactamente la misma que la ecuación (1.18) excepto por el término debido a las fuerzas viscosas $\nu\nabla^2\mathbf{V}$, a la cual se le conoce como *Ecuación de Euler* [16].

Si de manera similar se procede con los términos de segundo orden en ε, se obtienen finalmente según [4] y [5], las ecuaciones de Navier-Stokes en su forma *incompresible* (ecuación (1.18)):

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} = -\frac{1}{\rho}\nabla p + \mathbf{a} + \nu\nabla^2\mathbf{u} \qquad (1.41)$$

junto con la ecuación:

$$\nabla\cdot\mathbf{u} = 0 \qquad (1.42)$$

y en donde la viscosidad cinemática es:

$$\nu = \frac{c^2}{3}(\tau - \frac{\Delta t}{2}) \qquad (1.43)$$

Si el lector quiere profundizar en este tema puede consultar [4], [5] y [22], para expansión de Chapman-Enskog, y [W6], [23], [24], [25] y [26] para tratamiento perturbativo y análisis multiescala.

Capítulo 2

Programación Paralelo.

2.1 La Tecnología CUDA™.

Aunque el método de las redes de Boltzmann se ha mostrado como un método que permite crear algoritmos robustos y confiables para resolver aplicaciones de distintos tipos (véase por ejemplo [27] y [28]), el advenimiento de tecnologías para la computación paralela y a bajo costo así como la factibilidad de paralelización eficiente que muestra un algoritmo basado en el método de la redes de Boltzmann **[29]** hace que dicho algoritmo sea particularmente atractivo para paralelizarse utilizando alguna tecnología disponible actualmente.

Desde hace algunos años se ha venido produciendo una verdadera *revolución de concurrencia* (Sutter, 2005) lo que ha propiciado un mayor interés en la programación en paralelo. Es de notar que la continua innovación que caracterizaba a la industria de los microprocesadores y que permitía que los usuarios gozaran de varias nuevas características y más velocidad con cada nueva generación de microprocesadores ha venido en cierto declive debido a limitaciones físicas en la construcción de dichos circuitos integrados, por ejemplo la cantidad de calor que dichos circuitos pueden disipar limita el aumento de velocidad a la que pueden trabajar los microprocesadores, lo cual se ha venido observando desde 2003 [30]. Inclusive, en teoría, existe la limitación infranqueable de la velocidad a la cual puede viajar una señal eléctrica en un circuito, la cual no puede ser mayor a la de la luz lo que a su vez limitaría el número operaciones de punto flotante por segundo [31]

En este contexto de cambio, las Unidades Gráficas de Proceso (GPU) se han mostrado como atractivas alternativas para procesar las demandantes cantidades de procesamiento numérico de muchas aplicaciones en la actualidad tales como el procesamiento de señales, simulación de fenómenos físicos, finanzas computacionales, biología computacional, renderización en tercera dimensión, etc., debido a la gran cantidad de operaciones de punto flotante que pueden realizar respecto a un CPU multinúcleo[18], así como también al ancho de banda al que se

[18] Unidad Central de Proceso con dos o más Subunidades Centrales de Proceso independientes diseñado con la finalidad de optimizar la ejecución de un programa secuencial haciendo uso del paralelismo proporcionado por dichos núcleos y de varias otras bondades tecnológicas que proporciona cada uno de ellos (mencionadas en el siguiente párrafo en el texto).

puede comunicar dicho GPU con su memoria principal respecto al ancho de banda de la comunicación similar del CPU. En la Figura 3 se muestra un esquema del flujo del procesamiento en un GPU típico y en la Figura 4 se pueden apreciar las comparaciones mencionadas anteriormente entre el GPU de nVIDIA® y el CPU multinúcleo de Intel® para los años 2003 a 2012 observándose que la tendencia es que siga siendo cada vez más grande la relación de cantidades entre ambas tecnologías.

La razón de la notable diferencia entre ambas tecnologías radica en que los CPUs son diseñados para ejecutar código secuencial altamente complejo en la forma más eficiente posible, lo que hace que se incorporen al chip más circuitería de lógica de control que logren por ejemplo predicción de ramificaciones, asimismo se incluyen también grandes memorias de caché de instrucciones para reducir la latencia de acceso a instrucciones y datos, en cambio los GPUs motivados por el creciente mercado de videojuegos se diseñan pensando en resolver paralelamente miles de sencillos cómputos aritméticos [30].

Gracias a su costo relativamente bajo (pueden ser adquiridos por un usuario doméstico, por ejemplo), ahora es posible tener todo el poder de la programación en paralelo en una laptop.

La Arquitectura de Dispositivos de Cómputo Unificado (CUDA por sus siglas en inglés) de nVIDIA® es una reciente tecnología para cómputo que permite acceder a la potencia computacional de los GPUs de la misma compañía mediante una interfaz de programación que es simplemente una extensión del lenguaje C++ y que elimina la necesidad de utilizar Interfaces de Programación de Aplicaciones (APIs por sus siglas en inglés) Gráficas para la construcción de las aplicaciones [30].

La arquitectura de un GPU CUDA consta de un arreglo de multiprocesadores multihilos, donde a su vez cada multiprocesador consta de un conjunto de procesadores que comparten lógica de control y caché de instrucciones. El multiprocesador administra los hilos de un bloque (consultar la versión más actualizada de [32] para una introducción al modelo de programación de CUDA) en grupos de hilos llamados *warps*[19] ejecutándose concurrentemente, todos los hilos de warp ejecutan una y la misma instrucción a la vez, y aunque cada hilo del warp inicia su ejecución en la misma dirección del programa, lo hacen bajo su mismo contador de programa y estado de registros , por lo que si hay una bifurcación, ésta se ejecutará en forma serial ejecutándose primero los hilos de una opción de la bifurcación, deshabilitándose los demás hilos, continuando con los hilos de la siguiente opción de la bifurcación y deshabilitando los restantes, y continuando así consecutivamente con los demás hilos hasta que todos los hilos del warp vuelven a coincidir al final de la bifurcación[20] [32], de aquí se ve claro que

[19] El número de hilos por warp varía con la implementación del GPU.

[20] A esta forma que tiene el multiprocesador de agrupar los hilos por medio de warps y ejecutar el código instrucción por instrucción por medio de dichos warps se le conoce como arquitectura de Instrucción Individual Múltiples Hilos o SIMT por sus siglas en inglés [32].

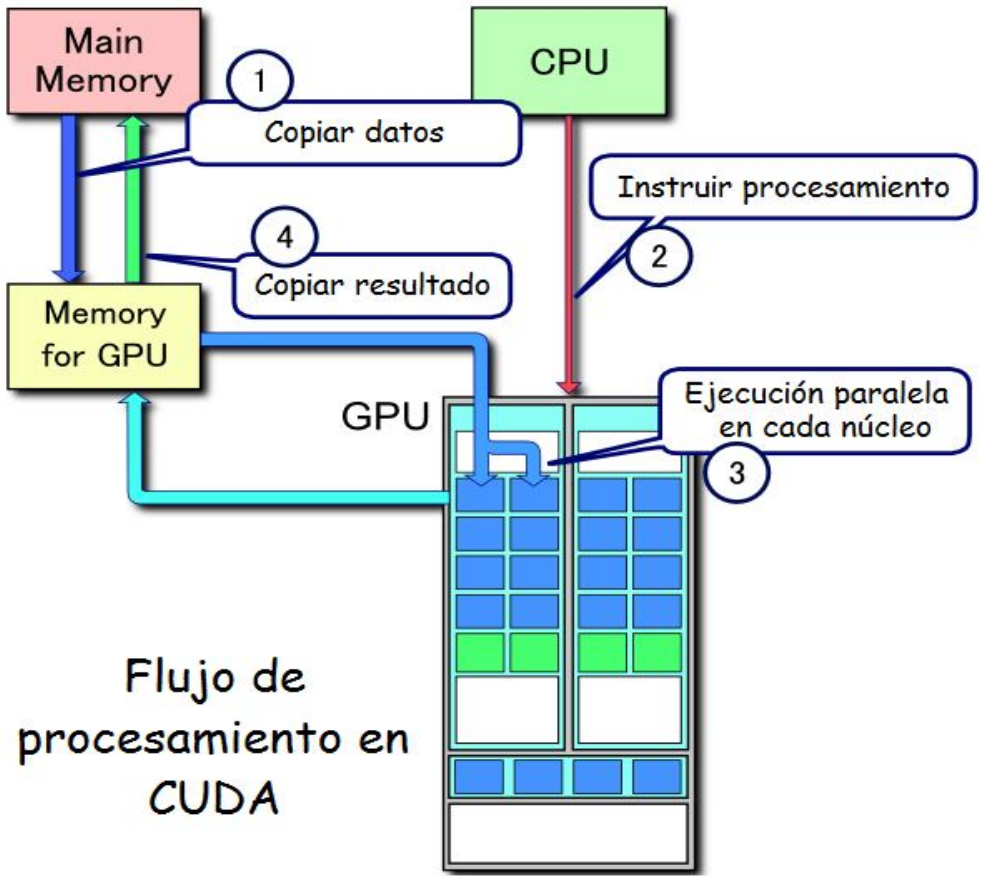
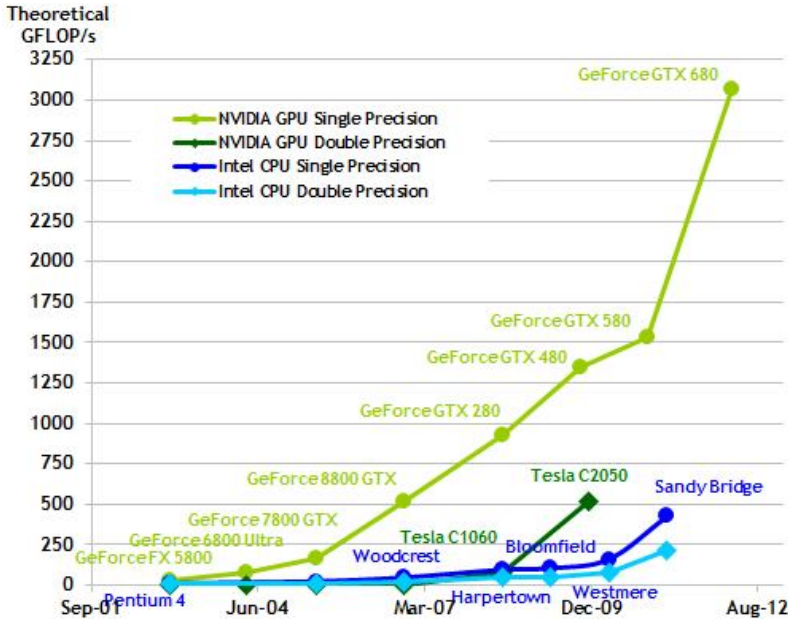


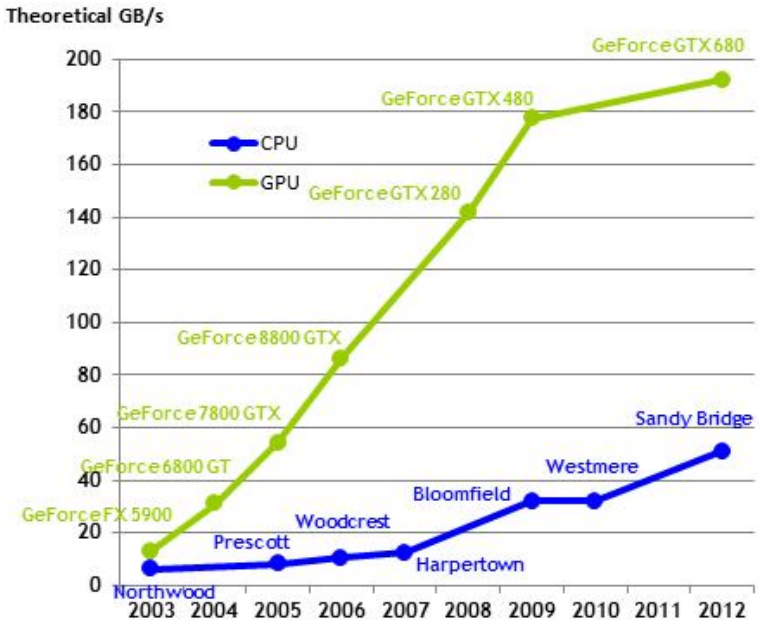
Figura 3. Representación esquemática del flujo de procesamiento para el GeForce® 8800. Permiso de publicación bajo los términos especificados en http://creativecommons.org/licenses/by/3.0/deed.en.

se podrá tener una mejor eficiencia si se cuida que no tenga demasiadas bifurcaciones el código que va a ejecutar el GPU.

Los *kernels* de CUDA, es decir las extensiones al lenguaje C donde se define el código a ejecutar en el GPU pueden accesar a varios tipos de memoria, entre las que se incluyen: registros locales a cada hilo, memoria compartida por todo el bloque de hilos, memoria global de tipo DRAM, memoria constante, memoria de textura, etc., las dos primeras tienen muy baja latencia respecto a la tercera, es decir son muy rápidas de accesar por los núcleos de CUDA dado que se encuentran en el mismo chip de los multiprocesadores, siendo la primera donde se almacenan las variables locales a cada hilo definidas en el kernel, la segunda comúnmente es utilizada para que los hilos puedan trabajar de manera cooperativa, es de acceso muy rápido y en forma paralela, la tercera es donde normalmente se almacenan los datos con los cuales va a trabajar el conjunto de hilos siendo esta común a todos, la memoria constante es parte de la memoria global pero los datos se almacenan en un cache para rápido acceso, es de solo



(a)



(b)

Figura 4. Evolución de la cantidad de operaciones de punto flotante por segundo **(a)** y el ancho de banda de comunicación su memoria principal **(b)** de el GPU de nVIDIA® y el CPU multinúcleo de Intel®. Imagen cortesía de nVIDIA®.

lectura desde los hilos y se puede utilizar para almacenar parámetros de entrada al kernel, finalmente la memoria de textura es utilizada para almacenar los datos necesarios para añadir los detalles gráficos a imágenes en tercera dimensión [30].

La arquitectura CUDA permite también que las aplicaciones sean escalables, es decir basta una simple modificación a los parámetros de definición de hilos por bloque y bloques en el grid para que automáticamente la aplicación se ejecute más rápido con una tarjeta con más núcleos [32].

Uno de los modelos de tarjeta GPU utilizados en este trabajo, la GeForce® GTX480, cuenta con una capacidad de cálculo de poco menos de 1350 Gigaflops (flops: operaciones de punto flotante por segundo) [W7], puede manejar un máximo de 1024 hilos por bloque, permite un máximo de 8 bloques residentes por multiprocesador, un máximo de 48 warps residentes en el mismo, y máximo de 1536 hilos residentes en dicho multiprocesador, el otro modelo de tarjeta que se utilizó fue la GTX660 ti, que con una capacidad de cómputo[21] de 3.0 tiene varias de las últimas innovaciones tecnológicas que junto con las correspondientes de la primera tarjeta (capacidad de cómputo 2.0) se pueden consultar en [32].

Para cerrar los comentarios acerca de la tecnología CUDA cabe mencionar que aunque cada hilo es capaz de ejecutarse independientemente de los otros hilos y seguir su propia ruta de bifurcación debido a que tiene su propio contador del programa y registro de estados, solo hay paralelismo en aquellos hilos que siguen la misma ruta de bifurcación, en esto la arquitectura SIMT se parece a la arquitectura SIMD, en la cual una instrucción es ejecutada en paralelo en múltiples unidades de procesamiento con sus propios datos que toma de un vector de datos inicializado previamente, pero en la que, a diferencia de la arquitectura SIMT, las unidades de procesamiento no son independientes de ejecutarse siguiendo su propia ruta de bifurcación si es que la hay, por lo que el programador debe manejar estas situaciones de forma manual. Asimismo el modelo de programación relacionado con la arquitectura SIMT tiene similitud con el modelo SPMD[22] en que cada hilo en un warp es capaz de ejecutarse independientemente de los otros hilos dentro del mismo warp, sin embargo difiere en que es el planificador del warp el que administra la ejecución o no ejecución de los hilos para diferentes rutas de bifurcación, en tanto en el modelo SPMD cada unidad de procesamiento es en si una máquina de Von Newman[23] ejecutando diferentes secciones del total de datos a procesar (todas las máquinas ejecutando el mismo programa) [32] y [33].

2.2 El Estándar MPI.

[21] Clasificación dada por nVIDIA® en base al tipo de arquitectura de los núcleos y mejoras tecnológicas.

[22] Single program, multiple data – un programa, multiples datos, considerada como una extensión a la taxonomía de Flynn, consultar F. Darema, *SPMD model: past, present and future*, Recent Advances in Parallel Virtual Machine and Message Passing Interface: 8th European PVM/MPI Users' Group Meeting, Santorini/Thera, Greece, September 23-26, 2001. Lecture Notes in Computer Science 2131, p. 1, 2001.

[23] Una máquina que realiza el ciclo de von Newman, es decir extraer una instrucción de la memoria, decodificarla, incrementar el contador del programa, ejecutar la siguiente instrucción y así consecutivamente, solo cuenta con un flujo de datos y un flujo de instrucciones [33].

Basado en el modelo de programación paralela de paso de mensajes, en el cual los procesos a los cuales están asignadas las distintas tareas en que se divide la aplicación trabajan como entidades independientes (con su propio espacio de direcciones, variables privadas y pila) unos de otros colaborando y compartiendo información entre si mediante mensajes[24] que pueden llevar datos o ser utilizados para sincronización, el estándar de programación MPI (Message Passing Interface) surge después de la colaboración de investigadores, afiliados industriales, desarrolladores, etc., que en conjunto se le conoce como Forum y que desean agrupar en una especificación el conjunto de librerías y programas dispersos que trataban de dar soporte al campo industrial de las arquitecturas distribuidas y cuyo fruto fueron primero el estándar MPI-1 liberado en 1994 y posteriormente el MPI-2 terminado en 1996[25]. Actualmente las implementaciones de MPI son una combinación de MPI-1 y MPI-2 [34] y [W8].

Aunque el modelo asociado al interfaz es MIMD (*Multiple Instruction streams, Multiple Data streams*, donde en esta arquitectura existe mas de un procesador independiente que ejecuta su propio programa con sus propios datos) existen extensiones para ejecutar las aplicaciones como un SPMD (Single Program, Multiple Data, donde todos los procesos ejecutan el mismo programa pero bajo su propia planificación de ejecución). Hay que hacer notar que el programador es responsable de establecer el comportamiento paralelo de su programa mediante las llamadas MPI correspondientes, así como de iniciar y terminar correctamente el ambiente MPI, manejar adecuadamente los códigos de error, etc., es decir el paralelismo es explícito.

Para nuestro caso lo que básicamente el sistema operativo realiza, bajo el estándar MPI, es ejecutar el código de la aplicación N veces mediante N procesos ordenados, cuyo entero asociado, llamado rango, se puede consultar mediante las llamadas MPI correspondientes.

Asimismo se crea un comunicador por default, el cual es un objeto que sirve para agrupar los procesos de forma que el envío/recepción de los mensajes entre los mismos estén plenamente identificados. Es posible definir más comunicadores incluyendo en él los procesos (o nodos) que se deseen [W8].

Es posible la comunicación entre dos procesos (o comunicación punto a punto) del mismo o diferente comunicador en forma síncrona o asíncrona utilizando las llamadas de envío y recepción correspondientes. Las llamadas correspondientes para la comunicación síncrona son las siguientes:

Envío:

MPI_SSEND (buf, count, datatype, dest, tag, comm)

[24] Los cuales no están limitados solamente a comunicaciones intermemoria sino que utilizan también el sistema de entrada /salida [34] (por lo que pueden ser distribuidos).

[25] Acualmente un grupo de universidades, centros de investigación, etc., trabajan en el desarrollo del MPI-3, puede consultarse la página http://meetings.mpi-forum.org/MPI_3.0_main_page.php.

donde:

buf	Es la dirección inicial de entrada del buffer a enviar.
count	Número de elementos que tiene dicho buffer a enviar debe ser un entero no negativo).
datatype	Tipo de datos del buffer a enviar.
dest	Rango de destino (debe ser un entero).
tag	Etiqueta que permite distinguir entre varios mensajes distintos que tienen el mismo destino (debe ser un entero positivo dentro del rango permitido).
comm	Comunicador al cual se va a enviar el buffer.

Recepción:

MPI_RECV (buf, count, datatype, source, tag, comm, status)

donde:

buf	Dirección inicial del buffer de recepción.
count	Número de elementos del buffer de recepción (debe ser un entero no negativo).
datatype	Tipo de datos del buffer de recepción.
source	Rango del nodo origen o la constante MPI_ANY_SOURCE (debe ser un entero).
tag	Etiqueta del mensaje o la constante MPI_ANY_TAG (debe ser un entero).
comm	Comunicador del nodo origen.
status	Estructura que contiene datos acerca del nodo origen del mensaje y que pueden variar según la plataforma de programación en que se utilice, por ejemplo el número de nodo, la etiqueta (datos que pueden ser útiles cuando se utilizaron las constantes comodines), códigos de error para situaciones específicas, etc.

La llamada para la *comunicación colectiva* es:

MPI_BCAST (buffer, count, datatype, root, comm)

donde:

buffer	Dirección inicial del buffer de envío/recepción.
count	Número de elementos en el buffer (debe ser un número entero no negativo).
datatype	Tipo de datos del buffer.
root	Rango del nodo raíz del *broadcast* (debe ser un entero).
comm	Comunicador del contexto del *broadcast*.

Fuente de lo anterior: [35].

Si el lector esta interesado en una introducción o profundizar en el entorno de programación MPI puede consultar [36] y [37], la guía del usuario, guía de instalación, y guía del desarrollador de Windows para la implementación MPICH2 se encuentran en [W9], otras ligas de interés son: [W10], [W11] y [W12].

2.3 El Modelo de Programación OpenMP®.

OpenMP (Open Multiprocessing) es un modelo de programación paralela multihilo que se basa en el modelo de programación paralela de memoria compartida y que toma de éste paradigma algunas de sus abstracciones, entre ellas: se comparten entre cada hilo trabajador cero o más variables para cooperación y comunicación entre los mismos hilos, y su esencia: que el modelo de programación se puede ver como un CPU compartiendo tiempo de procesamiento entre varios hilos trabajadores, solo reemplazando las porciones de tiempo por procesadores **[34]** (hay que tomar en cuenta que el modelo de programación de memoria compartida no es una simple multiprogramación de los hilos sino auténtico paralelismo donde, como ya se mencionó hay cooperación y comunicación entre los mismos).

La forma de funcionamiento de un programa en C/C++ o Fortran es la siguiente: se inicia la ejecución en forma secuencial mediante un hilo inicial llamado el hilo maestro que seguirá ejecutando el programa en forma secuencial hasta que encuentre una directiva de compilador que abarque una cierta región paralela dentro del código en donde el hilo maestro genera cero o más hilos con su propio contexto de ejecución[26] (equipo de hilos trabajadores, que incluye al hilo maestro siendo éste a su vez hilo de control del equipo), los cuales ejecutarán en forma paralela el código que enmarque dicha región paralela, al final de la cual existe un barrier o punto de sincronización implícito donde se desaparece todo el equipo de hilos generados, continuando el hilo maestro su ejecución hasta que encuentre otra directiva que abarque otra región paralela y continúe así consecutivamente hasta finalizar el programa. A esta forma de funcionamiento de OpenMP se conoce como dividir-unir (fork-join) [38].

Para una introducción general y/o profundización al modelo de programación de OpenMP, uso de la diferentes directivas, etc. se puede consultar [38], asimismo algunos tutoriales, especificaciones técnicas de la API de OpenMP, compiladores, etc., pueden hallarse en la página www.openmp.org.

Desde luego existen otras alternativas para crear aplicaciones paralelas como por ejemplo PVM, en el cual se permite que varias computadoras (Unix o Windows)

[26] Espacio de direcciones que incluye sus variables privadas y su propia pila que puede ser utilizada para llamadas a subrutinas.

conectadas mediante una red trabajen como una sola computadora **[W13]**, Fortran 90, las bibliotecas de programación de algún computador paralelo existente, etc., en el presente trabajo solo se mencionan dos de las más conocidas, las cuales tienen la ventaja de que se pueden ejecutar en arquitecturas convencionales[27] y que existen implementaciones de libre uso: por ejemplo, para el estándar MPI, el cual fue el utilizado en el presente trabajo, está la implementación MPICH2 del Laboratorio Nacional de Argonne, también está el HPC Server 2008 proporcionada por Microsoft®, en tanto que para OpenMP está el GCC cuyas versiones más recientes soportan la especificación 3.1 de OpenMP, versión de especificación más reciente también, en nuestro caso se utilizó la versión proporcionada por Microsoft® incluida en Visual Studio® 2010.

Cabe mencionar que aunque para el presente trabajo sólo se consideraron los modelos de programación antes mencionados, existen varios otros modelos de programación paralela y sus arquitecturas de hardware asociadas, por ejemplo el modelo de procesamiento paralelo de datos, en el cual se ejecutan instrucciones en paralelo (por un conjunto de unidades funcionales o procesadores que se podrían ver como ALUs muy simples) en cada elemento de un conjunto de datos, es decir haciendo un mapeo de cada instrucción ejecutante (y su procesador asociado correspondiente) a cada elemento de datos por medio de una unidad de control, los s*istemas distribuidos*, en los cuales los recursos físicos y lógicos están dispersos en el espacio[28], pero que el usuario ve como una sola computadora y que es capaz de realizar en forma *concurrente* y cooperativa las tareas asignadas, otras arquitecturas paralelas son arquitectura de flujo de datos, arquitecturas sistólicas, etc. [34] y [39].

2.4 Paralelización del Modelo Propuesto.

La forma en que se paralelizará el modelo propuesto en el presente trabajo será aprovechando las ventajas que por su misma forma ofrece el método de las Redes de Boltzmann, en primer lugar ya se ha mencionado en el capítulo anterior que dicha ecuación (1.23) representa dos sucesos que deben considerarse en el cálculo, el paso de propagación y el paso de colisión, en las primeras versiones del diseño de nuestro algoritmo se integró el paso de propagación y la aplicación de las condiciones de frontera tanto de las interacciones fluido-pared de la vena como las de fluido-pared de la partícula en un solo ciclo anidado, dejando que el cálculo del paso de colisión fuera realizado en ciclos anidados de manera separada, en una depuración posterior el cómputo del paso de propagación quedó sólo en un ciclo anidado, al igual que el cómputo del paso de colisión; esto ayudó a decidir paralelizar dicho paso de colisión, según mostraba de manera natural la forma final en que había quedo finalmente el ciclo principal del programa secuencial y aprovechando la intrínseca localidad del cálculo del mismo, de manera directa en el

[27] En este trabajo se entenderá esto por aquellos ordenadores personales dirigidos al usuario estándar para uso general que incluyen los nuevos procesadores multinúcleo de segunda generación de Intel®.

[28] Donde existen diferentes criterios para establecer que tan dispersos o que tan cooperativos deben ser los sistemas para considerarse distribuidos, véase por ejemplo [39].

GPU, y a que en el caso del paso de propagación y la aplicación de las condiciones de frontera realizarlas mediante un estándar de paralelización que aprovechara las bondades proporcionadas por las tecnologías de optimización de código que ofrece un CPU, debido principalmente a la intrínseca no localidad del cálculo de estos dos últimos.

La intrínseca localidad del cálculo del paso de colisión significa que su paralelización es total, es decir todos los nodos de la red (sus cálculos asociados) que participan en el cómputo global pueden ser enviados directamente para su ejecución al GPU, delegando un hilo de GPU a cada nodo de la red para tal efecto, por lo que, si no se abusa de los recursos limitados que existen por bloque de hilos que proporciona el GPU, se puede aprovechar casi al máximo la potencia de ejecución de dicho GPU.

Se decidió utilizar el estándar de paralelización MPI para el cómputo restante (cuya razón se mencionará más adelante, en el siguiente capítulo), distribuyendo el mismo entre los CPUs disponibles en el hardware utilizado, mediante la implementación para el sistema operativo Windows®, MPICH2, del Laboratorio Nacional de Argonne, que es donde finalmente se corrieron las pruebas finales, pero también verificando su correcto funcionamiento mediante la implementación HPC Server 2008 proporcionada por Microsoft®, comprobando que, aunque no hay diferencia alguna en los resultados obtenidos, se nota una eficiencia (aquí entiéndase velocidad de ejecución) ligeramente menor que para la implementación anterior.

La forma en que se combinaron las tecnologías de paralelización anteriores, y que de hecho es a lo que se denominará aquí como la arquitectura combinada, fue de la siguiente manera, primero que nada se discretiza el sistema (pared de la vena, fluido y partícula) como un conjunto de nodos en una red de Boltzmann que cubre dicho sistema, los cuáles, para nuestro caso y por razones que se darán más adelante, se seleccionaron con un patrón cuadrado y de manera que las funciones de distribución solo pueden “moverse” a lo largo de los enlaces que unen uno con otro todos los nodos de la red de la manera como se mostró en la Figura 2, enseguida se divide dicho sistema como un conjunto de franjas de datos (es decir de los datos asociados a los mismos, como son las funciones de distribución para las distintas velocidades, la densidad y la velocidad macroscópica) de forma rectangular, a lo largo de la vena y de igual tamaño, a excepción quizá, de la franja final situada en uno de los extremos de la dicha vena, de la forma que se muestra en la Figura 5, nótese que cada franja tendrá además dos hileras adicionales de datos, que servirán para la comunicación entre los nodos MPI asociados a cada franja (enseguida se explica que se entiende por esta asociación).

Cada franja de datos se asociará a un nodo de MPI, es decir será procesada por un proceso que corre en núcleo de CPU específico levantado por el sistema operativo bajo la administración de la implementación MPI utilizada.

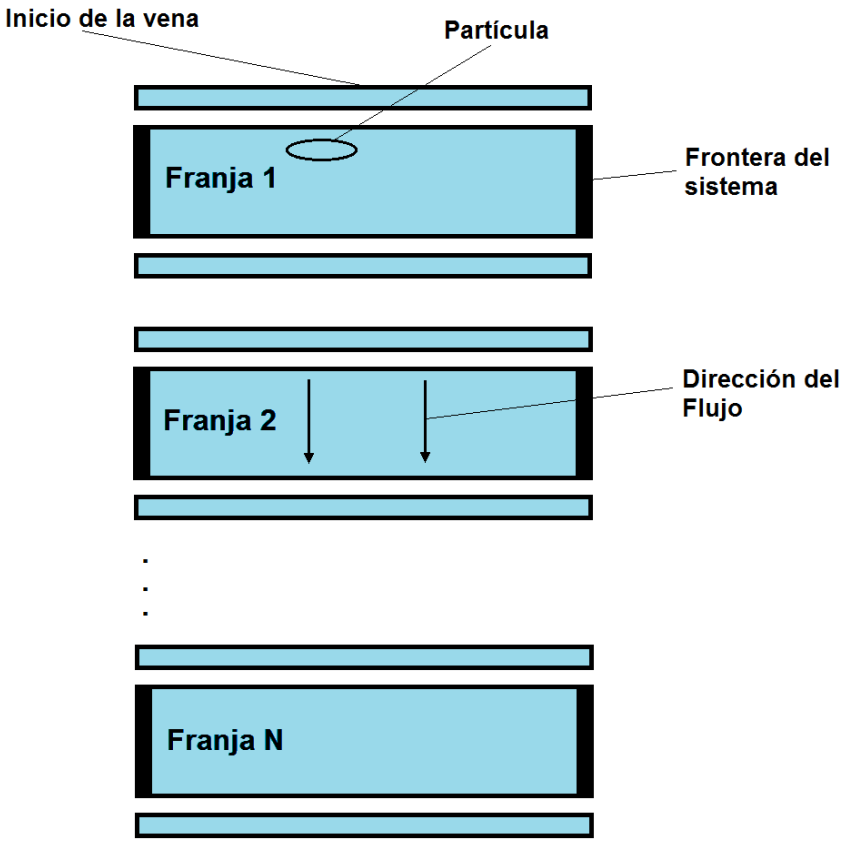


Figura 5. División del sistema vena + partícula + fluido en franjas de datos rectangulares.

Ahora bien, cada CPU computará el paso de propagación y el ciclo anidado donde se aplican las condiciones de frontera bounce back de medio enlace (más adelante en el capítulo 3 se mencionará a que se refiere esto), para las interacciones entre el fluido y la pared de la vena, así como las correspondientes entre el fluido y la pared de la partícula, el cálculo de las componentes cartesianas de la fuerza y la torca ejercida por el fluido sobre la partícula y la lógica de la simulación, que debido a su intrínseca no localidad para el primero y a su complejidad, para los restantes, se adaptan de manera natural para su cómputo mediante un CPU según nuestro esquema de diseño. Para el cálculo del paso de colisión, debido a las razones mencionadas anteriormente, cada nodo de MPI será el encargado de hacer la llamada al kernel del GPU para realizar el cómputo de dicho paso correspondiente a los nodos de la franja asociada.

La propagación de las distribuciones desde los nodos de fluido (cabe señalar que, al igual que [1] en el modelo propuesto se considera que no existe fluido al interior de las partículas, por lo que no habrá propagación desde los nodos correspondientes) de un nodo de MPI a los nodos de fluido de los nodos MPI adyacentes se realizará mediante las hileras adicionales de nodos de cada franja, propagando primero las distribuciones de los nodos frontera de cada franja a dichas hileras y después copiando, mediante las llamadas de comunicación de MPI

anteriormente vistas, estas propagaciones a los nodos correspondientes en los nodos frontera del nodo MPI adyacente hacia donde se esta realizando la propagación.

La aplicación de las diferentes condiciones de frontera, se realiza mediante el arreglo de la matriz de obstáculos, la cual es un arreglo de tipo entero de las mismas dimensiones de la red de Boltzmann del sistema, que sólo almacena en sus elementos un número entero positivo de un dígito para identificar si el nodo de la red asociado a dicho elemento pertenece a un nodo de fluido, de frontera de la pared de la vena o de partícula, este arreglo se replica en cada nodo, asegurándose la coherencia en todos los nodos de MPI ya que los resultados necesarios para calcular la posición de la partícula (que de hecho, si se observa, es lo único que puede actualizar la matriz de obstáculos, ya que la frontera se considera inamovible y son por lo tanto constantes los valores de sus nodos dentro de la matriz de obstáculos, en tanto que si la partícula se mueve esto implica que algunos elementos de la matriz de obstáculos pasarán de ser nodos de fluido a nodos de partícula y viceversa) se comunicarán a todos los nodos asegurándose así la coherencia en todos los datos de la simulación, ya que cada nodo realiza los mismos cálculos correspondientes a la lógica de la simulación como son: actualización de la matriz de obstáculos, cálculo de la posición de cada nodo de la partícula en el sistema, etc., nótese que lo anterior no aplica para el cómputo de las distribuciones pertenecientes a cada franja de datos pues cada núcleo de CPU sólo realiza los cálculos únicamente sobre las distribuciones asociadas a su franja de datos particular, por lo que en el caso, que se explicará con más detalle en el capítulo siguiente, de que se substituyan los nodos de partícula por nodos de fluido, si es que ésta ha cambiado lo suficiente su posición para realizar la substitución, se adecua la lógica de control para que cada CPU realice las actualizaciones necesarias solo en las distribuciones correspondientes dentro de su franja de datos particular.

En el capítulo 3 se darán más detalles acerca de cómo se realizó la paralelización del sistema.

2.5 Métrica de la Paralelización y Performance.

La métrica de la paralelización se realizó comparando el tiempo de ejecución del programa ya paralelizado contra el tiempo de ejecución del programa secuencial base para la primera de las pruebas que se realizaron, se modificó, además, el número de iteraciones de modo que el tiempo de ejecución no sobrepasara un límite razonable y representativo de tiempo, se realizaron 5 mediciones y se obtuvo la media aritmética de las mismas como resultado a comparar, lo mismo se hizo para un programa paralelizado mediante OpenMP, el cuál se utilizó para comparar contra éste el rendimiento obtenido por el programa final.

Para el cómputo realizado mediante el GPU, se calcula que el *performance* en GFLOPs (Giga Operaciones de Punto Flotante por segundo) para el kernel

finalizado, cargando de memoria palabras de tipo float de simple precisión (4 bytes), con un ancho de banda de acceso a memoria global de 177.4 GB/s (Giga bytes por segundo) y realizando 152 Operaciones de Punto Flotante por 18 accesos a memoria global, es de $\left(\frac{177.4}{4}\right)\times\left(\frac{152}{18}\right)=374.51$ GFLOPs, lo cual representa el 27.84 % aproximadamente del performance pico que puede alcanzar el GTX480, el cual fue el GPU utilizado para el cómputo del paso de colisión.

Para paliar lo anterior, se hicieron las siguientes mejoras al kernel de CUDA para mejorar el performance:

- ✓ Se realizan los accesos a la memoria global para que al menos los mismos sean de forma secuencial, esto es, haciendo que todos los hilos de un mismo warp lean localidades contiguas de memoria, los datos a leer o escribir tengan el tipo apropiado (en nuestro caso se escogió float de 4 bytes) y el ancho del bloque de hilos y del arreglo de datos definido en memoria sean múltiplos de dicho warp (32 hilos), de manera que la petición a la memoria global sea realizado por el GPU mediante el mínimo de transacciones necesarias [W14], reduciendo con ello el tráfico de accesos a memoria y por lo tanto maximizando el rendimiento del uso de la misma.

- ✓ Se optimizó la carga de datos a la memoria y el cómputo del kernel de CUDA dividiendo estos en pedazos de cómputo y de carga para ejecutarlos concurrentemente mediante *streams* (conjunto de instrucciones que realizan las acciones de carga y cómputo mencionadas) de CUDA [32], aprovechando que la tarjeta gráfica utilizada proporciona esta funcionalidad. La forma en que realizó la optimización se basó en la presentada en [W15] y en el apéndice A se muestra el código del ciclo principal paralelizado de la arquitectura combinada, en el cual se incluye la optimización mediante streams al principio de dicho ciclo.

- ✓ Uso reducido de registros y de memoria compartida para cargar el número máximo de bloques permitidos por multiprocesador y optimizar con ello el uso de los mismos.

- ✓ Uso de la memoria *no paginable* del programa host [32], con lo cual no es necesario copiar los datos desde o hacia el dispositivo GPU, (de hecho éste tipo de memoria se utilizó también para la optimización mediante streams mencionada anteriormente), entre otras correcciones y mejoras.

En el caso del cómputo realizado mediante los nodos MPI, la medida del performance se evaluará mediante el *radio de comunicación a cómputo* [34], el cual como se explicará más adelante en el siguiente capítulo, es un factor que puede influir en el tiempo de cómputo global, en cuanto a la carga computacional ésta se dividió entre los nodos MPI realizando un balance adicional de la misma en caso de que el número de nodos de la red no sea divisible por los nodos de procesador.

Las especificaciones del servidor así como de las tarjetas gráficas utilizadas para ejecutar todas las pruebas realizadas se muestran en el apéndice B.

Capítulo 3

Modelo Propuesto.

3.1 Preliminares.

Por simplicidad, eficiencia, conveniencia y exactitud computacional [1] y [8] usaremos el modelo de redes de Boltzmann D2Q9 así como el término de colisión según la aproximación BGK para la simulación del plasma sanguíneo, ecuación (1.16), y el término respectivo para la distribución de equilibrio, ecuación (1.27), donde, como ya se mencionó en el capítulo 1, el término de la viscosidad cinemática para el modelo D2Q9 es $\nu = c^2 \frac{(2\tau - \Delta t)}{6}$. Nuestro modelo está fuertemente basado en el propuesto por [1], [40], y algunos antecesores de éste último: [41] y [42], especialmente en la parte de las interacciones hidrodinámicas. El sistema consta de una célula de sangre roja, que al igual que [6][29], será modelada como una partícula rígida con forma de elipse, la cual se encuentra suspendida en plasma sanguíneo, donde este último, según indica [1], puede considerarse como un fluido Newtoniano, y las paredes de una vena que limitan el flujo del fluido a ingresar por uno de los extremos de la vena y salir por el otro (dado que se modelará en dos dimensiones, el sistema se podría visualizar como la sección longitudinal de un segmento de vena con forma recta, con cada extremo cortado perpendicularmente al eje de la misma), además se considerará que existe una fuerza de masa externa actuando sobre el fluido. Esto puede simularse mediante la ecuación (1.23) añadiendo a la misma un término que considere dicha fuerza externa **[7] y [10]**, por simplicidad se usará el término más comúnmente usado para simular flujos de fluidos [8], quedando la ecuación de redes de Boltzmann a utilizar de la siguiente forma:

$$f_i(\mathbf{x} + \mathbf{c}_i \Delta t, t + \Delta t) = f_i(\mathbf{x}, t) - \frac{\Delta t}{\tau}(f_i(\mathbf{x}, t) - f_i^{eq}(\mathbf{x}, t)) + \frac{\Delta t}{6mc^2} c_{i\alpha} K_\alpha(\mathbf{x}, t) \qquad (3.1)$$

para cada enlace i, donde Δt se considerará como un paso de tiempo (*pt*) o una iteración, **K** es la *fuerza de masa* y $c_{i\alpha} K_\alpha(\mathbf{x}, t)$ es la suma de la multiplicaciones de cada componente cartesiano α de $\mathbf{c}_i$ y **K**. Nótese que el término de la fuerza de masa

[29] En su artículo los autores demuestran que la modelación de una célula de sangre roja como una partícula rígida con forma de elipse es adecuada para obtener resultados muy aproximados a los experimentales.

es idéntico al único término de orden Δt en el desarrollo en serie de Taylor que se realizó para el término de fuerza de masa de la ecuación (1.32) en el capítulo 1.

Asimismo cabe mencionar que, aunque la ecuación (3.1) es muy simple y obtenida partiendo de un sistema compuesto por partículas de gas diluido, ha demostrado su efectividad para la simulación del flujo según lo demuestran diversos trabajos previos, por ejemplo (aparte de [1]) [2], [6], [43] y [44].

Ahora bien, en la presente simulación se trabajará con la versión adimensional de la ecuación (3.1), donde las equivalencias entre las cantidades adimensionales (o unidades de la red) y cantidades en unidades físicas que intervienen en el cómputo pueden ser determinadas en unidades MKS utilizando los factores de conversión mostrados en la Tabla 1.

Las unidades básicas adimensionales de masa, longitud, tiempo y volumen en unidades de red o de *lattice* son *um, ul, pt* y *uv,* respectivamente (nótese que 1 *uv* = 1 ul^3).

Para el cálculo de los factores de conversión de la tabla anterior se parte seleccionando $\delta x = {}^{2}\!/_{3}\ \mu m$, que según menciona [1] es adecuado para mantener un costo computacional bajo y un mínimo de resolución que permita eficientar la ejecución del programa computacional, asimismo en base a pruebas previas[30] se escoge δt = 0.25 μs para acelerar el proceso de cálculo, enseguida se calcula τ primero multiplicando la viscosidad cinemática adimensional, $\nu = \frac{(2\tau - 1)}{6}$, por ${}^{(\delta x)^2}\!/_{\delta t}$, y a continuación igualando con la viscosidad cinemática de la sangre que es de 1.2 x 10^{-6} m^2/s, según [6], y finalmente despejando, quedando como resultado:

$$\tau = \frac{1}{2}\left(\frac{6 \times 0.25 \times 10^{-6} s \times 1.2 \times 10^{-6}\ {}^{m^2}\!/_{s}}{({}^{2}\!/_{3} \times 10^{-6} m)^2} + 1 \right) = 2.525$$

con lo que la viscosidad adimensional queda:

$$\nu = \frac{2 \times 2.525 - 1}{6} = 0.675\ {}^{ul^2}\!/_{pt}$$

Realizando ahora el cálculo para δm partimos de multiplicar la densidad adimensional por ${}^{\delta m}\!/_{(\delta x)^3}$, enseguida igualamos con la densidad de la sangre que es de 1 x 10^3 kg/m^3, según [6], y finalmente despejamos, resultando lo

[30] Es decir, en base a resultados obtenidos en dichas pruebas, que aunque no se mencionan en el presente trabajo, sirvieron de base para seleccionar una prueba acorde a las necesidades y los tiempos del mismo.

nombre	símbolo	valor
masa	δm	$2.962963\times10^{-16}\,kg$
tiempo	δt	$0.25\times10^{-6}\,s$
longitud	δx	$2/3\ \mu m$
fuerza	δF	$3.160494\times10^{-9}\,N$
velocidad	δv	$8/3\ m/s$
torca	δT	$2.106996\times10^{-15}\,N\cdot m$

Tabla 1. Factores de conversión utilizados en la simulación.

siguiente:

$$\delta m = \frac{1}{\rho}\left(1\times10^{3}\ kg/m^{3}\right)(\delta x)^{3}$$

seleccionando ρ = 1 um/uv por simplicidad y substituyendo en la ecuación anterior resulta:

$$\delta m = \left(1\times10^{3}\ kg/m^{3}\right)\left(2/3\times10^{-6}\,m\right)^{3} = 2.962963\times10^{-16}\,kg$$

Para el cálculo de δF, δv y δT simplemente se combinan los factores de conversión obtenidos anteriormente, por ejemplo para la velocidad:

$$\delta v = \frac{\delta x}{\delta t} = \frac{2/3\times10^{-6}\,m}{1/4\times10^{-6}\,s} = 8/3\ m/s$$

de manera similar se procede para el cálculo de δF y δT.

Se calculará la masa de la partícula de acuerdo a los valores dados por [6] y considerando que la célula de sangre roja toma, en el fluido, una forma aproximada de un elipsoide oblato según el modelo presentado por [1]:

Densidad del citoplasma = 1098 kg/m^3

Volumen de elipsoide de revolución oblato para semieje mayor = 4.2 μm y semieje menor = 1.2 μm:

$$V = \frac{4}{3}\pi\left((4.2)^{2}\times1.2(\mu m)^{3}\right) = \frac{84.672\times\pi}{3}\times10^{-18}\,m^{3}$$

entonces la masa de la célula es de:

$$m = (1.098\times10^{3}\, kg/m^{3})(\frac{84.672\times\pi}{3}\times10^{-18}\, m^{3}) = 97.357806\times10^{-18}\, kg$$

realizando la conversión correspondiente a unidades de masa de la red:

$$m = 328.582594\, um$$

En adelante las cantidades físicas mostradas en todas las ecuaciones serán consideradas adimensionales y por lo tanto serán expresadas en unidades de lattice. De acuerdo con esto la densidad y la velocidad macroscópica se calculan de la siguiente manera:

$$\rho(\mathbf{x},t) = \sum_i f_i(\mathbf{x},t) \qquad (3.2)$$

$$\mathbf{u}(\mathbf{x},t) = \frac{\sum_i f_i(\mathbf{x},t)\mathbf{c}_i}{\rho(\mathbf{x},t)} \qquad (3.3)$$

Para la pruebas que se realizaron la densidad inicial ρ_0, es igual a 1 unidad de masa (*um*) por unidad de volumen (*uv*), el parámetro Δt se considerará igual a 1 paso de tiempo (*pt*), la longitud de un punto a otro de la red en sentido horizontal o vertical, Δx es 1 *unidad de lattice* o *ul*, la masa de una partícula es 1 *um*, y finalmente la velocidad básica de la red *c* será igual a $1\, ul/pt$.

En la Tabla 2 se listan las equivalencias de cantidades básicas mencionadas anteriormente en el sistema MKS, así como los parámetros del sistema circulatorio a utilizar como datos de entrada para la Prueba 1 del programa realizado. Dichos parámetros corresponden aproximadamente a los de una arteriola.

Para la Prueba 1 se utilizó una red de *LY* = 2000 *ul* x *LX* = 61 *ul* como se muestra en la Figura 6. En dicha figura se puede apreciar también que para la prueba (y para todas las demás pruebas) se alineará la vena con los ejes cartesianos de modo que la pared izquierda de la vena esté sobre el eje de las ordenadas y el final de la vena (es decir, por donde abandona el fluido) esté sobre el eje de las abscisas, además se colocará una célula de sangre roja al inicio de la vena capilar unos 6.103 μm a la izquierda del centro de la misma con velocidad inicial cero y se dejará arrastrar por el flujo sanguíneo en la dirección y sentido indicados hasta que llegue al otro extremo justo antes de que salga de la sección de la vena mostrada. Se registrarán como resultados los campos vectoriales de la torca y la fuerza resultantes sobre el centro de masa de la partícula así como de la velocidad angular de la misma, también se registrará la variación con respecto al tiempo de las componentes perpendiculares (al plano de la simulación) de la torca y la velocidad angular, entre otros datos.

Se consideran tres tipos de nodos en la red de Boltzmann: nodos de fluido, de

cantidad	símbolo	unidades de *lattice*	equivalencia en unidades MKS
unidad básica de longitud	Δx	1 ul	$\frac{2}{3}$ μm
paso de tiempo	Δt	1 pt	0.25×10^{-6} s
unidad básica de masa	Δm	1 um	2.963×10^{-16} kg
Ancho de la vena	$LX - 1$	60 ul	40 μm
Largo de la vena	LY	2000 ul	1333.333 μm
densidad inicial	ρ_0	1 um/uv	1×10^3 kg/m^3
tiempo de relajación	τ	2.525	
viscosidad cinemática del fluido	v	0.675 ul^2/pt	1.2×10^{-6} m^2/s
velocidad máxima del fluido en el centro de la vena	U	-0.01125 ul/pt	-0.03 m/s
semieje mayor de la partícula	a	6.3 ul	4.2 μm
semieje menor de la partícula	b	1.8 ul	1.2 μm
masa de la partícula	M	328.583 um	97.3578×10^{-15} kg
fuerza externa (dirección eje y)	F	-1.6875×10^{-5} $um \cdot ul/pt^2$	-5.333×10^{-14} N
posición inicial de la partícula	$\mathbf{x_0}$	(21.345 ul, 1996 ul)	

Tabla 2. Parámetros de entrada para la Prueba 1, la fuerza externa F se calculó tomando como base un flujo de Poiseuille que fluye en la vena. Si se considera una tercera dimensión z y que la vena tenga la forma de un cilindro, estas dimensiones de la red corresponde aproximadamente a 1333.333 μm x π x ((40/2) μm)^2 ≈ 1,675,516.082 (μm)^3 de sangre.

partícula y de frontera, que corresponden al plasma sanguíneo, la célula de sangre roja y las paredes de la vena.

En la misma Figura 6 se indica la numeración de los nodos que componen la red de Boltzmann tomando como origen del eje de coordenadas en 2D la esquina inferior izquierda de la vena, asimismo se muestran con líneas gruesas las paredes de la vena que es parte de la frontera del sistema con las cuales se usará la condición de frontera bounce-back de media red o enlace[31] (la cual consiste

[31] Se llama de medio enlace por que la distribución viaja, durante el rebote, ida y vuelta, en un tiempo Δt, un Δx, por lo que la frontera efectiva de la partícula esta a la mitad entre los nodos de frontera y los nodos de fluido.

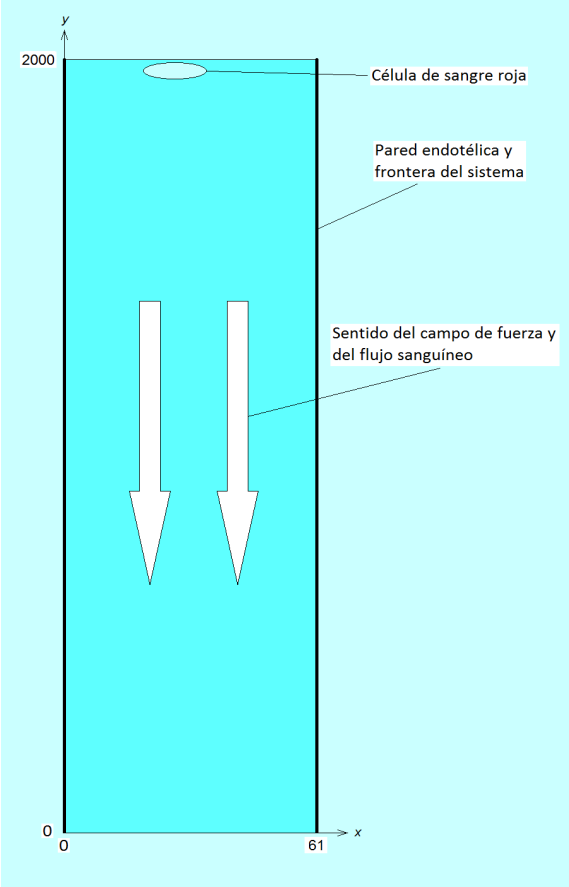


Figura 6. Esquema del sistema, indicando el sentido del flujo sanguíneo, la numeración de los nodos de la red, así como ejemplificando la posición inicial de la partícula.

básicamente en "rebotar" o proyectar la distribución del nodo de fluido en sentido opuesto al enlace que une dicho nodo con un nodo de la frontera[32] en el transcurso de un paso de tiempo), *no-slip* para la interacción de el fluido circundante con las paredes endotélicas de la vena, donde no-slip (no-deslizamiento) en la frontera significa que flujo neto de fluido en ella es cero [4]. Para la parte restante de la frontera del sistema, que es donde están ubicados los extremos de la vena, se usarán, al igual que [1], condiciones de frontera periódicas, las cuáles consisten simplemente en propagar las distribuciones en sentido "hacia fuera" al final de un extremo al principio del otro extremo (aplicando condiciones de frontera bounce-back de medio enlace en los nodos de fluido localizados en un extremo de la vena para aquellas distribuciones cuyo sentido sea sobre un enlace que une el nodo de fluido con un nodo de frontera en el otro extremo).

[32] O dicho de otra forma, para cada enlace i que une un nodo de fluido $\mathbf{x}$ y un nodo de frontera, la distribución correspondiente en el nodo $\mathbf{x}$, en el tiempo $t + \Delta t$, en el enlace i' exactamente opuesto a i es: $f_{i'}(\mathbf{x}, t+\Delta t) = f_i(\mathbf{x}, t^p)$, donde t^p indica que es la distribución en el tiempo t inmediatamente después de que ha ocurrido el paso de colisión. [4] También señala que con esta condición de frontera se consigue una mejor exactitud que con su similar, en la cual se considera que la frontera está *en* los nodos de la partícula, al costo de una pequeña adecuación o complicación a esta última.

El fluido se supone sujeto a una fuerza externa y se gobernará de acuerdo a la ecuación de Poiseuille[33]:

$$U_y(x) = -\frac{G}{2\mu}(x(l-x)) \qquad (3.4a)$$

la cual nos da la magnitud de la velocidad de un fluido que fluye por un canal que se encuentra alineado con respecto a los ejes cartesianos de la forma en que se muestra en la Figura 6 donde existe un gradiente de presión *G* constante en la dirección *y* con $G = \frac{\partial p}{\partial y}$, *l* es el ancho del canal, y μ es la viscosidad dinámica del fluido[34] [16] y [21]. Lo que nos indica la ecuación (3.4a) es que la magnitud de la velocidad en la dirección *y* forma una parábola con respecto al eje *x*, con el eje principal de dicha parábola desplazado *l*/2 hacia la derecha del origen del eje coordenado, que para el caso de la Figura 6 éste se situaría en la esquina inferior izquierda de la vena.

De acuerdo a lo anterior para calcular la fuerza externa *F* asociada al término K_α (campo de fuerza) de la ecuación (3.1) partimos de que el gradiente de potencial actuante en un fluido se podría definir como $G(x) = -\rho a_{accel}$, donde a_{accel} es la aceleración del fluido, substituyendo en la ecuación (3.4a) y despejando a_{accel} tenemos

$$a_{accel} = U_y(x)\frac{2\mu}{\rho(x(l-x))} \qquad (3.4b)$$

pero $\mu = \nu\rho$, substituyendo en la ecuación anterior queda

$$a_{accel} = U_y(x)\frac{2\nu}{(x(l-x))} \qquad (3.4c)$$

Ahora bien, debido a se aplicarán las condiciones de frontera de medio enlace en las paredes de la vena, significa que las distribuciones viajan sólo hasta la mitad de la distancia entre el nodo frontera y el nodo de fluido adyacente (recuérdese que la velocidad básica de la red es igual a $1^{ul}\!/_{pt}$), por lo que la longitud real del canal es de *LX* – 1 = 60 *ul*, asimismo la velocidad máxima adimensional, previamente calculada a partir de una velocidad de −0.03 *m*/*s* (tomada de [46] para un ancho de la vena de 40 μm, el signo negativo es debido a la orientación que tiene nuestro sistema respecto al sistema de referencia adoptado, vér Figura 6) mediante el factor de conversión correspondiente, y que se encuentra en el centro del canal (es decir, l = 60 *ul*, x = 30 *ul*) es de −0.01125 *ul*/*pt*, substituyendo lo anterior en la ecuación (3.4c) resulta

[33] Jean Léonard Marie Poiseuille (1797-1869), médico y fisiólogo francés interesado en el flujo sanguíneo humano, derivó experimentalmente su famosa ecuación alrededor de 1840 para tubos cilíndricos [45].

[34] Para nuestro análisis se consideraron la viscosidad dinámica, la viscosidad cinemática y la densidad, μ, ν y ρ como constantes.

$$a_{accel} = (-0.001125)\frac{2(0.675)}{(30(60-30))} = -1.6875\times10^{-5}\ ul/pt^2$$

Por último, tenemos de [W16] que la diferencial de la fuerza de masa o *fuerza másica* actuando en el fluido es:

$$dF = GdV \qquad (3.4d)$$

pero

$$dF = GdV = \rho a_{accel} dV \qquad (3.4e)$$

donde *dV* es la diferencial de volumen de dicho fluido que se está considerando en el cálculo, ahora bien, para nuestro caso la fuerza externa, el gradiente de presión y la densidad se considerarán como constantes, y por lo tanto la aceleración (solo despéjese de la ecuación anterior) es también constante, integrando la ecuación anterior tenemos,

$$\int dF = \int \rho a_{accel} dV = \rho a_{accel} \int dV$$

$$F = \rho a_{accel} V \qquad (3.4f)$$

para las condiciones iniciales $F = 0$, $V = 0$ (anulándose con ello la constante de integración). Finalmente, si consideramos la fuerza que actúa sobre una unidad de volumen (1 *uv*), para $\rho = 1\ um/uv$ (previamente seleccionada) y sabiendo que ésta es uniforme en todo el volumen del fluido, tenemos de la ecuación (3.4f),

$$F = (1\ um/uv)(-1.6875\times10^{-5}\ ul/pt^2)(1\ uv) = -1.6875\times10^{-5}\ ul\cdot um/pt^2 \qquad (3.4g)$$

3.2 Ecuaciones de Movimiento y Modelo Hidrodinámico.

El movimiento de la célula de sangre roja será, de acuerdo a las leyes de la mecánica clásica aplicadas a su centro de masa:

$$M\frac{d\mathbf{C}(t)}{dt} = \mathbf{F}(t) \qquad (3.5)$$

$$\mathbf{I}\cdot\frac{\mathrm{d}\,\boldsymbol{\Omega}(t)}{\mathrm{d}\,t} + \boldsymbol{\Omega}(t)\times(\mathbf{I}\cdot\boldsymbol{\Omega}(t)) = \mathbf{T}(t) \qquad (3.6)$$

$$\frac{d\mathbf{X}(t)}{dt} = \mathbf{C}(t) \tag{3.7}$$

según [40][35], con M, $\mathbf{C}(t)$, $\mathbf{F}(t)$, $\mathbf{I}$, $\boldsymbol{\Omega}(t)$, $\mathbf{T}(t)$ y $\mathbf{X}(t)$, la masa de la célula, la velocidad lineal, la fuerza resultante, el tensor de inercia, la velocidad angular, la torca y la posición, respectivamente. El movimiento de la célula o partícula solo será sobre un plano en dos dimensiones. (todo lo anterior referido al centro de masa de la partícula, $\mathbf{I}$ y $\mathbf{T}(t)$ se calculan respecto a un eje que atraviesa el centro de masa y que es perpendicular al plano donde rotará dicha partícula).

Ahora bien dado que se trabajará en 2D, la ecuación (3.6) queda de la siguiente forma:

$$I \cdot \frac{d\boldsymbol{\Omega}(t)}{dt} = \mathbf{T}(t) \tag{3.8}$$

con la velocidad angular $\boldsymbol{\Omega}(t)$ teniendo sólo el componente z en la dirección perpendicular al plano xy en el cual se mueve la partícula, lo mismo para la torca $\mathbf{T}(t)$, asimismo el tensor de momentos de inercia sólo tendrá el componente correspondiente al giro alrededor del eje perpendicular al plano xy que pasa por el centro de masa con un valor igual a:

$$I_{zz} = \frac{1}{4} M (a^2 + b^2) \tag{3.9}$$

según [W17], el cual corresponde al de una partícula con forma de elipse.

Ahora bien, se supondrá que durante un intervalo de tiempo, por ejemplo de t_0 a $t_0 + 1$, ocurrirá un intercambio de momento del fluido con la frontera efectiva de la partícula como se muestra en la Figura 7, y en donde, al igual que en el caso de la interacción del fluido con las paredes endotélicas, se aplicará una condición de frontera bounce-back de medio enlace, pero modificada según se muestra a continuación. Cabe hacer notar que para la aplicación de lo anterior se considerarán las partículas como un cuerpo sólido rígido.

La distribución de un nodo de fluido **x** adyacente a un nodo de la partícula sobre el enlace i, así como el momento transmitido a un nodo de partícula en la dirección del enlace i', está dada según [40] y [10], por:

[35] Aunque según [47] puede haber cierta inestabilidad en utilizar las ecuaciones (3.5) a (3.7) debido a que la densidad de masa de la partícula (para la prueba 1) es $\rho_p = 3M/(4\pi a^2 b) = 3 \times 328.5826/(4 \times \pi \times 6.3^2 \times 1.8) = 1.098$ um/ul^3 cercana a la densidad del fluido que es de 1 um/ul^3, previamente se realizaron varias pruebas con partículas de masa mucho menores que M sin que presentaran ningún tipo de comportamiento inestable.

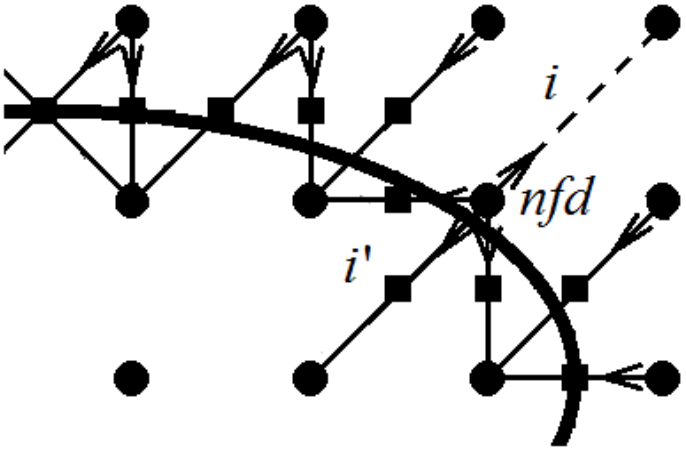


Figura 7. Sección de la partícula en la red de Boltzmann, se muestra la interacción de las distribuciones de los nodos de fluido que se encuentran afuera de la partícula dirigiéndose hacia los nodos internos de la misma así como los nodos frontera que se encuentran en la trayectoria (a medio camino o enlace) entre unos y otros nodos (en forma de pequeños cuadrados, los cuales en conjunto forman la frontera efectiva de la partícula), nótese el enlace i conectado al nodo de fluido *nfd* en donde las distribuciones parten de dicho nodo con dirección hacia fuera de la partícula y el enlace i' donde las distribuciones parten con dirección hacia la partícula, figura basada en [10].

$$f_i(\mathbf{x}, t+\Delta t) = f_{i'}(\mathbf{x}, t^p) + \frac{2\rho}{m} W_i \mathbf{c}_{nf} \cdot \mathbf{c}_i \qquad (3.10)$$

$$\Delta \mathbf{p}_{i'} = 2m\Delta x^3 \left(f_i(\mathbf{x}, t+\Delta t) - \frac{\rho}{m} W_i \mathbf{c}_{nf} \cdot \mathbf{c}_i \right) \mathbf{c}_{i'} \qquad (3.11)$$

considerando, al igual que [42], que la fuerza ejercida sobre la partícula es uniforme sobre todo el intervalo de tiempo [*t, t* + Δ*t*), donde $W_i = 3w_i/c^2$, w_i toma los valores según se mencionó en la sección **1.5** del capítulo 1, *c* es Δ*x*/Δ*t*, i' es el enlace exactamente opuesto a i, $\mathbf{c}_{nf}$ es la velocidad del nodo frontera situado entre $\mathbf{x}$ y $\mathbf{x}+\mathbf{c}_{i'}\Delta t$, y $f_{i'}(\mathbf{x}, t^p)$ es la distribución en el nodo $\mathbf{x}$ sobre el enlace o en la dirección i', inmediatamente después de que ha ocurrido el paso de colisión en dicho nodo (este tiempo posterior al paso de colisión se indica por t^p).

Substituyendo (3.10) en (3.11) tenemos:

$$\Delta \mathbf{p}_{i'} = 2m\Delta x^3 \left(f_{i'}(\mathbf{x}, t^p) + \frac{\rho}{m} W_i \mathbf{c}_{nf} \cdot \mathbf{c}_i \right) \mathbf{c}_{i'} \qquad (3.12)$$

De la ecuación anterior, la fuerza sobre la partícula en la dirección i' en el tiempo t_0 + ½ y la posición $\mathbf{x} + \frac{1}{2}\mathbf{c}_{i'}\Delta t$ (nótese que, debido a que es uniforme, la

magnitud de la fuerza es la misma en cualquier instante de tiempo), es decir sobre el nodo frontera de la partícula, es:

$$\mathbf{F}_{i'}(\mathbf{x}+\tfrac{1}{2}\mathbf{c}_{i'}\Delta t, t_0+\tfrac{1}{2}) = -\frac{2m\Delta x^3}{\Delta t}(f_{i'}(\mathbf{x},t^p)+\frac{\rho}{m}W_i\mathbf{c}_{nf}\cdot\mathbf{c}_i)\mathbf{c}_i \qquad (3.13)$$

La torca en el mismo punto de frontera, con respecto al centro de masa, de posición **X,** de la partícula es:

$$\mathbf{T}_{i'}(\mathbf{x}+\tfrac{1}{2}\mathbf{c}_{i'}\Delta t, t_0+\tfrac{1}{2}) = (\mathbf{x}+\tfrac{1}{2}\mathbf{c}_{i'}\Delta t-\mathbf{X}(t_0))\times\mathbf{F}_{i'}(\mathbf{x}+\tfrac{1}{2}\mathbf{c}_{i'}\Delta t, t_0+\tfrac{1}{2}) \qquad (3.14)$$

Asimismo la velocidad del nodo frontera correspondiente esta dado por:

$$\mathbf{c}_{nf}(t)=\mathbf{C}(t)+\boldsymbol{\Omega}(t)\times(\mathbf{x}+\tfrac{1}{2}\mathbf{c}_{i'}\Delta t-\mathbf{X}(t)) \qquad (3.15)$$

Aunque para el presente trabajo no se consideró, [1] indica que para el caso en que haya un contacto entre la partícula y alguna de las paredes endotélicas (o entre una partícula y otra partícula), la fuerza ejercida sobre la frontera de la partícula será:

$$\mathbf{F}_{i'}(\mathbf{x}+\tfrac{1}{2}\mathbf{c}_{i'}\Delta t, t_0+\tfrac{1}{2}) = 2f_{i'}^{eq}(\rho=\rho_0,\mathbf{u}=0)\mathbf{c}_{i'} \qquad (3.16)$$

dirigida hacia la partícula, donde el nodo de referencia $\mathbf{x}$ es un nodo de frontera del sistema (es decir pertenece a la pared endotélica de la vena). Nótese que la función de distribución utilizada es la función de equilibrio (ecuación (1.27)) para la velocidad $\mathbf{c}_{i'}$, inicializada con la densidad inicial ρ_0 (ver Tabla 2) y con velocidad macroscópica cero.

La torca correspondiente respecto al centro de masa de la partícula será:

$$\mathbf{T}_{i'}(\mathbf{x}+\tfrac{1}{2}\mathbf{c}_{i'}\Delta t, t_0+\tfrac{1}{2}) = (\mathbf{x}+\tfrac{1}{2}\mathbf{c}_{i'}\Delta t-\mathbf{X}(t_0))\times\mathbf{F}_{i'}(\mathbf{x}+\tfrac{1}{2}\mathbf{c}_{i'}\Delta t, t_0+\tfrac{1}{2}) \qquad (3.17)$$

La fuerza total resultante sobre el centro de masa de la partícula, en el tiempo t_0 + ½, es la suma de todas las fuerzas actuantes sobre la partícula según la Mecánica, lo mismo para la torca resultante en el mismo tiempo.

Ahora bien, para efectos de dar una mayor “suavidad” o exactitud a los valores calculados de la fuerza y de la torca **[48]**, se promediarán los obtenidos en el intervalo [t_0 - 1, t_0 + 1) para obtener los mismos en el tiempo t_0 como sigue:

$$\mathbf{F}(t_0)=\frac{1}{2}(\mathbf{F}(t_0+\tfrac{1}{2})+\mathbf{F}(t_0-\tfrac{1}{2})) \qquad (3.18)$$

$$\mathbf{T}(t_0)=\frac{1}{2}(\mathbf{T}(t_0+\tfrac{1}{2})+\mathbf{T}(t_0-\tfrac{1}{2})) \qquad (3.19)$$

Por simplicidad se utilizará el método de *Euler explícito* (es decir evaluando el paso siguiente utilizando la derivada del paso actual **[17]**) para el cálculo de **C**(*t*) y **Ω**(*t*) en el tiempo $t_0 + 1$ ($\Delta t = 1$):

$$\mathbf{C}(t_0 + 1) = \mathbf{C}(t_0) + \mathbf{F}(t_0) / M \tag{3.20}$$

$$\mathbf{\Omega}(t_0 + 1) = \mathbf{\Omega}(t_0) + \mathbf{T}(t_0) / I \tag{3.21}$$

Igualmente para el cálculo de $\mathbf{X}(t_0 + 1)$:

$$\mathbf{X}(t_0 + 1) = \mathbf{X}(t_0) + \mathbf{C}(t_0) \tag{3.22}$$

[1] Hace notar que si se compara la ecuación (3.16) con (3.11) y dependiendo de la elección del paso de tiempo, Δt, Δx y m, la ecuación (3.16) es exactamente el momento transferido entre la partícula y el fluido circundante en un paso de tiempo si se considera a la partícula como un sólido rígido en reposo y el fluido adyacente en reposo también y en estado de equilibrio, esto es importante ya que, si se considera una expansión futura del presente trabajo al estudio del flujo sanguíneo considerando varias partículas, debido a que en el modelo que se esta tomando como base para la presente simulación el fluido es incompresible y las velocidades son pequeñas, las fuerzas de lubricación que pueden resultar del contacto entre dos partículas son totalmente equilibradas y por lo tanto no causan atracción artificial entre las partículas, por lo que no es necesario hacer una corrección debido a estas fuerzas de lubricación como se hace normalmente en los modelos que consideran a las partículas como cuerpos sólidos rígidos.

3.3 Detalles de la Simulación.

Ahora bien para la simulación del movimiento de la partícula en el fluido se utilizaron como parámetros la posición (X_{prx}, X_{pry}) del extremo derecho del eje mayor de la partícula y el ángulo ϕ que el mismo forma con el eje *x*, ambos definidos con referencia al centro de masa de la partícula, donde el ángulo ϕ solo esta definido en el intervalo ($-\pi/2$, $\pi/2$), es decir si el ángulo ϕ excede $\pi/2$, por ejemplo, entonces automáticamente se toma como extremo derecho el otro extremo del eje mayor, y lo mismo si el ángulo ϕ disminuye menos que $-\pi/2$, de modo que el extremo derecho del eje mayor de la partícula siempre se encuentre en el primer o cuarto cuadrantes de un eje de coordenadas centrado en el centro de masa de dicha partícula.

El cálculo de la posición ($X_{prx}(t + \Delta t)$, $X_{pry}(t + \Delta t)$) se realizará de la siguiente forma: sabiendo de la cinemática rotacional que,

$$d\phi/dt = \Omega \qquad (3.23)$$

donde Ω es la rapidez angular instantánea (la cual es la magnitud de la velocidad angular mencionada anteriormente y que, al igual que la misma, es referida al centro de masa de la partícula), aplicando nuevamente el método de Euler explícito se tiene que la posición angular en el tiempo $t_0 + 1$ es,

$$\phi(t_0 + 1) = \phi(t_0) + \Omega(t_0) \qquad (3.24)$$

y en donde:

$$X_{prx}(t_0 + 1) = a(\cos(\phi(t_0 + 1))) \qquad (3.25a)$$

$$X_{pry}(t_0 + 1) = a(\text{sen}(\phi(t_0 + 1))) \qquad (3.25b)$$

donde a es el semieje mayor de la partícula (elipse).

Los nodos de la red que se encuentran al interior de la partícula serán nodos sin fluido por lo que a ningún nodo de dicha partícula se le aplicará la ecuación (3.10). Cuando la partícula se mueva, se hará según se menciona en [1], es decir a los nodos cubiertos por la misma se les borrará la información de las funciones de distribución f_i (**x**, **c**, t), en tanto que a los nodos descubiertos se les inicializará la distribución f_i (**x**, **c**, t) con la función f_i^{eq} de la ecuación (1.27), donde la velocidad macroscópica **u** se igualará con la velocidad que tenía el nodo correspondiente de la partícula que ocupaba el nodo un tiempo Δt anterior y cuya magnitud se calculará en base a la ecuación (3.15), y la densidad ρ será igual a la densidad inicial ρ_0.

El algoritmo para el cómputo de los valores hidrodinámicos mediante las redes de Boltzmann consta a grandes rasgos de la sucesión de dos pasos esenciales: el paso de colisión y el paso de propagación, en el primero se realiza el cálculo de los términos correspondientes al lado derecho de la ecuación (3.1) para todos los nodos de fluido solamente[36], el segundo paso es, como su nombre lo dice, la propagación de la distribución post-colisión de cada nodo de fluido a los nodos adyacentes si éstos son nodos de fluido, es decir la determinación del término $f_i(\mathbf{x} + \mathbf{c}_i\Delta t, t + \Delta t)$, en caso de que los nodos adyacentes sean nodos de partícula el paso de propagación es substituido por la ecuación (3.10), en caso de que los nodos adyacentes sean nodos de frontera el paso de propagación es substituido por la ejecución de la condición de frontera bounce-back de medio enlace, la cual consiste en “rebotar” o proyectar la distribución a su correspondiente en la dirección opuesta, para estos dos últimos casos se cumple la condición de no-slip para la frontera, es decir de velocidad cero del fluido para las inmediaciones de la frontera de la partícula y de la vena.

[36] En el programa secuencial se omite éste cálculo en el caso de los demás nodos (partícula y pared de la vena), y aunque para la paralelización del mismo explicado más adelante si se realiza éste cálculo para todos los nodos, de cualquier manera sólo se consideran los resultados obtenidos para los nodos de fluido.

En las Figuras 8a a 8e se muestra el diagrama de flujo de la parte del programa secuencial correspondiente al ciclo principal de cálculo, tomando como base para su construcción el algoritmo propuesto por [5].

Dicho proceso de cálculo se divide en cuatro partes: en la primera, que consta de un ciclo anidado que barre todos los nodos de la red, se realiza el cálculo de la densidad, las componentes cartesianas de la velocidad del fluido y el cálculo del paso de colisión para cada nodo de fluido en la red, los demás tipos de nodos son ignorados, esta primera parte corresponde a la Figura 8a.

Enseguida inicia el segundo ciclo anidado sobre cada nodo de la red, en el cual se realiza el paso de propagación, en donde, si el nodo actual en el ciclo es un nodo de fluido, se propagan las distribuciones correspondientes a cada velocidad en la misma dirección de la misma a todos los nodos adyacentes, los demás tipos de nodos se ignoran. Esta segunda parte corresponde a la parte media y superior de la Figura 8b.

A continuación inicia un tercer ciclo anidado sobre cada nodo de la red, en el cual se aplican las condiciones de frontera bounce-back de medio enlace tanto para las interacciones nodos de fluido–nodos de frontera como para las interacciones nodos de fluido–nodos de partícula. Para el caso en el que se encuentre un enlace fluido–particula la distribución del fluido actualizado será de acuerdo con (3.10) para cada enlace fluido–partícula, calculándose adicionalmente las contribuciones de fuerza y de torca sobre la partícula de acuerdo con (3.13) y (3.14). Similarmente, para cada enlace i nodo de fluido–nodo de frontera la distribución actualizada para el enlace i' será simplemente:

$$f_{i'}(\mathbf{x}, t+\Delta t) = f_i(\mathbf{x}, t) \qquad (3.23)$$

donde las distribuciones $f_i(\mathbf{x}, t)$ para los cálculos anteriores se tomarán de las distribuciones propagadas de los nodos de fluido a los nodos de partícula y de frontera respectivamente. Nótese que esto determina correctamente la distribución $f_{i'}(\mathbf{x}, t)$ en el siguiente paso de tiempo, es decir $f_{i'}(\mathbf{x}, t+\Delta t)$ (véase el tercer pie de página del presente capítulo). Lo anterior corresponde a la parte inferior de la Figura 8b y la Figura 8c.

En la cuarta parte del algoritmo, en base a las contribuciones de la fuerza y de la torca de los pasos anteriores se realizan las actualizaciones de las correspondientes variables globales y de las variables hidrodinámicas restantes que se consideraron (velocidad lineal, velocidad angular y posición), todo lo anterior calculándolo en base a (3.18) a (3.22), respectivamente. Asimismo se actualiza la posición del extremo derecho utilizando para su cálculo (3.24), (3.25a) y (3.25b). Esta cuarta parte corresponde a la parte superior de la Figura 8d.

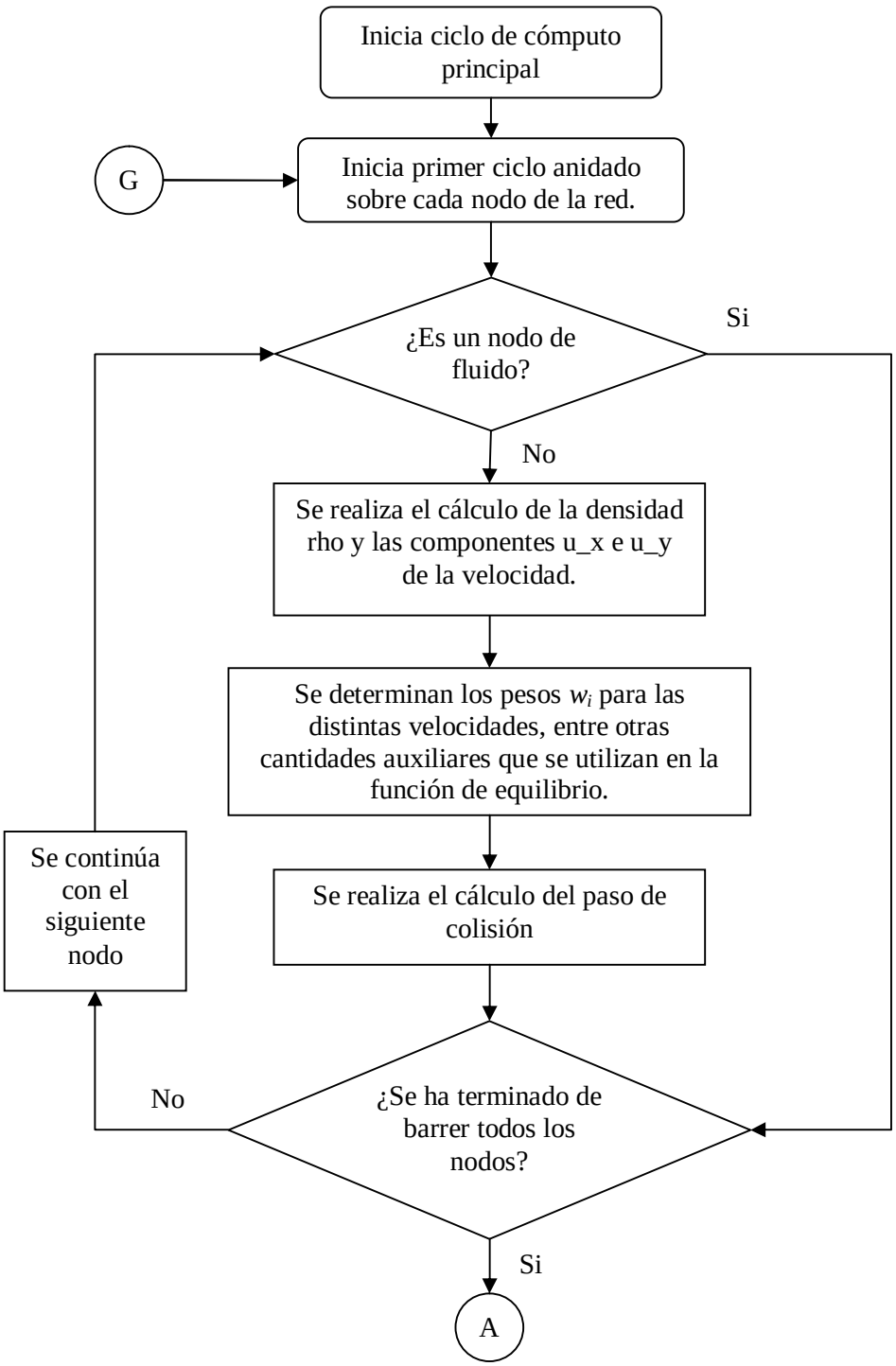


Figura 8a. Diagrama de flujo de la sección de código correspondiente al ciclo principal de cálculo.

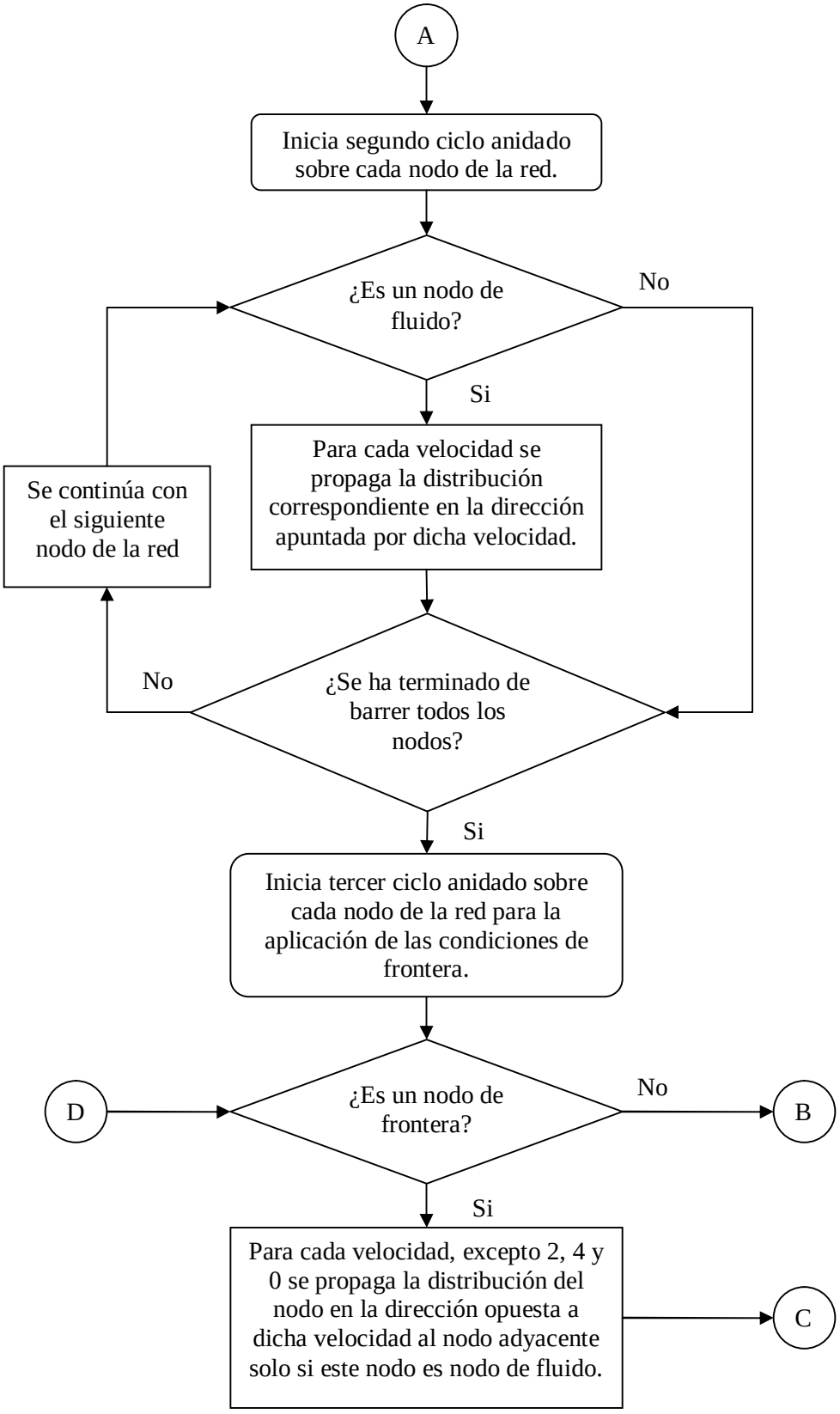

A
Inicia segundo ciclo anidado sobre cada nodo de la red.
¿Es un nodo de fluido?
No
Si
Para cada velocidad se propaga la distribución correspondiente en la dirección apuntada por dicha velocidad.
Se continúa con el siguiente nodo de la red
¿Se ha terminado de barrer todos los nodos?
No
Si
Inicia tercer ciclo anidado sobre cada nodo de la red para la aplicación de las condiciones de frontera.
D
¿Es un nodo de frontera?
No
B
Si
Para cada velocidad, excepto 2, 4 y 0 se propaga la distribución del nodo en la dirección opuesta a dicha velocidad al nodo adyacente solo si este nodo es nodo de fluido.
C

Figura 8b.

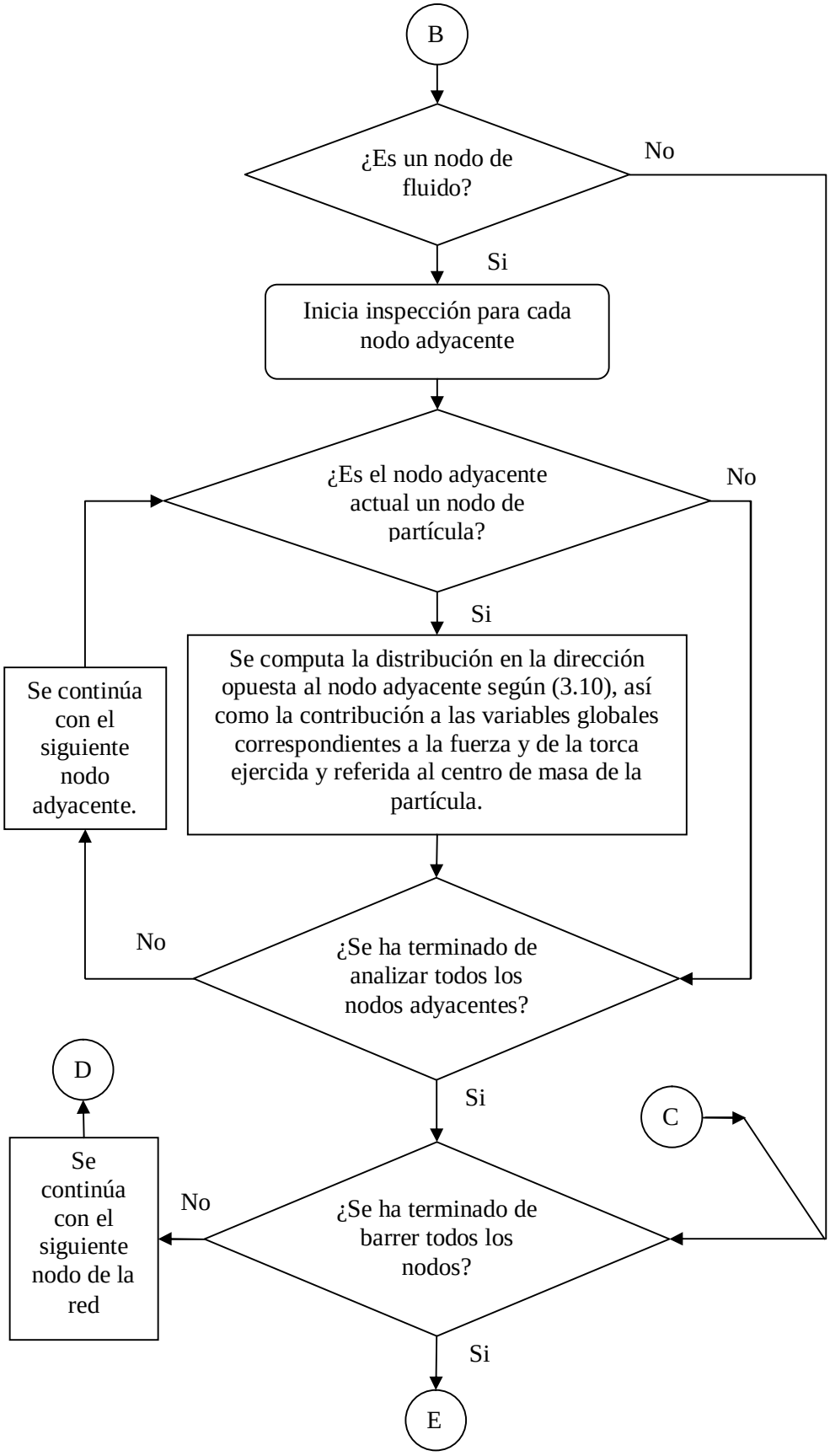

B
¿Es un nodo de fluido?
No
Si
Inicia inspección para cada nodo adyacente
¿Es el nodo adyacente actual un nodo de partícula?
No
Si
Se computa la distribución en la dirección opuesta al nodo adyacente según (3.10), así como la contribución a las variables globales correspondientes a la fuerza y de la torca ejercida y referida al centro de masa de la partícula.
Se continúa con el siguiente nodo adyacente.
No
¿Se ha terminado de analizar todos los nodos adyacentes?
Si
C
D
Se continúa con el siguiente nodo de la red
No
¿Se ha terminado de barrer todos los nodos?
Si
E

Figura 8c.

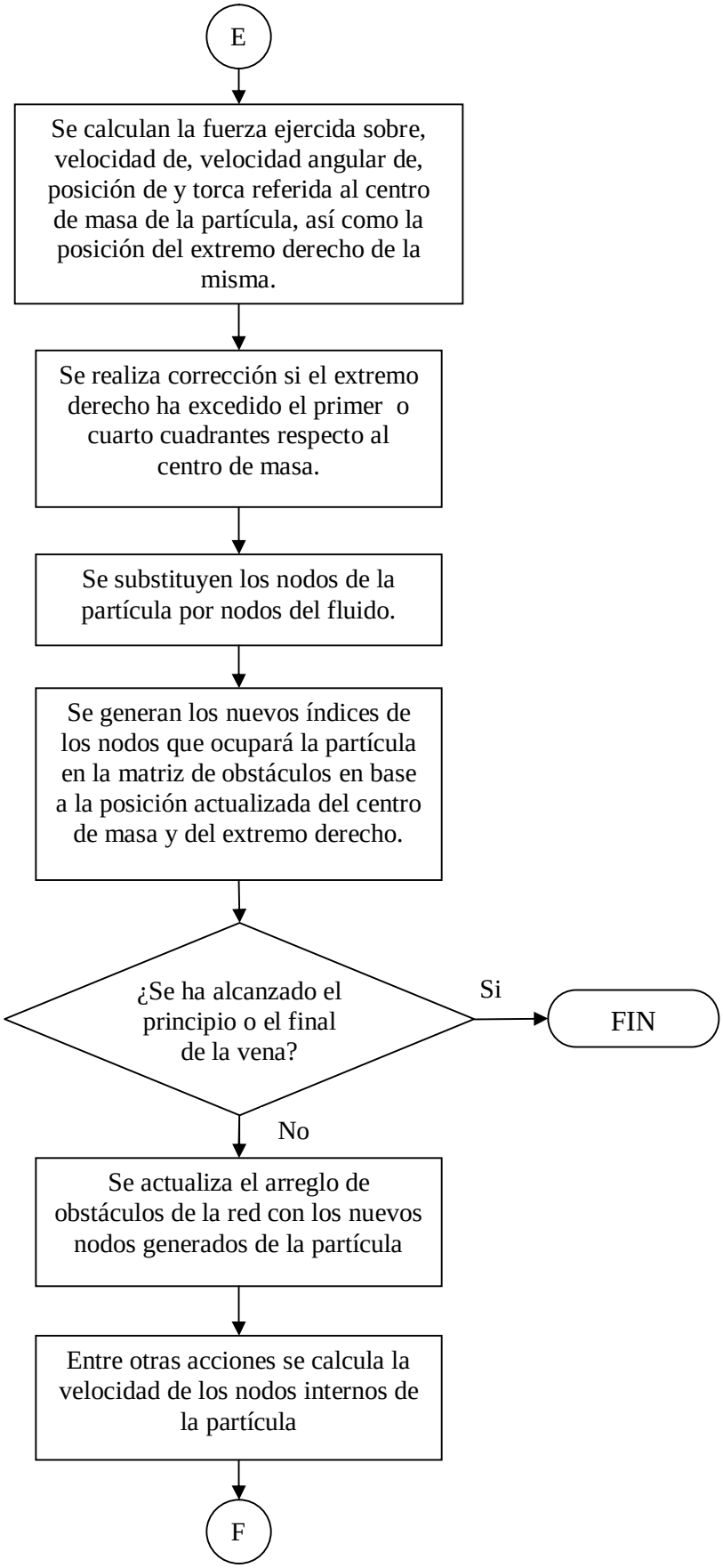

E
Se calculan la fuerza ejercida sobre, velocidad de, velocidad angular de, posición de y torca referida al centro de masa de la partícula, así como la posición del extremo derecho de la misma.
Se realiza corrección si el extremo derecho ha excedido el primer o cuarto cuadrantes respecto al centro de masa.
Se substituyen los nodos de la partícula por nodos del fluido.
Se generan los nuevos índices de los nodos que ocupará la partícula en la matriz de obstáculos en base a la posición actualizada del centro de masa y del extremo derecho.
¿Se ha alcanzado el principio o el final de la vena?
Si
FIN
No
Se actualiza el arreglo de obstáculos de la red con los nuevos nodos generados de la partícula
Entre otras acciones se calcula la velocidad de los nodos internos de la partícula
F

Figura 8d.

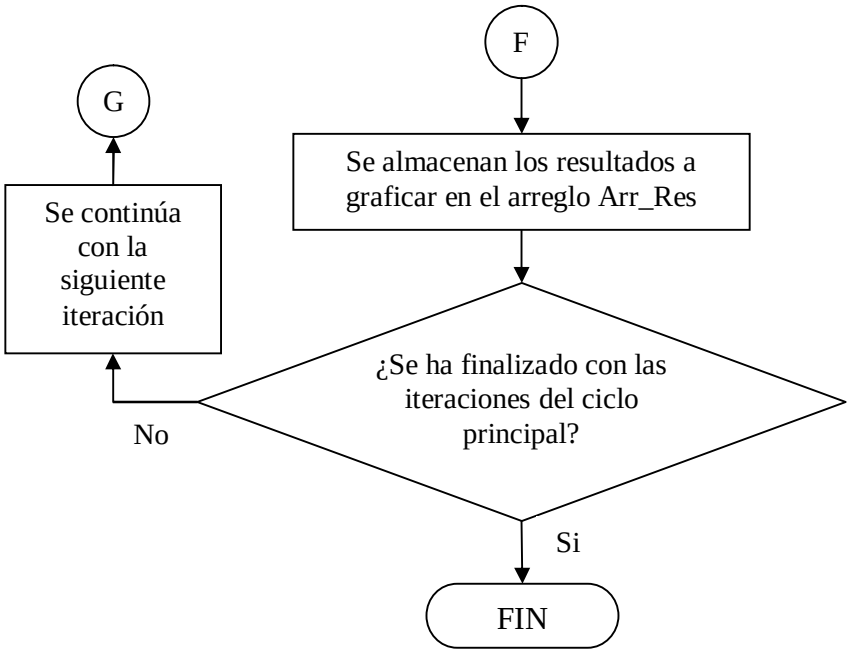


Figura 8e.

Posteriormente se realizan las acciones de lógica de control de la simulación, consistiendo éstas en establecer las nuevas coordenadas del extremo derecho si éste ha excedido el primer o cuarto cuadrantes, substituir los nodos de la partícula por nodos de fluido, cuyas distribuciones f_i, como ya se mencionó, serán inicializadas mediante la ecuación (1.27) donde la velocidad macroscópica **u** y la densidad serán inicializadas con la velocidad que tenga el nodo correspondiente de la partícula y la densidad inicial ρ_0, utilizando para esto último la ecuación (3.15) y un arreglo temporal que almacena las velocidades calculadas de cada nodo; enseguida se determinan los índices de los nodos de la partícula para las posiciones actualizadas del centro de masa y del extremo derecho, se comprueba si la partícula ha alcanzado la salida o entrada de la vena para que en caso de ser así se finalice con el ciclo principal de cálculo, se actualiza la matriz de obstáculos con los nuevos índices de los nodos de la partícula, etc., finalmente se almacenan los resultados a graficar posteriormente y se verifica si existen más iteraciones (cuyo número se selecciona apropiadamente grande y está predeterminado) por realizar, de modo que finalizará el ciclo principal de cálculo si se ha agotado el número dado de iteraciones o si, como ya se mencionó, la partícula ha alcanzado la entrada o salida de la vena, esta última parte corresponde a la parte restante de la Figura 8d y la Figura 8e.

Como se observa en la Figura 8a la determinación de la densidad y la velocidad del fluido, la función de equilibrio y su posterior utilización en la determinación del paso de colisión son tareas muy factibles de realizar por un GPU pues todas las cantidades se pueden calcular localmente en el nodo que actualmente se esté tratando, inclusive el cálculo del paso de colisión puede realizarse como una serie

secuencial de grupos de instrucciones, uno por cada velocidad, con lo que se minimiza el uso de sentencias bifurcativas tales como *if*, lo cual optimiza el rendimiento de la ejecución de los kernels en los GPUs.

Con lo que queda la selección de la tecnología a utilizar para la ejecución del segundo y tercer ciclos de cálculo que comprende el cómputo del paso de propagación, la aplicación de las condiciones de frontera de bounce-back de medio enlace para el caso de las interacciones del fluido con la frontera del sistema y la frontera de la partícula, añadiendo al último caso, además, el efecto de la transferencia del momento al fluido, así como el cálculo de las componentes cartesianas de la fuerza, la velocidad lineal, la velocidad angular y la torca resultantes de la acción que ejerce el fluido sobre la partícula. Un breve análisis indica que quizá sería factible de paralelizar mediante el GPU al menos el paso de propagación en el segundo ciclo anidado, esto podría hacerse separando dicho paso y conjuntarlo con el primer ciclo anidado de una manera similar a como se menciona en [W18], y añadiendo el cálculo de la acción que ejercen las paredes de la vena sobre la partícula al tercer ciclo anidado, pero dado su potencial tratamiento para una utilización óptima en el CPU debido a la intrínseca no localidad del cálculo se prefirió dejar que el paso de propagación sea realizado mediante otro estándar de paralelización considerando su posible paralelización mediante el GPU para trabajos futuros.

En cuanto al tercer ciclo anidado, dada su complejidad y la extensión del cálculo que abarca (que depende del tamaño y forma de la red y de la partícula, por ejemplo para la Prueba 1 realizada se calcula que no excederá del 1.77% comparado contra la cantidad de cálculo del paso de colisión) se considera que una elección del estándar MPI u OpenMP es perfectamente aplicable a este caso pues se pueden aprovechar las distintas optimizaciones, mejoras y demás innovaciones tecnológicas que puede ofrecer un núcleo CPU actual; debido a que se considera más adecuado para la continuación del presente trabajo[37] se eligió el estándar MPI para la implementación del segundo y tercer ciclo anidados, respectivamente. Para la ejecución de la cuarta y última parte se consideró, dada su reducida complejidad, extensión e impacto en el tiempo total de cómputo del ciclo principal, que es suficiente procurar optimizar su ejecución en forma secuencial seleccionando, donde sea recomendable, su ejecución solo por el nodo maestro de MPI.

En la Figura 9 se muestra el diagrama de flujo del ciclo principal de cálculo en donde se utilizó una arquitectura combinada CUDA − MPI. Tal y como se anticipó en el párrafo anterior el primer ciclo anidado es ejecutado mediante una función que llama al kernel de CUDA mientras que el resto del cálculo es ejecutado por cada uno de los nodos de MPI que se hayan levantado, que en nuestro caso fueron 7 de los 8 posibles núcleos[38] del procesador AMD® FX 8150, asociando un nodo de MPI a cada núcleo y a los cuales se les destina la parte de trabajo correspondiente al paso

[37] Aunque se han hecho esfuerzos por adaptar el modelo de memoria compartida de OpenMP para que funcione de manera igual de eficiente o que incluso supere a MPI en un sistema distribuido (véase por ejemplo [W19]), lo cierto es que MPI sigue siendo el estándar más ampliamente utilizado para trabajar bajo este tipo de arquitectura, la cuál probablemente sea el objetivo a alcanzar como continuación del presente trabajo.
[38] Fue el número de núcleos en el que se observó una mejor eficiencia.

de colisión y las restantes tareas del programa que incluyen la lógica de control de la simulación, es necesario señalar que, adelantándonos un poco a los resultados de las pruebas realizadas, las pruebas fueron ejecutadas asignando a un mismo GPU los kernels de todos los nodos MPI, notándose que las mismas se ejecutaron de forma relativamente estable a pesar de que los kernels se ejecutan de forma secuencial debido a que CUDA no puede ejecutar concurrentemente kernels con distinto contexto de ejecución (siendo éste nuestro caso ya que cada nodo MPI es un proceso distinto con su propio entorno o contexto de ejecución) **[32]**, siendo lo más recomendable que cada nodo MPI llame de manera exclusiva a un GPU para ejecutar trabajo en él.

Cabe mencionar que la arquitectura es genérica, en el sentido de que, aunque el programa se corrió bajo una arquitectura convencional, es posible correrla también, por ejemplo, en un clúster de computadoras o en cualquier otra arquitectura que admita el estándar MPI. La arquitectura también puede crecer, es decir, aunque las pruebas se corrieron asignando a cada nodo MPI un solo GPU, se le podrían asignar a cada nodo MPI más de un GPU y aún permitir que un nodo de MPI pueda acceder al GPU asignado a otro nodo, aunque esto último complicaría considerablemente la lógica de control, y que además probablemente sería inútil pues aumentaría también la inestabilidad del programa debido a la gran cantidad de cambios de contexto que haría CUDA al ejecutar kernels llamados por nodos MPI distintos ya que, como ya se mencionó CUDA no puede ejecutar concurrentemente kernels con distinto contexto.

Es importante señalar que la arquitectura combinada CUDA-MPI es la principal aportación del presente trabajo al cómputo numérico, es decir la utilización de una arquitectura combinada GPU-nodos de ejecución MPI independientes que permiten paralelizar eficientemente los pasos de colisión y de propagación del método de las redes de Boltzmann debido a que aprovechan las ventajas de cada interfaz de paralelización distribuyendo adecuadamente las partes del cómputo a aquella tecnología que lo eficientiza de manera óptima según las recomendaciones hechas por el mismo fabricante, en el caso de los GPUs, y la literatura especializada para el caso de las arquitecturas convencionales de hardware que incluyen los más avanzados CPUs multinúcleo.

La forma en que se diseñó la paralelización del sistema fue la siguiente: dado que básicamente el sistema operativo corre *N* procesos distintos e independientes en *N* núcleos de procesador, el cómputo se realiza dividiendo la red de Boltzmann de *LX* x *LY* nodos de red primeramente en franjas de *LX* x ((*n*+2) nodos) x (9 velocidades) datos, con *n* = 1, 2,…, $floor\left(LY/N\right)$ para las primeras *N*-1 franjas y *n* = 1, 2, …, $floor\left(LY/N\right)+\mathrm{mod}\left(LY/N\right)$ para la *N*-ésima franja (el +2 es debido a que se agregan dos filas de LX x (9 velocidades) datos cada una, arriba y debajo –a los que llamaremos límite superior e inferior respectivamente– de cada franja para comunicación con los demás procesos según se muestra en la Figura 10), donde

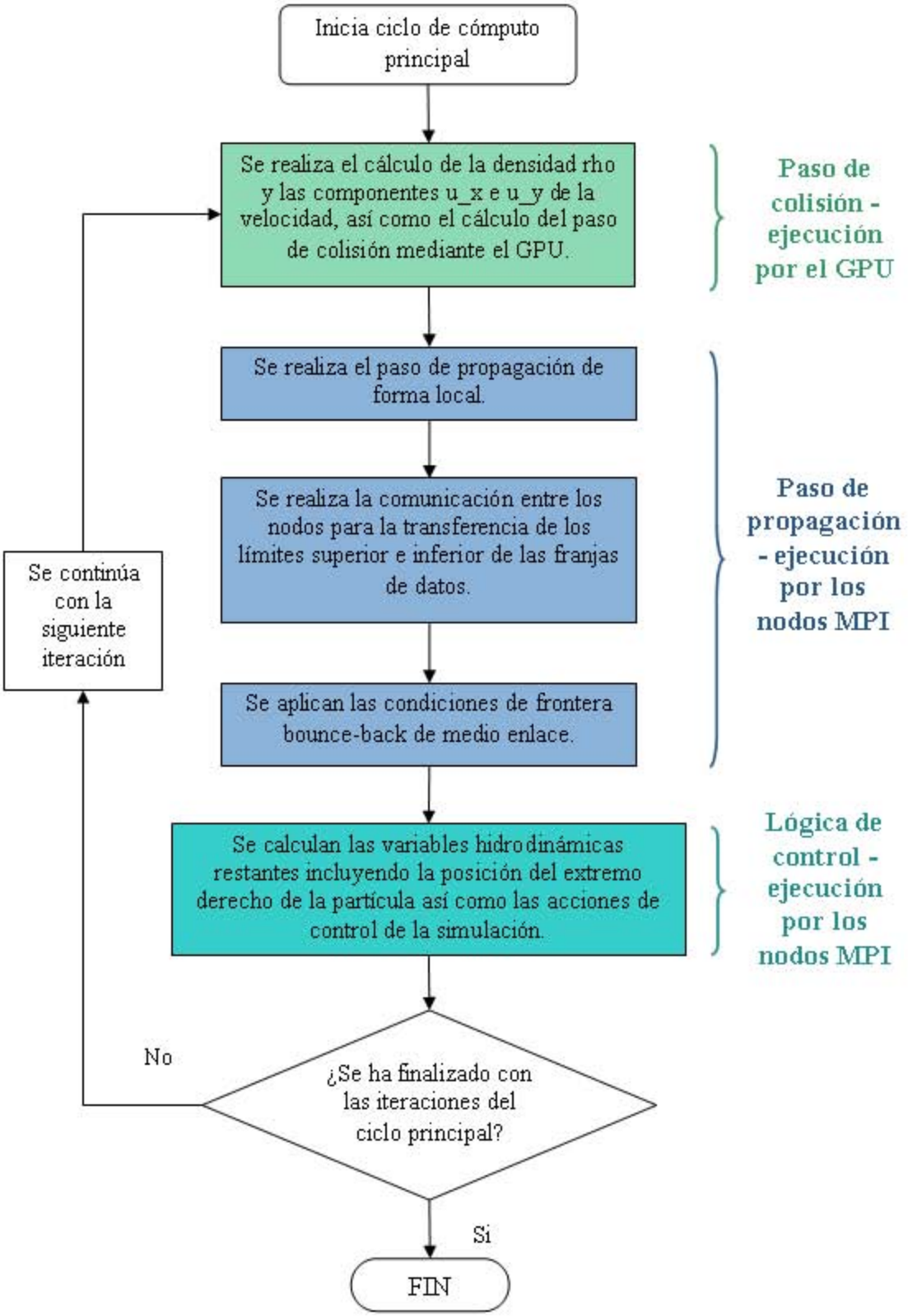


Figura 9. Diagrama de flujo del ciclo principal de cálculo del programa basado en la arquitectura combinada CUDA – MPI.

cada núcleo (o nodo MPI) corre una de estas franjas en forma simultánea con los demás núcleos. Posteriormente se realiza un somero balanceo de carga distribuyendo las filas de nodos de la red adicionales (es decir los datos asociados) en el último nodo MPI a los demás nodos MPI comenzando por el primero. Esta manera de particionar el cómputo es bastante simple, relativamente fácil de programar y que, si suponemos que la longitud del sistema *LY* es divisible por el número de procesadores *N*, se tiene un radio de comunicación a cómputo[39] $O\left(2LX \middle/ \left(LX \times LY/N\right)\right) = O\left(N \middle/ LY\right)$, lo cual se consideró aceptable pues para nuestro caso *LY* >> *N*. El cómputo del paso de colisión lo realiza cada nodo de MPI de manera local independiente de los demás nodos.

Es necesario mencionar que cada una de las franjas de datos asociadas a las correspondientes de nodos de la red son transferidas para su procesamiento en el GPU dentro de cada nodo de ejecución de MPI seleccionando bloques de hilos de 8 x 32, opción que permite utilizar siempre todos los hilos de un multiprocesador, cabe indicar que el número de registros utilizado por multiprocesador no excede el límite permitido por el mismo antes de que haya una *desutilización* de hilos en el multiprocesador por esta causa.

El detalle de prácticamente todas las tareas (haciendo, desde luego, las adecuaciones pertinentes para su ejecución en paralelo y además, según la tecnología utilizada para tal efecto) indicadas en la Figura 9 es exactamente el mismo de sus correspondientes en las Figuras 8a a 8e, pero realizadas por la parte correspondiente de la arquitectura combinada.

Para el cómputo del paso de propagación (donde las distribuciones de los extremos superior e inferior de las franjas de datos deben propagarse a los extremos inferior y superior correspondientes de las franjas situadas inmediatamente arriba y abajo de cada una según la organización esquemática de las mismas mostrado en la Figura 10) se realizó la comunicación entre los nodos de MPI utilizando una topología en anillo y el modo de comunicación síncrono mediante las llamadas MPI correspondientes[40]. Esta comunicación entre los nodos se realiza de la manera siguiente: se dividió la comunicación en envío/recepción del límite superior y envío/recepción del límite inferior de la franja de datos, donde en cada parte un nodo MPI inicia o "dispara" la transmisión/recepción enviando el límite superior o inferior según sea el caso al extremo inferior o superior de la franja correspondiente al nodo MPI destino, una vez terminada la transmisión/recepción, éste nodo destino envía su límite superior o inferior al extremo inferior o superior de la franja del siguiente nodo destino, y así se continua hasta que la transmisión/recepción del límite superior o inferior de las

[39] Se define como la cantidad de comunicación (en bytes) dividida por el tiempo de cómputo y que puede ser una medida del impacto que causa dicha cantidad de comunicación en el tiempo de cómputo global [34].

[40] Esta selección se realizó debido a que en un inicio se pensó que podría reducir la inestabilidad en la ejecución del programa debido posibles errores en las comunicaciones entre los procesos de MPI que se producían en ejecuciones realizadas de pruebas previas a las definitivas aquí mencionadas, posteriormente se cayó en la cuenta de que los errores producidos no eran propiamente debido a problemas en las comunicaciones, sino a que al parecer se estaba saturando la memoria convencional con demasiados programas residentes de utilería (por ejemplo el antivirus).

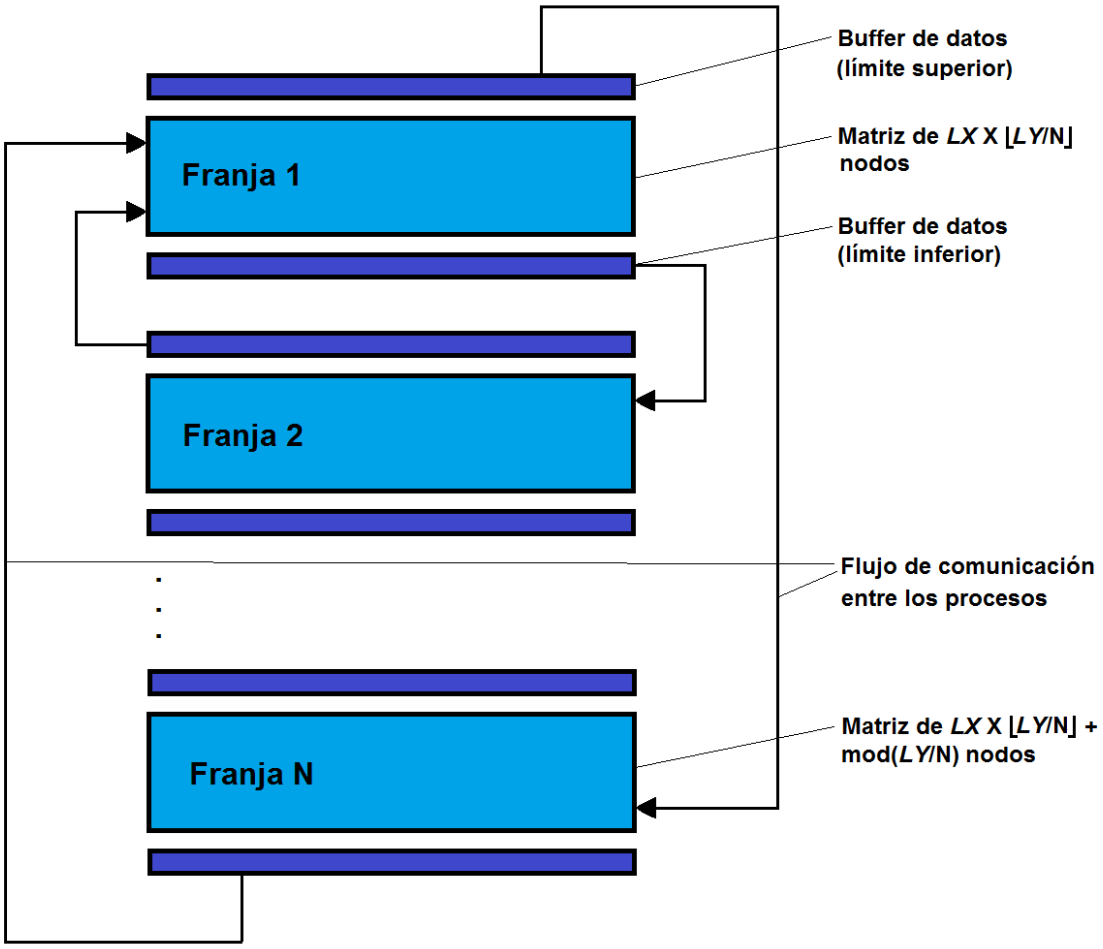


Figura 10. Flujo de comunicación entre las franjas de datos que componen a la red de Boltzmann de la simulación.

franjas finaliza con la recepción del límite superior o inferior de datos en los extremos correspondientes de la franja de datos del nodo MPI que inició el proceso.

Ahora bien, el cálculo de las variables tales como la Fuerza y Torca total resultantes sobre la partícula por parte del fluido, la velocidad de dicha partícula, etc., será común a todos los nodos de MPI, es decir cada nodo MPI realizará los mismos cálculos y, dado que serán alimentados con los mismos datos de entrada, se obtendrán los mismos resultados (lo que se comprobó verificando que los resultados obtenidos en todos los nodos son los mismos al final del ciclo de ejecución), lo mismo aplica para la actualización de la matriz de obstáculos y otros cálculos complementarios, como por ejemplo el cálculo de la velocidad de los nodos internos de la partícula[41], sin embargo si se distribuyó la carga de trabajo entre los

[41] Debido a que en algunas de estas tareas se involucran operaciones que no son $O(1)$ (en especial las directamente relacionadas con la partícula, las cuales de hecho son a lo más $O(2)$ e incluyen principalmente operaciones aritméticas y operaciones sencillas de control) podría parecer que esta forma de realizar dichas tareas en el programa afectaría severamente el performance del mismo, sin embargo, con excepción de la

distintos nodos MPI donde fue aplicable, que en el programa esto se realizó en el ciclo donde se actualizan los nodos de la partícula por nodos de fluido conforme dicha partícula se va moviendo, en el cual solo se actualizan los datos del arreglo de valores de la función de distribución asignado a cada nodo. Finalmente, la lógica interna que controla el movimiento paulatino de la partícula desde que sale de un extremo de la vena hasta que llega al otro extremo es global y común a todos los nodos MPI. En el apéndice A se muestra el código del ciclo principal programa paralelo, así como el código del kernel de CUDA del mismo programa, realizándose estos en la implementación de C++ proporcionada por Microsoft® Visual Studio® 2010 Profesional.

Se corrieron tres pruebas (Prueba 2) para las viscosidades del fluido de 2, 3 y 4 *cP* partiendo de un punto más cercano a una de las paredes de la vena (a comparación de la Prueba 1) de manera parecida a como se menciona en [44] donde, según los resultados obtenidos en dicho artículo, la célula de sangre roja tiende a moverse hacia el centro del canal, los valores iniciales de la simulación son similares a los de la Prueba 1 y se resumen en la Tabla 3 (al igual que para la Prueba 1, la velocidad inicial de la partícula para todos los casos será cero, lo mismo aplica para todas las demás pruebas realizadas). Esta y las demás tablas restantes se muestran en el apéndice C.

Cabe hacer notar que en la prueba se mantuvo la velocidad del fluido constante variando el campo de fuerza y el tiempo de relajación τ, de ahí los diferentes valores indicados para estos parámetros.

actualización de la matriz de obstáculos, dichas tareas (una por nodo de partícula) son realizadas para una sola partícula, que para nuestro caso se calcula que son alrededor de 40 nodos de partícula aproximadamente.

Capítulo 4

Resultados y Análisis.

En la Tabla 4 se tabula una muestra de 50 conjuntos de datos de los resultados obtenidos para la Prueba 1 formados por tiempo, ángulo de inclinación de la partícula, velocidad angular, torca, y las componentes cartesianas de la posición, velocidad lineal y fuerza, para el tiempo de la simulación, 0 a 4.9 x 10^{-2} *s,* con las unidades en el Sistema Internacional.

Para la misma Prueba 1, en la Figura 11a se muestra la variación respecto al tiempo de la componente Y de la posición y en la Figura 11b la variación respecto al tiempo de la componente Y de la velocidad.

De la Figura 11a se observa como la partícula se mueve a lo largo del eje Y de manera prácticamente uniforme según lo indica la línea recta que se traza en dicha figura con valores de la pendiente (velocidad lineal) mostrados en la Figura 11b, para los cuales se observa que sólo al principio (hasta aproximadamente 0.001s y considerando adicionalmente los datos para la velocidad lineal Y mostrados en la Tabla 4), muestran una rápida variación en la velocidad de 0 a −0.0266 m/s, pero posteriormente tienden a permanecer relativamente estables, lo cual es un indicio de que la célula de sangre roja realiza un movimiento suave (sin grandes variaciones en el desplazamiento) a lo largo de la vena[42] para las escalas espaciales y temporales consideradas.

En las Figuras 11c y 11d se muestra la variación temporal de la fuerza X y la torca, respectivamente.

De la Figura 11d se observa como la torca, a pesar de una variación aparentemente al azar, sigue un cierto patrón durante el intervalo de tiempo de la simulación como puede apreciarse, ello es congruente con el “giro” que realiza la célula alrededor de su propio eje como se muestra más adelante en la Figura 11f, así como la forma definida que toma la correspondiente variación temporal de la velocidad angular mostrada en la Figura 11g.

De hecho, de las Figuras 11c a 11g (exceptuando la Figura 11e) se observa como la fuerza, la torca, la velocidad angular y el ángulo de inclinación de la

[42] Para el caso de la posición X nótese el minúsculo desplazamiento (del orden de 10^{-1} μm) que realiza la partícula (incluyendo en esto las variaciones que se producen en el mismo) mostrados en la Tabla 4.

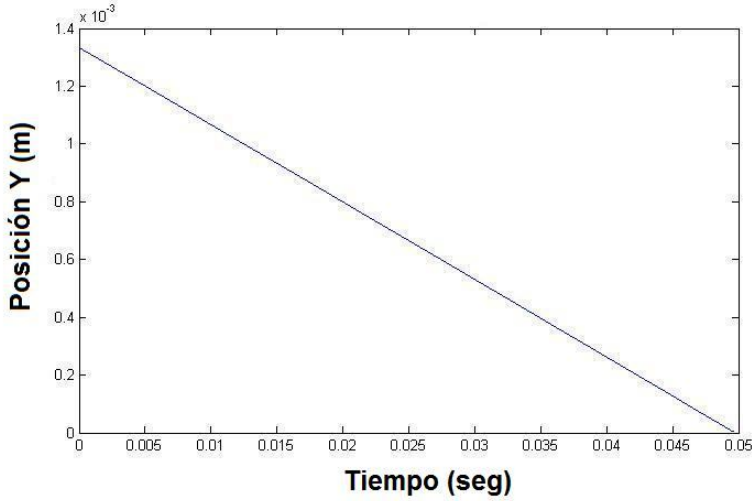


Figura 11a. Posición Y vs Tiempo.

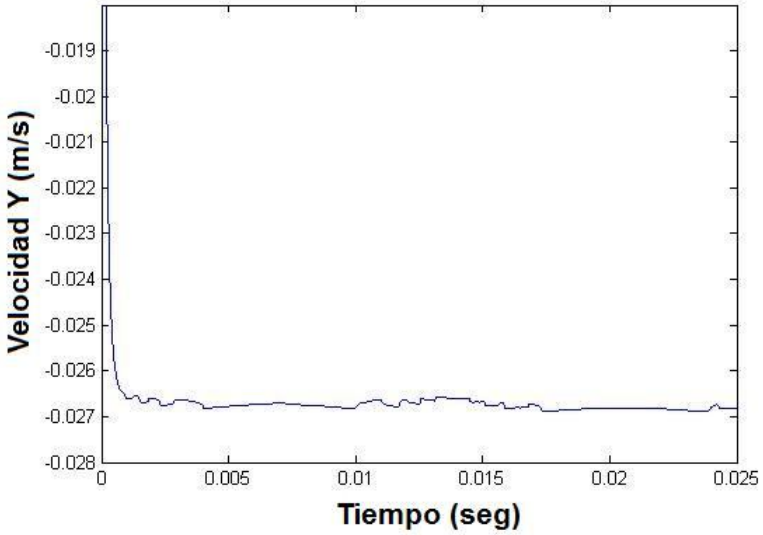


Figura 11b. Velocidad lineal Y vs. Tiempo.

partícula guardan muy similar patrón de variaciones con respecto al tiempo. Se observa que la forma de las gráficas de la velocidad angular y el ángulo de inclinación esta de acuerdo a las mostradas en [46], observándose que la partícula realiza un movimiento de rotación o vuelta (recuérdese que se tomó la convención de que el ángulo de inclinación de la partícula se tomaría con respecto al extremo derecho actual de la partícula en la simulación, lo que significa que solo toma valores entre $-\pi/2$ y $\pi/2$) lo cual esta de acuerdo a lo observado también en [49] y con diversos estudios previos según menciona [50].

En la Figura 11e se muestra un detalle del campo vectorial de la fuerza.

En la Figura 12 se muestra la variación respecto al tiempo de la posición X del

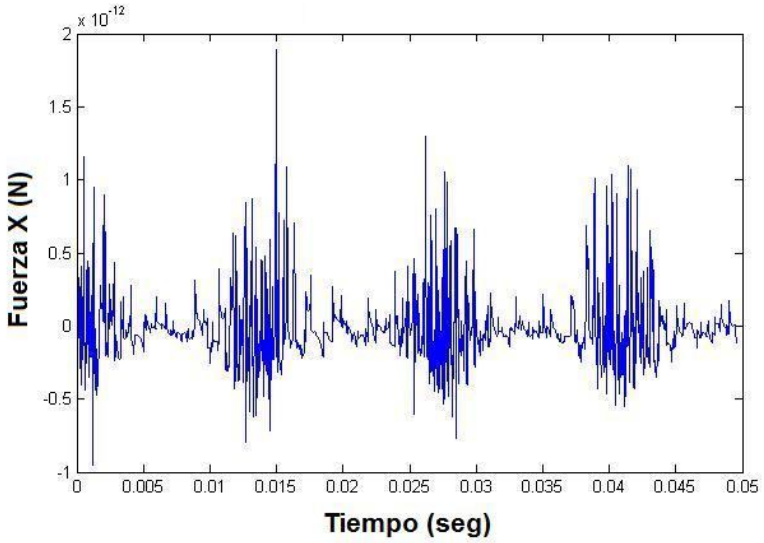


Figura 11c. Fuerza X vs. Tiempo.

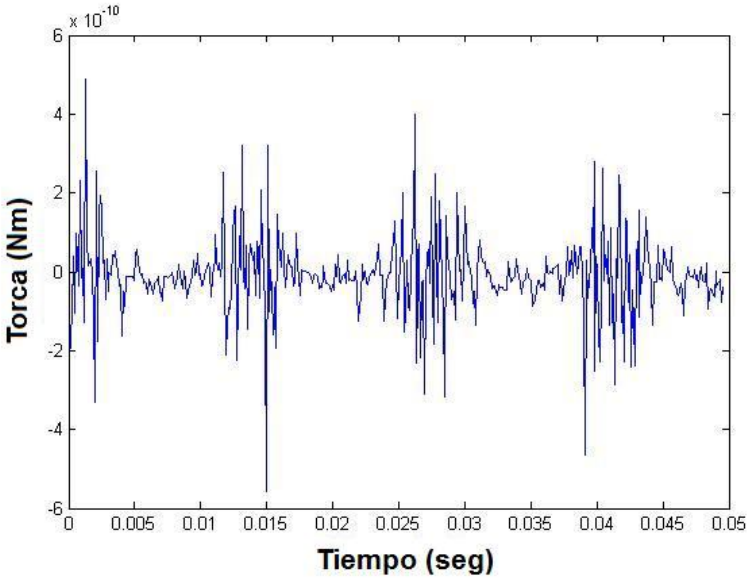


Figura 11d. Torca vs. Tiempo.

centro de masa de la partícula para el primer caso de la Prueba 2 (μ = 2 *cP*) donde se observa la tendencia de la partícula a acercarse o migrar al centro de la vena, sin embargo debido a que la velocidad de migración es muy baja se decidió correr una tercera prueba adicional con los mismos valores de la Prueba 2 con la diferencia de seleccionar una longitud básica de la red y un paso de tiempo más grande (para poder abarcar más espacio y tiempo de simulación respectivamente) y hacer más notable el comportamiento de migración al centro que tiene la célula de sangre roja, el resumen de los valores utilizados para la Prueba 3 así como la variación temporal de la Posición X se muestran en la Tabla 5 y Figura 13 respectivamente. De esta última figura se observa como la célula de sangre roja mantiene la tendencia de migración al centro de la vena recorriendo

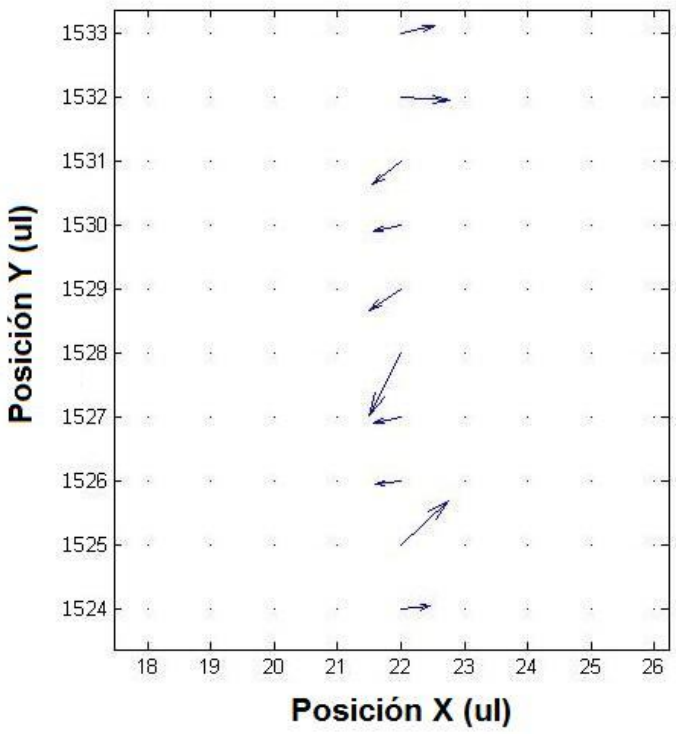


Figura 11e. Detalle del campo vectorial de la fuerza resultante sobre el centro de masa de la partícula.

aproximadamente 5.754 μm entre la posición inicial y la última posición registrada, con lo cual se comprueba el comportamiento de dicha célula de sangre roja en un flujo de Poiseuille anticipado en [49].

Cabe mencionar que, aunque se observa una velocidad más lenta de migración que la observada para el caso similar en [49], ello puede deberse en parte a que se esta considerando una célula rígida no deformable, lo que ralentiza considerablemente dicha velocidad de migración pues según lo observado en el mismo artículo, entre menos deformable es una célula es más lenta la migración de la célula al centro de la vena, y nuestro caso sería el límite de los casos mencionados en el artículo.

En las Figuras 14a a 14c se muestran las variaciones temporales de la posición X para los tres casos de la Prueba 2 añadiendo además las curvas de aproximación para un modelo lineal y un modelo cuadrático, así como las constantes de las ecuaciones de dichos modelos, realizándose esto mediante el programa IBM® SPSS Statistics 19.

Aunque no se nota a simple vista en las figuras mencionadas en el párrafo anterior la inclinación de la curva de ajuste para el modelo lineal es más pronunciada conforme aumenta la viscosidad según muestran los diferentes valores para la constante b1 del modelo lineal (nótese que en realidad es la

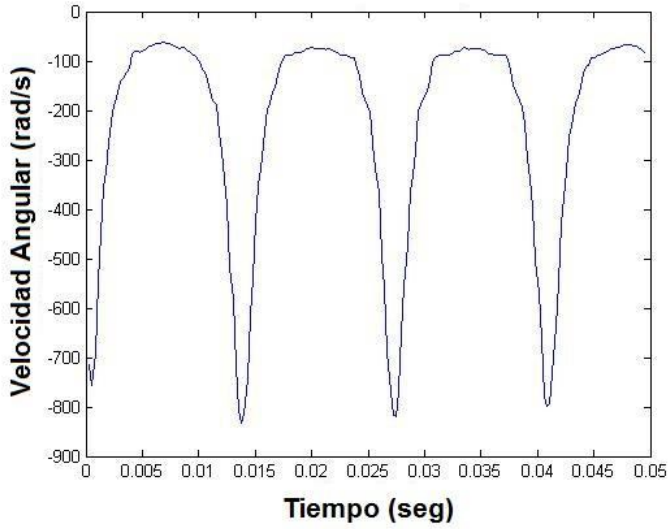


Figura 11f. Velocidad angular vs. Tiempo.

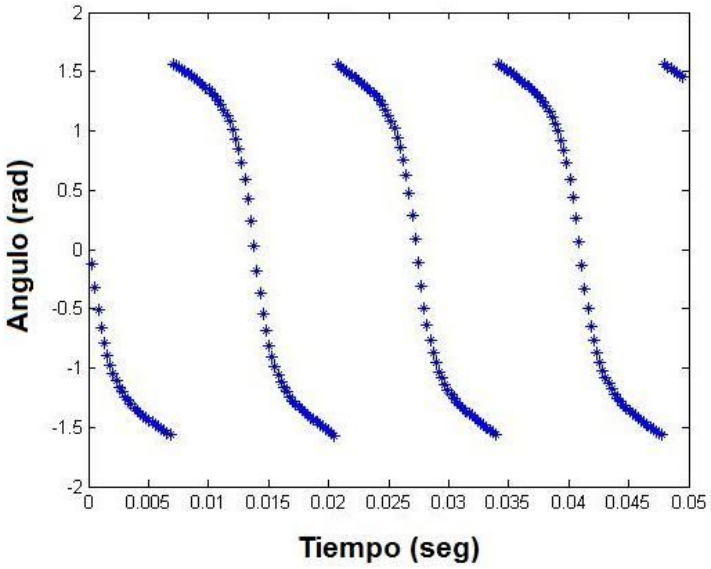


Figura 11g. Angulo de inclinación vs. Tiempo.

tangente de la ecuación de la curva de ajuste, la cual es una línea recta: Posición X = b0 + b1 × Tiempo) y dado que las gráficas se realizaron partiendo de idéntica posición y cubren casi el mismo rango de tiempo se concluye que la migración de la célula de sangre roja es más rápida en un medio más viscoso que en uno menos viscoso, lo cual esta de acuerdo con lo observado para el caso similar en [50].

De todo lo anterior se concluye que, al menos cualitativamente, la implementación de la arquitectura combinada basada en el modelo aquí

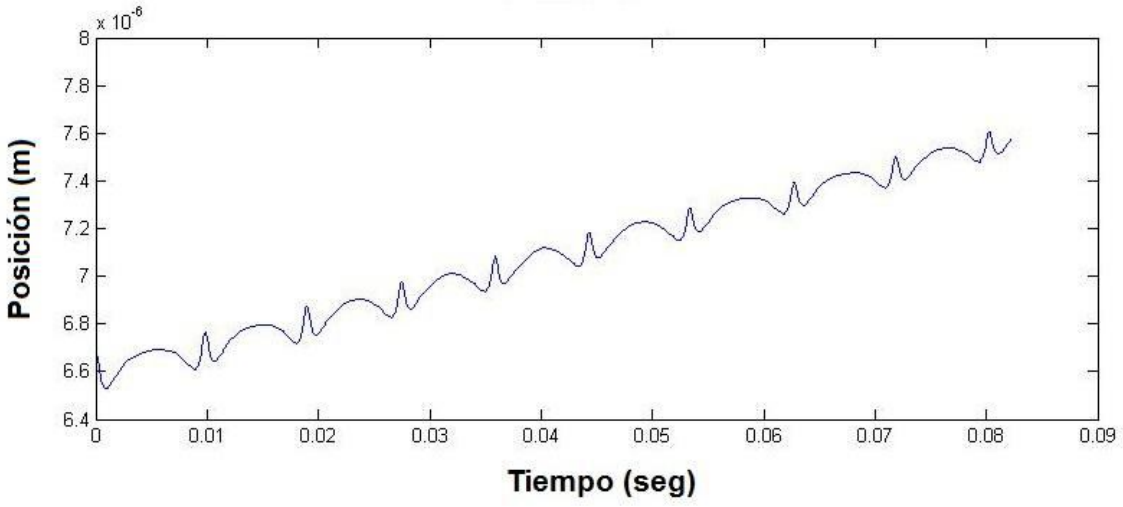


Figura 12. Posición X vs. Tiempo para μ = 2 *cP,* Prueba 2.

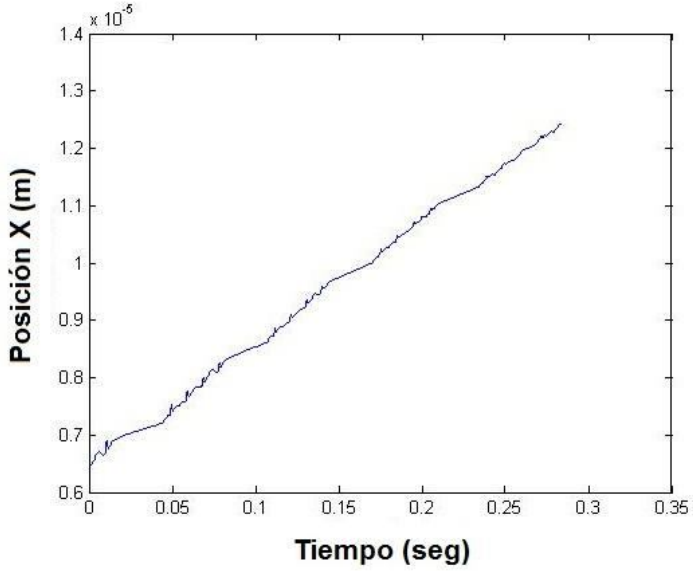
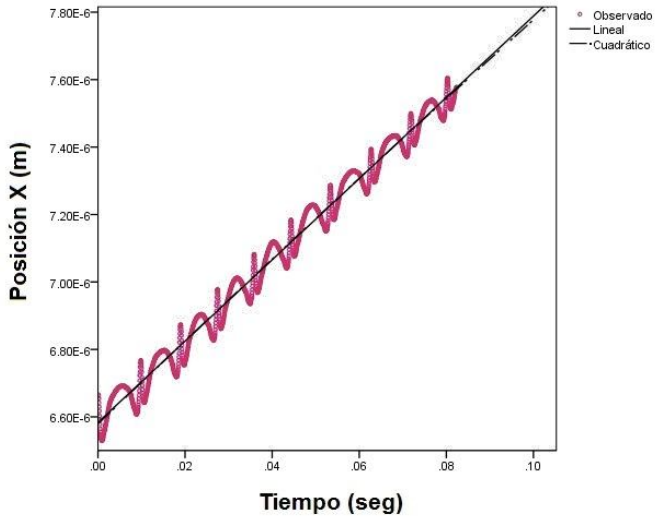


Figura 13. Posición X vs. Tiempo para μ = 2 *cP* y *LY* = 4000 *ul,* Prueba 3.

presentado reproduce aspectos importantes del comportamiento de una célula de sangre roja sujeta a un flujo parabólico o de Poiseuille.

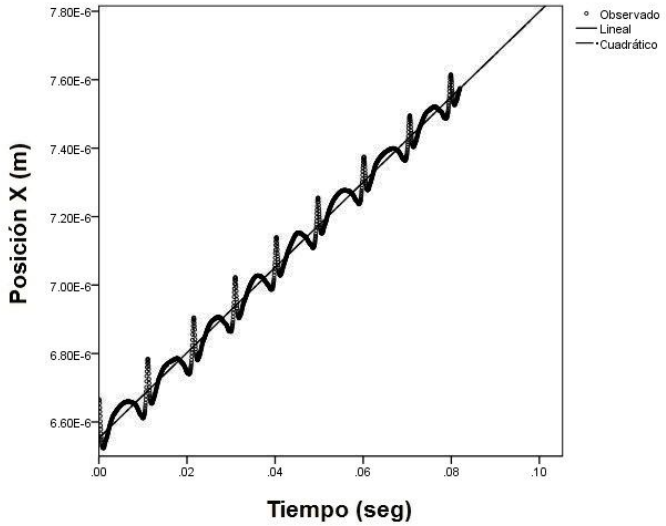
En cuanto a la eficiencia de la paralelización, se menciona que para la Prueba 1, se obtuvo una reducción de 89.35 % en el tiempo de ejecución del ciclo principal de la arquitectura combinada comparado contra la versión del programa secuencial (para un promedio de 198362 iteraciones entre ambos programas, promedio del tiempo de cómputo del ciclo secuencial: 5187.02 *s*, promedio del



Resumen del modelo y estimaciones de los parámetros

Ecuación	Estimaciones de los parámetros		
	Constante	b1	b2
Lineal	6.584E-6	1.205E-5	
Cuadrático	6.579E-6	1.237E-5	-3.907E-6

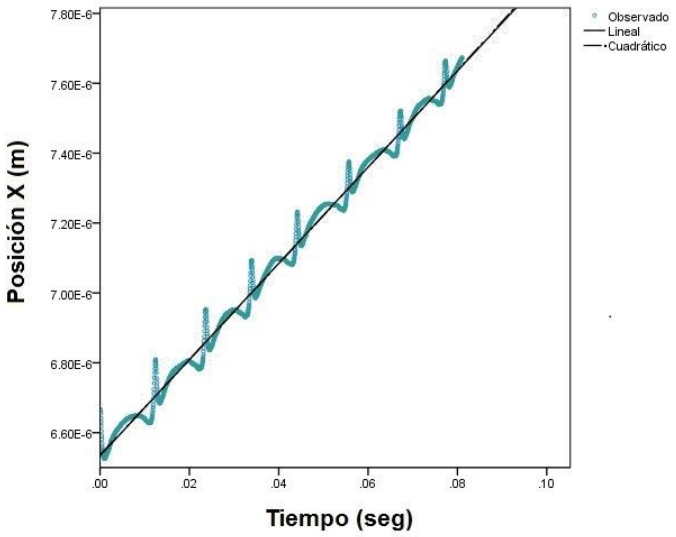
Figura 14a. Posición X vs. Tiempo, curvas de ajuste del modelo lineal y cuadrático y parámetros de las ecuaciones de dichas curvas para μ = 2 *cP*.



Resumen del modelo y estimaciones de los parámetros

Ecuación	Estimaciones de los parámetros		
	Constante	b1	b2
Lineal	6.553E-6	1.246E-5	
Cuadrático	6.553E-6	1.248E-5	-3.033E-7

Figura 14b. Posición X vs. Tiempo, curvas de ajuste del modelo lineal y cuadrático y parámetros de las ecuaciones de dichas curvas para μ = 3 *cP*.



Resumen del modelo y estimaciones de los parámetros

Ecuación	Estimaciones de los parámetros		
	Constante	b1	b2
Lineal	6.534E-6	1.377E-5	
Cuadrático	6.536E-6	1.359E-5	2.119E-6

Figura 14c. Posición X vs. Tiempo, curvas de ajuste del modelo lineal y cuadrático y parámetros de las ecuaciones de dichas curvas para μ = 4 *cP*.

tiempo de cómputo del ciclo paralelizado: 552.31 *s*)[43]. Cabe hacer notar que la reducción en el tiempo de procesamiento para un programa similar en 3D, el cual es presentado en [1] (aunque a diferencia del programa realizado en el presente trabajo, probablemente se procesaron gráficos adicionalmente en el mismo), fue de 95.7 %.

Para efecto de comparar contra otra tecnología de paralelización distinta, se menciona que se realizó en forma paralela al programa de la arquitectura combinada, un programa que paraleliza el algoritmo utilizando sólo directivas de OpenMP, obteniéndose en el mismo para la Prueba 1, una reducción del 65.46 % para 198316 iteraciones y un tiempo promedio de cómputo del ciclo paralelizado de 1791.54 s.

Finalmente se realizó la Prueba 1 utilizando dos nodos MPI asignando a cada uno un GPU distinto (las dos tarjetas mencionadas en el capítulo 3), obteniéndose como resultado de la prueba una reducción de 83.73 % para 198407 iteraciones,

[43] El tiempo de ejecución se midió utilizando la función proporcionada por la implementación de C++ utilizada QueryPerformanceCounter(), que devuelve el número de conteos realizados por el *contador de alta resolución* (tecnología inicialmente desarrollada por Microsoft® e Intel® para frecuencias mínimas de conteos de 10 Mhz, puede consultar especificación (2004) en [W20]) del sistema, la resolución (tiempo entre conteos) varía dependiendo del hardware donde se utilice, para nuestro caso fué de 0.2835 μs aproximadamente.

aunque para este caso no se realizó ningún balance de carga, aparte del ya mencionado, de nodos de la red de Boltzmann a los nodos de MPI del programa original de la arquitectura combinada, con simplemente asignar los correspondientes de cada nodo MPI, a un dispositivo de manera exclusiva mediante la instrucción de CUDA cudaSetDevice(), se consiguió un *incremento comparativo*[44] en la reducción del tiempo de cálculo del ciclo paralelizado de 3 %.

Suponiendo que con cada nueva tarjeta de GPU se agregue otro 3 %, y suponiendo que se utilizaran 7 nodos de MPI asignados a 7 tarjetas de GPU de forma exclusiva, esto significaría un aumento comparativo de 21 % más el incremento que podría añadir el balance de carga, entre otras mejoras (por ejemplo si el balance de carga es dinámico)[45], con lo que se puede dar una idea del potencial de la arquitectura aquí presentada.

[44] Es decir, comparando contra una corrida del programa paralelizado con sólo dos nodos MPI de ejecución.
[45] Aunque esto quizá no signifique mucho a nivel local, si podría ser más apreciado en cómputo distribuido, por ejemplo.

Glosario.

Advección. En un fluido es el transporte de una substancia o propiedad (como la temperatura, el calor, etc.) debido al movimiento de dicho fluido.

Autómatas en Redes Celulares de Gas. Se podría decir que es una máquina abstracta que evoluciona en el tiempo de acuerdo a reglas determinísticas que pueden incluir elementos probabilísticos, donde dicha evolución depende de estados anteriores de un vecindario determinado [5] (consulte también Wolfram (1984) [W21]).

Contexto de Ejecución de un Proceso. Espacio de direcciones que incluye sus variables privadas y su propia pila que puede ser utilizada para llamadas a subrutinas

Diámetro Molecular Efectivo. Se refiere a la distancia existente entre los centros de dos partículas al momento de colisionar **[51]**.

Entropía. En un sistema termodinámico cerrado (en el cual no existe transferencia de masa o energía hacia adentro o hacia fuera del sistema), la entropía es una propiedad de estado tal como la presión, la temperatura, etc., cuya principal característica es la de que si ocurre un proceso irreversible (procesos para los cuales no es posible revertir ni uno solo de sus pasos o etapas) dentro de él, su entropía siempre aumenta nunca disminuye, por ejemplo si ponemos una gota de tinta dentro de un vaso de agua, sus moléculas se esparcirán en todo el vaso pero no podrán espontáneamente volver a unirse para formar la gota de nuevo [52]. Existe también un punto de vista estadístico y microscópico para la definición de la entropía, consultar [52] y [53], a pesar de las asunciones anteriores (sistema cerrado y proceso irreversible), es posible calcular las variaciones de la entropía en un sistema abierto, en estado de no equilibrio y por lo tanto poder aplicar el teorema H.

Flujos estacionarios. Son aquellos en los cuales sus propiedades (masa, viscosidad, temperatura) y variables (velocidad, presión) no cambian con el tiempo.

Flujo laminar. Es aquel en el cual el fluido se mueve en forma de placas paralelas, en este flujo las partículas de fluido siguen trayectorias bien definidas.

Flujo turbulento. En éste tipo de flujo ocurre un comportamiento caótico en el fluido debido a la inercia del mismo, consultar capítulo 11 de [16].

Fuerza de masa. Es una fuerza asociada fuertemente a la masa del cuerpo donde esta actuando, que puede ser por ejemplo la fuerza gravitacional, la fuerza inercial, magnética, etc., estas fuerzas actúan en todo el cuerpo en cuestión **[16]**.

Mecánica Estadística. Rama de la física que trata de explicar las propiedades macroscópicas de los materiales desde un punto de vista microscópico aplicando

para ello la teoría de la probabilidad (para una introducción formal véase el capítulo 4 de [W22]).

Número de Mach. Razón entre la velocidad de un fluido y la velocidad del sonido en ese fluido.

Tratamiento perturbativo. Enfoque matemático que da solución a ecuaciones diferenciales y que puede aplicarse cuando es muy difícil o imposible resolver dichas ecuaciones por métodos tradicionales. Tiene además la ventaja de que en unión con otras herramientas matemáticas puede servir para analizar un mismo problema a varias escalas de las variables independientes.

Trayectoria Libre Media. Si consideramos partículas moviéndose libremente en, por ejemplo un gas, seria la distancia promedio que una partícula recorre entre colisiones sucesivas con otras partículas.

Viscosidad dinámica. Se define como la razón entre el estrés cortante aplicado sobre un fluido y la tasa de deformación que hay entre dos finas capas del mismo y expresa una medida de la resistencia interna de dichas capas a moverse una respecto de la otra, la cual es constante para fluidos newtonianos (tales como el agua, glicerina, plasma sanguíneo, etc.) y es una función de la tasa de tensión cortante para los fluidos no newtonianos, lo cual nos indica la naturaleza del fluido, además puede variar con otras propiedades, por ejemplo la temperatura, finalmente [16] indica que predice el comportamiento del estrés cortante con respecto a la tasa de deformación angular del fluido.

Viscosidad cinemática. Se define como la razón entre la viscosidad dinámica y la densidad de un fluido.

Conclusiones.

Fue posible simular, al menos cualitativamente, el movimiento de una célula de sangre roja en una arteriola utilizando para ello en método de las redes de Boltzmann, el cual se ha mostrado una vez más como una herramienta eficaz para la simulación de sistemas físicos microscópicos y que puede servir como una herramienta más en la investigación biomédica que ayude a los investigadores a combatir los diversos tipos de enfermedades que surgen como consecuencia de anomalías en la microcirculación.

De acuerdo al resultados obtenidos en los cuales se logró una reducción de 89.35 % en el tiempo de ejecución de un programa secuencial previamente construido, cercana a la obtenida por el trabajo similar del autor en el cual se basó el nuestro, se concluye que fue posible paralelizar eficientemente el ciclo principal de cálculo del programa secuencial que realiza la presente simulación, utilizando para ello una arquitectura combinada que utiliza Unidades Gráficas de Proceso y el estándar MPI corriendo dicha arquitectura bajo una plataforma de programación de uso común como es C++ y teniendo como base hardware convencional, como es un CPU de 8 núcleos de AMD® cuyo costo está prácticamente al alcance de cualquier presupuesto mediano para investigación y aún para el bolsillo de casi cualquier profesionista interesado en profundizar el tema, esperándose que si los costos siguen la tendencia a la baja, como se ha observado hasta ahora, se extienda la accesibilidad de los recursos de cada vez más sectores de la sociedad pudiendo con esto aumentar la probable contribución de su parte al avance de temas científicos como el que se ha tratado aquí.

También, gracias a lo anteriormente mencionado, se concluye que el presente trabajo ayudará a difundir más la simulación computacional de este tipo de fenómenos biofísicos entre la comunidad científica atraída por el bajo costo computacional del genial y relativamente sencillo método de las redes de Boltzmann, pero sobre todo por el bajo costo monetario de poder simular algo tan de capital importancia para la continuación del constante combate por mejorar la calidad de vida de la humanidad.

Apéndice A. Código del Ciclo Principal de Programas.

Código del Ciclo Principal del Programa Paralelo.

Arreglos principales:

- **ftemp, f** – Arreglos auxiliares para la determinación de la función de densidad de probabilidad.
- **ns** – Arreglo bidimensional que contiene la matriz de obstáculos.
- **rho** – Arreglo bidimensional que contiene la densidad de masa.

Variables principales (sobre, de o referidas al centro de masa de la partícula):

- **FX0** – Componente X fuerza.
- **FY0** – Componente Y fuerza.
- **Ux** – Componente X velocidad lineal.
- **Uy** – Componente Y velocidad lineal.
- **T** – Torca.
- **O0** – Velocidad angular.
- **X** – Componente X posición.
- **Y** – Componente Y posición.

Otras variables (extremo derecho de la partícula):

- **Xprx** - Componente X posición.
- **Xpry** - Componente Y posición.

```
//Preparación de las estructuras de datos.

CUDA_CALL(cudaMalloc((void**)&MA, size1));        // ftemp

dim3 dimBlock(BLOCK_SIZE, BLOCK_SIZE1);

int TAM_DIMX = 0, TAM_DIMY = 0, TAM_DIMY1 = 0;

// Se determina el número de bloques para la dimensión X.
TAM_DIMX = LX1/dimBlock.x;

// Se determina el número de bloques para la dimensión Y.
TAM_DIMY = floor((double) LY1/(dimBlock.y * nStreams));

if (band = LY1%(dimBlock.y * nStreams) > 0) {
      // Si no es divisible LY1 por dimBlock.y * nStreams, se determinan los renglones
      // restantes de la dimensión Y para dimensionar correctamente el último stream de
      // datos.
      Res = LY1 - dimBlock.y * TAM_DIMY * (nStreams - 1);
      TAM_DIMY1 = floor((double) Res/dimBlock.y);
      if (Res%dimBlock.y > 0)  {
            TAM_DIMY1++;
```

```
	}
}

dim3 dimGrid(TAM_DIMX, TAM_DIMY); // primeros nStreams – 1 o todos los streams.
dim3 dimGrid1(TAM_DIMX, TAM_DIMY1); // Posiblemente el último stream de datos.

// Creación de los Streams.
for (m = 0; m < nStreams; m++)
	CUDA_CALL(cudaStreamCreate(&stream1[m]));

int height_y = TAM_DIMY * dimBlock.y;

desliz1 = height_y * LX1 * VELOCITIES; // Número de elementos en deslizamiento de
                                       // ftemp[].
desliz2 = height_y * LX1; // Número de elementos en deslizamiento de rho[].

tamano1 = desliz1 * sizeof(float);    // Tamaño en bytes de deslizamiento de ftemp[].

if (band)    {     // Si no es divisible LY1 por dimBlock.y * nStreams.

	desliz1_1 = num_ele1 - (nStreams - 1) * desliz1;
	tamano1_1 = desliz1_1 * sizeof(float);
}

// **** Inicia ciclo principal de cálculo. ****

// Nota: el ciclo principal es realizado por todos los nodos de MPI pero los ciclos
// de trabajo internos son solo sobre la altura de su franja de datos correspondiente
// (height + 2).

CUDA_CALL(cudaDeviceSynchronize()); // Espera a que finalice cualquier llamada CUDA
                                    // realizada.
StartCounter(&PCFreq, &CounterStart); // Se inicia el timer del CPU.

while (t > 0)            {

	// *** llama al kernel de CUDA. ****

	for (m = 0; m < nStreams; m++)  {
		if (m < nStreams - 1)    {     // Primeros (nStreams - 1) streams de
		                               // cómputo.

			// Se copian los datos de las estructuras de entrada.
			CUDA_CALL(cudaMemcpyAsync(MA + m*desliz1, ftemp + m*desliz1,
			tamano1, cudaMemcpyHostToDevice, stream1[m]) );

			MatMulKernel<<<dimGrid, dimBlock, 0, stream1[m]>>>(f_c +
			m*desliz1, MA + m*desliz1, rho_c + m*desliz2, LX1);

		} else{     // Ultimo stream de cómputo.

			if (band)    {     // No es divisible LY1 por dimBlock.y *
			                   // nStreams.

				CUDA_CALL(cudaMemcpyAsync(MA + m*desliz1, ftemp + m*desliz1,
				tamano1_1, cudaMemcpyHostToDevice, stream1[m]) );

				MatMulKernel<<<dimGrid1, dimBlock, 0, stream1[m]>>>(f_c +
				m*desliz1, MA + m*desliz1, rho_c + m*desliz2, LX1);
			} else{           // No hubo cambio en último stream de datos.

				CUDA_CALL(cudaMemcpyAsync(MA + m*desliz1, ftemp + m*desliz1,
				tamano1, cudaMemcpyHostToDevice, stream1[m]) );
```

```
                    MatMulKernel<<<dimGrid, dimBlock, 0, stream1[m]>>>(f_c +
                    m*desliz1, MA + m*desliz1, rho_c + m*desliz2, LX1);
              }
        }
}

CUDA_CALL(cudaDeviceSynchronize());// Espera a que finalizen todos los streams
                                   // en ejecución.

// **** Inicia cálculo del paso de propagación. ****

FX = FY = 0.0; T = 0.0;
for (j = 1; j < height + 1; j++)      {

        // Ordenadas "abajo" y "arriba" tomando como referencia la matriz de
        // obstáculos original.
        jn = (j > 0)?(j-1):(LY-1);
        jp = (j < LY-1)?(j+1):(0);

        // Ordenadas "abajo" y "arriba" tomando como referencia la franja de
        // nodos particular.
        j1 = np + j-1;
        jn1 = (j1 > 0)?(j1-1):(LY-1);
        jp1 = (j1 < LY-1)?(j1+1):(0);

        for (i = 0; i < LX; i++) {

              in = (i > 0)?(i-1):(LX-1);
              ip = (i < LX-1)?(i+1):(0);

              if (!(ns[j1 * LX + i] == 2))    {

                    // Se propagan las distribuciones sin importar el tipo de
                    // nodo destino.
                    if (ns[j1 * LX + i] == 0){

                          ftemp[j*LX1*VELOCITIES + i] = f[j*LX1*VELOCITIES + i];
                          ftemp[j*LX1*VELOCITIES + LX1 * 1 + ip] =
                          f[j*LX1*VELOCITIES + LX1 * 1 + i];
                          ftemp[jp*LX1*VELOCITIES + LX1 * 5 + ip] =
                          f[j*LX1*VELOCITIES + LX1 * 5 + i];
                          ftemp[jp*LX1*VELOCITIES + LX1 * 2 + i] =
                          f[j*LX1*VELOCITIES + LX1 * 2 + i];
                          ftemp[jp*LX1*VELOCITIES + LX1 * 6 + in] =
                          f[j*LX1*VELOCITIES + LX1 * 6 + i];
                          ftemp[j*LX1*VELOCITIES + LX1 * 3 + in] =
                          f[j*LX1*VELOCITIES + LX1 * 3 + i];
                          ftemp[jn*LX1*VELOCITIES + LX1 * 7 + in] =
                          f[j*LX1*VELOCITIES + LX1 * 7 + i];
                          ftemp[jn*LX1*VELOCITIES + LX1 * 4 + i]
                          =f[j*LX1*VELOCITIES + LX1 * 4 + i];
                          ftemp[jn*LX1*VELOCITIES + LX1 * 8 + ip] =
                          f[j*LX1*VELOCITIES + LX1 * 8 + i];

                    }
              }
        }
}


// **** Envío y recepción de límite superior de las franjas de datos - versión
// bloqueante 2. ****

// Envío límite superior de las franjas de datos.
```

```
if (mynode == MASTER)    {

      proc =  nodo1;
      MPI_CALL (MPI_Ssend(&ftemp[(height + 1)*LX*VELOCITIES], LX*VELOCITIES,
        MPI_DOUBLE, proc, 2, MPI_COMM_WORLD));

      proc = totalnodes - 1;
      MPI_CALL (MPI_Recv(bufftmp, LX*VELOCITIES, MPI_DOUBLE, proc, 2,
        MPI_COMM_WORLD, &status));

} else if (mynode == nodo1) {

      proc =  MASTER;
      MPI_CALL (MPI_Recv(bufftmp, LX*VELOCITIES, MPI_DOUBLE, proc, 2,
        MPI_COMM_WORLD, &status));

      if (totalnodes >= 3)     {
            proc =  nodo2;
            MPI_CALL (MPI_Ssend(&ftemp[(height + 1)*LX*VELOCITIES],
              LX*VELOCITIES, MPI_DOUBLE, proc, 2, MPI_COMM_WORLD));
      } else {
            proc =  MASTER;
            MPI_CALL (MPI_Ssend(&ftemp[(height + 1)*LX*VELOCITIES],
              LX*VELOCITIES, MPI_DOUBLE, proc, 2, MPI_COMM_WORLD));
      }

} else if (mynode == nodo2) {

      proc =  nodo1;
      MPI_CALL (MPI_Recv(bufftmp, LX*VELOCITIES, MPI_DOUBLE, proc, 2,
        MPI_COMM_WORLD, &status));

      if (totalnodes >= 4)     {
            proc =  nodo3;
            MPI_CALL (MPI_Ssend(&ftemp[(height + 1)*LX*VELOCITIES],
              LX*VELOCITIES, MPI_DOUBLE, proc, 2, MPI_COMM_WORLD));
      } else {
            proc =  MASTER;
            MPI_CALL (MPI_Ssend(&ftemp[(height + 1)*LX*VELOCITIES],
              LX*VELOCITIES, MPI_DOUBLE, proc, 2, MPI_COMM_WORLD));
      }

} else if (mynode == nodo3) {
.
.
.

} else if (mynode == nodo7) {

      proc =  nodo6;
      MPI_CALL (MPI_Recv(bufftmp, LX*VELOCITIES, MPI_DOUBLE, proc, 2,
        MPI_COMM_WORLD, &status));

      proc =  MASTER;
      MPI_CALL (MPI_Ssend(&ftemp[(height + 1)*LX*VELOCITIES], LX*VELOCITIES,
        MPI_DOUBLE, proc, 2, MPI_COMM_WORLD));

}

// < En esta parte quizá se podría incluir un Barrier de sincronización de MPI para
// evitar hasta el mínimo errores en la comunicación... >


// Se transfiere el buffer de recepción al extremo inferior correspondiente en
// ftemp.
```

```
TransferirArreglos(&ftemp[LX*VELOCITIES], bufftmp);
memset(bufftmp, 0.0, LX1*VELOCITIES*sizeof(float));

// Envío y recepción del límite inferior de las franjas de datos.

if (mynode == MASTER)    {

      proc =  totalnodes-1;
      MPI_CALL (MPI_Ssend(&ftemp[0], LX*VELOCITIES, MPI_DOUBLE, proc, 1,
        MPI_COMM_WORLD));

      proc = nodo1;
      MPI_CALL (MPI_Recv(bufftmp, LX*VELOCITIES, MPI_DOUBLE, proc, 1,
        MPI_COMM_WORLD, &status));

} else if (mynode == nodo1)    {

      if (totalnodes >= 3)     {
            proc = nodo2;
            MPI_CALL (MPI_Recv(bufftmp, LX*VELOCITIES, MPI_DOUBLE, proc, 1,
              MPI_COMM_WORLD, &status));
      } else{
            proc = MASTER;
            MPI_CALL (MPI_Recv(bufftmp, LX*VELOCITIES, MPI_DOUBLE, proc, 1,
              MPI_COMM_WORLD, &status));
      }

      proc =  MASTER;
      MPI_CALL (MPI_Ssend(&ftemp[0], LX*VELOCITIES, MPI_DOUBLE, proc, 1,
        MPI_COMM_WORLD));

} else if (mynode == nodo2)    {

 .
 .
 .

} if (mynode == nodo7)   {

      proc = MASTER;
      MPI_CALL (MPI_Recv(bufftmp, LX*VELOCITIES, MPI_DOUBLE, proc, 1,
        MPI_COMM_WORLD, &status));

      proc =  nodo6;
      MPI_CALL (MPI_Ssend(&ftemp[0], LX*VELOCITIES, MPI_DOUBLE, proc, 1,
        MPI_COMM_WORLD));

}

// Se transfiere el buffer de recepción al extremo superior correspondiente en
// ftemp.
TransferirArreglos1(&ftemp[height*LX*VELOCITIES], bufftmp);
memset(bufftmp, 0.0, LX1*VELOCITIES*sizeof(float));


// **** Se realiza bounce back de medio enlace, se añade el efecto de la
// transferencia del momento al fluido y se calculan las componentes de la
// fuerza y la torca ejercidas sobre la partícula. ****

for (j = 0; j < height + 2; j++)      {

      jn = (j > 0)?(j-1):(LY-1);
      jp = (j < LY-1)?(j+1):(0);

      j1 = np + j-1;
```

```
if (np == 0 && j == 0)   {     // Corrección  si se encuentra en
                               // el renglón cero del nodo 0.
      j1 = LY-1;
}
else if (j1 == LY) {           // Corrección  si se encuentra en
                               // el último renglón del último
                               // nodo.
      j1 = 0;
}
jn1 = (j1 > 0)?(j1-1):(LY-1);
jp1 = (j1 < LY-1)?(j1+1):(0);

for (i = 0; i < LX; i++) {

      in = (i > 0)?(i-1):(LX-1);
      ip = (i < LX-1)?(i+1):(0);

      if ((ns[j1 * LX + i] == 0) && (j != height + 1) && (j != 0))   {

            if (ns[j1 * LX + ip] == 2)     {
                  ubx = Ux - O*(j1 - ns1[j1]);
                  ftemp[j*LX1*VELOCITIES + LX1 * 3 + i] =
                  ftemp[j*LX1*VELOCITIES + LX1 * 1 + ip] -
                  2*rt1*rho[j*LX1 + i]*f1*ubx;
                  Fx = 2*(ftemp[j*LX1*VELOCITIES + LX1 * 1 + ip] -
                  rt1*rho[j*LX1 + i]*f1*ubx);
                  FX += Fx;
                  T += -(j1 - ns1[j1])*Fx;
            }

                  .
                  .
                  .

            if (ns[jn1 * LX1 + ip] == 2) {
                  ubx = Ux - O*(jn1 + 0.5 - ns1[jn1]);
                  uby = Uy + O*(i + 0.5 - X);
                  ftemp[j*LX1*VELOCITIES + LX1 * 6 + i] =
                  ftemp[jn*LX1*VELOCITIES + LX1 * 8 + ip] +
                  2*rt2*rho[j*LX1 + i]*f1*(-ubx + uby);
                  Fx = 2*(ftemp[jn*LX1*VELOCITIES + LX1 * 8 + ip] -
                  rt2*rho[j*LX1 + i]*f1*(ubx - uby));
                  Fy = -2*(ftemp[jn*LX1*VELOCITIES + LX1 * 8 + ip] -
                  rt2*rho[j*LX1 + i]*f1*(ubx - uby));
                  FX += Fx; FY += Fy;
                  T += (i + 0.5 - X)*Fy-(jn1 + 0.5 - ns1[jn1])*Fx;
            }

      } else if (ns[j1 * LX + i] == 1)     {

            // Falta considerar el intercambio de velocidades 2 y 4
            // (considerar para versiones posteriores).
            if ((ns[j1 * LX1 + ip] == 0) && (j != height + 1) && (j != 0)) {
                  ftemp[j*LX1*VELOCITIES + LX1 * 1 + ip] =
                  ftemp[j*LX1*VELOCITIES + LX1 * 3 + i];
            }
            if (ns[jp1 * LX1 + ip] == 0 && (j != height + 1)) {
                  ftemp[jp*LX1*VELOCITIES + LX1 * 5 + ip] =
                  ftemp[j*LX1*VELOCITIES + LX1 * 7 + i];
            }
            if (ns[jn1 * LX1 + ip] == 0 && (j != 0))    {
                  ftemp[jn*LX1*VELOCITIES + LX1 * 8 + ip] =
                  ftemp[j*LX1*VELOCITIES + LX1 * 6 + i];
            }
```

```
                    if ((ns[j1 * LX1 + in] == 0) && (j != height + 1) && (j != 0)) {
                          ftemp[j*LX1*VELOCITIES + LX1 * 3 + in] =
                          ftemp[j*LX1*VELOCITIES + LX1 * 1 + i];
                    }
                    if (ns[jp1 * LX1 + in] == 0 && (j != height + 1)) {
                          ftemp[jp*LX1*VELOCITIES + LX1 * 6 + in] =
                          ftemp[j*LX1*VELOCITIES + LX1 * 8 + i];
                    }
                    if (ns[jn1 * LX1 + in] == 0 && (j != 0))    {
                          ftemp[jn*LX1*VELOCITIES + LX1 * 7 + in] =
                          ftemp[j*LX1*VELOCITIES + LX1 * 5 + i];
                    }
              }
        }
}


buffp1s[0] = FX;
buffp1s[1] = FY;
buffp1s[2] = T;

// Se reducen los valores de FX, FY, T, dbnpr, Uprx y Upry
MPI_CALL (MPI_Reduce(buffp1s, buffp1r, NUM_P1, MPI_DOUBLE, MPI_SUM, MASTER,
 MPI_COMM_WORLD));

// Se envían los datos reducidos a todos los nodos
MPI_CALL (MPI_Bcast(buffp1r, NUM_P1, MPI_DOUBLE, MASTER, MPI_COMM_WORLD));

FX = buffp1r[0];
FY = buffp1r[1];
T = buffp1r[2];

// **** Comienza cálculo de movimiento de la partícula. ****

FX12 = FX; FY12 = FY;
FX0 = (FX112 + FX12)/2.;
FY0 = (FY112 + FY12)/2.;

Ux = Uxold + (FX0/M2)*tpo;
Uy = Uyold + (FY0/M2)*tpo;

T12 = T;
T0 = (T12 + T112)/2.;

O = Oold + (T0/I)*tpo;

X = Xold + Uxold*tpo;
Y = Yold + Uyold*tpo;

// Se actualiza la posición del extremo derecho de la partícula.
phi = phi_old + Oold*tpo;

// Se comprueba si el extremo derecho ha excedido el primer cuadrante respecto
// al centro de masa (o en su defecto el ángulo fi, el cual es el ángulo
// asociado a dicho extremo).

if (phi < -PI25DT/2.)    {

      phi = phi + PI25DT;

} else if (phi > PI25DT/2.) {

      phi = phi – PI25DT;
}
```

```
Xprx = a2*cos(phi);
Xpry = a2*sin(phi);


// **** Se substituyen los nodos de la partícula por nodos de fluido. ****

// Se determina cuales nodos participan en la actualización.
Nodost = Nodos(lim_sup, lim_inf, totalnodes, height);

// Se realiza la substitución.
if (BuscaNodo(Nodost, mynode, totalnodes))  {

      for (j = 0; j < l; j++)  {

            ueqxij = sns[j].ux;
            ueqyij = sns[j].uy;

            uxsq = ueqxij * ueqxij;
            uysq = ueqyij * ueqyij;

            uxuy5 = ueqxij + ueqyij;
            uxuy6 = -ueqxij + ueqyij;
            uxuy7 = -ueqxij - ueqyij;
            uxuy8 = ueqxij - ueqyij;

            usq = uxsq + uysq;

            node = floor((double) (sns[j].j / height));
            if (node > (totalnodes - 1))          {
                  node = totalnodes - 1;
            }
            jf = sns[j].j - np + 1;
            if (mynode == node)      {
                  ftemp[jf*LX1*VELOCITIES + sns[j].i] = rt0 * (1. - f3 * usq);
            }
            if (mynode == node)      {
                  ftemp[jf*LX1*VELOCITIES + LX1 * 1 + sns[j].i] = rt1 * (1. + f1 *
                  ueqxij + f2 * uxsq - f3 * usq);
            }
            if (mynode == node)      {
                  ftemp[jf*LX1*VELOCITIES + LX1 * 2 + sns[j].i] = rt1 * (1. + f1 *
                  ueqyij + f2 * uysq - f3 * usq);
            }
            if (mynode == node)      {
                  ftemp[jf*LX1*VELOCITIES + LX1 * 3 + sns[j].i] = rt1 * (1. - f1 *
                  ueqxij + f2 * uxsq - f3 * usq);
            }
            if (mynode == node)      {
                  ftemp[jf*LX1*VELOCITIES + LX1 * 4 + sns[j].i] = rt1 * (1. - f1 *
                  ueqyij + f2 * uysq - f3 * usq);
            }
            if (mynode == node)      {
                  ftemp[jf*LX1*VELOCITIES + LX1 * 5 + sns[j].i] = rt2 * (1. + f1 *
                  uxuy5 + f2 * uxuy5 * uxuy5 - f3 * usq);
            }
            if (mynode == node)      {
                  ftemp[jf*LX1*VELOCITIES + LX1 * 6 + sns[j].i] = rt2 * (1. + f1 *
                  uxuy6 + f2 * uxuy6 * uxuy6 - f3 * usq);
            }
            if (mynode == node)      {
                  ftemp[jf*LX1*VELOCITIES + LX1 * 7 + sns[j].i] = rt2 * (1. + f1 *
                  uxuy7 + f2 * uxuy7 * uxuy7 - f3 * usq);
            }
            if (mynode == node)      {
```

```
                        ftemp[jf*LX1*VELOCITIES + LX1 * 8 + sns[j].i] = rt2 * (1. + f1 *
                        uxuy8 + f2 * uxuy8 * uxuy8 - f3 * usq);
                }
        }
}
free(Nodost);

// **** Actualización de la matriz de obstáculos. ****

// Se actualiza la matriz de nodos partícula en base a la nueva posición de la
// misma.
Particula = Nodos_particula(Xprx, Xpry, X, Y, a2, c2);

// Se comprueba si la partícula ha alcanzado el principio o el final de la
// vena, y en caso de ser así se sale del ciclo principal.
if ((lim_sup1 > LY - 1) || (lim_inf1 < 0))
        break;

// Se inicializa la matriz de obstáculos.
for (j = 0; j < l; j++)  {
        ns[sns[j].j * LX1 + sns[j].i] = 0; // Fluido.
        ns1[sns[j].j] = 0;
}

// Se inicializa la matriz de obstáculos con los nodos de partícula
// determinados anteriormente así como otras estructuras de control y se
// determina la velocidad de los nodos internos de la partícula.
l = 0;
while (Particula != NULL){
        dato = sacar(Particula);
        if (dato == NULL) throw "Excepción en función sacar().";
        ns[dato->dato0 * LX + dato->dato1] = 2;
        ns1[dato->dato0] = dato->dato2;
        sns[l].j = dato->dato0;
        sns[l].i = dato->dato1;

        // Se calcula la velocidad de los nodos internos de la partícula
        sns[l].ux = Ux - O*(dato->dato0 - dato->dato2);
        sns[l].uy = Uy + O*(dato->dato1 - X);
        free(dato);
        l++;
}

// Se almacenan los Resultados para graficar solo por el nodo maestro.
if (mynode == MASTER)    {

        if ((Y - floor(Y)) < 0.5){
                j = floor(Y);
        } else{
                j = ceil(Y);
        }
        if ((X - floor(X)) < 0.5){
                i = floor(X);
        } else{
                i = ceil(X);
        }

        t1 += delta_t;
        Arr_Res[j * LX + i].t = t1;
        Arr_Res[j * LX + i].Ux = Ux*delta_v;
        Arr_Res[j * LX + i].Uy = Uy*delta_v;
        Arr_Res[j * LX + i].X = X*delta_x;
        Arr_Res[j * LX + i].Y = Y*delta_x;
        Arr_Res[j * LX + i].FX = FX0*delta_f;
        Arr_Res[j * LX + i].FY = FY0*delta_f;
```

```
            Arr_Res[j * LX + i].T = T0*delta_t;
            Arr_Res[j * LX + i].O = O/delta_t;
            Arr_Res[j * LX + i].Ang = phi;
      }

      FX112 = FX12; FY112 = FY12;
      T112 = T12;

      Uxold = Ux; Uyold = Uy;
      Oold = O;

      Xold = X; Yold = Y;

      phi_old = phi;


      t--;
}

t_simul = GetCounter(&PCFreq, &CounterStart); // Se finaliza el timer del CPU.
```

Código del kernel de CUDA.

Arreglos principales:

- **A0, A** – Arreglos auxiliares para la determinación de la función de densidad de probabilidad.
- **A1** – Arreglo que contiene la matriz de obstáculos.
- **B, C** – Arreglos auxiliares para la determinación de las componentes X e Y de la fuerza respectivamente.
- **D** – Arreglo auxiliar para la determinación de la densidad local.

```
// ** Kernel de CUDA para la paralelización del método de redes de Boltzmann hasta el
// paso de colisión. **

// Algunos parámetros de la simulación.
__constant__ float  g = G;
__constant__ float  tau = TAU;
__constant__ int ex[] = {0, 1, 0, -1, 0, 1, -1, -1, 1}; __constant__ int ey[]  = {0,
                  0, 1, 0, -1, 1, 1, -1, -1};

__constant__ float f1 = 3.0f;
__constant__ float f2 = 9.0f/2.0f;
__constant__ float f3 = 3.0f/2.0f;

__constant__ float rt0c = 4.0f/9.0f;
__constant__ float rt1c = 1.0f/9.0f;
__constant__ float rt2c = 1.0f/36.0f;

__global__ void MatMulKernel(float *A0c, const float *Ac, float *Dc, const int LX1) {

      // Definición de variables locales y constantes de cada hilo.

      float Bvalue = 0.0f, Cvalue = 0.0f, Dvalue = 0.0f;
      float Avalue_0 = 0.0f, Avalue_1 = 0.0f, Avalue_2 = 0.0f, Avalue_3 = 0.0f, Avalue_4
            = 0.0f,
            Avalue_5 = 0.0f, Avalue_6 = 0.0f, Avalue_7 = 0.0f, Avalue_8 = 0.0f;
      float feq = 0.0f;

      // Renglón y columna correspondiente de cada hilo.
      int row = blockIdx.y * blockDim.y + threadIdx.y; // j
      int col = blockIdx.x * blockDim.x + threadIdx.x; // i

      // Inicialización de variables.
      float rt0 = 0.0f, rt1 = 0.0f, rt2 = 0.0f;
      float uxsq = 0.0f, uysq = 0.0f, uxuy5 = 0.0f, uxuy6 = 0.0f, uxuy7 = 0.0f, uxuy8 =
      0.0f, usq = 0.0f;

      // ** Inicia cálculo. **


      // Computando la densidad macroscópica, rho, y la velocidad u = (ux, uy).

      Avalue_0 = Ac[row * LX1 * VELOCITIES + col];
      Bvalue = ex[0] * Avalue_0;
      Cvalue = ey[0] * Avalue_0;
```

```
Dvalue = Avalue_0;
Avalue_1 = Ac[row * LX1 * VELOCITIES + LX1 * 1 + col];
Bvalue = Bvalue + ex[1] * Avalue_1;
Cvalue = Cvalue + ey[1] * Avalue_1;
Dvalue = Dvalue + Avalue_1;
Avalue_2 = Ac[row * LX1 * VELOCITIES + LX1 * 2 + col];
Bvalue = Bvalue + ex[2] * Avalue_2;
Cvalue = Cvalue + ey[2] * Avalue_2;
Dvalue = Dvalue + Avalue_2;
Avalue_3 = Ac[row * LX1 * VELOCITIES + LX1 * 3 + col];
Bvalue = Bvalue + ex[3] * Avalue_3;
Cvalue = Cvalue + ey[3] * Avalue_3;
Dvalue = Dvalue + Avalue_3;
Avalue_4 = Ac[row * LX1 * VELOCITIES + LX1 * 4 + col];
Bvalue = Bvalue + ex[4] * Avalue_4;
Cvalue = Cvalue + ey[4] * Avalue_4;
Dvalue = Dvalue + Avalue_4;
Avalue_5 = Ac[row * LX1 * VELOCITIES + LX1 * 5 + col];
Bvalue = Bvalue + ex[5] * Avalue_5;
Cvalue = Cvalue + ey[5] * Avalue_5;
Dvalue = Dvalue + Avalue_5;
Avalue_6 = Ac[row * LX1 * VELOCITIES + LX1 * 6 + col];
Bvalue = Bvalue + ex[6] * Avalue_6;
Cvalue = Cvalue + ey[6] * Avalue_6;
Dvalue = Dvalue + Avalue_6;
Avalue_7 = Ac[row * LX1 * VELOCITIES + LX1 * 7 + col];
Bvalue = Bvalue + ex[7] * Avalue_7;
Cvalue = Cvalue + ey[7] * Avalue_7;
Dvalue = Dvalue + Avalue_7;
Avalue_8 = Ac[row * LX1 * VELOCITIES + LX1 * 8 + col];
Bvalue = Bvalue + ex[8] * Avalue_8;
Cvalue = Cvalue + ey[8] * Avalue_8;
Dvalue = Dvalue + Avalue_8;
Bvalue = Bvalue/Dvalue;
Cvalue = Cvalue/Dvalue;

// Computar la función de distribución de equilibrio, feq.

rt0 = rt0c * Dvalue;
rt1 = rt1c * Dvalue;
rt2 = rt2c * Dvalue;

uxsq = Bvalue * Bvalue;
uysq = Cvalue * Cvalue;

uxuy5 = Bvalue + Cvalue;
uxuy6 = -Bvalue + Cvalue;
uxuy7 = -Bvalue - Cvalue;
uxuy8 = Bvalue - Cvalue;

usq = uxsq + uysq;

// Se calcula el paso de colisión.
feq = rt0 * (1. - f3 * usq);
A0c[row * LX1 * VELOCITIES + col] = Avalue_0 - (Avalue_0 - feq)/tau;;
feq = rt1 * (1. + f1 * Bvalue + f2 * uxsq - f3 * usq);
A0c[row * LX1 * VELOCITIES + LX1 * 1 + col] = Avalue_1 - (Avalue_1 - feq)/tau;
feq = rt1 * (1. + f1 * Cvalue + f2 * uysq - f3 * usq);
A0c[row * LX1 * VELOCITIES + LX1 * 2 + col] = Avalue_2 - (Avalue_2 - feq)/tau -
g/CALPHA;
feq = rt1 * (1. - f1 * Bvalue + f2 * uxsq - f3 * usq);
A0c[row * LX1 * VELOCITIES + LX1 * 3 + col] = Avalue_3 - (Avalue_3 - feq)/tau;
feq = rt1 * (1. - f1 * Cvalue + f2 * uysq - f3 * usq);
A0c[row * LX1 * VELOCITIES + LX1 * 4 + col] = Avalue_4 - (Avalue_4 - feq)/tau +
g/CALPHA;
```

```
        feq = rt2 * (1. + f1 * uxuy5 + f2 * uxuy5 * uxuy5 - f3 * usq);
        A0c[row * LX1 * VELOCITIES + LX1 * 5 + col] = Avalue_5 - (Avalue_5 - feq)/tau -
        g/CALPHA;
        feq = rt2 * (1. + f1 * uxuy6 + f2 * uxuy6 * uxuy6 - f3 * usq);
        A0c[row * LX1 * VELOCITIES + LX1 * 6 + col] = Avalue_6 - (Avalue_6 - feq)/tau -
        g/CALPHA;
        feq = rt2 * (1. + f1 * uxuy7 + f2 * uxuy7 * uxuy7 - f3 * usq);
        A0c[row * LX1 * VELOCITIES + LX1 * 7 + col] = Avalue_7 - (Avalue_7 - feq)/tau +
        g/CALPHA;
        feq = rt2 * (1. + f1 * uxuy8 + f2 * uxuy8 * uxuy8 - f3 * usq);
        A0c[row * LX1 * VELOCITIES + LX1 * 8 + col] = Avalue_8 - (Avalue_8 - feq)/tau +
        g/CALPHA;


        // Se cargan resultados en la memoria global.
        Dc[row * LX1 + col] = Dvalue;   // rho
}
```

Apéndice B. Especificaciones del Hardware.

Especificaciones del servidor.

- Procesador AMD® FX-8150, 8 procesadores a 3.61 GHz, 64 bits.
- 8 GB de RAM.
- Tarjeta Madre: Crosshair V Formula, Chipset AMD® 990 FX/SB950.
- Bus del Sistema de hasta 5200 MT/s Hypertransport™ 3.0.
- Sistema Operativo: Microsoft® Windows® 7 Profesional, SP 1, 64 bits.

Especificaciones tarjeta gráfica GeForce® GTX480.

- 480 Núcleos de CUDA™, tecnología Fermi.
- Compute Capability: 2.0.
- Versión del Driver de CUDA: 5.0, Versión de Tiempo de Ejecución de CUDA: 5.0.
- 15 Multiprocesadores x 32 Núcleos de CUDA/Multiprocesador.
- Total de Memoria Constante: 65536 bytes.
- Total de Memoria Compartida por Bloque: 49152 bytes.
- Número Total de Registros disponibles por Bloque: 32768.
- Tamaño del Warp: 32.
- Es posible la ejecución y copia concurrentes disponiendo de 1 motor de copia.
- Es posible la ejecución concurrente de Kernels.
- Se soporta el mapeo de Memoria no Paginable con el Host.
- Planificador dual de warps.
- Máximo Número de Bloques por Multiprocesador: 8.
- Máximo Número de Hilos por Multiprocesador: 1536.
- Medidas máximas de cada dimensión de un Bloque: 1024 x 1024 x 64.
- Medidas máximas de cada dimensión de un Grid: 65535 x 65535 x 65535.
- Reloj del GPU: 1401 MHz.
- Tasa de Relleno de Texturas: 42 Mil millones/s
- Reloj de la Memoria: 1848 MHz.
- Total de Memoria de Video (Global): 1536 MB GDDR5.
- Ancho de la Interfaz de la Memoria: 384 bits.
- Ancho de Banda de la Memoria: 177.4 GB/s.
- Bus Soportado: PCI-Express® 2.0 x 16.
- Versión de OpenGL® soportado: 4.2.
- Otras Tecnologías Soportadas: DirectX® 11, 3D Vision®, 3D Vision Surround®, PhysX™, CUDA™, SLI®.
- Longitud: 10.5 pulgadas (267 mm) x Altura: 4.376 pulgadas (111 mm).
- Anchura: ranura sencilla.
- Temperatura Máxima de la GPU: 105 °C.
- Potencia mínima requerida del sistema: 600 W.

Especificaciones tarjeta gráfica GeForce® GTX660Ti

- 1344 Núcleos de CUDA™, tecnología Kepler.
- Compute capability: 3.0.
- Versión del Driver de CUDA: 5.0, Versión de Tiempo de Ejecución de CUDA: 5.0.
- 7 Multiprocesadores x 192 Núcleos de CUDA/Multiprocesador.
- Total de Memoria Constante: 65536 bytes.
- Total de Memoria Compartida por Bloque: 49152 bytes.
- Número Total de Registros disponibles por Bloque: 65536.
- Tamaño del Warp: 32.
- Es posible la ejecución y copia concurrentes disponiendo de 1 motor de copia.
- Es posible la ejecución concurrente de Kernels.
- Se soporta el mapeo de Memoria no Paginable con el Host.
- Máximo Número de Bloques por Multiprocesador: 16.
- Máximo Número de Hilos por Multiprocesador: 2048.
- Medidas máximas de cada dimensión de un Bloque: 1024 x 1024 x 64.
- Medidas máximas de cada dimensión de un Grid: 2147483647 x 65535 x 65535.
- Reloj del GPU: 1.06 GHz.
- Reloj normal: 915 MHz.
- Reloj acelerado: 980 MHz.
- Tasa de Relleno de Texturas: 102.5 Mil millones/s.
- Reloj de la Memoria: 3004 MHz.
- Total de Memoria de Video (Global): 2048 MB GDDR5.
- Ancho de la Interfaz de la Memoria: 192 bits.
- Ancho de Banda de la Memoria: 144.2 GB/s.
- Bus Soportado: PCI-Express® 3.0 x 16.
- Versión de OpenGL® soportado: 4.3.
- Otras Tecnologías Soportadas: DirectX® 11, 3D Vision®, 3D Vision Surround®, PhysX™, CUDA™, SLI®, TXAA, FXAA, Adaptive VSync, GPU Boost.
- Longitud: 4.376 pulgadas (111 mm) x Altura: 9.5 pulgadas (241.3 mm).
- Anchura: doble ranura.
- Temperatura Máxima de la GPU: 97 °C.
- Potencia mínima requerida de la tarjeta gráfica: 450 W.

Fuente: [W22]

Apéndice C. Tablas.

cantidad	símbolo	unidades de *lattice*	equivalencia en unidades MKS
unidad básica de longitud	Δx	1 *ul*	$2/3$ μm
paso de tiempo	Δt	1 *pt*	0.25×10^{-6} *s*
unidad básica de masa	Δm	1 *um*	3.052×10^{-16} *kg*
Ancho de la vena	*LX – 1*	60 *ul*	40 μm
Largo de la vena	*LY*	2000 *ul*	1333.333 μm
densidad inicial	ρ_0	1 *um*/*uv*	1.03×10^3 *kg*/m^3 (Tomado de [1])
tiempo de relajación	τ	3.7767, 5.415, 7.0534	
viscosidad dinámica del fluido	μ	1.092, 1.638, 2.184 *um*/(*ul*·*pt*)	0.002, 0.003, 0.004 *Pa*·*s*
velocidad máxima del fluido en el centro de la vena	*U*	−0.01125 *ul*/*pt*	−0.03 *m*/*s*
semieje mayor de la partícula	*a*	6.3 *ul*	4.2 μm
semieje menor de la partícula	*b*	1.8 *ul*	1.2 μm
masa de la partícula	*M*	319.01223 *um*	97.35781×10^{-15} *kg*
fuerza externa (dirección eje *y*)	*F*	-2.73058×10^{-5}, -4.09587×10^{-5}, -5.46117×10^{-5} *um*·*ul*/pt^2	-8.8888×10^{-14}, -1.3333×10^{-13}, -1.7778×10^{-13} *N*
posición inicial de la partícula	$\mathbf{x_0}$	(10 *ul*, 1996 *ul*)	

Tabla 3. Parámetros de entrada para la Prueba 2.

Tpo. (s)	Posición X (m)	Posición Y (m)	Ang. (rad)	Vel. X $\left(m/s\right)$	Vel. Y $\left(m/s\right)$	Fuerza X (N)	Fuerza Y (N)	Vel. ang. $\left(rad/s\right)$	Torca ($N{\cdot}m$)
0.000	0.00E+00	0.00E+00	0.000	0.00E+00	0.0000	0.00E+00	0.00E+00	0.000	0.00E+00
0.001	1.42E-05	1.31E-03	-0.664	-3.55E-05	-0.0266	-2.08E-13	-2.25E-13	-546.761	9.63E-12
0.002	1.42E-05	1.28E-03	-1.050	-2.65E-05	-0.0266	8.98E-13	-1.53E-13	-265.472	2.55E-10
0.003	1.41E-05	1.25E-03	-1.240	-5.16E-06	-0.0266	-2.13E-13	1.03E-13	-140.670	-1.99E-13
0.004	1.41E-05	1.23E-03	-1.362	-6.67E-06	-0.0268	2.74E-13	-2.58E-14	-86.768	-1.62E-10
0.005	1.41E-05	1.20E-03	-1.442	8.07E-07	-0.0268	-1.23E-14	4.91E-14	-78.739	4.47E-11
0.006	1.41E-05	1.17E-03	-1.513	1.93E-06	-0.0267	2.85E-14	-1.20E-13	-67.863	-1.84E-11
0.007	1.41E-05	1.15E-03	1.566	5.74E-06	-0.0267	-5.72E-14	3.91E-13	-63.376	-7.51E-11
0.008	1.41E-05	1.12E-03	1.502	8.82E-06	-0.0267	-5.20E-14	2.80E-13	-71.607	-4.22E-11
0.009	1.41E-05	1.09E-03	1.429	1.31E-05	-0.0268	7.05E-14	5.57E-15	-79.661	-4.79E-11
0.010	1.42E-05	1.07E-03	1.345	1.55E-05	-0.0268	-1.52E-13	1.56E-14	-101.242	-3.45E-11
0.011	1.42E-05	1.04E-03	1.218	3.71E-05	-0.0267	6.42E-14	7.04E-14	-160.424	2.80E-11
0.012	1.42E-05	1.01E-03	1.010	3.60E-05	-0.0266	-1.31E-13	-2.26E-13	-283.137	-1.30E-10
0.013	1.43E-05	9.86E-04	0.588	1.01E-04	-0.0266	3.88E-13	4.33E-14	-582.332	4.18E-12
0.014	1.43E-05	9.60E-04	-0.179	1.07E-05	-0.0266	-3.53E-13	-2.93E-14	-812.993	5.10E-11
0.015	1.43E-05	9.33E-04	-0.813	-4.27E-05	-0.0267	-3.18E-13	-5.36E-13	-416.598	3.22E-10
0.016	1.43E-05	9.06E-04	-1.112	-1.84E-05	-0.0268	-1.21E-13	2.71E-14	-196.142	4.65E-11
0.017	1.43E-05	8.80E-04	-1.278	3.42E-06	-0.0267	-1.96E-13	3.09E-13	-127.520	-3.73E-11
0.018	1.42E-05	8.53E-04	-1.375	-6.50E-06	-0.0269	-6.86E-14	2.08E-14	-87.984	-8.71E-14
0.019	1.42E-05	8.26E-04	-1.458	-4.45E-07	-0.0268	-9.77E-14	1.36E-13	-82.958	-3.50E-11
0.020	1.42E-05	8.00E-04	-1.532	4.81E-06	-0.0268	4.72E-14	-6.85E-15	-72.080	-3.17E-11
0.021	1.42E-05	7.73E-04	1.538	9.85E-06	-0.0268	-1.29E-13	2.36E-13	-76.206	-2.77E-11
0.022	1.43E-05	7.46E-04	1.464	1.03E-05	-0.0268	5.83E-14	2.03E-13	-80.595	-8.68E-11
0.023	1.43E-05	7.20E-04	1.381	1.87E-05	-0.0269	-3.98E-14	4.60E-14	-88.607	8.55E-13
0.024	1.43E-05	6.93E-04	1.290	1.27E-05	-0.0268	3.52E-14	-2.93E-13	-115.056	-6.40E-11
0.025	1.43E-05	6.66E-04	1.130	3.66E-05	-0.0268	4.48E-14	1.09E-13	-190.123	-2.66E-11
0.026	1.44E-05	6.40E-04	0.857	7.20E-05	-0.0268	-2.26E-14	1.77E-13	-364.020	1.06E-10
0.027	1.44E-05	6.13E-04	0.288	5.32E-05	-0.0267	5.85E-13	-1.44E-13	-772.827	-3.11E-10
0.028	1.44E-05	5.86E-04	-0.494	-4.27E-05	-0.0268	1.25E-13	-4.36E-14	-646.293	-1.30E-10
0.029	1.44E-05	5.60E-04	-0.956	-3.00E-05	-0.0268	-9.49E-14	-1.68E-13	-328.571	-6.95E-11
0.030	1.44E-05	5.33E-04	-1.182	-1.72E-05	-0.0269	5.78E-14	-4.11E-14	-176.104	1.67E-10
0.031	1.44E-05	5.06E-04	-1.314	-1.08E-05	-0.0269	-1.40E-13	1.17E-13	-94.000	1.00E-12
0.032	1.44E-05	4.80E-04	-1.404	8.98E-07	-0.0269	-2.43E-14	7.01E-14	-87.512	-8.93E-12
0.033	1.44E-05	4.53E-04	-1.484	2.61E-06	-0.0269	1.25E-14	1.13E-13	-80.748	-4.72E-11
0.034	1.44E-05	4.26E-04	-1.557	9.20E-06	-0.0269	2.36E-14	5.79E-13	-77.097	3.36E-11
0.035	1.44E-05	4.00E-04	1.512	5.30E-06	-0.0269	-5.05E-14	-1.57E-13	-75.371	-1.80E-11
0.036	1.44E-05	3.73E-04	1.435	1.33E-05	-0.0269	-2.22E-14	7.17E-15	-86.178	-2.76E-11
0.037	1.44E-05	3.46E-04	1.348	1.40E-05	-0.0269	-7.61E-14	-4.49E-14	-87.857	1.13E-11
0.038	1.44E-05	3.20E-04	1.245	2.77E-05	-0.0269	-1.32E-13	9.34E-14	-148.288	-7.42E-12
0.039	1.44E-05	2.93E-04	1.063	3.79E-05	-0.0270	8.17E-13	5.74E-13	-228.466	-7.23E-12
0.040	1.45E-05	2.66E-04	0.724	6.43E-05	-0.0268	-1.32E-14	-5.78E-14	-502.875	-2.51E-10
0.041	1.46E-05	2.40E-04	0.059	4.66E-05	-0.0268	-1.19E-13	-2.54E-14	-798.528	7.57E-11
0.042	1.45E-05	2.13E-04	-0.649	-4.37E-05	-0.0268	-4.27E-13	1.80E-13	-547.368	-1.29E-10
0.043	1.45E-05	1.86E-04	-1.024	-4.46E-05	-0.0270	4.36E-14	2.28E-13	-254.810	-2.37E-10
0.044	1.45E-05	1.60E-04	-1.220	9.72E-07	-0.0270	1.28E-13	-1.03E-13	-151.304	7.65E-11
0.045	1.45E-05	1.33E-04	-1.336	-1.63E-06	-0.0270	-1.50E-13	3.41E-13	-94.304	-1.66E-11
0.046	1.45E-05	1.06E-04	-1.422	1.79E-06	-0.0270	1.68E-15	-4.79E-14	-84.376	-2.85E-11
0.047	1.45E-05	7.97E-05	-1.498	5.10E-06	-0.0270	-7.29E-14	7.66E-14	-74.720	-1.45E-11
0.048	1.45E-05	5.30E-05	-1.566	6.43E-06	-0.0270	-4.41E-14	1.08E-13	-67.102	-1.02E-12
0.049	1.45E-05	2.63E-05	1.508	6.62E-06	-0.0270	-2.50E-14	-1.87E-13	-69.926	-5.94E-11

Tabla 4. Muestra de 50 conjuntos de datos extraídos de los resultados obtenidos para la Prueba 1.

cantidad	símbolo	unidades de *lattice*	equivalencia en unidades MKS
unidad básica de longitud	Δx	1 ul	1.4 μm
paso de tiempo	Δt	1 pt	1 x 10^{-6} s
unidad básica de masa	Δm	1 um	2.82632 x 10^{-15} kg
Ancho de la vena	$LX - 1$	29 ul	40 μm
Largo de la vena	LY	4000 ul	5600 μm
densidad inicial	ρ_0	1 um/uv	1.03 x 10^3 kg/m^3
tiempo de relajación	τ	3.4720626	
viscosidad dinámica del fluido	μ	0.990688 $um/(ul \cdot pt)$	0.002 $Pa \cdot s$
velocidad máxima del fluido en el centro de la vena	U	-0.0214 ul/pt	-0.03 m/s
semieje mayor de la partícula	a	3 ul	4.2 μm
semieje menor de la partícula	b	0.857 ul	1.2 μm
masa de la partícula	M	34.4468 um	97.3578 x 10^{-15} kg
fuerza externa (dirección eje y)	F	-2.019407 x 10^{-4} $um \cdot ul/pt^2$	-7.99 x 10^{-13} N
posición inicial de la partícula	$\mathbf{x_0}$	(4.7619 ul, 3998 ul)	

Tabla 5. Parámetros de entrada para la Prueba 3.

Bibliografía.

[1] F. Janosheck, F. Toschi y J. Harting, Simplified Particulate Model for Coarse-Grained Hemodynamics Simulations, Physical Review E, 2010, Volumen 82, Artículo 56710, 11 Páginas.

[2] Simone Melchionna, A Model for Red Blood Cells in Simulations of Large-Scale Blood Flows, Macromolecular Theory and Simulations, 2011, Volumen 20, Número 7, Páginas 548-561.

[3] Krishnan B. Chandran, H. S. Udaykumar y Joseph M. Reinhardt, Image-Based Computational Modeling of the Human Circulatory and Pulmonary Systems, Methods and Applications, Springer, 2010, 465 Páginas.

[4] Sauro Succi, The Lattice Boltzmann Equation for Fluid Dynamics and Beyond, Oxford Science Publications, 2001, 288 Páginas.

[5] Dieter A. Wolf-Gladrow, Lattice-Gas Cellular Automata and Lattice Boltzmann Models: An Introduction, Springer, 1ª Ed. 2000, 308 Páginas.

[6] Cristiano Migliorini, YueHong Quian, Hudong Chen, Edward B. Brown, Rakesh K. Jain y Lance L. Munn, Red Blood Cells Augment Leukocyte Rolling in a Virtual Blood Vessel, Biophysical Journal, 2002, Volumen 83, Páginas 1834-1841.

[7] He Xiaochuan, Cui JianQiang, Wei Shoushui y Yang Ting, Lattice Boltzmann Method with Immersed Boundary Method for Simulating RBC Movement in Micro-vessel, 2010 8TH WORLD CONGRESS ON INTELLIGENT CONTROL AND AUTOMATION (WCICA), Institute of Electrical and Electronic Engineers, 2010, Páginas 1953-1958.

[8] Jian Guo Zhou, Lattice Boltzmann Methods for Shallow Water Flows, Springer, 2004, 136 Páginas.

[9] M. M. Dupin, I. Halliday y C. M. Care, A Lattice Boltzmann Model for Flow Blunting, Philosophical Transactions. Series A, Mathematical, Physical and Engineering Sciences, Royal Society, 2004, Volumen 362, Número 1821, Páginas 1755-1761.

[10] Anthony J. C. Ladd y R. Verberg, Lattice-Boltzmann Simulations of Particle-Fluid Suspensions, Journal of Statistical Physics, 2001, Volumen 104, Números 5-6, Páginas 1191-1251.

[11] Thanh N. Phung, John F. Brady y Georges Bossis, Stokesian Dynamics Simulation of Brownian Suspensions, Journal of Fluid Mechanics, 1996, Volumen 313, Páginas 181-207.

[12] Qingming Chang, J. Iwan y D. Alexander, Application of Lattice Boltzmann Method, Thermal Multiphase Fluid Dynamics, Verlag Dr. Müller, 2008, 208 Páginas.

[13] A. A. Mohamad, Lattice Boltzmann Method Fundamentals and Engineering Applications with Computer Codes, Springer, 2011, 178 Páginas.

[14] Erick Iván Román Román, Estudio de Plumas Térmicas Usando la Técnica de la Ecuación de Boltzmann en Redes, Tesis de Maestría en Ingeniería, Universidad Nacional Autónoma de México, 2007, 71 Páginas.

[15] Sigvat Stensholt, Lattice Boltzmann Methods for Use in Irregular Channels, Theoretical Foundations, Simulation, and Working Code, VDM Verlag Dr. Müller, 2009, 68 Páginas.

[16] Robert A. Granger, Fluid Mechanics, Dover Publications, 1995, 896 Páginas.

[17] Steven C. Chapra y Raymond P. Canale, Métodos Numéricos para Ingenieros, Mc Graw Hill, Traducción de la 5ª edición en inglés 2007, 977 Páginas.

[18] José Luis Velázquez Ortega, Simulación de Fluidos no Newtonianos en Presencia de Medios Porosos, Tesis Doctoral en Ciencias, Universidad Nacional Autónoma de México, 2009, 119 Páginas.

[19] Xiaoyi He y Li-Shi Luo, Theory of the Lattice Boltzmann Method: From the Boltzmann Equation to the Lattice Boltzmann Equation, Physical Review E, 1997, Volumen 56, Número 6, Páginas 6811-6817.

[20] Gene H. Golub y John H. Welsch, Calculation of Gauss Quadrature Rules, Mathematics of Computation, American Mathematical Society, 1969, Volumen 23, Número 106, Páginas 221-230.

[21] Michael C. Sukop y Daniel T. Thorne, Jr., Lattice Boltzmann Modeling, An Introduction for Geoscientists and Engineers, Springer, 1ª Ed. 2006, 172 Páginas.

[22] Carlo Cercignani, The Boltzmann Equation and Its Aplicattions, Applied Mathematical Sciences, Springer-Verlag, 1988, Volumen 67, 457 Páginas.

[23] R. S. Johnson, Singular Perturbation Theory: Mathematical and Analytical Techniques with Application to Engineering, Springer Science+Business Media, Inc., 2005, 300 Páginas.

[24] Bhimsen K. Shivamoggi, Perturbation Methods for Differential Equations, Birkhäuser Verlag, 2003, 354 Páginas.

[25] J. Kevorkian y J. D. Cole, Multiple Scale and Singular Perturbation Methods, Applied Mathematical Sciences, Springer Verlag, 1996, 632 Páginas.

[26] Ali H. Nayfeh, Perturbation Methods, Pure and Applied Mathematics, A Wiley–Intershence Series of Texts, Monographs & Tracts, John Wiley & Sons, 1973, 440 Páginas.

[27] Amir Nejat y Vahid Abdollahi, A Critical Study of the Compressible Lattice Boltzmann Methods for Riemann Problem, Journal of Scientific Computing, Springer, 2012, 20 Páginas.

[28] J. Y. Shao, C. Shu y Y. T. Chew, Development of an Immersed Boundary-Phase Field-Lattice Boltzmann Method for Neumann Boundary Condition to study Contact Line Dynamics, Journal of Computational Physics, Elsevier, 2013, 25 Páginas.

[29] E. Riegel, T. Indiger y N. A. Adams, Implementation of a Lattice-Boltzmann Method for Numerical Fluid Mechanics using the nVIDIA® CUDA Technology, Journal of Computational Physics, Elsevier, 2013, 25 Páginas.

[30] David B. Kirk y Wen-mei W. Hwu, Programming Massively Parallel Processors, A Hands-on Approach, Morgan Kaufmann, 1ª Ed. 2010, 258 Páginas.

[31] Behrooz Parhami, Introduction to Parallel Processing, Algorithms and Architectures, Kluwer Academic Publishers, 1ª Ed. 1999, 532 Páginas.

[32] NVIDIA® Corporation, NVIDIA CUDA C Programming Guide Version 4.2, 2012, 160 Páginas.

[33] Robert J. Baron y Lee Higbie, Computer Architecture, Addison Wesley, 1992, 560 Páginas.

[34] David E. Culler, Jaswinder Pal Singh y Anoop Gupta, Parallel Computer Architecture A Hardware/Software Approach, Morgan Kauffman, 1ª Ed. 1998, 1056 Páginas.

[35] Message Passing Interface Forum, MPI: A Message-Passing Interface Standard Version 2.2, University of Tennessee, 2009, 647 Páginas.

[36] Peter S. Pacheco, Parallel Programming with MPI, Morgan Kauffman, 1ª Ed. 1996, 418 Páginas.

[37] Michael J. Quinn, Parallel Programming in C with MPI and OpenMP, Mc Graw Hill Education, 1ª Ed. 2003, 529 Páginas.

[38] Rohit Chandra, Ramesh Menon, Leo Dagum, David Khor, Dror Maydan y Jeff McDonald, Parallel Programming in OpenMP, Morgan Kauffman, 1ª Ed. 2000, 231 Páginas.

[39] Jie Wu, Distributed System Design, CRC Press, 1999, 482 Páginas.

[40] E-Jiang Ding y Cyrus K. Aidun, Extension of the Lattice-Boltzmann Method for Direct Simulation of Suspended Particles Near Contact, Journal of Statistical Physics, 2003, Volumen 112, Números 3-4, Páginas 685-708.

[41] Cyrus K. Aidun y Yannan Lu, Lattice Boltzmann Simulation of Solid Particles Suspended in Fluid, Journal of Statistical Physics, 1995, Volumen 81, Números 1 y 2, Páginas 49-61.

[42] Cyrus K. Aidun, Yannan Lu y E.-Jiang Ding, Direct Analysis of Particulate Suspensions with Inertia Using the Discrete Boltzmann Equation, Journal of Fluid Mechanics, 1998, Volumen 373, Páginas 287-311.

[43] Lindsay M. Crowl y Aaron L. Fogelson, Computational Model of Whole Blood Exhibiting Lateral Platelet Motion Induced by Red Blood Cells, International Journal for Numerical Methods in Biomedical Engineering, 2010, Volumen 26, Páginas 471-487.

[44] Junfeng Zhang, Paul C. Johnson y Aleksander S. Popel, Effects of Erythrocite Deformability and Aggregation on the Cell Free Layer and Apparent Viscosity of Microscopic Blood Flows, Microvascular Research, 2009, Volumen 77, Páginas 265-272.

[45] Salvatore P. Sutera y Richard Skalak, The History of Poiseuille's Law, Annual Review of Fluids Mechanics, 1993, Volumen 5, Páginas 1-19.

[46] Choeng-Ryul Choi, Chang-Nyung Kim y Tae-Hyub Hong, Blood Cell Dynamics in a Simple Shear Flow using an Implicit Fluid-Structure Interaction Method Based on the ALE Approach, 2010, Volumen 6, Páginas 228-233.

[47] N. Q. Nguyen y Anthony J. C. Ladd, Lubrication Corrections for Lattice-Boltzmann Simulation of Particle Suspensions, Physical Review E, 2002, Volumen 66, Artículo 46708, 12 Páginas.

[48] Anthony J. C. Ladd, Numerical Simulations of Particulate Suspensions via a Discretized Boltzmann Equation. Part 1. Theoretical Foundation, Journal of Fluid Mechanics, Cambridge University Press, 1994, Volumen 271, Páginas 285-309.

[49] Prosenjit Bagchi, Mesoscale Simulation of Blood Flow in Small Vessels, Biophysical Journal, 2007, Volumen 92, Páginas 1858-1877.

[50] Junfeng Zhang, Effect of Suspending Viscosity on Red Blood Cell Dynamics and Blood Flows in Microvessels, Microcirculation, 2011, Volumen 18, Páginas 562-573.

[51] Wayne E. Wentworth y S. Jules Ladner, Fundamentos de Química Física, Reverté, 1975, 645 Páginas.

[52] Robert Resnick, David Halliday y Kenneth S. Krane, Física Volumen 1, Compañía Editorial Continental, 4a Edición en Español 2002, 624 Páginas.

[53] Smith-Van Ness, Introducción a la Termodinámica en Ingeniería Química, Mc Graw Hill, 1989, 718 Páginas.

[54] Rolf Verberg, Alexander Alexeev y Ana C. Balazs, Modeling the Release of Nanoparticles from Mobile Microcapsules, The Journal of Chemical Physics, 2006, Volumen 125, Número 22, Artículo 224712, 10 Páginas.

[55] T. Krüger, F. Varnik y D. Raabe, Efficient and Accurate Simulations of Deformable Particles Immersed in a Fluid Using a Combined Immersed Boundary Lattice Boltzmann Finite Element Method, Computers and Mathematics with Applications, 2011, Volumen 61, Páginas 3485-3505.

[56] John Nickolls, Ian Buck, Michael Garland y Kevin Skadron, Scalable Parallel Programming, ACM QUEUE, 2008, 53 Páginas.

[57] Andrzej Mysliwski y Anna Korczak, Increase of Size and Dry Mass of Human Erythrocites Depending on Age of Donors, Mechanisms of Ageing and Development, 1986, Volumen 34, Páginas 111-115.

[58] Wenjuan Xiong y Junfeng Zhang, Shear Stress Variation Induced by Red Blood Cell Motion in Microvessel, Annals of Biomedical Engineering, 2010, Volumen 38, Número 8, Páginas 2649-2659.

[59] H. T. Low, Y. Sui, Y. T. Chew y P. Roy, The Transient Deformation of Red Blood Cells in Shear Flow, Modern Physics Letters B, 2009, Volumen 23, Número 3, Páginas 545-548.

Páginas Web.

[W1] **http://www.medterms.com/script/main/art.asp?articlekey=4034**, Fecha de consulta: 12 de Abril de 2013.

[W2] **http://www.merriam-webster.com/dictionary/in%20vitro**, Fecha de consulta: 12 de Abril de 2013.

[W3] **https://en.wikipedia.org/wiki/In_vitro**, Fecha de consulta: 12 de Abril de 2013.

[W4] **http://ocw.uv.es/ciencias/2/1-2/112733mats70.pdf**, Fecha de consulta: 30 de Agosto de 2012.

[W5] **http://www.dpye.iimas.unam.mx/eduardo/MCB/node14.html**, Fecha de consulta: 18 de Marzo de 2013.

[W6] **http://www.cims.nyu.edu/~eve2/reg_pert.pdf**, Fecha de consulta: 5 de Julio de 2012.

[W7] **http://www.mv4t.com/Hardware_Graphics_CUDA-Capable-Cards.php**, Fecha de consulta: 12 de abril de 2012.

[W8] **https://computing.llnl.gov/tutorials/mpi/**, Fecha de consulta: 19 de Junio de 2012.

[W9] **http://www.mcs.anl.gov/research/projects/mpich2/documentation/index.php?s=docs**, Fecha de consulta: 5 de Septiembre de 2012.

[W10] **http://www.mpi-forum.org/**, Fecha de consulta: 18 de Marzo de 2013.

[W11] **http://www.mcs.anl.gov/research/projects/mpi/**, Fecha de consulta: 21 de Junio de 2012.

[W12] **http://www.open-mpi.org/**, Fecha de consulta: 21 de Junio de 2012.

[W13] **http://www.csm.ornl.gov/pvm/**, Fecha de consulta: 18 de Abril de 2013.

[W14] **http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html**, Fecha de consulta: 12 de Marzo de 2013.

[W15] **http://docs.nvidia.com/cuda/cuda-c-best-practices-guide/index.html**, Fecha de consulta: 19 de Marzo de 2013.

[W16] **http://www.unioviedo.es/Areas/Mecanica.Fluidos/docencia/**

_asignaturas/mecanica_de_fluidos/07_08/tema%207%20%20ESTATICA%20FLUIDOS.pdf, Fecha de consulta: 20 de Febrero de 2013.

[W17] http://www.efunda.com/math/solids/solids_display.cfm?SolidName=EllipticalCylinder, Fecha de consulta: 8 de mayo de 2012.

[W18] http://www.icp.uni-stuttgart.de/~icp/mediawiki/images/2/26/Roehm_diplomathesis.pdf, Fecha de consulta: 13 de Septiembre de 2012.

[W19] https://www.cs.rochester.edu/~cding/Announcements/HIPS07/openmp.pdf, Fecha de consulta: 17 de Noviembre de 2012.

[W20] http://www.intel.com/content/dam/www/public/us/en/documents/technical-specifications/software-developers-hpet-spec-1-0a.pdf, Fecha de consulta: 27 de Abril de 2013.

[W21] http://www.stephenwolfram.com/publications/articles/ca/84-universality/index.html, Fecha de consulta: 24 de Abril de 2013.

[W22] Instituto de Tecnología de Massachusetts, Notas del Curso Abierto: Mecánica Estadística de Partículas, http://ocw.mit.edu/courses/physics/8-333-statistical-mechanics-i-statistical-mechanics-of-particles-fall-2007/download-course-materials/, Fecha de consulta: 19 de Marzo de 2013.

[W23] http://www.nvidia.com/page/home.html, Fecha de consulta: 11 de Marzo de 2013.

[W24] Instituto de Tecnología de Ilinois, Las Ecuaciones de Navier-Stokes, http://www.iit.edu/arc/workshops/pdfs/Navier_Stokes.pdf, Fecha de consulta: 18 de Marzo de 2013.

[W25] http://en.wikipedia.org/wiki/Ludwig_Boltzmann, Fecha de consulta: 25 de abril de 2012.

[W26] Universidad de Virginia, Transformaciones Galileanas, http://galileo.phys.virginia.edu/classes/252/lecture1.htm, Fecha de consulta: 9 de mayo de 2012.

[W27] Universidad de Nuevo México, Invarianza Galileana, http://www.math.unm.edu/mctp.BAK/mctp_archive/lecture_notes/Nakamaye/physics2.pdf, Fecha de consulta: 14 de abril de 2013.

[W28] Universidad Estatal de Mississippi, Notas de MPI, http://www.cse.msstate.edu/~ioana/Courses/CS6163/mpi/index.htm, Fecha de consulta: 19 de Junio de 2012.

[W29] Universidad de Alcalá, Expansiones en Series de Taylor de Derivadas Parciales de Orden Superior,

http://www2.uah.es/fsegundo/calcTeleco/esquemas/220-DerivadasParcialesSegundasPolinomiosTaylor.pdf, Fecha de consulta: 12 de Julio de 2012.

[W30] Departamento de Comercio de los EEUU, Guía Para el Uso del Sistema Internacional de Unidades, NIST publicación especial 811, **http://physics.nist.gov/cuu/pdf/sp811.pdf**, Fecha de consulta: 2 de Septiembre de 2012.

[W31] Universidad Estatal de Georgia, Notas de Sistemas Distribuidos, **http://www.cs.gsu.edu/~cscskp/DistSystems/chap01-prasad.pdf**, Fecha de consulta: 8 de Octubre de 2012.

[W32] Leslie Lamport y Nancy Lynch, Microsoft Research, Cómputo Distribuido, **http://research.microsoft.com/en-us/um/people/lamport/pubs/lamport-chapter.pdf**, Fecha de consulta: 8 de Octubre de 2012.

[W33] Timothy John Thompson, Entropía y Segunda Ley de la Termodinámica en Sistemas Abiertos, **http://www.tim-thompson.com/entropy3.html**, Fecha de consulta: 27 de Octubre de 2012.

[W34] Rajesh R. Parwani, Universidad Nacional de Singapore, Entropía para Sistemas Abiertos, **http://staff.science.nus.edu.sg/~parwani/c1/node49.html**, Fecha de consulta: 27 de Octubre de 2012.

[W35] Universidad de la Habana, Notas de Tensores, **http://www.fisica.uh.cu/fisteo/resources/asignaturas/AlgebraII/conf2.pdf**, Fecha de consulta: 21 de Enero de 2013.

[W36] Santiago Muelas, Universidad Politécnica de Madrid, Elasticidad Lineal: Fuerzas de Masa, Fuerzas de Superficie y Tensiones **http://w3.mecanica.upm.es/~smuelas/elasticidad/node3.html**, Fecha de consulta: 25 de Enero de 2013.

[W37] Silicon Madness, Nvidia Announces Tesla 20 Series, **http://siliconmadness.blogspot.mx/2009/11/nvidia-announces-tesla-20-series.html**, Fecha de consulta: 11 de Marzo de 2013.

[W38] Scott Wasson, The Tech Report, Nvidia's GeForce® GTX 480 y 470 Graphic Processors, **http://techreport.com/review/18682/nvidia-geforce-gtx-480-and-470-graphics-processors**, Fecha de consulta: 11 de Marzo de 2013.

[W39] Nvidia, Especificaciones del GeForce® GTX 480, **http://www.geforce.com/hardware/desktop-gpus/geforce-gtx-480/specifications**, Fecha de consulta: 11 de Marzo de 2013.

[W40] Andrea Puglisi, Universidad de Roma La Sapienza, **http://denali.phys.uniroma1.it/~puglisi/thesis/node24.html**, Fecha de consulta: 29 de Abril de 2013.